Neural Computing and Applications (2019) 31:4849–4864
https://doi.org/10.1007/s00521-018-03981-1

ORIGINAL ARTICLE

A novel support vector regression algorithm incorporated with prior knowledge and error compensation for small datasets

Zhenyu Liu[1] · Yunkun Xu[1] · Chan Qiu[1] · Jianrong Tan[1]

Received: 1 August 2018 / Accepted: 26 December 2018 / Published online: 3 January 2019

Abstract
To solve the modeling problem under the conditions where the measured data are insufficient but biased prior knowledge from a simulator is available, we propose a novel multi-scale ν-linear programming support vector regression (ν-LPSVR) called ν-MPESVR. The proposed algorithm constructs a nested support vector regression model, which incorporates prior knowledge into ν-LPSVR, compensates for the errors between prior knowledge data and the measured data, and simultaneously achieves small training error for both the prediction model and the error compensation model. Considering that measured data may exist in multiple feature spaces, we extend the algorithm to multi-scale ν-LPSVR to achieve accurate modeling for complex problems. In addition, a strategy for parameter selection for ν-MPESVR is presented in this paper. The performance of the proposed algorithm is estimated in a synthetic example and a practical application. The performances of all models are evaluated with the root mean square error (RMSE), mean absolute error (MAE), and coefficient of determination (R2). Taking the three groups of experiments in the synthetic example as an instance, we find that the ν-MPESVR performs better, and it can still maintain high accuracy when the biases of prior knowledge data change (RMSE values of 0.1962, 0.1904, and 0.2261, MAE values of 0.1396, 0.1375, and 0.1623, and R2 values of 0.9919, 0.9923, and 0.9892 for the three groups of experiments, respectively). The experimental results indicate that the proposed algorithm can obtain a satisfactory model with a finite amount of measured data, and the performance is better than that of existing algorithms.

Keywords Support vector regression · Prior knowledge · Error compensation · Multi-scale · Small dataset

1 Introduction

Support vector regression (SVR) is based on the theory of structure risk minimization in statistical learning, which has been proven to provide excellent performance in various applications [25, 26]. In general, the sequential minimal optimization method [21] is used to solve quadratic programming problems in support vector regression (QPSVR) [25]. In order to minimize the tube parameter ε automatically, Smola [23] incorporated parameter ν into the optimization objective (ν-SVR) and proposed ν-linear programming support vector regression (ν-LPSVR) to solve the optimization problem fast and easily, and to enforce sparseness in the solution [24]. All of the above algorithms compute a nonlinear estimate in terms of kernel functions. In addition, ν-LPSVR is robust to local perturbations of the training set's target values [24], can use more general kernel functions flexibly, and offers higher calculation efficiency [14, 15].

In nonlinear mapping function estimation, ν-LPSVR has proven to be able to adapt accuracy ε to the noise level in the data automatically, and to provide excellent performance in various applications based on sufficient training data [24]. However, in some practical applications, such as performance prediction of small-batch aerospace components, the production of these parts is small, and the measuring methods are extremely limited and expensive [27, 30]. This leads to scarcity in measured data samples. For this reason, the accuracy and generalization performance of the model obtained with these measured data cannot meet the requirements. Moreover, a certain type and

✉ Chan Qiu
qc@zju.edu.cn

1 State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou 310027, Zhejiang, People's Republic of China

amount of prior knowledge, as with mathematical model or simulation model, is generally available. Although these models may not fully consider all the actual conditions and input characteristics, and the output results may be biased, prior models can still approximately build the main part of the real model. Consequently, obtaining an accurate model with only a few training samples and prior knowledge is the key issue in applying artificial intelligence technology to the industrial field.

In related works, a literature review of incorporating prior knowledge into SVC was presented in [10]. Prior knowledge was defined as knowledge about class-invariance and knowledge about data. Methods for incorporating prior knowledge into SVC were classified into three categories: sample methods, kernel methods, and optimization methods. In [9], the author introduced equality and inequality constraints to the SVR optimization problem by utilizing prior knowledge of particular points, derivatives, prior models, and the correlation between outputs. Considering that prior knowledge provided by simulators or prior models may be biased, Bloch [3] incorporated a vector v of positive slack variables bounding the error on this knowledge, and the author also showed the importance of adding potential support vectors of knowledge samples to the model. In [5], Chen built a new feature according to prior distribution knowledge of data to improve the accuracy of prediction. Tahir [7] proposed a kernel function construction method based on prior knowledge and Green's kernel. Because many complex functions comprise both steep and smooth variations [31], Zhou [32, 33] proposed a LPSVR incorporated with prior knowledge, and extended it to multiple kernel LPSVR. In addition, Zhang [28] incorporated prior knowledge into SVM with sample confidence and used weighted features to express prior knowledge. However, all the aforementioned methods are based on unbiased prior knowledge or only contain small noise error, which is inaccurate in the actual applications.

In reality, biases between prior knowledge data and measured data not only contain noise error but also the deviation caused by coupling features, other unknown properties, and the correlation between some factors is not considered in the simulator. An accurate data-driven model cannot be easily obtained by just incorporating positive slack variables to bound errors without the correction of the biased prior knowledge. In addition, the mapping function in practical applications may contain uneven distributions in high-dimensional feature spaces, and the express ability of the decision function could be improved by using the multi-scale kernel method. Compared with other multiple kernel methods, the multi-scale kernel method is more flexible and can provide a more complete scale selection.

To solve the above problem, we propose a multi-scale ν-LPSVR algorithm incorporated with prior knowledge and error compensation (ν-MPESVR). We first incorporate the prior knowledge data, which may be biased from the measured data into the ν-LPSVR by modifying and adding the inequality constraints (ν-PSVR). By setting the appropriate punishment coefficients of the slack variables in the objective function, we can adjust the tolerance of the decision function on the error in both the measured data and the prior data. Subsequently, we construct the error compensation model based on measured data and corresponding prior data. The two sets of data have the same input, but the former is the measured output and the latter is the output of the simulator. In order to improve the generalization ability of the obtained model, we incorporate the error compensation model into the ν-PSVR to make it part of the ν-PSVR optimization problem. In consequence, the goal is to achieve a small training error in both the error compensation model and the ν-PSVR simultaneously; thus, we call it ν-PESVR. Finally, multi-scale feature spaces have been utilized by incorporating multi-scale kernel functions into the ν-PESVR to adapt to the multi-scale characteristic of the data. In addition, to find the global optimum or a good approximation with high probability [13], we use the chaotic particle swarm algorithm to find the optimal parameters for the model.

The rest of the paper is organized as follows. Section 2 introduces ν-LPSVR briefly. The proposed ν-MPESVR algorithm incorporated with compensated prior knowledge data and multi-scale kernel functions is described in Sect. 3. In Sect. 4, numerical experiments are performed on a synthetic example and a practical application. Finally, we conclude our work in Sect. 5.

In this study, all the vectors are assumed to be column vectors. Lowercase symbols like x_{ij}, y refer to scalars, lowercase bold symbols like $\boldsymbol{x}, \boldsymbol{\alpha}$ refer to vectors, and uppercase bold symbols like $\boldsymbol{K}, \boldsymbol{G}$ refer to matrices. For any two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, the scalar or matrix $\boldsymbol{A}^{\mathrm{T}} \cdot \boldsymbol{B}$ is the inner product of the matrices, where $\boldsymbol{A}^{\mathrm{T}}$ denotes the transpose of $\boldsymbol{A}$. For $x, y \in R^d$, $\boldsymbol{X} \in R^{d\times m}$, and $\boldsymbol{Y} \in R^{d\times n}$, the kernel function $k(\boldsymbol{x}, \boldsymbol{y})$ is a scalar, and the kernel matrix $\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{Y})$ is an $R^{m\times n}$ matrix that maps $R^{d\times m} \times R^{d\times n}$ into $R^{m\times n}$. The identity matrix is denoted by $\boldsymbol{E}$, and $\boldsymbol{0}$ is a matrix of appropriate dimensions with all components equal to 0.

2 Review of ν-LPSVR

Given a dataset $\{(\boldsymbol{x_i}, y_i), i = 1, 2, 3, \ldots, N\}$, where $\boldsymbol{x_i} \in R^d$ denotes the d-dimension input vector, $y_i \in R$ denotes the real-valued output, i is the i-th training sample, and N is the number of training samples. The regression task amounts to finding a linear function in the feature space by using the kernel trick. The estimate equation can be described as follows:

$$y_i = f(\boldsymbol{x_i}) = \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\boldsymbol{x_i}) + b \tag{1}$$

where $\boldsymbol{\omega}$ is the normal vector, b is a bias term, and $\boldsymbol{\phi}(\boldsymbol{x})$ is a nonlinear mapping function [18–20].

Because the normal vector $\boldsymbol{\omega}$ can be considered a linear combination of the training patterns by using the dual representation, i.e., $\boldsymbol{\omega} = \sum_{j=1}^{N}(\alpha_j - \alpha_j^*) \cdot \boldsymbol{\phi}(\boldsymbol{x_j})$, one can obtain the following kernel expansion of the regression function as

$$y_i = f(\boldsymbol{x_i}) = \sum_{j=1}^{N}(\alpha_j - \alpha_j^*)k(\boldsymbol{x_j}, \boldsymbol{x_i}) + b \tag{2}$$

where $k(\boldsymbol{x_j}, \boldsymbol{x_i}) = \boldsymbol{\phi}(\boldsymbol{x_j}) \cdot \boldsymbol{\phi}(\boldsymbol{x_i})$ is the kernel function; Gaussian kernel and wavelet kernel are the two most widely used kernel functions in practical engineering [17]. Moreover, as for LPSVR, the non-Mercer kernel is available [14].

Unlike the standard SVR, which uses $\frac{1}{2}\|\boldsymbol{\omega}\|^2$ to make the function as flat as possible, LPSVR seeks a $\boldsymbol{\omega}$ that can be represented as the smallest combination of training patterns by using the coefficient parameter $\boldsymbol{\alpha}^{(*)}$, so it minimizes

$$\min \sum_{i=1}^{N}(\alpha_i + \alpha_i^*) + C\sum_{i=1}^{N} L(y_i - f(\boldsymbol{x_i})) \tag{3}$$

where the first term determines the complexity of the model, and the penalty parameter $C > 0$ is introduced to tune the trade-off between the error minimization and the maximization of the function flatness. $L(y_i - f(\boldsymbol{x_i}))$ refers to the ε-insensitive loss function as follows, and it penalizes any deviations larger than the precision ε for all the training data.

$$L(y_i - f(\boldsymbol{x_i})) = \begin{cases} 0 & |y_i - f(\boldsymbol{x_i})| \le \varepsilon \\ |y_i - f(\boldsymbol{x_i})| - \varepsilon & \text{otherwise} \end{cases} \tag{4}$$

By introducing slack variables $\boldsymbol{\xi_i}, \boldsymbol{\xi_i}^* \ge 0$ for each sample point $(\boldsymbol{x_i}, y_i)$, and using an ε-insensitive loss function, the LPSVR can be formulated as

$$\begin{aligned} &\text{find} && \boldsymbol{\alpha}^{(*)}, \boldsymbol{\xi}^{(*)}, b \\ &\min && \sum_{i=1}^{N}(\alpha_i + \alpha_i^*) + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ &\text{s.t.} && \alpha_i^{(*)}, \xi_i^{(*)} \ge 0 \\ &&& \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k(\boldsymbol{x_i}, \boldsymbol{x_j}) + b - y_i \le \varepsilon + \xi_i \\ &&& y_i - \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k(\boldsymbol{x_i}, \boldsymbol{x_j}) - b \le \varepsilon + \xi_i^* \end{aligned} \tag{5}$$

