EXPLORATION OF EFFICIENT HYPERPARAMETERS ADAPTION OF SUPPORT VECTOR REGRESSION FOR AERODYNAMIC DESIGN

Ke-Shi Zhang[1,2], Hai-Long Qiao[1,2], Peng-Hui Wang[1,2], You-Quan Du[1,2] & Zhong-Hua Han[1,2]

[1]School of Aeronautics, Northwestern Polytechnical University, Xi'an, 710072, P.R.China
[2]National Key Laboratory of Aircraft Configuration Design, Xi'an, 710072, P.R.China

Abstract

Support vector regression (SVR), due to good generalization capability that has been validated in machine learning and pattern recognition, was introduced into aerodynamic design to build surrogate models based on the training data with numerical noise in our former work. However, hyperparameters tuning is still a key problem to solve because it not only has critical impact on the prediction accuracy but also brings high computational cost. Therefore, the hyperparameter optimization model and algorithms are investigated in this work. The objective of the hyperparameter optimization model, generalization error (GE), is obtained via the popular cross validation (CV) method, and compared with the leave-one-out bound (LooB) method due to its high efficiency. The hyperparameter design spaces are plotted and it is found that the curves of GE w.r.t the hyperparameters (the insensitive factor, penalty factor and kernel parameter) are commonly characteristic of multi-modal, large "flat" region and non-smoothness. Therefore, the gradient optimization is not recommended because of its local-search attribute. Three popular global optimization algorithms, including the Genetic Algorithm (GA), Bayesian Optimization (BO) and Covariance Matrix Adaptation Evolution Strategy (CMA-ES), are applied to hyperparameters adaption and compared via a set of benchmark problems to evaluate its training efficiency and prediction accuracy for analytical problems of low/high nonlinearity, based on the samples with low/medium/high intensity noise. The results show that, 1) the design space of hyperparameters tunning is characteristic of multi-modal, large "flat" region and non-smoothness; 2) In terms of accuracy, CMA-ES behaves well for almost all the test cases, while BO is better in the low-dimensional (≤ 10) cases and is still comparable in the higher-dimensional cases when the noise is not too strong but becomes slightly worse when the noise becomes stronger; 2) In the high-dimensional (>10) cases, the BO algorithm has apparent superiority of efficiency; 3) the parallel CV can not only enable higher mode accuracy but also has high efficiency even faster than LooB. Finally, it is applied in modeling based on the computational aerothermal data and the wind-tunnel experimental data respectively, in which the reasonable results are obtained.

Keywords: support vector regression; hyperparameters adaption; cross validation; leave-one-out bound; global optimization algorithm

1. Introduction

It is well known that all numerical simulations are not "clean": numerical noises always exist, as they are based on discretization. In the computational fluid dynamics (CFD) simulations, coarse grid, strong shock wave, apparent flow separation, … may have convergence problems so that incur numerical noises. Numerical noises, the numerically induced oscillations with small wavelengths, was sometimes particularly troublesome, which can lead to problems in identifying optimum designs and will become a hindrance for the further applications of the aerodynamic optimization.

A regression surrogate model can inherently filter numerical noises, in addition, surrogate-based optimization (SBO) is an efficient global-optimization (EGO) method that is widely used in the design optimizations associated with different areas of aerospace science and engineering. When these methods are applied, it is hopeful that the numerical noises can be filtered so that the subsequent optimization will be less affected.

SVR[1] is usually considered as a special case of support vector machines[2] (SVMs) that is popular

in machine learning and pattern recognition. SVR has been proven to have good generalization capability and robustness in data regression[15], i.e., approximating highly nonlinear functions well and not so sensitive to numerical noises. However, for SVR, the predictive performance is critically affected by the hyperparameters values used to train them, like most of the machine learning algorithms. Inappropriate setting can lead to overfitting or underfitting. Then, selecting an optimal model that minimizes error and generalizes well to the unseen data derives a problem of tuning or optimizing these parameters. It is called hyperparameters adaption. Hyperparameters adaption usually accompanies with high computational cost, especially on larger datasets, while the tuning settings do not always significantly outperform the default values. So, the manual setting is preferred in many real problems and popular SVR codes like [LibSVM][3], and efficient and effective adaption of the hyperparameters is still an ongoing extremely important issue of the SVR-related researches. Despite the advances, hyperparameter tuning on large datasets remains challenging.

For the classical ε-SVR, we have the insensitive factor ε to control the width of the ε-tube, the penalty factor C to determine the tradeoff between training accuracy and model complexity, and the kernel parameter σ when introducing nonlinearity by the radial basis function (RBF). Traditionally, grid search was employed to search the optimal parameters by varying them with a fixed step size through the parameter space and evaluate each parameter combination. Undoubtfully, it's time-consuming and not suitable for the engineering problems with computational-expensive analysis. The random search runs faster, but probably miss the optimum. The gradient optimization algorithms perform fast search in the parameter space but may trap in a local optimum. Then, various heuristic (or so-called nature-inspired) algorithms that are not relying on the gradients were introduced for tuning parameters, including GA[4][5], particle swarm optimization[6][7] (PSO), Harris Hawks optimization[8] (HHO), grasshopper optimization algorithm[9] (GOA), grey wolf optimizer[10]19 (GWO), etc. The CMA-ES[11], as an Evolutionary Strategy (ES) algorithm, is also used for this purpose due to its good capabilities of fast, global search through the non-convex parameter space. In addition, the BO has emerged as an efficient method for tuning hyperparameters in variety of surrogate models including SVR. It offers robust solutions for optimizing expensive black-box functions, using a Gaussian Process as a probabilistic measure to model the unknown function and guides the search focusing on the region including the global optimum in a short time. Due to its robustness and high efficiency, BO[12] is becoming popular for hyperparameters adaption in variety of machine learning algorithms.

This paper aims to explore the most efficient method of hyperparameters adaption by comparing two optimization models and several popular optimization algorithms, in order to make the SVR model applicable to the aerodynamic design problems and the other time-consuming engineering design problems. This paper is organized as following. Section 2 gives a brief introduction to the SVR theory. In Section 3 and Section 4, the hyperparameter-adaption models are established based on the CV method and the LooB method respectively. The hyperparameters design space, modeling error and computational cost are analyzed. Section 5 explains process of the hyperparameters adaption and briefly introduces the optimization algorithms. In Section 6, the modeling methods and optimization algorithms are systematically investigated and compared via series of numerical examples and then preliminarily applied in the aerodynamic design.

2. Background: support vector regression

The ε-SVR is one of the most popular SVR methods. All the investigations on the hyperparameter-adaption methods will be performed based on it.

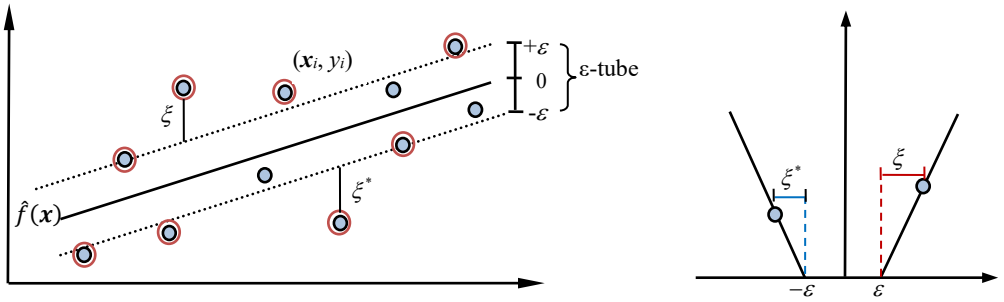
Given is a training data set $\boldsymbol{D} = (\boldsymbol{X}_S, \boldsymbol{Y}_S) = \{(\boldsymbol{x}^{(i)}, y^{(i)}) | i = 1,2,\cdots,n\}$, where $\boldsymbol{x}^{(i)} \in \mathbb{R}^m$ denotes the input vector and $y^{(i)}$ is its corresponding response ($y^{(i)} = f(\boldsymbol{x}^{(i)})$). In ε-SVR, $\boldsymbol{x}$ is first mapped to $\boldsymbol{z} = \psi(\boldsymbol{x})$ in a feature space via a nonlinear map ψ that is often called kernel function. Then a regression function $\hat{f}(\boldsymbol{x}) = \langle \boldsymbol{w} \cdot \psi(\boldsymbol{x}) \rangle + b$ is constructed so that it deviates least from the training set according to Vapnik's ε-insensitive loss function

$$L_I(x) = \begin{cases} 0 & \text{if } \left|\hat{f}(x) - y\right| < \varepsilon \\ \left|\hat{f}(x) - y\right| - \varepsilon & \text{otherwise} \end{cases} \tag{1}$$

while at the same time is as "flat" as possible (i.e., $\|\boldsymbol{w}\|$ is as small as possible). Mathematically, this means

$$\begin{aligned} \min\ & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\left(\xi_i + \xi_i^*\right) \\ \text{s.t.}\ & y_i - \langle w \bullet \boldsymbol{x}_i \rangle - b \le \varepsilon + \xi_i \\ & \langle w \bullet \boldsymbol{x}_i \rangle + b - y_i \le \varepsilon + \xi_i^* \\ & \xi_i, \xi_i^* \ge 0 \end{aligned} \tag{2}$$

A key assumption of this formulation is that there exists a function $f(\boldsymbol{x})$ that can approximate all pairs of $(\boldsymbol{x}_i, y_i)$ in ε precision by a so-called "ε-tube" or "ε-insensitive zone". In a linear case, it is shown in Figure 1. The constant C>0 controls the tradeoff between complexity of model and the deviations larger than ε.



a) ε-tube in the sample space b) ε-insensitive loss

Figure 1 – ε-tube and error allowance of the ε-SVR (samples in a red circle: support vectors)

Nonlinear regression can be achieved by simply processing the training data by mapping into some feature space and then applying the linear SVR algorithm, i.e. replacing the dot product of input vectors with a nonlinear transformation on the input vectors. This transformation is achieved by the so-called kernel function, $k(\boldsymbol{x}, \boldsymbol{x}')$. Table 1 lists some common kernel functions. Gaussian radial basis function (RBF)[13], as the most commonly used kernel function, will be adopted in all of the examples in this paper.

