Applied Soft Computing 15 (2014) 113–120

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Software reliability prediction model based on support vector regression with improved estimation of distribution algorithms

Cong Jin [a,*], Shu-Wei Jin [b]

[a] School of Computer Science, Central China Normal University, Wuhan 430079, PR China
[b] Département de Physique, École Normale Supérieure, 24, rue Lhomond, 75231 Paris Cedex 5, France

ARTICLE INFO

Article history:
Received 10 July 2012
Received in revised form
30 September 2013
Accepted 23 October 2013
Available online 31 October 2013

Keywords:
Support vector regression
Improved estimation of distribution algorithms
Software reliability prediction
Parameters optimization

ABSTRACT

Software reliability prediction plays a very important role in the analysis of software quality and balance of software cost. The data during software lifecycle is used to analyze and predict software reliability. However, predicting the variability of software reliability with time is very difficult. Recently, support vector regression (SVR) has been widely applied to solve nonlinear predicting problems in many fields and has obtained good performance in many situations; however it is still difficult to optimize SVR's parameters. Previously, some optimization algorithms have been used to find better parameters of SVR, but these existing algorithms usually are not fully satisfactory. In this paper, we first improve estimation of distribution algorithms (EDA) in order to maintain the diversity of the population, and then a hybrid improved estimation of distribution algorithms (IEDA) and SVR model, called IEDA-SVR model, is proposed. IEDA is used to optimize parameters of SVR, and IEDA-SVR model is used to predict software reliability. We compare IEDA-SVR model with other software reliability models using real software failure datasets. The experimental results show that the IEDA-SVR model has better prediction performance than the other models.

1. Introduction

Reliability is the ability of software system to perform its required functions under stated conditions for a specified period of time, and it is an important characteristic inherent in the concept of software quality. It is intimately connected with defects and faults. As more and more faults are encountered, the software reliability will decrease. Software reliability generally changes with time, and these changes can be treated as a time series process.

Artificial neural networks (ANN) have general nonlinear mapping capabilities, and have increasingly attracted attention in the field of time series predicting [1–3]. In [4], the reliability of the systems can be predicted by feed-forward multi-layer ANN and radial basis function ANN respectively. The ANN technology has better prediction performance than the autoregressive integrated moving average (ARIMA) approach. In [5], ANN has contributed significantly to software reliability prediction, and which achieved better prediction performance than traditional statistical models. In [6], the counter-propagation and back-propagation ANN models were used to estimate parameters of a reliability distribution with only a small dataset. The experimental results show that the proposed approach improves the accuracy of reliability predicting. In [7], the system reliability may be predicted by a hybrid learning neural fuzzy system. Numerical results demonstrate that the proposed model achieved more accurate predicting results than ARIMA and generalized regression ANN model (GRNN). However, the ANN suffers from a number of weaknesses, e.g., it is based on gradient descent, and it is easy to local minima.

Recently, support vector machines (SVMs) [8–11] have been widely applied to solve nonlinear predicting problems in many fields. With the introduction of ε-insensitive loss function, it has been also extended to solve nonlinear regression estimation problems, such as new techniques known as support vector regression (SVR) [12]. In [13], the SVM was used to solve financial time series problems. The experimental results demonstrate that SVM forecasts better than back propagation (BP) algorithm. In [14], a two-step kernel learning method based on SVR was proposed for predicting financial time series. The results confirm the advantage of SVR. However, although SVR has very good learning performance and generalization ability, there is no structured way to determine the parameters of SVR.

Estimation of distribution algorithms (EDA) [15], sometimes called probabilistic model-building genetic algorithm (GA) [16], have emerged as a generalization of GA, for overcoming the two main problems: poor performance in certain deceptive problems and the difficulty of mathematically modeling a huge number of algorithm variants [17]. In GA, a population of candidate solutions

* Corresponding author. Tel.: +86 02788664026.
E-mail address: jincong@mail.ccnu.edu.cn (C. Jin).

1568-4946/$ – see front matter © 2013 Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.asoc.2013.10.016

to a problem is maintained as part of the search for an optimum solution. This population is typically represented explicitly as an array of objects. Depending on the specifics of the GA, the objects might be bit strings, vectors of real numbers or some custom representation. In EDA, this explicit representation of the population is replaced with a probability distribution over the choices available at each position in the vector that represents a population member. Moreover, in GA, new candidate solutions are often generated by combining and modifying existing solutions in a stochastic way. The underlying probability distribution of new solutions over the space of possible solutions is usually not explicitly specified. In EDA, a population may be approximated with a probability distribution and new candidate solutions can be obtained by sampling this distribution. Compared with traditional GA, EDA can solve nonlinear variable coupling problems for complex optimization.

Software reliability predictions are used for various purposes, such as software planning, reliability assessment, detecting faults in manufacturing processes, and evaluating risks. As reliability prediction plays an increasingly important role in assessing the performance of software systems, intensive studies have been carried out to ensure software reliability. The rest of this paper is organized as follows. Section 2 describes SVR model, expressing it as a combinatorial optimization problem with constraints. Section 3 explains the improved EDA (IEDA) and gives a model of optimizing SVR parameters based on IEDA. In Section 4, we give some assessing methods of the software reliability. Section 5 describes the numerical experiments and the results. Finally, Section 6 shows the conclusions from the experiment results.