Here (*) is shorthand referring to both the variables with and without asterisks. In order to overcome problems with $|\alpha_i|$ in the objective function, α_i is substituted by the two positive variables α_i and α_i^* [24]. By solving the coefficient parameter $\boldsymbol{\alpha}^{(*)}$ in Eq. (5), the decision function Eq. (2) can be expressed as

$$y_i = f(\boldsymbol{x_i}) = \sum_{j=1}^{N}(\alpha_j - \alpha_j^*)k(\boldsymbol{x_j}, \boldsymbol{x_i}) + b \tag{6}$$

The tube width parameter ε determines the number of support vectors and errors in the model, which means that we must choose the optimal ε to ensure the accuracy of the data-driven model. For this reason, Bernhard [23] achieved automatic accuracy control by making the parameter ε part of the optimization problem. As a consequence, we obtain a solution with small training error and small ε by rewriting Eq. (5) as

$$\begin{aligned} &\text{find} && \boldsymbol{\alpha}^{(*)}, \boldsymbol{\xi}^{(*)}, \varepsilon, b \\ &\min && \frac{1}{N}\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) + \frac{C}{N}\sum_{i=1}^{N}(\xi_i + \xi_i^*) + C\nu\varepsilon \\ &\text{s.t.} && \alpha_i^{(*)}, \xi_i^{(*)}, \varepsilon \ge 0 \\ &&& \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k(\boldsymbol{x_i}, \boldsymbol{x_j}) + b - y_j \le \varepsilon + \xi_j \\ &&& y_j - \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k(\boldsymbol{x_i}, \boldsymbol{x_j}) - b \le \varepsilon + \xi_j^* \end{aligned} \tag{7}$$

where $\nu \in (0, 1]$ is an upper bound on the fraction of training errors and a lower bound of the fraction of support vectors. In addition, with probability 1, ν asymptotically equals both the fraction of SVs and the fraction of margin errors [4], and the decision function of the ν-LPSVR is the same as Eq. (6).

3 Proposed algorithm

In this section, to improve the prediction performance of the regression model developed with insufficient measured data and biased prior knowledge data, a novel constrained optimization-based multi-scale ν-LPSVR algorithm implemented with prior knowledge and error compensation is proposed.

3.1 Incorporating prior knowledge into ν-LPSVR

In this work, prior knowledge is defined as the samples generated by the mathematical models or the simulation models, which are called the prior knowledge dataset.

Consider a prior knowledge dataset $S_p = \{(\boldsymbol{x}_k^p, y_k^p), k = 1, 2, 3, \ldots, N_p\}$ and a measured dataset $S_r = \{(\boldsymbol{x}_t^r, y_t^r), t = 1, 2, 3, \ldots, N_r\}$, where $\boldsymbol{x}_k^p, \boldsymbol{x}_t^r \in R^d$ are the d-dimension prior knowledge input vector and measured input vector, $y_k^p, y_t^r \in R$ are the corresponding target output, and N_p, N_r are the number of prior knowledge samples and measured samples, respectively. It is obvious that prior knowledge samples satisfy the mapping relationship constructed in the simulator

$$y_k^p = T\left(\boldsymbol{x}_k^p\right) \qquad (k = 1, 2, 3 \ldots, N_p) \tag{8}$$

When prior knowledge is absolutely accurate, we would like the model to yield an exact value for these prior points rather than an approximation. The solution to this problem is to impose hard constraints on these prior points, such as

$$y_k^p = f\left(\boldsymbol{x}_k^p\right) = \sum_{j=1}^{N} \left(\alpha_j - \alpha_j^*\right) k\left(\boldsymbol{x}_j, \boldsymbol{x}_k^p\right) + b \tag{9}$$

However, in practical engineering, prior knowledge is always biased, and the equality constraints will lead to an exact fit to the prior points, which may not be advised. All these constraints may result in unsolvable problems if they cannot be satisfied simultaneously [3]. To manage this situation, we change the hard constraints to soft constraints by introducing the positive slack variable $\boldsymbol{u}^{(*)} = [u_1^{(*)}, u_2^{(*)}, u_3^{(*)}, \ldots, u_{N_p}^{(*)}]$ in constraints to bound the upper and lower deviations between the prior data $(\boldsymbol{x}^p, y^p)$ and the regression function $f(\boldsymbol{x}^p)$. Moreover, in order to include almost exact or biased prior knowledge, we use the ε-insensitive loss function (4) on the prior knowledge errors $\boldsymbol{u}^{(*)}$ with threshold ε_p to authorize violations of the equality constraints (9). Therefore, by applying the ε-insensitive loss function to the positive slack variables $\boldsymbol{u}^{(*)}$, we can obtain the following inequality constraints

$$-\varepsilon_p - u_k^* \leq y_k^p - f\left(\boldsymbol{x}_k^p\right) \leq \varepsilon_p + u_k \quad (k = 1, 2, 3, \ldots, N_p) \tag{10}$$

The l_1-norm of the slack vectors $\boldsymbol{u}^{(*)}$ is incorporated into the objective function of Eq. (7) with a trade-off parameter C_p to minimize the error $\boldsymbol{u}^{(*)}$, and the trade-off parameter C_p allows for tuning the influence of the prior data on the regression function. In addition, like ν-LPSVR , we make ε_p part of the objective function with the parameter ν_p to tune ε_p automatically. By consequence, approximate prior knowledge is incorporated by modifying the optimization problem (7). The modified algorithm, called ν-PSVR, is expressed as

$$\begin{aligned}
\text{find} \quad & \boldsymbol{\alpha}^{(*)}, \boldsymbol{\xi}^{(*)}, \boldsymbol{u}^{(*)}, \varepsilon_r, \varepsilon_p, b \\
\min \quad & \frac{1}{N}\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) + \frac{C_r}{N_r}\sum_{t=1}^{N_r}(\xi_t + \xi_t^*) \\
& + \frac{C_p}{N_p}\sum_{k=1}^{N_p}(u_k + u_k^*) + C_r\nu_r\varepsilon_r + C_p\nu_p\varepsilon_p \\
\text{s.t.} \quad & \alpha_i^{(*)}, \xi_t^{(*)}, u_k^{(*)}, \varepsilon_r, \varepsilon_p \geq 0 \\
& \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k\left(\boldsymbol{x}_i, \boldsymbol{x}_t^r\right) + b - y_t^r \leq \varepsilon_r + \xi_t \\
& y_t^r - \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k\left(\boldsymbol{x}_i, \boldsymbol{x}_t^r\right) - b \leq \varepsilon_r + \xi_t^* \\
& \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k\left(\boldsymbol{x}_i, \boldsymbol{x}_k^p\right) + b - y_k^p \leq \varepsilon_p + u_k \\
& y_k^p - \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k\left(\boldsymbol{x}_i, \boldsymbol{x}_k^p\right) - b \leq \varepsilon_p + u_k^*
\end{aligned} \tag{11}$$

where the two last sets of inequality constraints stand for the incorporation of prior knowledge. In this equation, $N = N_r + N_p$ is the total number of input training samples, and $\boldsymbol{x} = \boldsymbol{x}^r \bigcup \boldsymbol{x}^p$. $C_r > 0$ and $C_p > 0$ are the trade-off parameters for the slack variables $\boldsymbol{\xi}^{(*)}$ and $\boldsymbol{u}^{(*)}$. C_r tunes the trade-off between the error minimization and the maximization of the function flatness, and C_p can tune the influence of the prior knowledge on the model. By using linear programming to solve the optimization above, we can obtain the coefficient parameter $\boldsymbol{\alpha}^{(*)}$ and the bias term b; thus, we obtain the decision function as Eq. (6).

3.2 Compensating errors for prior knowledge

As we mentioned in Sect. 1, the biases between prior knowledge and actual mapping function are caused by noise error and the simplification of the simulator. During the construction of the simulator, some hard-to-measure, difficult-to-compute features, and complex coupling relationships between different features could not fully be considered. Alternatively, the simulator can build an approximation of the main part of the real model, implying that biases tend to be a relatively low-order mapping relationship.

Considering that the biased prior knowledge cannot be incorporated directly, we build a model developed by the ν-LPSVR to compensate the biased prior knowledge data obtained from the simulator based on all measured data samples and corresponding prior data samples. As we have defined before, $S_r = \{(\boldsymbol{x}_t^r, y_t^r), t = 1, 2, 3, \ldots, N_r\}$ is the measured dataset, and $S_p = \{(\boldsymbol{x}_k^p, y_k^p), k = 1, 2, 3, \ldots, N_p\}$ is the prior dataset. Here, we add an error compensation dataset $S_e = \{(\boldsymbol{x}_t^r, z_t^p), t = 1, 2, 3, \ldots, N_r\}$ generated by the simulator, where the input of S_e is consistent with the measured data input, and $z_t^p = T(\boldsymbol{x}_t^r) \in R$ is the corresponding target output from the simulator. Thus, similar to ν-LPSVR described above, the optimization problem of error compensation can be formulated as

$$\begin{aligned}
\text{find} \quad & \boldsymbol{\beta}^{(*)}, \boldsymbol{\mu}^{(*)}, \varepsilon_e, b_e \\
\min \quad & \frac{1}{N_r}\sum_{t=1}^{N_r}(\beta_t + \beta_t^*) + \frac{C_e}{N_r}\sum_{t=1}^{N_r}(\mu_t + \mu_t^*) + C_e \nu_e \varepsilon_e \\
\text{s.t.} \quad & \beta_t^{(*)}, \mu_t^{(*)}, \varepsilon_e \geq 0 \\
& \sum_{j=1}^{N_r}(\beta_j - \beta_j^*)k_e(\boldsymbol{x}_j^r, \boldsymbol{x}_t^r) + b_e - y_t^r + z_t^p \leq \varepsilon_e + \mu_t \\
& y_t^r - z_t^p - \sum_{j=1}^{N_r}(\beta_j - \beta_j^*)k_e(\boldsymbol{x}_j^r, \boldsymbol{x}_t^r) - b_e \leq \varepsilon_e + \mu_t^*
\end{aligned} \tag{12}$$

where $\boldsymbol{\beta}^{(*)}$ are the coefficient parameters, $\boldsymbol{\mu}^{(*)}$ are the positive slack variables, ε_e is the tube width, $C_e > 0$ is the penalty parameter to penalize nonzero coefficients $\boldsymbol{\mu}^{(*)}$. Hence, the decision function of the error compensation model can be expressed as

$$\Delta = y_k^{pec} - y_k^p = f_e(\boldsymbol{x}_k^p) = \sum_{i=1}^{N_r}(\beta_i - \beta_i^*)k_e(\boldsymbol{x}_i^r, \boldsymbol{x}_k^p) + b_e \tag{13}$$

where y^{pec} is the output of the compensated prior knowledge data samples. Thus, the compensated sample points can be written as $(\boldsymbol{x}_k^p, y_k^{pec})$.

We use the sample set $(\boldsymbol{x}_k^p, y_k^{pec})$ instead of $(\boldsymbol{x}_k^p, y_k^p)$ in the optimization equation (11); then, we obtain the data-driven model based on the corrected prior knowledge. This method of sequential model construction we call ν-PESVR$_{\text{sq}}$. However, one disadvantage of this method is that the measured data will be used twice, which may result in overfitting of the measured data.