Table 1 – Common kernel functions

Kernel function	Expression
Linear	$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle$
Polynomial	$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{x}, \boldsymbol{x}' \rangle^d$
Gaussian RBF	$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\sigma^2}\right)$
Exponential RBF	$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|}{2\sigma^2}\right)$
Multi-layer perceptron	$k(\boldsymbol{x}, \boldsymbol{x}') = \tanh(\beta\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + \theta)$

Applying the kernel function into the dot product of input vectors, the following optimization problem is obtained.

$$\begin{aligned} \max & \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{n}\left(\alpha_i - \alpha_i^*\right)\left(\alpha_j - \alpha_j^*\right)k\left(x_i, x_j\right) \\ -\varepsilon\sum_{i=1}^{n}\left(\alpha_i + \alpha_i^*\right) + \sum_{i=1}^{n} y_i\left(\alpha_i - \alpha_i^*\right) \end{cases} \\ \text{s.t.}\ & \sum_{i=1}^{n}\left(\alpha_i - \alpha_i^*\right) = 0 \\ & \alpha_i, \alpha_i^* \hat{I}\left[0, C\right] \end{aligned} \tag{3}$$

Then the regression formulation becomes

$$\hat{f}(x) = \sum_{i=1}^{n}\left(\alpha_i - \alpha_i^*\right)k\left(\boldsymbol{x}_i, \boldsymbol{x}\right) + b \tag{4}$$

3. Hyperparameter adaption modeling

3.1 Optimization problem

The hyperparameters critically affect the predictive performance of the model. As such, selecting an optimal model that minimizes error and generalizes well to unseen data becomes a problem of tuning or optimizing these hyperparameters. The error can be approximately taken as the sum of bias and variance[14], neglecting a random error that is unavailable in the data of learning process,

$$[f(x_0) - \hat{f}(x_0)]^2 = \underbrace{Var[\hat{f}(x_0)]}_{\text{variance}} + \underbrace{Bias[\hat{f}(x_0)]}_{\text{bias}}^2 \tag{5}$$

Bias is the error introduced by approximating a real-world phenomenon, while variance measures the sensitivity of $\hat{f}$ to the training dataset and how much its fit would change if estimated using different data. Figure 2 illustrates the bias and variance trade-off with respect to model complexity. In general, as a model becomes more complicated or flexible, bias decreases and variance increases. In other words, a flexible $\hat{f}$ fits closer to given training dataset but is more sensitive to training data variability. Too much variance leads to $\hat{f}$ overfitting the data, and too much bias leads to $\hat{f}$ underfitting the data. Therefore, selecting an optimal $\hat{f}$, ranging in complexity from simple linear to highly nonlinear, involves balancing the bias and variance trade-off. Possessing both low bias and low variance gives the learned model a higher probability of generalizing well to unseen data during model training and predict more accurately.

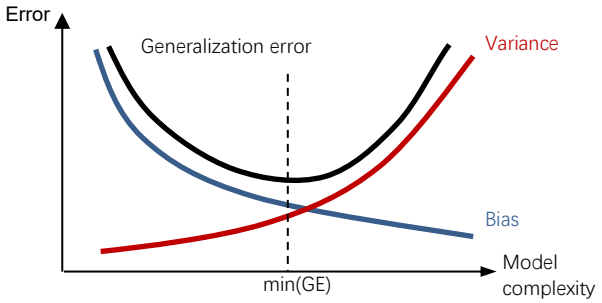


Figure 2 – Bias and variance trade-off w.r.t. model complexity

The hyperparameters of ε-SVR include (ε, C, σ). The parameter ε controls the width of the ε-tube; the parameter C penalizes any deviation beyond the tube and determines the tradeoff between training accuracy and model complexity; and the kernel parameter σ affects flexibility of the approximation function: a very small σ means the function is more localized, while a large σ makes it less flexible. The hyperparameter adaption is an optimization process in essence, which is to search for the optimal hyperparameters combination that minimizes an estimate of the generalization error (GE):

$$\min \lg GE(\varepsilon, C, \sigma) \tag{6}$$

We set the design range of (ε, C, σ) as listed in Table 2 by experience. Although the value of ε is proportional to noise variance, such a setting is proven to be appropriate for our test cases.

Table 2 – Design region of the hyperparameters

Logarithm of the hyperparameters	$\lg\varepsilon$	$\lg C$	$\lg\sigma$
Design range	[-7,1]	[-1,7]	[-2,2]

3.2 GE estimation methods

For solving the optimization problem of hyperparameters adaption in Eq.(6), one of the key problems is how to evaluate GE. Taking both accuracy and efficiency into consideration, the cross-validation method and leave-one-out bound method will be compared for this purpose.

3.2.1 CV method

CV method[15]-[17] is the most common method for tuning hyperparameters for variety of surrogate models. The advantages of CV are that, in most cases, it captures the actual test error that balances bias and variance well enough[18].

The procedure of CV randomly and evenly divides the given dataset D into k subsets ($D_1 \cup D_2 \cup \cdots \cup D_k = D; D_i \cap D_j = \emptyset, i \neq j, i, j = 1,2,\cdots,k$), and uses k-1 subsets for training and the remaining one for testing, which is called k-fold CV. This process is repeated k times by changing the remaining subset. Then the GE is evaluated by the MSE (mean squared error), over all test results, as following

$$MSE_{\mathrm{CV}} = \frac{1}{k}\sum_{i=1}^{k}\left[\frac{1}{s_i}\sum_{t=1}^{s_i}\left(y_t - \hat{y}_t\right)^2\right] \tag{7}$$

where s_i is number of the samples in the ith subset. The k-fold CV is the most popular way of performing cross validation. Theoretically the result of CV gets closer to unbiased estimate of the modeling error when increasing k, however its computational cost might be unacceptable, which often happens in engineering designs, as model training must be repeated k times in each objective evaluation. Typical choice of k is between 5 and 10. From the consideration of efficiency, k=5 is applied in this work, the basic principle of which is shown in Figure 3.

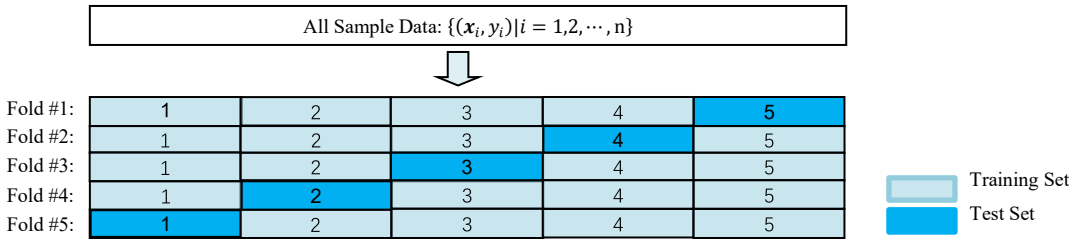


Figure 3 – An overview of 5-fold CV

3.2.2 LooB method

The leave-one-out (LOO) method is the extreme case of CV, in which a single sample is excluded from the training set and used for evaluating the model. The LOO error is defined as

$$LOO = \frac{1}{n}\sum_{t=1}^{n}\left|y_t - \hat{y}_t\right| \tag{8}$$

LOO provides an unbiased estimate of the true GE, however, is highly time consuming. Therefore, a more popular approach is to approximate the error by its upper bound that is a function of the hyperparameters. Then we search for parameters so that this bound is minimized, which must lead to minimization of the GE.

The LooB is computationally efficient, so that it's used in this work as the alternative method for approximating the GE, to explore an efficient method of hyperparameters adaption. The LOO bounds, such as radius margin bound and span bound for L2-SVR, and the bound for L1-SVR, were derived in Ref.[19]. Here we use the bound for L1-SVR that matches the SVR optimization model in Eq.(3) and is a commonly used form for regression, as following:

$$LooB = \sum_{t=1}^{n}\left(\alpha_i + \alpha_i^*\right)S_t^2 + \sum_{t=1}^{n}\left(\xi_i + \xi_i^*\right) + n\varepsilon \tag{9}$$

It is obvious that the LooB value obtained via Eq.(9) will vary within a wide range for large-scale sample problems (i.e. n is large), which is not easy for observation. So, we divide it by the number of samples, n, then get a varied form:

$$LooB = \frac{1}{n}\left(\sum_{t=1}^{n}\left(\alpha_i + \alpha_i^*\right)S_t^2 + \sum_{t=1}^{n}\left(\xi_i + \xi_i^*\right)\right) + \varepsilon \tag{10}$$

S_t^2 is not a continuous function, so a modified item is proposed:

$$\tilde{S}_t^2 = \frac{1}{\left(\tilde{M}^{-1}\right)_{tt}} - \tilde{D}_{tt} \tag{11}$$

in which

$$\tilde{M} = \begin{bmatrix} K_{SV} + D_{SV} & 1_{SV} \\ 1_{SV}^T & 0 \end{bmatrix} \text{ and } \tilde{D}_{tt} = \frac{\eta}{\alpha_i + \alpha_i^*}$$

where η is a user defined positive constant for smoothing regularization and is set to 0.01 in current

work. SV is number of free support vectors that are the samples lying on the bounds of ε-tube.

Eq.(10) will be used for objective evaluation for the hyperparameter optimization in Eq.(6). Obviously, its computational cost will usually be much lower than that of the CV method because model training is implemented only once in each objective evaluation. However, Eq.(11) indicates that inverse of matrix $\widetilde{M}$ will incur computational burden when applying the LooB method if SV is large.

The LooB in Eq.(10) was derived under the assumption that the set of support vectors remains the same during the leave-one-out procedure. This assumption is valid in condition that the function is smooth because most of the samples are always free support vectors. However, when there are numerical noises, only few samples are free support vectors and most samples are inside and some are outside ε-tube, and the set of free support vectors varies a lot in the process of hyperparameters adaption. That's why the LooB method becomes not so accurate as the CV method when numerical noises exist.

4. Some observations on the hyperparameter adaption model

In this section, we aim to know more about the hyperparameter adaption model, by observing what the design space looks like and how the computational cost is to measure the GE. This work will be done based on the classical sinc function in Eq.(12). Besides the smooth function, the Gaussian noises subject to normal distribution, $N(0,\sigma_n^2)$, are also added to simulate the noisy function for investigating the related attributes.

$$f(x)=\mathrm{a}\sin x/x,\quad x\in[-10,10] \tag{12}$$

Firstly, from 100 evenly and smoothly distributed samples, the hyperparameters are set to get the baseline prediction model in Figure 4. Then the remained research in this sub-section will be done based on this baseline model by adjusting one of the hyperparameters while keeping the others constant.

Table 3 – Empirical hyperparameter setting for the sinc function without and with numerical noises

Function	N	σ_n	ε	C	σ
Sinc (noise free)	100	0	1e-6	1e5	0.09
Sinc (with noise)	100	2	1e-4	300	0.14

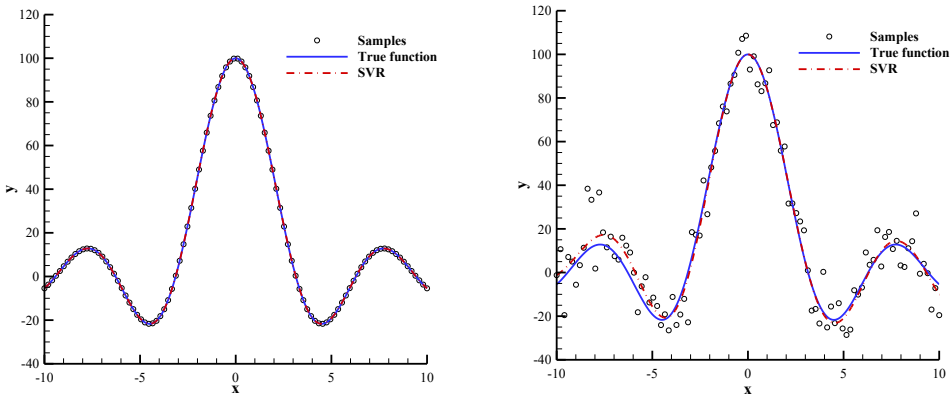


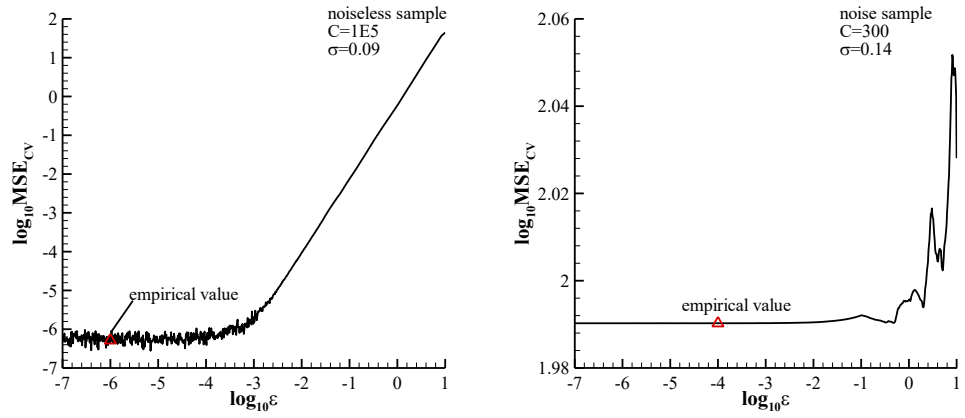
Figure 4 – SVR predictions of the sinc function (left: noise free, right: with noise)

To show how the change of the hyperparameters affects the GE of the prediction model, the design spaces of them are illustrated, so that an appropriate optimization algorithm would be applied subsequently.