2. Support vector regression

The performance of SVR depends on the rational optimization of parameters, and the optimization of these parameters is important to predict accurately. The traditional methods of optimizing parameters are: experience selection method (ESM), gradient descent method (GDM), and Bayesian method (BM). However, these methods have their own disadvantages. For example, ESM requires a large amount of experience and domain knowledge in order to obtain the appropriate parameters, and otherwise it is difficult to obtain the appropriate parameters. GDM is very sensitive to the initial point. In addition, GDM is a linear search method, and it is easy to fall into local minimum. Disadvantage of BM is to need some priori knowledge of parameter space for optimizing parameters, and it also needs more computation and computational complexity. In addition, this technique does not guarantee the outcome of better parameters. In fact, some researches have studied how to apply intelligence method to optimize parameters of SVR [18–20].

Suppose $\{(x_1, d_1), (x_2, d_2), \ldots, (x_n, d_n)\} \subset R^m \times R$ is training set, where R^m is the space of the input features x_i, and d_i is the phenomenon under investigation, i.e., the actual value. In ε-SVR [19], the goal is to find a function $f(x)$ whose deviation from each target d_i is at most ε for all training data, and at the same time, is as "flat" as possible. For the sake of clarity, we consider the following objective function in the linear case, i.e., F: $R^m \to R$, such that

$$y = f(x) = w\phi_i(x) + b \tag{1}$$

where $\phi_i(x)$ is the input features, and w and b are coefficients. The coefficients (w and b) are estimated by minimizing the following regularized risk function:

$$R_{\mathrm{SVR}}(C) = R_{\mathrm{emp}} + \frac{1}{2}\|w\|^2 = C\frac{1}{n}\sum_{i=1}^{n} L_\varepsilon(d_i, y_i) + \frac{1}{2}\|w\|^2 \tag{2}$$

$$L_\varepsilon(d, y) = \begin{cases} \left|d - y\right| - \varepsilon, & \text{if } \left|d - y\right| \geq \varepsilon \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where R_{SVR} and R_{emp} represent the regression and empirical risk, respectively, C and ε are two parameters. In Eq. (2), $L_\varepsilon(d, y)$ is called the ε-insensitive loss function. $\|w\|^2/2$ is used as a measure of the flatness of the function.

Two positive slack variables ξ and ξ^*, which represent the distance from actual values to the corresponding boundary values of ε-tube, are introduced. Then, Eq. (2) is transformed into the following convex optimization problem:

$$\text{Min} \quad R_{\mathrm{SVR}}(w, \xi, \xi^*) = C\sum_{i=1}^{n}(\xi_i + \xi_i^*) + \frac{1}{2}\|w\|^2 \tag{4}$$

$$s.t. \begin{cases} w\phi(x_i) + b_i - d_i \leq \varepsilon + \xi_i^*, \\ d_i - w\phi(x_i) - b_i \leq \varepsilon + \xi_i, \qquad i = 1, 2, \ldots, n \\ \xi_i, \xi_i^* \geq 0. \end{cases} \tag{5}$$

By introducing Lagrange multipliers and exploiting the optimality constraints, the decision function given by Eq. (1) has the following explicit form [21]

$$f(x, \alpha_i, \alpha_i^*) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x, x_i) + b \tag{6}$$

where $K(x_i, x_j)$ is called the kernel function, α_i and α_i^* are the so-called Lagrange multipliers. In Eq. (6), they satisfy the equality $\alpha_i * \alpha_i^* = 0$. α_i and α_i^* are calculated by maximizing the dual function of Eq. (4), and the maximal dual function in Eq. (4), which has the following form:

$$\text{Max} \quad R(\alpha_i, \alpha_i^*) = \sum_{i=1}^{n} d_i(\alpha_i - \alpha_i^*) - \varepsilon\sum_{i=1}^{n}(\alpha_i + \alpha_i^*) - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)K(x_i, x_j) \tag{7}$$

under the constraints, $\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$; $0 \leq \alpha_i \leq C, i = 1, 2, \ldots, n$; $0 \leq \alpha_i^* \leq C, i = 1, 2, \ldots, n$.

The value of the kernel is the inner product of the two vectors x_i and x_j in the feature space $\phi(x_i)$ and $\phi(x_j)$, so $K(x_i, x_j) = \phi(x_i) * \phi(x_j)$.

Any function that satisfies Mercer condition [21] can be used as the kernel function. Generally, the Gaussian function will yield better prediction performance [15]. Thus, in this work, the Gaussian function, $\exp(-\left\|x_i - x_j\right\|^2/2\sigma^2)$, is used in the SVR. Where, σ^2 represents the bandwidth of Gaussian kernel.

So, to build a SVR model efficiently, we need to select three positive parameters ε, σ and C.

3. IEDA and IEDA-SVR model

Although performance of the EDA is better than GA's, the EDA still has drawbacks. For example, in EDA evolutionary process, the individuals in the population are easy to trend to the same solution and the population diversity declines rapidly. These drawbacks affect the performance of the EDA. In order to maintain population diversity, we improve EDA, and obtain the IEDA, and then the IEDA is used to optimize parameters of the SVR.

3.1. Improved EDA

The chaotic sequence has the characteristics of ergodicity, randomness, initial sensitivity and regularity, and the chaotic mutation operation is an important way to maintain population diversity [22,23]. In this paper, the chaotic mutation was introduced into the traditional EDA. IEDA is described in detail as follows.

Algorithm IEDA. IEDA evolutionary process

Step 1. Generate m initial values $x_1, x_2, \ldots, x_m$ randomly in the interval (0, 1). Let $X_0 = (x_1, x_2, \ldots, x_m)$ be an initial point of logistic map, and m is the number of parameters to be optimized.

