Determination of Optimal SVM Parameters by Using GA/PSO

Yuan Ren
School of Jet Propulsion, Beijing University of Aeronautics and Astronautics, Beijing, China
Email: renyuan116@sjp.buaa.edu.cn

Guangchen Bai
School of Jet Propulsion, Beijing University of Aeronautics and Astronautics, Beijing, China
Email: dlxbgc@buaa.edu.cn

***Abstract*—The use of support vector machine (SVM) for function approximation has increased over the past few years. Unfortunately, the practical use of SVM is limited because the quality of SVM models heavily depends on a proper setting of SVM hyper-parameters and SVM kernel parameters. Therefore, it is necessary to develop an automated, reliable, and relatively fast approach to determine the values of these parameters that lead to the lowest generalization error. This paper presents two SVM parameter optimization approaches, i.e. GA-SVM and PSO-SVM. Both of them adopt a objective function which is based on the leave-one-out cross-validation, and the SVM parameters are optimized by using GA (genetic algorithm) and PSO (particle swarm optimization) respectively. From experiment results, it can be concluded that both approaches, especially PSO-SVM, can solve the problem of estimating the optimal SVM parameter settings at a reasonable computational cost. Further, we point out the importance of a proper population size for GA/PSO-SVM, and present the recommended population size for GA-SVM and PSO-SVM.**

***Index Terms*—support vector machine, cross validation, genetic algorithm, particle swarm optimization**

I. INTRODUCTION

Artificial neural networks (ANN) have been proved to be able to approximate nonlinear functions with arbitrary accuracy. Nevertheless structure and types of ANN are usually selected by trial and error [1], and the training of ANN is based on the empirical risk minimization (ERM) principle, which only aims at minimizing the training error [2]. Therefore, users may be confronted with difficulties in the application of ANN, and the generalization performance of ANN models obtained is often far from satisfactory [3].

Support vector machine (SVM), which is a statistical learning theory based machine learning method, is gaining popularity due to its many attractive features and promising generalization performance. Some prominent features of SVM are: (i) the ability to model non-linear relationships, (ii) SVM generalization performance does not depend on the dimensionality of the input space, (iii) the regression function is related to a quadratic programming problem whose solution is global and in general unique. Apart from these features, SVM also has a drawback that limits the use of SVM on academic and industrial platforms: there are free parameters (SVM hyper-parameters and SVM kernel parameters) that need to be defined by the user. Since the quality of SVM regression models depends on a proper setting of these parameters, the main issue for practitioners trying to apply SVM is how to set these parameter values (to ensure good generalization performance) for a given training data set. Whereas existing sources on SVM regression give some recommendations on appropriate setting of SVM parameters, there is no general consensus and many contradictory opinions. Reference [4] summarized the existing approaches to setting SVM parameters and presented a practical method for selecting the values of C (the regularization parameter) and ε (the radius of the insensitive tube). However, all these approaches (including the one proposed in [4]) are based on prior knowledge, user expertise, or experimental trial, and hence there is no guarantee that the parameter values obtained are truly optimal. On the other hand, the problem of optimal parameter selection is further complicated by the fact that the SVM generalization performance depends on all of these parameters (both hyper-parameters and kernel parameters) together. This means that the interaction of SVM parameters has to be considered, and that a separate optimization of each parameter is not sufficient enough to find the optimal regression model [3, 5]. Due to all the reasons mentioned above, in the practical application of SVM regression, usually a time-consuming grid search method is invoked to estimate the optimal SVM parameter settings [6].

When applying grid search method, one might need to increase the parameter range and / or decrease the step size to increase the accuracy of the optimal solution. However, this will result in a cumbersome time-consuming search process. Because the SVM parameter selection can be regarded as a constrained nonlinear

Manuscript received October 10, 2009; revised December 5, 2009; accepted December 25, 2009.
Foundation item: National high technology research and development program under the project No. 2006AA04Z405.

doi:10.4304/jcp.5.8.1160-1168

optimization problem, some scholars choose another way to solve this problem: X. F. Yuan et al. employed mutative scale chaos optimization algorithm to search for the optimal SVM parameter values [3]; Z. Y. Luo et al. proposed a novel SVM parameter tuning approach based on quantum-inspired evolutionary algorithm [7].

In this work, two methods, i.e. GA-SVM and PSO-SVM, were developed as relatively fast alternatives for the grid search approach, and they are based on genetic algorithm (GA) and particle swarm optimization (PSO) respectively. The motivation for selecting GA and PSO is: (i) the error surface produced by cross-validation is full of sharp edges, and this kind of landscape is not suited to gradient-based search; (ii) neither GA nor PSO requires that the objective function should be smooth, and both of them can efficiently locate the global optima even when the objective function is discontinuous; (iii) both GA and PSO are the most representative methods among the present intelligent optimization techniques, and it would be interesting to make an experimental comparison of their performance in the context of SVM parameter optimization. Further, we investigated the effect of population size on the quality of solutions and algorithm convergence speed.

Section II gives a brief introduction to SVM algorithm and discusses the crucial effects of hyper-parameters and kernel parameters. Section III describes the proposed approaches for estimating the optimal SVM parameter settings. In Section IV, detailed experiments are carried out to evaluate grid search, gradient-based search, and our GA/PSO-SVM method. Finally, the conclusions are presented in Section V.