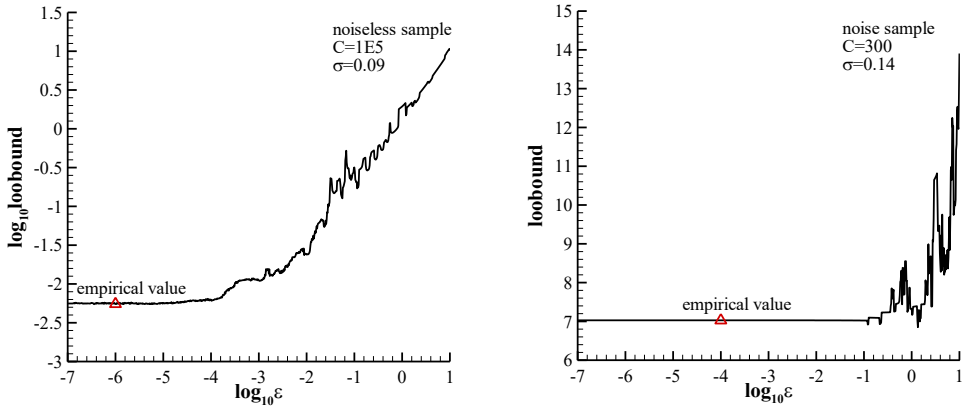
(1) insensitive factor ε

Parameter ε controls the width of the ε-insensitive tube, which is proportional to noise variance. A very thin ε-tube does not provide enough margin to tolerate the numerical noise in data points, so SVR function tends to interpolate the data, while a thick ε-tube has enough margin, having a tendency to get flat to generate a regression fit. For a given dataset (smoothly distributed or with numerical noise), the influence of ε on accuracy of the prediction model is illustrated in Figure 5, estimating GE by CV method via Eq.(7) or LooB method via Eq.(10). It is found that, 1) the minimal

GE can be obtained by changing ε; 2) the curve near the optimal ε seems relatively "flat", where the GE is not sensitive with change of ε, and the empirical value of ε in Table 3 lies in this region; 3) the ε curve behaves not smooth that might be caused by the numerical error that happened in SVR model training. The attributes 2) and 3) of the ε curve present a challenge to the optimization algorithms and indicate that a global optimization not dependent upon gradients should be adopted. The results in Figure 5 also prove the statement[20] that ε is proportional to noise variance.



(a) GE estimation by CV method

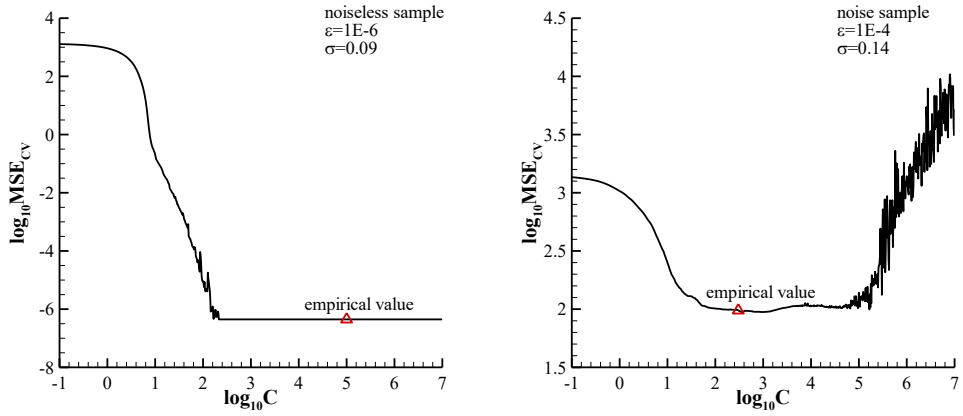


(b) GE estimation by LooB method

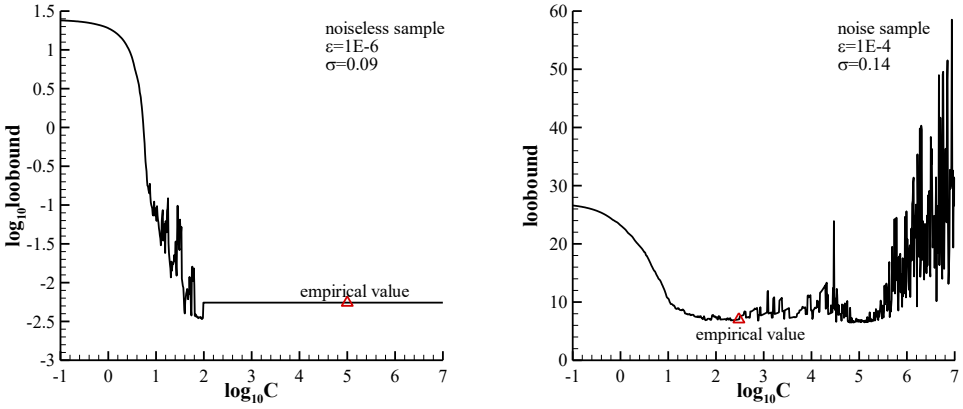
Figure 5 – Design space of the insensitive factor ε (left: noise free, right: with noise)

(2) penalty factor C

Parameter C determines the tradeoff between the model complexity (flatness) and the degree to which deviations larger than ε are tolerated. If C is too large (infinity), the objective in the SVR optimization formulation is to minimize the empirical risk only. If C is small, the penalty tends negligible and the SVR function gets flat. As shown in Figure 6, the design space of C behaves similar to that of ε, i.e. large "flat" region and oscillated curve, which indicates a gradient optimization method is not appropriate due to local search and gradient-dependent. Furthermore, it's proven that a relatively large value of C is preferred to punish any deviation out of the ε-tube to get a good prediction function when numerical noises don't exist. But in case of noisy samples, C should be carefully valued.



(a) GE estimation by CV method

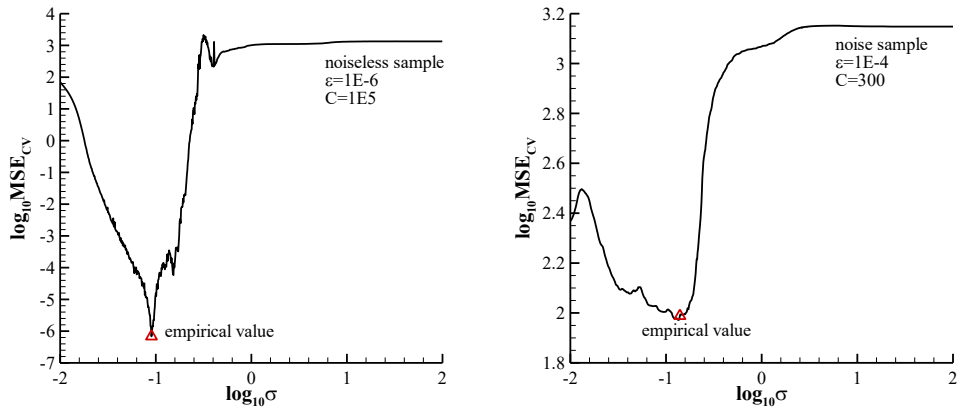


(b) GE estimation by LooB method

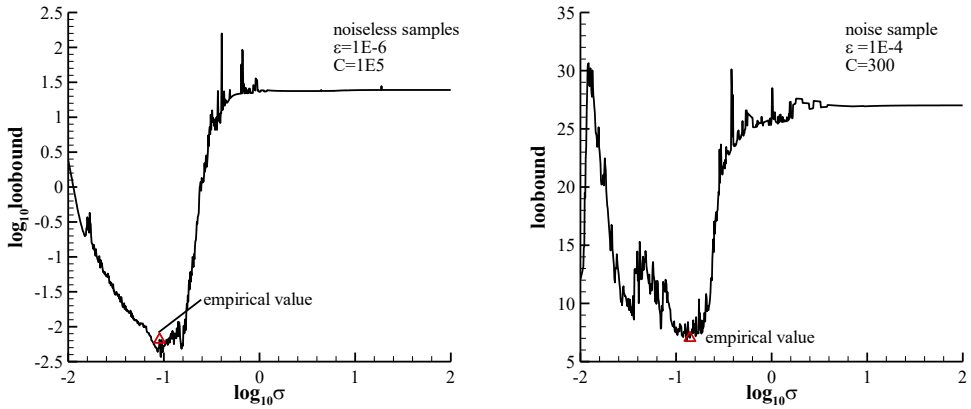
Figure 6 – Design space of the penalty factor C (left: noise free, right: with noise)

(3) kernel parameter σ

σ is the width parameter of RBF kernel function. A very small σ means the kernel is more localized, thus, the SVR function tends to overfit, while a large σ makes the SVR function less flexible. The influence of σ on the GE can be observed via Figure 7. Different from the curves of ε and C in Figure 5 and Figure 6, the GE is very sensitive to the change of σ near the optimum, which implies that we should be more cautious in valuing σ to prevent exacerbation of model accuracy.



a) GE estimation by CV method



b) GE estimation by LooB method

Figure 7 – Design space of kernel function parameter (left: noise free, right: with noise)

5. Optimization algorithms for hyperparameters adaption

In this section, we aim to evaluate effectiveness and efficiency of different optimization algorithms to search for the optimal hyperparameters. based on the numerical examples of low- and high-dimensional benchmarks. Due to local flatness, multimodality and non-smoothness of the design space observed in the last section, some popular global optimization algorithms for training hyperparameters will be applied and compared in this work, including GA, BO and CMA-ES.

