A Study on Hyperparameter Tuning of Twin Support Vector Regression Based on Bayesian Optimization

Zirui Mao [*, #], Liwen Feng [#]

School of Mathematics and Statistics Mathematics, Henan University of Science and Technology, Luoyang, China, 471000

* Corresponding Author Email: 17597981767@163.com

[#]These authors contributed equally.

Abstract. This study addresses the challenge of optimizing the predictive performance of Twin Support Vector Regression (TSVR) models on complex datasets by employing Bayesian Optimization (BO) for automatic hyperparameter tuning. As an enhanced variant of Support Vector Regression (SVR), TSVR achieves significantly greater computational efficiency by solving two smaller-scale quadratic programming problems. Within the Bayesian optimization framework, the construction of a posterior probability model enables efficient exploration of the hyperparameter space, thereby overcoming the inefficiency limitations of traditional methods. Experimental validation demonstrates that the BO-optimized TSVR model achieves significant reductions in Mean Squared Error (MSE) across multiple synthetic datasets, while simultaneously exhibiting superior predictive accuracy and generalization capability. The significance of this research lies not only in its improvement of TSVR model performance but also in its provision of a novel approach for hyperparameter optimization in machine learning, possessing substantial theoretical and practical merit.

Keywords: Twin Support Vector Regression, Bayesian Optimization, Hyperparameter Tuning, Regression.

1. Introduction

As a unified kernel-learning framework, support-vector techniques have been repeatedly validated by numerous engineering examples. In classification tasks, early mainstream approaches—represented by C-SVC and LS-SVC—rest on the classical geometric assumption of "one hyperplane, two half-spaces"[1][3]. Yet, when class distributions intersect or overlap, the expressive power of a single plane is often insufficient. GEPSVM was the first to introduce a generalized-eigenvalue perspective, projecting each class onto its own proximal plane; the subsequent TWSVM further decomposes the global problem into two smaller quadratic-programming sub-problems that each involve only one class [4][5]. In contrast, C-SVC must solve one large-scale quadratic program in a single shot; TWSVM's divide-and-conquer strategy markedly reduces computational complexity and endows the model with an inherent robustness to cross-plane data. This advantage not only shortens training time but has also inspired a wealth of follow-up work on non-parallel decision surfaces[6][7] . Recent advances focus on enhancing robustness and adaptability. Yuan et al. proposed CL2,p-LSTSVM using capped L2,p-norm metrics to suppress outlier influence in classification[8],

For instance, Krein TWSVM leverages indefinite kernels in reproducing kernel Krein spaces to handle imbalanced data[12], while symmetric LINEX loss TWSVM enhances robustness against label and feature noise[14]. Recently, the Granular Ball K-Class Twin Support Vector Classifier has been introduced to tackle key challenges in multi-class classification by integrating Twin Support Vector Machines with granular ball computing, demonstrating substantial improvements in both accuracy and computational efficiency [15].

In regression settings, the support-vector philosophy has likewise evolved into two complementary lines: ε-SVR and LS-SVR [3][16]. Taking linear ε-SVR as an example, its optimization objective can be intuitively interpreted as constructing an "ε-wide tube" in feature space so that the vast majority of training samples fall inside; simultaneously, a regularization term constrains the norm of the weight

vector to curb function oscillation, thereby integrating empirical risk and structural risk within a single framework. The working principle of linear LS-SVR is similar. Unlike ε-SVR and LS-SVR, which are commonly used methods in the field of support vector regression, Peng [17] proposed a regressor based on the idea of TWSVM, called TSVR. The formulation of TSVR is similar to that of TWSVM, realized through two non- parallel planes, and solves two smaller QPPs, whereas the traditional SVR solves one larger QPP. Kumari et al. proposed LSTSVR incorporating privileged information via least squares frameworks [17]. Recent extensions include twin proximal SVR with heteroscedastic Gaussian noise (TPSVR-HGN) for wind prediction [18]and twin support vector quantile regression (TSVQR) for capturing heterogeneous/asymmetric distributions [19]. Experimental results in [5][19][23] show that TSVR and its variants outperform classical SVR in generalization and efficiency. Moreover, the development of intuitionistic fuzzy twin support vector machines for imbalanced data (IFTSVM-ID) has shown promising results in handling imbalanced datasets with noises and outliers [21]. The robust twin depth support vector machine based on average depth (TDSVM) has also been introduced to enhance its resilience against noise or outliers [22]. Furthermore, the fuzzy regular least squares twin support vector machine (FRLSTSVM) has been introduced to enhance the generalization performance and reduce the impact of outliers on results [23].

Recent advancements in TWSVM and its variants have shown promise. Che et al. enhanced TWSVM with privileged information [24]. For quantile regression, L1-norm TSVQR (L1-TSVQR) further enables sparse feature selection in high-dimensional data [25]. In imbalanced learning, reduced universum TWSVM incorporates prior data distribution via universum points [26], and adaptive Adaboost-based TWSVM integrates robust loss functions with universum learning [27]. Multi-view extensions include deep multi-view TSVM using DNN/AE networks [28] and hypergraph-regularized Lp-norm least squares TSVM for semi-supervised learning [29]. Additionally, Huang et al. provided a comprehensive review of TSVR's theoretical developments [30]. Despite these improvements, hyperparameter optimization remains a bottleneck. Bayesian optimization offers a solution by constructing a posterior probability model to efficiently search for optimal hyperparameters. This paper proposes a Bayesian optimization-based method to tune the hyperparameters of Twin Support Vector Regression (TSVR), aiming to enhance its predictive performance on complex datasets. Experimental results demonstrate significant improvements in Mean Squared Error (MSE) and generalization ability.