II. SVM Algorithm and SVM Parameters

Let the training data set be represented by $\{(\boldsymbol{x}_i, y_i),\ i=1,\ldots,N\}$, where $\boldsymbol{x}_i \in R^D$ is an input vector, $y_i \in R$ is its corresponding desired output. The input vector is first mapped into a high dimensional feature space using a nonlinear mapping ϕ, and then a linear model can be constructed in the feature space

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

where $\boldsymbol{\omega}$ is a m-dimensional coefficient vector and b is a bias term. The ε-insensitive loss function is usually adopted for minimizing the empirical risk on the training data, and it is defined as

$$L_\varepsilon(y, f(\boldsymbol{x})) = \max(0, |y - f(\boldsymbol{x})| - \varepsilon) \tag{2}$$

where ε is a positive hyper-parameter that will make the loss function equal zero when $|y - f(\boldsymbol{x})|$ is smaller than it. Then the SVM empirical risk can be obtained as follows:

$$R_{\mathrm{emp}}(\boldsymbol{\omega}) = \frac{1}{N}\sum_{i=1}^{N} L_\varepsilon(y_i, f(\boldsymbol{x}_i)). \tag{3}$$

SVM performs linear regression in the high-dimensional feature space using ε-insensitive loss function and, at the same time, tries to reduce model complexity by minimizing $\|\boldsymbol{\omega}\|^2$. After introducing slack variables to measure the deviation of training samples outside ε-insensitive zone, SVM regression can be formulated as minimization of the following functional:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ &\text{s.t.} \quad f(\boldsymbol{x}_i) - y_i \le \varepsilon + \xi_i,\ y_i - f(\boldsymbol{x}_i) \le \varepsilon + \xi_i^* \quad \xi_i, \xi_i^* \ge 0 \end{aligned} \tag{4}$$

where C is a positive hyper-parameter that is usually called regularization parameter. By using the Lagrange multipliers, this optimization formulation can be transformed into the following dual problem:

$$\begin{aligned} \text{maximize} \ & \sum_{i=1}^{N} y_i(\alpha_i^* - \alpha_i) - \varepsilon\sum_{i=1}^{N}(\alpha_i^* + \alpha_i) - \\ & \frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j)K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{s.t.} \ & \sum_{i=1}^{N}(\alpha_i^* - \alpha_i) = 0,\ 0 \le \alpha_i^*, \alpha_i \le C \end{aligned} \tag{5}$$

where α_i^* and α_i are Lagrange multipliers, and $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is the kernel function that satisfies Mercer's conditions, which is equivalent to the dot product in the feature space:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi^{\mathrm{T}}(\boldsymbol{x}_i)\phi(\boldsymbol{x}_j). \tag{6}$$

Several kernel functions have been proposed in literature, in this work the focus is put on the widely used radial basis function (RBF), which is defined in (7):

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / 2\sigma^2) \tag{7}$$

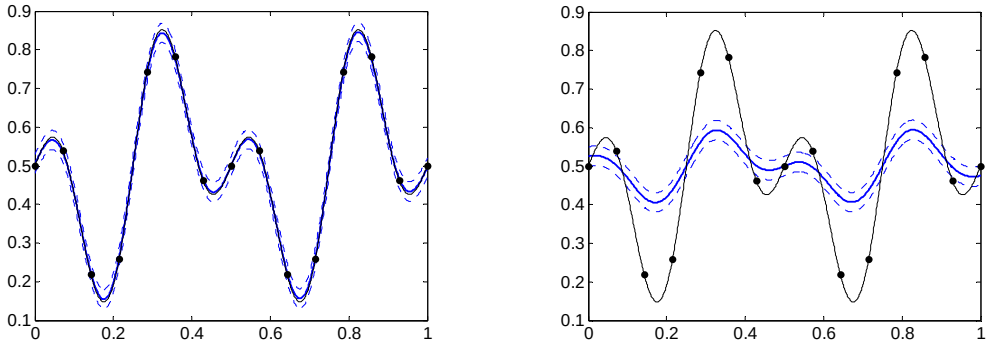
where σ is the kernel parameter that is always greater than zero. The solution of (5) can be written as

$$f(\boldsymbol{x}) = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)K(\boldsymbol{x}_i, \boldsymbol{x}) + b. \tag{8}$$

The sample points that appear with non-zero coefficients in (8) are called support vectors (SVs).

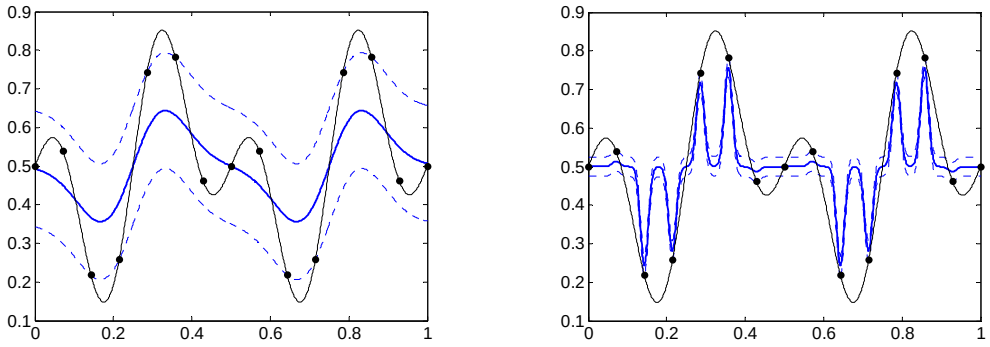
The SVM generalization performance strongly depends on the proper setting of C, ε and σ. These parameters affect the quality of SVM models in different respects, and Fig. 1 illustrates the influence of the three parameters on SVM regression respectively. Fig. 1(a) shows the true function to be approximated (fine curve), training samples (solid points), prediction results of SVM with properly chosen parameters (heavy curve), and the boundaries of ε-insensitive zone (dotted curves). It can be seen from Fig. 1(a) that SVM is able to accurately approximate nonlinear functions with a small quantity of training data as long as hyper-parameters and kernel parameter are properly set.