5.1 Steady GA algorithm

Genetic algorithm, a type of heuristic algorithm, search for the optimal solution by simulating the natural evolutionary process. According to the different population generation mechanisms, Galib[21], a genetic algorithm program library based on C++, includes three basic types: Incremental GA, Simple GA and Steady GA. This paper adopts Steady GA to optimize hyperparameters because of its faster convergence speed and stability[22]. The Steady GA directly passes excellent individuals to the next generation in a proportion controlled by *nReplacement*. Meanwhile, only the individuals with low fitness undergo crossover and mutation operations. The Steady GA algorithm can be summarized below:

Algorithms 1 – Steady GA algorithm

Input: population size λ, crossover probability P_c, mutation probability P_m, nReplacement
Output: P_{t+1}

Initialize Population $P_0 = (x_1, \ldots, x_\lambda)$; t=0
repeat
  for (i=1 to λ) do
    Evaluate fitness $f(x_i)$
  end for
  Sort fitness in a descending order and its result is $f(x_{1:\lambda}) \geq \ldots \geq f(x_{\lambda:\lambda})$, in which $x_{i:\lambda}$ means the i -th best individual
  $P_{t+1} = (x_{1:\lambda}, \ldots, x_{\lambda(1-n\text{Replacement}):\lambda})$
  repeat
    Get x_{parents} by roulette wheel selection operation to $C_t = (x_{\lambda(1-n\text{Replacement})+1:\lambda}, \cdots, x_{\lambda:\lambda})$
    if $random(0,1) < P_c$ then
      Crossover to generate offspring individuals x_{children}
    if $random(0,1) < P_m$ then
      Mutation to x_{children}
    Evaluate fitness $f(x_{\text{children}})$
    if $f(x_{\text{children}}) \geq f(x_{\text{parents}})$ then
      Add x_{children} to P_{t+1}
    else
      Add x_{parents} to P_{t+1}
  until (the length of P_{t+1} is λ)
  $t = t + 1$
until stop condition met

5.2 CMA-ES algorithm

CMA-ES is an evolution algorithm for non-linear non-convex optimization problems in continuous domain [23]. Its core idea is to adjust the search direction by adjusting the covariance matrix, increasing the probability of generating good solutions. The algorithm can be divided into four aspects: sampling, updating the mean, updating the covariance matrix, and controlling the step size.

Algorithms 2 – CMA-ES algorithm

Input: population size λ, step size $\sigma^{(0)}$
Output: $x_{1:\lambda}^{(t+1)}$

Initialize $c_c, c_1, c_\sigma, d_\sigma, \mu_\omega, \mu$
Initialize mean vectors $m^{(0)}$; $C^{(0)} = I$; $p_\sigma^{(0)} = 0$; $p_c^{(0)} = 0$; t=0
repeat
$x_i^{(t+1)} = m^{(t)} + \sigma^{(t)} N(0, C^{(t)}), i = 1, \ldots, \lambda$
    Evaluate fitness $f(x_i^{(t+1)}), i = 1, \ldots, \lambda$
    Sort fitness in a descending order and its result is $f(x_{1:\lambda}^{(t+1)}) \geq \ldots \geq f(x_{\lambda:\lambda}^{(t+1)})$, in which $x_{i:\lambda}$ means the i -th best individual;

$$m^{(t+1)} \leftarrow \sum_{i=1}^{\mu} \omega_i x_{i:\lambda}^{(t+1)}$$

$$p_\sigma^{(t+1)} \leftarrow (1 - c_\sigma) p_\sigma^{(t)} + \sqrt{c_\sigma (2 - c_\sigma) \mu_\omega} C^{(t)^{-\frac{1}{2}}} \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}}$$

$$\sigma^{(t+1)} \leftarrow \sigma^{(t)} exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\left\| p_\sigma^{(g+1)} \right\|}{E\|N(0, I)\|} - 1 \right) \right)$$

$$p_c^{(t+1)} \leftarrow (1 - c_c) p_c^{(t)} + \sqrt{c_c (2 - c_c) \mu_\omega} C^{(t)^{-\frac{1}{2}}} \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}}$$

$$C^{(t+1)} \leftarrow (1 - c_\sigma - c_\mu) C^{(t)} + c_1 p_c^{(t+1)} p_c^{(t+1)^T} + c_\mu \sum_{i=1}^{\mu} \omega_i \left(\frac{x_{i:\lambda}^{(g+1)} - m^{(g)}}{\sigma^{(g)}} \right) \left(\frac{x_{i:\lambda}^{(g+1)} - m^{(g)}}{\sigma^{(g)}} \right)^T$$

    $t = t + 1$
until stop condition met

5.3 Bayesian optimization algorithm

Bayesian optimization is one of the most advanced and promising techniques in the fields of probabilistic machine learning and artificial intelligence, capable of obtaining an approximate optimal solution at a low cost of evaluations. It can be concluded as follows:

Algorithms 3 – BO algorithm

Input: black-box function f, hyperparameter search space χ, max iterations T
Output: best result

Initialize $D_0 \leftarrow ((x_1, y_1), \ldots, (x_n, y_n))$, in which $y_i = f(x_i)$
t=0;
repeat
$p(y|x, D_t) \leftarrow FitModel(D_t)$
    $x_i \leftarrow argmax_{x \in \chi}\, \alpha\,(x, p(y|x, D_t))$, in which α is Acquisition Function
    $y_i \leftarrow f(x_i)$
    $D_{t+1} \leftarrow D_t \cup (x_i, y_i)$
    $t = t + 1$
until t=T

6. Analytical benchmark test cases

In this section, different hyperparameters adaption schemes, i.e. different optimization algorithms applied to search for the minimal GE estimated by the CV or LooB method, will be evaluated in terms of modeling accuracy and efficiency. We employ seven analytical benchmark functions (listed in Table 4) and the Gaussian noise is added to model noisy responses. The test functions are chosen to cover variety of problem properties and dimensions. Rosenbrock is a unimodal function that is characteristic of a narrow valley like banana that makes it difficult to be modeled. Branin-Hoo, Hartman and Griewank are multimodal functions. Ellipsoid is a separable function. G07 is a non-separable function so that the interrelation among the variables makes it more difficult to model than a separable function.

Table 4 – Formulation of the benchmark functions

No.	Range	*m*	Function	Formulation
1	$[-10,10]$	1	Sinc	$y(x)=\sin(x)/x$
2	$[0,1]$	2	Branin-Hoo	$y(x)=\frac{1}{51.95}\left[\left(\bar{x}_2-\frac{5.1\bar{x}_1^2}{4\pi^2}+\frac{5\bar{x}_1}{\pi}-6\right)^2+\left(10-\frac{10}{8\pi}\right)\cos(\bar{x}_1)-44.81\right]$ where $\bar{x}_1=15x_1-5,\bar{x}_2=15x_2$
3	$[0,1]$	4	Rosenbrock	$y(x)=\frac{1}{3.755\times10^5}\left[\sum_{i=1}^{3}\left(100\left(\bar{x}_{i+1}-\bar{x}_i^2\right)^2+\left(1-\bar{x}_i\right)^2\right)-3.827\times10^5\right]$ where $\bar{\boldsymbol{x}}=15\boldsymbol{x}-5$
4	$[0,1]$	6	Hartman	$y(x)=\frac{-1}{1.94\times10}\left[2.58+\sum_{i=1}^{4}C_i\exp\left(-\sum_{j=1}^{6}a_{ji}(x_j-p_{ji})^2\right)\right]$ where $\boldsymbol{C}=[1.0\ \ 1.2\ \ 3.0\ \ 3.2]$, $\boldsymbol{a}=\begin{bmatrix}10.00&0.05&3.00&17.00\\3.00&10.00&3.50&8.00\\17.00&17.00&1.70&0.05\\3.50&0.10&10.00&10.00\\1.70&8.00&17.00&0.10\\8.00&14.00&8.00&14.00\end{bmatrix}$ $\boldsymbol{p}=\begin{bmatrix}0.1312&0.2329&0.2348&0.4047\\0.1696&0.4135&0.1451&0.8828\\0.5569&0.8307&0.3522&0.8732\\0.0124&0.3736&0.2883&0.5743\\0.8283&0.1004&0.3047&0.1091\\0.5886&0.9991&0.6650&0.0381\end{bmatrix}$
5	$[-10,10]$	10	G07	$f(x)=(x_1^2+x_2^2+x_1x_2-14x_1-16x_2-(x_3-10)^2+4(x_4-3)^2+(x_5-3)^2+2(x_6-1)^2+5x_7^2+7(x_8-11)^2+2(x_9-10)^2+(x_{10}-7)^2+45)\times10^{-4}$
6	$[-5,5]$	20	Ellipsoid	$f(x)=10^{-4}\times\sum_{i=1}^{20}ix_i^2$
7	$[-600,600]$	60	Griewank	$f(x)=(\sum_{i=1}^{60}\frac{x_i^2}{4000}-\prod_{i=1}^{60}\cos(\frac{x_i}{\sqrt{i}})+1)\times10^{-4}$

The definition of the test cases is explained in Table 5. The observation noise is added artificially using Gaussian random variables based on the deterministic functions in Table 4. The noise variance is formulated proportional to the function standard-variance. The noise intensity varies from low- to high-level. Three optimization algorithms respectively combined with two GE-estimation methods for SVR hyperparameters tunning will be investigated. The parameter setting of the optimization algorithms is listed in Table 6. As the optimization algorithms have different convergence rate, for fair comparison and acceptable time cost consideration, the termination condition of the hyperparameter optimization is set as number of the GE-evaluation times no more than 50.

Table 5 – Definition of the test cases

Factor	Division	Explanation
Training dataset	$40\times m$	Number of the training samples
Test dataset	1000	Number of the test samples
Noise intensity	Low level	Noise variance is 5% of the function standard variance.
	Medium level	Noise variance is 20% of the function standard variance.
	High level	Noise variance is 50% of the function standard variance.

Table 6 – Parameter setting of the optimization algorithms

Algorithm	population size	generation	Crossover probability	Mutation probability	Initial step size	Number of GE times
GA	10	5	0.6	0.05		50
CMA-ES	10	5			2.4	50
BO						50

6.1 Comparison of hyperparameter optimization algorithms

6.1.1 One-dimensional test case

Based on the Sinc function, the Gaussian noise is added to the 40 samples generated by Latin hypercube sampling (LHS) and the noise variance is 20% of the function standard-variance. Figure 8 shows the SVR models in case of different hyperparameter optimization algorithms. The SVR models optimized by BO and CMA-ES matches better with the true function. To check the optimization results and figure out why BO and CMA-ES obtains the hyperparameters that enables better modeling, changes of MSE w.r.t. the hyperparameters are given at the location of the optimal hyperparameters of each optimization algorithm, as shown in Figure 9. It is found that, with the same computational cost (i.e. same number of GE-evaluation times), BO successfully finds the global optimums of all the three hyperparameters and achieves the minimal value of MSE (6.37e-4). CMA-ES and GA optimizations have not completely converged, so the optimal MSE of CMA-ES is slightly worse (7.74e-4) but at the same level and that of GA is the worst. If more iterations are allowed, CMA-ES and GA optimization may further reduce GE, however, computational cost must be higher.