In order to avoid this problem and ensure the generalization performance of the data-driven model, we improve the above method by incorporating an error compensation model into the optimization problem (11) of the ν-PSVR. That is, the overall optimal solution is obtained by solving the optimization problem only once. According to the above analysis, the algorithm called ν-PESVR can be solved by the following optimization problem

$$\begin{aligned}
\text{find} \quad & \boldsymbol{\alpha}^{(*)}, \boldsymbol{\xi}^{(*)}, \boldsymbol{\beta}^{(*)}, \boldsymbol{\mu}^{(*)}, \boldsymbol{u}^{(*)}, \varepsilon_r, \varepsilon_e, \varepsilon_p, b, b_e \\
\min \quad & \frac{1}{N}\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) + \frac{C_r}{N_r}\sum_{t=1}^{N_r}(\xi_t + \xi_t^*) \\
& + \frac{1}{N_r}\sum_{t=1}^{N_r}(\beta_t + \beta_t^*) + \frac{C_e}{N_r}\sum_{t=1}^{N_r}(\mu_t + \mu_t^*) \\
& + \frac{C_p}{N_p}\sum_{k=1}^{N_p}(u_k + u_k^*) + C_r\nu_r\varepsilon_r + C_e\nu_e\varepsilon_e \\
& + C_p\nu_p\varepsilon_p \\
\text{s.t.} \quad & \alpha_i^{(*)}, \xi_t^{(*)}, \beta_t^{(*)}, \mu_t^{(*)}, u_k^{(*)}, \varepsilon_r, \varepsilon_e, \varepsilon_p \geq 0 \\
& \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k(\boldsymbol{x}_i, \boldsymbol{x}_t^r) + b - y_t^r \leq \varepsilon_r + \xi_t \\
& y_t^r - \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k(\boldsymbol{x}_i, \boldsymbol{x}_t^r) - b \leq \varepsilon_r + \xi_t^* \\
& \sum_{j=1}^{N_r}(\beta_j - \beta_j^*)k_e(\boldsymbol{x}_j^r, \boldsymbol{x}_t^r) + b_e - y_t^r + z_t^p \leq \varepsilon_e + \mu_t \\
& y_t^r - z_t^p - \sum_{j=1}^{N_r}(\beta_j - \beta_j^*)k_e(\boldsymbol{x}_j^r, \boldsymbol{x}_t^r) - b_e \leq \varepsilon_e + \mu_t^* \\
& \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k(\boldsymbol{x}_i, \boldsymbol{x}_k^p) + b - y_k^p \\
& - \sum_{t=1}^{N_r}(\beta_t - \beta_t^*)k_e(\boldsymbol{x}_t^r, \boldsymbol{x}_k^p) - b_e \leq \varepsilon_p + u_k \\
& y_k^p + \sum_{t=1}^{N_r}(\beta_t - \beta_t^*)k_e(\boldsymbol{x}_t^r, \boldsymbol{x}_k^p) + b_e \\
& - \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k(\boldsymbol{x}_i, \boldsymbol{x}_k^p) - b \leq \varepsilon_p + u_k^*
\end{aligned} \tag{14}$$

where we substitute the decision function of the error compensation model into the constraint formulas of prior knowledge in Eq. (11). The parameter C_r penalizes those measured samples with errors greater than ε_r, C_e controls the intensity of the error compensation, and C_p tunes the influence of prior knowledge. In order to improve the generalization performance, we can structure the 3rd and 4th sets of inequality constraints with only a part of the measured samples and the corresponding prior samples. Equation (14) finds the corresponding overall optimal solution by setting the value of penalty parameters C_r, C_e, and C_p. Compared with the sequential execution method, this method is more efficient, and it has higher accuracy and better generalization performance.

The above ν-PESVR algorithm can be solved easily with linear programming.

3.3 Extending ν-PESVR to multi-scale space

If the unknown mapping function we need to fit is non-flat and comprises both the steep variations and the smooth variations, it would not be appropriate to estimate this mapping function using the SVR algorithm of a single kernel [31]. The small-scale kernels may lead to overfitting of training samples, and the large-scale kernels may lead to underfitting of training samples. As for multi-scale kernel SVR, the kernels with small scale can deal with the steep variations and the kernels with large scale can deal with the smooth variations. Thus, to adapt to the input space for each local area, we extend the ν-PESVR algorithm to multi-scale space(ν-MPESVR).

The decision function of multi-scale LPSVR can be expressed as

$$f(x)=\sum_{m=1}^{L}\sum_{i=1}^{N}(\alpha_{mi}-\alpha_{mi}^{*})k_m(\boldsymbol{x_i},\boldsymbol{x})+b \tag{15}$$

where L is the number of the scale kernels used in the regression, the kernel function $k_m(\boldsymbol{x_i},\boldsymbol{x})$ denotes the m-th scale kernel. Taking the Gaussian kernel as an example, $k_m(\boldsymbol{x_i},\boldsymbol{x})=exp(-\frac{\|x_i-x\|^2}{2\sigma_m^2})$, and $\alpha_{mi}^{(*)}$ are the dual variables of the corresponding kernel function. Similar to the target function of LPSVR in (3), we can obtain the optimization objective of the multi-scale ν-LPSVR as

$$\min\sum_{m=1}^{L}C_m\sum_{i=1}^{N}(\alpha_{mi}+\alpha_{mi}^{*})+C\sum_{i=1}^{l}L(f(\boldsymbol{x_i}),y_i) \tag{16}$$

where the parameter C_m penalizes nonzero dual variables $\alpha_{mi}^{(*)}$ and C determines the trade-off between the model complexity and the training error. In order to avoid overfitting the smooth variations by the small-scale kernels, generally, a decreasing penalty sequence$(C_1>\cdots>C_m>\cdots>C_L)$ should be given for the kernel sequence with increasing scale $(\sigma_1<\cdots<\sigma_m<\cdots<\sigma_L)$.

Because the deviation mapping function between measured data and simulation data is relatively smooth, and to reduce the amount of calculation, we use a single kernel in the error compensation model. Based on the basic principles and solution of ν-PESVR, the optimization problem of ν-MPESVR can be formulated as

$$\begin{aligned}
\text{find}\quad & \boldsymbol{\alpha_m}^{(*)},\boldsymbol{\xi}^{(*)},\boldsymbol{\beta}^{(*)},\boldsymbol{\mu}^{(*)},\boldsymbol{u}^{(*)},\varepsilon_r,\varepsilon_e,\varepsilon_p,b,b_e\\
& (m=1,2,\ldots,L)\\
\min\quad & \frac{1}{N}\sum_{m=1}^{L}C_m\sum_{i=1}^{N}(\alpha_{mi}+\alpha_{mi}^{*})+\frac{C_r}{N_r}\sum_{t=1}^{N_r}(\xi_t+\xi_t^{*})\\
& +\frac{1}{N_r}\sum_{t=1}^{N_r}(\beta_t+\beta_t^{*})+\frac{C_e}{N_r}\sum_{t=1}^{N_r}(\mu_t+\mu_t^{*})\\
& +\frac{C_p}{N_p}\sum_{k=1}^{N_p}(u_k+u_k^{*})+C_r\nu_r\varepsilon_r+C_e\nu_e\varepsilon_e\\
& +C_p\nu_p\varepsilon_p\\
\text{s.t.}\quad & \alpha_{mi}^{(*)},\xi_t^{(*)},\beta_t^{(*)},\mu_t^{(*)},u_k^{(*)},\varepsilon_r,\varepsilon_e,\varepsilon_p\geq 0\\
& \sum_{m=1}^{L}\sum_{i=1}^{N}(\alpha_{mi}-\alpha_{mi}^{*})k_m\left(\boldsymbol{x_i},\boldsymbol{x_t^r}\right)+b-y_t^r\leq\varepsilon_r+\xi_t\\
& y_t^r-\sum_{m=1}^{L}\sum_{i=1}^{N}(\alpha_{mi}-\alpha_{mi}^{*})k_m\left(\boldsymbol{x_i},\boldsymbol{x_t^r}\right)-b\leq\varepsilon_r+\xi_t^{*}\\
& \sum_{j=1}^{N_r}(\beta_j-\beta_j^{*})k_e\left(\boldsymbol{x_j^r},\boldsymbol{x_t^r}\right)+b_e-y_t^r+z_t^p\leq\varepsilon_e+\mu_t\\
& y_t^r-z_t^p-\sum_{j=1}^{N_r}(\beta_j-\beta_j^{*})k_e\left(\boldsymbol{x_j^r},\boldsymbol{x_t^r}\right)-b_e\leq\varepsilon_e+\mu_t^{*}\\
& \sum_{m=1}^{L}\sum_{i=1}^{N}(\alpha_{mi}-\alpha_{mi}^{*})k_m\left(\boldsymbol{x_i},\boldsymbol{x_k^p}\right)+b-y_k^p\\
& -\sum_{t=1}^{N_r}(\beta_t-\beta_t^{*})k_e\left(\boldsymbol{x_t^r},\boldsymbol{x_k^p}\right)-b_e\leq\varepsilon_p+u_k\\
& y_k^p+\sum_{t=1}^{N_r}(\beta_t-\beta_t^{*})k_e\left(\boldsymbol{x_t^r},\boldsymbol{x_k^p}\right)+b_e\\
& -\sum_{m=1}^{L}\sum_{i=1}^{N}(\alpha_{mi}-\alpha_{mi}^{*})k_m\left(\boldsymbol{x_i},\boldsymbol{x_k^p}\right)-b\leq\varepsilon_p+u_k^{*}
\end{aligned} \tag{17}$$

In order to facilitate the optimization solution of the problem, we reformulate Eq. (17) in the following matrix form for standard linear programming

$$\begin{aligned}
\text{find}\quad & \boldsymbol{s}\\
\min\quad & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{s}\\
\text{s.t.}\quad & \boldsymbol{s}\geq\boldsymbol{l}\\
& \boldsymbol{Gs}\leq\boldsymbol{h}
\end{aligned} \tag{18}$$

where

$$\boldsymbol{s}=\left[\boldsymbol{\alpha_1}^{(*)},\ldots,\boldsymbol{\alpha_L}^{(*)},\boldsymbol{\xi}^{(*)},\boldsymbol{\beta}^{(*)},\boldsymbol{\mu}^{(*)},\boldsymbol{u}^{(*)},\varepsilon_r,\varepsilon_e,\varepsilon_p,b,b_e\right]^{\mathrm{T}}$$

$$\boldsymbol{c}=\left[\frac{C_1}{N},\ldots,\frac{C_L}{N},\frac{C_r}{N_r},\frac{1}{N_r},\frac{C_e}{N_r},\frac{C_p}{N_p},C_r\nu_r,C_e\nu_e,C_p\nu_p,0,0\right]^{\mathrm{T}}$$

$$\boldsymbol{l}=[\boldsymbol{0},\ldots,\boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0},\boldsymbol{0},0,0,0,-\infty,-\infty]^{\mathrm{T}}$$

$$\boldsymbol{G}=\begin{bmatrix} \pm\boldsymbol{K}_1^r & \cdots & \pm\boldsymbol{K}_L^r & [-\boldsymbol{E},\boldsymbol{0}] & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & -1 & 0 & 0 & 1 & 0 \\ \mp\boldsymbol{K}_1^r & \cdots & \mp\boldsymbol{K}_L^r & [\boldsymbol{0},-\boldsymbol{E}] & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & -1 & 0 & 0 & -1 & 0 \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \pm\boldsymbol{K}_r^e & [-\boldsymbol{E},\boldsymbol{0}] & \boldsymbol{0} & 0 & -1 & 0 & 0 & 1 \\ \boldsymbol{0} & \cdots & \boldsymbol{0} & \boldsymbol{0} & \mp\boldsymbol{K}_r^e & [\boldsymbol{0},-\boldsymbol{E}] & \boldsymbol{0} & 0 & -1 & 0 & 0 & -1 \\ \pm\boldsymbol{K}_1^p & \cdots & \pm\boldsymbol{K}_L^p & \boldsymbol{0} & \pm\boldsymbol{K}_p^e & \boldsymbol{0} & [-\boldsymbol{E},\boldsymbol{0}] & 0 & 0 & -1 & 1 & -1 \\ \mp\boldsymbol{K}_1^p & \cdots & \mp\boldsymbol{K}_L^p & \boldsymbol{0} & \mp\boldsymbol{K}_p^e & \boldsymbol{0} & [\boldsymbol{0},-\boldsymbol{E}] & 0 & 0 & -1 & -1 & 1 \end{bmatrix}^{\mathrm{T}}$$

$$\boldsymbol{h}=\left[y_t^r,-y_t^r,y_t^r-z_t^p,z_t^p-y_t^r,y_k^p,-y_k^p\right]^{\mathrm{T}}$$

