Neural Comput & Applic (2018) 30:3799–3814
DOI 10.1007/s00521-017-2966-z

ORIGINAL ARTICLE

Asymmetric ν-twin support vector regression

Yitian Xu[1] · Xiaoyan Li[2] · Xianli Pan[1] · Zhiji Yang[1]

Received: 28 April 2015 / Accepted: 24 March 2017 / Published online: 27 April 2017

Abstract Twin support vector regression (TSVR) aims at finding ϵ-insensitive up- and down-bound functions for the training points by solving a pair of smaller-sized quadratic programming problems (QPPs) rather than a single large one as in the conventional SVR. So TSVR works faster than SVR in theory. However, TSVR gives equal emphasis to the points above the up-bound and below the down-bound, which leads to the same influences on the regression function. In fact, points in different positions have different effects on the regressor. Inspired by it, we propose an asymmetric ν-twin support vector regression based on pinball loss function (Asy-ν-TSVR). The new algorithm can effectively control the fitting error by tuning the parameters ν and p. Therefore, it enhances the generalization ability. Moreover, we study the distribution of samples and give the upper bounds for the samples locating in different positions. Numerical experiments on one artificial dataset, eleven benchmark datasets and a real wheat dataset demonstrate the validity of our proposed algorithm.

Keywords Support vector regression · Twin support vector regression · Pinball loss · Asymmetric

✉ Yitian Xu
xytshuxue@cau.edu.cn

Xiaoyan Li
b20150365@xs.ustb.edu.cn

Xianli Pan
pxlcjs@cau.edu.cn

Zhiji Yang
yzhiji@cau.edu.cn

[1] College of Science, China Agricultural University, Beijing 100083, China

[2] Donlinks School of Economics and Management, University of Science and Technology Beijing, Beijing 100083, China

1 Introduction

The support vector machine (SVM), introduced by Vapnik, is a successful model and prediction tool for classification and regression, which has spread to many fields [1, 2]. Compared with other machine learning approaches like artificial neural networks, SVM has many advantages. First, SVM solves a quadratic programming problem (QPP), assuring that once an optimal solution is obtained, it is the unique solution. Second, by maximizing the margin between two classes of samples, SVM derives a sparse and robust solution. Third, SVM implements the structural risk minimization principle rather than the empirical risk minimization principle, which minimizes the upper bound of the generalization error. The introduction of kernel function extends the linear case to the nonlinear case, and effectively overcomes the "curse of dimension" [3, 4]. Because of its great generalization performance, SVM has been successfully applied in various aspects ranging from pattern recognition, text categorization, and financial regression.

The ν-support vector regression (ν-SVR) [5], which is based on statistical learning theory, has become a standard tool in regression tasks. The ν-SVR extends standard SVR [2] techniques by Vapnik via enforcing a fraction of the data samples to lie inside an ϵ-tube, as well as minimizing the width of this tube [6]. It introduces a new parameter ν to control the fitting error. ν-SVR has better generalization

performance than the traditional techniques. In the ν-SVR, the ϵ-insensitive loss function is defined as follows

$$L_\epsilon(u) = \begin{cases} u-\epsilon, & u \geq \epsilon, \\ 0, & -\epsilon < u < \epsilon, \\ -u-\epsilon, & u \leq -\epsilon. \end{cases} \tag{1}$$

ν-SVR owns better generalization ability compared with other machine learning methods. However, one of the main challenges for it is high computational complexity. In order to improve the computational speed, Peng proposed an efficient twin support vector machine for regression problem (TSVR) [7–9] based on twin support vector machine (TSVM) [11]. It aims at generating two nonparallel bound functions [10] by solving two smaller-sized QPPs such that each function determines the ϵ-insensitive up- or down-bounds of the unknown regressor. Each QPP involves only one group of constraints for all samples, which makes TSVR work faster than the standard SVR in theory [7, 12]. Then, it receives many attentions, and many variants have also been proposed in literatures [13–16].

Recently, Huang extends ϵ-insensitive loss $L_\epsilon(u)$ to the pinball loss $L_\epsilon^p(u)$. Samples locating in different positions are given different penalties [17–20], and then it yields better generalization performance. Inspired by it, we propose an asymmetric ν-twin support vector regression (Asy-ν-TSVR) in this paper. Where the asymmetric tube is used, and then Asy-ν-TSVR produces good generalization performance. Asy-ν-TSVR is especially suitable for dealing with the asymmetric noise [21–24].

Asy-ν-TSVR aims at finding two nonparallel functions: ϵ_1-insensitive down function $f_1(x) = w_1^T x + b_1$ and ϵ_2-insensitive up function $f_2(x) = w_2^T x + b_2$ [25, 26]. Similar to the TSVR [7], Asy-ν-TSVR also solves two smaller-sized QPPs rather than a larger one, and each involves only one group of constraints for all samples. By introducing the pinball loss [27] into it, the samples lying above and below the bounds are given different punishments [29, 30]. To verify the validity of our proposed algorithm, an artificial experiment, eleven benchmark experiments and a real wheat dataset have been performed. Compared with ν-SVR, Asy-ν-SVR, least squares for support vector regression (LS-SVR) [28], and TSVR, our proposed Asy-ν-TSVR has better generalization ability.

The paper is organized as follows: Section 2 briefly dwells on ν-SVR, Asy-ν-SVR, and TSVR. Asy-ν-TSVR is proposed in Section 3, which includes both the linear and nonlinear cases. The bounds are discussed in Section 4. Section 5 performs experiments on three kinds of datasets to investigate the feasibility and validity of our proposed algorithm. Section 6 ends the paper with concluding remarks.

2 Related works

In this section, we give a brief description of the ν-SVR, Asy-ν-SVR, and TSVR. Given a training set $T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_l, y_l)\}$, where $x_i \in R^d$ and $y_i \in R$. For the sake of conciseness, let matrix $A = (x_1, x_2, \cdots, x_l)^T$, and matrix $Y = (y_1, y_2, \cdots, y_l)^T$. e is a vector of ones of appropriate dimensions.

2.1 ν-support vector regression

The nonlinear ν-SVR seeks to find a regression function $f(x) = w^T\phi(x)+b$ in a high-dimensional feature space tolerating the small error in fitting the given data points. This can be achieved by utilizing the ϵ-insensitive loss function $L_\epsilon(u)$ that sets an ϵ-insensitive "tube" as small as possible, within which errors are discarded. The ν-SVR can be obtained by solving the following QPP,

$$\begin{aligned} \min_{w,b,\xi,\xi^*,\epsilon} \quad & \frac{1}{2}\|w\|^2 + C\nu\epsilon + \frac{C}{l}\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ \text{s.t.} \quad & (w^T\phi(x_i) + b) - y_i \leq \epsilon + \xi_i, i = 1, 2, \cdots, l, \\ & y_i - (w^T\phi(x_i) + b) \leq \epsilon + \xi_i^*, i = 1, 2, \cdots, l, \\ & \epsilon \geq 0, \xi_i \geq 0, \xi_i^* \geq 0, i = 1, 2, \cdots, l. \end{aligned} \tag{2}$$

where C, ν are parameters chosen a priori. Parameter C controls the trade-off between the fitting errors and flatness of the regression function. ν has its theoretical interpretation that controls the fractions of the support vectors and the margin errors. To be more precise, ν is an upper bound on the fraction of errors and a lower bound on the fraction of support vectors [31–33]. ξ_i and ξ_i^* are the slack vectors reflecting whether the samples locate into the ϵ-tube or not.

By introducing the Lagrangian multiplies α and α_i^*, we can derive the dual problem of the ν-SVR as follows

$$\begin{aligned} \min_{\alpha^{(*)}} \quad & \frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)K(x_i, x_j) - \sum_{i=1}^{l}(\alpha_i^* - \alpha_i)y_i \\ \text{s.t.} \quad & \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0, \\ & 0 \leq \alpha_i^{(*)} \leq C/l, i = 1, 2, \cdots, l, \\ & \sum_{i=1}^{l}(\alpha_i + \alpha_i^*) \leq C\nu. \end{aligned} \tag{3}$$

Once the QPP (3) is solved, we can achieve its solution $\alpha^{(*)} = (\alpha_1, \alpha_1^*, \alpha_2, \alpha_2^*, \cdots, \alpha_l, \alpha_l^*)$ and the threshold b, and then obtain the regression function,

$$f(x) = \sum_{i=1}^{l}(\alpha_i^* - \alpha_i)K(x_i, x) + b. \tag{4}$$

Here $K(x_i, x)$ represents a kernel function which gives the dot product in the high-dimensional feature space.

2.2 Asymmetric ν-support vector regression

The ν-SVR considers only one possible location of the ϵ-tube: it imposes that the numbers of samples above and below the tube are equal. To further improve the computational accuracy, Huang imposes that those outliers can be divided asymmetrically over both regions. To pursue an asymmetric tube, he introduces the following asymmetric loss function

$$L_\epsilon^p(u) = \begin{cases} \frac{1}{2p}(u-\epsilon), & u \geq \epsilon, \\ 0, & -\epsilon < u < \epsilon, \\ \frac{1}{2(1-p)}(-u-\epsilon), & u \leq -\epsilon, \end{cases} \tag{5}$$

where p is a parameter related to asymmetry. As we can learn that $L_\epsilon^p(u)$ can reduce to L_ϵ when $p = 0.5$.

The Asy-ν-SVR solves the following QPP,

$$\begin{aligned} \min_{w,b,\epsilon,\xi_i,\xi_i^*} \quad & \frac{1}{2\gamma} w^T w + \nu\epsilon + \frac{1}{l}\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ \text{s.t.} \quad & (w^T\phi(x_i) + b - \epsilon) - y_i \leq 2(1-p)\xi_i, i = 1, \cdots, l, \\ & y_i - (w^T\phi(x_i) + b + \epsilon) \leq 2p\xi_i^*, i = 1, 2, \cdots, l, \\ & \epsilon \geq 0, \xi_i \geq 0, \xi_i^* \geq 0, i = 1, 2, \cdots, l. \end{aligned} \tag{6}$$

The coefficients γ and ν control the trade-off among the margin, the size of the slack variables and the width of ϵ-tube. The ϵ as unknown controls the width of the ϵ-insensitive zone, which is used to fit the training data, and it can affect the number of support vectors. ξ_i and ξ_i^* are the slack vectors reflecting whether the samples locating into the ϵ-tube or not. Parameter p is related to the asymmetry. Where parameters γ, ν and p are chosen in advance. Apparently Asy-ν-SVR reduces to ν-SVR when $p = 0.5$. So Asy-ν-SVR is an extension of the ν-SVR.

We can derive the dual formulation of the Asy-ν-SVR as follows

$$\begin{aligned} \min_{\lambda^*,\lambda} \quad & \frac{1}{2}\sum_{i,j=1}^{l}(\lambda_i^* - \lambda_i)^T K(x_i, x_j)(\lambda_j^* - \lambda_j) - \sum_{i=1}^{l} y_i(\lambda_i^* - \lambda_i) \\ \text{s.t.} \quad & \sum_{i=1}^{l}(\lambda_i^* - \lambda_i) = 0, \\ & \sum_{i=1}^{l}(\lambda_i^* + \lambda_i) \leq \nu\gamma, \\ & 0 \leq \lambda_i^* \leq \frac{\gamma}{2pl}, i = 1, 2, \cdots, l, \\ & 0 \leq \lambda_i \leq \frac{\gamma}{2(1-p)l}, i = 1, 2, \cdots, l. \end{aligned} \tag{7}$$

Once the QPP (7) is solved, we can achieve its solution $\lambda^{(*)} = (\lambda_1, \lambda_1^*, \lambda_2, \lambda_2^*, \cdots, \lambda_l, \lambda_l^*)$ and threshold b, and then obtain the following regressor

$$f(x) = \sum_{i=1}^{l}(\lambda_i^* - \lambda_i)K(x, x_i) + b. \tag{8}$$

This extension gives an effective way to deal with skewed noise in regression problems.