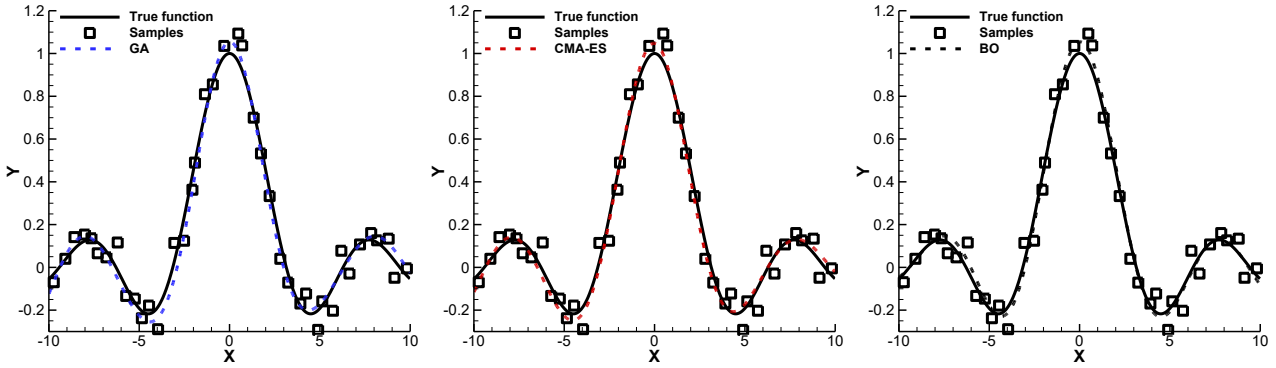
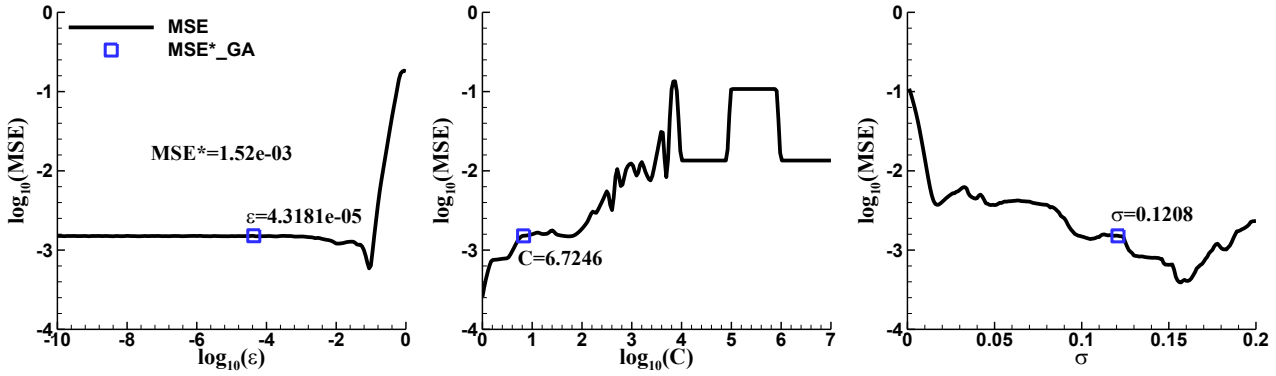
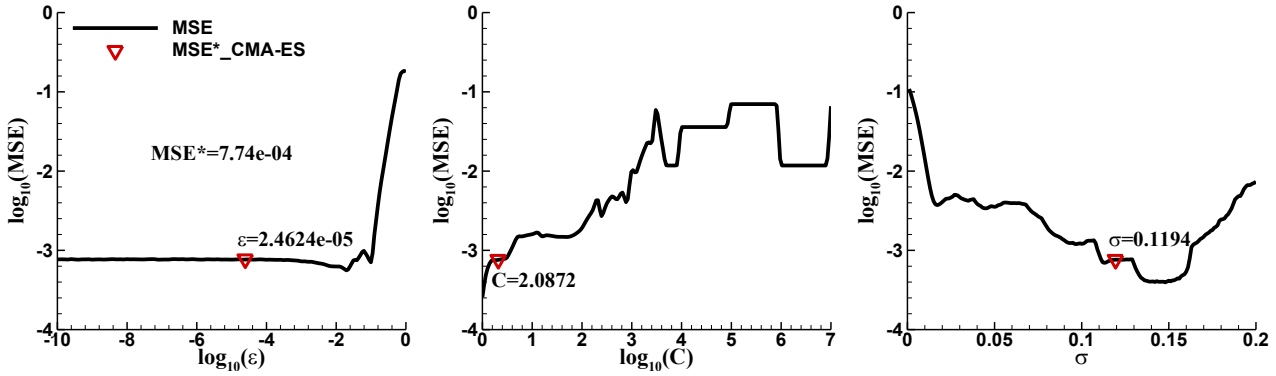


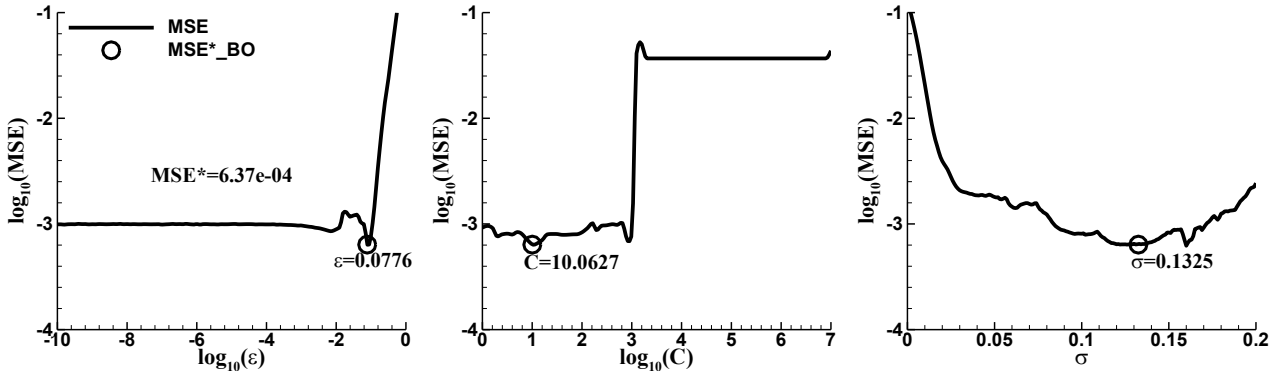
Figure 8 – SVR models of the Sinc function in case of different hyperparameter optimization algorithms



(a) GA



(b) CMA-ES



(c) BO

Figure 9 – Modeling error varies with the hyperparameters (Sinc function)

To further understand the convergence rate, for each optimization algorithm, the hyperparameter tuning is repeated 50 times based on the same data set, take the average at each convergence step and the convergence history is shown in Figure 10. It explains the results above again as the BO and CMA-ES converges fast and the GA converges slowly.

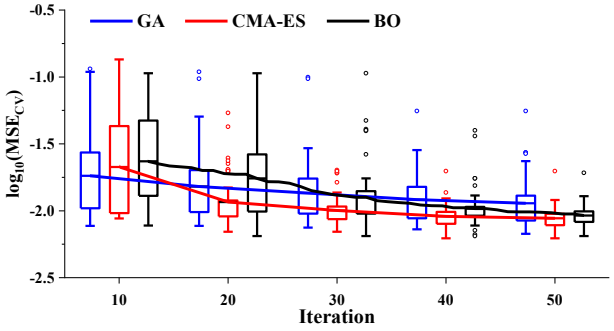
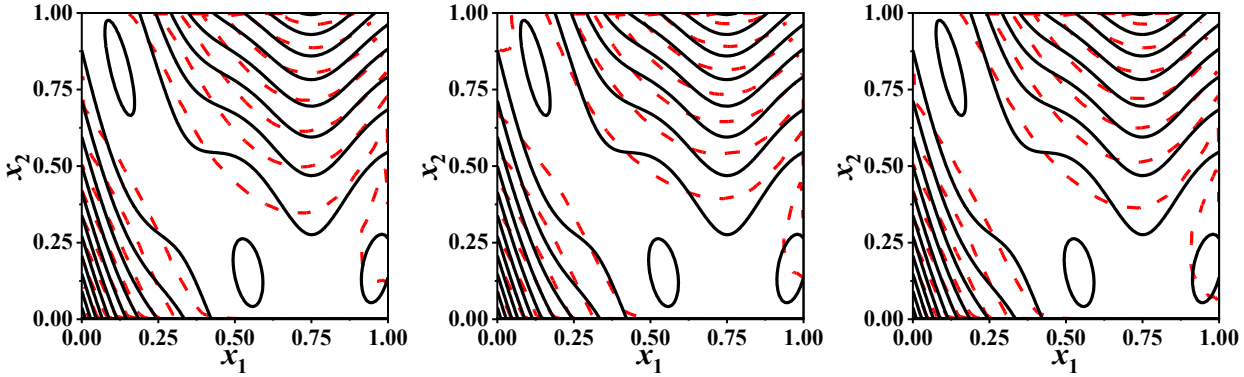


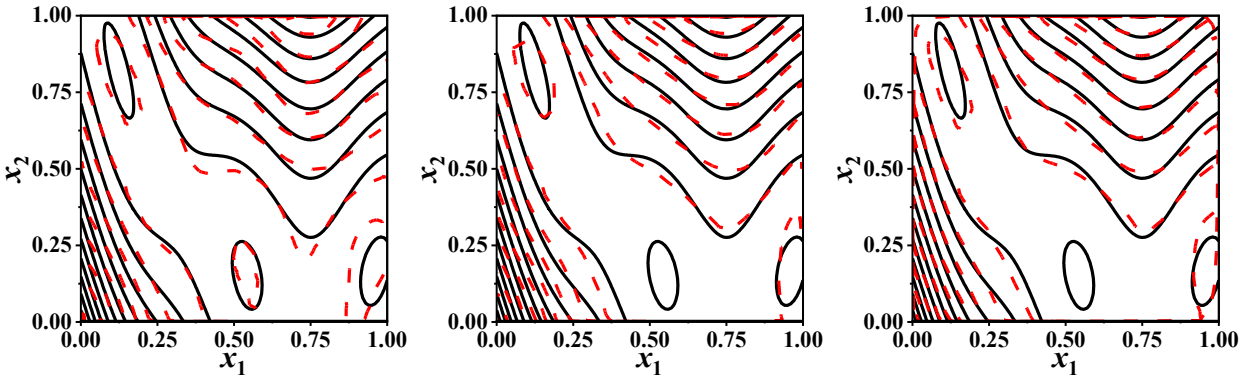
Figure 10 – Convergence history of hyperparameter optimization (Sinc function)

6.1.2 Two-dimensional test case

For further validation and comparison purpose, the surrogate model of the Branin-Hoo function is built based on the samples generated by LHS, in which the Gaussian noise variance is 20% of the function standard-variance. Figure 11(a) shows the SVR models based on 80 samples in case of different hyperparameter optimization algorithms, however, all the three surrogate models do not predict the function well although it seems that the optimizer has already found the global optimum. So we increase the sample number to 300 and regenerate the surrogate model. As shown in Figure 11(b) and Figure 12, when the samples are sufficient, accuracy of the surrogate models is highly improved. Figure 12 gives changes of MSE w.r.t. the hyperparameters at the location of the optimal hyperparameters of each optimization algorithm, in which the results of the optimizations are marked by the symbols. It is found that, in the limited GE evaluations, CMA-ES and BO successfully find the global optimum which GA still need more iterations to improve its results.