Step 1.1 Many researchers have demonstrated [24,25] that, for logistic map

$$X_{t+1} = \lambda \cdot X_t(1 - X_t) \tag{8}$$

when the parameter λ satisfying $3.5699 \le \lambda \le 4$, after n times iteration, the generated n individuals $X_1, X_2, \ldots, X_n$ by Eq. (8) is a chaotic sequence. Let $X = \{X_1, X_2, \ldots, X_n\}$ be a matrix with n rows and m columns, representing the initial population:

$$X = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} & \ldots & x_{1m} \\ x_{21} & x_{22} & \ldots & x_{2m} \\ \ldots & \ldots & \ldots & \ldots \\ x_{n1} & x_{n2} & \ldots & x_{nm} \end{pmatrix}$$

Step 1.2 The initial population X is extended to the range of the solution space, and the ith variable of the jth individual can be expressed as follows:

$$x_{ji} = low + (high - low)x_{ji}$$

where *low* and *high* are the lower and upper limits of the solution space, respectively.

Step 2. Calculate fitness value for each individual of the current population. For individual X_j, its fitness value is calculated as follows

$$F(X_j) = \frac{1}{n}\sum_{j}^{n}\sum_{i}^{m}(d_{ji} - y_{ji})^2 \tag{9}$$

where, d_{ji} is the ith actual output value of the jth individual, y_{ji} is the ith predict output value of the jth individual. For Eq. (9), $F(\cdot)$ is as small as possible.

Step 3. Select individuals according to the mutation probability $p_m < 1$ to realize chaotic mutation. We calculate the mutation radius of selected individual. The mutation radius of ith variable x_{ji} of the individual X_j is calculated according to the following equation

$$R_{ji} = \frac{v(i)(1 - F(X_j))}{\max\limits_{1 \le s \le n} F(X_s)}$$

where $v(i) = (2(\max x_{*i} - \min x_{*i})/n)$. And, ith variable of the individual X_j after mutation can be expressed as $x'_{ji} = x_{ji} + 2R_{ji}(1 - x_{ji})$. After mutating each selected individual, we calculate the fitness value F of each new individual. If the fitness value F of new individual is less than F value of old individual, then new individuals replace old individuals, and we can obtain the new population, go to *Step* 4. Otherwise maintain the population not to change, go to *Step* 5.

Step 4. Determine whether to meet the stop criterion of the algorithm, if met, the output current the best individual, otherwise go to *Step* 5.

Step 5. Generate next population.

Step 5.1 Select the first $0.5n$ individuals from the current population. After statistical analysis of selected individuals, build a Gaussian model for each variable of the individual.

Step 5.2 Sample $1.5n$ new individuals accordance to the Gaussian model.

Step 5.3 Calculate F values of these $1.5n$ new individuals, and compare their F values, then choose n better individuals as the next generation population.

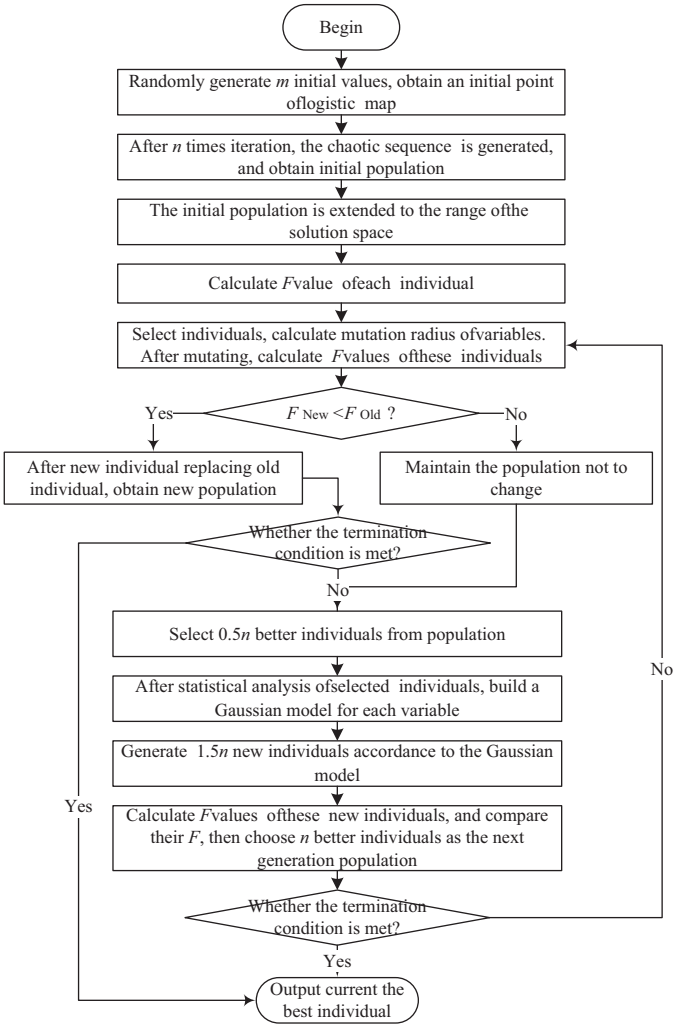

Fig. 1. Framework of IEDA model.

Step 6. Determine whether to meet the stop criterion of the algorithm, if met, the output current the best individual, otherwise go to *Step* 3.

IEDA is illustrated in Fig. 1.

3.2. IEDA-SVR model

Through the IEDA evolutionary process, we can acquire three optimization parameters, which construct optimized SVR model for predicting software reliability.