In the optimization, the matrices $\boldsymbol{s},\boldsymbol{c},\boldsymbol{l}$ denote $(2L\cdot N+6N_r+2N_p+5)\times 1$ column vector, $\boldsymbol{G}$ represents the $(4N_r+2N_p)\times(2L\cdot N+6N_r+2N_p+5)$ matrix, and $\boldsymbol{h}$ is a $(4N_r+2N_p)\times 1$ column vector. The coefficient vector $\boldsymbol{\alpha_m}^{(*)}=[\alpha_{m1},\alpha_{m2},\ldots,\ \alpha_{mN},\alpha_{m1}^*,\alpha_{m2}^*,\ldots,\alpha_{mN}^*]^{\mathrm{T}}$ denotes a $2N\times 1$ column vector. Similar to $\boldsymbol{\alpha}^{(*)}$, $\boldsymbol{\xi}^{(*)},\boldsymbol{\beta}^{(*)},\boldsymbol{\mu}^{(*)}$ are $2N_r\times 1$ column vectors, and $\boldsymbol{u}^{(*)}$ represents a $2N_p\times 1$ column vector. $\pm\boldsymbol{K}_{\boldsymbol{m}}^{\boldsymbol{r}}=[\boldsymbol{K}_{\boldsymbol{m}}^{\boldsymbol{r}},-\boldsymbol{K}_{\boldsymbol{m}}^{\boldsymbol{r}}]$ and $\mp\boldsymbol{K}_{\boldsymbol{m}}^{\boldsymbol{r}}=[-\boldsymbol{K}_{\boldsymbol{m}}^{\boldsymbol{r}},\boldsymbol{K}_{\boldsymbol{m}}^{\boldsymbol{r}}]$ are $N_r\times 2N$ matrices. Similarly, $\pm\boldsymbol{K}_{\boldsymbol{r}}^{\boldsymbol{e}}$ and $\mp\boldsymbol{K}_{\boldsymbol{r}}^{\boldsymbol{e}}$ are $N_r\times 2N_r$ matrices, $\pm\boldsymbol{K}_{\boldsymbol{m}}^{\boldsymbol{p}}$ and $\mp\boldsymbol{K}_{\boldsymbol{m}}^{\boldsymbol{p}}$ are $N_p\times 2N$ matrices, and $\pm\boldsymbol{K}_{\boldsymbol{p}}^{\boldsymbol{e}}$ and $\mp\boldsymbol{K}_{\boldsymbol{p}}^{\boldsymbol{e}}$ are $N_p\times 2N_r$ matrices. In addition, the kernel matrix $\boldsymbol{K}_{\boldsymbol{m}}^{\boldsymbol{r}}=\boldsymbol{K}_{\boldsymbol{m}}(\boldsymbol{x},\boldsymbol{x}^{\boldsymbol{r}})$, $\boldsymbol{K}_{\boldsymbol{r}}^{\boldsymbol{e}}=\boldsymbol{K}_{\boldsymbol{e}}(\boldsymbol{x}^{\boldsymbol{r}},\boldsymbol{x}^{\boldsymbol{r}})$, $\boldsymbol{K}_{\boldsymbol{m}}^{\boldsymbol{p}}=\boldsymbol{K}_{\boldsymbol{m}}(\boldsymbol{x},\boldsymbol{x}^{\boldsymbol{p}})$, and $\boldsymbol{K}_{\boldsymbol{p}}^{\boldsymbol{e}}=\boldsymbol{K}_{\boldsymbol{e}}(\boldsymbol{x}^{\boldsymbol{r}},\boldsymbol{x}^{\boldsymbol{p}})$. Moreover, in $[-\boldsymbol{E},\boldsymbol{0}]$ and $[\boldsymbol{0},-\boldsymbol{E}]$, $\boldsymbol{E}$ is the identity matrix and $\boldsymbol{0}$ is the zero matrix.

Algorithm 1 ν-MPESVR

```
Input:
    • The measured dataset $S_r=\{(\boldsymbol{x}_t^r,y_t^r),t=1,\ldots,N_r\}$;
    • The prior dataset $S_p=\{(\boldsymbol{x}_k^p,y_k^p),k=1,\ldots,N_p\}$;
    • The error compensation dataset $S_e=\{(\boldsymbol{x}_t^r,z_t^p),t=1,\ldots,N_r\}$ corresponding to the measured dataset;
    • The parameters $\nu_r$, $\nu_e$, $\nu_p$;
    • Kernel functions $k_m$, $k_e$ and the number of $k_m$ $L$.
1: Determine the total number $N=N_r+N_p$ of the training dataset.
2: Define the kernel parameters of the used kernel functions.
3: Select the hyperparameters $C_m$, $C_r$, $C_e$, and $C_p$.
4: Calculate the $N_r\times N$ kernel matrix $\boldsymbol{K}_{\boldsymbol{m}}^{\boldsymbol{r}}(m=1,2,\ldots,L)$ from the measured data and the training data.
5: Calculate the $N_r\times N_r$ kernel matrix $\boldsymbol{K}_{\boldsymbol{r}}^{\boldsymbol{e}}$ from the measured data.
6: Calculate the $N_p\times N$ kernel matrix $\boldsymbol{K}_{\boldsymbol{m}}^{\boldsymbol{p}}(m=1,2,\ldots,L)$ from the prior data and the training data.
7: Calculate the $N_p\times N_r$ kernel matrix $\boldsymbol{K}_{\boldsymbol{p}}^{\boldsymbol{e}}$ from the prior data and the measured data.
8: Construct the matrix $\boldsymbol{G}$ and the vector $\boldsymbol{c}$, $\boldsymbol{l}$ and $\boldsymbol{h}$.
9: Solve the optimization problem in Eq.(18) by using linear programming.
10: Substitute the solved coefficient parameters $\boldsymbol{\alpha}_{\boldsymbol{m}}^{(*)}$ and parameter $b$ into the decision function Eq.(15).
Output:
    • The decision function Eq.(15).
```

Using linear programming to solve the above optimization formulation, we can obtain a decision function in Eq. (15). Overall, ν-MPESVR can be summarized as shown in Algorithm 1.

3.4 Parameter selection strategy for ν-MPESVR

The generalization performance and accuracy of the support vector machine (SVM) are greatly influenced by its model parameters, both the error penalty parameter C and the kernel parameter, such as σ. Recently, intelligent optimization methods like simulated annealing (SA) [22], genetic algorithm (GA) [1], and particle swarm optimization (PSO) [11, 12] have became very popular in solving optimization problems. In the practical engineering application, the PSO can be programmed easily and produces superior performance [11, 12, 16]. Reference [8] demonstrates that in the parameter selection of SVM, PSO converges faster and has higher accuracy than other algorithms. Additionally, in order to avoid the problem of premature convergence, Liu [13] improved PSO by combining it with an adaptive inertia weight factor and chaotic local search, which is called chaotic particle swarm optimization (CPSO). It has been proved in [13] that CPSO is much faster than other meta-heuristics and can enable the search to escape from the local optima trap. Thus, to find the global optimal parameters in ν-MPESVR efficiently, in this section, we present an intelligent parameter selection strategy that uses expert knowledge for initial value screening of the CPSO to reduce the search space.

3.4.1 Expert knowledge about SVR parameters

Generally, the number L of kernel functions set to 2 or 3 may be sufficient for dealing with most practical problems, and a large L ($L\geq 4$) may be necessary for some complex problems [31]. The kernel parameter of the m-th kernel function is defined as σ_m. To avoid overfitting, we choose $C_m=1/\sigma_m$. As for the error penalty parameter C (C_r, C_e, and C_p in ν-MPESVR), it is closely related to the statistical characteristics of the training data, and Cherkassky [6] proposed the prescription for penalty parameter C as

$$C = \max(|\bar{y} + 3s_y|, |\bar{y} - 3s_y|) \tag{19}$$

where $\bar{y}$ and s_y are the mean and standard deviation of the output values in the training dataset. Moreover, it is well accepted that the kernel parameter should be associated with the distribution characteristics of the training data. The kernel parameter σ is set to the following form in [29] as

$$\sigma \sim (0.1 - 0.5) \cdot range(x) \tag{20}$$

where $range(x) = |\max(x) - \min(x)|$; thus, we get the initial value $(C_r, C_p, C_e, \sigma_e, C_1, \ldots, C_L, \sigma_1, \ldots, \sigma_L)_{init}$.

3.4.2 Parameter selection with CPSO

PSO is a population-based optimization technique, where each solution is a "particle" and multiple candidate solutions coexist and collaborate simultaneously to find the optimal solution in the problem search space. The state of the particle in the search space is characterized by its position and velocity, which can be updated by following equations

$$\begin{aligned} \boldsymbol{v_i} = w\boldsymbol{v_i} + c_1 \cdot random(0,1) \cdot (\boldsymbol{p_i} - \boldsymbol{x_i}) \\ + c_2 \cdot random(0,1) \cdot (\boldsymbol{p_g} - \boldsymbol{x_i}) \end{aligned} \tag{21}$$

$$\boldsymbol{x_i} = \boldsymbol{x_i} + \boldsymbol{v_i} \tag{22}$$

where the column vectors $\boldsymbol{v_i}, \boldsymbol{x_i}, \boldsymbol{p_i}, \boldsymbol{p_g} \in R^{np}$, np is the number of particles. $\boldsymbol{v_i}$ is the velocity of particle i, $\boldsymbol{x_i}$ represents the position of particle i, $\boldsymbol{p_i}$ denotes the best previous position of particle i, $\boldsymbol{p_g}$ denotes the best position among all particles, w is the inertia factor that controls the impact of the velocity of the previous particle on its current particle. c_1 and c_2 are acceleration coefficients that control the maximum step size of the particle, and $random(0, 1)$ is a random number between 0 and 1.

The inertia factor w in (21) controls the momentum of the current particle. A large w may cause the particle to miss the optimal region and make the algorithm unable to converge, while a small w may cause particles to become trapped in a local optimum. Thus, Liu [13] proposed the adaptive inertia weight factor (AIWF) as follows

$$w = \begin{cases} w_{\min} + \dfrac{(w_{\max} - w_{\min})(f - f_{\min})}{f_{\text{avg}} - f_{\min}} & f \le f_{\text{avg}} \\ w_{\max} & f > f_{\text{avg}} \end{cases} \tag{23}$$

where $w_{\max}$ and $w_{\min}$ are the maximum and minimum of w respectively, f is the evaluation value of the current particle, and f_{avg} and f_{min} are the average and minimum evaluation values of all particles, respectively.

