Research On Twin Support Vector Regression by Hiker Optimization Algorithm

Li Su *

Department of Basic, China Fire and Rescue Institute, Beijing, China, 102202

* Corresponding Author Email: suli125@163.com

Abstract. This paper presents an enhanced twin support vector regression (TSVR) model integrated with the hiker optimization algorithm (HOA) and successive 0ver-relaxation (SOR) method to address challenges in regression accuracy and computational efficiency. HOA is employed to optimize hyperparameters, while SOR accelerates the solution of quadratic programming problems by reducing memory usage for sparse matrices. Experimental results on 8 synthetic test functions with diverse noise types and 18 benchmark datasets demonstrate that the proposed HOA-TSVR significantly outperforms traditional models. However, performance fluctuations on high-dimensional small-sample datasets highlight sensitivities to data distribution and feature redundancy. The study concludes that the hybrid approach enhances generalization and robustness, offering a promising framework for complex data modeling, with future research directions focusing on adaptive feature selection and multi-scenario applications.

Keywords: Twin Support Vector Regression, Successive Over-Relaxation, Hyperparameter Optimization.

1. Introduction

The Support Vector Machines (SVMs), serve as a cornerstone in machine learning. However, they suffer from high computational complexity. To address this, Jayadeva et al. proposed the Twin Support Vector Machine (TSVM), reducing the complexity to one-fourth of the traditional SVM and spurring related research. In practical applications, many issues require not only data classification but also urgent predictions of continuous data, based on this, the application scope of SVM has been extended from classification problems to regression problems, giving rise to the support vector regression (SVR). To tackle SVR's complexity, Peng [1] proposed the Twin Support Vector Regression (TSVR) in 2010. Rooted in statistical learning theory, TSVR maps samples to a high-dimensional space to achieve linear separability, defines regression boundaries with a pair of hyperplanes, and solves via dual quadratic programming. Nevertheless, the strong coupling between its penalty factor and kernel function risks local sub-optimization, while the sparse Hessian matrix hampers solution efficiency.

Therefore, the selection of hyperparameters is a matter that must be carefully considered for TSVR. During the early stages of hyperparameter selection, researchers primarily explored classical optimization algorithms, such as sequential quadratic programming, quasi-Newton conjugate gradient methods, and fast steepest descent methods. These algorithms demonstrating high efficiency in solving specific problems. However classical optimization algorithms have limited optimization capabilities due to their strict mathematical requirements for problems and high sensitivity to initial parameter settings. At this time, meta-heuristic algorithms have come into the sight of relevant researchers due to their characteristics of intuitive principles, simple structures, no need for complex mathematical modeling and assumptions of problems, and easy programming implementation. Meta-heuristic algorithms can be roughly divided into (i) meta-heuristic algorithms based on a single solution and (ii) meta-heuristic algorithms based on a population according to the type of problem they solve. Currently, popular methods based on a single solution include the Adaptive Large Neighborhood Search algorithm (ALNS), the Iterated Local Search algorithm (ILS) [2], and the Extremal Optimization algorithm (EO) [3]. Meta-heuristic algorithms based on a population show a vibrant trend. In the past five years, for example, the Red-billed Blue Magpie Optimization algorithm

(RMBO) [4], the Wave Search algorithm (WSA) [5], the Artemisinin Optimization algorithm (AO) [6], the Goose Optimization algorithm (GOOSE) [7], the Puma Optimization algorithm (PO) [8], and the Hiker Optimization algorithm (HOA) [9], etc.

The HOA algorithm, by simulating the behavior of hikers adapting and adjusting their paths, can explore the solution space more efficiently during global search. Meanwhile, it exhibits excellent convergence speed. When dealing with hyperparameter optimization problems of complex models such as TSVR, it can quickly find better solutions. Considering classical optimization algorithms' inherent defects in TSVR hyperparameter selection and meta-heuristic algorithms' advantages in complex optimization, we use the HOA algorithm to optimize the TSVR model.

Another critical issue that demands careful consideration in the context of TSVR is the computational efficiency. Conventional TSVR implementations in MATLAB typically rely on built-in quadratic programming (QP) solvers, which necessitate the storage of quadratic matrices during computation. When dealing with large-scale datasets, this approach leads to excessive memory consumption and diminished computational efficiency. To address this challenge, the Successive Over-Relaxation (SOR) iterative method emerges as a promising alternative. The SOR method offers a significant advantage in that it only requires the storage of a single row vector at each iteration, drastically reducing memory requirements while maintaining computational stability.

This paper presents a TSVR model that utilizes the HOA algorithm to optimize hyperparameters, and employs the SOR method to accelerate computational efficiency in solving QP problems.This work includes:

(1) Optimizing the hyperparameters of five models, namely TSVR, SVR, Extreme Learning Machine (ELM), Ridge, and Elastic Net, using the HOA algorithm.

(2) Utilizing SOR to accelerate convergence when TSVR solves the QP problem.