The prediction procedure based on IEDA-SVR is illustrated in Fig. 2.

3.3. Measure of the population diversity

Hamming distance (HD) is used in this paper since it can describe the difference between individuals. When HD is larger, it means that the difference between individuals is larger, whereas the difference between individuals is smaller. Differences between individuals are calculated as $Ham = \sum_{X_i, X_j \in X} \left| Sgn(X_i - X_j) \right|$.

In this study, we represent an individual using bit strings. Assuming the three parameters ε, σ and C are represented by the three bit strings with equal to lengths. For example, If ε, σ and C are $l_1 = 100 \ldots 010 \ldots 001$, $l_2 = 000 \ldots 110 \ldots 100$, $l_3 = 011 \ldots 010 \ldots 010$ respectively, then the individual is coded into $l_1 l_2 l_3$.

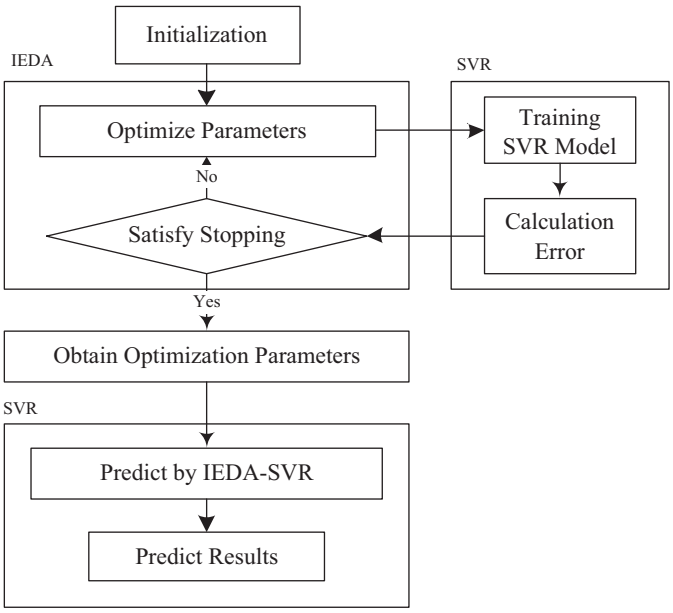

Fig. 2. Framework of IEDA-SVR.

Let p_{ij}^k be the difference value of the individuals X_i and X_j for the kth bit, i.e., $p_{ij}^k = \begin{cases} 1, & f\, x_{ik} \neq x_{jk} \\ 0, & \text{other} \end{cases}$. Therefore, the diversity of population X in the kth bit is $p^k = \sum_{i=1}^{n}\sum_{j=1}^{l} p_{ij}^k$. So, the diversity measure of population X is calculated as follows

$$Div(X) = \frac{\sum_{k=1}^{l} p^k}{n(n-1)l}$$

where l is the bit number of the bit string $l_1 l_2\, l_3$.

4. Assess prediction performance

We use x-fold cross-validation method (CVM) to train the SVR model on the training set. In fact, this is a fitting process which optimizes the parameters of model to make the model fit the training data as well as possible. In x-fold CVM, the original training data is randomly partitioned into x subsets. Of the x subsets, a single subset is retained as the test data for testing the SVR model, and the remaining $x-1$ subsets are used as training data. The CVM is then repeated x times (the *folds*), with each of the x subsets used exactly once as the test data. The x results from the folds then can be averaged to produce a single estimation. The advantage of CVM over repeated random sub-sampling is that all observations are used for both training and test, and each observation is used for testing exactly once [26,27].

In this paper, training process of SVR is built with a given set of parameters $\{\varepsilon, \sigma, C\}$, using $x-1$ subsets as training data. The fitting evaluation of SVR with optimized parameters is calculated by the sum of square error (SSE) or normalized root mean square error (NRMSE) on the test data. This procedure is repeated x times, and each subset is used once for testing. To measure the goodness of fit of IEDA-SVR, we use the correlation coefficient (R). Suppose q is the number of elements in a training data, SSE, and NRMSE, and R represent as follows:

$$\text{SSE}=\sum_{i=1}^{q}(d_i - y_i)^2, \quad \text{NRMSE} = \sqrt{\frac{\sum_{i=1}^{q}(d_i - y_i)^2}{\sum_{i=1}^{q} {d_i}^2}}, \quad R = \frac{Cov(d,y)}{\sigma_d \cdot \sigma_y}.$$

Table 1
Reliability of turbochargers.

i	T_i	i	T_i	i	T_i	i	T_i
1	5.50	7	3.47	13	11.42	19	0.45
2	1.83	8	9.96	14	18.94	20	129.31
3	2.75	9	11.39	15	65.30	21	31.61
4	70.89	10	19.88	16	0.04	22	47.60
5	3.94	11	7.81	17	125.67		
6	14.98	12	14.59	18	82.69		

For simplicity, where d_i is the actual value, y_i is the predict value, and $Cov(d,y) = \frac{1}{q}\sum_{i=1}^{q}(d_i - \mu_d)(d_i - \mu_y)$.

Stopping condition of training SVR is either the training error is less than 10^{-4} or not obtaining an individual with a better estimate value after a certain number L of generations.

5. Numerical examples and experimental results

5.1. Software failure data

In this paper, two numerical examples are used to demonstrate the predicting performance of IEDA-SVR for predicting software reliability. The software inter-failure times taken from a telemetry network system by AT&T Bell Laboratories [28] are used as the first numerical example. Original data is listed in Table 1. The value T_i is the actual inter-failure time and i is the number of the failure.