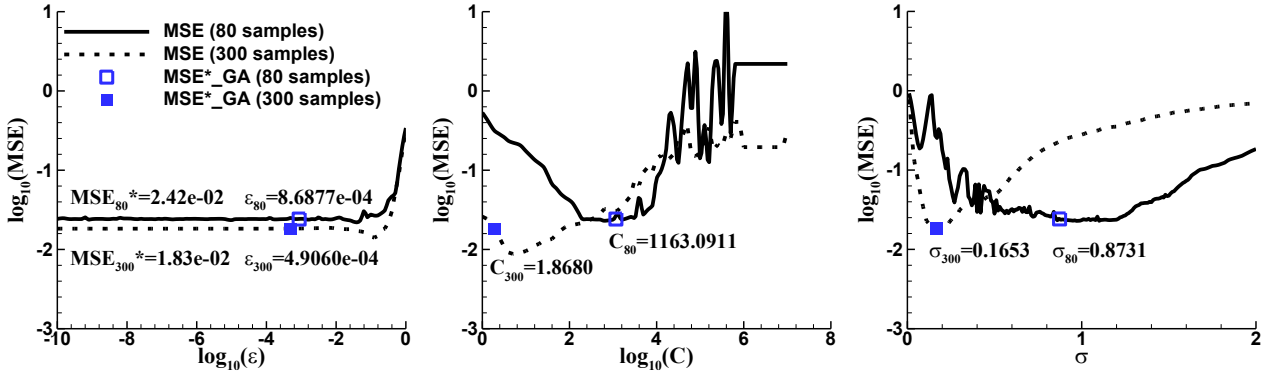


(a) 80 samples

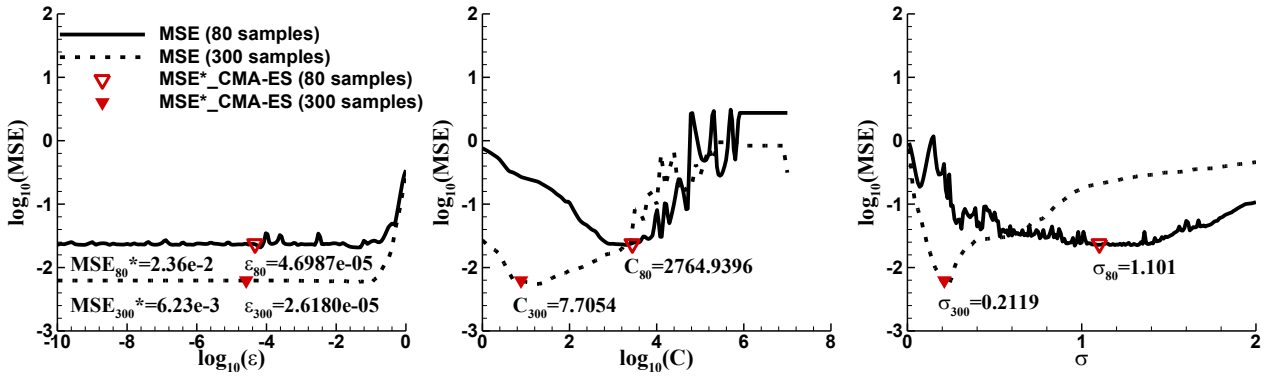


(b) 300 samples

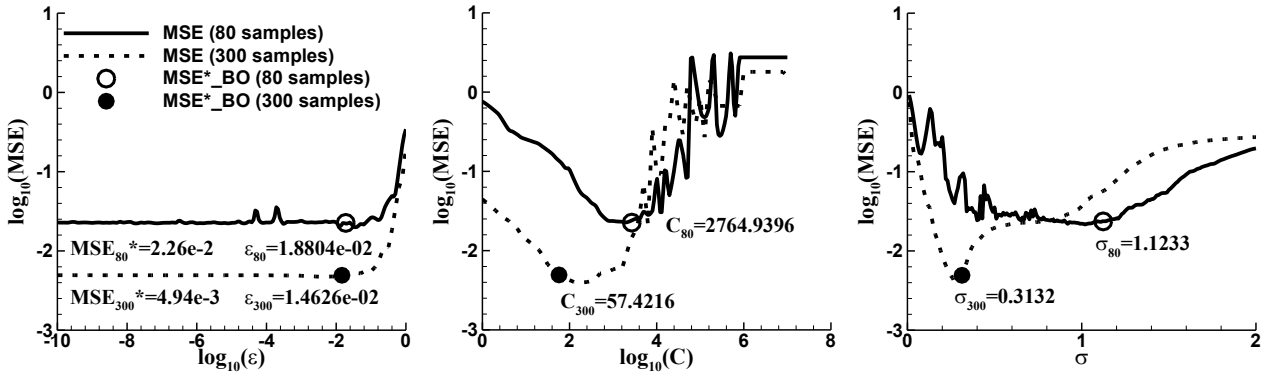
Figure 11 – SVR models of the Branin-Hoo function in case of different hyperparameter optimization algorithms (from left to right: GA, CMA-ES and BO)



(a) GA



(b) CMA-ES



(c) BO

Figure 12 – Modeling error varies with the hyperparameters (Branin-Hoo function)

The hyperparameter tuning is repeated 50 times based on the same data set, take the average at each convergence step and the convergence history is shown in Figure 13. Similar to the last test

case, CMA-ES and BO still converges faster and achieves higher accuracy, while GA converges slowly and is not so robust.

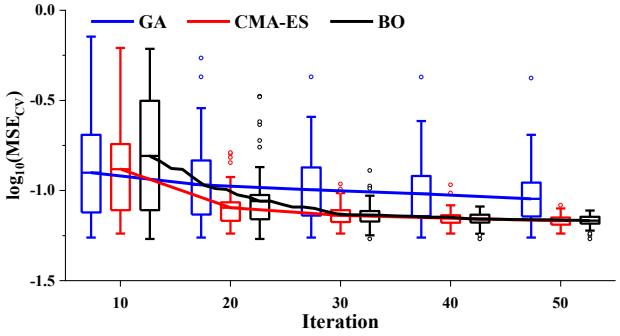


Figure 13 – Hyperparameter-tunning convergence history based on 80 samples (Branin-Hoo function)

6.1.3 Comparison based on all the test cases

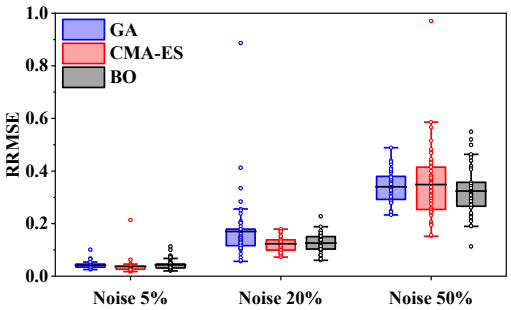
In this sub-section, all of the test cases defined in Table 4 will be run by the in-house SVR modeling codes with the hyperparameters tuned respectively by the GA, CMA-ES and BO algorithms. Two statistical measures, including the relative root mean square error (RRMSE) and the R-Square (R2), will be used for evaluation of the surrogate models.

$$RRMSE = \sqrt{\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 / N} \Big/ \sqrt{\sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2 / N} \tag{13}$$

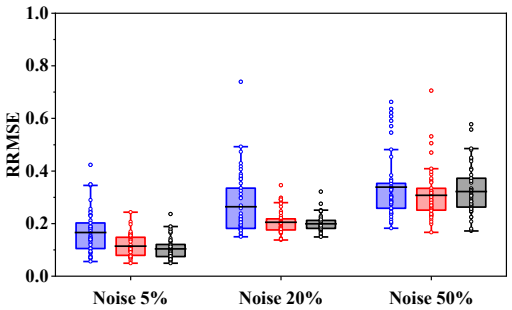
$$R^2 = 1 - \sum_{i=1}^{N}(y_i - \hat{y}_i)^2 / \sum_{i=1}^{N}(y_i - \bar{y})^2 \tag{14}$$

where N is number of the test samples, y_i and $\hat{y}_i$ are the true response and its corresponding predicted value, and $\bar{y}$ is mean of the true responses. The lower value of RRMSE is, more accurate the model is. The closer R2 is to 1 indicates better fitting of the model. Due to the heuristic searching mechanisms, every time we run the hyperparameter tunning by any of the optimization algorithms, the results would be different even based on the same data set. Therefore, for a trustable comparison, modeling of the 1st-6th function is repeated 50 times based on the regenerated samples by the LHS method, while that of the 7th function is repeated 20 times due to higher computational cost, then take the average. Additional 1000 points are generated by LHS for testing. Note that, all the numerical experiments are conducted in our in-house SVR code and run on a PC with Intel (R) Core (TM) i9-13900H @ 3.00 GHz and 128GB RAM.

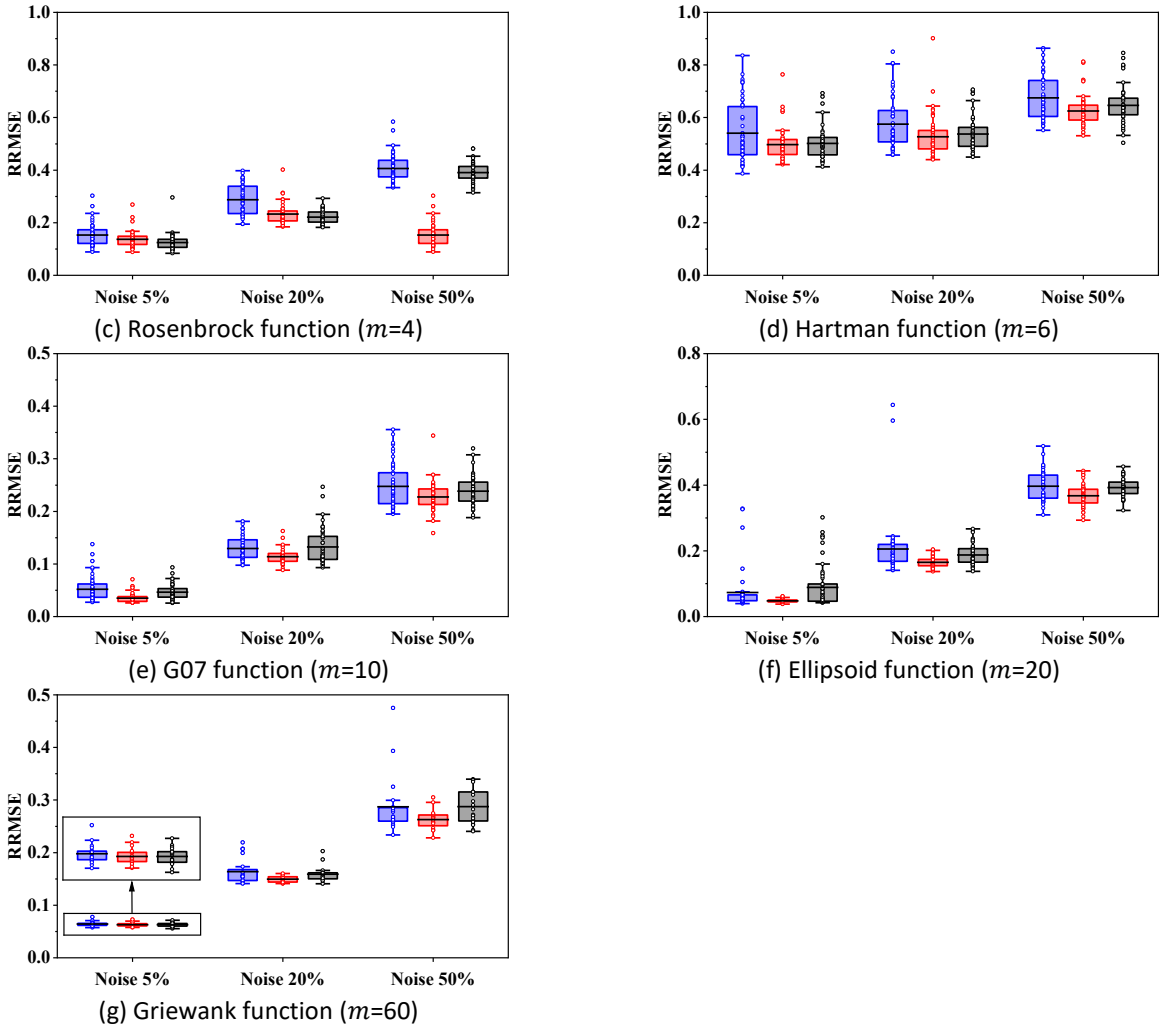
The box plots of the RRMSE results are shown in Figure 14, in which the median is the mean value of the results. The mean values of RRMSE and R2 are listed in Table 7, in which the highest-accuracy values are marked bold. It is found that, when the dimension is low ($m < 10$) and the numerical noise is not too strong, BO is slightly better than CMA-ES, and GA is apparently worse. With the dimension is higher ($m \geq 10$) or the noise is strong, CMA-ES enables the highest accuracy of the surrogate models, and the models obtained by BO and GA are almost at the same level. The results of GA are not so good no matter in terms of accuracy or robustness, as it converges slowly.



