Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 385372, 8 pages
http://dx.doi.org/10.1155/2013/385372

Research Article

A New Method on Software Reliability Prediction

Zhang Xiaonan,[1,2] Yang Junfeng,[1] Du Siliang,[1] and Huang Shudong[3]

[1] *The 28th Research Institute of China Electronics Technology Group Corporation, Nanjing 210007, China*
[2] *College of Field Engineering, PLA University of Science and Technology, Nanjing 210007, China*
[3] *Jiangsu Posts & Telecommunications Planning and Designing Institute Co., Ltd., Nanjing 210007, China*

Correspondence should be addressed to Zhang Xiaonan; zxn8206@163.com

Received 27 December 2012; Accepted 14 February 2013

Academic Editor: Zheng-Guang Wu

As we all know, relevant data during software life cycle can be used to analyze and predict software reliability. Firstly, the major disadvantages of the current software reliability models are discussed. And then based on analyzing classic PSO-SVM model and the characteristics of software reliability prediction, some measures of the improved PSO-SVM model are proposed, and the improved model is established. Lastly, simulation results show that compared with classic models, the improved model has better prediction precision, better generalization ability, and lower dependence on the number of samples, which is more applicable for software reliability prediction.

1. Introduction

Reliability is an important software quality characteristic related to the probability that system works without failures over a period of time in a certain environment. The estimation or prediction of the reliability level is a very important task. This level can be used to plan test, deployment, and maintenance activities. To help in this task, the use of modeling and prediction of software reliability are a crucial issue.

Different types of software reliability prediction models consider different elements of the software project, such as the specification and codification of the programs, and are usually based on characteristics of the testing activity. Some of those models consider the time between failures [1–3]. Others consider the coverage of a test criterion [4–7]. A criterion can be viewed as a predicate to be satisfied by the test cases and can be used to evaluate test sets [8]. The advantage of models based on coverage is that they are independent of the operation profile. However, models based on time are most commonly used. Due to the general nonlinear function mapping capabilities, artificial neural networks have received increasing attention in time series forecasting [9–11]. These works show that ANN nonparametric models present better results than traditional ones. However, most of those works explore only models based on time. In addition, the ANN itself is filled with strong experience, the theory is not strict or easily interpreted, and then it easily converges at the local minimum point. So the application of artificial neural networks is very limited for software reliability prediction.

Recently, a novel machine learning technique, called support vector machine (SVM), has drawn much attention in the fields of pattern classification and regression forecasting. SVM was first introduced by Vapnik and his colleagues in 1995 [5]. SVM is a kind of classifier's studying method on statistic study theory. This algorithm derives from linear classifier and can solve the problem of two kinds classifiers; later this algorithm is applied in nonlinear fields; that is to say, we can find the optimal hyperplane (large margin) to classify the samples set. It is an approximate implementation to the structure risk minimization (SRM) principle in statistical learning theory, rather than the empirical risk minimization (ERM) method [5].

Compared with traditional neural networks, SVM can use the theory of minimizing the structure risk to avoid the problems of excessive study, calamity data, local minimal value, and so on. SVM has been successfully used for machine learning with large and high-dimensional data sets. These attractive properties make SVM become a promising technique. This is due to the fact that the generalization property of an SVM does not depend on the complete training data but only a subset thereof, the so-called support vectors. Now, SVM has been applied in many fields [12–14]. However, the essence of SVM training is solving convex quadratic programming problems with linear equality constraint. The classic methods for solving nonlinear programming, such as the Newton method and quasi-Newton method, have large computing. So the predicted effect is not so perfect.

In order to overcome the limitations of SVM mentioned previously, the researchers apply particle swarm optimization (PSO) to the training of the SVM [15, 16]. PSO is an intuitive and easy-to-implement algorithm from the swarm intelligence community. To replace the need for numeric solvers, a PSO algorithm based on chaos searching (CPSO) which improves the convergence speed and the abilities of searching for the global optima is proposed and shown to be feasible in solving the SVM quadratic programming problem, but the research is fit for the large sample set, which is ineffective for less sample data in early software reliability prediction.