2. Model

In this study, we consider a set of training samples, each represented by a vector in an n-dimensional real space, where the range of i is from 1 to l, and the ith sample is denoted as $A_i = (A_{i1}, A_{i2}, \ldots, A_{in})$. We denote this set of samples as A, and the corresponding response values form a vector, which contains the real values corresponding to each sample. Our goal is to investigate Standard SVR, LS-SVR, and TSVR.

2.1. Standard Support Vector Regression

In linear SVR, the objective is to identify a linear regression function

$$f(x) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + \boldsymbol{b}, \tag{1}$$

where $\boldsymbol{w} \in R^n$ is the weight vector and $\boldsymbol{b} \in R$ is the bias. This function aims to fit the given dataset while allowing for a small error tolerance. This result can be achieved using the ε-insensitive loss function, which defines an ε-insensitive "tube" surrounding the data points and disregards deviations within this tube. The research aspires to maintain the regression function $f(x)$ as smooth as possible during the fitting of training data. The linear SVR formulation can be expressed as the following constrained minimization task:

$$\begin{aligned} &\min \frac{1}{2}\boldsymbol{w}^{T}\boldsymbol{w}+C\left(\boldsymbol{e}^{T}\boldsymbol{\xi}+\boldsymbol{e}^{T}\boldsymbol{\xi}^{*}\right) \\ &\text{s.t.}\quad \boldsymbol{Y}-\left(\boldsymbol{A}\boldsymbol{w}+\boldsymbol{b}\boldsymbol{e}\right)\le \grave{o}\boldsymbol{e}+\boldsymbol{\xi},\boldsymbol{\xi}\ge 0, \\ &\left(\boldsymbol{A}\boldsymbol{w}+\boldsymbol{b}\boldsymbol{e}\right)-\boldsymbol{Y}\le \grave{o}\boldsymbol{e}+\boldsymbol{\xi}^{*},\boldsymbol{\xi}^{*}\ge 0. \end{aligned} \tag{2}$$

In this context, $C>0$ acts as a regularization factor, weighing the trade-off between the fitting error and the smoothness of the regression function. Meanwhile, ξ and ξ^{*} represent the respective slack variables. and e is a vector of appropriate dimension with all elements equal to one.

For the nonlinear case, we introduce a nonlinear mapping $\varphi:\ R^{n}\to \mathrm{M}$, where R^{n} is referred to as the feature space. The inner product in the feature space can be represented by certain kernel functions, such as the Gaussian kernel $k\left(\boldsymbol{u}^{\mathrm{T}},\boldsymbol{v}\right)=\exp\{-\beta\|\boldsymbol{u}-\boldsymbol{v}\|^{2}\}$, with parameter $\beta>0$. This study denote the kernel matrix by $\mathrm{K}\left(\boldsymbol{A},\boldsymbol{B}^{\mathrm{T}}\right)$, where $\boldsymbol{k}_{i,j}=\boldsymbol{k}\left(\boldsymbol{A}_{i},\boldsymbol{B}_{i}^{\mathrm{T}}\right)$. Similarly, the formulation for nonlinear SVR is:

To address nonlinear relationships, this work employs a transformation $\varphi:\ R^{n}\to \mathrm{M}$, projecting input data into a high-dimensional feature space R^{n}. Inner products within R^{n} are computed through kernel functions, exemplified by the Gaussian kernel $k\left(\boldsymbol{u}^{\mathrm{T}},\boldsymbol{v}\right)=\exp\{-\beta\|\boldsymbol{u}-\boldsymbol{v}\|^{2}\}$ parameterized by $\beta>0$. The kernel matrix is denoted $\mathrm{K}\left(\boldsymbol{A},\boldsymbol{B}^{\mathrm{T}}\right)$, with entries defined as $\boldsymbol{k}_{i,j}=\boldsymbol{k}\left(\boldsymbol{A}_{i},\boldsymbol{B}_{i}^{\mathrm{T}}\right)$. Consequently, the nonlinear Support Vector Regression (SVR) formulation is expressed as:

$$\begin{aligned} &\min \frac{1}{2}\boldsymbol{w}^{T}\boldsymbol{w}+C\left(\boldsymbol{e}^{T}\boldsymbol{\xi}+\boldsymbol{e}^{T}\boldsymbol{\xi}^{*}\right) \\ &\text{s.t.}\quad \boldsymbol{Y}-\left(\varphi(\boldsymbol{A})\boldsymbol{w}+\boldsymbol{b}\boldsymbol{e}\right)\le \grave{o}\boldsymbol{e}+\boldsymbol{\xi},\boldsymbol{\xi}\ge 0, \\ &\left(\varphi(\boldsymbol{A})\boldsymbol{w}+\boldsymbol{b}\boldsymbol{e}\right)-\boldsymbol{Y}\le \grave{o}\boldsymbol{e}+\boldsymbol{\xi}^{*},\boldsymbol{\xi}^{*}\ge 0. \end{aligned} \tag{3}$$