(a) Sinc function (m=1)



(b) Branin-Hoo function (m=2)



(c) Rosenbrock function (m=4)

(d) Hartman function (m=6)

(e) G07 function (m=10)

(f) Ellipsoid function (m=20)

(g) Griewank function (m=60)

Figure 14 – RRMSE box-plots of model accuracy by different hyperparameter optimization algorithms

Table 7 – GE of GA, CMA-ES and BO (noise intensity: 5%, 20% and 50%)

Functions	Dimension	Optimization Algorithms	5%		20%		50%	
			RRMSE	R^2	RRMSE	R^2	RRMSE	R^2
Sinc1	1	GA	0.0690	0.9896	0.1702	0.9562	0.3398	0.8809
		CMA-ES	0.0396	0.9973	**0.1187**	**0.9848**	0.3482	0.8610
		BO	**0.0377**	**0.9985**	0.1260	0.9830	**0.3236**	**0.8869**
Branin-Hoo	2	GA	0.1660	0.9664	0.2644	0.9169	0.3387	0.8703
		CMA-ES	0.1144	0.9852	0.2052	0.9558	**0.3074**	**0.8966**
		BO	**0.1041**	**0.9874**	**0.1994**	**0.9593**	0.3218	0.8882
Rosenbrock	4	GA	0.1530	0.9748	0.2877	0.9139	0.4065	0.8320
		CMA-ES	0.1366	0.9804	0.2330	0.9442	**0.3679**	**0.8629**
		BO	**0.1248**	**0.9834**	**0.2214**	**0.9503**	0.3904	0.8461
Hartman	6	GA	0.5407	0.6949	0.5747	0.6601	0.6752	0.5365
		CMA-ES	**0.4969**	**0.7490**	**0.5270**	**0.7158**	**0.6252**	**0.6054**
		BO	0.5015	0.7448	0.5373	0.7068	0.6461	0.5771
G07	10	GA	0.0519	0.9968	0.1294	0.9827	0.2474	0.9370
		CMA-ES	**0.0351**	**0.9987**	**0.1139**	**0.9868**	**0.2276**	**0.9474**
		BO	0.0463	0.9976	0.1324	0.9813	0.2385	0.9423
Ellipsoid	20	GA	0.0732	0.9908	0.2056	0.9498	0.3968	0.8404
		CMA-ES	**0.0476**	**0.9977**	**0.1651**	**0.9725**	**0.3674**	**0.8638**
		BO	0.0544	0.9968	0.1872	0.9640	0.3925	0.8450
Griewank	60	GA	0.0643	0.9958	0.1640	0.9725	0.2872	0.9145
		CMA-ES	**0.0630**	**0.9960**	**0.1497**	**0.9775**	**0.2627**	**0.9306**
		BO	0.0630	0.9960	0.1590	0.9745	0.2874	0.9162

For the efficiency comparison, the average time of each test case is collected as well and listed in Table 8 in which the shortest times are marked bold. The results indicates that: GA has a distinct superiority of efficiency in the low-dimensional cases ($m < 10$). But in the higher-dimensional cases ($m \geq 10$), the BO becomes the most efficient method.

Table 8 – Average modeling time (/s) of GA, CMA-ES and BO (noise intensity: 5%, 20% and 50%)

Function	Dimension	Optimization Algorithms	5%	20%	50%
Sinc	1	GA	**0.0235**	**0.0197**	**0.0155**
		CMA-ES	0.0343	0.0306	0.0286
		BO	0.4339	0.4282	0.4252
Branin-Hoo	2	GA	**0.0662**	**0.0595**	**0.0432**
		CMA-ES	0.1098	0.0739	0.0835
		BO	0.5274	0.4775	0.4473
Rosenbrock	4	GA	**0.3018**	**0.2122**	**0.1704**
		CMA-ES	0.3304	0.3058	0.2503
		BO	0.9029	0.7327	0.5835
Hartman	6	GA	**0.3457**	**0.3635**	**0.2748**
		CMA-ES	0.5068	0.4780	0.6288
		BO	0.6754	0.6642	0.6729
G07	10	GA	1.6248	1.4660	**1.0363**
		CMA-ES	2.7387	2.2779	1.7405
		BO	**1.3605**	**1.1101**	1.1455
Ellipsoid	20	GA	5.1942	4.7202	4.2082
		CMA-ES	7.0025	7.8510	6.3853
		BO	**3.7529**	**3.5470**	**2.8547**
Griewank	60	GA	71.0773	67.4167	51.0448
		CMA-ES	77.3284	82.9799	80.4204
		BO	**39.0394**	**38.7733**	**46.4693**

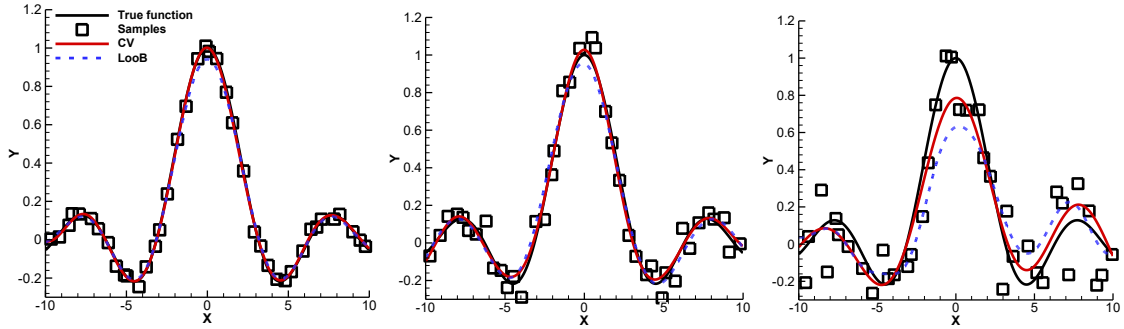
By the comprehensive comparison of three global optimization algorithms, it can be concluded that: 1) In terms of accuracy, CMA-ES behaves well for almost all the test cases, while BO is better in the low-dimensional (m <10) cases and is still comparable in the higher-dimensional cases when the noise is not too strong. GA is apparently worse than the other two algorithms in the low-dimensional cases as more iterations are needed but is at the same level with BO when the dimension is higher ($m \geq 10$). 2) In terms of efficiency, GA has distinct superiority in the low-dimensional cases (m <10), but as it needs more iterations to improve the accuracy which would offset its efficiency to some extent. In contrast, BO is time-consuming in the low-dimensional cases but becomes the most efficient in the higher-dimensional ($m \geq 10$) cases.

6.2 Comparison of GE-estimation methods

In aerodynamic design and optimization, numerical simulations are time-consuming. Therefore, it aims to find the appropriate methods that are both efficient and accurate. In order to establish a fast hyperparameter-tunning process to build a trustable surrogate model, the last section is focused on the hyperparameter optimization algorithms and this section will compare two GE-estimation methods, CV and LooB. Due to fast convergence and satisfied accuracy, BO will be used for the following work.

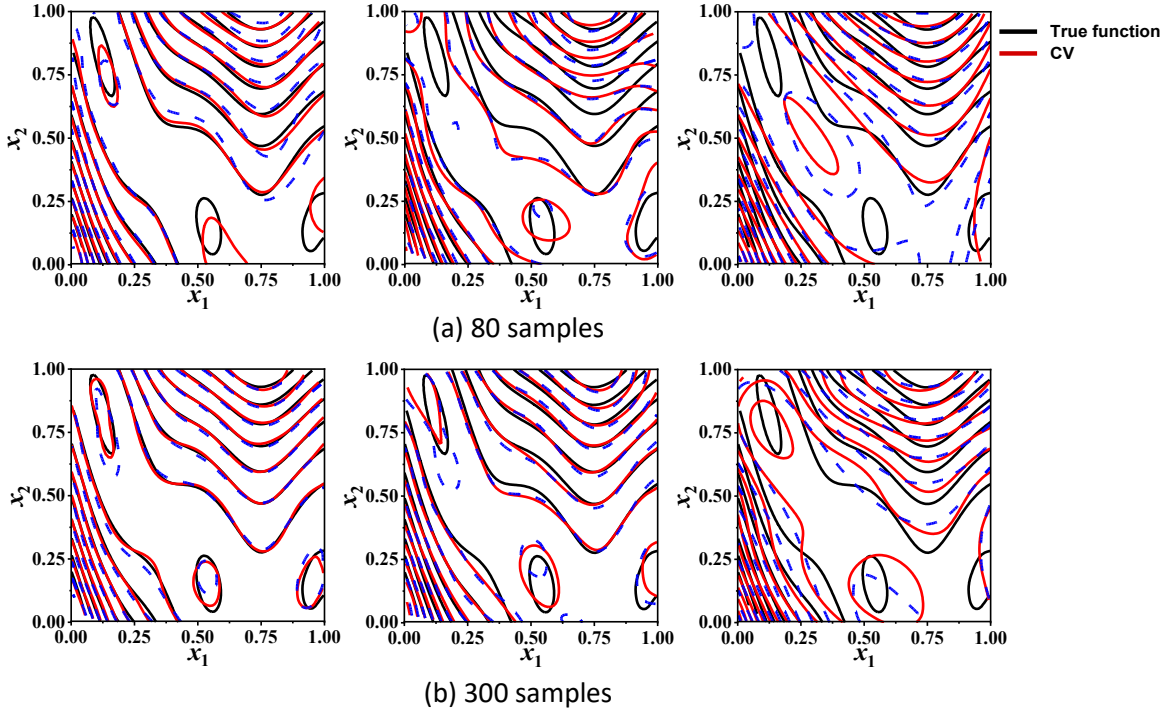
6.2.1 Modeling comparison

The Sinc and Branin-Hoo functions are predicted by SVR based on CV and LooB respectively, as shown in Figure 15 and Figure 16. Besides, the models are rebuilt based on 300 samples for the Branin-Hoo function. The results indicate that, 1) the prediction of CV fits better with the true function, 2) when the samples are sufficient and the noise intensity is low, model accuracy of LooB can be comparable, 3) when the noise is strong, the SVR based on either CV or LooB is not able to give a trustable prediction.



(a) Case 1: 5% noise intensity (b) Case 2: 20% noise intensity (c) Case 3: 50% noise intensity

Figure 15 – SVR models of the Sinc function in case of different GE-estimation methods



(a) 80 samples

(b) 300 samples

Figure 16 – SVR models of the Branin-Hoo function in case of different GE-estimation methods (noise intensity: 5%, 20% and 50% from left to right)

6.2.2 Comparison based on all the test cases

In principle, LooB sacrifices some accuracy to reduce modeling time. The average modeling time of each test case is listed in Table 9 verifies it. But can we win on both accuracy and efficiency? The further work is done on parallelization of the k-fold CV process, i.e. k times of SVR modeling for GE evaluation are run at the same time instead of in sequence. The results are listed in Table 9 too, labeled as CV (parallel). It is interesting to find that the parallel CV wins both accuracy and efficiency.