The hyper-parameter C determines the trade-off between the model complexity and the degree to which deviations larger than ε are tolerated. A poor choice of C will lead to an imbalance between model complexity minimization (MCM) and empirical risk minimization (ERM), and Fig. 1(b) illustrates the situation in which MCM has achieved overwhelming superiority over ERM.



(a) SVM with properly chosen parameters

(b) SVM with a poor choice of C



(c) SVM with a poor choice of ε

(d) SVM with a poor choice of σ

Figure 1. Influence of C, ε, and σ on SVM regression

The hyper-parameter ε controls the width of the ε-insensitive zone, and its value affects the number of SVs used to construct the regression function. If ε is set too large, the insensitive zone will have ample margin to include data points; this would result in too few SVs selected and lead to unacceptable 'flat' regression estimates (see Fig. 1(c)). The parameter σ represents the width of RBF kernel. If σ is set too small, the SVM will tend to overfit the training data (see Fig. 1(d)). On the other hand, a too large σ would make SVM not flexible enough for complex function approximation. In this case the regression result is similar to that of Fig. 1(b).

III. GA-SVM AND PSO-SVM

Since the SVM generalization performance heavily depends on the right setting of C, ε, and σ, these three parameters need to be set properly by the user. According to the experience from numerical experiments [3, 5], C, ε and σ exhibit a (strong) interaction. As a consequence, they should be optimized simultaneously, rather than separately.

A. Objective Function for C-ε-σ Optimization

In order to obtain an objective function which can reflect SVM generalization performance without the help of test data, the cross-validation technique is adopted in this work. The procedure of cross-validation divides training data D at random into S distinct segments $\{G_s, s = 1, \ldots, S\}$, and uses $(S-1)$ segments for training, and uses the remaining one for test. This process is repeated S times by changing the remaining segment, and the generalization performance is evaluated by using the following *MSE* (mean squared error) over all test results.

$$MSE_{\mathrm{CV}} = \frac{1}{N}\sum_{s=1}^{S}\sum_{i\in G_s}(y_i - f(\boldsymbol{x}_i|\boldsymbol{\theta}_s))^2 \qquad (9)$$

Here G_s denotes the s-th segment for the test, and $\boldsymbol{\theta}_s$ denotes the solution vector obtained by using $D-G_s$ for training. The solution vector can be acquired through solving a quadratic programming problem, and it consists of Lagrange multipliers and the estimate of bias term:

$$\boldsymbol{\theta} = [\alpha_1^*, \alpha_1, \alpha_2^*, \alpha_2, \ldots, \alpha_N^*, \alpha_N, \hat{b}]. \qquad (10)$$

When only a small sample amount is available, the number of segments S is usually set to value N. The extreme case of $S=N$ is known as the leave-one-out (LOO) method, and according to this method equation (9) can be rearranged into

$$MSE_{\mathrm{LOO}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(\boldsymbol{x}_i|\boldsymbol{\theta}_i))^2. \qquad (11)$$

In this equation $\boldsymbol{\theta}_i$ denotes the solution vector obtained by using $D-\{(\boldsymbol{x}_i, y_i)\}$ for training. Because the solution to a quadratic programming problem is global and unique, it can be concluded that for a given training data set there is a unique solution vector $\boldsymbol{\theta}$ assigned to each combination of (C, ε, σ). Hence equation (11) can be rewritten as

$$MSE_{\mathrm{LOO}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(\boldsymbol{x}_i|\boldsymbol{\theta}_i(C,\varepsilon,\sigma)))^2 = F(C,\varepsilon,\sigma) \qquad (12)$$

That is, for a given training data set, SVM MSE_{LOO} can be regarded as a function whose arguments are C, ε, and σ. In this work, we adopt MSE_{LOO} as the objective function for C-ε-σ optimization. In most cases, the gradient-based optimization algorithms such as quasi-Newton method and conjugate gradient method are highly efficient, and they might converge to the optimum within a few iterations. Unfortunately, SVM MSE_{LOO} surface is usually far from smooth and full of sharp edges [6], and a landscape like this is not suited to gradient-based search. Intelligent optimization methods, such as GA and PSO, do not require that the objective function should be smooth, and they can efficiently locate the global optimum even when the objective function is discontinuous. Therefore in this work we use GA and PSO to estimate the optimal SVM parameter settings.

B. Genetic Algorithm and Particle Swarm Optimization

The core of GA lies on the evolution from the current generation to the next, and this process consists of four steps, which are fitness scaling, selection, crossover and mutation. Fitness scaling converts the raw fitness scores returned by the objective function to values in a range that is suitable for selection. In this work, the rank fitness scaling method, which scales the raw scores based on the rank of each individual instead of its score, is adopted, so that the effect of the spread of the raw scores can be removed.

The selection uses the scaled fitness values to select the parents of the next generation, and it usually assigns a higher probability of selection to individuals with higher scaled values. In this work, we adopt the most commonly used selection method, i.e. the roulette wheel scheme.

Crossover enables the algorithm to extract the best genes from different individuals and recombine them into potentially superior children. In this work, we adopt the heuristic crossover method, which returns a child that lies on the line containing the two parents, a small distance away from the parent with the better fitness value in the direction away from the parent with the worse fitness value. The child returned can be expressed as

$$child = parent2 + R_{\mathrm{cros}} \times (parent1 - parent2), \quad (13)$$

where R_{cros} is the ratio indicating how far the child is from the better parent, and $parent1$ denotes the parent having the better fitness value. In addition, the parameter F_{cros}, which represents the fraction of individuals in the next generation (other than elite children) that are created by crossover, also has a significant effect on GA performance.