where $\varphi(\boldsymbol{A})=\left(\varphi\left(\boldsymbol{A}_{1}\right);\varphi\left(\boldsymbol{A}_{2}\right);\ldots;\varphi\left(\boldsymbol{A}_{l}\right)\right)$. Observe that Equations (2) and (3), two sets of constraints are employed to position as many training samples as possible within the flat region. In practical applications, we usually do not directly solve (2) and (3), but obtain the appropriate linear or nonlinear regression function by solving their dual problems. For example, the dual quadratic programming problem (QPP) of (3) is given by:

$$\begin{aligned} \min\quad & \grave{o}\boldsymbol{e}^{\mathrm{T}}\left(\gamma+\gamma^{*}\right)-\boldsymbol{Y}^{\mathrm{T}}\left(\gamma-\gamma^{*}\right) \\ & +\frac{1}{2}\left(\gamma-\gamma^{*}\right)^{\mathrm{T}}\boldsymbol{K}\left(\boldsymbol{A},\boldsymbol{A}^{\mathrm{T}}\right)\left(\gamma-\gamma^{*}\right) \\ \text{s.t.}\quad & \boldsymbol{e}^{\mathrm{T}}\left(\gamma-\gamma^{*}\right)=0,0\le\gamma,\gamma^{*}\le C\boldsymbol{e}, \end{aligned} \tag{4}$$

here, γ and γ^{*} represent vectors of positive and negative Lagrange coefficients, respectively. Solving this QPP problem is computationally intensive.

2.2. Least Squares Support Vector Regression

In this section, This study will briefly introduce LS-SVR.

The conventional SVR is known for its significant computational demands. To tackle large-scale regression problems more efficiently, the LS-SVR was developed. It redefines the regression problem through a least square's formulation, as follows:

$$\begin{aligned} &\min \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + C\boldsymbol{\xi}^T\boldsymbol{\xi} \\ &\text{s.t.} \quad \boldsymbol{Y} = \varphi(\boldsymbol{A})\boldsymbol{w} + \boldsymbol{be} + \boldsymbol{\xi} \end{aligned} \tag{5}$$

By adding the Lagrange function and taking its derivative, the LS-SVR is given by:

$$\begin{bmatrix} \boldsymbol{K}(\boldsymbol{A},\boldsymbol{A}^{\cdot}) + \frac{1}{C}\boldsymbol{I} & \boldsymbol{e} \\ \boldsymbol{e}^{\cdot} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{b} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Y} \\ \boldsymbol{0} \end{bmatrix}, \tag{6}$$

In this formulation, I denote the identity matrix of suitable size. Unlike standard SVR - which yields sparse solutions with predominantly zero-valued Lagrange multipliers - LS-SVR exhibits proportional scaling between its multiplier vectors ($\boldsymbol{\alpha}$, $\boldsymbol{\xi}$) and the training error vector.

2.3. Twin Support Vector Regression

Twin Support Vector Regression determines the upper and lower bounds of the data points by solving two smaller quadratic programming problems (QPPs). This method is more efficient compared to traditional SVR because it does not solve one large QPP but solves two smaller QPPs separately.

$$\begin{aligned} \min \quad & \frac{1}{2}\left(\boldsymbol{Y} - \boldsymbol{e}\text{蚪} - (\boldsymbol{Aw}_1 + \boldsymbol{eb}_1)\right)^{\cdot} \left(\boldsymbol{Y} - \boldsymbol{e}_{\ 1} - (\boldsymbol{Aw}_1 + \boldsymbol{eb}_1)\right) + C_1\boldsymbol{e}^{\cdot}\boldsymbol{\xi} \\ \text{s.t.} \quad & \boldsymbol{Y} - (\boldsymbol{Aw}_1 + \boldsymbol{eb}_1) \geq \boldsymbol{e}\text{ò}_1 - \boldsymbol{\xi}, \quad \boldsymbol{\xi} \geq 0, \end{aligned} \tag{7}$$

$$\begin{aligned} \min \quad & \frac{1}{2}\left(\boldsymbol{Y} + \boldsymbol{e}\text{蚪} - (\boldsymbol{Aw}_2 + \boldsymbol{eb}_2)\right)^{\cdot} \left(\boldsymbol{Y} + \boldsymbol{e}_{\ 2} - (\boldsymbol{Aw}_2 + \boldsymbol{eb}_2)\right) + C_2\boldsymbol{e}^{\cdot}\boldsymbol{\eta} \\ \text{s.t.} \quad & (\boldsymbol{Aw}_2 + \boldsymbol{eb}_2) - \boldsymbol{Y} \geq \boldsymbol{e}\text{ò}_2 - \boldsymbol{\eta}, \quad \boldsymbol{\eta} \geq 0, \end{aligned} \tag{8}$$