(3) Testing the fitting accuracy of each model on eight test functions and in five different noise environments.

(4) Testing the performance of each model on 18 benchmark datasets.

(5) Summarizing the above improvement strategies and giving the future research directions.

2. Mathematical Models

2.1. Basic SVR Model

SVR is an improvement based on SVM. It uses the kernel function to map data into a high-dimensional space for regression and has good robustness and prediction performance. The classical SVR model is as follows. Suppose a set of independent and identically distributed training samples is given:

$D = \{(\boldsymbol{x}_i, y_i) \big| \boldsymbol{x}_i \in \square^d, y_i \in \square\}, i = 1, 2, \cdots, m.$ And the set of hypothesis functions: $F = \{f \big| f : \square^d \to \square\}$. The goal of the regression problem is to find a function $f \in F$, through the training of samples, so that the error between the value of the training sample under this function and its actual value is not greater than a given deviation ε. Let the function f have the following linear form (for the non-linear case, it can be extended through the kernel function):

$$f(\boldsymbol{x}) = \boldsymbol{W} \cdot \boldsymbol{x} + b, \tag{1}$$

where $\boldsymbol{W} \in \square^d, b \in \square$ represent the threshold. Solving the function f is reduced to the following programming problem:

$$\begin{aligned} &\min \frac{1}{2}\|\boldsymbol{W}\|^2 + C\sum_{i=1}^{m}(x_i, x_i^*) \\ &s.t.\ \ y_i - (W^T\Phi(\boldsymbol{x}_i) + b) \le e + x_i, (W^T\Phi(\boldsymbol{x}_i) + b) - y_i \le e + x_i^*, \\ &\quad x_i, x_i^* \ge 0, i = 1, 2, \cdots, m, \end{aligned} \tag{2}$$

where C is the penalty coefficient, andx_i ,x_i^* are the slack variables. The dual problem of (2) can be used to solve it:

$$\begin{aligned} L(\boldsymbol{W},b,x^*,a) &= \frac{1}{2}\boldsymbol{W}^T\boldsymbol{W} + C\sum_{i=1}^{l}(x_i + x_i^*) - \sum_{i=1}^{l}(h_i x_i + h_i^* x_i^*) \\ &- \sum_{i=1}^{l} a_i[e + x_i + y_i - \boldsymbol{W}^T\Phi(\boldsymbol{x}_i) - b] - \sum_{i=1}^{l} a_i^*[e + x_i^* - y_i + \boldsymbol{W}^T\Phi(\boldsymbol{x}_i) - b]. \end{aligned} \tag{3}$$

By taking the partial derivatives of $\boldsymbol{W},b,x^*$ in the above formula and setting them to 0, and substituting them into formula (3), we can get

$$L(\boldsymbol{W},b,x^*,a) = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(a_i^* - a_i)(a_j^* - a_j)Ker(x_i \cdot x_j) - e\sum_{i=1}^{m}(a_i^* + a_i) + \sum_{i=1}^{m}(a_i^* - a_i). \tag{4}$$

Now,(2) becomes

$$\begin{aligned} &\min \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}(a_i - a_i^*)(a_j - a_j^*)Ker(x_i \cdot x_j) + e\sum_{i=1}^{m}(a_i + a_i^*) - \sum_{i=1}^{m} y_i(a_i - a_i^*) \\ &s.t. \quad 0 \le a_i + a_i^* \le C, \sum_{i=1}^{m}(a_i - a_i^*) = 0, i = 1,2,\cdots,m. \end{aligned} \tag{5}$$

This quadratic programming problem employs the kernel function $Ker(x_i \cdot x_j)$. Solving for a_i and a_i^* yields the objective function value.

2.2. The Twin Support Vector Regression

Peng's TSVR (2010) improves SVR training efficiency using two non-parallel functions around the training data, which set the ε-insensitive upper and lower bounds for the target function.

For the linear case, TSVR determines the final target function through the e_1-insensitive lower bound of the training data.

$$F_1(x) = \boldsymbol{W}_1^T x + b_1, \tag{6}$$

and the e_2-insensitive upper bound

$$F_2(x) = \boldsymbol{W}_2^T x + b_2, \tag{7}$$

we can solve the following quadratic programming problems to obtain the solutions of the above functions.

$$\begin{aligned} &\min \ \frac{1}{2}\left\|Y - e\varepsilon_1 - (A\boldsymbol{W}_1 + eb_1)\right\|^2 + C_1 e^T\xi^* \\ &s.t. \quad Y - (A\boldsymbol{W}_1 + eb_1) \ge e\varepsilon_1 - \xi^*, \xi^* \ge 0. \end{aligned} \tag{8}$$

$$\begin{aligned} &\min \ \frac{1}{2}\left\|Y + e\varepsilon_2 - (A\boldsymbol{W}_2 + eb_2)\right\|^2 + C_2 e^T\eta^* \\ &s.t. \quad (A\boldsymbol{W}_2 + eb_2) - Y \ ? \ e\varepsilon_2 \quad \eta^*, \eta^* \ ? \ 0. \end{aligned} \tag{9}$$

Where $C_1, C_2 > 0$ and $\varepsilon_1, \varepsilon_2 \ge 0$ are predefined parameters with slack vectors ξ^* and η^*. Formulas (8) and (9) have only half the constraints of standard SVR, Consequently, TSVR trains at least four times faster than SVR. The absence of equality constraints in these formulations further accelerates TSVR training.