Besides crossover children, the genetic algorithm creates mutation children by applying random changes to individual parents in the current generation. Mutation adds to the diversity of a population and thereby increases the likelihood that the algorithm will generate individuals with better fitness values. In this research we choose the uniform mutation method, and it is a two-step process: first, the algorithm selects a fraction of the vector entries of an individual for mutation, where each entry has a probability P_{mut} of being mutated; in the second step, the algorithm replaces each selected entry by a random number selected uniformly from the range for that entry.

PSO is motivated by social behavior of organisms. As in GA, a population of individuals exists, and each individual is named as a "particle" which represents a potential solution. Each particle is treated as a point in a D-dimensional space. The i-th particle is represented as $\boldsymbol{z}_i = (z_{i1}, z_{i2}, \ldots, z_{id}, \ldots, z_{iD})$. The best previous position of any particle is recorded and represented as $\boldsymbol{p}_i = (p_{i1}, p_{i2}, \ldots, p_{id}, \ldots, p_{iD})$. The index of the best particle among all the particles in the population is represented by the symbol g. The rate of the position change (velocity) for the i-th particle is represented as $\boldsymbol{v}_i = (v_{i1}, v_{i2}, \ldots, v_{id}, \ldots, v_{iD})$. The updated velocity and position of the i-th particle at the k-th iteration are

$$v_{id}^{k+1} = w \cdot v_{id}^{k} + c_1 \cdot r_1 \cdot (p_{id} - z_{id}^{k}) + c_2 \cdot r_2 \cdot (p_{gd} - z_{id}^{k}) \quad (14)$$

$$z_{id}^{k+1} = z_{id}^{k} + v_{id}^{k+1} \quad (15)$$

where c_1 and c_2 are two positive constants, r_1 and r_2 are two random numbers in the range $[0, 1]$, and w is the inertia weight. The second part of (14) is the "cognition" part, which represents the private thinking of the particle itself. The third part is the "social" part, which represents the collaboration among the particles. In addition, the implementation of PSO also requires placing a limit on the particle velocity, and the limit, i.e. the maximum allowed velocity $\boldsymbol{v}_{\max}$, determines the searching granularity of space. The inertia weight w plays the role of balancing the global search and local search, and it can be a positive constant or even a positive linear or nonlinear function of time.

C. Proposed Approaches

The proposed approaches for SVM parameter optimization, i.e. GA-SVM and PSO-SVM, are illustrated in Fig. 2, and both of them were developed as relatively fast alternatives for the time-consuming grid search approach. MSE_{LOO} is adopted as the objective function for C-ε-σ optimization, and these three parameters are optimized by using GA and PSO, respectively.

IV. NUMERICAL EXPERIMENTS

A. Test Function Used in this Study

To evaluate our proposed methods, we used a 2-D test function which is illustrated in Fig. 3. This test function was selected from [8], and its mathematical expression is shown in (16). A 7^2 full factorial design was adopted, and exact data values of this function were obtained at the 49 sampling points. The training data set is consisted of the 49 samples.

$$y = 3639.35(1 - x_1^2)(1 - x_2^2)(4 + x_1)[0.99^{x_1} - 0.99^{x_2}]^2 \quad (16)$$

In the initial population, the range for each design variable is set as follows:

$$C \in [1, 10^8],\ \varepsilon \in (0, 1],\ \sigma \in [0.01, 10].$$

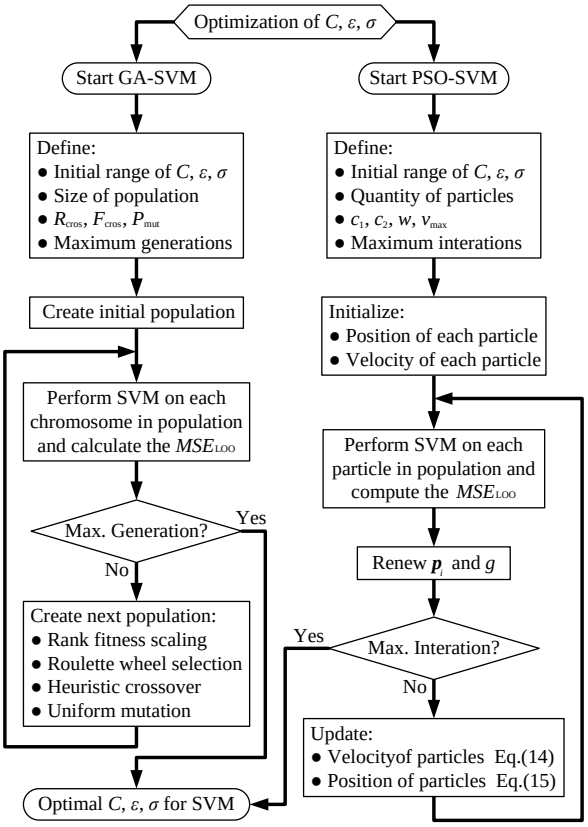


Figure 2. GA/PSO-SVM procedure

For convenience, we adopt three new design variables, and they are $V_1 = \log_{10} C$, $V_2 = \log_{10} \varepsilon$ and $V_3 = \log_{10} \sigma$. The initial search ranges for the three variables are $V_1 \in [0, 8]$, $V_2 \in [-12, 0]$ and $V_3 \in [-2, 1]$. As a consequence, for a given training data set, equation (12) can also be rewritten as

$$MSE_{\mathrm{LOO}} = F(C, \varepsilon, \sigma) = F'(V_1, V_2, V_3) \ . \qquad (17)$$

B. Comparison between MSE_{LOO} and ARE

We adopt MSE_{LOO} as the objective function for C-ε-σ optimization. It is necessary to verify the inference: an SVM model with the lowest MSE_{LOO} has or tends to have the best generalization performance. The average relative error (*ARE*) is adopted to evaluate the generalization performance of regression models, and it is defined as:

$$ARE = \frac{\sum_{i=1}^{N'} \left| (exact)_i - (predicted)_i \right|}{\sum_{i=1}^{N'} \left| (exact)_i \right|} \qquad (18)$$

where the "exact" values in the summation come from the evaluation of the exact function (equation (16)), and the "predicted" values come from the regression model approximation at N' test points. For the target function illustrated in Fig. 3 N' is set to 441 and selected from equally spaced points on a 21×21 square grid. Because the quantity of test data is sufficiently large, there is no doubt that *ARE* will accurately reflect the generalization performance of a regression model.