here, C_1 and C_2 are regularization parameters control the smoothness of the model, ξ and η are slack variables, and ε_1 , ε_2 is the insensitive region boundary. To obtain the dual quadratic programming problem (QPP) of TSVR, we formulate the Lagrange function for optimization problem (7) as:

$$\begin{aligned} \boldsymbol{L}(\boldsymbol{w}_1,\boldsymbol{b}_1,\boldsymbol{\xi},\boldsymbol{\alpha},\boldsymbol{\beta}) \quad = & \frac{1}{2}\left(\boldsymbol{Y} - \boldsymbol{e}\text{蚪} - (\boldsymbol{Aw}_1 + \boldsymbol{eb}_1)\right)^{\cdot} \left(\boldsymbol{Y} - \boldsymbol{e}_{\ 1} - (\boldsymbol{Aw}_1 + \boldsymbol{eb}_1)\right) \\ & + \mathrm{C}_1\boldsymbol{e}^{\text{暗}}\boldsymbol{\xi} - \boldsymbol{\alpha}\ \left(\boldsymbol{Y} - (\boldsymbol{Aw}_1 + \boldsymbol{eb}_1) - \boldsymbol{e}\text{ò}_1 + \boldsymbol{\xi}\right) - \boldsymbol{\beta}^{?}\boldsymbol{\xi}, \end{aligned} \tag{9}$$

where α and β denote the Lagrange multiplier vectors. For optimization problem (7), the Karush-Kuhn-Tucker (KKT) conditions – which constitute necessary and sufficient criteria for optimality – take the following form:

$$-\boldsymbol{A}^{\text{暗}}(\boldsymbol{Y} - \boldsymbol{Aw}_1 - \boldsymbol{eb}_2 - \boldsymbol{e}\text{ò}_1) + \boldsymbol{A}\ \boldsymbol{\alpha} = \boldsymbol{0}, \tag{10}$$

$$-\boldsymbol{e}^{\text{暗}}(\boldsymbol{Y} - \boldsymbol{Aw}_1 - \boldsymbol{eb}_2 - \boldsymbol{e}\text{ò}_1) + \boldsymbol{e}\ \boldsymbol{\alpha} = \boldsymbol{0}, \tag{11}$$

$$C_1\boldsymbol{e} - \boldsymbol{\alpha} - \boldsymbol{\beta} = \boldsymbol{0}. \tag{12}$$

$$\boldsymbol{Y} - (\boldsymbol{Aw}_1 + \boldsymbol{eb}_1) \geq \boldsymbol{e}\text{ò}_1 - \boldsymbol{\xi}, \quad \boldsymbol{\xi} \geq \boldsymbol{0}. \tag{13}$$

$$\boldsymbol{\alpha}^{\cdot}\ (\boldsymbol{Y} - (\boldsymbol{Aw}_1 + \boldsymbol{eb}_1) - \boldsymbol{e}\text{ò}_1 + \boldsymbol{\xi}) = \boldsymbol{0}, \quad \boldsymbol{\alpha} \geq \boldsymbol{0}, \tag{14}$$

$$\boldsymbol{\beta}^{\cdot}\ \boldsymbol{\xi} = \boldsymbol{0}, \quad \boldsymbol{\beta} \geq \boldsymbol{0}. \tag{15}$$

Since $\boldsymbol{\beta} \geq \mathbf{0}$, we have $\mathbf{0} \leq \boldsymbol{\alpha} \leq C_1\boldsymbol{e}$. Combining (10) and (11), we obtain:

$$-\begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{e}^{\text{T}} \end{bmatrix}\left((\boldsymbol{Y} - \boldsymbol{e}\grave{o}_1) - \begin{bmatrix} \boldsymbol{A} & \boldsymbol{e} \end{bmatrix}\begin{bmatrix} \boldsymbol{w}_1 \\ \boldsymbol{b}_1 \end{bmatrix}\right) + \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{e} \end{bmatrix}\boldsymbol{\alpha} = \mathbf{0}. \tag{16}$$

Define:

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{e} \end{bmatrix}, \quad \boldsymbol{f} = \boldsymbol{Y} - \boldsymbol{e}\grave{o}_1, \quad \boldsymbol{u}_1 = \begin{bmatrix} \boldsymbol{w}_1^{\cdot} & \boldsymbol{b}_1 \end{bmatrix}^{\cdot}, \tag{17}$$

Then we have

$$-\boldsymbol{G}^{\text{T}}\boldsymbol{f} + \boldsymbol{G}\ \boldsymbol{G}\boldsymbol{u}_1 + \boldsymbol{G}^{\text{T}}\boldsymbol{\alpha} = \mathbf{0}, \quad \text{i.e., } \boldsymbol{u}_1 = (\boldsymbol{G}\ \boldsymbol{G})^{-1}\boldsymbol{G}^{?}(\boldsymbol{f} - \boldsymbol{\alpha}). \tag{18}$$