2.3 Twin support vector regression

To improve the computational speed, Peng [7] proposed an efficient TSVR for the regression problem. TSVR generates an ϵ-insensitive down-bound function $f_1(x) = w_1^T x + b_1$ and an ϵ-insensitive up-bound function $f_2(x) = w_2^T x + b_2$. TSVR is illustrated in Fig. 1.

The final regressor $f(x)$ is decided by the mean of these two bound functions, i.e.,

$$f(x) = \frac{1}{2}(f_1(x) + f_2(x)) = \frac{1}{2}(w_1 + w_2)^T x + \frac{1}{2}(b_1 + b_2). \tag{9}$$

For the nonlinear case, TSVR solves the following pair of smaller-sized QPPs,

$$\begin{aligned} \min_{w_1,b_1,\xi} \quad & \frac{1}{2}\|Y - e\epsilon_1 - (K(A, A^T)w_1 + eb_1)\|^2 + C_1 e^T\xi \\ \text{s.t.} \quad & Y - (K(A, A^T)w_1 + eb_1) \geq e\epsilon_1 - \xi, \\ & \xi \geq 0, \end{aligned} \tag{10}$$

and

$$\begin{aligned} \min_{w_2,b_2,\eta} \quad & \frac{1}{2}\|Y + e\epsilon_2 - (K(A, A^T)w_2 + eb_2)\|^2 + C_2 e^T\eta \\ \text{s.t.} \quad & (K(A, A^T)w_2 + eb_2) - Y \geq e\epsilon_2 - \eta, \\ & \eta \geq 0. \end{aligned} \tag{11}$$

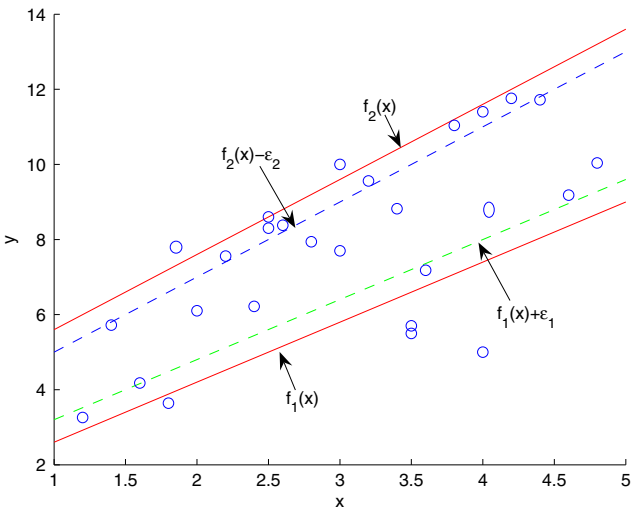

Fig. 1 Illustration of the TSVR

In the objective function of (10) or (11), the first term minimizes the squared distances from the training points to $f_1(x)+\epsilon_1$ and $f_2(x)-\epsilon_2$, the second term aims to minimize the sum of error variables. Where parameters C_1 and C_2 chosen in advance determines the trade-off between above goals. The constraints require the training points should be larger than the function $f_1(x)$ at least ϵ_1, at the same time they should be smaller than the function $f_2(x)$ at least ϵ_2. ξ and η are slack vectors. For the outliers, the same penalty is given to them in TSVR.

After introducing the Lagrangian function L, and differentiating L with respect to variables, we can derive their dual formulations of (10) and (11) as follows

$$\begin{aligned} \max_{\alpha} \quad & -\frac{1}{2}\alpha^T H(H^T H)^{-1}H^T\alpha + f^T H(H^T H)^{-1}H^T\alpha - f^T\alpha \qquad (12)\\ \text{s.t.} \quad & 0 \le \alpha \le C_1 e, \end{aligned}$$

and

$$\begin{aligned} \max_{\beta} \quad & -\frac{1}{2}\beta^T H(H^T H)^{-1}H^T\beta - h^T H(H^T H)^{-1}H^T\beta + h^T\beta \qquad (13)\\ \text{s.t.} \quad & 0 \le \beta \le C_2 e, \end{aligned}$$

where $H = [K(A, A^T)\ e]$, $f = Y - e\epsilon_1$, and $h = Y + e\epsilon_2$.

Once the dual QPPs (12) and (13) are solved, we can get

$$\begin{bmatrix} w_1 \\ b_1 \end{bmatrix} = (H^T H)^{-1}H^T(f-\alpha), \qquad (14)$$

and

$$\begin{bmatrix} w_2 \\ b_2 \end{bmatrix} = (H^T H)^{-1}H^T(h+\beta). \qquad (15)$$

Note that TSVR is comprised of a pair of QPPs such that each QPP determines one of up- or down-bound function by using only one group of constraints compared with the standard SVR. Hence, TSVR solves two smaller-sized QPPs rather than a single large one, which implies that TSVR works faster than the standard SVR in theory.

3 Asymmetric ν-twin support vector regression

As we know that the same penalties are given to the points above the up-bound and below the down-bound in TSVR. In fact, they have different effects on the regression function. Motivated by studies above, we propose the following asymmetric ν-twin support vector regression based on the pinball loss function.

3.1 Linear case

We extend TSVR to the asymmetric case, where p is the parameter related to asymmetric. Asy-ν-TSVR generates an ϵ_1-insensitive down-bound function $f_1(x) = w_1^T x + b_1$ and an ϵ_2-insensitive up-bound function $f_2(x) = w_2^T x + b_2$, and they are nonparallel. Asy-ν-TSVR is illustrated in Fig. 2.

The final regressor $f(x)$ is decided by the mean of these two bound functions, i.e.

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

In TSVR, after replacing the ϵ-insensitive loss function L_ϵ by the pinball loss $L_\epsilon^p(u)$, Asy-ν-TSVR solves the following pair of smaller-sized QPPs,

$$\begin{aligned} \min_{w_1,b_1,\epsilon_1,\xi} \quad & \frac{1}{2}\|Y-(Aw_1+eb_1)\|^2 + C_1\nu_1\epsilon_1 + \frac{1}{l}C_1 e^T\xi \qquad (16)\\ \text{s.t.} \quad & Y-(Aw_1+eb_1) \ge -e\epsilon_1 - 2(1-p)\xi,\\ & \epsilon_1 \ge 0, \xi \ge 0, \end{aligned}$$

and

$$\begin{aligned} \min_{w_2,b_2,\epsilon_2,\eta} \quad & \frac{1}{2}\|Y-(Aw_2+eb_2)\|^2 + C_2\nu_2\epsilon_2 + \frac{1}{l}C_2 e^T\eta \qquad (17)\\ \text{s.t.} \quad & (Aw_2+eb_2)-Y \ge -e\epsilon_2 - 2p\eta,\\ & \epsilon_2 \ge 0, \eta \ge 0 \end{aligned}$$

where C_1, C_2, ν_1, ν_2 are parameters chosen in advance, ξ and η are slack vectors.

The first term in the objective function of (16) or (17) minimizes the sum of squared distances from the estimated function $f_1(x) = w_1^T x + b_1$ or $f_2(x) = w_2^T x + b_2$ to the training points. The second term means the ϵ_1-tube and ϵ_2-tube are as narrow as possible. The third term minimizes the sum of error variables. The constraints require the training points lie above $f_1(x) - \epsilon_1 = w_1^T x + b_1 - \epsilon_1$ or below $f_2(x) + \epsilon_2 = w_2^T x + b_2 + \epsilon_2$ as possible. The slack vector ξ or η is introduced to measure the error wherever the distance is closer than ϵ_1 or ϵ_2. Note that the equal emphasis is given to ξ, η in TSVR. Here, for the outliers, we apply a slightly different penalty to them with the parameter p, i.e., pinball loss. Moreover, it degrades into ϵ-intensive loss when $p = 0.5$.

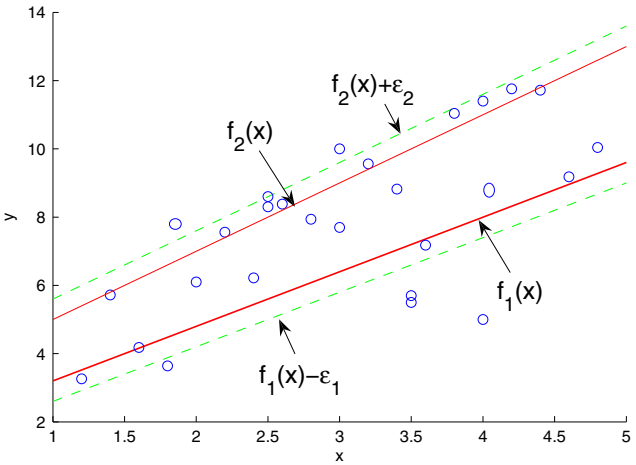

Fig. 2 Illustration of the Asy-ν-TSVR

To derive the dual formulations of Asy-ν-TSVR, we first introduce the following Lagrangian function for the problem (16), which is

$$L(w_1,b_1,\epsilon_1,\xi,\alpha,\beta,\gamma)=\frac{1}{2}\|Y-(Aw_1+eb_1)\|^2+C_1\nu_1\epsilon_1+\frac{1}{l}C_1e^T\xi$$
$$-\alpha^T(Y-(Aw_1+eb_1)+e\epsilon_1+2(1-p)\xi)-\beta\epsilon_1-\gamma^T\xi, \quad (18)$$

where α, β, and γ are nonnegative and the Lagrangian multipliers. After differentiating L in (18) with respect to variables w_1, b_1, ϵ_1 and ξ, we have

$$\frac{\partial L}{\partial w_1}=-A^T(Y-(Aw_1+eb_1))+A^T\alpha=0, \quad (19)$$

$$\frac{\partial L}{\partial b_1}=-e^T(Y-(Aw_1+eb_1))+e^T\alpha=0, \quad (20)$$

$$\frac{\partial L}{\partial \epsilon_1}=C_1\nu_1-e^T\alpha-\beta=0, \quad (21)$$

$$\frac{\partial L}{\partial \xi}=\frac{1}{l}C_1e-2(1-p)\alpha-\gamma=0. \quad (22)$$

Combining (19) and (20) leads to

$$-\begin{bmatrix}A^T\\e^T\end{bmatrix}\left((Y-e\epsilon_1)-[A\ e]\begin{bmatrix}w_1\\b_1\end{bmatrix}\right)+\begin{bmatrix}A^T\\e^T\end{bmatrix}\alpha=0. \quad (23)$$

Define $G=[A\ \ e]$, and $\mu_1=[w_1,b_1]^T$, then we have

$$-G^T(Y-G\mu_1)+G^T\alpha=0. \quad (24)$$

From (24), we can get

$$\mu_1=(G^TG)^{-1}(G^TY-G^T\alpha)=(G^TG)^{-1}G^T(Y-\alpha). \quad (25)$$

Then,

$$L=\frac{1}{2}\|Y-G\mu_1\|^2+C_1\nu_1\epsilon_1+\frac{1}{l}C_1e^T\xi-\alpha^T(Y-G\mu_1+e\epsilon_1+2(1-p)\xi)$$
$$-\beta\epsilon_1-\gamma^T\xi=\frac{1}{2}\|Y-G\mu_1\|^2-\alpha^T(Y-G\mu_1)$$
$$=\frac{1}{2}\|Y-G(G^TG)^{-1}G^T(Y-\alpha)\|^2-\alpha^T(Y-G(G^TG)^{-1}G^T(Y-\alpha)). \quad (26)$$

We can get

$$L=-\frac{1}{2}\alpha^TG(G^TG)^{-1}G^T\alpha+Y^TG(G^TG)^{-1}G^T\alpha-Y^T\alpha. \quad (27)$$

From (21) and (22), we can obtain the following constraints,

$$e^T\alpha\le C_1\nu_1,\quad 0\le\alpha\le C_1e/2(1-p)l. \quad (28)$$

Finally, we can derive the dual formulation of (16) as follows

$$\max_{\alpha}\ -\frac{1}{2}\alpha^TG(G^TG)^{-1}G^T\alpha+Y^TG(G^TG)^{-1}G^T\alpha-Y^T\alpha \quad (29)$$
$$\text{s.t.}\quad e^T\alpha\le C_1\nu_1,$$
$$0\le\alpha\le C_1e/2(1-p)l.$$