In this paper, based on analyzing classic PSO-SVM model and the characteristic of software reliability prediction, we propose concrete measures of the improved PSO-SVM model and establish the improved PSO-SVM model. This paper is organized as follows: Section 2 summarizes the classic PSO-SVM model. Section 3 analyzes characteristic of software reliability prediction and PSO-SVM applicability and then proposes specific improved strategy. Section 4 describes results of two compared simulation experiments. Finally, Section 5 concludes the paper.

2. Traditional PSO-SVM Characteristics Analysis

Traditional PSO-SVM model used PSO algorithm to optimize the model parameters and kernel parameter of SVM and improved the prediction accuracy by searching the optimal parameters value. SVM classification was first proposed for the second largest interval algorithm and then gradually extended to the field of nonlinear regression. SVM nonlinear regression prediction is similar to classification problems, which are calculated according to the given decision function, and then classify and predict. Regression problem retains the main features of the largest interval algorithm, which is to minimize a convex function, and the nonlinear function can be got by studying the linear devices in the kernel feature space; the difference is mainly reflected in a given data set.

Suppose that a given data set is $\{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_l, y_l)\}$, where $x_i \in R^n$, $y_i \in R$, $i = 1, 2, 3, \ldots, l$.

The original SVM problem can be expressed as

$$\begin{aligned} \min_{\omega \in R^n,\, b \in R} \quad & J(\omega, \xi) = \frac{1}{2}\omega^T \omega + C\sum_{i=1}^{l} (\xi_i + \xi_i^*) \\ \text{s.t.} \quad & (\omega \cdot \phi(x_i) + b) - y_i \le \xi_i^* + \varepsilon, \\ & \qquad i = 1, \ldots, l \\ & y_i - (\omega \cdot \phi(x_i) + b) \le \xi_i + \varepsilon, \quad i = 1, \ldots, l \\ & \xi_i, \xi_i^* \ge 0, \quad i = 1, \ldots, l. \end{aligned} \tag{1}$$

Since the dimension of feature space is high, and the objective function is nondifferentiable, in order to facilitate the calculation, the dot product kernel function technology and Wolf dual theory are introduced, which is transformed into the dual problem. The original problem is transformed into the dual quadratic programming problem. Specific methods are firstly constructing the Lagrangian function (such as type 2) and then calculating the partial derivative of each variable; the result is substituted into the original problem:

$$\begin{aligned} & L(\omega, b, \xi, \xi^*, \alpha, \alpha^*, \gamma, \gamma^*) \\ & = \frac{1}{2}\omega^T \omega + C\sum_{i=1}^{l} (\xi_i + \xi_i^*) \\ & \quad - \sum_{i=1}^{l} \alpha_i (\xi_i + \varepsilon - y_i + (\omega \cdot \phi(x_i) + b)) \\ & \quad - \sum_{i=1}^{l} \alpha_i^* (\xi_i^* + \varepsilon - (\omega \cdot \phi(x_i) + b) + y_i) \\ & \quad - \sum_{i=1}^{l} (\xi_i \gamma_i + \xi_i^* \gamma_i^*). \end{aligned} \tag{2}$$

Lagrange function requires ω, b, ξ, ξ^* to be minimized; thus

$$\begin{aligned} \min_{\alpha^{(*)} \in R^{2l}} \quad & \frac{1}{2}\sum_{i,j=1}^{l} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) K(x_i \cdot x_j) \\ & + \varepsilon \sum_{i=1}^{l} (\alpha_i^* + \alpha_i) - \sum_{i=1}^{l} y_i (\alpha_i^* + \alpha_i) \\ \text{s.t.} \quad & \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) = 0 \\ & 0 \le \alpha_i^{(*)} \le C, \quad i = 1, \ldots, l. \end{aligned} \tag{3}$$

Function $f(x)$ can be directly expressed as

$$f(x) = \sum_{i=1}^{l} (\overline{\alpha_i^*} - \overline{\alpha_i}) K(x_i, x) + \overline{b}. \tag{4}$$

When checking

$$\overline{\alpha_j} : \overline{b} = y_j - \sum_{i=1}^{l} \left(\overline{\alpha_i^*} - \overline{\alpha_i}\right) K\left(x_i, x_j\right) + \varepsilon. \quad (5)$$