Introducing Lagrange multiplier vectors α and β, and applying the KKT conditions (Karush-Kuhn-Tucker), yields the dual optimization problems of formulas (8) and (9):

$$\max \ -\frac{1}{2}\alpha^T G(G^T G)^{-1}G^T\alpha + F^T G(G^T G)^{-1}G^T\alpha - F^T\alpha \tag{10}$$
$$s.t. \quad 0 \le \alpha \le C_1 e,$$

$$\max \ -\frac{1}{2}\beta^T G(G^T G)^{-1}G^T\beta - H^T G(G^T G)^{-1}G^T\beta - H^T\beta \tag{11}$$
$$s.t. \quad 0 \le \beta \le C_2 e,$$

where $G=[A \quad e], F=Y-\varepsilon_1 e, H=Y+\varepsilon_2 e$. Optimization yields the objective function:

$$F(x)=\frac{1}{2}(F_1(x)+F_2(x))=\frac{1}{2}(W_1+W_2)^T x+\frac{1}{2}(b_1+b_2). \tag{12}$$

For nonlinear cases, the kernel function is applied:

$$F_1(x)=Ker(x^T, A^T)\boldsymbol{W}_1+b_1, F_2(x)=Ker(x^T, A^T)\boldsymbol{W}_2+b_2. \tag{13}$$

Analogous to the linear case, solving this dual problem pair derives Equation (13):

$$\min \ \frac{1}{2}\left\|F-(Ker(x^T, A^T)\boldsymbol{W}_1+eb_1)\right\|^2 + C_1 e^T\xi^* \tag{14}$$
$$s.t. \quad Y-(Ker(x^T, A^T)\boldsymbol{W}_1+eb_1) \ge e\varepsilon_1-\xi^*, \xi^* \ge 0,$$

$$\min \ \frac{1}{2}\left\|H-(Ker(x^T, A^T)\boldsymbol{W}_2+eb_2)\right\|^2 + C_2 e^T\eta^* \tag{15}$$
$$s.t. \quad (Ker(x^T, A^T)\boldsymbol{W}_2+eb_2)-Y \ge e\varepsilon_2-\eta^*, \eta^* \ge 0,$$

here $V=[\ Ker(x^T, A^T) \quad e]$, and the objective function can be rewritten in the following vector form:

$$F(x)=\frac{1}{2}Ker(x^T, A)(\boldsymbol{W}_1+\boldsymbol{W}_2)^T x+\frac{1}{2}(b_1+b_2), \tag{16}$$

where $[\boldsymbol{W}_1^T \quad b_1]^T=(V^T V)^{-1}V^T(F-\alpha), [\boldsymbol{W}_2^T \quad b_2]^T=(V^T V)^{-1}V^T(H-\beta)$.

For the linear TSVR model, the hyperparameters are the penalty factor C_1, C_2 and the insensitive upper and lower bounds $\varepsilon_1, \varepsilon_2$. For nonlinear models, hyperparameters also include the bandwidth parameter σ of Gaussian kernel.

2.3. The Successive Over-Relaxation Algorithm

Solving the quadratic programming problem in TSVR involves the inverse operation of large-scale sparse matrices. We introduce the successive over-relaxation method to improve computational efficiency. This is an efficient iterative algorithm for solving large-scale sparse linear equations $Ax=b$. It is based on the Gauss-Seidel iteration method and accelerates the convergence speed by introducing a relaxation factor ω.

For the linear equation $Ax=b$, decompose A as: $A=D-L-U$, where D, L ,and U are diagonal,strictly lower triangular and strictly upper triangular parts of A, respectively.

The Gauss-Seidel iteration is: $x^{(k+1)}=(D-L)^{-1}(Ux^{(k)}+b)$, By introducing the relaxation factor ω (usually $1<\omega<2$), after taking the weighted average of the Gauss-Seidel iteration result and the new step, we have

$$x_i^{(k+1)}=(1-\omega)x_i^{(k)}+\frac{\omega}{a_{ii}}\cdot(b_i-\sum_{j=1}^{i-1}a_{ij}x_j^{(k+1)}-\sum_{j=i+1}^{n}a_{ij}x_j^{(k)}), \quad i=1,2,\mathrm{L}\ ,n. \tag{17}$$

When $\omega=1$, it degenerates into the Gauss-Seidel method, when $\omega>1$, the iteration is accelerated, which is suitable for diagonally dominant matrices, when $\omega<1$, it may suppress divergence, but the convergence speed slows down.