Similary, we can obtain the dual formulation of (17) as

$$\max_{\gamma}\ -\frac{1}{2}\gamma^TG(G^TG)^{-1}G^T\gamma-Y^TG(G^TG)^{-1}G^T\gamma+Y^T\gamma \quad (30)$$
$$\text{s.t.}\quad e^T\gamma\le C_2\nu_2,$$
$$0\le\gamma\le C_2e/2pl.$$

Once (30) is solved, we can obtain

$$\begin{bmatrix}w_2\\b_2\end{bmatrix}=(G^TG)^{-1}G^T(Y+\gamma). \quad (31)$$

3.2 Nonlinear case

In order to extend our model to the nonlinear case, we consider the following kernel-generated functions instead of linear functions,

$$f_1(x)=K(x,A^T)w_1+b_1\ and\ f_2(x)=K(x,A^T)w_2+b_2. \quad (32)$$

The corresponding formulations are designed as follows

$$\min_{w_1,b_1,\epsilon_1,\xi}\ \frac{1}{2}\|Y-(K(A,A^T)\mu_1+eb_1)\|^2+C_1\nu_1\epsilon_1+\frac{1}{l}C_1e^T\xi \quad (33)$$
$$\text{s.t.}\quad Y-(K(A,A^T)\mu_1+eb_1)\ge-e\epsilon_1-2(1-p)\xi,$$
$$\epsilon_1\ge0,\xi\ge0,$$

and

$$\min_{w_2,b_2,\epsilon_2,\eta}\ \frac{1}{2}\|Y-(K(A,A^T)\mu_2+eb_2)\|^2+C_2\nu_2\epsilon_2+\frac{1}{l}C_2e^T\eta \quad (34)$$
$$\text{s.t.}\quad (K(A,A^T)\mu_2+eb_2)-Y\ge-e\epsilon_2-2p\eta$$
$$\epsilon_2\ge0,\eta\ge0,$$

where C_1, C_2, ν_1, and ν_2 are parameters chosen in advance, ξ and η are slack vectors. The dual formulations of (33) and (34) can be derived as follows

$$\max_{\alpha}\ -\frac{1}{2}\alpha^TH(H^TH)^{-1}H^T\alpha+Y^TH(H^TH)^{-1}H^T\alpha-Y^T\alpha \quad (35)$$
$$\text{s.t.}\quad e^T\alpha\le C_1\nu_1,$$
$$0\le\alpha\le C_1e/2(1-p)l,$$

and

$$\max_{\gamma}\ -\frac{1}{2}\gamma^TH(H^TH)^{-1}H^T\gamma-Y^TH(H^TH)^{-1}H^T\gamma+Y^T\gamma \quad (36)$$
$$\text{s.t.}\quad e^T\gamma\le C_2\nu_2,$$
$$0\le\gamma\le C_2e/2pl,$$

where $H=[K(A,A^T)\ \ e]$. Once (35) and (36) are solved, we can get the augmented vectors

$$\begin{bmatrix}w_1\\b_1\end{bmatrix}=(H^TH)^{-1}H^T(Y-\alpha), \quad (37)$$

and

$$\begin{bmatrix}w_2\\b_2\end{bmatrix}=(H^TH)^{-1}H^T(Y+\gamma). \quad (38)$$

After the optimal values w_1, w_2 and b_1, b_2 are calculated, the regression function is expressed as

$$f(x)=\frac{1}{2}(f_1(x)+f_2(x))=\frac{1}{2}(w_1+w_2)^T K(x, A^T)+\frac{1}{2}(b_1+b_2). \quad (39)$$

Compared with other algorithms, Asy-ν-TSVR owns following characteristics: First, it considers the sum of squared distances from the training points to the shifted functions $f_1(x)$ and $f_2(x)$ instead of others. Second, Asy-ν-TSVR introduces the asymmetric loss function $L_\epsilon^p(u)$ not ϵ-insensitive loss L_ϵ, and different penalty is proposed to give the samples locating in different positions. Third, Asy-ν-TSVR degrades into the TSVR when $p = 0.5$, so Asy-ν-TSVR is an extension of TSVR.

4 Discussion about the bound

In this section, we discuss the bounds of three algorithms, i.e., ν-SVR, Asy-ν-SVR, and Asy-ν-TSVR. In ν-SVR, there are same upper bounds for the points lying above and below the ν-tube. However, in Asy-ν-SVR and Asy-ν-TSVR, there are different upper bounds for the points above the hyper-plane $w^T x+b+\epsilon = 0$ and below the hyper-plane $w^T x + b - \epsilon = 0$.

Proposition 1 *The optimal solution in ν-SVR satisfies:*

$$\begin{aligned}
&\sum_{i=1}^{l} L(y_i < w^T x_i + b - \epsilon) \le \frac{1}{2} l\nu,\\
&\sum_{i=1}^{l} L(y_i > w^T x_i + b + \epsilon) \le \frac{1}{2} l\nu,\\
&\sum_{i=1}^{l} L(w^T x_i + b - \epsilon \le y_i \le w^T x_i + b + \epsilon) \ge l - l\nu, \quad (40)
\end{aligned}$$

where $L(a)$ stands for an indicator function, which is equal to one when a is true and is zero otherwise.

Please refer to literature [5] for proof of this proposition.

Proposition 2 *The optimal solution in Asy-ν-SVR satisfies:*

$$\begin{aligned}
&\sum_{i=1}^{l} L(y_i < w^T x_i + b - \epsilon) \le (1-p) l\nu,\\
&\sum_{i=1}^{l} L(y_i > w^T x_i + b + \epsilon) \le p l\nu,\\
&\sum_{i=1}^{l} L(w^T x_i + b - \epsilon \le y_i \le w^T x_i + b + \epsilon) \ge l - l\nu, \quad (41)
\end{aligned}$$

where p is the parameter related to asymmetric. It means that the points lying above and below the ν-tube have different upper bounds. And when p=0.5, it has the same upper bound as ν-SVR.

The proof of this proposition is in [18]. We can find that there are different upper bounds for the samples lying above and below the ϵ-tube in the Asy-ν-SVR.

Proposition 3 *The optimal solution in Asy-ν-TSVR satisfies:*

$$\begin{aligned}
&\sum_{i=1}^{l} L(y_i < w_1^T x_i + b_1 - \epsilon_1) \le 2(1-p) l\nu_1,\\
&\sum_{i=1}^{l} L(y_i > w_2^T x_i + b_2 + \epsilon_2) \le 2 p l\nu_2,\\
&\sum_{i=1}^{l} L(w_1^T x_i + b_1 - \epsilon_1 \le y_i \le w_2^T x_i + b_2 + \epsilon_2) \ge l - 2(1-p) l\nu_1 - 2 p l\nu_2, \quad (42)
\end{aligned}$$

where $L(a)$ stands for an indicator function, which is equal to one when a is true and is zero otherwise.

Proof Any point below the line $w_1^T x + b_1 - \epsilon_1 = 0$ satisfies $(w_1^T x_i + b_1 - \epsilon_1) - y_i = 2(1-p)\xi_i$, and $\xi_i > 0$. According to the complementary slackness condition, we have $\gamma_i = 0$. We can further get $\alpha_i = \frac{C_1}{2(1-p)l}$ from (22). Since $\beta \ge 0$, we get $e^T\alpha \le C_1\nu_1$ from (21), it means that $\sum_{i=1}^{l} \alpha_i \le C_1\nu_1$. From above, we get

$$\sum_{i=1}^{l} L(y_i < w_1^T x_i + b_1 - \epsilon_1) \le 2(1-p) l\nu_1. \quad (43)$$

Similarly, the points lying above the line $w_2^T x + b_2 + \epsilon_2 = 0$ satisfy

$$\sum_{i=1}^{l} L(y_i > w_2^T x_i + b_2 + \epsilon_2) \le 2 p l\nu_2. \quad (44)$$

Combining (43) and (44), we can get

$$\sum_{i=1}^{l} L(w_1^T x_i + b_1 - \epsilon_1 \le y_i \le w_2^T x_i + b_2 + \epsilon_2) \ge l - 2(1-p) l\nu_1 - 2 p l\nu_2. \quad (45)$$

Note that there are different upper bounds for the samples lying above the upper bound and below the down-bound in the Asy-ν-TSVR.

□

5 Numerical experiments

To demonstrate the validity of our Asy-ν-TSVR, we compare it with other four algorithms, i.e., ν-SVR, LS-SVR, TSVR, and Asy-ν-SVR using one artificial dataset, eleven benchmark datasets, and a real wheat dataset. For the experiment on each dataset, we use 5-fold cross-validation to evaluate the performance of five algorithms. That is to say, the dataset is split randomly into five subsets; one of those sets is reserved as a test set, and the rest is training set. This process is repeated five times, and the average of five testing results is used as the performance measure.

5.1 Evaluation criteria

In order to evaluate the performance of our Asy-ν-TSVR, the evaluation criteria are specified before presenting the experimental results. The size of testing set is denoted by l, while y_i denotes the real value of a sample point x_i, $\widehat{y_i}$ denotes the predicted value of x_i, $\overline{y} = \frac{1}{l}\sum_{i=1}^{l} y_i$ is the mean of $y_1, y_2, \cdots, y_l$. We use the following criteria for algorithm evaluation [34].

MAE: Mean absolute error, defined as

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}|y_i - \hat{y}_i|. \tag{46}$$

MAE is also a popular deviation measurement between the real and predicted values.

RMSE: Root mean squared error, defined as

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}. \tag{47}$$

SSE/SST: Ratio between sum of squared error and sum of squared deviation of testing samples, defined as

$$\text{SSE/SST} = \sum_{i=1}^{m}(y_i - \hat{y}_i)^2 / \sum_{i=1}^{m}(y_i - \bar{y})^2. \tag{48}$$

SSR/SST: Ratio between interpretable sum of squared deviation and real sum of squared deviation of testing samples, defined as

$$\text{SSR/SST} = \sum_{i=1}^{m}(\hat{y}_i - \bar{y})^2 / \sum_{i=1}^{m}(y_i - \bar{y})^2. \tag{49}$$

In most cases, small SSE/SST means there is good agreement between the estimates and the real values, and decreasing SSE/SST is usually accompanied by an increase in SSR/SST.

Time: The total training time and testing time.

5.2 Parameter selection

The performance of these five algorithms depends heavily on the choices of parameters. In our experiments, we choose optimal values for the parameters by the grid search method. In five algorithms, the Gaussian kernel parameter σ is selected from the set $\{2^i | i = -4, -3, \cdots, 8\}$. In TSVR and Asy-$\nu$-TSVR, we set C_1=C_2, ν_1=ν_2 and ϵ_1=ϵ_2 to degrade the computational complexity of parameter selection. The parameter C is searched from the set $\{2^i | i = -3, -2, \cdots, 8\}$. The optimal values for ν in four algorithms are chosen from the set $\{0.1, 0.2, \cdots, 0.9\}$. The optimal values for ϵ in algorithms are selected from the set $\{0.1, 0.2, \cdots, 0.9\}$. Parameter p is searched from the set $\{0.2, 0.4, 0.45, 0.5, 0.55, 0.6, 0.8\}$.