When checking

$$\overline{\alpha_k^*} : \overline{b} = y_k - \sum_{i=1}^{l} \left(\overline{\alpha_i^*} - \overline{\alpha_i}\right) K\left(x_i, x_j\right) - \varepsilon. \quad (6)$$

When SVM solves nonlinear regression problems, the nonlinear mapping $\phi(\cdot)$ makes feature space mapping into high-dimensional feature space and then finishes linear regression in high-dimensional space. In order to reduce the sensitivity of the prediction error, the objective function of nonlinear regression model is defined insensitive loss function, and the slack variable ε is introduced to ignore the fitting error of less than ε, which ensures that the model is the existence of global minimum and reliable generalization sector optimization.

The original problem $(1/2)\omega^T\omega$ is for the regularized part, whose role is to make the function smoother to enhance the generalization ability; ξ_i and ξ_i^* reflect the training point margin of error; $\sum_{i=1}^{l}(\xi_i+\xi_i^*)$ reflects the experience risk of the model; error penalty factor C is the model parameters, which determines the balance between the empirical risk and the regularization parts.

Dual problem is the quadratic programming problem, where $K(x_i \cdot x_j) = \langle \phi(x_i) \cdot \phi(x_j) \rangle$ is called kernel function, including linear kernel, polynomial kernel function, RBF kernel function, and sigmoid kernel function. The RBF kernel function can fully reflect the software reliability nonlinear characteristics, which is used in the construction of prediction model.

Consider RBF kernel function:

$$K\left(x_i, x_j\right) = \exp\left(-\frac{\left|x_i - x_j\right|^2}{2\sigma^2}\right). \quad (7)$$

When making PSO optimize C, ε and σ^2 in SVM model, the population is constantly updating from the best local position to the best global locations in the iterative process. Assuming that the population size is m, the d dimensional space position of the particle I is $x_i = \{x_{i1}, x_{i2}, \ldots, x_{id}\}$, speed is $v_i = \{v_{i1}, v_{i2}, \ldots, v_{id}\}$, the optimal local location is $p\text{best}_i = \{p_{i1}, p_{i2}, \ldots, p_{id}\}$, and the best global position is $g\text{best} = \{g_1, g_2, \ldots, g_d\}$. Specific methods are as follows.

Speed:

$$v_{ij}(t+1) = \omega \cdot v_{ij}(t) + c_1 \cdot r_1 \cdot \left(p_{ij} - x_{ij}\right) + c_2 \cdot r_2 \cdot \left(g_j - x_{ij}\right). \quad (8)$$

Location:

$$x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \quad (9)$$

where t is the current iteration number, c_1 and c_2 are the acceleration factor, c_1 is own dependence on memory of particles, c_2 is the impact of other particles on the particle itself, which make each particle close to $p$best and $g$best, and r_1 and r_2 are uniform distribution random numbers in $(0, 1)$, which is used to simulate the slight disturbance.

3. Model Applicability and Improved Measures Analysis

The traditional PSO-SVM has many outstanding advantages, which are adapted to software reliability prediction characteristics, which are shown in Table 1.

Although traditional PSO-SVM prediction model has many advantages, because of inherent weaknesses and deficiencies of PSO and SVM algorithms, this paper proposes the correspondent improved strategy to get the optimal software reliability prediction model. The model shortcomings and correspondent improved measures are shown in Table 2.

4. Improved Model

4.1. Block Population Initialized Measure. Particle swarm optimization (PSO) is a global optimal search algorithm, so this algorithm should quickly search to obtain the optimal value. However, the traditional particle swarm is randomly generated within the region in the whole population, which cannot fully guarantee that it is dispersed throughout the search space. If we can put search space into many blocks, it will be able to improve the nonuniform status. The main idea is that each particle is almost evenly distributed; assuming that the number of particles is n, then the entire search space is divided into n small areas:

$$\begin{aligned} X_{i,k} &\in \left[a_k + f_k\left(b_k - a_k\right), a_k + f_{k+1}\left(b_k - a_k\right)\right] \\ &\qquad k = 1, 2, \ldots, D, \\ f_k &= (i-1) \bmod C^{D-K}, \\ f_D &= (i-1) - \sum_{j=1}^{D-1} f_j C^{D-j}, \\ C &= \sqrt[D]{n}, \end{aligned} \quad (10)$$

where a_k and b_k are expressed in the range of values in k dimension; then the initial position of particle i is

$$X_{i,k} = a_k + f_k\left(b_k - a_k\right) + \frac{1}{\sqrt[D]{n}}\left(b_k - a_k\right) \cdot r, \quad (11)$$

where r is random number values in $[0, 1]$.