3. Hyperparameter Optimization

In order to solve the problem that the objective function of the TSVR algorithm falls into a locally suboptimal state due to the strong coupling between the penalty factor and the kernel function, we innovatively use HOA optimization algorithm in the selection of the penalty factor. This is because HOA outperforms several classical optimization algorithms such as TLBO, GA, DE and GWO on multiple test functions and some engineering problems.

The mathematical model of HOA is based on Tobler's Hiking Function (THF) [10]. The walking speed of hikers (i.e., agents) is determined by the terrain slope and the distance traveled.

$$v_{i,t}=6e^{-3.5|s+0.05|}, \tag{18}$$

Here $v_{i,t}$ denotes the speed of hiker *i* at iteration or time *t* (i.e., in km/h), and s is the slope of the trail or terrain. The slope s is calculated via the following formula:

$$s=\frac{\Delta h}{d}=\tan\theta, \tag{19}$$

where Δh and d signify the height difference and distance covered by the hiker, respectively. Moreover, θ is the inclination angle of the trail or terrain, with its range being $[0,\frac{\pi}{3}]$.

The updated or actual speed of a hiker is a function of the initial velocity (determined by THF), the position of the lead hiker, the current hiker position and the sweep factor. Hence, the current speed of the hiker *i* is given by

$$v_{i,t+1}=v_{i,t}+rand(0,1)\times(v_{i,0}-v_{i,t})\times\frac{\left|x_{l,t}-x_{i,t}\right|}{\left|x_{l,t}-x_{i,t}\right|+SF_i}. \tag{20}$$

Here $rand(0,1)$ denotes a uniformly random number in [0,1]; while $v_{i,t}$ and $v_{i,t+1}$ represent the initial and updated speeds of hiker *i*, respectively. The leading hiker's position is $x_{l,t}$ and $SF_i\in[1,3]$ is hiker *i's* sweep factor (SF). Thisfactor prevents excessive distancing from the leader, ensuring visibility of movement direction and signal reception.

Considering the the hiker speed in the above formula (18), the updated location of hiker *i* is derived by

$$x_{i,t+1}=x_{i,t}+v_{i,t+1}\times\Delta t. \tag{21}$$

Among diverse meta-heuristic algorithms, such as the HOA, the initial configuration of agents holds paramount importance, as it exerts a substantial influence on the acquisition of viable solutions and the convergence rate. In this paper, the HOA utilizes the random initialization method to set the starting positions of its agents.

Hiker positions are initialized using the solution's bounds:

$$x_{i,t,0}=x_j^{\min}+rand(0,1)\times(x_j^{\max}-x_j^{\min}), \tag{22}$$

where $rand(0,1)$ is a uniformly distribution number within the range [0,1].

The algorithm's core loop per iteration:

Initialize all hiker positions using Tobler's Hiking Function.

Update hiker positions based on a given time step.

Evaluate all hikers' fitness; the highest-fitness hiker becomes the leader.

The leading hiker position is reassessed after each iteration to ensure optimal fitness. This continuous update and evaluation process exemplifies a global optimization algorithm, systematically searching for the global optimum within the problem space.

4. Experiments and Analysis

4.1. Evaluation Criteria and Experimental Steps

To assess the efficacy of the approach put forward in this study, we compared TSVR with some classical regression algorithms, such as SVR, Extreme Learning Machine (abbreviated as ELM), Ridge, and Elastic Net, on some artificial datasets and Benchmark datasets. All experiments in this paper were implemented in MATLAB(2022b), with the operating environment of Windows 11 system, 32GB of memory, and a CPU main frequency of 2.60 GHz. Since the learning performance of the algorithm is affected by parameters, only the Gaussian kernel $K(x_i \cdot x_j) = \exp(-\frac{\|x_i - x_j\|^2}{2\sigma^2})$ is considered as the kernel function here to evaluate the performance of the regression machine.

All parameters of the algorithm were verified using 1000 randomly selected data points, with 80% as training data and 20% as test data, obtained through 10-fold cross-validation. The parameter values were set as $C_1 = C_2$ and $\varepsilon_1 = \varepsilon_2$ in the experiment.

4.2. Artificial Datasets

In this section, we tested the TSVR algorithm with hyperparameters optimized by HOA against the traditional SVR, ELM, Ridge, and Elastic Net, which also had their hyperparameters optimized in the same way, on 8 functions. The test functions after adding noise are shown in Table 1 and the specific noise ε_i types in Table 1 are given in Table 2.The benchmark functions covered linear, nonlinear, periodic, and high-dimensional interaction functions. Among then ,$U(a,b)$ represents a uniform random variable on $[a,b]$ and $N(\mu,\sigma^2)$ represents a Gaussian random variable with mean μ and variance σ^2. Table 3 shows the Mean Squared Error(MSE) values of various models under different noise types after 10-fold cross-validation.