5.3 Experiment on artificial dataset with noises

To evaluate the performance of Asy-ν-TSVR, we carry out an artificial experiment under different cases. We firstly generate 100 points $(X_i, i = 1, \cdots, 100)$ following a uniform distribution in $[0, 1]^5$, linear function $Y(X) = w^T X + b + (\delta_{\chi^2} - 4)$ with $w = [1; 0.5; -0.5; -1; 2]$, $b = -3$ is used to calculate their values, here δ_{χ^2} follows a chi-squared distribution with 4 degrees of freedom. The generated points $(X_i, Y_i), i = 1, \cdots, 100$ are regarded as samples. Subsequently we randomly select 5% of the samples and replace their observed results by random values following a uniform distribution in the range of $[-15, 15]$; thus, the new samples $(X_i, Y_i), i = 1, \cdots, 100$ are produced. Finally, a uniform distribution in the range of $[-30, 30]$ is used to build another samples. Fivefold cross-validation is applied as before, the experimental results of five algorithms are listed in Table 1. Apparently, our proposed algorithm Asy-ν-TSVR always performs the lowest MAE among five algorithms on this artificial dataset with different noises. The reason may lie in that our Asy-ν-TSVR adopts the pinball loss function but ν-SVR, LS-SVR, and TSVR adopt ϵ-insensitive loss. It makes Asy-ν-TSVR less sensitive to noises and has better generalization ability since samples from different positions are given different punishments. The Gaussian kernel function $k(x_i, x_j) = \exp(-||x_i - x_j||^2/\sigma^2)$ is used on this artificial dataset. The optimal parameters used in the experiment are listed in the last column of Table 1.

5.4 Experiments on benchmark datasets

To further verify the efficiency of our algorithm, we conduct experiments on eleven benchmark datasets from the UCI machine learning repository[1]. The datasets are Auto Price, Bodyfat, Chwirut, Con. S, Diabetes, Machine-Cpu,

[1] http://archive.ics.uci.edu/ml/datasets.html

Table 1 Testing errors on artificial datasets with different noises

Datasets	Algorithms	MAE	RMSE	SSE/SST	SSR/SST	Time(s)	$(C,\ \sigma,\ \nu,\ \epsilon,\ p)$
Artificial dataset	ν-SVR	2.4868 ± 0.4598	3.4496 ± 0.7586	1.2682 ± 0.1901	0.2661 ± 0.1880	0.0801	(64, 0.0625, 0.8, ∼, ∼)
(100 × 5)	Asy-ν-SVR	2.3798 ± 0.4255	3.1726 ± 0.7513	1.0600 ± 0.0481	0.0565 ± 0.0417	0.0490	(256, 0.0625, 0.7, ∼, 0.6)
	LS-SVR	2.4207 ± 0.3673	3.0895 ± 0.6751	1.0121 ± 0.0690	0.0425 ± 0.0307	0.0101	(0.125, 0.5, ∼, ∼, ∼)
	TSVR	2.4198 ± 0.3215	3.0952 ± 0.6285	1.0247 ± 0.1248	0.0680 ± 0.0507	0.0977	(4, 256, ∼, 0.9, ∼)
	Asy-ν-TSVR	*2.3720 ± 0.3324*	3.1457 ± 0.6380	1.0622 ± 0.1647	0.1184 ± 0.0932	0.0903	(256, 256, 0.6, ∼, 0.8)
Artificial dataset	ν-SVR	2.5779 ± 0.4800	3.2318 ± 0.6819	0.9925 ± 0.1314	0.1322 ± 0.0678	0.0735	(256, 0.25, 0.2, ∼, ∼)
[−15, 15]	Asy-ν-SVR	2.5650 ± 0.4166	3.2239 ± 0.5760	0.9993 ± 0.1552	0.1023 ± 0.0879	0.1036	(256, 2, 0.4, ∼, 0.6)
(100 × 5)	LS-SVR	2.5186 ± 0.3909	3.2381 ± 0.6609	1.0234 ± 0.3011	0.1797 ± 0.1425	0.0253	(128, 16, ∼, ∼, ∼)
	TSVR	2.4962 ± 0.4871	3.2477 ± 0.7039	1.0593 ± 0.4585	0.2801 ± 0.2304	0.1141	(16, 128, ∼, 0.9, ∼)
	Asy-ν-TSVR	*2.4588 ± 0.4915*	3.2822 ± 0.7753	1.0551 ± 0.3539	0.2495 ± 0.1525	0.1310	(256, 128, 0.6, ∼, 0.8)
Artificial dataset	ν-SVR	3.6298 ± 0.5305	5.9945 ± 1.3656	1.0408 ± 0.0544	0.0409 ± 0.0544	0.6224	(256, 0.0625, 0.3, ∼, ∼)
[−30, 30]	Asy-ν-SVR	3.5152 ± 0.5740	5.9683 ± 1.3446	1.0323 ± 0.0416	0.0322 ± 0.0414	0.0841	(128, 0.0625, 0.9, ∼, 0.2)
(100 × 5)	LS-SVR	3.5512 ± 0.6631	5.7974 ± 1.3532	0.9742 ± 0.0838	0.0899 ± 0.0659	0.0222	(128, 32, ∼, ∼, ∼)
	TSVR	3.4882 ± 0.6788	5.7888 ± 1.2884	0.9754 ± 0.0739	0.0823 ± 0.0539	0.1600	(16, 256, ∼, 0.1, ∼)
	Asy-ν-TSVR	*3.4517 ± 0.8390*	5.8438 ± 1.3738	0.9901 ± 0.0939	0.1081 ± 0.0939	0.0908	(256, 256, 0.8, ∼, 0.8)

Italic type shows the best result

Pyrimidibes, Triazines, Housing, Istanbul Stock Exchange, and Yacht Hydrodynamics. Both linear kernel and Gaussian kernel are considered in five algorithms. Moreover, statistical tests, including the Wilcoxon signed-rank test and Friedman test, are also used to demonstrate the validity of our proposed method. At last, we also study the relationship between the efficiency of our Asy-ν-TSVR and the asymmetry of the datasets in Section 5.4.4.

5.4.1 Result comparison and discussion

The experimental results of five algorithms are summarized in Table 2 when linear kernel is employed, and in Table 3 when Gaussian kernel is used. In the error items, the first item denotes the mean value of five times testing results, and the second item stands for plus or minus the standard deviation. Time denotes the mean value of time taken by five time experiments, and each experimental time consists of training time and testing time.

In terms of MAE criterion, from Table 2, we can find that Asy-ν-SVR produces the lowest testing error among five algorithms in most cases when linear kernel is employed, followed by Asy-ν-TSVR. Both of them employ the pinball loss, which implies that pinball loss is more suitable than ϵ-insensitive loss for these datasets. In addition, in terms of RMSE criterion, Asy-ν-TSVR yields the comparable testing error with Asy-ν-SVR and TSVR. It further shows that pinball loss is suitable for these datasets. Meanwhile, we can find that small MAE and RMSE corresponds to small SSE/SST and large SSR/SST in most cases.

No matter from MAE criterion or RMSE criterion, from Table 3, we can find that Asy-ν-TSVR yields the lowest testing errors among five algorithms on most datasets. Followed by TSVR. On dataset Pyrimidibes, although Asy-ν-TSVR produces lightly higher MAE than ν-SVR and Asy-ν-SVR. Asy-ν-TSVR produces the lowest RMSE among five algorithms. In addition, we can also find that Asy-ν-SVR yields slightly lower MAE than ν-SVR. It further testifies that the pinball loss function is effective in the Asy-ν-TSVR. Meanwhile, we can find that small MAE and RMSE corresponds to small SSE/SST and large SSR/SST in most cases.

In terms of computational time, from Tables 2 and 3, we can really find that TSVR costs more running time than Asy-ν-TSVR for most cases. It means that the introduction of pinball loss function does not increase computational cost of Asy-ν-TSVR. In addition, ν-SVR and Asy-ν-SVR cost larger running time than TSVR and Asy-ν-TSVR. The main reason lies in that they solve a larger-sized QPP but TSVR and Asy-ν-TSVR solve a pair of smaller-sized QPPs. LS-SVR costs the least running time among five algorithms since it solves a system of linear equations rather than a QPP.

To further verify the computational burden with different p values in our Asy-ν-TSVR, we analyze the experiments on 11 benchmark datasets with different p values. The average values of computational time on each dataset are summarized in Table 4, and the MAE of Asy-ν-SVR and Asy-ν-TSVR are displayed in Table 5.

From Table 4, we can find that Asy-ν-TSVR costs less time when p value is near to 0.5. That is, smaller or larger p

Table 2 Performance comparisons of five algorithms with linear case on eleven benchmark datasets