4.2. Adaptive Inertia Factor Measure. Inertia factor w in the PSO algorithm makes the particle velocity update with historical memory, which adjusts history speed to the local and global optimal speed in order to balance the relationship between the global search ability and the local one. When the iteration begins, the larger inertia weight can enhance the global search capability; that is, the larger the search area, in the latter, the smaller the inertia weight which can be

Table 1: Model advantages and prediction characteristics.

Model advantages	Prediction characteristics
The parameter C adjusts the ratio between configuration risk and experience risk, avoiding the overlearning problems and improving the prediction model generalization ability.	Less software reliability sample set
The slack variable is introduced to reduce the error sensitivity of prediction model. The model transforms the original problem into a dual problem, making solving process into a convex quadratic programming problem, getting the optimal solution easily.	Complex prediction process
The kernel function is introduced, which makes multidimensional input space into high dimensional space in order to solve multidimensional space problems.	More software reliability characteristic parameters
The prediction process is operated in the high-dimensional space, which makes original non-linear prediction problem transform into a linear problem, and then the results are reduced to the nonlinear problem solution.	Nonlinear characteristics of software reliability prediction
PSO is used to search the optimal solution of parameters to improve the overall prediction accuracy.	It is difficult to get the optimal solution for the software reliability prediction model parameters.

Table 2: Model shortcomings and improved measures.

Model shortcomings	Improved measures
The model parameters and kernel parameter of SVM are random, which are not conducive to search the optimal parameters, thereby reducing the prediction accuracy and efficiency.	Block population initialized measure
PSO inertia factor is fixed, and the local and global search abilities are limited, which reduces obtaining the optimal solution ability.	Adaptive inertia factor
PSO algorithm is easy to fall into local minimum in the latter prediction part.	Nonevolution number of mutation strategies
When low-dimensional space transforms into high dimensional space and solves quadratic programming problems, if the number of input spaces is high, computational efficiency will be a new problem.	Transforming SVM into LSSVM

enhanced local search ability, which is conducive to better local search. If we can prolong the former and latter search times, we will improve the overall algorithm performance, so the adaptive weight update method is as follows:

$$w = w_{\min} + (w_{\max} - w_{\min}) \times \exp\left(-20 \times \left(\frac{t}{t_{\max}}\right)^6\right). \quad (12)$$

Suppose that $w_{\max}, w_{\min}$ are 0.9 and 0.1. The corresponding inertia weight curve is shown in Figure 1.

In the previous table, the curve expresses the relationship between the power of $(t/t_{\max})$. and w. Compared with other values, when the power is 6, the particle search time is the longest. The method makes the inertia weight a larger value in the iterative initial time, and smaller in the latter, which extends the global and local search times, strengthens the search ability, and balances the global search ability and local search ability.

4.3. Nonevolution Number of Mutation Measures. Mutation mechanism comes from the genetic algorithm, which is mainly used to overcome the problem of converging at local minimum in the iterative process. Standard PSO is easy to converge at local optimal solution in high-dimensional function optimization problems, and nonevolution number of the particles can determine whether it is entering into the local optimal solution. Therefore, if nonevolution number and mutation operators can be introduced into the PSO, they will be selection criteria as the mutation time in order to overcome the local minimum problem. Specific strategies are as follows.