In the second example, the software failure data obtained from [29] are applied to investigate the prediction performance of IEDA-SVR. Original data is listed in Table 2. The data contains 101 observations of the time series (t, Y_t) pertaining to software failure. Here Y_t represents the failure space time of the software after the tth modification has been made.

5.2. Architecture of IEDA-SVR

For predicting a univariate time series, the inputs used in SVR are the past, lagged observations of the time series, while the outputs are the future values. Fig. 3 illustrates the basic architecture of IEDA-SVR.

In Fig. 3, p denotes the number of lagged variables, which t–p represents the total number of training samples. Following successful training, the SVR can predict future outcomes y_{t+k} at different time steps k. If $k=1$, the prediction is a one-step-ahead predict, while when $k>1$ the prediction is a multi-step predict. Since the accumulation of errors, the performance of multi-step predicting is usually poor [4]. In practice, one-step-ahead predicting results are more useful since they provide timely information for preventive and corrective maintenance plans. Therefore, we consider one-step-ahead predictions in this study.

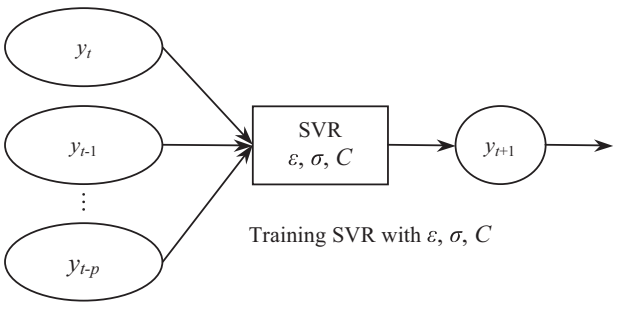

Fig. 3. Basic architecture of IEDA-SVR.

Table 2
Data of software failures.

t	Y_t	t	Y_t	t	Y_t	t	Y_t	t	Y_t	t	Y_t
0	5.7683	17	9.6027	34	10.6301	51	10.3534	68	12.5982	85	12.7206
1	9.5743	18	9.3736	35	8.3333	52	10.0998	69	12.0859	86	14.192
2	9.105	19	8.5869	36	11.315	53	12.6078	70	12.2766	87	11.3704
3	7.9655	20	8.7877	37	9.4871	54	7.1546	71	11.9602	88	12.2021
4	8.6482	21	8.7794	38	8.1391	55	10.0033	72	12.0246	89	12.2793
5	9.9887	22	8.0469	39	8.6713	56	9.8601	73	9.2873	90	11.3667
6	10.1962	23	10.8459	40	6.4615	57	7.8675	74	12.495	91	11.3923
7	11.6399	24	8.7416	41	6.4615	58	10.5757	75	14.5569	92	14.4113
8	11.6275	25	7.5443	42	7.6955	59	10.9294	76	13.3279	93	8.3333
9	6.4922	26	8.5941	43	4.7005	60	10.6604	77	8.9464	94	8.0709
10	7.901	27	11.0399	44	10.0024	61	12.4972	78	14.7824	95	12.2021
11	10.2679	28	10.1196	45	11.0129	62	11.3745	79	14.8969	96	12.7831
12	7.6839	29	10.1786	46	10.8621	63	11.9158	80	12.1399	97	13.1585
13	8.8905	30	5.8944	47	9.4372	64	9.575	81	9.7981	98	12.753
14	9.2933	31	9.546	48	6.6644	65	10.4504	82	12.0907	99	10.3533
15	8.3499	32	9.6197	49	9.2294	66	10.5866	83	13.0977	100	12.4897
16	9.0431	33	10.3852	50	8.9671	67	12.7201	84	13.368		

Table 3
IEDA-SVR parameters.

Population size n	50
Max generations L	500
Chaotic parameter λ	3.859
Mutation probability p_m	0.03
Bit number l	24

5.3. Parameters of IEDA-SVR

After large number of experiments, the parameters of the IEDA-SVR are listed in Table 3. Of course, they were chosen arbitrarily.

5.4. Experimental results

5.4.1. Experimental results of Example 1

In this example, the model with the minimum testing SSE value is chosen as IEDA-SVR. We used 5-fold CVM. Three parameters ε, σ, C of IEDA-SVR with the smallest SSE value is 1.1395, 0.2145, and 7023 respectively.

We let $p=4$. Table 4 compares the predict results of the IEDA-SVR with four other Weibull Bayes predicting models proposed by [28,29]. Moreover, we also direct using traditional EDA to construct predictive model, which is called EDA-SVR. The performances of IEDA-SVR and EDA-SVR also are compared, still see Table 4.

It is shown that IEDA-SVR obtained the better prediction results in terms of the SSE results. Actual and predict results are shown in Fig. 4.

In addition to these experiments, RMSE values (see Eq. (10)) of three computational intelligence models (i.e., multiple regression (MR), feed-forward neural networks (FFN), and SVR) and functional networks (FunNets) [30] are used to describe the performance of these four prediction models, therefore, we compare these four models with IEDA-SVR respectively according to RMSE values. The results are shown in Table 5.