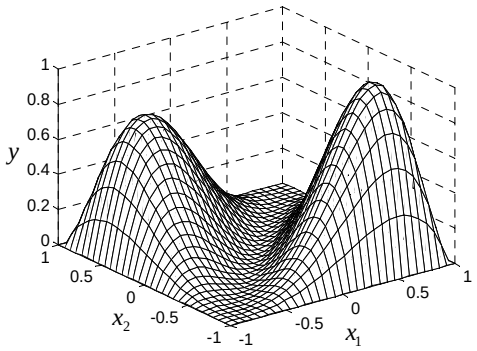


Figure 3. Target function to be approximated

According to the learning mechanism of SVM, it can be concluded that for a given training set and a given test set, the *ARE* of SVM model can be regarded as a function whose arguments are C, ε and σ (or V_1, V_2, and V_3). That is, we can obtain the following equation:

$$ARE = G(C, \varepsilon, \sigma) = G'(V_1, V_2, V_3) \ . \qquad (19)$$

After gaining (17) and (19), the above-mentioned inference about MSE_{LOO} and generalization performance can be verified by comparing the value of SVM MSE_{LOO} with that of SVM *ARE*. If the inference is correct, the (C, ε, σ) / (V_1, V_2, V_3) point with the lowest MSE_{LOO} will have or tend to have the minimum *ARE*.

With the help of Monte Carlo simulation, 30 points were randomly selected in the 3-D space defined by the initial search ranges for V_1, V_2 and V_3, and these points are illustrated in Fig. 4. On the basis of the training set consisting of 49 samples, the SVM MSE_{LOO} was evaluated at each of the 30 (V_1, V_2, V_3) points. In addition, the SVM *ARE* was also calculated at these points based on the test set containing 441 samples. The obtained results are illustrated in Fig. 5.

By comparing the values of SVM MSE_{LOO} with those of SVM *ARE*, it can be seen that: (i) the MSE_{LOO} curve has roughly the same variation tendency as the *ARE* curve; (ii) the three (C, ε, σ) points having the lowest MSE_{LOO}, No. 9, No. 14 and No. 25, also have the lowest *ARE*. The details about the three points are listed in Table I, where I_{MSE} is the index of a (C, ε, σ) point after sorting MSE_{LOO} values in ascending order, and I_{ARE} is the index of a (C, ε, σ) point after sorting *ARE* values in ascending order.

According to Fig. 5 and Table I, it can be concluded that the combinations of (C, ε, σ) having the lowest MSE_{LOO} can make SVM models have the best generalization performance (or lowest prediction error on unseen data). Though *ARE* (as well as some other similar indices) can effectively reflect the generalization performance of regression models, its application is limited by its demand for the test data, which should be different from the training data. On most occasions, the quantity of data samples available for analysis is very limited, so it is usually impractical to prepare an adequate number of test data. In these cases, it is more suitable to adopt MSE_{LOO}, which is totally based on the training data, as the objective function for C-ε-σ optimization.

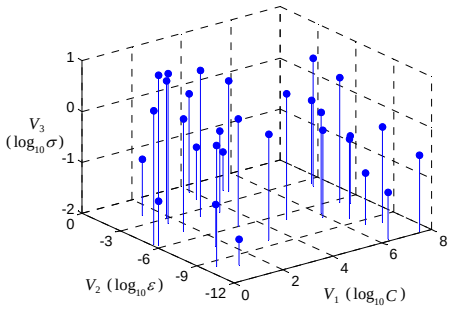


Figure 4. Thirty points selected using Monte Carlo simulation

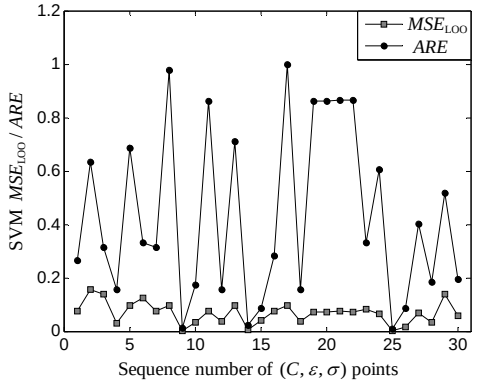


Figure 5. Comparison between MSE_{LOO} and ARE

TABLE I.
DETAILS ABOUT THE THREE MENTIONED POINTS

No.	MSE_{LOO}	ARE	I_{MSE}	I_{ARE}
9	0.001452	0.011780	1	2
25	0.001636	0.010038	2	1
14	0.006530	0.021789	3	3

C. Some other Methods of Minimizing MSE_{LOO}

Before applying our proposed methods, we will try two other methods first, and they are grid search and gradient-based search. The grid search method is sometimes referred to as the exhaustive method, and the required number of MSE_{LOO} evaluations depends on the parameter range in combination with the chosen interval size (resolution in the parameter space). In this work, each range is divided into 20 equal intervals, thus the required number of MSE_{LOO} evaluations is 9261. The step sizes for the three design variables are $\Delta V_1 = 0.4$, $\Delta V_2 = 0.6$, $\Delta V_3 = 0.15$. The grid search method is used to find a minimum of MSE_{LOO} when establishing the SVM model of the target function illustrated in Fig. 3. All the data in the training set are the same as before, and the results returned are: $V_1 = 6.0$, $V_2 = -10.2$, $V_3 = -0.05$, $MSE_{LOO} = 6.3919\text{E}{-4}$.