It's worth mentioning that GG^T is inherently positive semi-definite, yet it can sometimes exhibit poor conditioning. To mitigate this, the study incorporates a regularization term σ_{i}, with σ being a tiny positive constant, like $\sigma = 1\text{e} - 7$. As a result, equation (19) is adjusted to:

$$\boldsymbol{u}_1 = (\boldsymbol{G}^{\text{T}}\boldsymbol{G} + \boldsymbol{\sigma}\boldsymbol{I})^{-1}\boldsymbol{G}\ \ (\boldsymbol{f} - \boldsymbol{\alpha}). \tag{19}$$

The dual quadratic programming problem (QPP) for TSVR is derived by substituting equation (19) and the established KKT conditions into equation (9), then removing constant terms. This yields the following formulation for problem (7):

$$\begin{aligned} \max \quad & -\frac{\mathbf{1}}{\mathbf{2}}\boldsymbol{\alpha}^{\text{T}}\boldsymbol{G}(\boldsymbol{G}\ \boldsymbol{G})^{-1}\boldsymbol{G}^{\text{T}}\boldsymbol{\alpha} + \boldsymbol{f}\ \ \boldsymbol{G}(\boldsymbol{G}^{\text{T}}\boldsymbol{G})^{-1}\boldsymbol{G}\ \ \boldsymbol{\alpha} - \boldsymbol{f}^{?}\boldsymbol{\alpha} \\ \text{s.t.} \quad & \mathbf{0} \leq \boldsymbol{\alpha} \leq C_1\boldsymbol{e}. \end{aligned} \tag{20}$$

Similarly, considering problem (8), the dual form of the problem is derived as:

$$\begin{aligned} \max \quad & -\frac{1}{2}\boldsymbol{\gamma}^{\text{T}}\boldsymbol{G}(\boldsymbol{G}\ \boldsymbol{G})^{-1}\boldsymbol{G}^{\text{T}}\boldsymbol{\gamma} - \boldsymbol{h}\ \boldsymbol{G}(\boldsymbol{G}^{\text{T}}\boldsymbol{G})^{-1}\boldsymbol{G}\ \ \boldsymbol{\gamma} + \boldsymbol{h}^{?}\boldsymbol{\gamma} \\ \text{s.t.} \quad & 0 \leq \boldsymbol{\gamma} \leq C_2\boldsymbol{e}, \end{aligned} \tag{21}$$

where $h = Y + \grave{o}_2$. subject to

$$\boldsymbol{u}_2 = \begin{bmatrix} \boldsymbol{w}_2 \\ \boldsymbol{b}_2 \end{bmatrix} = (\boldsymbol{G}^{\text{T}}\boldsymbol{G})^{-1}\boldsymbol{G}\ \ (\boldsymbol{h} + \gamma). \tag{22}$$

3. Hyperparameter Optimization Strategy

The performance of Support Vector Regression (SVR) and its variants is highly dependent on the configuration of hyperparameters, including the regularization parameter *C* and kernel function parameters (e.g., γ for the RBF kernel). These hyperparameters significantly influence the model's generalization ability and prediction accuracy. Conventional hyperparameter selection techniques (e.g., grid/random search) prove inefficient and ill-suited for complex parameter spaces. This work thus introduces a Bayesian optimization framework to automatically determine TSVR's optimal hyperparameter configuration.

3.1. Formulation of the Optimization Problem

The objective is to minimize the Mean Squared Error (MSE) of the model, defined as:

$$\min_{\varsigma} \text{MSE}(\varsigma) = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{j}_i - j_i\right)^2 \tag{23}$$

where ς represents the set of hyperparameters, $\hat{j}_i$ is the predicted value, j_i is the actual value, and *N* is the number of samples in the dataset.

3.2. Hyperparameter Optimization Strategy

Bayesian optimization efficiently identifies optimal hyperparameters by constructing a probabilistic surrogate model (e.g., Gaussian process) of the objective function and selecting subsequent evaluation points through acquisition functions such as Expected Improvement. This method can identify near-optimal solutions with fewer evaluations, making it suitable for computationally expensive objective functions.

The procedure comprises these steps:

1. Initialization: Sample a random hyperparameter configuration, evaluate its Mean Squared Error (MSE), and construct the initial training dataset.
2. Surrogate Model Construction: Model the objective function using a Gaussian process to create a surrogate model.
3. Next Evaluation Point Selection: Choose the next evaluation point based on the acquisition function (e.g., Expected Improvement).
4. Objective Function Evaluation: Compute the MSE for the new evaluation point.
5. Dataset Update: Add the new evaluation point and its corresponding MSE to the dataset.
6. Iteration: Repeat steps 2-5 until the maximum iteration count is reached or convergence conditions are satisfied.
7. Output Optimal Solution: Output the optimal hyperparameter combination.

4. Bayesian Optimization

Bayesian optimization is an optimization framework that constructs a surrogate model to approximate and understand problems that cannot be explicitly modeled. The surrogate model captures current knowledge and uncertainty about the problem. By continuously updating the model with new observations, Bayesian optimization identifies the optimal solution.

The working principle involves establishing a prior distribution (typically a Gaussian process) to represent the global behavior of the objective function. This prior is updated with observations at different input points to construct a posterior distribution. The subsequent sampling point is determined based on this posterior, balancing exploitation of known optimal values and exploration of uncharted regions, usually guided by an acquisition function, like Expected Improvement. This approach allows Bayesian optimization to efficiently search the hyperparameter space using existing knowledge and avoid unnecessary attempts.