(1) Calculate the fitness changing rate (abbreviated as FCR hereinafter): the FCR is the fitness changing rate of p_g (history optimal particle position) between the current iteration and the previous M times ($M = 1$):

$$F\ \text{Slope} = \frac{f\left(p_g(t)\right) - f\left(p_g(t-M)\right)}{f\left(p_g(t)\right)}. \quad (13)$$

(2) Count nonevolution number: in the beginning of the evolution, the non-evolution number is stop time; the fitness changing threshold value is slope value; non-evolution limit is MaxStep; mutation probability is p_m. In the iterative process, the non-evolution number is determined by the fitness changing rate, as follows:

$$\begin{aligned} F\ \text{Slope} < \text{Slope value} \quad & \text{stop time} = \text{stop time} + 1 \\ F\ \text{Slope} > \text{Slope value} \quad & \text{stop time} = 0. \end{aligned} \quad (14)$$

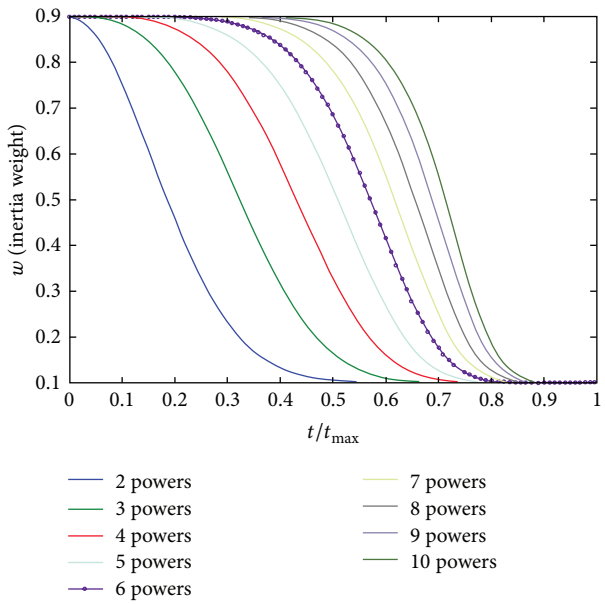

Figure 1: Adaptive inertia weight curve.

If non-evolution number is more than the limit (MaxStep), the algorithm may be stopped, and we can do mutation operation based on the mutation probability:

$$p_{gi}(t+1) = p_{gi}(t) + 0.5 \times r \times p_{gi}(t), \tag{15}$$

where r is the random number in $[0, 1]$. The improvement makes the particles continue to approach the global optimum when converging at local minimum in training.

4.4. LSSVM Measure. Least squares support vector machine (LSSVM) has the two main deformations. Firstly, the least squares linear system is introduced as a loss function, which makes equality constraints replace inequality ones in SVM; secondly the quadratic programming solving replaces linear equations, which avoids insensitive loss function and greatly improves the learning efficiency and the training accuracy.

The standard SVM problem can be simplified as follows:

$$\begin{aligned} \min \quad & J(\omega, \xi) = \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{n}\xi_i^2 \\ \text{s.t.} \quad & y_i = \phi(x_i)\cdot\omega + b + \xi_i + \varepsilon. \end{aligned} \tag{16}$$

It should be noted that the one equality constraint in LSSVM is used instead of the two inequality constraints in SVM; the corresponding objective function $C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$ can also be replaced by $C\sum_{i=1}^{n}\xi_i^2$. Based on the standard method of transforming into dual problem in SVM, LSSVM can be converted into the dual problem through the derivation

$$\begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & K(x_1, x_1) + \dfrac{2}{c} & \cdots & K(x_1, x_n) \\ \vdots & \vdots & \cdots & \vdots \\ 1 & K(x_n, x_1) & \cdots & K(x_n, x_n) + \dfrac{2}{c} \end{bmatrix} \cdot \begin{bmatrix} b+\varepsilon \\ \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix} = \begin{bmatrix} 0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}. \tag{17}$$

Decision function is

$$f(x) = \sum_{i=1}^{n}\alpha_i K(x_i, x) + \overline{b}. \tag{18}$$

As solving linear equations, $\overline{b}$ in the decision function can be obtained through the equation, which can greatly reduce the computation and the model is more simple.

Table 3: Experimental data.

SN	LOC	FO	FI	PATH	FAULTS
1	29	4	1	4	0
2	29	4	1	4	2
3	32	2	2	2	1
4	33	3	27	4	1
5	37	7	18	16	1
6	41	7	1	14	4
7	55	1	1	12	2
8	64	6	1	14	0
9	69	3	1	8	1
10	101	4	4	12	5
11	120	3	10	22	6
12	164	14	10	221	11
13	270	9	1	80	17

5. The Flow Chart of the Improved PSO-LSSVM Model

The flow chart of improved PSO-LSSVM prediction model is shown in Figure 2; the dashed part expresses the improved process of PSO-LSSVM model.