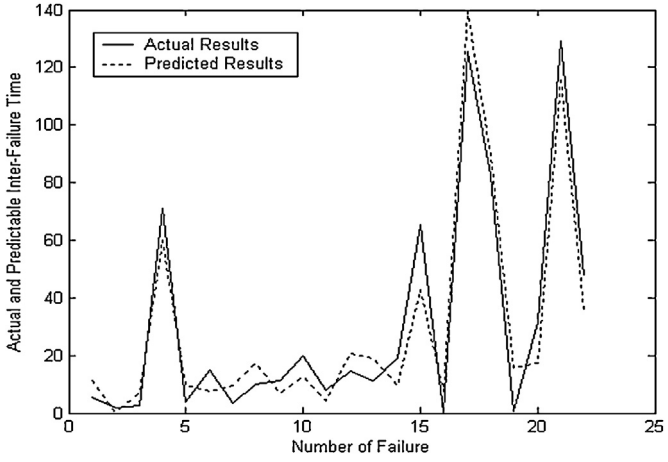

Fig. 4. Prediction and actual results.

$$\mathrm{RMSE} = 100 \cdot \sqrt{\frac{1}{q}\sum_{i=1}^{q}\left(\frac{d_i - y_i}{d_i}\right)^2} \tag{10}$$

In Table 5, the RMSE values of superscript a, b, c, and d are proposed by [29]. From Table 5, it can be observed that the RMSE values of IEDA-SVR are the lowest in all compared models in training and testing sets. The RMSE value of EAD-SVR is also lower than the other compared models. The experimental results indicate that to select SVR parameters using IEAD is feasible. Performance of IEAD-SVR is more satisfactory.

We also compared the goodness of fit of IEDA-SVR using R, mean square error and mean absolute error respectively. These measures can describe the predictive capability of IEDA-SVR from different views. The summary results are shown in Table 6. The R of the

Table 4
Predicting results of different models.

Actual results	IEDA-SVR	EDA-SVR	Weibull Bayes I[a] (hazard function)	Weibull Bayes II[a] (hazard function)	Weibull Bayes III[a] (hazard function)	Weibull Bayes IV[a] (hazard function)
SSE	2582.282	2759.164	16,084.081	16,119.145	9910.025	9611.389

[a] is proposed by [28].

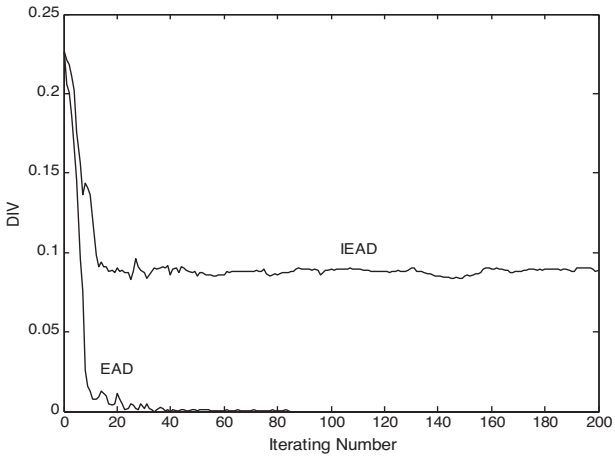

Fig. 5. DIV values of IEAD and EDA.

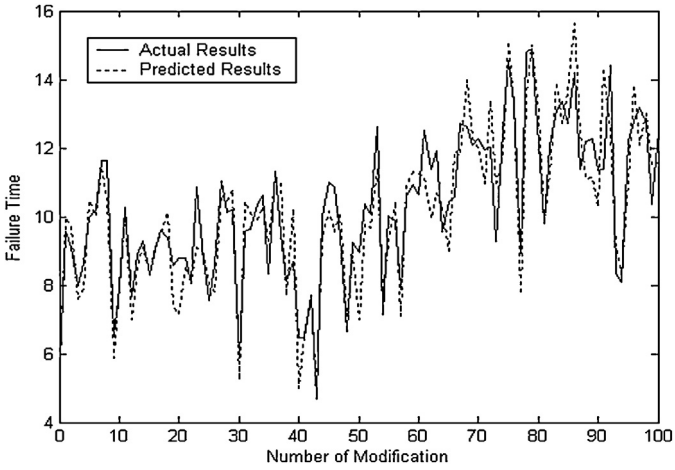

Fig. 6. Predict and actual results.

IEDA-SVR is 0.9621, which represents high correlation between actual and predict results.

The above results not only indicate a close relationship between actual and predict values, but also show that IEDA-SVR has better performance for predicting software reliability.

In addition, we observed difference of the population diversity for IEDA and EDA with DIV. We experiment 20 times. DIV value is the average of 20 times experiments. The experimental results of the 200 iterations are shown in Fig. 5.

From Fig. 5, the DIV value of EAD does not change after less iteration, and the population diversity has lost quickly. For IEAD, the population diversity of IEAD has been better maintained. DIV value of IEAD is higher than EAD's, which shows that IEAD is superior for maintaining population diversity.

For *Example* 1, the above these experiments have shown that to maintain the population diversity can improve the performance of the prediction model.

5.4.2. Experimental results of Example 2

The second numerical example is also used to evaluate the three parameters of SVR and to build IEDA-SVR prediction model. The model with the minimum testing NRMSE in the training set is chosen as IEDA-SVR. We used 5-fold CVM, and $p = 5$. Three parameters ε, σ, C of IEDA-SVR is 0.4376, 65.0738, and 523 respectively. The predict results of IEDA-SVR and actual results were shown in Fig. 6.

We compare the predict results of proposed model with four other models proposed by [31] and EDA-SVR. The compare results are listed in Table 7. It is shown that IEDA-SVR obtained the better prediction results in terms of NRMSEs.