Bayesian optimization is characterized by high sample efficiency, converging to the global optimum with relatively few iterations. Its popularity has increased due to the demand for automated hyperparameter tuning in machine learning and artificial intelligence, where training costs are significant. It is widely used to automatically set hyperparameters to achieve better training outcomes.

The problem that Bayesian optimization aims to address is how to efficiently identify the optimal combination of hyperparameters to minimize the objective function MSE

$$\min_{\theta} \mathrm{MSE}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i - y_i\right)^2 \tag{24}$$

For the aforementioned optimization problem, several approaches are typically considered. One is grid search, another is random search, and the third is sequential search, which includes Bayesian optimization. Below is an introduction to these methods:

Grid Search

As the name implies, grid search involves discretizing the search space into a grid and then exhaustively searching through it. The advantage is its simplicity and ease of understanding. However,

it is resource-intensive and not suitable for problems with high computational costs. Additionally, it suffers from the curse of dimensionality, where the number of points to be searched grows exponentially with the dimensionality of the variables.

Random Search

If we assume a probability distribution for each dimension of the variables and sample accordingly, we can partially improve the inefficiency of grid search. However, due to its randomness, we may end up sampling several very similar points, wasting resources. Moreover, there is a risk of missing the optimal region, potentially failing to find the global optimum, or increasing the number of iterations required.

A significant issue with both grid search and random search is that they do not effectively utilize previous iterations and explorations to inform subsequent ones. If the order of sampling is considered, we can use the results of previous samples to guide the location of future samples. On one hand, we can exploit the regions that have already been explored to determine where higher returns might be obtained. On the other hand, we can also decide which unexplored regions require further information. A good strategy is to balance exploitation and exploration. For expensive black-box functions, sequential search strategies are very necessary because they can prioritize reducing the number of optimization iterations. This is where the advantage of Bayesian optimization lies. The Bayesian optimization algorithm framework is as follows:

Bayesian optimization algorithm
Input: $f, \mathrm{X}, S, \mathrm{M}$
$\mathrm{D} \leftarrow \mathrm{InitSamples}(f, \mathrm{X})$
for $i \leftarrow \lvert \mathrm{D} \rvert$ to T do
$p(y \mid \mathbf{x}, \mathrm{D}) \leftarrow MSE(\mathrm{M}, \mathrm{D})$
$\mathbf{x}_i \leftarrow \arg\max_{\mathbf{x} \in \mathrm{X}} S(\mathbf{x}, p(y \mid \mathbf{x}, \mathrm{D}))$
$y_i \leftarrow f(\mathbf{x}_i)$ > Expensive step
$\mathrm{D} \leftarrow \mathrm{D} \cup (\mathrm{x}_i, y_i)$
end for

where f represents the input-output mapping relationship of the problem to be solved, also known as the black box. X is the domain of the independent variables, i.e., the search space of the parameters. This dataset named D comprises multiple pairs of data, with each pair represented as (x, y), where x is the input and y is the output. The acquisition function, denoted as f, determines the subsequent input combination x to be observed. The model fitted to the dataset is represented by M, and various types of models can be employed to characterize this model, such as Gaussian processes, random forests, Tree-Parzen Estimators, etc. Here, we mainly discuss Gaussian processes. This assumption is a prior that is incorporated into the subsequent optimization process. The algorithm initialization involves obtaining an initial batch of data $\mathrm{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where $y_i = f(x_i)$ to provide an initial estimate of M. Note that the initial data collection does not yet use the acquisition function. With each iteration, the existing observations (represented by the dataset D) become the prior. Based on the prior and the search space, a point to be observed is determined, usually a point that balances the needs of exploration and exploitation. Subsequently, the value function or cost function f is calculated. Since each calculation is costly, we typically set a fixed number of iterations, denoted as T.Next, based on our use of the model M, the dataset D is fitted with the model, which includes our prior assumptions about the problem, such as Gaussian processes, random forests, etc. After fitting, we obtain a

relationship between input and output $p(y|\ x,\mathrm{D})$ based on observations/prior, which is M . Here, Gaussian processes are most commonly used for modeling and fitting:

$$p(y|\ x,\mathrm{D}) \leftarrow \mathrm{FitModel}(\mathrm{M},\mathrm{D}) \tag{25}$$

In the next step, we need to determine which point (input combination) to observe in the current iteration, which requires the use of the acquisition function S . The acquisition function needs to balance exploration and exploitation to determine where to observe next. There are several popular methods for calculating the acquisition function, and the observation point is determined by the acquisition function.

$$x_i \leftarrow \arg\max_{x\in\mathrm{X}} S(x, p(y|\ x,\mathrm{D})) \tag{26}$$

After determining the observation point, we input this input combination into the system (function) to obtain the output (value/cost).

$$y_i \leftarrow f(x_i) \tag{27}$$

This input-output combination can then be used to further update the observations D :

$$\mathrm{D} \leftarrow \mathrm{D} \cup (x_i, y_i) \tag{28}$$

This is a brief introduction to the Bayesian optimization algorithm.

