Sigma J Eng Nat Sci, Vol. 42, No. 2, pp. 578–589, April, 2024

Sigma Journal of Engineering and Natural Sciences
Web page info: https://sigma.yildiz.edu.tr
DOI: 10.14744/sigma.2024.00045

Research Article

Comparative performance analysis of epsilon-insensitive and pruning-based algorithms for sparse least squares support vector regression

Ömer KARAL[1,*]

[1]Department of Electrical and Electronics Engineering, Ankara Yildirim Beyazit University, Ankara, 06760, Türkiye

ARTICLE INFO

Article history
Received: 06 May 2022
Revised: 28 June 2022
Accepted: 12 October 2022

Keywords:
Least Squares Support Vector Regression; Pruning; Epsilon Insensitiveness; Robustness; Sparseness

ABSTRACT

Least Squares Support Vector Regression (LSSVR) which is a least squares version of the Support Vector Regression (SVR) is defined with a regularized squared loss without epsilon-insensitiveness. LSSVR is formulated in the dual space as a linear equality constrained quadratic minimization which can be transformed into solution of a linear algebraic equation system. As a consequence of this system where the number of Lagrange multipliers is half that of classical SVR, LSSVR has much less time consumption compared to the classical SVR. Despite this computationally attractive feature, it lacks the sparsity characteristic of SVR due to epsilon-insensitiveness. In LSSVR, every (training) input data is treated as a support vector, yielding extremely poor generalization performance. To overcome these drawbacks, the epsilon-insensitive LSSVR with epsilon-insensitivity at quadratic loss, in which sparsity is directly controlled by the epsilon parameter, is derived in this paper. Since the quadratic loss is sensitive to outliers, its weighted version (epsilon insensitive WLSSVR) has also been developed. Finally, the performances of epsilon-insensitive LSSVR and epsilon-insensitive WLSSVR are quantitatively compared in detail with those commonly used in the literature, pruning-based LSSVR and weighted pruning-based LSSVR. Experimental results on simulated and 8 different real-life data show that epsilon-insensitive LSSVR and epsilon-insensitive WLSSVR are superior in terms of computation time, generalization ability, and sparsity.

Cite this article as: Karal Ö. Comparative performance analysis of epsilon-insensitive and pruning-based algorithms for sparse least squares support vector regression. Sigma J Eng Nat Sci 2024;42(2):578–589.

INTRODUCTION

Support Vector Machines (SVMs), a machine learning method, were initially introduced by Vapnik in 1995 [1]. They were quickly recognized as an effective tool in classification and regression tasks and have since found widespread application in various real-world scenarios [2-7]. Within the realm of regression, support vector machines are called Support Vector Regression (SVR) [8, 9]. SVR uses the ε-insensitive l_1 loss function, which disregards noise while trying to suppress the influence of outliers, providing better generalization ability compared to least squares regression. Similar to the ridge regression, which minimizes a regularized l_2 loss, SVR is adept at constructing models with enhanced

*Corresponding author.
*E-mail address: omerkaral@aybu.edu.tr

This paper was recommended for publication in revised form by Regional Editor Md. Sabir Hossain

generalization abilities. This is attained by minimizing both regularized and ε-insensitive empirical error.

The optimal parameters for an SVR model are typically determined by minimizing the convex quadratic cost function formulated in the dual space. This involves utilizing Lagrange multipliers and a kernel to achieve the optimal solution. While minimizing convex quadratic costs is efficient, dual representations for classical SVR encounter significant time consumption due to the substantial number of optimization variables, especially when dealing with large datasets. To address this, a more computationally efficient alternative, known as the Least Squares Support Vector Regression (LSSVR), has been presented [10]. The formulation of LSSVR resembles ridge regression, using a regularized l_2 loss, where experimental errors are treated as linear constraints [11]. Typically, LSSVR is formulated as a quadratic minimization problem subject to a linear equality constraint on Lagrange multipliers in the dual space.

A significant advantage of LSSVR over SVR is the possibility of having the ability to represent the primary cost constraint as a penalty term in the dual formulation representation using a single Lagrange multiplier, as opposed to the pair of Lagrange multipliers required for training samples in SVR. This formulation results in LSSVR requiring only half the Lagrange multipliers required for classical SVR compared to classical SVR, leading to a remarkable reduction in calculation time.

In the LSSVR approach, the loss function used is the regularized squared (l_2) loss; The error terms are symbolized as the equality constraints, resulting in the formulation of a linear system of equations. Though this feature offers computational advantages, the sparsity property inherent in traditional SVR, induced through ε-insensitivity, is vanished in LSSVR, where each input instance is treated as a support vector. Moreover, due to the squared loss used, LSSVR lacks robustness to outliers compared to SVR.

To deal with the lack of sparseness for LSSVR, current methods can be classified in two approaches: iterative and direct methods. In iterative methods, training samples are progressively eliminated either forward or backward, one by one, in each iteration. For instance, Suykens et al. introduced the Pruned LSSVR (PLSSVR) model, which initially runs the standard LSSVR model and sort the resulting support vectors from largest to smallest. Then, it gradually prunes the support vectors beginning from the smallest value in the spectrum [12]. Additionally, they proposed a weighted version to enhance robustness against outliers [13]. Kruif and Vries proposed an alternative pruning algorithm where the training sample yielding the smallest error after its ignorance in the previous iteration is removed [14]. Kuh and De Wilde [15] extended the Kruif and Vries's pruning algorithm which is applied to a non-regularized loss to the regularized loss. Hoegaerts et al. [16] presented two pruning algorithms: one is based on deleting the sample with the smallest correlation with the output and the other on removing the sample with the least similarity to the best fitting span. Zeng and Chen [17] introduced the SMO-based pruning scheme, which eliminates samples that contribute the least change in the dual objective function, rather than being merely on errors.

Zhao and Sun [18] presented a technique called recursive reduced LSSVR (RRLSSVR), where data contributing more to the objective function are selected as support vectors while considering all constraints yielded by all training samples. Subsequently, the improved version of RRLSSVR (IRRLSSVR) was introduced to get much sparser solution than RLLSSVR in [19]. Later, refined versions of them [20] were proposed to improve their performance. Si et al. [21] introduced the reconstructed LSSVR algorithm (RCLSSVR), applied in mill load prediction, which utilizes reconstructed support vectors. It selects reconstructed data based on density clustering information in the training dataset and to improve sparseness and robustness simultaneously. Sun et al. [22] proposed a localized generalization error model based on the training mean square error and sensitivity measure to prune support vectors in the LS-SVM.

In direct methods, the algorithm begins with a full dense solution and then eliminates training samples based on objective criteria. For instance, Espinoza et al. [23] introduced a fixed-size least squares least squares support vector machine (FS-LSSVM) method, utilizing Nystrm approximation with a predefined set of prototype vectors (PVs) to provide a solution in the primal space. Based on the similar idea, Mall and Suykens [24] proposed two L0-norm-reduced models: the sparsified primal FS-LSSVM for the input space and sparsified subsampled dual LSSVM for the dual space. Yang et al. [25] introduced a one-step compressive pruning strategy to construct a sparse LSSVM. Zhou [26] introduced a low-rank representation technique using pivoted Cholesky decomposition for the kernel matrix to sparsify the LSSVM. Later, this method was extended to a robust LSSVM using a non-convex truncated loss function [27]. Xia [28] used the Kernel Matching Tracking technique, which exploits the number of support vectors as the regularization parameter to achieve sparsity in the LS-SVM solution. Ma et al. [29] designed an indicator to assess the global representation of data points based on density and distribution of them in the feature space. Next, they presented a fast sparse LS-SVM method by choosing support vectors with a non-recursive strategy using global representation.