Datasets	Algorithms	MAE	RMSE	SSE/SST	SSR/SST	Time (s)	(C, σ, ν, ϵ, p)
Auto Price	ν-SVR	1960.7 ± 504.5	2775.2 ± 763.1	0.3969 ± 0.2865	0.9727 ± 0.5616	1.3080	(2, ∼, 0.4, ∼, ∼)
(159 × 15)	Asy-ν-SVR	*1955.2 ± 538.8*	*2772.9 ± 795.7*	0.3864 ± 0.2623	0.9795 ± 0.5608	0.7302	(8, ∼, 0.5, ∼, 0.45)
	LS-SVR	2355.4 ± 691.6	3351.5 ± 1245.1	0.4839 ± 0.1957	1.0103 ± 0.7162	0.0277	(0.1250, ∼, ∼, ∼, ∼)
	TSVR	2486.1 ± 602.5	3512.1 ± 1121.0	0.5693 ± 0.2756	1.0918 ± 0.7527	0.1683	(256, ∼, ∼, 0.9, ∼)
	Asy-ν-TSVR	2493.5 ± 599.3	3517.3 ± 1116.4	0.5717 ± 0.2769	1.0937 ± 0.7536	0.3989	(256, ∼, 0.9, ∼, 0.8)
Bodyfat	ν-SVR	0.0808 ± 0.0155	0.0998 ± 0.0194	0.3924 ± 0.0798	0.7378 ± 0.2306	1.7772	(256, ∼, 0.4, ∼, ∼)
(252 × 13)	Asy-ν-SVR	0.0761 ± 0.0118	0.0972 ± 0.0159	0.3830 ± 0.1145	0.7420 ± 0.2935	1.5051	(256, ∼, 0.4, ∼, 0.2)
	LS-SVR	0.0755 ± 0.0071	0.0940 ± 0.0080	0.3906 ± 0.2275	0.8575 ± 0.4109	0.0839	(256, ∼, ∼, ∼, ∼)
	TSVR	0.0744 ± 0.0074	0.0936 ± 0.0074	0.3887 ± 0.2330	0.8377 ± 0.4309	0.2571	(1, ∼, ∼, 0.9, ∼)
	Asy-ν-TSVR	*0.0743 ± 0.0071*	*0.0933 ± 0.0079*	0.3898 ± 0.2428	0.8332 ± 0.4275	0.2683	(128, ∼, 0.8, ∼, 0.4)
Chwirut	ν-SVR	0.1860 ± 0.0380	0.2156 ± 0.0352	0.7261 ± 0.3106	1.2780 ± 0.3868	0.7406	(256, ∼, 0.1, ∼, ∼)
(214 × 1)	Asy-ν-SVR	0.1860 ± 0.0380	0.2156 ± 0.0352	0.7261 ± 0.3106	1.2780 ± 0.3868	0.3815	(256, ∼, 0.1, ∼, 0.5)
	LS-SVR	0.1251 ± 0.0141	0.1449 ± 0.0186	0.3097 ± 0.0539	0.6670 ± 0.1285	0.0265	(1, ∼, ∼, ∼, ∼)
	TSVR	0.1239 ± 0.0143	*0.1447 ± 0.0183*	0.3085±0.0530	0.7265±0.1346	0.1365	(0.25, ∼, ∼, 0.1, ∼)
	Asy-ν-TSVR	*0.1214 ± 0.0144*	0.1492 ± 0.0221	0.3262 ± 0.0513	0.7425 ± 0.1155	0.1558	(32, ∼, 0.8, ∼, 0.8)
Con. S	ν-SVR	0.1873 ± 0.0230	0.2315 ± 0.0171	0.6660 ± 0.2019	0.5204 ± 0.3562	0.1378	(256, ∼, 0.4, ∼, ∼)
(103 × 7)	Asy-ν-SVR	*0.1855 ± 0.0329*	0.2298 ± 0.0365	0.6798 ± 0.2773	0.4178 ± 0.3431	0.0909	(32, ∼, 0.7, ∼, 0.8)
	LS-SVR	0.1921 ± 0.0368	0.2314 ± 0.0290	0.6807 ± 0.2438	0.4450 ± 0.3291	0.0179	(4, ∼, ∼, ∼, ∼)
	TSVR	0.1936 ± 0.0359	0.2353 ± 0.0314	0.7038 ± 0.2615	0.5251 ± 0.4112	0.0741	(0.5, ∼, ∼, 0.9, ∼)
	Asy-ν-TSVR	0.1898 ± 0.0387	*0.2271±0.0284*	0.6544 ± 0.2336	0.4056 ± 0.2361	0.0798	(128, ∼, 0.1, ∼, 0.8)
Diabetes	ν-SVR	0.4451 ± 0.0809	0.5836 ± 0.0898	0.8833 ± 0.3579	0.4607 ± 0.2994	0.0452	(4, ∼, 0.2, ∼, ∼)
(43×2)	Asy-ν-SVR	*0.4407 ± 0.0763*	*0.5788 ± 0.0867*	0.8512 ± 0.2673	0.3709 ± 0.1551	0.0292	(8, ∼, 0.2, ∼, 0.4)
	LS-SVR	0.4940 ± 0.0686	0.6187 ± 0.0400	1.0257 ± 0.4498	0.5972 ± 0.3892	0.0160	(256, ∼, ∼, ∼, ∼)
	TSVR	0.4842 ± 0.0976	0.6248 ± 0.0876	1.0849 ± 0.5940	0.7703 ± 0.4881	0.0382	(16, ∼, ∼, 0.1, ∼)
	Asy-ν-TSVR	0.4830 ± 0.0935	0.6349 ± 0.0846	1.0745 ± 0.5154	0.7420 ± 0.3795	0.0367	(128, ∼, 0.4, ∼, 0.8)
Machine-Cpu	ν-SVR	34.8133 ± 16.9816	64.1274 ± 48.3133	0.2256 ± 0.1593	0.5957 ± 0.2035	2.2050	(8, ∼, 0.9, ∼, ∼)
(209×6)	Asy-ν-SVR	*33.774 ± 15.659*	*62.6452 ± 46.3595*	0.2173 ± 0.1526	0.5876 ± 0.1940	2.2909	(64, ∼, 0.6, ∼, 0.55)
	LS-SVR	43.3543 ± 16.0953	71.1977 ± 40.4697	0.3836 ± 0.4108	1.0469 ± 0.4421	0.1174	(0.1250, ∼, ∼, ∼, ∼)
	TSVR	41.6387 ± 15.7333	69.4367 ± 41.8549	0.3489 ± 0.3562	0.9557 ± 0.3817	0.1961	(128, ∼, ∼, 0.1, ∼)
	Asy-ν-TSVR	43.2571 ± 16.1525	71.1210 ± 40.6607	0.3823 ± 0.4094	1.0434 ± 0.4405	0.1910	(256, ∼, 0.9, ∼, 0.8)
Pyrimidibes	ν-SVR	0.0589 ± 0.0244	*0.0843 ± 0.0481*	1.3703 ± 1.9814	1.4539 ± 1.7678	0.0902	(16, ∼, 0.7, ∼, ∼)
(74×27)	Asy-ν-SVR	*0.0574 ± 0.0198*	0.0844 ± 0.0440	1.2238 ± 1.5000	1.3261 ± 1.4229	0.0520	(16, ∼, 0.3, ∼, 0.45)
	LS-SVR	0.0624 ± 0.0195	0.0891 ± 0.0483	1.2949 ± 1.5905	1.2571 ± 1.3706	0.0161	(2, ∼, ∼, ∼, ∼)
	TSVR	0.0976 ± 0.0418	0.1513 ± 0.0758	5.2084 ± 8.6257	5.1894 ± 8.0761	0.0653	(0.25, ∼, ∼, 0.9, ∼)
	Asy-ν-TSVR	0.0948 ± 0.0402	0.1452 ± 0.0702	4.9105 ± 8.2625	4.9572 ± 7.6816	0.0858	(8, ∼, 0.3, ∼, 0.2)
Triazines	ν-SVR	0.1010 ± 0.0136	0.1435 ± 0.0127	0.8867 ± 0.1675	0.2204 ± 0.0644	0.4743	(8, ∼, 0.8, ∼, ∼)
(186 × 60)	Asy-ν-SVR	*0.1007 ± 0.0132*	0.1427 ± 0.0121	0.8756 ± 0.1559	0.2198 ± 0.0691	0.6135	(8, ∼, 0.8, ∼, 0.55)
	LS-SVR	0.1041 ± 0.0064	*0.1421 ± 0.0065*	0.8668 ± 0.1106	0.2312 ± 0.1262	0.0221	(0.2500, ∼, ∼, ∼, ∼)
	TSVR	0.1167 ± 0.0103	0.1599 ± 0.0125	1.1278 ± 0.3143	0.6550 ± 0.3033	0.1258	(8, ∼, ∼, 0.1, ∼)
	Asy-ν-TSVR	0.1135 ± 0.0082	0.1586 ± 0.0134	1.1032 ± 0.2847	0.6046 ± 0.2553	0.1395	(16, ∼, 0.5, ∼, 0.2)
Housing	ν-SVR	*3.3147 ± 0.1912*	5.0119 ± 0.7285	0.3039 ± 0.0751	0.6392 ± 0.1127	6.7073	(64, ∼, 0.9, ∼, ∼)
(506 × 13)	Asy-ν-SVR	3.2242 ± 0.1997	5.0568 ± 0.9189	0.3122 ± 0.1055	0.6703 ± 0.1116	15.9027	(256, ∼, 0.7, ∼, 0.6)
	LS-SVR	3.4189 ± 0.2000	4.9763 ± 0.7865	0.3001 ± 0.0828	0.7441 ± 0.1227	0.7548	(2, ∼, ∼, ∼, ∼)
	TSVR	3.3564 ± 0.2078	*4.9689 ± 0.8472*	0.3000 ± 0.0920	0.7350 ± 0.1201	1.2754	(8, ∼, ∼, 0.1, ∼)
	Asy-ν-TSVR	3.3927 ± 0.2136	4.9748 ± 0.7973	0.2999 ± 0.0836	0.7447 ± 0.1194	1.1837	(256, ∼, 0.9, ∼, 0.8)

Table 2 (continued)

Datasets	Algorithms	MAE	RMSE	SSE/SST	SSR/SST	Time (s)	(C, σ, ν, ϵ, p)
Istanbul Stock Exchange	ν-SVR	0.0055 ± 0.0016	0.0067 ± 0.0016	0.4360 ± 0.2043	0.9139 ± 0.1943	3.1161	(256, ∼, 0.5, ∼, ∼)
(536 × 8)	Asy-ν-SVR	0.0051 ± 0.0014	0.0062 ± 0.0014	0.3689 ± 0.1426	0.8565 ± 0.1489	4.1568	(256, ∼, 0.9, ∼, 0.2)
	LS-SVR	*0.0037* ± *0.0004*	*0.0049* ± *0.0005*	0.2254 ± 0.0449	0.7758 ± 0.1142	0.0961	(256, ∼, ∼, ∼, ∼)
	TSVR	0.0038 ± 0.0003	0.0050 ± 0.0005	0.2333 ± 0.0467	0.8137 ± 0.1636	0.8496	(0.125, ∼, ∼, 0.1, ∼)
	Asy-ν-TSVR	*0.0037* ± *0.0003*	*0.0049* ± *0.0005*	0.2267 ± 0.0441	0.8125 ± 0.1266	0.4692	(2, ∼, 0.6, ∼, 0.55)
Yacht Hydrodynamics	ν-SVR	7.8657 ± 2.1482	15.0612 ± 3.2395	1.0231 ± 0.0995	0.2416 ± 0.0709	3.1683	(256, ∼, 0.9, ∼, ∼)
(308 × 6)	Asy-ν-SVR	7.8272 ± 2.1403	14.8498 ± 3.2994	0.9932 ± 0.1095	0.2219 ± 0.0803	2.3730	(256, ∼, 0.9, ∼, 0.45)
	LS-SVR	*7.0280* ± *0.8833*	9.7910 ± 1.9086	0.4343 ± 0.0207	0.3887 ± 0.1873	0.1289	(1, ∼, ∼, ∼, ∼)
	TSVR	7.2462 ± 0.3822	*9.3299* ± *1.2581*	0.4039 ± 0.0603	0.6221 ± 0.2909	0.4217	(32, ∼, ∼, 0.1, ∼)
	Asy-ν-TSVR	7.4967 ± 0.2038	9.3972 ± 0.9939	0.4155 ± 0.0892	0.7320 ± 0.3466	0.3702	(256, ∼, 0.9, ∼, 0.8)

Italic type shows the best result

value will increase the computational burden. From Table 5, we can see that the MAE of both Asy-ν-SVR and Asy-ν-TSVR are monotonic as p varies from 0.2 to 0.8 for most datasets. For example, the MAE is monotonically decreasing on Auto Price and Machine-Cpu. However, it is monotonically increasing on Pyrimidibes. The p value controls the imbalance ratio of punishments on different samples. The experimental results in Table 5 verify that the introduction of pinball loss can actually improve the performance of the model for Asymmetric datasets. And different datasets fit for different p values.

5.4.2 Wilcoxon signed-rank test

In these benchmark experiments, both linear kernel and nonlinear kernel are employed. To verify that which is more effective, the Wilcoxon signed-rank test [35] is used.

The Wilcoxon signed-rank test is a nonparametric statistical hypothesis test used when comparing two related samples, matched samples, or repeated measurements on a single sample to assess whether their population mean ranks differ (i.e., it is a paired difference test). Here, we use the "signrank" function in Matlab to do the Wilcoxon signed-rank test. For each algorithm, the null hypothesis H_0 demonstrates that the MAE in linear case is larger than that in the nonlinear case.

Table 6 lists the p values of a right-sided Wilcoxon signed-rank test, the test statistic W and a logical value indicating the test decision. $h = 1$ indicates a rejection of the null hypothesis, and $h = 0$ indicates a failure to reject the null hypothesis at the 5% significance level. We can see that the results are $h = 0$ for all five algorithms, which means that the nonlinear case is superior to the linear case.

5.4.3 Paired t test

From Table 3, one can easily observe that our proposed Asy-ν-TSVR does not outperform four other algorithms for all datasets in the nonlinear case. We use five times testing results corresponding to the lowest error to perform the paired t test. The null hypothesis H_0 demonstrates that there is no significant difference between the two algorithms tested. The hypothesis H_0 is rejected if the p value is less than 0.05 under the significance level $\alpha = 0.05$. We compute the p values between Asy-ν-TSVR and other algorithms. From the experimental results, we can find that there are significant differences between Asy-ν-TSVR and other algorithms on datasets Chwirut, Housing, and Yacht Hydrodynamics. However, there is no significant difference on other datasets.