5. Experiments

The model algorithms were implemented on the MATLAB programming platform by the researchers. The experimental results indicated that the TSVR model achieved the smallest MSE on eleven datasets compared to the other four models, suggesting that the TSVR model had better predictive performance on these eleven datasets. To more intuitively present these results, the MSE values of each model on the test sets of all datasets were listed in Table 1 by the researchers, with the model having the smallest MSE on each dataset highlighted in bold

Table 1. Comparison of Model Performance on Synthetic Datasets.

Model	TSVR	Ridge	ELM	ElasticNet	ASVR
Sinc	**1.88E-04**	5.43E-02	1.70E-03	5.43E-02	8.17E-04
NoName	**6.98E-05**	4.51E-02	6.22E-03	4.63E-02	3.39E-03
2dMexicanhat	**1.30E-04**	9.41E-02	1.32E-04	9.41E-02	6.75E-04
3dMexicanhat	**2.12E-03**	1.22E-02	1.60E-02	1.22E-02	3.38E-03
Friedman#3	**4.88E-03**	1.90E-02	6.02E-03	2.12E-02	6.58E-03
Multi	**2.25E-05**	3.34E-03	5.80E-05	3.34E-03	2.25E-03
F1	**1.77E-03**	6.04E-02	3.83E-02	5.90E-02	3.01E-03
F2	**5.12E-04**	3.50E-02	2.07E-03	3.50E-02	1.95E-03
F3	**8.00E-05**	3.22E-02	3.72E-03	3.35E-02	3.96E-03
F4	**2.16E-04**	9.47E-02	3.13E-02	9.73E-02	4.18E-03
F5	**3.63E-05**	4.80E-02	3.85E-02	4.80E-02	4.51E-03

According to the table, it is evident that the TSVR model achieved the smallest MSE on datasets those on the Table 1. The particular functions associated with these eleven datasets are detailed in the table below.

The experimental results clearly demonstrate the superiority of the TSVR model optimized by Bayesian optimization. The significantly lower MSE values achieved by TSVR on the majority of datasets indicate its enhanced predictive accuracy and generalization capability. This is particularly evident in complex datasets such as the 3dMexicanhat and Friedman#3, where TSVR outperformed other models by a substantial margin. The consistent performance across various datasets highlights

the robustness of the TSVR model and the effectiveness of the Bayesian optimization approach in tuning its hyperparameters. Overall, these findings suggest that the optimized TSVR model is well-suited for handling a wide range of regression tasks, especially those involving complex and non-linear relationships.

Table 2. Correspondence between Datasets and Functions.

Datasets	Corresponder Function
Sinc	$y = \sin x / x$
NoName	$y = \lvert(x-1)/4\rvert + \lvert\sin\left(\pi\left(1+(x-1)/4\right)\right)\rvert + 1$
2dMexicanhat	$y = \sin(\lvert x\rvert)/\lvert x\rvert$
3dMexicanhat	$y = \sin\left(\sqrt{\sum x_i^2}\right)/\sqrt{\sum x_i^2}$
Friedman#3	$y = \arctan\left(\left(x_2 x_3 - 1/x_2 x_4\right)/x_1\right)$
Multi	$y = 0.79 + 1.27\,x_1 x_2 + 1.56\,x_1 x_4 + 3.42\,x_4 x_5 + 2.06\,x_3 x_4 x_5$
F1	$y = \sin(x)\cdot\cos(x^2)$
F2	$y = e^{x_1 \sin(\pi x_2)}$
F3	$y = 1.9\left(1.35 + e^{x_1}\sin(13(x_1-0.6)^2) + e^{3(x_2-0.5)}\sin(4\pi(x_2-0.9)^2)\right)$
F4	$y = \sin\left(2\pi(0.35\cdot 10+1)/\left(0.35x+1\right)\right)$
F5	$y = 4/\left(\lvert x\rvert+2\right) + \cos(2x) + \sin(3x)$

6. Conclusions

This study systematically explores the application of Bayesian optimization for hyperparameter tuning in Twin Support Vector Regression (TSVR) models. Through extensive experiments on complex synthetic datasets, the optimized TSVR model demonstrates significantly lower Mean Squared Error (MSE) compared to other regression models, such as Support Vector Regression (SVR), Extreme Learning Machine (ELM), Ridge regression, and ElasticNet. This highlights the superior predictive accuracy and generalization capability of TSVR in handling complex data. The use of Bayesian optimization not only enhances model performance but also streamlines the hyperparameter tuning process, saving time and effort in manual adjustments. The results further confirm the robustness and efficiency of TSVR in various datasets, solidifying its potential for practical applications.

The application prospects of TSVR optimized by Bayesian optimization appear promising. As machine learning and artificial intelligence continue to advance, the demand for efficient and accurate regression models will only increase. TSVR, with its enhanced computational efficiency and strong generalization ability, is well-suited for complex regression tasks in fields such as finance, healthcare, and environmental science. Moreover, the integration of Bayesian optimization provides a reliable and automated approach to hyperparameter tuning, which can be extended to other machine learning models and applications. Future research may focus on exploring more sophisticated optimization techniques and expanding the application scope of TSVR to real-world problems with high-dimensional and noisy data.