6. Simulation Comparison

In order to evaluate the performance of the new model and compare it with the traditional model, the simulation experiment is shown as follows. Here, taking a military software system as an example, thirteen module indexes and module defect number are shown in Table 3.

SN is module number; LOC is module size (the number of line codes is units); FO is module output; FI is module input; PATH is module control flow path; FAULTS is the number of module defects.

In order to evaluate the prediction accuracy of the optimization model, we carry out two experiments. Experiment

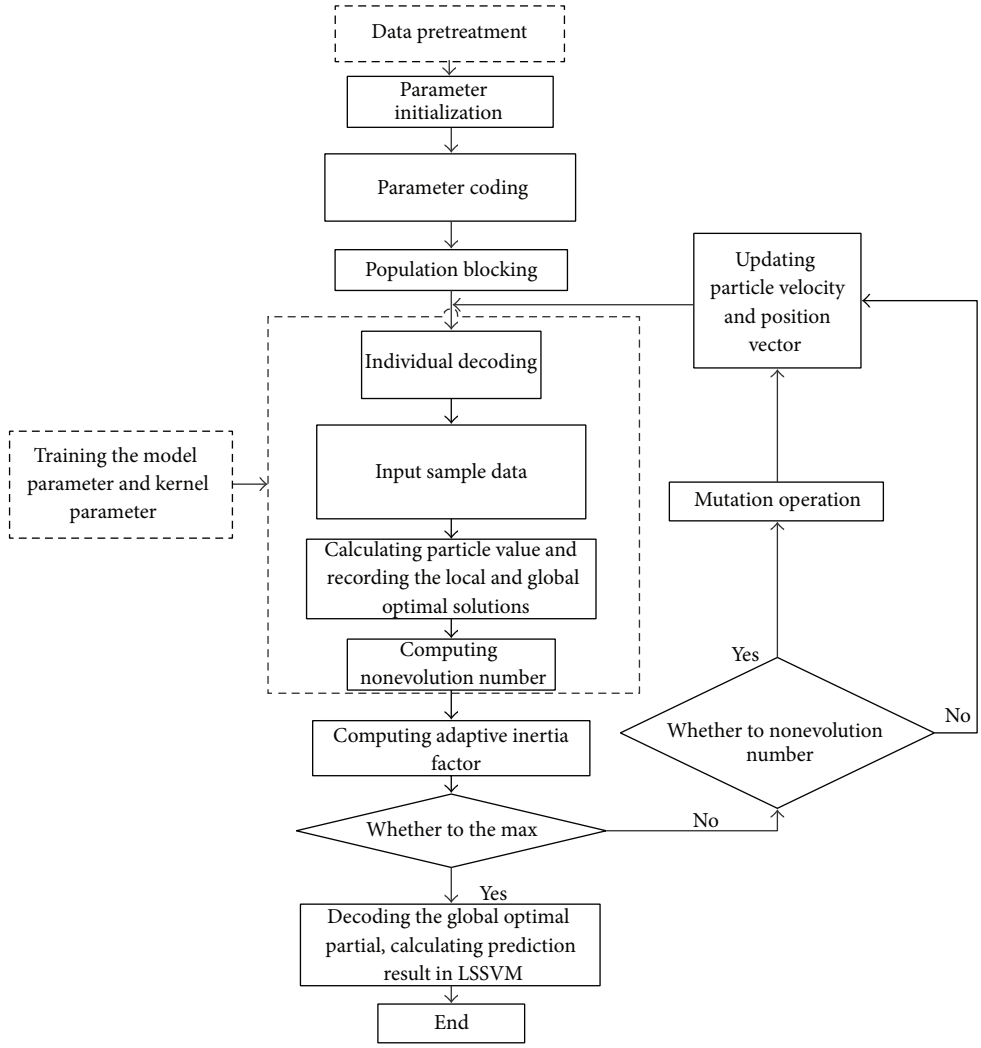

Figure 2: The flow chart of improved PSO-LSSVM prediction model.