Table 1: Function Expressions

Function	Function Expressions	Upper and Lower Bound
Sinc	$y = \sin x/x + \varepsilon_i$	$x:\ U[-4\pi, 4\pi]$
NoName	$y = \lvert\sin x/x\rvert + \varepsilon_i$	$x:\ U[-10,10]$
2dMexicanhat	$y = \lvert(x-1)/4\rvert + \lvert\sin\left(\pi\left(1+(x-1)/4\right)\right)\rvert + 1 + \varepsilon_i$	$x:\ U[-2\pi, 2\pi]$
3dMexicanhat	$y = \sin\left(sqrt\left(\sum_{i=1}^{d} x_i^2\right)\right) \Big/ sqrt\left(\sum_{i=1}^{d} x_i^2\right) + \varepsilon_i\ (d \geq 1)$	$x_i:\ U[-2\pi, 2\pi], \forall i$
Friedman#1	$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \varepsilon_i$	$x_i:\ U[0,1], \forall i$
Polynomial	$y = 1 + 2x + 3x^2 + 4x^3 + 5x^4 + \varepsilon_i$	$x:\ U[0,1]$
F1	$y = \exp(x_1 \sin(\pi x_2) + \varepsilon_i$	$x_i:\ U[0,10], \forall i$
F2	$y = 4/\left(\lvert x\rvert + 2\right) + \cos\left(2x\right) + \sin\left(3x\right) + \varepsilon_i$	$x:\ U[0,6]$

Table 2: Noise Types

Type A	Type B	Type C	Type D	Type E
$\varepsilon_1:\ U[-0.1, 0.1]$	$\varepsilon_2:\ U[-0.2, 0.2]$	$\varepsilon_3:\ U[-0.3, 0.3]$	$\varepsilon_4:\ N(0, 0.1^2)$	$\varepsilon_5:\ N(0,1)$

Table 3: Simulation Results of Five Algorithms on Eight Functions under Different Noise Levels

Function Name	Model	Type A MSE	Type B MSE	Type C MSE	Type D MSE	Type E MSE
Sinc	TSVR	2.49E-02	3.96E-02	7.45E-03	6.84E-02	2.81E-03
	SVR	4.35E-02	6.48E-02	7.47E-02	6.94E-02	5.83E-02
	ELM	7.83E-02	8.78E-02	7.40E-02	3.49E-02	4.70E-02
	Ridge	5.16E-02	6.45E-02	1.53E-02	8.36E-02	8.35E-02
	ElasticNet	4.88E-02	8.66E-02	9.93E-02	5.92E-02	5.48E-02
NoName	TSVR	2.76E-02	9.97E-03	9.48E-03	9.85E-02	1.05E-02
	SVR	7.36E-03	7.92E-03	8.52E-04	4.34E-02	4.53E-02
	ELM	1.65E-02	7.44E-02	2.14E-02	3.81E-02	9.05E-02
	Ridge	2.27E-02	6.68E-02	8.40E-02	6.19E-02	7.45E-02
	ElasticNet	7.51E-02	9.01E-02	4.08E-02	2.13E-02	6.39E-02
2dMexicanhat	TSVR	2.47E-02	4.58E-02	2.94E-02	2.69E-03	7.98E-02
	SVR	2.83E-02	2.08E-02	8.24E-02	5.44E-02	3.75E-02
	ELM	3.95E-02	3.19E-02	9.84E-02	8.23E-02	6.36E-02
	Ridge	9.28E-02	1.17E-02	1.13E-02	4.02E-03	6.63E-02
	ElasticNet	7.83E-02	3.14E-02	2.03E-02	3.39E-02	8.87E-02
3dMexicanhat	TSVR	9.35E-02	3.18E-02	2.92E-02	8.76E-03	4.97E-03
	SVR	4.48E-02	6.08E-02	4.78E-02	7.20E-02	6.44E-02
	ELM	7.26E-02	9.29E-02	8.61E-02	5.22E-03	2.58E-02
	Ridge	6.59E-02	8.88E-02	7.67E-02	2.09E-02	4.64E-02
	ElasticNet	7.53E-02	4.68E-03	3.78E-03	2.71E-02	9.87E-02
Friedman#1	TSVR	4.42E-02	2.62E-02	5.93E-02	1.39E-02	6.43E-02
	SVR	2.39E-02	9.21E-02	1.26E-02	3.94E-02	9.23E-02
	ELM	5.25E-02	7.98E-02	6.42E-02	3.29E-02	7.15E-02
	Ridge	1.09E-02	6.15E-02	4.52E-02	7.54E-02	3.84E-02
	ElasticNet	5.30E-02	4.70E-02	4.92E-03	1.62E-02	9.00E-03
Polynomial	TSVR	3.32E-02	7.73E-03	1.33E-02	2.70E-02	1.84E-02
	SVR	2.19E-02	2.44E-02	7.05E-02	8.42E-02	3.59E-02
	ELM	4.77E-02	5.18E-02	6.50E-02	6.41E-02	3.00E-02
	Ridge	2.68E-02	2.98E-02	4.49E-02	9.16E-02	3.24E-02
	ElasticNet	5.52E-02	9.92E-02	1.69E-02	6.35E-02	6.86E-02
F1	TSVR	2.86E-02	2.99E-03	4.44E-02	2.84E-02	5.26E-02
	SVR	3.50E-02	1.19E-02	7.89E-02	5.44E-02	1.12E-02
	ELM	3.08E-02	7.93E-02	8.90E-02	2.00E-02	3.48E-02
	Ridge	2.40E-02	8.77E-02	5.15E-02	4.79E-02	4.83E-02
	ElasticNet	2.36E-03	8.82E-02	5.16E-02	6.81E-02	5.71E-03
F2	TSVR	4.12E-02	4.55E-02	1.06E-02	6.36E-02	8.00E-02
	SVR	3.20E-02	5.88E-02	8.87E-02	7.74E-02	9.88E-03
	ELM	8.92E-02	6.98E-02	4.01E-02	8.08E-02	5.58E-02
	Ridge	6.23E-02	4.33E-02	9.90E-02	6.31E-02	2.66E-02
	ElasticNet	3.11E-02	2.60E-02	8.34E-02	7.31E-02	3.36E-02