Recently, a new sparse LSSVR model (ε-LSSVR) with ε-insensitivity at quadratic loss has been introduced in [30]. ε-LSSVR ignores errors within a given ε band, as in SVM, and its sparsity is controlled only by the ε parameter. Inspired by [30], this paper theoretically derives ε-LSSVR from the LSSVR model in detail, which results in fewer support vectors that provide sparsity in dual space without necessitating computationally expensive algorithms. However, the quadratic loss function of ε-LSSVR may comprise robustness in the presence of outliers. To address this limitation and improve robustness, a weighted version, ε-WLSSVR, will be presented in Section 2.2, for the specific case of this study.

The pruning approach iteratively removes the constraints of non-support vectors backwards to build a sparse LSSVR model. This may result in better convergence and higher stability for inhomogeneous and unbalanced datasets. Conversely, the ε-insensitive strategy forces certain support vectors to move to zero, allowing direct control over the selection of non-zero support vectors, which directly influences solution sparsity and computation time.

In this study, the performances of iterative based PLSSVR, WPLSSVR algorithms and ε-LSSVR and ε-WLSSVR methods that provide direct sparsity are analyzed for the first time in terms of generalization ability, sparsity, and computation time on both synthetic and 8 different real-world data.

The main contributions of this paper are summarized as follows:

i. Directly defining the sparseness of LSSVR in the input space by using ε-insensitivity within the quadratic loss function and providing a theoretical solution.
ii. Addressing the robustness matter inherent in the quadratic loss function of ε-LSSVR by introducing its weighted version, ε-WLSSVR.
iii. Analyzing the performances of PLSSVR, WPLSSVR, ε-LSSVR, and ε-WLSSVR methods for the first time in terms of sparsity, generalization ability, and computation time across both synthetic and 8 different real-world datasets.

The remainder of the paper is structured as follows: Section 2 presents a review of the basic concepts of LSSVR followed by a full portrait of the four methods of interest. Section 3 examines and compares pruning-based, and ε-insensitivity based approaches in both synthetic and real-world datasets. Finally, Section 4 presents result descriptions and potential feature guidelines.

Least Squares Support Vector Regression

Given a training set $\{(\mathbf{x}_s, y_s)\}_{s=1}^{L}$, where $\mathbf{x}_s$, represents the sth input data vector, y_s denotes the target output data points for the input $\mathbf{x}_s$, and L is the number of training data points, LSSVR is formulated in primal space as follows [10]:

$$\min_{\substack{w\in R^n,\\ b\in R}} \frac{1}{2}\| \boldsymbol{w} \|_2^2 + \frac{C}{2}\sum_{s-1}^{L}(e_s)^2 \tag{1}$$

$$\text{subject to } e_s = y_s - (\boldsymbol{w}^T \boldsymbol{\varphi}(\mathbf{x}_s) + b),\ s \in \{1,\ldots,L\} \tag{2}$$

Where, C is a user defined regularization constant that controls the balance between empirical error (for large C) and generalization ability (for small C), $\boldsymbol{w}$ is the unknown model parameter, e_s representing the deviation of the actual output from the predicted output for each training example, $\varphi(\cdot)$ is a nonlinear basis function, and b is the unknown threshold parameter. LSSVR optimization problem (1) in primal space is transformed into the following unconstrained optimization problem in dual space by applying the Lagrange multipliers method.

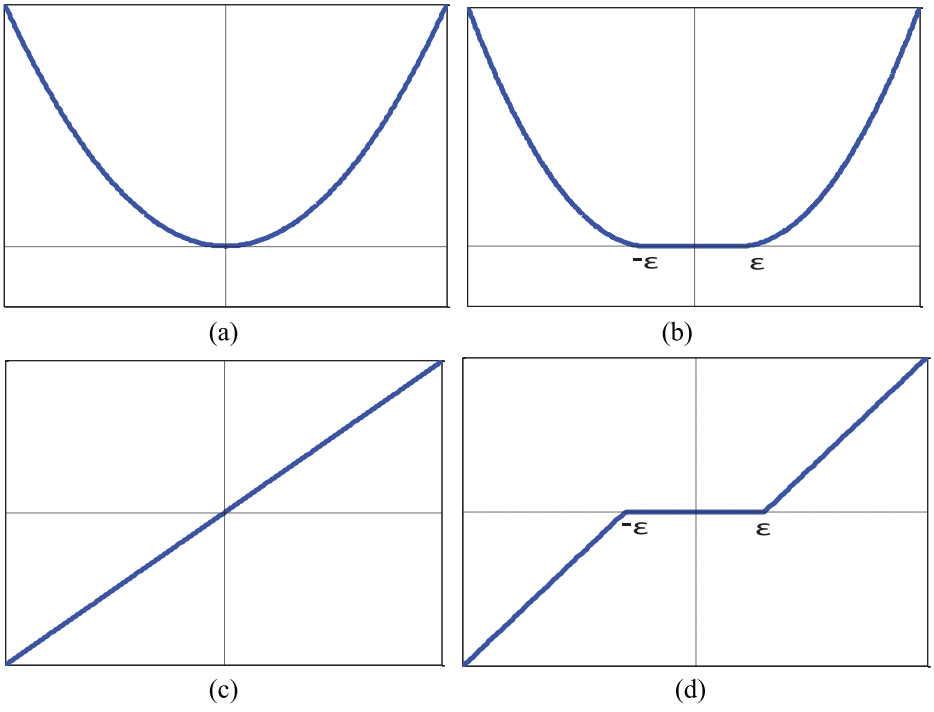

Figure 1. a) Loss function of LSSVR b) ε-insensitivity in loss function of LSSVR c) Derivative of LSSVR loss function d) Derivative of LSSVR loss function with ε-insensitivity.

$$\min_{\alpha\in R^L} J(\alpha_s)=\frac{1}{2}\sum_{s=1}^{L}\sum_{r=1}^{L}\alpha_s\mathbf{K}(\mathbf{x}_s,\mathbf{x}_r)\alpha_r+\frac{1}{2C}\sum_{s=1}^{L}{\alpha_s}^2-\sum_{s=1}^{L}\alpha_s(y_s-b) \quad (3)$$

Where:

- α_s represents the Lagrange multipliers.
- $\mathbf{K}(\mathbf{x}_s, \mathbf{x}_r)$ represents kernel functions characterized as the inner product of two input samples $\varphi(\mathbf{x}_s)$ and $\varphi(\mathbf{x}_r)$ in the high dimensional space i.e. $[\mathbf{K}(\mathbf{x}_s,\mathbf{x}_r)]_{s,r}=[\boldsymbol{\varphi}^{\mathrm{T}}(\mathbf{x}_s)\ \boldsymbol{\varphi}(\mathbf{x}_r)]_{s,r}\in R^{L\times L}$.