In order to enable the particles to escape from the local optima trap, Liu [13] incorporated chaotic dynamics into the above PSO with AIWF. By using the logistic equation, the process of the chaotic local search could be defined as

$$c\boldsymbol{x_i}^{(k)} = \frac{\boldsymbol{x_i}^{(k)} - \boldsymbol{x}_{\min,i}}{\boldsymbol{x}_{\max,i} - \boldsymbol{x}_{\min,i}}, \quad (i = 1, 2, \ldots, np) \tag{24}$$

$$c\boldsymbol{x_i}^{(k+1)} = 4c\boldsymbol{x_i}^{(x)}(1 - c\boldsymbol{x_i}^{(k)}) \tag{25}$$

$$\boldsymbol{x_i}^{(k+1)} = \boldsymbol{x}_{\min,i} + c\boldsymbol{x_i}^{(k+1)}(\boldsymbol{x}_{\max,i} - \boldsymbol{x}_{\min,i}) \tag{26}$$

where $c\boldsymbol{x_i}$ is the i-th chaotic variable, and k is the current iteration number. The iteration stops if the new solution is better than $\boldsymbol{x}^{(0)}$ and the difference between two iterations is smaller than the set threshold, or if it reached the predefined maximum iteration.

By using fivefold cross-validation as the evaluation function in CPSO, the ν-MPESVR algorithm based on the above parameter selection strategy can be summarized as Fig. 1.

As Fig. 1 shows, we first obtain the training dataset S_r and the test dataset S_{test} from the real world, and build an error compensation dataset S_e and an additional prior knowledge dataset S_p through the simulation model. We input S_r, S_e and S_p into the ν-MPESVR model as the final training dataset. Before training the model, we need to specify the number of kernel functions k_m and the type of kernel functions k_m and k_e. Then, we initialize the model parameters C $(C_r, C_p, C_e, C_1, \ldots, C_L)$ and σ $(\sigma_1, \ldots, \sigma_L)$ according to the expert knowledge in Sect. 3.4.1, and specify the parameters of CPSO (inertia factor, acceleration coefficients, search scope, and maximum iteration). Because the supply of data for training and testing is limited, we use fivefold cross-validation to find the optimal parameters of the model [2]. In each iteration of CPSO, we divide the training dataset S_r into five groups (S_e is also divided into five groups corresponding to S_r). Four of the groups are used to train a set of models that are then evaluated on the remaining group. This procedure is then repeated for all five possible choices for the held-out group, and the performance scores from the five runs are then averaged to represent the estimation of the model performance (fitness evaluation of CPSO) under these model parameters. CPSO will update the parameters and proceed to the next iteration until the stopping condition is satisfied or the maximum number of iterations is reached. Finally, we use the final optimized model parameters and all of the S_r, S_e, and S_p to train the ν-MPESVR, and run the model on S_{test} to get the final model prediction.

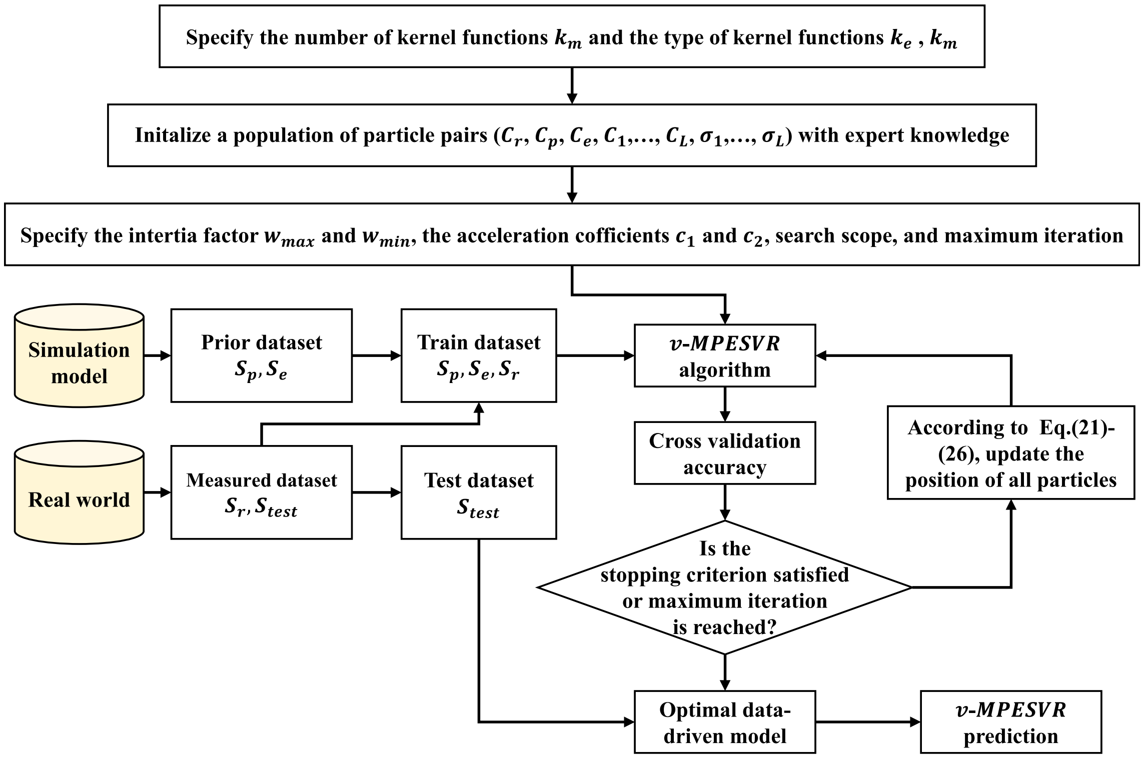
Specify the number of kernel functions k_m and the type of kernel functions k_e , k_m
Initalize a population of particle pairs (C_r, C_p, C_e, C_1,..., C_L, σ_1,..., σ_L) with expert knowledge
Specify the intertia factor w_{max} and w_{min}, the acceleration cofficients c_1 and c_2, search scope, and maximum iteration
Simulation model
Prior dataset S_p, S_e
Train dataset S_p, S_e, S_r
v-MPESVR algorithm
Real world
Measured dataset S_r, S_{test}
Test dataset S_{test}
Cross validation accuracy
According to Eq.(21)-(26), update the position of all particles
Is the stopping criterion satisfied or maximum iteration is reached?
Optimal data-driven model
v-MPESVR prediction

Fig. 1 Flowchart of ν-MPESVR parameters optimization and model constructing procedure

4 Experimental results

In this section, we validate the efficiency of our proposed algorithm by using an artificial example and a practical example. Root mean squared error (RMSE), mean absolute error (MAE), and coefficient of determinations (R(2)) are used as indicators to evaluate the accuracy and generalization performance of the obtained data-driven model. As known in statistical analysis, the best prediction performance for R2 is one and will be zero for other indices [18].

4.1 Complex function estimation

We consider the following complex piecewise function as the base function of the real model and the simulation model.

$$f(x) = \begin{cases} -4x - 8 & -3 \le x < -1 \\ -3x^3 - 5x^2 + 5x + 3 & -1 \le x < 1 \\ 2\sin(e^{1.2x}) + 0.3552 & 1 \le x < 3 \end{cases} \qquad (27)$$

We generated the measured dataset S_r with 15 samples, the error compensation dataset S_e with 15 samples, the prior knowledge dataset S_p with 35 samples, and the testing dataset S_{test} with 1000 samples from the above function randomly in the range of [− 3,3]. We should have applied the error term to S_r and S_{test} on the basis of base function Eq. (27). However, for the convenience of the comparison and display of the predicted results, we apply the error term on S_p and S_e instead.

In order to validate the performance of our proposed algorithm, we designed three groups of experiments. In the first group of experiments, we modified the mapping function of the simulation model as $f_{p1}(x) = f(x) + f_{\text{noise}}$, where f_{noise} is the Gaussian noise $N(0.1, 0.2^2)$. In the second group of experiments, we added the error term with a simple mapping relationship to the mapping function of the simulation model as $f_{p2}(x) = f(x) + f_{\text{noise}} + 0.7x$. In the third group of experiments, we added the error term with a high-order mapping relationship to the mapping function of the simulation model as $f_{p3}(x) = f(x) + f_{\text{noise}} + 0.7x + 0.1x^4$. Figure 2 shows the S_r, S_e, S_p, and S_{test} in the three groups of experiments.

For each group of experiments, we constructed five models using different algorithms, including MKPLPSVR, MSSVR, ν-PESVR, ν-MPESVR$_{\text{sq}}$, and ν-MPESVR. The MKPLPSVR was proposed in [33], MSSVR denotes standard multi-scale support vector regression [31], and ν-PESVR, and ν-MPESVR are proposed above. The ν-MPESVR$_{\text{sq}}$ algorithm is the multi-scale extension of the ν-PESVR$_{\text{sq}}$ algorithm proposed in Sect. 3.2 that first constructs the error compensation model to correct the prior samples and then builds the prediction model to output the final results. Using the training data samples and setting the

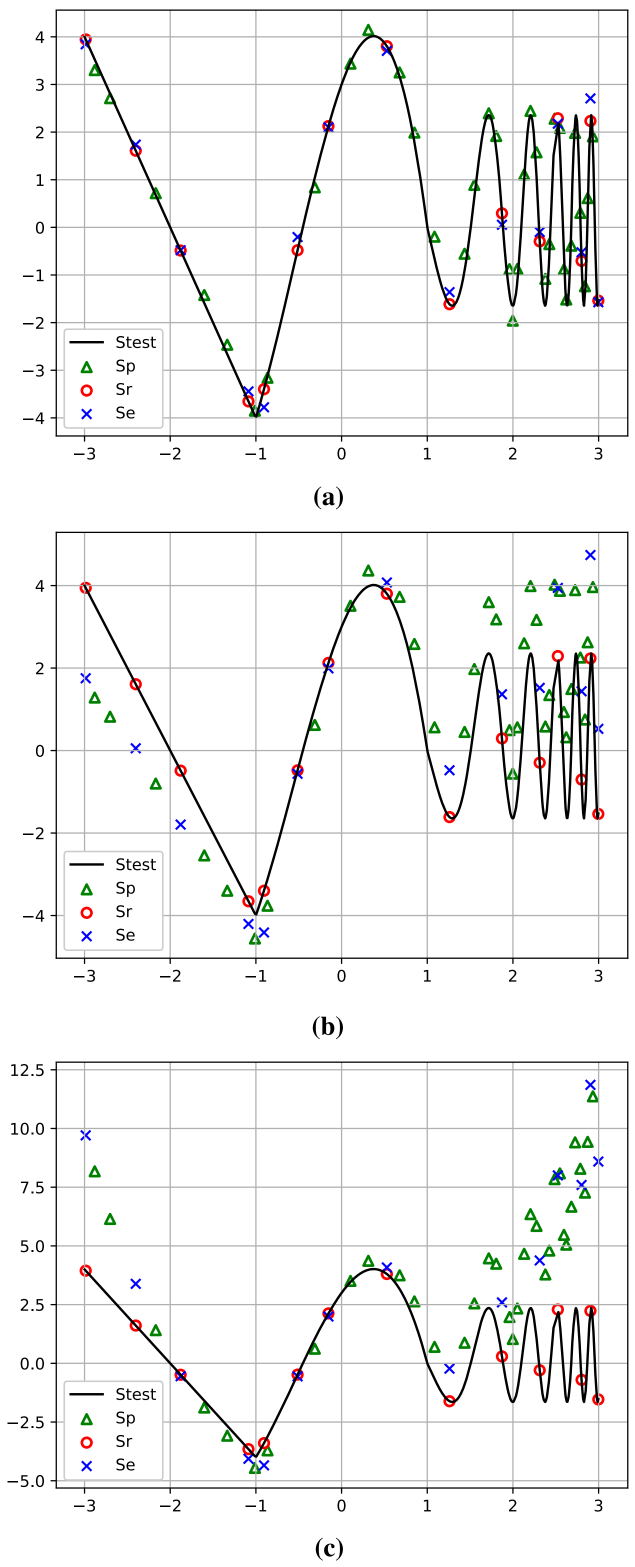

Fig. 2 Data samples S_r, S_e, S_p, and S_{test} in three groups of experiment

model parameters appropriately, we build the models separately using the above five algorithms.