5.4.4 Friedman test

To further demonstrate the validity of our proposed Asy-ν-TSVR in the nonlinear case, Friedman test [36, 37] is employed. We assume Friedman test with the corresponding post hoc tests which is considered to be a simple, nonparametric yet safe test. For this, the average ranks of five algorithms on MAE for all datasets are calculated and listed in Table 7. Under the null hypothesis that all algorithms are equivalent, one can compute the Friedman statistic according to (50),

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right], \tag{50}$$

Table 3 Performance comparisons of five algorithms with Gaussian kernel on eleven benchmark datasets

Datasets	Algorithms	MAE	RMSE	SSE/SST	SSR/SST	Time (s)	$(C,\ \sigma,\ \nu,\ \epsilon,\ p)$
Auto Price	ν-SVR	4944.5 ± 2580.1	6308.9 ± 3140.0	1.6240 ± 1.0536	0.6240 ± 1.0536	0.4577	(1, 16, 0.4, ∼, ∼)
(159 × 15)	Asy-ν-SVR	4944.4 ± 2580.1	6308.9 ± 3140.0	1.6240 ± 1.0536	0.6241 ± 1.0536	0.4356	(4, 256, 0.3, ∼, 0.55)
	LS-SVR	*2618.7 ± 1127.9*	*3944.5 ± 2296.6*	0.6653 ± 0.4088	0.8397 ± 0.6294	0.0305	(2, 256, ∼, ∼, ∼)
	TSVR	4629.7 ± 1108.4	5689.2 ± 1884.4	1.6373 ± 1.3147	0.6974 ± 1.2752	0.2408	(64, 8, ∼, 0.1, ∼)
	Asy-ν-TSVR	4629.7 ± 1108.4	5689.2 ± 1884.4	1.6373 ± 1.3147	0.6974 ± 1.2752	0.1899	(256, 8, 0.9, ∼, 0.8)
Bodyfat	ν-SVR	0.0764 ± 0.0120	0.0952 ± 0.0143	0.3761 ± 0.1428	0.6898 ± 0.2317	0.5418	(256, 1, 0.5, ∼, ∼)
(252×13)	Asy-ν-SVR	0.0750 ± 0.0100	0.0937 ± 0.0135	0.3684 ± 0.1489	0.6744 ± 0.2565	0.6891	(256, 1, 0.5, ∼, 0.8)
	LS-SVR	0.0724 ± 0.0083	0.0903 ± 0.0106	0.3544 ± 0.1831	0.7766 ± 0.3058	0.0234	(256, 4, ∼, ∼, ∼)
	TSVR	0.0720 ± 0.0076	*0.0896 ± 0.0086*	0.3555 ± 0.2044	0.8021 ± 0.3618	0.3710	(1, 16, ∼, 0.1, ∼)
	Asy-ν-TSVR	*0.0717 ± 0.0084*	*0.0896 ± 0.0097*	0.3608 ± 0.2238	0.8182 ± 0.3923	0.2899	(128, 16, 0.8, ∼, 0.4)
Chwirut	ν-SVR	0.0725 ± 0.0136	0.0806 ± 0.0094	0.1008 ± 0.0353	1.0839 ± 0.1568	0.4774	(256, 0.0625, 0.5, ∼, ∼)
(214 × 1)	Asy-ν-SVR	0.0615 ± 0.0111	0.0724 ± 0.0086	0.0814 ± 0.0290	1.1626 ± 0.1930	0.4150	(256, 0.125, 0.6, ∼, 0.2)
	LS-SVR	0.0283 ± 0.0120	0.0387 ± 0.0170	0.0242 ± 0.0168	1.0153 ± 0.1420	0.0190	(256, 0.25, ∼, ∼, ∼)
	TSVR	0.0274 ± 0.0131	0.0377 ± 0.0176	0.0234 ± 0.0169	1.0110 ± 0.1344	0.2730	(0.125, 0.5, ∼, 0.1, ∼)
	Asy-ν-TSVR	*0.0272 ± 0.0135*	*0.0376 ± 0.0181*	0.0234 ± 0.0175	1.0113 ± 0.1356	0.2290	(16, 0.5, 0.7, ∼, 0.55)
Con. S	ν-SVR	0.1940 ± 0.0485	0.2351 ± 0.0446	0.6748 ± 0.2543	0.4114 ± 0.2027	0.0770	(128, 1, 0.9, ∼, ∼)
(103 × 7)	Asy-ν-SVR	0.1885 ± 0.0453	0.2283 ± 0.0443	0.6478 ± 0.2943	0.3353 ± 0.1512	0.0974	(128, 1, 0.9, ∼, 0.45)
	LS-SVR	0.1884 ± 0.0287	0.2263 ± 0.0259	0.6521 ± 0.2406	0.4460 ± 0.2833	0.0348	(128, 4, ∼, ∼, ∼)
	TSVR	0.1814 ± 0.0304	0.2219 ± 0.0266	0.6255 ± 0.2460	0.5378 ± 0.2824	0.1240	(0.125, 64, ∼, 0.9, ∼)
	Asy-ν-TSVR	*0.1802 ± 0.0290*	*0.2194 ± 0.0238*	0.6077 ± 0.2263	0.5171 ± 0.2709	0.1063	(16, 64, 0.1, ∼, 0.2)
Diabetes	ν-SVR	0.4514 ± 0.1034	0.5765 ± 0.0827	0.8948 ± 0.4455	0.5266 ± 0.3351	0.0462	(256, 16, 0.3, ∼, ∼)
(43 × 2)	Asy-ν-SVR	0.4471 ± 0.0933	0.5628 ± 0.0739	0.8339 ± 0.3441	0.4231 ± 0.2517	0.0226	(128, 16, 0.3, ∼, 0.45)
	LS-SVR	0.4774 ± 0.0913	0.5832 ± 0.0414	0.9163 ± 0.4262	0.6505 ± 0.4362	0.0117	(256, 32, ∼, ∼, ∼)
	TSVR	0.4549 ± 0.0986	0.5419 ± 0.0729	0.7591 ± 0.2888	0.6417 ± 0.2493	0.1078	(4, 64, ∼, 0.1, ∼)
	Asy-ν-TSVR	*0.4383 ± 0.0840*	*0.5375 ± 0.0750*	0.7578 ± 0.3408	0.6357 ± 0.2768	0.0642	(128, 64, 0.2, ∼, 0.8)
Machine-Cpu	ν-SVR	92.7969 ± 23.2384	148.3253 ± 70.9127	1.2708 ± 0.5598	0.2939 ± 0.5545	0.6891	(256, 256, 0.5, ∼, ∼)
(209 × 6)	Asy-ν-SVR	92.4516 ± 23.1962	148.0403 ± 70.9974	1.2657 ± 0.5606	0.2930 ± 0.5521	0.4715	(256, 256, 0.5, ∼, 0.4)
	LS-SVR	*62.3803 ± 21.4439*	*118.7040 ± 79.3577*	0.8996 ± 0.7644	0.6011 ± 0.4166	0.0199	(128, 256, ∼, ∼, ∼)
	TSVR	81.4597 ± 40.3451	147.4851 ± 90.0133	1.0404 ± 0.2255	0.2314 ± 0.1800	0.2309	(4, 8, ∼, 0.1, ∼)
	Asy-ν-TSVR	80.8976 ± 40.0417	147.1105 ± 89.9115	1.0350 ± 0.2267	0.2255 ± 0.1725	0.2128	(256, 8, 0.2, ∼, 0.8)
Pyrimidibes	ν-SVR	*0.0468 ± 0.0170*	0.0715 ± 0.0455	0.9105 ± 1.1604	1.2259 ± 1.1602	0.1071	(256, 2, 0.4, ∼, ∼)
(74 × 27)	Asy-ν-SVR	*0.0468 ± 0.0170*	0.0715 ± 0.0455	0.9105 ± 1.1604	1.2259 ± 1.1602	0.0519	(256, 2, 0.4, ∼, 0.5)
	LS-SVR	0.0468 ± 0.0180	0.0704 ± 0.0449	0.9234 ± 1.1962	1.2161 ± 1.1616	0.0244	(128, 2, ∼, ∼, ∼)
	TSVR	0.0494 ± 0.0132	0.0670 ± 0.0282	0.7628 ± 0.8594	1.1787 ± 0.9547	0.0456	(0.25, 64, ∼, 0.1, ∼)
	Asy-ν-TSVR	0.0483 ± 0.0120	*0.0661 ± 0.0271*	0.7203 ± 0.7521	1.1637 ± 0.8907	0.0860	(32, 64, 0.7, ∼, 0.8)
Triazines	ν-SVR	0.0968 ± 0.0128	*0.1354 ± 0.0161*	0.7907 ± 0.1889	0.2962 ± 0.0918	0.4574	(256, 2, 0.3, ∼, ∼)
(186 × 60)	Asy-ν-SVR	*0.0946 ± 0.0147*	0.1431 ± 0.0200	0.8858 ± 0.2455	0.3457 ± 0.1253	0.2158	(64, 1, 0.7, ∼, 0.4)
	LS-SVR	0.0981 ± 0.0116	0.1404 ± 0.0106	0.8493 ± 0.1587	0.3570 ± 0.1511	0.0245	(4, 1, ∼, ∼, ∼)
	TSVR	0.1026 ± 0.0080	0.1424 ± 0.0097	0.8664 ± 0.0881	0.2431 ± 0.1202	0.2175	(0.5, 256, ∼, 0.9, ∼)
	Asy-ν-TSVR	0.1006 ± 0.0129	0.1451 ± 0.0172	0.8937 ± 0.1105	0.2355 ± 0.0243	0.1427	(64, 256, 0.8, ∼, 0.2)
Housing	ν-SVR	5.4063 ± 0.3851	8.2144 ± 0.5051	0.8119 ± 0.0732	0.2435 ± 0.0503	11.0028	(256, 64, 0.8, ∼, ∼)
(506 × 13)	Asy-ν-SVR	5.3764 ± 0.3421	8.1876 ± 0.5161	0.8062 ± 0.0693	0.2468 ± 0.0387	5.0205	(256, 64, 0.9, ∼, 0.45)
	LS-SVR	3.0499 ± 0.0831	4.4672 ± 0.3207	0.2410 ± 0.0333	0.8581 ± 0.0969	0.0905	(256, 64, ∼, ∼, ∼)
	TSVR	*2.4920 ± 0.1778*	*3.6393 ± 0.4425*	0.1588 ± 0.0240	0.8670 ± 0.0631	2.0754	(8, 256, ∼, 0.1, ∼)
	Asy-ν-TSVR	2.5397 ± 0.1867	3.6822 ± 0.4688	0.1627 ± 0.0264	0.8680 ± 0.0535	1.4429	(256, 256, 0.9, ∼, 0.8)
Istanbul Stock Exchange	ν-SVR	0.0043 ± 0.0004	0.0059 ± 0.0007	0.3273 ± 0.0752	0.7896 ± 0.1670	3.3291	(256, 0.0625, 0.3, ∼, ∼)

Table 3 (continued)

Datasets	Algorithms	MAE	RMSE	SSE/SST	SSR/SST	Time (s)	(C, σ, ν, ϵ, p)
(536 × 8)	Asy-ν-SVR	0.0041 ± 0.0003	0.0056 ± 0.0007	0.2925 ± 0.0615	0.7848 ± 0.1470	1.6578	(128, 0.0625, 0.6, ∼, 0.4)
	LS-SVR	*0.0037 ± 0.0004*	*0.0049 ± 0.0005*	0.2262 ± 0.0444	0.7747 ± 0.1139	0.0833	(256, 1, ∼, ∼, ∼)
	TSVR	*0.0037 ± 0.0004*	*0.0049 ± 0.0005*	0.2314 ± 0.0495	0.8354 ± 0.1355	1.3920	(0.125, 8, ∼, 0.2, ∼)
	Asy-ν-TSVR	*0.0037 ± 0.0004*	*0.0049 ± 0.0005*	0.2304 ± 0.0498	0.8391 ± 0.1328	2.2095	(64, 8, 0.5, ∼, 0.5)
Yacht Hydro-dynamics	ν-SVR	7.3725 ± 2.1892	14.8909 ± 3.4073	0.9964 ± 0.1181	0.2567 ± 0.0684	1.2566	(256, 0.125, 0.9, ∼, ∼)
(308 × 6)	Asy-ν-SVR	7.2913 ± 2.1579	14.6866 ± 3.3689	0.9690 ± 0.1142	0.2427 ± 0.0650	1.2460	(256, 0.125, 0.9, ∼, 0.45)
	LS-SVR	1.6644 ± 0.5340	3.1793 ± 1.3592	0.0452 ± 0.0230	0.8160 ± 0.1036	0.0395	(256, 0.125, ∼, ∼, ∼)
	TSVR	0.4125 ± 0.1925	0.9950 ± 0.5742	0.0047 ± 0.0034	0.9552 ± 0.0334	0.5196	(128, 0.125, ∼, 0.1, ∼)
	Asy-ν-TSVR	*0.4119 ± 0.1917*	*0.9930 ± 0.5717*	0.0047 ± 0.0033	0.9554 ± 0.0333	0.5434	(256, 0.125, 0.4, ∼, 0.8)

Italic type shows the best result

where $R_j = \frac{1}{N}\sum_i r_i^j$, and r_i^j denotes the j-th of k algorithms on the i-th of N datasets. Friedman's χ_F^2 is undesirably conservative and derives a better statistic

$$F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}, \tag{51}$$

which is distributed according to the F-distribution with k-1 and $(k-1)(N-1)$ degrees of freedom.