The predicted range of learned LSSVR for the test sample **x,** can be expressed as follows:

$$y=f(\mathbf{x})=\boldsymbol{w}^T\boldsymbol{\varphi}(\mathbf{x})+b=\sum_{s=1}^{L}\alpha_s\boldsymbol{\varphi}(\mathbf{x}_s)^T\boldsymbol{\varphi}(\mathbf{x})+b=\sum_{s=1}^{L}\alpha_s\mathrm{K}(\mathbf{x}_s,\mathbf{x})+b \quad (4)$$

ε-insensitive Least Squares Support Vector Regression

ε-LSSVR, the ε-insensitive variant of LSSVR, is introduced in detail in this section. According to Equation (4), every training data point $\mathbf{x}_s$ contributes to the solution representation of model except training data points with α_s = 0. Therefore, the significance of a training data point $\mathbf{x}_s$ is determined by its support value α_s. The values of α_s, which are Lagrange multipliers, rarely equal zero in most practical scenarios, leading to numerous support vectors in the LSSVR solution representation. The optimization problem required to derive ε-LSSVR from LSSVR is formulated as follows [30]:

$$\min_{\substack{w\in R^n,\\ b\in R}} J(\boldsymbol{w},e)=\frac{1}{2}\|\boldsymbol{w}\|_2^2+\frac{C}{2}\sum_{s=1}^{L}(e_s)_\varepsilon^2 \quad (5)$$

$$\text{subject to } e_s=y_s-(\boldsymbol{w}^T\boldsymbol{\varphi}(\mathbf{x}_s)+b),\ s\in\{1,\ldots,L\} \quad (6)$$

where, the $(e)_\varepsilon^2$ is defined as:

$$(e)_\varepsilon^2=\begin{cases}(e-\varepsilon)^2 & if\ e\geq\varepsilon\\ 0 & if\ -\varepsilon\leq e\leq\varepsilon\\ (e+\varepsilon)^2 & if\ e\leq-\varepsilon\end{cases} \quad (7)$$

It's important to note that $(e)_\varepsilon^2$ is a continuously differentiable function. Figure 1 illustrates the LSSVR loss function and its derivative in comparison, as well as the ε-insensitivity in loss function of LSSVR and its derivative.

The canonical representation of the first derivative of the ε-LSSVR loss function is as follows:

$$\frac{\partial(e)_\varepsilon^2}{\partial e}=2e+|e-\varepsilon|-|e+\varepsilon| \quad (8)$$

The formulation model in terms of Lagrange multipliers becomes:

$$\max_{\alpha\in R^L}\min_{\substack{w\in R^m,\\ b\in R}} J(\boldsymbol{w},e;\alpha_s)=\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w}+\frac{C}{2}\sum_{s=1}^{L}(e_s)_\varepsilon^2-\sum_{s=1}^{L}\alpha_s(\boldsymbol{w}^T\boldsymbol{\varphi}(\mathbf{x}_s)+b+e_s-y_s) \quad (9)$$

The conditions required for the optimal solution of the Lagrangian formulation model (9) are as follows:

$$\frac{\partial J}{\partial \boldsymbol{w}}=0\rightarrow\boldsymbol{w}=\sum_{s=1}^{L}\alpha_s\boldsymbol{\varphi}(\mathbf{x}_s) \quad (10)$$

$$\frac{\partial J}{\partial b}=0\rightarrow\sum_{s=1}^{L}\alpha_s=0 \quad (11)$$

$$\frac{\partial J}{\partial e_s}=0\rightarrow\frac{C}{2}[2e_s+|e_s-\varepsilon|-|e_s+\varepsilon|]=\alpha_s \quad (12)$$

$$\frac{\partial J}{\partial \alpha_s}=0\rightarrow\boldsymbol{w}^T\boldsymbol{\varphi}(\mathbf{x}_s)+b+e_s-y_s=0 \quad (13)$$

To derive a cost function based on the Lagrange multipliers α_s it is necessary to solve for e_s in terms of α_s in equation (12). There are 3 distinct regions, each of which establishes a relationship between α_s and e_s in an affine manner:

$$e_s=\begin{cases}\frac{1}{C}\alpha_s+\varepsilon, & if\ e_s>\varepsilon\\ 0, & if\ -\varepsilon\leq e_s\leq\varepsilon\\ \frac{1}{C}\alpha_s-\varepsilon & if\ e_s<-\varepsilon.\end{cases}$$

Since $e_s>\varepsilon$ implies $a_s>0$, $-\varepsilon\leq e_s\leq\varepsilon$ implies $a_s=0$, and $e_s<-\varepsilon$ implies $a_s<0$ equation (12) can be written as

$$e_s=\frac{1}{C}\alpha_s+\varepsilon\cdot sign^\star(\alpha_s) \quad (14)$$

where,

$$sign^\star(\alpha_s)=\begin{cases}1 & if\ \alpha_s>0\\ \lambda\ with\ \lambda\in[-1,1] & if\ \alpha_s=0\\ -1 & if\ \alpha_s<0\end{cases}$$

Substituting equations (10) and (14) into equation (9) and using equation (11) to eliminate variables e_s and $\boldsymbol{w}$, the optimization formulation of ε-LSSVR becomes:

$$\max_{\alpha\in R^L} J(\alpha_s)=-\frac{1}{2}\sum_{s=1}^{L}\sum_{r=1}^{L}\alpha_s\boldsymbol{\varphi}(\mathbf{x}_s)^T\boldsymbol{\varphi}(\mathbf{x}_r)\alpha_r+\sum_{s=1}^{L}\alpha_s y_s+\frac{C}{2}\sum_{s=1}^{L}\left(\frac{1}{C}\alpha_s+\varepsilon\cdot sign^\star(\alpha_s)\right)_\varepsilon^2-\sum_{s=1}^{L}\alpha_s\left(\frac{1}{C}\alpha_s+\varepsilon\cdot sign^\star(\alpha_s)\right) \quad (15)$$

$$\text{subject to}\sum_{s=1}^{L}\alpha_s=0 \quad (16)$$

The, the last term, in equation (15), $\alpha_s[\varepsilon \cdot sign^*(\alpha_s)]$, is equal to $\varepsilon \cdot |\alpha_s|$. Similarly, third term, $(\cdot)_{\varepsilon}^2$ refers to the ε-insensitive squared loss function and is equal to $\frac{1}{2C}(\alpha_s)^2$. Under these conditions, equation (15) can be rearranged as follows to compactly represent the dual minimization for ε-LSSVR.

$$\min_{a \in R^L} J(\alpha) = \frac{1}{2}\sum_{s=1}^{L}\sum_{r=1}^{L}\alpha_s \mathrm{K}(\mathbf{x}_s, \mathbf{x}_r)\alpha_r - \sum_{s=1}^{L} y_s \alpha_s + \frac{1}{2C}\sum_{s=1}^{L}(\alpha_s)^2 + \varepsilon\sum_{s=1}^{L}|\alpha_s| \quad (17)$$

$$\text{subject to} \sum_{s=1}^{L}\alpha_s = 0 \quad (18)$$

The only difference observed when comparing equation (3) with equation (17), is that the last term, $\varepsilon\sum_{s=1}^{L}|\alpha_s|$ appears because $b\sum_{s=1}^{L}(\alpha_s) = 0$ in the ε-LSSVR optimization problem of equation (17), does not seen in LSSVR model (8). In other words, when ε = 0, classical LSSVR is obtained as a particular scenario of ε-LSSVR. The ε-LSSVR optimization problem can be solved using any convex algorithm that doesn't necessitate taking derivatives of the variables in the optimization formulation. Finally, the *b* parameter can be calculated using equation (13).