For these three groups of experiments, the single-kernel algorithm exploited only a Gaussian kernel, and the multi-kernel algorithms employed three Gaussian kernels. In ν-PESVR, we chose $C_r = 100, C_p = 100, C_e = 100$, $\nu_r = \nu_p = \nu_e = 0.5$, and the kernel parameters $\sigma = 0.1118$, and $\sigma_e = 2.2361$. In ν-MPESVR$_{\text{sq}}$ and ν-MPESVR, we chose $C_1 = 0.6071, C_2 = 0.3141, C_3 = 2.0707, C_r = 150$, $C_p = 100, C_e = 100$, and $\nu_r = \nu_p = \nu_e = 0.5$, and the kernel parameters $\sigma_1 = 0.7071, \sigma_2 = 0.2041, \sigma_3 = 0.0707$, $\sigma_e = 2.2361$. Moreover, the parameters in MKPLPSVR and MSSVR are the same as those in ν-MPESVR$_{\text{sq}}$ and ν-MPESVR, except for the parameters related to error compensation.

Figure 3 shows the comparison between the prediction of each algorithm and the actual results in the first group of experiments. It shows that all of the algorithms fit the curve accurately. However, compared with Fig. 3a, b, we can find that the multi-kernel algorithms like ν-MPESVR, MKPLPSVR, and MSSVR have better performance for fitting functions with steep variations and smooth variations than the single-kernel algorithm ν-PESVR and ν-MPESVR$_{\text{sq}}$. One possible explanation is that the ν-PESVR with a small kernel parameter caused overfitting of the smooth variations, and that the ν-MPESVR$_{\text{sq}}$ utilizing the measured samples S_r twice to compensate for the random Gaussian noise may have led to the overfitting of the training data.

Figures 4 and 5 show the estimation results in the second and third group of experiments. We can see that the algorithms with error compensation, like ν-PESVR, ν-MPESVR$_{\text{sq}}$, and ν-MPESVR, fit the practical curve more accurately than other algorithms. When the error of the prior data is large, those algorithms without error compensation cannot estimate the mapping function very well by only incorporating slack variables, and with the growing number of prior samples, the prediction curve will increasingly approximate the simulation curve, which is not what we expected.

In order to show the performance of the proposed algorithm more intuitively, Table 1 lists the predicted RMSE, MAE, R(2), the number of SVs and iterations, and the training time of all the algorithms in each group of experiments. From Table 1, we can see that for those algorithms without error compensation, RMSE and MAE become larger, and R(2) becomes smaller with an increase in the value and complexity of the error term. However, those algorithms with error compensation still maintain the approximation accuracy. Owing to the increased complexity of the algorithms with error compensation, their training time is relatively long. In addition, we found that the model generated by ν-MPESVR had the best RMSE, MAE, and R(2) of all the models.

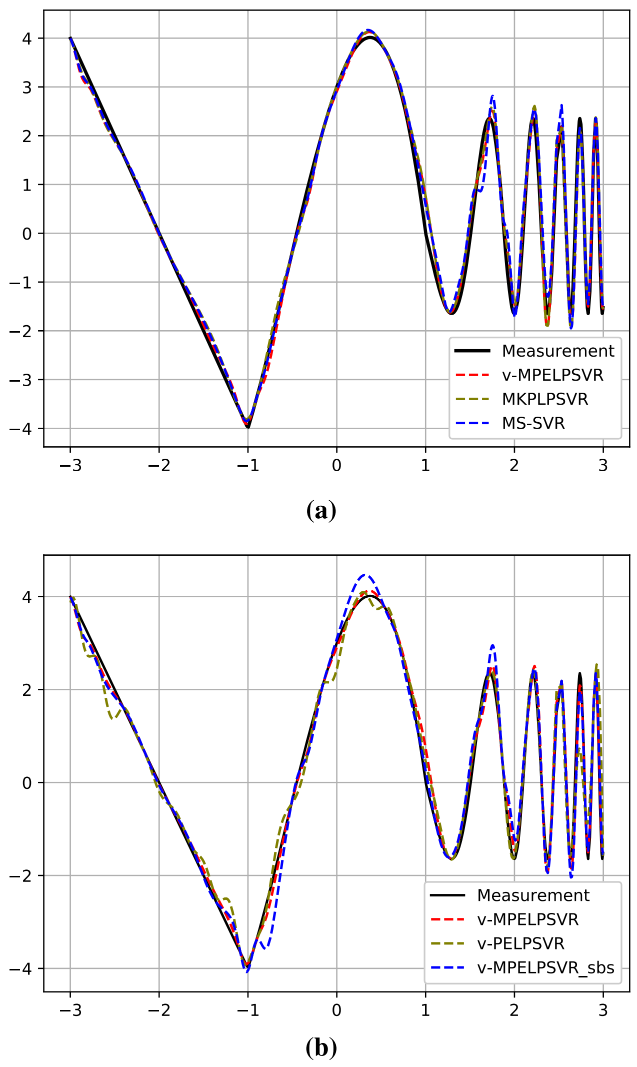

Fig. 3 Comparison of predicted results in the first group of experiment. **a** Results of measurement,ν-MPESVR, MKPLPSVR, and MSSVR. **b** Results of measurement,ν-MPESVR,ν-PESVR, and ν-MPESVR$_{sq}$

It can be seen from the results of the three groups of experiments that the data-driven model constructed by ν-MPESVR obtained the most accurate predictive performance when only a few pieces of measured data were available. The multi-kernel algorithm obtained better generalization performance and accuracy when estimating a complex function with both steep and smooth variations than did the single-kernel algorithm. Incorporating the error compensation optimization formula into the solution process of the prediction model prevented the need to use the dataset S_r twice, and thus improved the generalization performance and accuracy of the obtained model. As a consequence, the ν-MPESVR solved the problem of deviation in prior knowledge data and scarcity in the measured data.

4.2 Coordinator gyro rotor performance prediction

The coordinator gyro rotor is the core component of the infrared missile and small spacecraft. It can search, capture, and track the target in the field of view, and then adjust the flight direction and attitude of the missile. Therefore, the quality of the coordinator gyro rotor directly determines the tactical performance of the infrared missile. In this section, in order to improve productive efficiency, we use the above method to predict the gyro drift with the assembly process parameters and adjustment parameters to guide product assembly.

As shown in Fig. 6, a two-degree-of-freedom gyro is primarily composed of the rotor, the inner-ring and the outer-ring. We take the inertial coordinate system as $ox_iy_iz_i$, the outer-ring coordinate system as $ox_wy_wz_w$, the inner-ring coordinate system as $ox_ny_nz_n$, and the rotor coordinate system as $ox_hy_hz_h$. Then, the gyro dynamic equation considering the assembly error can be expressed as

$$\begin{cases} \frac{\mathrm{dH_h^{hz}}}{\mathrm{dt}} = 0 \\ \frac{\mathrm{d(H_n^{ny} + H_h^{ny} + H_{h\_e}^{ny})}}{\mathrm{dt}} + (H_n^{nz} + H_h^{nz} + H_{h\_ae}^{nz})(\omega_n^{nz} \\ \quad +\omega_w^{nz}) - (H_n^{nx} + H_h^{nx} + H_{h\_e}^{nx})(\omega_n^{nz} + \omega_w^{nz}) = M_y \\ \frac{\mathrm{d(H_w^{wx} + H_h^{wx} + H_{h\_e}^{wx} + H_n^{wx} + H_{n\_e}^{wx})}}{\mathrm{dt}} = M_x \end{cases} \tag{28}$$

where H_h^{hz} is the projection of the rotor angular momentum on the z-axis of the rotor coordinates. $H_n^{nx}, H_n^{ny}, H_n^{nz}$ denote the projection of the inner-ring angular momentum on the x-, y-, and z-axis of the inner-ring coordinates, respectively. H_w^{wx} is the projection of the outer-ring angular momentum on the x-axis of the outer-ring coordinates. $H_{h\_e}, H_{n\_e}, H_{w\_e}$ denote the angular momentums of the rotor, the inner-ring, and the outer-ring caused by assembly errors. $\omega_n^{nz}, \omega_w^{nz}$ represent the projection of the angular velocity of the inner and outer-rings on the z-axis of the inner-ring coordinates, respectively. Moreover, M_x, M_y are the torque of the outer-ring shaft and the inner-ring shaft, respectively.

In order to solve Eq. (28), we calculate the angular momentum of each component in the corresponding moving coordinate system. In the rotor coordinates $ox_hy_hz_h$:

$$H_h^{hz} = J_{hz} \cdot (\Omega - \dot{\theta_w} \cdot \sin\theta_n) \tag{29}$$

where J_h denotes the moment of inertia about the rotor. θ_n is the rotation angle of the inner-ring, and θ_w is the rotation angle of the outer-ring.

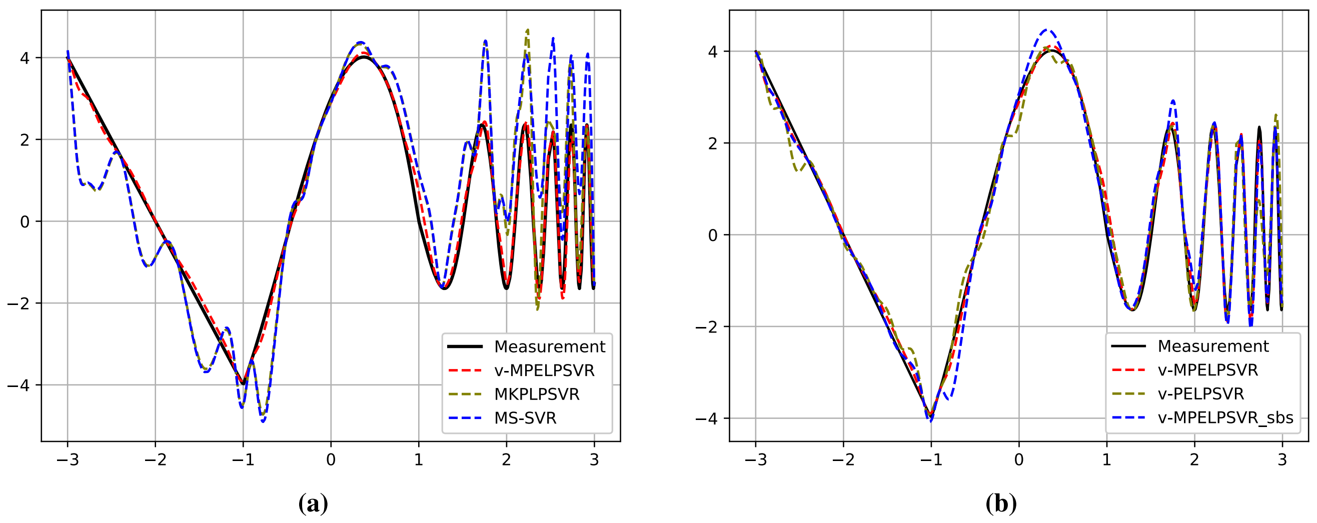

Fig. 4 Comparison of predicted results in the second group of experiment. **a** Results of measurement,ν-MPESVR, MKPLPSVR, and MSSVR. **b** Results of measurement,ν-MPESVR,ν-PESVR, and ν-MPESVR$_{\mathrm{sq}}$