As shown in Table 3, the performance of TSVR on eight different test functions and five different types of noise is generally superior to that of the other four models. The performance improvement of TSVR compared with other models on the test functions is shown in Figure 1. In terms of the performance on the 3dMexicanhat, 2dMexicanhat, and NoName functions, TSVR exhibits significant improvements over the other four baseline models in both training and test sets. However, for the Sinc function (a low-dimensional smooth function), TSVR showed negative train and test improvement percentages compared to SVR. This may be attributed to SVR's better alignment with the kernel function assumptions of low-dimensional smooth functions. Additionally, on the F1 function, although TSVR achieved an extremely low Train MSE, its test improvement percentage was negative(vs SVR), potentially due to overfitting leading to reduced generalization ability.

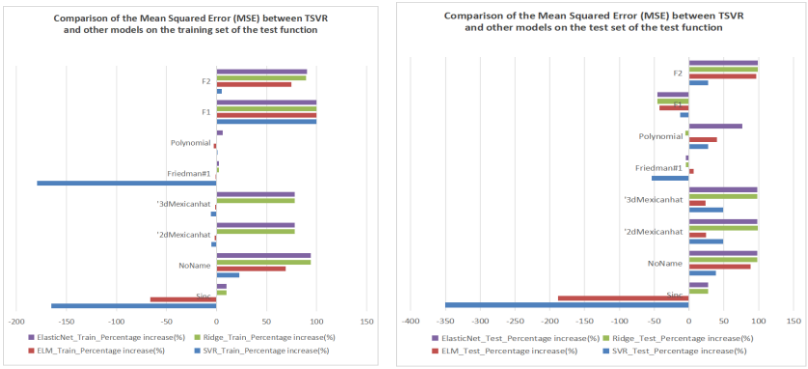

Figure 1: Comparison of MSE values of TSVR and other models on test functions.

4.3. Bechmark Datasets

In this section, we continue to test the model's performance on 18 benchmark datasets. Data source: https://www.openml.org/. The test samples also account for 80% of the total sample size. The experimental results are shown in Table 4.