ε-insensitive Weighted Least Squares Support Vector Regression

To increase robustness against outliers, it is a common practice in the literature to employ weighting techniques for training data points [13]. However, while ε-LSSVR suppresses noise compared to classical LSSVR, its performance against outliers remains suboptimal owing to the inherent quadratic error characteristic shared like LSSVR. In this section, we apply the weighting technique to ε-insensitive LSSVR to improve its performance against outliers, resulting in the following optimization formulation.

$$\min_{\substack{\mathbf{w}^* \in R^n, \\ b^* \in R}} J(\mathbf{w}^*, e^*) = \frac{1}{2}\| \mathbf{w}^* \|_2^2 + \frac{C}{2}\sum_{s=1}^{L}\delta_s (e_s^*)_{\varepsilon}^2 \quad (19)$$

$$\text{subject to } e_s^* = y_s - ((\mathbf{w}^*)^T \boldsymbol{\varphi}(\mathbf{x}_s) + b^*),\ s \in \{1, \ldots, L\} \quad (20)$$

In this formulation, the parameters δs are employed to weight the influence of errors relative to data points on the loss function. The * symbol distinguishes optimization variables from those of ε-LSSVR. Initially, the δs remain constant in the initial execute of ε-WLSSVR and are subsequently determined as described in equation (24) to calibrate the effects of data point errors based on their distribution. Employing a derivation akin to that in Section 2, we obtain the ε-LSSVR optimization regarding Lagrange multipliers as follows.

$$\max_{\alpha^* \in R^L} \min_{\substack{\mathbf{w}^* \in R^m, \\ b^* \in R}} J(\mathbf{w}^*, e^*, b^*; \alpha_s^*) = \frac{1}{2}(\mathbf{w}^*)^T \mathbf{w}^* + \frac{C}{2}\sum_{s=1}^{L}\delta_s (e_s^*)_{\varepsilon}^2 - \sum_{s=1}^{L}\alpha_s^*((\mathbf{w}^*)^T \boldsymbol{\varphi}(\mathbf{x}_s) + b^* + e_s^* - y_s) \quad (21)$$

By employing optimality conditions (first derivative with respect to the optimization variables) and removing e_s^* and $\mathbf{w}^*$ variables to solve the optimization problem in (21), the ε-WLSSVR optimization problem can be expressed in dual space as follows:

$$\min_{\alpha_s^* \in R^L} J(\alpha_s^*) = \frac{1}{2}\sum_{s=1}^{L}\sum_{r=1}^{L}\alpha_s^* \mathrm{K}(\mathbf{x}_s, \mathbf{x}_r)\alpha_r^* - \sum_{s=1}^{L} y_s \alpha_s^* + \frac{1}{2C}\sum_{s=1}^{L}\frac{1}{\delta_s}(\alpha_s^*)^2 + \varepsilon\sum_{s=1}^{L}|\alpha_s^*| \quad (22)$$

$$\text{subject to} \sum_{s=1}^{L}\alpha_s^* = 0 \quad (23)$$

The robustness of the developed ε-WLSSVR model against outliers is attained by appropriately selecting the δ_s weights based on the relationship delineated in [13].

$$\delta_s = \begin{cases} 1 & if\ |e_s / \hat{\rho}| \le c_1 \\ \dfrac{c_2 - |e_s / \hat{\rho}|}{c_2 - c_1} & if\ c_1 \le |e_s / \hat{\rho}| \le c_2 \\ 10^{-4} & \text{otherwise} \end{cases} \quad (24)$$

where the parameter $\hat{\rho}$ is defined regarding Inter Quartile Range (IQR) denoting the difference between the the 25th percentile (lower quartile) and 75th percentile (upper quartile) of the error distribution [13].

$$\hat{\rho} = \frac{IQR}{2 \times 0.6745} \quad (25)$$

The choices of c_1 = 2.5 and c_2 = 3 are recognized as appropriate for a Gaussian error distribution.

The minimization problem of ε-WLSSVR, as described by Equations (22) to (23), is convex, and therefore any convex algorithm from the existing literature can be employed for its solution, similar to ε-LSSVR.

PRUNED LEAST SQUARES SUPPORT VECTOR REGRESSION

As can be seen from (4), the decision hyperplane of LSSVR contains all the data in the training dataset. This means that LSSVR loses sparsity. The pruning approach for sparse LSSVR aims to obtain a sparse decision hyperplane with fewer data. For this, non-support vectors are extracted recursively from the training dataset according to some specified criteria (error rate, number of support vectors, etc.). First, an initial model is built based on the

entire training dataset and a spectrum of support vectors is plotted. The data that contributes the least to the model are omitted gradually. The reduced LSSVR is rebuilt with the remaining data. These processes are continued until the desired error value is reached [12], which is described in algorithm 1.

Algorithm 1. PLSSVR

1. Train LSSVR (3) based on L training samples
2. Omit a small amount of training samples (5% of the training set) with the smallest value in the sorted support vector spectrum.
3. Re-train the LSSVR based on the reduced training set.
4. Go to step 2, unless the user specified performance index degrades.

WEIGHTED PRUNED LEAST SQUARES SUPPORT VECTOR REGRESSION

The weighted PLSSVR minimization problem in dual space is as follows [13].

$$\min_{\alpha_s^* \in R^L} J(\alpha_s^*) = \frac{1}{2}\sum_{s=1}^{L}\sum_{r=1}^{L}\alpha_s^* \mathrm{K}(\mathbf{x}_s, \mathbf{x}_r)\alpha_r^* - \sum_{s=1}^{L} y_s \alpha_s^* + \frac{1}{2C}\sum_{s=1}^{L}\frac{1}{\delta_s}(\alpha_s^*)^2 \quad (26)$$

$$\text{subject to} \sum_{s=1}^{L}\alpha_s^* = 0$$

The difference between PLSSVR and WPLSSVR is the δ_s parameter. The pruning approach is performed in WPLSSVR as follows [13].

Algorithm 2. WPLSSVR

1. Set $L = Ltot$ equal to the number of training samples.
2. Given $Ltot$ training samples, find an optimal combination (kernel parameter and C) by solving (3).
3. Compute $\hat{\rho}$ from the error distribution.
4. Determine the weights δ_s based upon e_s and $\hat{\rho}$
5. Solve the WLSSVR (equation 26) with respect to $y(x) = \sum_{s=1}^{L}\alpha_s^* \mathrm{K}(\mathbf{x}_s, \mathbf{x}) + b^*$
6. Sort the support values, α_s^*
7. Delete a small amount of N sample points (5% of the $Ltot$ samples) that have the smallest values in the sorted spectrum.

Table 1. Optimization formulations for PLSSVR, WPLSSVR, ε-LSSVR, ε-WLSSVR, and LSSVR models in dual space representations

Models	Optimization formulations in dual space
LSSVR	$\min_{\alpha \in R^L} J(\alpha_s) = \frac{1}{2}\sum_{s=1}^{L}\sum_{r=1}^{L}\alpha_s \mathbf{K}(\mathbf{x}_s, \mathbf{x}_r)\alpha_r + \frac{1}{2C}\sum_{s=1}^{L}\alpha_s^2 - \sum_{s=1}^{L} y_s \alpha_s$ subject to $\left\{\sum_{s=1}^{L}(\alpha_s) = 0,\ \alpha_s \in R^L\right.$
ε-LSSVR	$\min_{\alpha \in R^L} J(\alpha) = \frac{1}{2}\sum_{s=1}^{L}\sum_{r=1}^{L}\alpha_s \mathbf{K}(\mathbf{x}_s, \mathbf{x}_r)\alpha_r + \frac{1}{2C}\sum_{s=1}^{L}(\alpha_s)^2 - \sum_{s=1}^{L} y_s \alpha_s + \varepsilon\sum_{s=1}^{L}\lvert\alpha_s\rvert$ subject to $\left\{\sum_{s=1}^{L}(\alpha_s) = 0,\ \alpha_s \in R^L\right.$
ε-WLSSVR	$\min_{\alpha_s^* \in R^L} J(\alpha_s^*) = \frac{1}{2}\sum_{s=1}^{L}\sum_{r=1}^{L}\alpha_s^* \mathrm{K}(\mathbf{x}_s, \mathbf{x}_r)\alpha_r^* - \sum_{s=1}^{L} y_s \alpha_s^* + \frac{1}{2C}\sum_{s=1}^{L}\frac{1}{\delta_s}(\alpha_s^*)^2 + \varepsilon\sum_{s=1}^{L}\lvert\alpha_s^*\rvert$ subject to $\sum_{s=1}^{L}\alpha_s^* = 0,\ \alpha_s^* \in R^L$
PLSSVR	$\min_{\alpha \in R^L} J(\alpha_s) = \frac{1}{2}\sum_{s=1}^{L}\sum_{r=1}^{L}\alpha_s \mathbf{K}(\mathbf{x}_s, \mathbf{x}_r)\alpha_r + \frac{1}{2C}\sum_{s=1}^{L}\alpha_s^2 - \sum_{s=1}^{L} y_s \alpha_s$ subject to $\left\{\sum_{s=1}^{L}(\alpha_s) = 0,\ \alpha_s \in R^L\right.$
WPLSSVR	$\min_{\alpha_s^* \in R^L} J(\alpha_s^*) = \frac{1}{2}\sum_{s=1}^{L}\sum_{r=1}^{L}\alpha_s^* \mathrm{K}(\mathbf{x}_s, \mathbf{x}_r)\alpha_r^* - \sum_{s=1}^{L} y_s \alpha_s^* + \frac{1}{2C}\sum_{s=1}^{L}\frac{1}{\delta_s}(\alpha_s^*)^2$ subject to $\sum_{s=1}^{L}\alpha_s^* = 0,\ \alpha_s^* \in R^L$