The gradient-based search method has numerous variants. In this work we adopt the quasi-Newton method with a line search procedure. The adopted quasi-Newton method utilizes the BFGS formula, which is thought to be the most effective one among a large number of Hessian updating algorithms. As for the line search procedure which is used to determine how far to move in the search direction, there are two alternative strategies: the cubic polynomial method and the mixed quadratic and cubic polynomial method. The former generally requires fewer objective-function evaluations but more gradient evaluations. Because it is impossible to express SVM MSE_{LOO} as an explicit function of C, ε and σ, the gradient information can only be obtained by using finite difference. In this case, the cubic polynomial method would require more MSE_{LOO} evaluations than the mixed quadratic and cubic polynomial method. Therefore we choose the latter.

The result returned by the gradient-based method largely depends on the initial point at which the optimization starts, and a number of repeated experiments are necessary to obtain a reliable conclusion. Hence the quasi-Newton procedure was repeated 100 times, and the 100 initial points were randomly selected. The reasons why the algorithm terminates can be categorized into the following four types:

Type 1: Magnitude of gradient is smaller than the specified tolerance (1.0E−6), and this means that the algorithm has terminated normally at a local optimum.

Type 2: Numerical problems are encountered when evaluating finite difference gradients, and the algorithm fails from the beginning. Because in this case the optimization terminates at the initial point, we do not think that the algorithm has returned any valid results.

Type 3: The algorithm terminates after some iterations because line search cannot find an acceptable point along the final search direction.

Type 4: The number of MSE_{LOO} evaluations reaches the maximum value allowed (it is set to 200 in this research) before the algorithm converges to a solution.

In the third case and the fourth case above, although the algorithm fails to converge normally, it can still return a (C, ε, σ) point having a lower MSE_{LOO} than the initial point. Statistical analysis was performed on the results of the 100 independent experiments, and two pie charts were obtained (see Fig. 6).

From Fig. 6, it can be seen that: (i) only 13 out of 100 trials converged to a local optimum; (ii) only 9 out of 100 trials yielded a solution at which MSE_{LOO} is smaller than 1.0E−3. The above facts indicate that the quasi-Newton method is not suitable for minimizing SVM MSE_{LOO}. We have also tried the steepest descent method and the conjugate gradient method, and results similar to Fig. 6 were obtained. Therefore it can be concluded that the gradient-based search is not a good method for minimizing SVM MSE_{LOO}.

D. Application of the Proposed Methods

The parameter settings for GA-SVM and PSO-SVM are both shown in Table II. To investigate the effect of various population sizes, three different values, i.e. 10, 20, and 30, are adopted in this research. The stopping criteria are set as follows: (a) the algorithm stops if there is no improvement in the minimum MSE_{LOO} for five consecutive iterations; (b) the algorithm stops if the number of iterations performed reaches the maximum value allowed, i.e. 50.

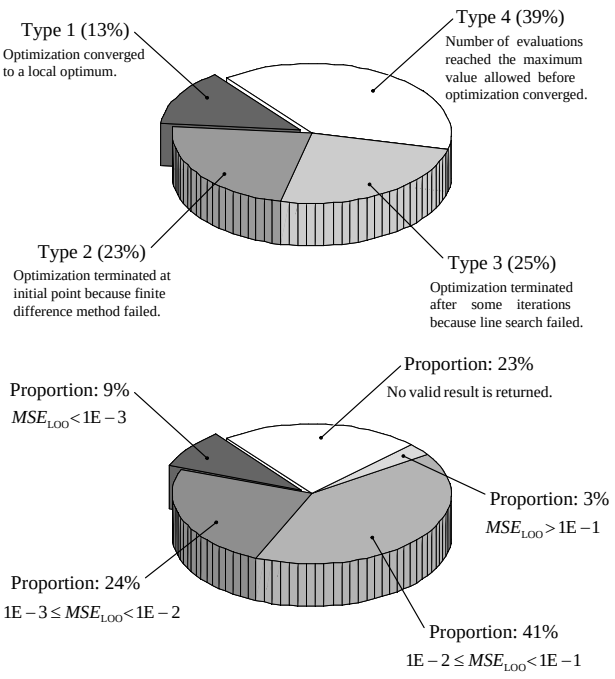


Figure 6. Statistics on the results of the 100 independent experiments

The GA/PSO-SVM algorithm will terminate when either one of the two conditions is met. If the stopping criterion (a) is met, we think that the algorithm has converged to a solution, i.e. a (C, ε, σ) point. In that case, we take $N^*_{\text{itr}} = N_{\text{itr}} - 4$ as the number of iterations required to find this solution (N_{itr} is the number of iterations executed before the algorithm terminates). In addition, another parameter which denotes the number of MSE_{LOO} evaluations is also introduced, and it is

$$N^*_{\text{eva}} = (N^*_{\text{itr}} + 1) \times S_{\text{pop}} \tag{20}$$

where S_{pop} is the size of the population. Because the calculation time consumed is directly proportional to the number of MSE_{LOO} evaluations executed, N^*_{eva} can be used to measure the convergence speed of an algorithm. The smaller N^*_{eva} is, the faster this algorithm converges.