Table 4: Empirical Analysis of Five Regression Methods over 18 Datasets

Dateset	TSVR		SVR		ELM		Ridge		ElasticNet	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Stoke (950*9)	1.99E-04	5.87E-04	2.26E-03	2.08E-03	4.47E-04	1.07E-03	6.90E-03	6.81E-03	6.90E-03	6.82E-03
arsenic_female_bladder (559*4)	9.07E-05	1.10E-03	2.35E-03	1.99E-03	1.01E-04	3.87E-03	6.22E-03	3.22E-03	6.70E-03	1.95E-03
auto_price (159*15)	7.80E-04	9.81E-03	4.36E-03	1.17E-02	5.79E-03	1.48E-02	6.72E-03	1.28E-02	9.40E-03	1.32E-02
Automap 398*6)	3.41E-03	5.18E-03	4.99E-03	7.08E-03	4.07E-03	5.34E-03	7.38E-03	9.31E-03	7.37E-03	9.39E-03
Cpu (209*7)	1.32E-05	6.61E-04	1.94E-03	7.22E-03	9.07E-06	1.72E-04	1.64E-03	6.85E-03	1.64E-03	6.92E-03
EchoMonths (130*9)	3.96E-02	4.27E-02	4.04E-02	4.55E-02	3.35E-02	5.25E-02	4.00E-02	4.62E-02	4.25E-02	4.74E-02
lungcancer_shedden 442*23)	1.82E-02	2.16E-02	1.85E-02	2.11E-02	2.22E-02	2.16E-02	2.22E-02	2.14E-02	2.19E-02	2.13E-02
Lowbwt (189*9)	9.52E-03	9.17E-03	2.62E-02	2.04E-02	2.55E-02	2.14E-02	2.87E-02	2.43E-02	2.87E-02	2.43E-02
machine_cpu(209*6)	1.13E-03	7.15E-04	1.02E-02	9.52E-03	9.54E-03	8.84E-03	1.01E-02	9.00E-03	1.01E-02	9.00E-03
Pharynx (195*10)	2.27E-02	2.79E-02	2.34E-02	2.80E-02	2.34E-02	3.24E-02	2.38E-02	2.72E-02	2.38E-02	2.72E-02
Places (329*8)	1.76E-02	2.11E-02	1.76E-02	2.15E-02	1.85E-02	1.90E-02	2.05E-02	2.07E-02	2.02E-02	2.01E-02
plasma_retinol (315*13)	1.50E-02	2.70E-02	1.39E-02	2.01E-02	1.57E-02	2.43E-02	1.60E-02	2.57E-02	1.60E-02	2.62E-02
rabe_266 (120*2)	5.77E-06	4.34E-05	5.91E-03	1.87E-02	1.28E-02	1.72E-02	3.42E-02	4.76E-02	3.42E-02	4.69E-02
rmftsa_ctoarrivals (264*2)	7.19E-04	1.06E-02	3.63E-03	5.06E-03	3.36E-03	5.32E-03	4.59E-03	6.55E-03	4.58E-03	6.41E-03
rmftsa_ladata(508*10)	2.90E-03	4.51E-03	3.63E-03	5.06E-03	3.36E-03	5.32E-03	4.59E-03	6.55E-03	4.58E-03	6.41E-03
visualizing_environmental (11*3)	1.89E-02	4.03E-02	1.99E-02	4.33E-02	1.98E-02	3.73E-02	2.24E-02	3.24E-02	2.24E-02	3.19E-02
visualizing_galaxy (323*4)	1.19E-03	1.06E-03	2.74E-03	2.09E-03	1.22E-03	1.67E-03	6.91E-03	5.82E-03	6.91E-03	5.82E-03
Wisconsin (194*32)	5.92E-02	6.97E-02	6.08E-02	7.46E-02	5.77E-02	7.43E-02	5.52E-02	7.21E-02	5.74E-02	7.31E-02

On benchmark datasets, TSVR also demonstrates distinct advantages over other models, manifested in the following aspects:

1. High-dimensional datasets (e.g., Stoke (950×9), Cpu (209×7), machine_cpu (209×6)): TSVR shows significant superiority, with train and test improvement percentages generally exceeding 90% when compared to SVR, Ridge, and Elastic Net. This benefit arises from HOA optimizing TSVR's regularization parameters, which effectively captures feature interactions.

2. Low-dimensional datasets (e.g., rabe_266 (120×2)): TSVR achieves a test improvement percentage of 99.75% against ELM and nearly 100% against Ridge and Elastic Net, demonstrating its high-precision approximation capability in low-dimensional datasets.

However, TSVR performs poorly on certain datasets. For instance, on the high-dimensional feature dataset lungcancer_shedden (442×23), the test improvement percentage against ELM is -0.1%. This is primarily due to the dataset's sample-to-feature ratio approaching 19:1, where redundant features interfere with the model's optimization direction. Feature selection techniques (e.g., Least Absolute Shrinkage and Selection Operator (LASSO)) may be necessary to reduce dimensionality and mitigate this issue.

The performance improvements of TSVR compared with other models on test functions and benchmark datasets are shown in Figure 2.