8. Retain *Ltot* − *N* samples and set *Ltot*:= *Ltot* − *N*.
9. Go to step 2 and retrain on the reduced training set, unless the user-specified performance index degrades.

Optimization formulations in dual space for PLSSVR, WPLSSVR, ε-LSSVR, ε-WLSSVR, and LSSVR models, are shown in Table 1.

From Table 1 it can be clearly seen that the only difference between ε-LSSVR and classical LSSVR is $\varepsilon\sum_{s=1}^{L}|\alpha_s|$. If ε=0, ε-LSSVR is equivalent to conventional LSSVR. In addition, PLSSVR and classical LSSVR are shown with the same optimization formula. However, an LSSVR model is created based on the whole training dataset, and then the number of samples is iteratively reduced until it reaches the specified error value and the PLSSVR model is obtained. On the other hand, the only difference between ε-WLSSVR and WPLSSVR is again $\varepsilon\sum_{s=1}^{L}|\alpha_s|$. When ε=0, ε-WLSSVR is equivalent to WPLSSVR. Similarly, all samples are used when creating the first model, and then the weighted and pruned model (WPLSSVR) is provided.

Computational Complexity Analysis

In academic literature, the computational complexity of algorithms is frequently evaluated by using Big-O notation. The computational complexity of the standard LSSVR solution (i.e., Ax=B) is O(kL^2) using the conjugate gradient method [13]. Here, A=I+C and A∈RLxL with row(C)=*k*. As discussed in [31], iterative algorithms (such as PLSSVR and WPLSSVR) incur a cost around O(tL^2); where *t* is considered the number of iterations. Together with the SMO technique [32], ε-WLSSVR and ε-LSSVR algorithms generate a cost around (*pL*). Because *p* is generally less than *k*, it provides an additional falling in the computational complexity of ε-WLSSVR and ε-LSSVR algorithms. If the matrix A requires a large amount of memory, it may be recomputed at each iteration step. However, this incurs a cost of O(L^2) per step and decreases the memory requirement to O(*L*). It's worth noting that the computational complexity can vary depending on the chosen kernel type and regularization parameter.

RESULTS AND DISCUSSION

In this section, direct models (ε-LSSVR and ε-WLSSVR) and iterative models (PLSSVR and WPLSSVR) are comparatively analyzed on synthetic and real-life benchmark datasets. In order to ensure identical circumstances for all models, experiments were conducted using SMO algorithm [31, 32] in MATLAB 2012b environment on a PC with Intel Core I5 processors clocked at 3.0 GHz, 4 GB RAM, 64-bit Windows-7 operation system. The parameters and performance metric used in the comparison are given in section 3.1. Their performance on synthetic and real-life data in terms of number of support vectors, percentage of support vectors, complexity parameter, training and testing times are reported in Sections 3.2 and 3.3, respectively. Finally, in Section 3.4, the effects of the compared models on performance are discussed in detail and the observed findings are reported.

Experimental Setup

In all the compared models, the Gaussian function $K(\mathbf{x}_s,\mathbf{x}) = exp\left(-\|\mathbf{x}-\mathbf{x}_s\|_2^2/2\sigma^2\right)$ was selected as the kernel function [33]. Optimal values of regularization parameter (C) and kernel parameter (σ) were determined from sets $\{2^{-4},2^{-3},\cdots,2^{4}\}$ and $\{2^{0},2^{1},\cdots,2^{20}\}$, respectively, by employing classical LSSVR with 5-fold cross-validation approach. Root Mean Square Error (RMSE), defined as follows, was used as the performance index in the study.

$$RMSE = \sqrt{\frac{1}{L}\sum_{s=1}^{L}(y_s - f(\mathbf{x}_s))^2}$$

where, $f(\mathbf{x}_s)$ represents the estimation of the target value y_s when $\mathbf{x}_s$ is entered, and *L* representing sample count.

Synthetic Data Sets

A synthetic data set was produced using the sinc function, which is frequently preferred in machine learning-based regression problems [9, 33].

$y\ sinc(x/\pi)$ with $x \in [-10,10]$.

Using the sinc function, 251 training and 250 test instances were derived with both uniform and random sampling techniques. This approach allowed the introduced models to be evaluated on non-uniform data points. Input data points were normalized to the range [0, 1], while output data points remained unchanged. The training (output) samples were subjected to Gaussian noise $\frac{1}{\sqrt{2\pi}}exp(-(x-\mu)^2/2\sigma^2)$ with μ=0 and σ=0,1. To improve robustness testing, nine artificial outliers were added to the noisy training set, resulting in a total of 260 data points.

The experimental findings are illustrated through Figure 2 and Figure 3, while a detailed numerical analysis, including the number of support vectors, complexity (flatness), and RMSE, is presented in Table 2.

The quantities of training and test data are detailed in Table 2, with the dataset randomly split in each sample. This procedure was repeated 10 times to remove sample dependence and the results were averaged and entered in Table 2.

Figure 2 compares the number of support vectors obtained when the test accuracies (RMSE test values) of the ε-LSSVR and PLSSVR models are equivalent. Both models achieved equal test accuracies under the conditions of $C = 2^3$, $\sigma = 2^{-3}$ for PLSSVR, and $C = 2^3$, $\varepsilon = 0.12$, $\sigma = 2^{-3}$ for ε-LSSVR. From Figure 2 and Table 2, It is evident that ε-LSSVR requires fewer support vectors (120) compared to PLSSVR (174) to attain near the same test accuracy. In