References

[1] Vapnik V N. Statistical learning theory Wiley, 1998.

[2] Deng N Y, Tian Y J, Zhang C H. Support vector machines: theory, algorithms, and extensions CRC Press, 2012.

[3] Suykens J A K, Lukas L, van Dooren P, De Moor B, Vandewalle J. Least squares support vector machine classifiers: a large-scale algorithm Proceedings of European conference of circuit theory design, 1999: 839–844.

[4] Mangasarian O L, Wild E W. Multisurface proximal support vector classification via generalized eigenvalues IEEE Trans Pattern Anal Mach Intell, 2006, 28(1): 69–74.

[5] Peng X. Primal twin support vector regression and its sparse approximation Neurocomputing, 2010, 73(16–18): 2846–2858.

[6] Shao Y H, Deng N Y. A novel margin-based twin support vector machine with unity norm hyperplanes Neural Comput Appl, 2012.

[7] Peng X. TPMSVM: a novel twin parametric-margin support vector machine for pattern recognition Pattern Recognit, 2011, 44(10–11): 2678–2692.

[8] Chao Yuan, Liming Yang. Capped L2, p-norm metric based robust least squares twin support vector machine for pattern classification. *Neural Networks*, 2021, 142: 457-478.

[9] Jun Ma, Liming Yang, Qun Sun. Capped L1-norm distance metric-based fast robust twin bounded support vector machine. Neurocomputing, 2020, 412: 295-311.

[10] Lan Bai, Xu Chen, Zhen Wang, Yuan-Hai Shao. Safe intuitionistic fuzzy twin support vector machine for semi-supervised learning. *Applied Soft Computing*, 2022, 123: 108906.

[11] Ran An, Yitian Xu, Xuhua Liu. Multi-task twin bounded support vector machine and its safe screening rule. *Applied Soft Computing*, 2023, 138: 110188.

[12] Jimenez-C et al. Krein twin support vector machines for imbalanced data classification

[13] Pattern Recognition Letters, 2024, 182: 39-45.

[14] Si, Q., Yang, Z., Ye, J. Symmetric LINEX loss twin support vector machine for robust classification and its fast iterative algorithm Neural Networks, 2023, 168: 143-160.

[15] M.A. Ganaie, Vrushank Ahire, Anouck Girard. Granular Ball K-Class Twin Support Vector Classifier. *Pattern Recognition*, 2025, 166: 111636.

[16] Suykens J A K, Vandewalle J. Least squares support vector machine classifiers Neural Process Lett, 1999, 9(3): 293–300.

[17] Peng X. TSVR: an efficient twin support vector machine for regression Neural Netw, 2010, 23(3): 365–372.

[18] Anuradha Kumari, M. Tanveer. LSTSVR+: Least square twin support vector regression with privileged information. *Engineering Applications of Artificial Intelligence*, 2024, 136: 108964.

[19] Liu C. Twin proximal support vector regression with heteroscedastic Gaussian noise. Expert Syst Appl, 2024, 250: 123840.

[20] Yafen Ye et al. Twin support vector quantile regression Expert Systems with Applications, 2024.

[21] Salim Rezvani, Xizhao Wang. Intuitionistic fuzzy twin support vector machines for imbalanced data. *Neurocomputing*, 2022, 507: 16-25.

[22] Jiamin Xu, Huamin Wang, Libo Zhang, Shiping Wen. Robust twin depth support vector machine based on average depth. *Knowledge-Based Systems*, 2023, 274: 110627.

[23] Chengjiang Zhou, Hao Li, Jintao Yang, Qihua Yang, Limiao Yang, Shanyou He, Xuyi Yuan. Fuzzy regular least squares twin support vector machine and its application in fault diagnosis. *Expert Systems with Applications*, 2023, 231: 120804.

[24] Qianfei Liu et al. Multi-view structural twin support vector machine with the consensus and complementarity principles and its safe screening rules Expert Systems with Applications, 2025.

[25] Zhiyong Che et al. Twin support vector machines with privileged information Information Sciences, 2021.

[26] Ye Y. L1-norm twin support vector quantile regression. Appl Soft Comput, 2025, 169: 112580.

[27] Richhariya B. A reduced universum twin support vector machine for class imbalance learning. Pattern Recognit, 2020,102: 107150.

[28] Liu, B., Huang, R., Xiao, Y., Liu, J., Wang, K., Li, L., Chen, Q. Adaptive robust Adaboost-based twin support vector machine with universum data Information Sciences, 2022, 609: 1334-1352.

[29] Xie, X., Li, Y., Sun, S. Deep multi-view multiclass twin support vector machines Information Fusion, 2023, 91: 80-92.

[30] Lu, J., Xie, X., Xiong, Y. Multi-view hypergraph regularized Lp norm least squares twin support vector machines for semi-supervised learning. *Pattern Recognition*, 2024, 156: 110753.

[31] Huang, H., Wei, X., Zhou, Y. An overview on twin support vector regression Neurocomputing, 2022, 490: 80-92.