In this experiments, RMSE values of four SVMSA models with different p values and IEDA-SVR, EAD-SVR are compared respectively. The results are shown in Table 8.

In Table 8, the RMSE values of superscript a in Table 8 are proposed by [32]. From Table 8, the RMSE values of IEDA-SVR with different p values are the lowest in training and testing sets. The experimental results show that to select SVR parameters using IEAD is feasible. Performance of IEAD-SVR is more satisfactory.

We also compared the goodness of fit of IEDA-SVR using R, mean square error and mean absolute error respectively. The summary results are shown in Table 9.

The R of IEDA-SVR is 0.9179, which represents high correlation between actual and prediction results. The results not only indicate

Table 5
RMSE values of compared models based on Table1.

Models	Training set	Testing set
MR[a]	51.83	132.29
FFN[b]	29.92	50.47
SVM[c]	28.79	19.56
FunNets[d]	24.00	11.20
EDA-SVR	20.15	9.63
IEDA-SVR	17.35	6.37

Table 6
Experiments results for *Example* 1.

	IEDA-SVR
R	0.9621
Mean square error	2.2743
Mean absolute error	0.6557

Table 8
RMSE values of compared models based on Table2.

p value	Models	Testing set
5	SVMSA[a]	0.1823
	EDA-SVR	0.1084
	IEDA-SVR	0.0806
10	SVMSA[a]	0.1694
	EDA-SVR	0.1251
	IEDA-SVR	0.0884
20	SVMSA[a]	0.1589
	EDA-SVR	0.1283
	IEDA-SVR	0.0960
29	SVMSA[a]	0.1562
	EDA-SVR	0.1381
	IEDA-SVR	0.1062

[a] Proposed by [31].

Table 7
The compare results with other models.

Actual results	IEDA-SVR	EDA-SVR	Model I[a] (normal distribution)	Model II[a] (Kalman filter I)	Model III[a] (Kalman filter II)	Model IV[a] (adaptive Kalman filter)
NRMSE	0.0885	0.1480	0.2095	0.3462	0.2886	0.3237

[a] Proposed by [31].

Table 9
Experiment results for *Example* 2.

	IEDA-SVR
R	0.9179
Mean square error	0.2012
Mean absolute error	0.0848

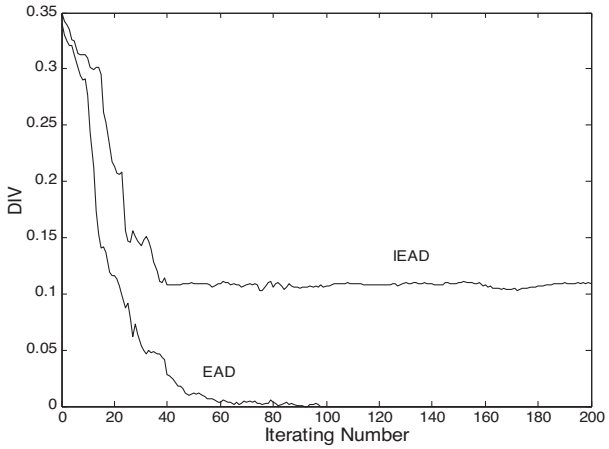

Fig. 7. DIV values of IEAD and EDA.

a close relationship between actual and predict values, but also show that IEDA-SVR model has better performance for predicting software reliability.

We observed difference about population diversity of IEDA and EDA with DIV. We experiment 20 times. DIV value is the average of 20 times experiments. The experimental results of the 200 iterations are shown in Fig. 7.

For *Example* 2, the above these experiments also have shown that to maintain the population diversity can improve the performance of the prediction model.

6. Conclusions

In this paper, we proposed a hybrid IEDA-SVR. In order to maintain the population diversity, the chaos mutation was introduced into traditional EDA. IEDA is used to optimize parameters of SVR, and IEDA-SVR is used to predict software reliability. We used two real software failure datasets in the experiments. We compare prediction performance of proposed model with other various models. The experimental results demonstrate that it is very effective of utilizing IEDA-SVR to predict software reliability, and IEDA-SVR has better prediction performance than the other compared models and a fairly accurate prediction capability. The experimental results also show that to maintain the population diversity can improve the performance of the prediction model. In addition, IEDA-SVR does not require priori knowledge of parameters and as such can give useful results.

The contribution of this paper is as follows:

(1) To maintain the population diversity can improve the performance of EAD, which confirms the software reliability prediction model based on IEAD is effective.
(2) In this paper, a method to optimize the SVR parameters is proposed. The proposed method can avoid the blindness of the SVR parameter selection and provide a technical means for the better use of SVR.
(3) Software reliability prediction is very important for software engineering. In this paper, an intelligent approach is adopted in order to achieve software reliability prediction and obtain good performance.

We note that the sensitivity of predefined parameters is an important issue. Our future work is to analyze the impact of these parameters for performance of software reliability predict model.

Acknowledgments

This work was supported by Natural Social Science Foundation of China (Grant No. 13BTQ050), the social science foundation from Chinese Ministry of Education (Grant No. 11YJAZH040), and the science and technology research program of Wuhan of China (Grant No. 201210121023).