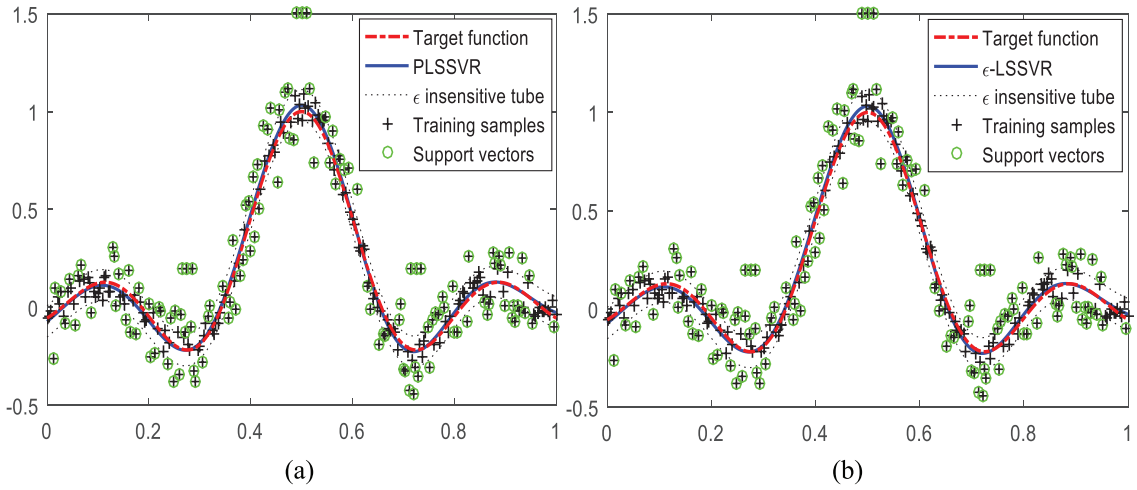

Figure 2. The experimental results for the artificial datasets, with parameters $C = 2^3$ and $\sigma = 2^{-3}$ on almost the same value of RMSE=0.0178 a) PLSSVR produces 174 support vectors b) ε-LSSVR yields 120 support vectors with ε = 0.12.

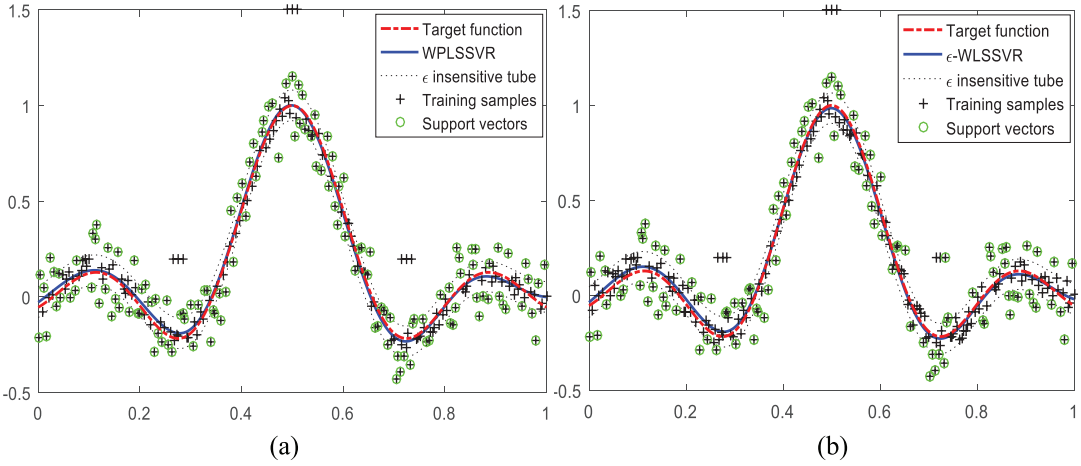

Figure 3. The experimental results for the artificial datasets, with parameters $C = 2^3$ and $\sigma = 2^{-3}$ on almost the same value of RMSE=0.0195 a) WPLSSVR generates 142 support vectors. b) ε-WLSSVR with ε = 0.08 gives 117 support vectors.

addition, Table 2 shows that ε-LSSVR demonstrates sparsity both in the dual space, as indicated by the quantity of support vectors, and in the primary space, as evidenced by the $\boldsymbol{w}$. Accordingly, in randomly chosen training data points, the ε-LSSVR exhibits fewer support vectors and lower $\boldsymbol{w}$ value compared to PLSSVR.

The second comparison was conducted between WPLSSVR with $\sigma = 2^{-3}$, $C = 2^3$ and ε-WLSSVR with $\sigma = 2^3$, $C = 2^3$, $\varepsilon = 0.08$ when the test accuracies of the LSSVR models are identical circumstances. From Figure 3 and Table 2, it's clear that ε-WLSSVR (117) demands a reduced number of support vectors compared to WPLSSVR (142), all while retaining the same test accuracy. This indicates that ε-WLSSVR achieves a significantly sparser solution compared to WPLSSVR. Additionally, in the case of random sampling, ε-WLSSVR requires fewer support vectors compared to WPLSSVR.

As seen in Table 2, while whole training examples are utilized in the LSSVR representation, only 40% of them are used in the ε-LSSVR. Conversely, pruning-based algorithms generally yield a solution according to user-specified error tolerance, thus providing a near-optimal solution. This means that pruning-based algorithms must be run again and again to obtain the favored result. On the other hand, ε-WLSSVR and ε-LSSVR models are quite advantageous as they provide a globally optimal solution without the need for repeated algorithms to achieve optimality. Based on the simulation results above, it can be inferred that ε-WLSSVR effectively reduces the impact of outliers while also yielding a sparse solution in primary and dual spaces.

Real-Life Benchmark Data Sets

The direct models (ε-LSSVR and ε-WLSSVR) and iterative models (PLSSVR and WPLSSVR) were comparatively

Table 2. Experimental results on synthetics dataset

Hyperparameters		Algortihm	ε	#SV	$\|\| \boldsymbol{w} \|\|_2^2$	RMSE test
	Random Sampling	ε-LSSVR	**0.08**	124	1,305796	**0,022351**
		ε-WLSSVR	**0.08**	118	1,358898	**0,020651**
		LSSVR	-	260	1,591505	**0,027793**
#TS=260		PLSSVR	-	145	1,568524	**0,026620**
σ=0.125		WPLSSVR	-	142	1,399055	**0,020790**
C=8	Uniform Sampling	ε-LSSVR	**0.08**	123	1,318347	**0,020921**
N (0, 0.01)		ε-WLSSVR	**0.08**	117	1,365382	**0,019580**
		LSSVR	-	260	1,682521	**0,015634**
		PLSSVR	-	168	1,642521	**0,018634**
		WPLSSVR	-	164	1,403433	**0,019466**

analyzed with 8 distinct real life benchmark datasets given in Table 3. Space, CPU small, and Mg are from Statlib collection[1] while the remaining datasets are from the UCI machine learning repository[2]. To provide consistency, the inputs were normalized within the closed range [0,1]. However, no normalization was applied to the outputs, as scaling is carried out by the C and ε model parameters. For further evaluation, the performances of the ε-WLSSVR and ε-LSSVR models were compared with WPLSSVR and PLSSVR with respect to percentage of support vectors (%SV), the flatness (*w*), number of support vectors (#SV), computation time, and training and test approach error, with test performances of all models nearly equivalent**.**

The experimental findings of LSSVR, ε-LSSVR, ε-WLSSVR, PLSSVR, and WPLSSVR, models on each dataset are presented in Table 4. The user-defined σ and C parameters of the model used for each dataset are presented under their respective names in the first column of Table 4.

From Table 4, it is evident that the sparseness of the weighted and unvweighted ε-LSSVR models surpasses that of PLSSVR, WPLSSVR and LSSVR, across all datasets. For instance, in the Concrete dataset, PLSSVR, WPLSSVR, LSSVR, ε-LSSVR and ε-WLSSVR models required 356, 369, 450,197 and 103 support vectors, respectively. This indicates that ε-LSSVR, and ε-WLSSVR models require fewer support vectors to achieve almost the same test performance. Furthermore, the flatness measure of ε-WLSSVR is lower compared to all LSSVR methods, resulting in a sparser solution within the input space. For example, on the Boston dataset, PLSSVR, WPLSSVR, LSSVR, ε-LSSVR, and ε-WLSSVR have flatness measures of 246, 208, 258,172, and 141, respectively, under identical circumstances. Moreover, ε-LSSVR outperforms all models in terms of computational time. Given the superior performance of the ε-LSSVR and ε-WLSSVR methods, they can be applied to any engineering field such as energy [36] and mechanics [37].