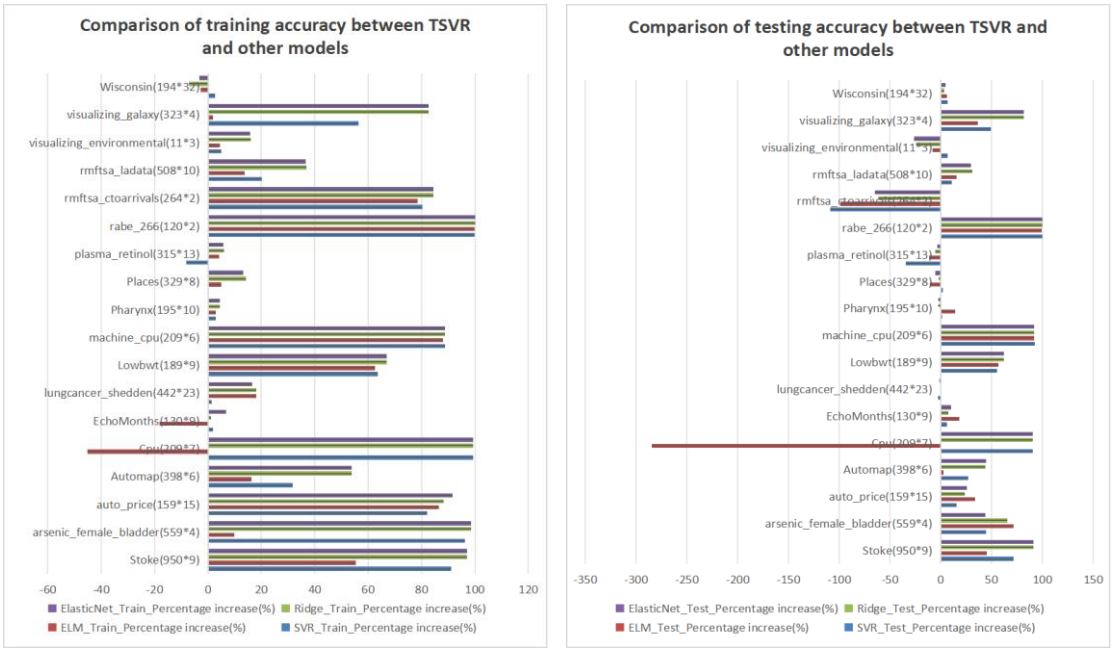

Figure 2: Comparison of MSE values of TSVR and other models on benchmark datasets

5. Conclusion

This paper provides an improved TSVR method combining the HOA and the SOR iteration. Through the efficient optimization of the regularization parameters and insensitive boundaries of TSVR by HOA, and combining with SOR to accelerate the solution process of the quadratic programming problem, the regression accuracy and computational efficiency of the model are significantly improved.

The experimental results show that on artificial datasets and 18 benchmark datasets, compared with traditional SVR, ELM, Ridge, and Elastic Net models, HOA-TSVR shows significant advantages in terms of the MSE index on both the training and test sets, which verifies the effectiveness of the algorithm in avoiding overfitting and enhancing generalization ability.

However, in high-dimensional feature datasets (such as lungcancer_shedden the sample feature ratio is approximately 19:1), TSVR's performance fluctuates significantly, highlighting its sensitivity to uneven data distribution and its vulnerability to redundant feature space. The model is easily

affected by noisy features during training, causing the optimization direction to deviate from the true data distribution, which in turn leads to overfitting or degradation of generalization ability. To systematically address the aforementioned issues, future research can explore innovative approaches from the following dimensions:

(1). Data preprocessing and feature space optimization. By introducing an adaptive feature selection mechanism, redundant features can be automatically identified and removed, or dimensionality reduction techniques can be used to map high-dimensional features to a low-dimensional manifold space, thereby preserving the intrinsic geometric structure of the data while reducing computational complexity.

(2). Intelligent and adaptive hyperparameter optimization. Design adaptive penalty factor update rules to enhance the efficiency of hyperparameter optimization while avoiding getting stuck in suboptimal solutions.

(3). Improvement of algorithm structure and design of hybrid framework. In the TSVR objective function, introduce a sparse regularization term and low-rank matrix decomposition to force the model to learn sparse sample representations and low-rank feature subspaces. Construct a hybrid model combining TSVR with attention mechanisms to dynamically weight the importance of different features.

Based on the above research, it is expected to develop a comprehensive solution chain of “data preprocessing-feature optimization-algorithm improvement”, significantly enhancing TSVR's modeling capabilities in complex data environments.

References

[1] Peng Xinjun. An Efficient Twin Support Vector Machine for Regression [J]. Neural Networks, 2010, 23: 365-372.

[2] Lourenço H R, Martin O C, Stützle T. Iterated Local Search: Framework and Applications (A). Gendreau M, Potvin J Y. Handbook of Metaheuristics [M]. New York: Springer, 2019: 129-168.

[3] Chen, W, & Lu, Z. Extremal Optimization for Multi-Objective Problems: A Case Study on Power Systems [J]. Applied Soft Computing, 2022, 128: 109432.

[4] Fu S, Li K, Huang H, et al. Red - billed blue magpie optimizer: a novel metaheuristic algorithm for 2D/3D UAV path planning and engineering design problems [J]. Artificial Intelligence Review, 2024, 57(6): 134.

[5] Zhang H, San H, Sun H, et al. A novel optimization method: wave search algorithm [J]. The Journal of Supercomputing, 2024: 1-36.

[6] Yuan C, Zhao D. Heidari A A, et al. Artemisinin optimization based on malaria therapy: Algorithm and application to medical image segmentation [J]. Displays, 2024, 102740.

[7] Hamad R K Rashid T A. GOOSE algorithm: a powerful optimizaton tool for real-world engineering challenges and beyond [J]. Evolving Systems, 2024, 1-26.

[8] Abdollahzadeh B, Khodadadi N, Barshandeh S, et al. Puma optimizer (PO): a novel metaheuristic optimization algorithm and its application in machine learning [J]. Cluster Computing, 2024, 27 (4): 1-49.

[9] Sunday O O, Stephen O E, Seyedali M. The Hiking Optimization Algorithm: A novel human - based metaheuristic approach [J]. Knowledge - Based Systems, 2024, 296: 111880.

[10] Goodchild M F. Beyond Tobler's hiking function [J]. Geographical Analysis, 2020, 52(4): 558-569.