TABLE II.
PARAMETER SETTINGS FOR GA/PSO-SVM

Initial search range for V_1	0 ~ 8
Initial search range for V_2	−12 ~ 0
Initial search range for V_3	−2 ~ 1
Population size	10, 20, 30
Max generation/iteration	50
R_{cros}	1.2
F_{cros}	0.8
P_{mut}	0.01
c_1	2
c_2	2
w	0.9
$v_{\max}$ for V_1	8
$v_{\max}$ for V_2	12
$v_{\max}$ for V_3	3

As mentioned before, three different population sizes, i.e. 10, 20 and 30, are considered. Hence a total of six cases are investigated, and they are GA-SVM-10, GA-SVM-20, GA-SVM-30, PSO-SVM-10, PSO-SVM-20 and PSO-SVM-30. Due to the stochastic character of the proposed methods, each of the above six approaches was repeated fifty times.

The stopping criterion (a) was met in all 300 runs. Namely, all trials performed converged to a (C, ε, σ) point within 50 iterations. Statistics including the maximum (max), minimum (min), average (avg), and standard deviation (std), are calculated on the obtained results such as N^*_{itr}, N^*_{eva}, and MSE_{LOO}, and they are listed in Table III, Table IV, and Table V.

By examining the data from columns 2, 3 and 4 in Table III, we can see that for GA-SVM, the approach with the smallest S_{pop}, i.e. GA-SVM-10, tends to take the most iterations to converge. Apparently, a similar conclusion can be drawn for PSO-SVM (on the basis of the data from columns 5, 6 and 7 in Table III). Comparing the data in column 5 with 2, column 6 with 3, and column 7 with 4 (of Table III), we can also see that for a given S_{pop}, PSO-SVM tends to take fewer iterations to converge.

Not only does the calculation time consumed depend on the iterations executed but also on the chosen population size. Compared with N^*_{itr}, the N^*_{eva} parameter which is defined in (20) is more useful for measuring the convergence speed. By examining the second row and the third row of Table IV, where the maximum/minimum of N^*_{eva} is shown, we can see that for both GA-SVM and PSO-SVM, the larger S_{pop} is, the wider the distribution of N^*_{eva} values is.

Besides, we can also see that for both GA-SVM and PSO-SVM, the average and standard deviation of N^*_{eva} values both increase as population size goes up. This means that (for both GA-SVM and PSO-SVM): (i) the larger S_{pop} is, the larger the expected value of calculation time will be; (2) the larger S_{pop} is, the larger the upper limit of calculation time will be. Thus, we can conclude that from the viewpoint of convergence speed, it is not appropriate to adopt a large S_{pop}. If we compare the data in column 5 with 2, column 6 with 3, and column 7 with 4 (of Table IV), it can be seen that for a given S_{pop}, PSO-SVM is superior to GA-SVM in convergence speed.

After analyzing the data from each row in Table IV separately, we find that PSO-SVM-10 has the lowest value for all the four statistics investigated, i.e. max of N^*_{eva}, min of N^*_{eva}, avg of N^*_{eva}, and std of N^*_{eva}. This fact implies that among the six approaches investigated, PSO-SVM-10 tends to take the fewest MSE_{LOO} evaluations to converge, i.e. PSO-SVM-10 has the highest average convergence speed.

It can be seen from Table V that: (i) GA-SVM-10 has the highest value for all the four statistics investigated, i.e. max of MSE_{LOO}, min of MSE_{LOO}, avg of MSE_{LOO}, and std of MSE_{LOO}; (ii) the data in the other five columns are close together.

TABLE III.
STATISTICS ON THE N^*_{itr} RESULTS OF GA-SVM AND PSO-SVM

	GA-SVM			PSO-SVM		
	$S_{pop}=10$	$S_{pop}=20$	$S_{pop}=30$	$S_{pop}=10$	$S_{pop}=20$	$S_{pop}=30$
max of N^*_{itr}	30	19	20	10	8	8
min of N^*_{itr}	6	3	3	3	1	1
avg of N^*_{itr}	14.8400	7.7200	6.6800	6.0200	4.5000	3.9800
std of N^*_{itr}	7.1866	4.6425	4.5689	1.9112	1.8763	1.6224

TABLE IV.
STATISTICS ON THE N^*_{eva} RESULTS OF GA-SVM AND PSO-SVM

	GA-SVM			PSO-SVM		
	$S_{pop}=10$	$S_{pop}=20$	$S_{pop}=30$	$S_{pop}=10$	$S_{pop}=20$	$S_{pop}=30$
max of N^*_{eva}	310	400	630	110	180	270
min of N^*_{eva}	70	80	120	40	40	60
avg of N^*_{eva}	158.4000	174.4000	230.4000	70.2000	110	149.4000
std of N^*_{eva}	71.8661	92.8497	137.0678	19.1119	37.5255	48.6726

TABLE V.
STATISTICS ON THE MSE_{LOO} RESULTS OF GA-SVM AND PSO-SVM

	GA-SVM			PSO-SVM		
	$S_{pop}=10$	$S_{pop}=20$	$S_{pop}=30$	$S_{pop}=10$	$S_{pop}=20$	$S_{pop}=30$
max of MSE_{LOO} ($\times$1E−4)	78.2636	7.8368	7.3383	7.7006	7.1218	6.1025
min of MSE_{LOO} ($\times$1E−4)	5.3872	4.9036	4.9026	4.9004	4.9005	4.9069
avg of MSE_{LOO} ($\times$1E−4)	31.5237	5.4456	5.4219	5.2967	5.3671	5.2921
std of MSE_{LOO} ($\times$1E−4)	19.8315	0.6670	0.5602	0.5012	0.4712	0.4123

In summary, from the viewpoint of solution quality, GA-SVM-10 is the worst one among the six approaches investigated, while there are no significant differences among the other five.