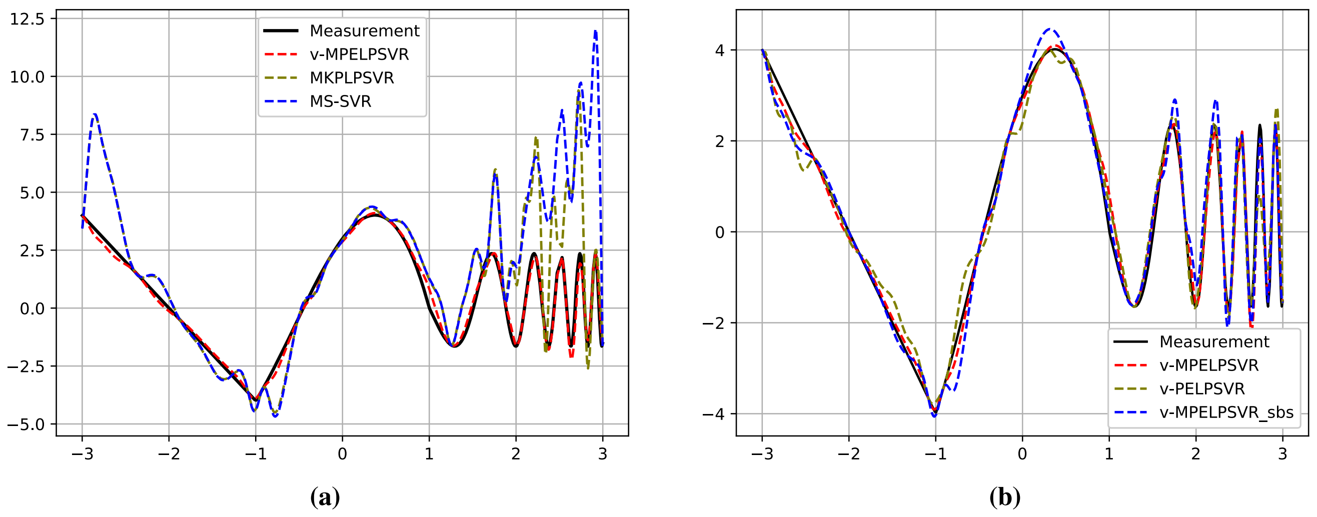

Fig. 5 Comparison of predicted results in the third group of experiment. **a** Results of measurement,ν-MPESVR, MKPLPSVR, and MSSVR. **b** Results of measurement,ν-MPESVR,ν-PESVR, and ν-MPESVR$_{\mathrm{sq}}$

In the inner-ring coordinates $ox_ny_nz_n$:

$$\begin{cases} H_n^{nx} = J_{nx} \cdot \dot{\theta_w} \cdot \cos\theta_n \\ H_n^{ny} = J_{ny} \cdot \dot{\theta_n} \\ H_n^{nz} = -J_{nz} \cdot \dot{\theta_w} \cdot \sin\theta_n \end{cases} \tag{30}$$

$$\begin{cases} H_{hw\_e}^{nx} = m_h v_{cw}(\delta_{hy}\cos\alpha\cos\theta_n - \delta_{hx}\sin\alpha\sin\theta_n) \\ H_{hw\_e}^{ny} = -m_h v_{cw}\delta_{hx}\cos\alpha\cos\theta_n \\ H_{hw\_e}^{nz} = -m_h v_{cw}(\delta_{hy}\cos\alpha\sin\theta_n - \delta_{hx}\sin\alpha\cos\theta_n) \end{cases} \tag{31}$$

$$\begin{cases} H_{hn\_e}^{nx} = -m_h \cdot \delta_{hy} \cdot \delta_{hx} \cdot \dot{\theta_n} \\ H_{hn\_e}^{ny} = m_h \cdot \delta_{hx}^2 \cdot \dot{\theta_n} \\ H_{hn\_e}^{nz} = m_h \cdot \delta_{hy} \cdot \delta_{hz} \cdot \dot{\theta_n} \end{cases} \tag{32}$$

$$H_h^n = H_h^h \tag{33}$$

$$H_{h\_e}^n = H_{hw\_e}^n + H_{hn\_e}^n \tag{34}$$

where

$$v_{cw} = \dot{\theta_w}\sqrt{(\delta_{hx}\sin\theta_n - \delta_{hz}\cos\theta_n)^2 + (\delta_n + \delta_{hy})^2}$$

$$\cos\alpha = \frac{\delta_n + \delta_{hy}}{\sqrt{(\delta_n + \delta_{hy})^2 + (\delta_{hx}^2 + \delta_{hz}^2)\sin\gamma^2}}$$

$$\gamma = \arctan(\frac{\delta_{hz}}{\delta_{hx}}) - \theta_n$$

In the above equations, m_h is the mass of the rotor, J is the moment of inertia, Ω is the rotor speed, and $\delta_{hx}, \delta_{hy}, \delta_{hz}$ are the offsets of the rotor barycenter position on the x-, y-, and

Table 1 Result comparisons of MKPLPSVR, MSSVR, ν-PESVR, ν-MPESVR$_{sq}$, ν-MPESVR on the complex function estimation example

	Experimental group	MKPLPSVR	MSSVR	ν-PESVR	ν-MPESVR$_{sq}$	ν-MPESVR
Group 1	RMSE	0.2039	0.2457	0.3417	0.2938	**0.1962**
	MAE	0.1445	0.1646	0.2275	0.2087	**0.1396**
	R(2)	0.9912	0.9872	0.9737	0.9831	**0.9919**
	SVs	31	36	37	38	36
	Time(s)	0.7164	0.6690	0.4802	0.9704	0.9197
	Iter.	16	15	23	20	17
Group 2	RMSE	1.0320	1.1477	0.3408	0.2910	**0.1904**
	MAE	0.7853	0.8924	0.2273	0.2045	**0.1375**
	R(2)	0.8201	0.7843	0.9737	0.9835	**0.9923**
	SVs	39	39	37	38	35
	Time(s)	0.4673	0.5793	0.4761	0.5980	0.8060
	Iter.	12	16	20	18	17
Group 3	RMSE	1.9805	2.6567	0.3844	0.3194	**0.2261**
	MAE	1.2126	1.5804	0.2729	0.2346	**0.1623**
	R(2)	0.5826	0.3811	0.9655	0.9799	**0.9892**
	SVs	37	36	36	37	34
	Time(s)	0.7902	0.6047	0.2805	0.6803	0.8238
	Iter.	14	16	20	18	17

Bold values indicate the best ones

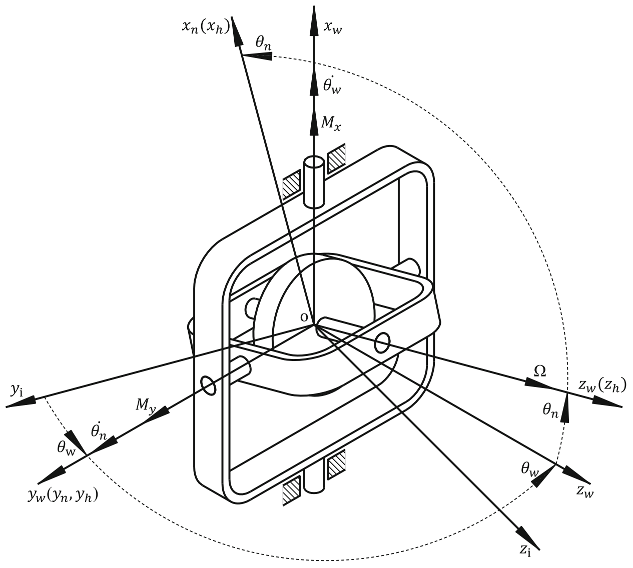

Fig. 6 Schematic diagram of two-degree-of-freedom gyro

z-axis, respectively. $H^n_{hw\_e}, H^n_{hn\_e}$ are the projection of the angular momentum of the rotor in the inner-ring coordinates caused by assembly error under the effect of $\dot{\theta_w}$ and $\dot{\theta_n}$, respectively.

In the outer-ring coordinates $ox_wy_wz_w$:

$$H^{wx}_w = J_{wx} \cdot \dot{\theta_w} \tag{35}$$

$$H^{wx}_h = -J_{hz}(\Omega - \dot{\theta_w} \cdot \sin\theta_n)\sin\theta_n + J_{hx}\dot{\theta_w}\cos^2\theta_n \tag{36}$$

$$H^{wx}_{hw\_e} = m_h\dot{\theta_w}[(\delta_n + \delta_{hy})^2 - \delta_{hx}\sin\gamma\sin\theta_n\sqrt{\delta^2_{hx} + \delta^2_{hz}}] \tag{37}$$

$$H^{wx}_{hn\_e} = -[m_h\dot{\theta_n}(\delta_n + \delta_{hy})](\delta_{hz}\sin\theta_n + \delta_{hx}\cos\theta_n) \tag{38}$$

$$H^{wx}_{h\_e} = H^{wx}_{hw\_e} + H^{wx}_{hn\_e} \tag{39}$$

$$H^{wx}_n = \dot{\theta_w}(J_{nz}\sin^2\theta_n + J_{nx}\cos^2\theta_n) \tag{40}$$

$$H^{wx}_{n\_e} = m_n\delta^2_n\dot{\theta_w} + m_n\delta_n\delta_{nz}\dot{\theta_n}\sin\theta_n \tag{41}$$

where m_n is the mass of the inner-ring, δ_n, δ_{nz} are the offsets of the center of mass of the inner-ring on the y- and z-axis, respectively. $H^w_{hw\_e}, H^w_{hn\_e}$ are the projection of the angular momentum of the rotor in the outer-ring coordinates caused by assembly error under the effect of $\dot{\theta_w}$ and $\dot{\theta_n}$, respectively, and $H^w_{nw\_e}, H^w_{nn\_e}$ are the projection of the angular momentum of the inner-ring in the outer-ring coordinates caused by assembly error under the effect of $\dot{\theta_w}$ and $\dot{\theta_n}$, respectively. According to Eqs. (28)–(41), we can get an approximate dynamic formula of gyro drift as

$$d = D(M_x, M_y, \Omega, \delta_{hx}, \delta_{hy}, \delta_n) \tag{42}$$

According to the model construction method proposed in Seciton3, we first obtained 30 pieces of measured data S_r collected by adept operators. Second, we obtained 70 samples as the input features of the prior dataset S_p, and calculated the drift performance by Eq. (42) as the objective value. In addition, the error compensation dataset S_e was constructed from the input in S_r, and the corresponding

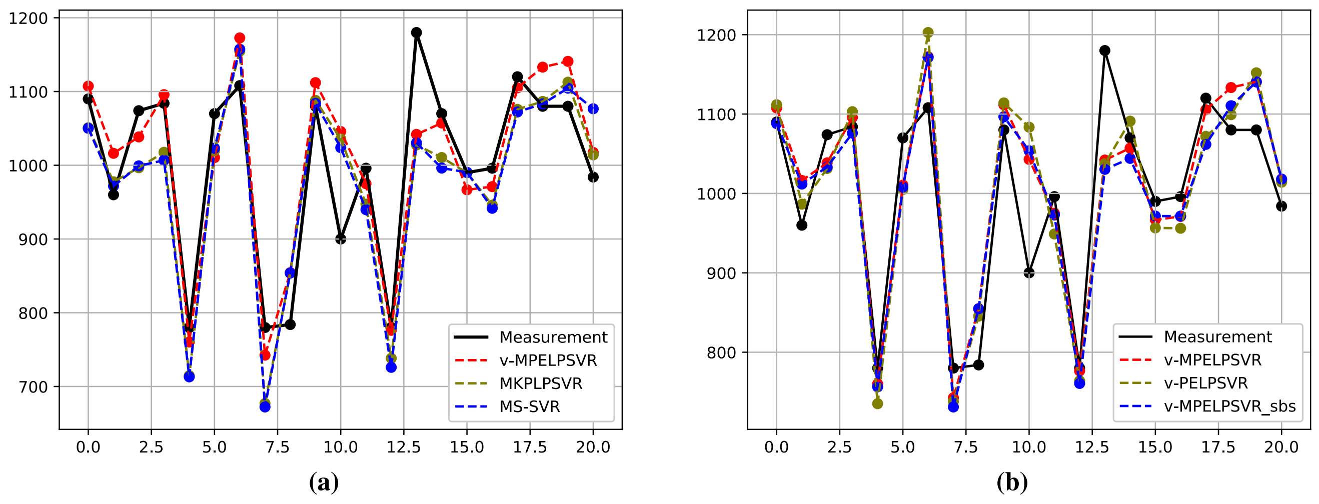

Fig. 7 Comparison of the gyro drift prediction results and the measurement result. **a** Results of measurement,ν-MPESVR, MKPLPSVR, and MSSVR. **b** Results of measurement,ν-MPESVR,ν-PESVR, and ν-MPESVR$_{sq}$

Table 2 Result comparisons of MKPLPSVR, MSSVR, ν-PESVR, ν-MPESVR$_{sq}$, ν-MPESVR on the coordinator gyro rotor performance prediction example

Algorithm	RMSE	MAE	R(2)
MKPLPSVR	66.3795	54.5286	0.7237
MSSVR	69.8931	58.5169	0.7055
ν-PESVR	66.7652	52.6081	0.7386
ν-MPESVR$_{sq}$	61.0216	46.9493	0.7472
ν-MPESVR	**57.4937**	**44.4290**	**0.7811**

Bold values indicate the best ones

drift performance was calculated by Eq. (42). Thus, we obtained a measured dataset S_r, a prior dataset S_p, and an error compensation dataset S_e. Finally, we used 20 sets of testing data S_{test} to verify the models.