It is evident from the last column of the Table 4 that ε-LSSVR requires less training time compared to LSSVR, PLSSVR and WPLSSVR. This difference can be attributed to the iterative nature of pruning-based algorithms, which continuously refine the objective function by removing less significant training examples until the user-specified error threshold is met. In contrast, during the training process, ε-LSSVR disregards training samples situated within the ε-insensitive region of the target function, all without the necessity of employing recursive (and computationally expensive) algorithms. The user-defined error tolerance parameter ε directly affects the sparsity and computational efficiency of the solution by controlling the number of support vectors. However, it's important to note that ε-LSSVR, like LSSVR, remains sensitive to outliers. To address this limitation, the ε-WLSSVR model incorporates a weighting technique to enhance its robustness.

Table 3. In-depth details regarding benchmark regression datasets

Datasets	Sample Size	Number of Features
CPU Small	8192	12
Space	3107	6
Airfoil	1503	6
Mg	1385	6
Concrete	1030	9
Boston	506	13
Yatch	308	7
Servo	167	4

1 http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression.html.
2 http://archive.ics.uci.edu/ml/datasets.html.

Table 4. Experimental results on real word regression benchmark datasets

Dataset	Learning Models	#TS	#SV	%SV	$\|\| w \|\|_2^2$	RMSE train	RMSE test	Training time
	LSSVR		100	100	65,24784	0,366781	0,584084	0,081989
Servo	PLSSVR		59,8	59,8	64,42886	0,478413	0,672223	0,798908
C=1024	WPLSSVR	100	66,8	66,8	43,84360	0,568590	0,675431	0,946633
σ=1,	ε-LSSVR (ε=0.45)		43,6	43,6	47,38996	0,479219	0,674658	0,045995
	ε-WLSSVR(ε=0.4)		39,2	39,2	23,12014	0,635481	0,671208	0,076589
	LSSVR		160	100	551,7821	0,085507	1,214634	3,34804
Yatch	PLSSVR		156,1	97,5625	551,3142	0,277917	1,270256	6,796333
C=131072	WPLSSVR	160	157,6	98,5	504,2195	0,338374	1,292471	9,248233
σ=0.5	ε-LSSVR(ε=0.2)		110,1	68,8125	449,739	0,192999	1,262674	0,59776
	ε-WLSSVR, ε=0.1		127,8	79.875	439,2768	0,340269	1,262524	4,118371
	LSSVR		400	100	258,0354	1,993912	2,851667	0,099069
Boston	PLSSVR		191,5	47,875	246,5892	2,68649	3,231517	1,551074
C=128	WPLSSVR	400	253,1	63,275	208,0421	2,760021	3,230102	1,709849
σ=1	ε-LSSVR,ε=3		119,1	29,775	172,2871	2,581367	3,237997	0,068996
	ε-WLSSVR(ε=2.6)		127,9	31,975	141,8667	2,649894	3,230158	0,146366
	LSSVR		450	100	2670,986	3,623914	6,319183	1,198805
Concrete	PLSSVR		356,5	79,22222	2646,297	3,971043	6,58039	6,507052
C=4096	WPLSSVR	450	369,9	82,2	2495,326	4,041854	6,571906	11,51769
σ=1	ε-LSSVR,ε=4.8,		197,1	43,8	1564,716	4,644485	6,578823	0,489621
	ε-WLSSVR,ε=4.7		193,1	42,91111	1380,526	4,777453	6,580034	1,344742
	LSSVR		799,9	99,9875	1,514474	0,104216	0,119987	0,095426
Mg	PLSSVR		391	48,875	1,431114	0,108516	0,122091	1,494786
C=2	WPLSSVR	800	708	88,5	1,461827	0,106925	0,122079	0,596783
σ=0.125	ε-LSSVR,ε=0.9		364,3	45,5375	1,106773	0,112639	0,122211	0,072719
	ε-WLSSVR,ε=0.9		348,3	43,5375	1,069306	0,112863	0,122406	0,14033
	LSSVR		900	100	280,2655	1,463983	2,665281	0,412481
Airfoil	PLSSVR		760	84,4444	278,2486	1,882925	2,875038	1,837134
C=128	WPLSSVR	900	886,8	98,53333	203,8609	1,932055	2,86893	1,677196
σ=0.125	ε-LSSVR,ε=1.4		497,5	55,27778	211,7781	1,773606	2,87319	0,194541
	ε-WLSSVR,ε=0.3		750	83,3333	190,6877	1,876238	2,874714	0,633096
	LSSVR		1600	100	51,74661	0,089768	0,103348	10,84307
Space	PLSSVR		1135,8	70,9875	51,07372	0,095491	0,107249	77,11352
C=4096	WPLSSVR	1600	1094,1	68,38125	47,60751	0,097645	0,10753	132,5333
σ=0.5	ε-LSSVR,ε=0.09,		540,1	33,75625	33,2183	0,094133	0,106865	1,446744
	ε-WLSSVR,ε=0.09		522,4	32,65	27,89796	0,094786	0,107426	12,03891
	LSSVR		2500	100	504,8244	2,483706	3,080268	2,422676
CPU Small	PLSSVR		1855	74,2	497,1927	2,789671	3,222043	17,8483
C=128	WPLSSVR	2500	1765	70,6	460,419	2,856357	3,224067	34,5773
σ=0.5	ε-LSSVR, ε=2.5		969,7	38,788	346,417	2,738471	3,215779	0,517647
	ε-WLSSVR, ε=3.1		673,7	26,948	276,9492	2,840616	3,218984	2,756323

CONCLUSION

LSSVR stands out as a computationally efficient method for tackling regression problems. Nevertheless, it does have two notable disadvantages. Firstly, LSSVR tends to lack of sparsity, resulting in every input sample treated as a support vector. Secondly, the solution obtained with LSSVR is sensitive to outliers and noise within the training dataset. In order to address these issues, theoretically derives ε-LSSVR from the LSSVR model. In addition, a weighted version, ε-WLSSVR, is introduced to improve robustness against outliers.

To improve the sparsity of classical LSSVR, the performances of the PLSSVR, ε-LSSVR, WPLSSVR and ε-WLSSVR methods are analyzed in terms of generalization ability, sparsity, and computation time on both artificial and 8 different real-life datasets. Experimental results show that ε-LSSVR and ε-WLSSVR models achieve sparser solution representation compared to PLSSVR, WPLSSVR, and LSSVR across all datasets while maintaining nearly same generalization performance (RMSE test values). These models exhibit advantages over PLSSVR, WPLSSVR, and classical LSSVR regarding the number of support vectors. For example, in the space dataset, PLSSVR, WPLSSVR, LSSVR, ε-LSSVR and ε-WLSSVR models required 1135, 1094, 1600, 540 and 522 support vectors, respectively, to achieve almost the nearly identical circumstances. The norm of $\boldsymbol{w}$ (flatness measure) of both weighted and unweighted ε-LSSVR models is lower than those of all LSSVR models, indicating sparser solution representations in the primal space. For instance, in the CPU Small dataset, PLSSVR, WPLSSVR, LSSVR, ε-LSSVR, and ε-WLSSVR demonstrates flatness measures of 497, 460, 504, 346, and 276, respectively. Additionally, ε-LSSVR outperforms all models in terms of computational time.

The regression models introduced in this study can be optimized using any optimization algorithm developed for nonlinear convex large-scale quadratic problems subject to linear inequalities/equalities, potentially increasing computational efficiency. Furthermore, these models can be applied to various engineering problems in which other regression models have shown success.