1: all 13 data samples are divided into two parts, where the first 10 are as the training set, and the last 3 are as the test set. Experiment 2: the first 6 are as the training set, and the last 3 are also as the test set to evaluate the model as a result.

After the training samples and test samples are normalized [17], respectively, we input them into the BP network model, the traditional PSO-SVM model, and the optimization PSO-LSSVM model. Where BP prediction model uses the momentum factors model, the hidden nodes of model are 18; the training objective is 0.00001. In accordance with the cross-validation algorithm and the depth search algorithm, after 2 rounds of selection, the traditional PSO-SVM prediction model parameter is $C = 499$, and the nuclear kernel parameter is $\sigma^2 = 5$. Both the traditional and the optimization PSO-SVM models make RBF as kernel function. In the model training process, the error curve of BP prediction model and the optimization PSO-LSSVM model are shown in Figures 3 and 4, respectively.

We can see from the tables that the improved PSO-LSSVM prediction model training error decreases rapidly, and about 200 times training tends to stop; however BP prediction model can meet the training requirements after 1733 times, which is significantly higher than the improved PSO-LSSVM prediction model.

Table 4: Prediction results.

Experiment	True value	BP	Traditional PSO-SVM	Optimization PSO-LSSVM
1	6	5.7204	6.2664	6.1009
	11	10.1666	11.4748	11.1924
	17	16.0432	17.6283	17.2037
2	6	5.2234	6.6326	6.1345
	11	9.1998	12.3895	11.2078
	17	14.586	19.0068	17.3257

After training, the three methods get corresponding prediction models applicable to sample data; therefore we can input prediction sample data into each model to forecast.

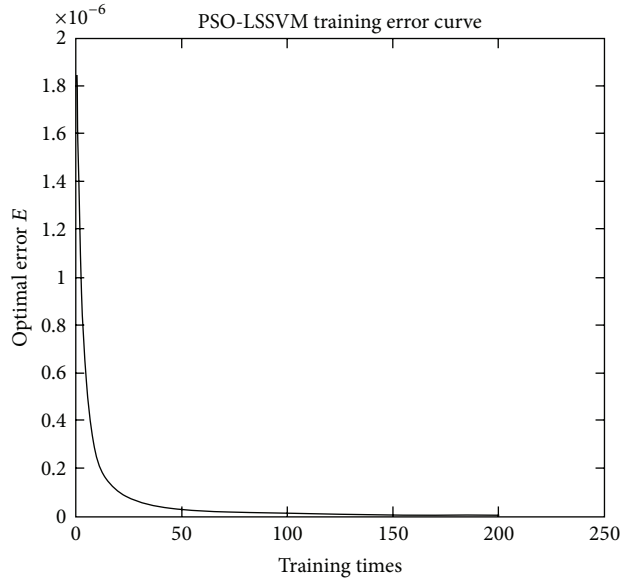

Figure 3: Improved PSO training error curve LSSVM.

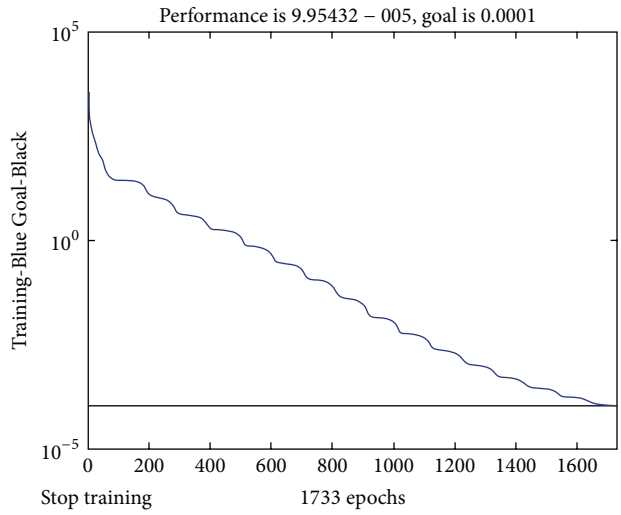

Figure 4: BP network error curve.

Because BP prediction model is greatly influenced by the initial parameters, in order to reduce the randomness, we calculate the average of 10 consecutive operations. Calculating the average percentage prediction error, the prediction results are shown in Table 4; the comparing result is shown in Table 5.