Table 9 – Average modeling time (/s) of CV and LooB (noise intensity: 5%, 20% and 50%)

Function	Dimension	Noise intensity	CV	CV (parallel)	LooB
Sinc	1	5%	0.7774	0.4339	**0.4200**
		20%	0.5907	0.4282	**0.4142**
		50%	0.5166	0.4252	**0.4054**
Branin-Hoo	2	5%	3.5167	0.5274	**0.5123**
		20%	2.6885	0.4775	**0.4769**
		50%	1.7758	**0.4473**	0.4502
Rosenbrock	4	5%	22.3195	0.9029	**0.9027**
		20%	13.5967	**0.7327**	0.8174
		50%	6.0914	**0.5835**	0.7189
Hartman	6	5%	7.0938	**0.6754**	1.1088
		20%	8.2261	**0.6642**	1.0942
		50%	8.7837	**0.6729**	1.0712
G07	10	5%	26.8963	**1.3605**	2.4495
		20%	23.1536	**1.1101**	2.3839
		50%	17.9062	**1.1455**	2.2352
Ellipsoid	20	5%	59.4847	**3.7529**	8.1366
		20%	50.4716	**3.5470**	8.2945
		50%	64.7712	**2.8547**	7.8739
Griewank	60	5%	318.4338	**39.0394**	99.0103
		20%	472.0227	**38.7733**	111.2166
		50%	500.1409	**46.4693**	353.1617

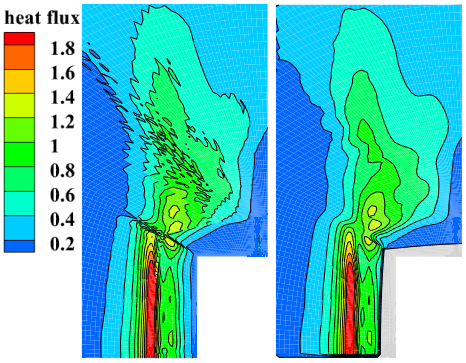
7. Applications to aerodynamic data modeling

Based on the results of numerical examples, BO performs well in terms of both efficiency and accuracy. Therefore, the BO algorithm will be used in the application examples, and its GE evaluation

number is increased to 100 to achieve better results.

7.1 CFD Aerothermal-data fitting

The aerothermal heat flux of the thermal protection system for a rocket warhead is obtained by CFD and the contour of the extremely high temperature region is plotted in Figure 17. As shown in the figure, the contour lines are very rough, which indicates apparent numerical noise. Then based on all the 5730 samples in this region, the SVR model is built to predict the aerothermal heat flux, as shown in Figure 17. It is obvious that the contour becomes smooth and its distribution fits the original data.



(a) original data (b) SVR (CV)

Figure 17 - Prediction of the extremely-high temperature region of the rocket warhead

7.2 Wind-tunnel experimental data fitting

The wind-tunnel experimental data is unavoidably accompanied by some physical error due to unexpected inaccuracy and randomness in the experiment equipment and environment as well as model quality. The experimental data of the NPU-MWA-250 wind-turbine airfoil[25] at Re=1×10^6 is shown in Figure 18 and the SVR model is built. It is found that there is apparent numerical noise in the data of $C_{\mathrm{L}}/C_{\mathrm{D}}$ when approaching the stall angle of attack. No matter with or without data noise, the good predictions are obtained by the SVR model.

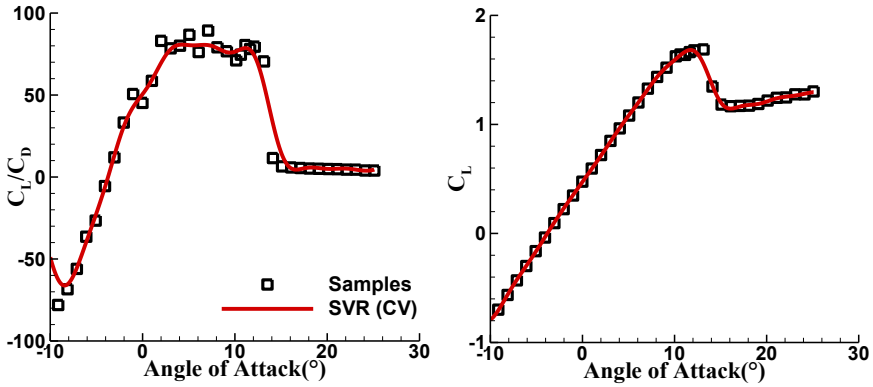


Figure 18 – Predictions of $C_{\mathrm{L}}/C_{\mathrm{D}}$ and C_{L} based on the wind-tunnel experimental data

8. Conclusion and Outlook

SVR is one of most popular regression modeling method in machine learning. Due to its good generalization ability and good adaptability to high-dimensional problem, we are always striving to introduce it into aerodynamic design and analysis as the data noise dramatically deteriorate the training efficiency and prediction accuracy. However, for SVR modeling, the hyperparameters have critical impact on model accuracy, and the hyperparameters adaption may be accompanied with high computational cost as well. To build a trustable SVR model in an efficient way, the hyperparameters adaption is investigated.

1) The hyperparameters design spaces are plotted and it is found that the generalization error curves of all the three hyperparameters are characteristic of multi-modal, large “flat” region and non-smoothness. So the global optimization algorithms are necessary for hyperparameter optimization.

2) The comparisons of three popular global optimization algorithms for hyperparameters-tuning are performed and some conclusions can be drawn as follows:

In terms of accuracy, CMA-ES behaves well for almost all the test cases, while BO is better in the low-dimensional (m <10) cases and is still comparable in the higher-dimensional cases when the noise is not too strong. GA is apparently worse than the other two algorithms in the low-dimensional cases as more iterations are needed but is at the same level with BO when the dimension is higher ($m \geq 10$).

In terms of efficiency, GA has distinct superiority in the low-dimensional cases (m <10), but as it needs more iterations to improve the accuracy which would offset its efficiency to some extent. In contrast, BO is time-consuming in the low-dimensional cases but becomes the most efficient in the higher-dimensional ($m \geq 10$) cases.

3) The comparison of two GE-estimation methods is performed as well. It is found that the parallel CV can not only enable higher mode accuracy but also has high efficiency even faster than LooB.

9. Acknowledgment

This research was sponsored by the Aeronautical Science Fund under Grant No.20230014053006, the National Key Research and Development Program of China under Grant No. 2023YFB3002800, the National Natural Science Foundation under Grant No.U20B2007. The authors are members of The Youth Innovation Team of Shaanxi Universities.

10. Contact Author Email Address

Keshi Zhang: zhangkeshi@nwpu.edu.cn

Hailong Qiao: qiaohailong@mail.nwpu.edu.cn

Penghui Wang: 1971891864@qq.com

Youquan Du: duyq@mail.nwpu.edu.cn

Zhonghua Han: hanzh@nwpu.edu.cn

11. Copyright Statement

References

[1] Vapnik V. *The nature of Statistical Learning Theory*. Springer, New York, 1995.

[2] Smola A J, Schölkopf B. A tutorial on support vector regression. *Statistics and computing*, Vol. 14, pp 199-222, 2004.

[3] Tharwat A and Hassanien A E. Chaotic antlion algorithm for parameter optimization of support vector machine, *Appl Intell*, No. 48, pp 670-686, 2018.

[4] Chen K Y. Forecasting systems reliability based on support vector regression with genetic algorithms. *Reliability Engineering & System Safety*, Vol. 92, No. 4, pp 423-432, 2007.

[5] Hong W C, Dong Y, Chen L Y, et al. SVR with hybrid chaotic genetic algorithms for tourism demand forecasting. *Applied Soft Computing*, Vol. 11, No. 2, pp 1881-1890, 2011.

[6] Lin S W, Ying K C, Chen S C, et al. Particle swarm optimization for parameter determination and feature selection of support vector machines. *Expert systems with applications*, Vol. 35, No. 4, pp 1817-1824, 2008.

[7] Lins I D, Moura M D C, Zio E, et al. A particle swarm - optimized support vector machine for reliability prediction. *Quality and Reliability Engineering International*, Vol. 28, No. 2, pp 141-158, 2012.

[8] Heidari A A, Mirjalili S, Faris H, et al. Harris hawks optimization: Algorithm and applications. *Future generation computer systems*, Vol. 97, pp 849-872, 2019.

[9] Saremi S, Mirjalili S and Lewis A. Grasshopper optimisation algorithm: theory and application. *Advances*

in engineering software, Vol. 105, pp 30-47, 2017.

[10]Mirjalili S, Mirjalili S M and Lewis A. Grey wolf optimizer. *Advances in engineering software*, Vol. 69, pp 46-61, 2014.

[11]Friedrichs F and Igel C. Evolutionary tuning of multiple SVM parameters. *Neurocomputing*, Vol. 64, pp 107-117, 2005.

[12]Alade I O, Abd Rahman M A and Saleh T A. Predicting the specific heat capacity of alumina/ethylene glycol nanofluids using support vector regression model optimized with Bayesian algorithm. *Solar Energy*, Vol. 183, pp 74-82, 2019.

[13]Smola A J, Schölkopf B and Müller KR. The Connection between Regularization Operators and Support Vector Kernels. *Neural Networks*, Vol. 11, No. 4, pp 637-649, 1998.

[14]James G, Witten D, Hastie T and Tibshirani R. *An introduction to statistical learning*, New York: Springer, 2013.

[15]Gunn S R. Support vector machines for classification and regression. *ISIS technical report*, Vol. 14, No. 1, pp 5-16, 1998.

[16]Wang J, Li C, Xu G, et al. Efficient structural reliability analysis based on adaptive Bayesian support vector regression. *Computer Methods in Applied Mechanics and Engineering*, Vol. 387, 2021.

[17]Alade I O, Rahman M A A and Hassan A. Modeling the viscosity of nanofluids using artificial neural network and Bayesian support vector regression. *Journal of Applied Physics*, Vol. 128, No. 8, 2020.

[18]Chen Y, Liao Y, Hu B, et al. A Novel Model for Electromagnetic Properties of Complex Microstructure Composites Based on Support Vector Regression. *IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization*, pp 1-4, 2020.

[19]Santos C E D S., Sampaio R C, Coelho L D S, Bestard G A and Llanos C H. Multi-objective adaptive differential evolution for SVM/SVR hyperparameters selection, *Pattern Recognition*, No. 110, 2021.

[20]Mantovani R G, Rossi A L. A meta-learning recommender system for hyperparameter tuning: Predicting when tuning improves SVM classifiers. *Information Sciences*, No. 501, pp 193-221, 2019.

[21]M. Wall. GAlib: A C++ Library of Genetic Algorithm Components. *MIT, http://lancet.mit.edu/ga/*, 1996.

[22]Vavak F and Fogany T. Comparison of Steady State and Generational Genertic Algorithms for use in Nonstationary Environments. *Pmc of the 1996 IEEE Conference on Evolutionary Computation*, Nagoya, Japan, pp 192-195, 1996.

[23]N. Hansen, The CMA Evolution Strategy: A Tutorial, 2016.

[24]Biedrzycki R. Handling bound constraints in CMA-ES: An experimental study. *Swarm and Evolutionary Computation*, No. 52, pp 100627, 2020.

[25]Han Z H and Görtz S. Hierarchical kriging model for variable-fidelity surrogate modeling. *AIAA journal*, Vol. 50, No. 9, pp 1885-1896, 2012.