AUTHORSHIP CONTRIBUTIONS

Authors equally contributed to this work.

DATA AVAILABILITY STATEMENT

The authors confirm that the data that supports the findings of this study are available within the article. Raw data that support the finding of this study are available from the corresponding author, upon reasonable request.

CONFLICT OF INTEREST

The author declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

ETHICS

There are no ethical issues with the publication of this manuscript.

REFERENCES

[1] Cortes C, Vapnik V. Support-vector networks. Mach Learn 1995;20:273–297. [CrossRef]

[2] Karal Ö. Compression of ECG data by support vector regression method. J Fac Eng Archit Gazi Univ 2018;33:743–755. [CrossRef]

[3] Karadurmuş E, Göz E, Taşkın N, Yüceer M. Bromate removal prediction in drinking water by using the least squares support vector machine (LS-SVM). Sigma J Eng Nat Sci 2020;38:2145–2153.

[4] Filiz E, Ersoy ÖZ. Educational data mining methods for TIMSS 2015 mathematics success: Turkey case. Sigma J Eng Nat Sci 2020;38:963–977.

[5] Bakay MS, Ağbulut Ü. Electricity production-based forecasting of greenhouse gas emissions in Turkey with deep learning, support vector machine and artificial neural network algorithms. J Clean Prod 2021;285:125324. [CrossRef]

[6] Mir AA, Çelebi FV, Alsolai H, Qureshi SA, Rafique M, Alzahrani JS, et al. Anomalies prediction in radon time series for earthquake likelihood using machine learning based ensemble model. IEEE Access 2022;10:1. [CrossRef]

[7] Sonmez ME, Eczacıoglu N, Gumuş NE, Aslan MF, Sabanci K, Aşikkutlu B. Convolutional neural network-Support vector machine-based approach for classification of cyanobacteria and chlorophyta microalgae groups. Algal Res 2022;61:102568. [CrossRef]

[8] Vapnik, VN, Vapnik V. Statistical Learning Theory. New York: Wiley; 1998.

[9] Smola AJ, Schölkopf B. A tutorial on support vector regression. Stat Comput 2004;14:199–222. [CrossRef]

[10] Suykens JA, Vandewalle J. Least squares support vector machine classifiers. Neural Process Lett 1999;9:293–300. [CrossRef]

[11] Saunders C, Gammerman A, Vovk V. Ridge regression learning algorithm in dual variables. In: Proceedings of the Fifteenth International Conference on Machine Learning (ICML); 1998 Jul 24-27; Madison, Wisconsin, USA. 1998.

[12] Suykens JA, Lukas L, Vandewalle J. Sparse approximation using least squares support vector machines. In: Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium; 2000 Feb; Geneva. 2000. pp.757–760.

[13] Suykens JA, De Brabanter J, Lukas L, Vandewalle J.bWeighted least squares support vector machines: robustness and sparse approximation. Neurocomput 2002;48:85–105. [CrossRef]

[14] De Kruif BJ, De Vries TJ. Pruning error minimization in least squares support vector machines. IEEE Trans Neural Netw 2003;14:696–702. [CrossRef]
[15] Kuh A, De Wilde P. Comments on pruning error minimization in least squares support vector machines. IEEE Trans Neural Netw 2007;18:606–609. [CrossRef]
[16] Hoegaerts L, Suykens JA, Vandewalle J, De Moor B. (2004, November). A comparison of pruning algorithms for sparse least squares support vector machines. In: International Conference on Neural Information Processing; 2004 Nov; Heidelberg, Berlin: Springer; 2004. pp.1247–1253. [CrossRef]
[17] Zeng X, Chen XW. SMO-based pruning methods for sparse least squares support vector machines. IEEE Trans Neural Netw 2005;16:1541–1546. [CrossRef]
[18] Zhao Y, Sun J. Recursive reduced least squares support vector regression. Pattern Recognit 2009;42:837–842. [CrossRef]
[19] Zhao YP, Sun JG, Du ZH, Zhang ZA, Zhang YC, Zhang HB. An improved recursive reduced least squares support vector regression. Neurocomput 2012;87:1–9. [CrossRef]
[20] Zhao YP, Wang KK, Li F. A pruning method of refining recursive reduced least squares support vector regression. Inf Sci 2015;296:160–174. [CrossRef]
[21] Si G, Shi J, Guo Z, Jia L, Zhang Y. Reconstruct the support vectors to improve LSSVM sparseness for Mill Load prediction. Math Probl Eng 2017;2017:1–12. [CrossRef]
[22] Sun B, Ng WW, Chan PP. Improved sparse LSSVMS based on the localized generalization error model. Int J Mach Learn Cybern 2017;8:1853–1861. [CrossRef]
[23] Espinoza M, Suykens JA, Moor BD. Fixed-size least squares support vector machines: a large scale application in electrical load forecasting. Comput Manag Sci 2006;3:113–129. [CrossRef]
[24] Mall R, Suykens JA. Sparse reductions for fixed-size least squares support vector machines on large scale data. In: Pei, J., Tseng, V.S., Cao, L., Motoda, H., Xu, G., editors. Pacific-Asia Conference on Knowledge Discovery and Data Mining. Heidelberg, Berlin: Springer; 2013. [CrossRef]
[25] Yang L, Yang S, Zhang R, Jin H. Sparse least square support vector machine via coupled compressive pruning. Neurocomput 2014;131:77–86. [CrossRef]
[26] Zhou S. Sparse LSSVM in primal using Cholesky factorization for large-scale problems. IEEE Trans Neural Netws Learn Sys 2015;27:783–795. [CrossRef]
[27] Chen L, Zhou S. Sparse algorithm for robust LSSVM in primal space. Neurocomput 2018;275:2880–2891. [CrossRef]
[28] Xia XL. Training sparse least squares support vector machines by the QR decomposition. Neural Netw 2018;106:175–184. [CrossRef]
[29] Ma Y, Liang X, Sheng G, Kwok JT, Wang M, Li G. Noniterative sparse LS-SVM based on globally representative point selection. IEEE Trans Neural Netw Learn Sys 2020;32:788–798. [CrossRef]
[30] Karal Ö. Piecewise affine and support vector models for robust and low complex regression (Doctoral dissertation). İzmir: DEÜ Fen Bilimleri Enstitüsü; 2011.
[31] Li K, Peng JX, Bai EW. A two-stage algorithm for identification of nonlinear dynamic systems. Automatica 2006;42:1189–1197. [CrossRef]
[32] Jiao L, Bo L, Wang L. Fast sparse approximation for least squares support vector machine. IEEE Trans Neural Netw 2007;18:685–697. [CrossRef]
[33] Fan RE, Chen PH, Lin CJ. Working set selection using second order information for training support vector machines. J Mach Learn Res 2005;6:1889–1918.
[34] Keerthi SS, Shevade SK. SMO algorithm for least-squares SVM formulations. Neural Comput 2003;15:487–507. [CrossRef]
[35] Karal O. Maximum likelihood optimal and robust Support Vector Regression with lncosh loss function. Neural Netw 2017;94:1–12. [CrossRef]
[36] Armin M, Gholinia M, Pourfallah M, Ranjbar AA. Investigation of the fuel injection angle/time on combustion, energy, and emissions of a heavy-duty dual-fuel diesel engine with reactivity control compression ignition mode. Energy Rep 2021;7:5239–5247. [CrossRef]
[37] Zadeh MN, Pourfallah M, Sabet S, Gholinia M, Mouloodi S, Ahangar AT. Performance assessment and optimization of a helical Savonius wind turbine by modifying the Bach's section. SN Appl Sci 2021;3:1–11. [CrossRef]