7. Conclusion

Because of using the optimized model parameters and kernel parameters in the improved PSO-LSSVM prediction model, the prediction accuracy is much higher than the traditional PSO-SVM model and BP prediction model; as the number of training samples decreases, the prediction accuracy of the improved PSO-LSSVM model is significantly higher than the traditional PSO-SVM model and BP model owing to its good generalization performance in less training samples. Thus, the improved PSO-LSSVM prediction model is better than the traditional PSO-SVM and BP prediction models in both training efficiency and prediction accuracy. Due to the current situation that the prediction sample set is small and the cost is high in software reliability prediction, the proposed model has important practical significance, and it may become the preferred prediction method for the less samples prediction projects.

Table 5: Prediction error comparing results.

Experiment	BP	Traditional PSO-SVM	Optimization PSO-LSSVM
1	4.66%	4.44%	1.68%
	7.57%	5.22%	1.75%
	5.62%	3.69%	1.21%
MPAPE	5.95%	4.45%	1.54%
2	12.94%	10.54%	2.24%
	16.37%	12.62%	1.89%
	14.20%	11.76%	1.93%
MPAPE	14.51%	11.64%	2.02%

References

[1] L. C. Briand, W. L. Melo, and J. Wüst, "Assessing the applicability of fault-proneness models across object-oriented software projects," *IEEE Transactions on Software Engineering*, vol. 28, no. 7, pp. 706–720, 2002.

[2] T. M. Khoshgoftaar and R. M. Szabo, "Using neural networks to predict software faults during testing," *IEEE Transactions on Reliability*, vol. 45, no. 3, pp. 456–462, 1996.

[3] N. Cristianini and J. S. Taylor, *An Introduction to Support Vector Machines*, Zhengguo Translation, Electronic Industry Press, Beijing, China, 2004.

[4] X. Li and S. Yanhua, "SVM early prediction of software reliability," *Hefei University of Technology*, vol. 7, no. 7, pp. 859–862, 2007.

[5] F. Zhe, "SVR-based software reliability prediction model," *Computer Engineering and Applications*, vol. 43, no. 13, pp. 120–123, 2007.

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[7] U. Paquet and A. P. Engelbrecht, "A new particle swamr optimizer for linearly constrained optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 227–233, Canberra, Australia, December 2003.

[8] U. Paquet and A. P. Engelbrecht, "Training support vector machines with particle swarms," in *Proceedings of the International Joint Conference on Neural Networks*, pp. 1593–1598, July 2003.

[9] Xiaodong, L. Xiangdong, and W. Rui, *Particle Swarm Algorithm and Its Application*, 12, Liaoning University Press, Shenyang, China, 2007.

[10] C. Fun, "Integrated model of software reliability analysis and research," *Computer Science*, vol. 36, no. 4, pp. 181–184, 2009.

[11] Z. Yanyan and G. Wei, "Summary of software reliability engineering," *Computer Science*, vol. 36, no. 2, pp. 20–25, 2009.

[12] A. Widodo and B. S. Yang, "Application of nonlinear feature extraction and support vector machines for fault diagnosis of induction motors," *Expert Systems with Applications*, vol. 33, no. 1, pp. 241–250, 2007.

[13] Q. Wu, "The forecasting model based on wavelet ν-support vector machine," *Expert Systems with Applications*, vol. 36, no. 4, pp. 7604–7610, 2009.

[14] Q. Wu, "The hybrid forecasting model based on chaotic mapping, genetic algorithm and support vector machine," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1776–1783, 2010.

[15] Q. Wu, "A hybrid-forecasting model based on Gaussian support vector machine and chaotic particle swarm optimization," *Expert Systems with Applications*, vol. 37, no. 3, pp. 2388–2394, 2010.

[16] L. Yushi, "Understanding the network-centered command information system," *Command Information System and Technology*, vol. 1, no. 1, pp. 1–4, 2010 (Chinese).

[17] L. Yu and Z. Wenyu, "Command and control technology for unmanned combat vehicles," *Command Information System and Technology*, vol. 2, no. 6, pp. 6–9, 2011 (Chinese).