We can obtain $\chi_F^2 = 12.09$ and $F_F = 3.789$ according to (50) and (51), where F_F is distributed according to F-distribution with (4, 40) degrees of freedom. The critical value of $F(4, 40)$ is 2.61 for the significance level $\alpha = 0.05$, and similarly it is 3.13 for $\alpha = 0.025$ and 3.83 for $\alpha = 0.01$. Since the value of F_F is much larger than the critical value, there is significant difference between five algorithms. Note that the average rank of Asy-ν-TSVR is far lower than the remaining algorithms. It means that our Asy-ν-TSVR is more valid than other four algorithms.

5.4.5 Analyze the asymmetry of two datasets

In order to analyze the asymmetry of datasets, the kernel function-based method is used to estimate the p.d.f. of $y - \widehat{f}(x)$. In the following, the p.d.f. of $y - \widehat{f}(x)$ obtained by the ν-SVR and the Asy-ν-TSVR are listed. Where the results of Chwirut dataset are shown in Fig. 3, and Bodyfat dataset in Fig. 4.

Apparently, Figs. 3 and 4 imply that the asymmetry of Bodyfat dataset is not obvious. On the contrary, the Chwirut dataset shows noticeable asymmetry. However, we

Table 4 The average of computational time with different p values for Asy-ν-TSVR with Gaussian kernel

Datasets	p values						
	0.20	0.40	0.45	0.50	0.55	0.60	0.80
Auto Price	0.1671	0.1607	0.1599	*0.1585*	0.1601	0.1606	0.1682
Bodyfat	0.2627	*0.2585*	0.2603	0.2586	0.2608	0.2593	0.2620
Chwirut	0.2471	*0.2459*	0.2468	0.2467	0.2475	0.2471	0.2462
Con. S	0.0990	0.0986	*0.0963*	0.0966	0.0969	0.0983	0.0992
Diabetes	*0.0684*	0.0686	0.0685	0.0690	0.0693	0.0689	0.0686
Machine-Cpu	0.2473	0.2395	0.2399	0.2394	0.2384	*0.2370*	0.2513
Pyrimidibes	*0.0775*	0.0782	0.0785	0.0778	0.0779	0.0783	0.0780
Triazines	0.1952	*0.1922*	0.1933	0.1930	0.1941	0.1933	0.1940
Housing	1.1037	1.0865	1.0855	*1.0714*	1.0817	1.0751	1.1054
Istanbul Stock Exchange	1.5912	1.5817	1.5995	1.6254	*1.5623*	1.5658	1.5717
Yacht Hydrodynamics	0.5089	*0.4865*	0.4880	0.4873	0.4899	0.4938	0.5117

Italic type shows the minimal values

Table 5 The average of MAE with different p values for Asy-ν-SVR and Asy-ν-TSVR with Gaussian kernel

Datasets	Algorithms	p values				
		0.20	0.40	0.50	0.60	0.80
Auto Price	Asy-ν-SVR	6559.3 ± 2080.6	5836.4 ± 2226.6	5563.2 ± 2300.5	5465.9 ± 2340.1	*5366.9 ± 2398.7*
	Asy-ν-TSVR	13825 ± 8597.5	13825 ± 8596.9	13825 ± 8596.8	13825 ± 8597.0	13826 ± 8597.8
Bodyfat	Asy-ν-SVR	0.1354 ± 0.0286	0.1265 ± 0.0274	0.1262 ± 0.0278	0.1261 ± 0.0278	0.1258 ± 0.0281
	Asy-ν-TSVR	0.1077 ± 0.0244	0.1072 ± 0.0245	0.1071 ± 0.0246	*0.1070 ± 0.0247*	0.1072 ± 0.0249
Chwirut	Asy-ν-SVR	0.3266 ± 0.0593	0.3018 ± 0.0479	0.2943 ± 0.0456	0.2901 ± 0.0437	0.2845 ± 0.0394
	Asy-ν-TSVR	0.0538 ± 0.0100	0.0527 ± 0.0106	0.0524 ± 0.0110	*0.0522 ± 0.0113*	*0.0522 ± 0.0120*
Con. S	Asy-ν-SVR	*0.2744 ± 0.0482*	0.2774 ± 0.0463	0.2773 ± 0.0460	0.2784 ± 0.0448	0.2878 ± 0.0377
	Asy-ν-TSVR	0.6615 ± 0.3957	0.6614 ± 0.3966	0.6619 ± 0.3976	0.6625 ± 0.3985	0.6653 ± 0.4018
Diabetes	Asy-ν-SVR	0.5879 ± 0.1485	*0.5824 ± 0.1384*	0.5983 ± 0.1363	0.5992 ± 0.1371	0.6211 ± 0.1547
	Asy-ν-TSVR	1.8246 ± 1.8910	1.7975 ± 1.8574	1.7902 ± 1.8498	1.7842 ± 1.8465	1.7770 ± 1.8467
Machine-Cpu	Asy-ν-SVR	143.0002 ± 20.3020	113.8695 ± 19.4701	110.2614 ± 20.0042	105.8753 ± 22.0350	*100.2744 ± 22.0369*
	Asy-ν-TSVR	550.0271 ± 359.1591	549.1505 ± 358.7061	548.9730 ± 358.6089	548.9139 ± 358.5667	548.9806 ± 358.5524
Pyrimidibes	Asy-ν-SVR	*0.0968 ± 0.0365*	0.0977 ± 0.0361	0.0989 ± 0.0362	0.0994 ± 0.0365	0.1059 ± 0.0379
	Asy-ν-TSVR	0.1471 ± 0.1003	0.1470 ± 0.1004	0.1471 ± 0.1005	0.1473 ± 0.1006	0.1481 ± 0.1013
Triazines	Asy-ν-SVR	0.1226 ± 0.0216	*0.1193 ± 0.0214*	0.1203 ± 0.0227	0.1247 ± 0.0265	0.1510 ± 0.0445
	Asy-ν-TSVR	0.3331 ± 0.1466	0.3324 ± 0.1456	0.3325 ± 0.1454	0.3327 ± 0.1453	0.3339 ± 0.1458
Housing	Asy-ν-SVR	9.6178 ± 0.8573	7.8251 ± 0.9101	7.2803 ± 0.6326	7.0581 ± 0.5623	*6.8911 ± 0.4372*
	Asy-ν-TSVR	7.9811 ± 2.4585	7.9705 ± 2.4465	7.9717 ± 2.4465	7.9752 ± 2.4483	7.9952 ± 2.4621
Istanbul Stock Exchange	Asy-ν-SVR	0.0080 ± 0.0023	0.0080 ± 0.0023	0.0080 ± 0.0023	0.0080 ± 0.0023	0.0080 ± 0.0022
	Asy-ν-TSVR	0.0048 ± 0.0007	*0.0047 ± 0.0006*	*0.0047 ± 0.0006*	*0.0047 ± 0.0006*	0.0048 ± 0.0006
Yacht Hydrodynamics	Asy-ν-SVR	14.5247 ± 3.1103	11.8741 ± 2.4088	11.3315 ± 2.4376	11.0097 ± 2.3522	10.7104 ± 2.3067
	Asy-ν-TSVR	4.9008 ± 0.4203	4.8984 ± 0.4210	4.8973 ± 0.4212	4.8962 ± 0.4216	*4.8931 ± 0.4221*

Italic type shows the minimal values

Table 6 Results of the Wilcoxon signed-rank test between the linear case and nonlinear case on five algorithms

Algorithms	p value	W	h
ν-SVR	0.7114	27	0
Asy-ν-SVR	0.6812	28	0
LS-SVR	0.2158	36	0
TSVR	0.1602	45	0
Asy-ν-TSVR	0.2158	36	0

Table 7 Average ranks of five algorithms with Gaussian kernel on MAE values

Datasets	ν-SVR	Asy-ν-SVR	LS-SVR	TSVR	Asy-ν-TSVR
Auto Price	4.5	4.5	1	2.5	2.5
Bodyfat	5	4	3	2	1
Chwirut	5	4	3	2	1
Con. S	5	4	3	2	1
Diabetes	3	2	5	4	1
Machine-Cpu	5	4	1	3	2
Pyrimidibes	1.5	1.5	3	5	4
Triazines	2	1	3	5	4
Housing	5	4	3	1	2
Istanbul Stock Exchange	5	4	2	2	2
Yacht Hydrodynamics	5	4	3	2	1
Average rank	4.18	3.36	2.73	2.77	1.95

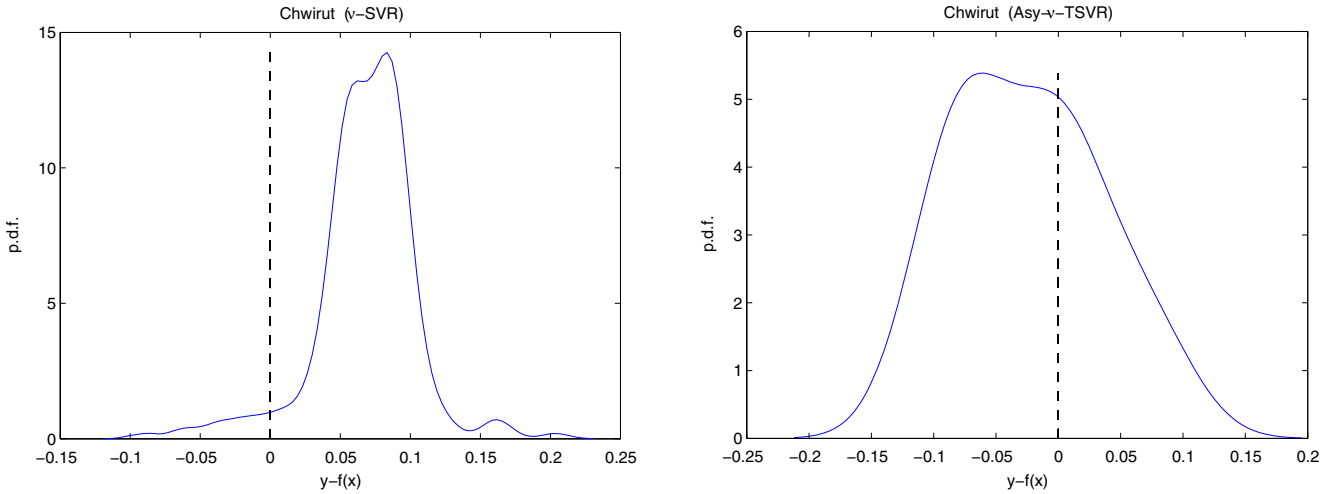

Fig. 3 P.d.f. of y-f(x) with ν-SVR and Asy-ν-TSVR on Chwirut dataset

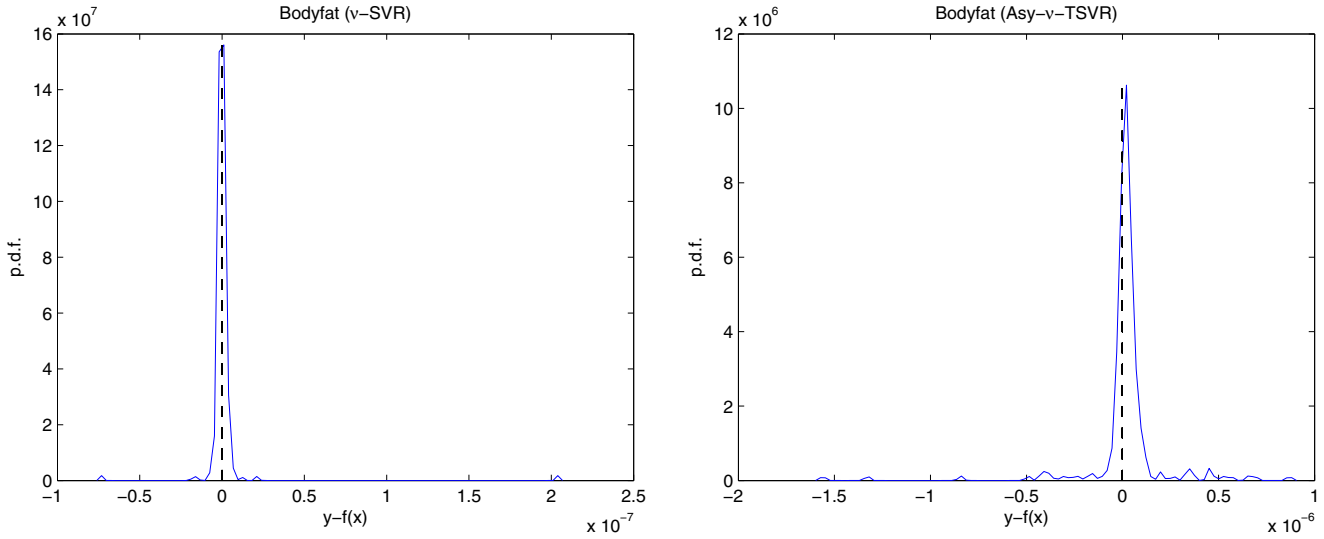

Fig. 4 P.d.f. of y-f(x) with ν-SVR and Asy-ν-TSVR on Bodyfat dataset

Table 8 Performance comparisons of five algorithms with Gaussian kernel on protein content of wheat