We can also rank the approaches investigated in order of convergence speed, and the result is PSO-SVM-10 > PSO-SVM-20 > PSO-SVM-30 > GA-SVM-20 > GA-SVM-30. We intentionally ignored GA-SVM-10 because numerical experiments prove that it is ineffective in minimizing SVM MSE_{LOO}. In general, PSO-SVM is preferred to GA-SVM. If one chooses PSO-SVM, 10 could be suggested as a suitable value for the size of population. If one chooses GA-SVM, 20 could be suggested as a suitable value for the size of population.

V. CONCLUSIONS

Because SVM generalization performance strongly depends on the right setting of hyper-parameters C, ε, and the kernel parameter σ, these three parameters need to be selected properly. In this paper two methods are presented for estimating the optimal SVM parameter settings, and they are GA-SVM and PSO-SVM. We used a 2-D test function to evaluate grid search, gradient-based search, and our GA/PSO-SVM method, and the experiment results show that:

- If the step sizes for C, ε, and σ are sufficiently small, the grid search method can achieve a solution of high quality. However, in that case the required number of MSE_{LOO} evaluations is extremely large. In the example presented in this paper, the required number is 9621, and the CPU time consumed is about 42 hours.
- The gradient-based search is not a good method for finding a minimum of MSE_{LOO}. In the example presented in this paper, the quasi-Newton procedure was repeated 100 times, but only 9 out of 100 trials yielded a solution at which MSE_{LOO} is smaller than 1.0E−3.
- GA-SVM and PSO-SVM can both yield solutions of comparable quality to that achieved by the grid search method. GA-SVM-10, however, was the exception.
- From an overall point of view, the proposed methods (GA-SVM and PSO-SVM) are slightly inferior to gradient-based method in convergence speed, which is measured in terms of N^*_{eva}. However, when the population is relatively small, PSO-SVM can achieve a convergence speed comparable to that of gradient-based search.
- Both of the two proposed methods have a much lower computational cost when compared with the grid search. In the example presented in this paper, the CPU time consumed by GA-SVM ranges from 0.66 to 3.32 hours, and the CPU time consumed by PSO-SVM ranges from 0.38 to 1.71 hours.
- For a given population size, PSO-SVM is superior to GA-SVM in convergence speed. The recommended population size for GA-SVM is 20, and the recommended population size for PSO-SVM is 10.

Summarizing, GA-SVM and PSO-SVM were developed as relatively fast alternatives for the time consuming grid search approach. Both methods, especially the latter, can efficiently solve the problem of estimating the optimal SVM parameter settings at a reasonable computational cost.

ACKNOWLEDGMENT

This research was supported by the National High Technology Research and Development Program under the project No. 2006AA04Z405, which is gratefully acknowledged by the authors.

REFERENCES

[1] E. Rigoni and A. Lovison, "Automatic sizing of neural networks for function approximation," *IEEE International Conference on Systems, Man and Cybernetics*, pp. 2005–2010, October 2007.

[2] R. R. Barton and M. Meckesheimer, "Metamodel-based simulation optimization," in *Handbooks in Operations Research and Management Science*, vol. 13, Elsevier Science, 2006, pp. 535–574.

[3] X. F. Yuan and Y. N. Wang, "Parameter selection of SVM for function approximation based on chaos optimization," *Journal of Systems Engineering and Electronics*, vol. 19, pp. 191–197, 2008.

[4] V. Cherkassky and Y. Ma, "Practical selection of SVM parameters and noise estimation for SVM regression," *Neural Networks*, vol. 17, pp. 113–126, 2004.

[5] B. Üstün, W. J. Melssen, M. Oudenhuijzen and L. M. C. Buydens, "Determination of optimal support vector regression parameters by genetic algorithms and simplex optimization," *Analytica Chimica Acta*, vol. 54, pp. 292–305, 2005.

[6] K. Ito and R. Nakano, "Optimizing support vector regression hyper-parameters based on cross-validation," *Proceedings of the International Joint Conference on Neural Networks*, vol. 3, pp. 2077–2082, July 2003.

[7] Z. Y. Luo, P. Wang, Y. G. Li, W. F. Zhang, W. Tang and M. Xiang, "Quantum-inspired evolutionary tuning of SVM parameters," *Progress in Natural Science*, vol. 18, pp. 475–480, 2008.

[8] A. C. Keys, L. P. Rees and A. G. Greenwood, "Performance measures for selection of metamodels to be used in simulation optimization," *Decision Sciences*, vol. 33, pp. 31–57, 2002.

Yuan Ren was born in Xinjiang Province, China, in 1982. He received his B.S. degree in flight vehicle propulsion engineering from Beijing University of aeronautics and astronautics in 2004.

Currently, he is a Ph.D. candidate at the school of jet propulsion, Beijing University of aeronautics and astronautics. He has already published 15 articles in academic journals and conference proceedings. He was granted Airbus Scholarship and Guanghua Scholarship for his research effort. His main research interest focuses on artificial intelligence and its application in reliability engineering and optimization design.

Mr. Ren is a student member of China Computer Federation.

Guangchen Bai was born in Heilongjiang Province, China, in 1962. He received his Ph.D. degree in mechanical engineering from Harbin institute of technology in 1993.

Now he is a professor of Beijing University of aeronautics and astronautics and conducts research in the areas of reliability engineering and optimization design. His research activities were supported by National Natural Science Foundation, National Postdoctoral Science Foundation, Aeronautical Supporting Technology Foundation and National High-tech Research and Development Program of China.