As in Sect. 4.1, we used MKPLPSVR, MSSVR, ν-PESVR, ν-MPESVR$_{sq}$, and ν-MPESVR separately to build the data-driven model, and 20 pieces of measured data, S_{test}, were used to validate the accuracy and generalization performance of the obtained model. In this example, the single-kernel algorithm exploited only a Gaussian kernel, and the multi-kernel algorithms employed three Gaussian kernels. In ν-PESVR, we chose $C_r = 150, C_p = 140$, $C_e = 100$, and $\nu_r = \nu_p = \nu_e = 0.5$, and the kernel parameters $\sigma = 9.8058$, and $\sigma_e = 2.2361$. In ν-MPESVR$_{sq}$ and ν-MPESVR, we chose $C_1 = 0.0126, C_2 = 2.1380, C_3 = 23.1623, C_r = 150, C_p = 100$, $C_e = 100$, and $\nu_r = \nu_p = \nu_e = 0.5$, and the kernel parameters $\sigma_1 = 79.0569, \sigma_2 = 1.000, \sigma_3 = 0.3162$, and $\sigma_e = 2.2361$. Moreover, the parameters in MKPLPSVR and MSSVR are the same as those in ν-MPESVR$_{sq}$ and ν-MPESVR, except for the parameters related to error compensation. Using the training data samples and the model parameters above, we built the models separately using the five algorithms. Figure 7 shows the approximating results of the gyro drift. Table 2 shows the RMSE, MAE, and R(2) of the five predicted functions.

As we can see in Fig. 7, the results estimated by ν-MPESVR are in good agreement with the measurements from the five algorithms. As can be seen in Table 2, compared with the single-kernel algorithm, the multi-kernel algorithms have higher accuracy, especially those implementing error compensation. The model developed by ν-MPESVR is more accurate than the others, and has the best RMSE, MAE, and R(2) among the five algorithms. The projected results indicate that ν-MPESVR can effectively compensate for the errors between the prior knowledge data and the measured data, improve the accuracy of the obtained model, and has good generalization performance in the case of insufficient measured data.

5 Conclusion

For the purpose of building an accurate data-driven model from an insufficient amount of measured data and biased prior knowledge data, this paper proposed a nested multi-scale ν-LPSVR algorithm implementing prior knowledge and error compensation. The ν-MPESVR algorithm realized the incorporation of prior knowledge and the compensation of biased prior knowledge data by the addition of constraints and objective function. The proposed algorithm also used multi-scale kernel functions to incorporate multiple feature spaces into the process of modeling, which achieved accurate modeling of non-flat, complex, and changeful problems. Moreover, a strategy for hyper-parameter selection based on expert knowledge and CPSO was presented to search the optimal parameters of the ν-MPESVR algorithm autonomously.

In a word, the major contributions of this study are highlighted as follows: (1) ν-MPESVR considers the errors between the prior knowledge data and the measured data in the modeling process, and incorporates the compensated prior knowledge into the prediction model. Hence, the estimation accuracy of ν-MPESVR is significantly better than MSSVR and MKPLPSVR, when there are biases between the prior knowledge data and the measured data; (2) unlike the method of sequentially executing error compensation and prediction (ν-MPESVR$_{sq}$), ν-MPESVR simultaneously optimizes the prediction model and the error compensation model, so the proposed algorithm has better generalization performance; (3) the ν-MPESVR model is constructed automatically where the model parameters are optimized autonomously with the use of the expert knowledge based CPSO algorithm; (4) the results of a synthetic example and a practical application demonstrate that ν-MPESVR can still maintain high accuracy when the biases of prior knowledge data change, and it has higher prediction accuracy and better generalization performance. Accordingly, the proposed algorithm ν-MPESVR shows great potential in solving problems when only a few measured samples can be obtained, but some approximate and biased prior knowledge is available.

In the future research, the correlation between model parameters will be considered to narrow the parameter spaces that the intelligent optimization algorithm needs to search, and integrating the prior knowledge within intelligent searching processes will be explored to enhance the stability of the algorithm.

Acknowledgements Support from the National Natural Science Foundation of China (Grant Nos. 51475418, 51490663, 51475417) is gratefully acknowledged.

References

1. Aslahi-Shahri BM, Rahmani R, Chizari M, Maralani A, Eslami M, Golkar MJ, Ebrahimi A (2016) A hybrid method consisting of GA and SVM for intrusion detection system. Neural Comput Appl 27(6):1–8
2. Bishop CM (2006) Pattern recognition and machine learning (Information science and statistics). Springer, New York
3. Bloch G, Lauer F, Colin G, Chamaillard Y (2008) Support vector regression from simulation data and few experimental samples. Inf Sci Int J 178(20):3813–3827
4. Chang C, Lin C (2002) Training v-support vector regression: theory and algorithms. Neural Comput 14(8):1959–1977
5. Chen J, Xue X, Ha M, Yu D (2014) Support vector regression method for wind speed prediction incorporating probability prior knowledge. Math Probl Eng 2014(2014):1–10
6. Cherkassky V, Ma Y (2004) Practical selection of SVM parameters and noise estimation for SVM regression. Neural Netw 17(1):113–126
7. Farooq T, Guergachi A, Krishnan S (2010) Knowledge-based Green's kernel for support vector regression. Math Probl Eng 1024(1024–123X):16
8. Hasanipanah M, Shahnazar A, Amnieh HB, Armaghani DJ (2017) Prediction of air-overpressure caused by mine blasting using a new hybrid PSO–SVR model. Eng Comput 33(1):23–31
9. Lauer F (2008) Incorporating prior knowledge in support vector regression. Mach Learn 70(1):89–118
10. Lauer F, Bloch G (2008) Incorporating prior knowledge in support vector machines for classification: a review. Neurocomputing 71(7):1578–1594
11. Leung AYT, Zhang H (2009) Particle swarm optimization of tuned mass dampers. Eng Struct 31(3):715–728
12. Leung AYT, Zhang H, Cheng CC, Lee YY (2010) Particle swarm optimization of TMD by non-stationary base excitation during earthquake. Earthq Eng Struct Dyn 37(9):1223–1246
13. Liu B, Wang L, Jin YH, Tang F, Huang DX (2005) Improved particle swarm optimization combined with chaos. Chaos Solitons Fractals 25(5):1261–1271
14. Lu Z, Sun J (2009) Non-mercer hybrid kernel for linear programming support vector regression in nonlinear systems identification. Appl Soft Comput 9(1):94–99
15. Lu Z, Sun J, Butts K (2014) Multiscale asymmetric orthogonal wavelet kernel for linear programming support vector learning and nonlinear dynamic systems identification. IEEE Trans Cybern 44(5):712
16. Milner S, Davis C, Zhang H, Llorca J (2012) Nature-inspired self-organization, control, and optimization in heterogeneous wireless networks. IEEE Trans Mobile Comput 11(7):1207–1222
17. Moazami S, Noori R, Amiri BJ, Yeganeh B, Partani S, Safavi S (2016) Reliable prediction of carbon monoxide using developed support vector machine. Atmos Pollut Res 7(3):412–418
18. Noori R, Abdoli M, Ghasrodashti AA, Jalili Ghazizade M (2008) Prediction of municipal solid waste generation with combination of support vector machine and principal component analysis: a case study of mashhad. Environ Prog Sustain Energy 28(2):249–258
19. Noori R, Deng Z, Kiaghadi A, Kachoosangi FT (2016) How reliable are ANN, ANFIS, and SVM techniques for predicting longitudinal dispersion coefficient in natural rivers? J Hydraul Eng 142(1):04015039
20. Noori R, Karbassi A, Moghaddamnia A, Han D, Zokaei-Ashtiani M, Farokhnia A, Gousheh MG (2011) Assessment of input variables determination on the SVM model performance using PCA, gamma test, and forward selection techniques for monthly stream flow prediction. J Hydrol 401(3):177–189
21. Platt JC (1999) Fast training of support vector machines using sequential minimal optimization. In: Schölkopf B, Burges C, Smola A (eds) Advances in kernel methods, chap 12. MIT press, Cambridge, MA, pp 185–208
22. Sartakhti JS, Afrabandpey H, Saraee M (2017) Simulated annealing least squares twin support vector machine (SA-LSTSVM) for pattern classification. Soft Comput 21(15):4361–4373
23. Schölkopf B, Smola AJ, Williamson RC, Bartlett PL (2000) New support vector algorithms. Neural Comput 12(5):1207–1245
24. Smola A, Scholkopf B, Ratsch G (1999) Linear programs for automatic accuracy control in regression. In: Ninth international conference on artificial neural networks, 1999. ICANN 99, vol 2, pp 575–580
25. Smola AJ, Schölkopf B (2004) A tutorial on support vector regression. Stat Comput 14(3):199–222

26. Vapnik VN (1997) The nature of statistical learning theory. IEEE Trans Neural Netw 38(4):409
27. Wang Y, Jiang P (2017) Fluctuation evaluation and identification model for small-batch multistage machining processes of complex aircraft parts. Proc Inst Mech Eng Part B J Eng Manuf 231(10):1820–1837
28. Zhang W, Yu L, Yoshida T, Wang Q (2018) Feature weighted confidence to incorporate prior knowledge into support vector machines for classification. Knowl Inf Syst 1:1–27
29. Zhao W, Tao T, Zio E, Wang W (2016) A novel hybrid method of parameters tuning in support vector regression for reliability prediction: particle swarm optimization combined with analytical selection. IEEE Trans Reliab 65(3):1393–1405
30. Zhao Y, Jiang P (2017) Angular rate sensing with gyrowheel using genetic algorithm optimized neural networks. Sensors 17(7):1692
31. Zheng D, Wang J, Zhao Y (2006) Non-flat function estimation with a multi-scale support vector regression. Neurocomputing 70(1):420–429
32. Zhou J, Duan B, Huang J, Cao H (2014) Data-driven modeling and optimization for cavity filters using linear programming support vector regression. Neural Comput Appl 24(7–8):1771–1783
33. Zhou J, Duan B, Huang J, Li N (2015) Incorporating prior knowledge and multi-kernel into linear programming support vector regression. Soft Comput 19(7):2047–2061