Algorithms	MAE	RMSE	SSE/SST	SSR/SST	Time (s)	Optimal parameters
ν-SVR	1.0578 ± 0.2437	1.3467 ± 0.3504	0.9112 ± 0.2903	0.1363 ± 0.062	0.3375	$(C = 128, \epsilon = 0.3, \sigma = 1)$
Asy-ν-SVR	1.0543 ± 0.1998	1.3378 ± 0.306	0.9109 ± 0.2872	0.2309 ± 0.1342	0.3764	$(C = 256, \epsilon = 0.5, \sigma = 1, t = 0.45)$
LS-SVR	1.0153 ± 0.2145	1.2824 ± 0.2802	0.8705 ± 0.3884	0.3931 ± 0.1699	*0.0286*	$(C = 256, \sigma = 4)$
TSVR	0.9227 ± 0.2038	1.1407 ± 0.253	0.7553 ± 0.5073	0.5523 ± 0.167	0.2336	$(C = 4, \epsilon = 0.9, \sigma = 8)$
Asy-ν-TSVR	*0.9122 ± 0.1994*	*1.1402 ± 0.2338*	0.7565 ± 0.4973	0.5274 ± 0.1559	0.213	$(C = 128, \epsilon = 0.8, \sigma = 8, t = 0.8)$

Italic type shows the best result

Table 9 Performance comparisons of five algorithms with Gaussian kernel function on wet gluten of wheat

Algorithms	MAE	RMSE	SSE/SST	SSR/SST	Time (s)	Optimal parameters
ν-SVR	2.9378 ± 0.3981	3.7859 ± 0.6028	0.7974 ± 0.0921	0.1812 ± 0.0601	0.6105	$(C = 256, \epsilon = 0.5, \sigma = 1)$
Asy-ν-SVR	2.9048 ± 0.4308	3.7764 ± 0.6250	0.7954 ± 0.1206	0.2449 ± 0.1321	0.4844	$(C = 256, \epsilon = 0.8, \sigma, t = 0.8)$
LS-SVR	2.5220 ± 0.2562	3.2826 ± 0.3478	0.6114 ± 0.1204	0.5026 ± 0.2327	*0.0356*	$(C = 256, \sigma = 2)$
TSVR	2.3029 ± 0.2618	3.0135 ± 0.2923	0.5188 ± 0.1130	0.6774 ± 0.2436	0.2640	$(C = 4, \epsilon = 0.9, \sigma = 4)$
Asy-ν-TSVR	*2.2957 ± 0.2591*	3.0208 ± 0.3030	0.5204 ± 0.1097	0.6533 ± 0.2506	0.1684	$(C = 256, \epsilon = 0.6, \sigma = 4, t = 0.8)$

Italic type shows the best result

can find from Tables 2 and 3 that our proposed algorithm Asy-ν-TSVR outperforms the rest algorithms on these two datasets. This is because the pinball loss function enhances the generalization ability of our Asy-ν-TSVR for Asymmetric datasets. And the symmetric datasets can be regarded as the special case of the Asymmetric datasets.

5.5 Experiment on wheat dataset

There are 210 wheat samples from all over China in this real data experiments [34]. The protein content of wheat ranges from 9.83 to 20.26%, and the wet gluten of wheat ranges from 14.8 to 44.6%. They are provided by Heilongjiang research institute of agricultural science. Each sample has 1193 spectral features. They were scanned in transmission mode using a commercial spectrometer MATRIX-I. Samples were acquired in a rectangular quartz cuvette of 1-mm path length with air as reference at room temperature (20–24 °C). The reference spectrum was subtracted from the sample spectra to remove background noise. The rectangular quartz cuvette was cleaned after each sample was scanned to minimize cross-contamination.

In this high-dimensional data experiment, we predict the content of protein and the wet gluten of wheat using 1193 spectral features of wheat. Now that Gaussian kernel yields better generalization performance than linear kernel, we only consider the former, the prediction errors of five algorithms are summed in Tables 8 and 9.

From Tables 8 and 9, we can learn that Asy-ν-TSVR yields the lowest prediction errors (0.9122% for protein content and 2.2957% for wet gluten) among five algorithms. The prediction errors of TSVR follow. LS-SVR outperforms ν-SVR and Asy-ν-SVR either on the prediction of protein content or the prediction of wet gluten. ν-SVR and Asy-ν-SVR produce the comparable testing errors. In addition, Asy-ν-SVR produces lightly lower testing errors than ν-SVR. It implies that the pinball loss is effective in ν-SVR and TSVR. In terms of the computational time, LS-SVR costs the shortest running time since it solves a system of linear equations, but other four algorithms solve a pair of smaller-sized QPPs or a larger-sized QPP.

6 Conclusion

In this paper, we propose an Asy-ν-TSVR for the regression problem. Asy-ν-TSVR solves a pair of smaller-sized QPPs instead of a larger-sized one as in the traditional ν-SVR. So it works faster than Asy-ν-SVR. Asy-ν-TSVR employs the pinball loss function $L_\epsilon^p(u)$ as opposed to the ϵ-insensitive loss function $L_\epsilon(u)$; then, it can effectively reduce the disturbance of the noise and improve the generalization performance. Three kinds of experiments demonstrate the validity of our proposed Asy-ν-TSVR. Asy-ν-TSVR degrades into the ν-TSVR when $p = 0.5$, so our Asy-ν-TSVR is an extended version of the ν-TSVR, and it is applicable to the symmetric and asymmetric datasets. How to apply the pinball loss function to other TSVMs is our future work.

Acknowledgments The authors gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation. This work was supported in part by the National Natural Science Foundation of China (No. 11671010) and Natural Science Foundation of Beijing Municipality (No. 4172035).

Compliance with ethical standards

Conflict of interests The authors declare that they have no conflict of interest.

References

1. Steinwart I, Christmann A (2008) Support vector machines. Springer, New York
2. Vapnik V (1995) The nature of statistical learning theory. Springer, New York
3. Shawe-Taylor J, Cristianini N (2004) Kernel methods for pattern analysis. Cambridge University Press

4. Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press
5. Schölkopf B, Smola A, Williamson RC, Bartlett PL (2000) New support vector algorithms. Neural Comput 12:1207–1245
6. Bi J, Bennett KP (2003) A geometric approach to support vector regression. Neurocomputing 55:79–108
7. Peng XJ (2010) TSVR: an efficient twin support vector machine for regression. Neural Netw 23:365–372
8. Peng XJ (2012) Efficient twin parametric insensitive support vector regression model. Neurocomputing 79:26–38
9. Peng XJ, Xu D, Shen JD (2014) A twin projection support vector machine for data regression. Neurocomputing 138:131–141
10. Santanu G, Mukherjee A, Dutta PK (2009) Nonparallel plane proximal classifier. Signal Process 89:510–522
11. Jayadeva KR, Chandra S (2007) Khemchandani Twin support vector machines for pattern classification. IEEE Trans Pattern Anal Mach Intell 29:905–910
12. Xu YT, Xi WW, Lv X (2012) An improved least squares twin support vector machine. J Info Comput Sci 9:1063–1071
13. Singh M, Chadha J, Ahuja P, Jayadeva S (2011) Chandra Reduced twin support vector regression. Neurocomputing 74:1474–1477
14. Zhao YP, Zhao J, Zhao M (2013) Twin least squares support vector regression. Neurocomputing 118:225–236
15. Kumar MA, Gopal M (2009) Least squares twin support vector machines for pattern classification. Expert Syst Appl 36:7535–7543
16. Tomar D, Agarwal S (2015) Twin Support Vector Machine: a review from 2007 to 2014. Egyptian Info J 16:55–69
17. Huang XL, Shi L, Suykens JAK (2014) Support vector machine classifier with pinball loss. IEEE Trans Pattern Anal Mach Intell 36:984–997
18. Huang XL, Shi L, Pelckmans K, Suykens JAK (2014) Asymmetric ν-tube support vector regression. Comput Stat Data Anal 77:371–382
19. Xu YT, Yang ZJ, Zhang YQ, Pan XL, Wang LS (2016) A maximum margin and minimum volume hyper-spheres machine with pinball loss for imbalanced data classification. Knowl-Based Syst 95:75–85
20. Xu YT, Yang ZJ, Pan XL (2016) A novel twin support vector machine with pinball loss. IEEE Trans Neural Netw Learn Syst 28(2):359–370
21. Hao PY (2010) New support vector algorithms with parametric insensitive/margin model. Neural Netw 23:60–73
22. Le Masne Q, Pothier H, Birge NO, Urbina C, Esteve D (2009) Asymmetric noise probed with a josephson junction. Phys Rev Lett 102:067002
23. Yu K, Moyeed RA (2001) Bayesian quantile regression. Stat Prob Lett 54:437–447
24. Sengupta RN (2008) Use of asymmetric loss functions in sequential estimation problems for multiple linear regression. J Appl Stat 35:245–261
25. Xu YT, Guo R (2014) An improved ν-twin support vector machine. Appl Intell 41:42–54
26. Xu YT, Wang L, Zhong P (2012) A rough margin-based ν-twin support vector machine. Neural Comput Applic 21:1307–1317
27. Steinwart I, Christmann A (2011) Estimating conditional quantiles with the help of the pinball loss. Bernoulli 17:211–225
28. Suykens JAK, Tony VG, Jos DB et al (2002) Least squares support vector machines. World Scientific Pub Co, Singapore
29. Xu YT (2012) A rough margin-based linear ν support vector regression. Stat Prob Lett 82:528–534
30. Xu YT, Wang LS (2012) A weighted twin support vector regression. Knowl-Based Syst 33:92–101
31. Navia-Vzquez F, Prez-Cruz A, Arts-Rodrguezand A, Figueiras-Vidal R (2001) Weighted least squares training of support vectors classifiers which leads to compact and adaptive schemes. IEEE Trans Neural Netw 12(5):1047–1059
32. Prez-Cruz J, Herrmann DJL, Scholkopf B (2003) Weston Weston Extension of the nu-SVM range for classification. In: Prez-Cruz J, Herrmann DJL, Scholkopf B (eds) Advances in learning theory: methods, models and applications. IOS Press, pp 179–196
33. Scholkopf B, Smola A, Bartlett P, Williamson R (2000) New support vector algorithms. Neural Comput 12(5):1207–1245
34. Xu YT, Wang LS (2014) k-nearest neighbor-based weighted twin support vector regression. Appl Intell 41:299–309
35. Gibbons JD, Chakraborti S (2011) Nonparametric statistical inference, 5th Ed. Chapman & Hall CRC Press, Taylor & Francis Group, Boca Raton
36. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
37. García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining. Experimental analysis of power. Info Sci 180:2044–2064