References

[1] V.L. Berardi, P.G. Zhang, An empirical investigation of bias and variance in time-series forecasting: modeling considerations and error evaluation, IEEE Transactions on Neural Networks 14 (3) (2003) 668–679.
[2] D.K. Chaturvedi, M. Mohan, R.K. Singh, P.K. Karla, Improved generalized neuron model for short-term load forecasting, International Journal on Soft Computing-A Fusion of Foundations, Methodologies and Applications 8 (1) (2004) 10–18.
[3] M. Khashei, M. Bijari, An artificial neural network (*p*: *d*, *q*) model for time series forecasting, Expert Systems with Applications 37 (1) (2010) 479–489.
[4] K. Xu, M. Xie, L.C. Tang, S.L. Ho, Application of neural networks in forecasting engine system reliability, Applied Soft Computing 2 (4) (2003) 255–268.
[5] L. Tian, A. Noore, Evolutionary neural network modeling for software cumulative failure time prediction, Reliability Engineering and System Safety 87 (1) (2005) 45–51.
[6] M.C. Liu, T. Sastri, W. Kuo, An exploratory study of a neural network approach for reliability data analysis, Quality and Reliability Engineering International 11 (2) (1995) 107–112.
[7] P.T. Chang, K.P. Lin, P.F. Pai, Hybrid learning fuzzy neural models in forecasting engine system reliability, in: The Fifth Asia Pacific Industrial Engineering and Management Systems Conference, Australia, Gold Coast, 2004, pp. 2361–2366.
[8] I. Steinwart, A. Christmann, Support Vector Machines, New York, Springer-Verlag, 2008.
[9] H.M. Azamathulla, F.C. Wu, Support vector machine approach to for longitudinal dispersion coefficients in streams, Applied Soft Computing 11 (2) (2011) 2902–2905.
[10] A. Guven, A predictive model for pressure fluctuations on sloping channels using support vector machine, International Journal for Numerical Methods in Fluids 66 (11) (2011) 1371–1382.
[11] H.M. Azamathulla, A.A. Ghani, A.C.K. Chang, Z.A. Hassan, N.A. Zakaria, Machine learning approach to predict sediment load—a case study, CLEAN – Soil, Air, Water 38 (10) (2010) 969–976.
[12] V.N. Vapnik, S. Golowich, A. Smola, Support vector method for function approximation, regression estimation and signal processing, in: H Mozer, et al. (Eds.), Advance in Neural Information Processing System, vol. 9, MIT Press, Cambridge, MA, 1997, pp. 281–287.
[13] L.J. Cao, F.E.H. Tay, Financial forecasting using support vector machines, Neural Computing & Applications 10 (2) (2001) 184–192.
[14] L. Wang, J. Zhu, Financial market forecasting using a two-step kernel learning method for the support vector regression, Annals of Operations Research 174 (1) (2010) 103–120.
[15] A.J. Smola, Learning with kernels, Department of Computer Science, Technical University Berlin, Germany, 1998 (PhD thesis).
[16] P. Larranaga, J.A. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer Academic Publishers, Boston, 2001.
[17] J. Zhang, W.L. Lo, H. Chung, Pseudo-coevolutionary genetic algorithms for power electronic circuits optimization, IEEE Transactions on Systems, Man, and Cybernetics, Part C 36 (4) (2006) 590–598.
[18] C.H. Wu, G.H. Tzeng, R.H. Lin, A novel hybrid genetic algorithm for kernel function and parameter optimization in support vector regression, Expert Systems with Applications 36 (3) (2009) 4725–4735.
[19] V. Cherkassky, Y.Q. Ma, Practical selection of SVM parameters and noise estimation for SVM regression, Neural Networks 17 (1) (2004) 113–126.
[20] J.X. Wang, Y. Wang, C. Zhang, W. Du, C.G. Zhou, Y.C. Liang, Parameter selection of support vector regression based on a novel chaotic immune algorithm, in: Fourth International Conference on Innovative Computing, Information and Control, 7–9 December 2009, Kaohsiung, Taiwan, 2009, pp. 652–655.
[21] V.N. Vapnik, Statistical Learning Theory, John Wiley and Sons, New York, 1998.
[22] L.D.S. Coelho, A quantum particle swarm optimizer with chaotic mutation operator, Chaos, Solitons and Fractals 37 (5) (2008) 1409–1418.
[23] H. Handa, The effectiveness of mutation operation in the case of estimation of distribution algorithms, Biosystems 87 (2/3) (2007) 243–251.
[24] L. Kocarev, G. Jakimoski, Logistic map as a block encryption algorithm, Physics Letters A 289 (4/5) (2001) 199–206.

[25] C. Li, S. Li, G. Chen, L. Hu, Cryptanalysis of a new signal security system for multimedia data transmission, EURASIP Journal on Applied Signal Processing 2005 (8) (2005) 1277–1288.
[26] P. Richard, C. Dennis, Cross-validation of regression models, Journal of the American Statistical Association 79 (387) (1984) 575–583.
[27] K. Duan, S. Keerthi, A. Poo, Evaluation of simple performance measures for tuning SVM hyperparameters, Neurocompuing 51 (2003) 41–59.
[28] L. Pham, H. Pham, Software reliability models with time-dependent hazard function based on Bayesian approach, IEEE Transactions on Systems, Man, and Cybernetics, Part A 30 (1) (2000) 25–35.
[29] L. Pham, H. Pham, A Bayesian predictive software reliability model with pseudo-failures, IEEE Transactions on Systems, Man, and Cybernetics, Part A 31 (3) (2001) 233–238.
[30] E.A. Sebakhy, Software reliability identification using functional networks: a comparative study, Expect Systems with Applications 36 (2) (2009) 4013–4020.
[31] N.D. Singpurwalla, R. Soyer, Assessing (software) reliability growth using a random coefficient autoregressive process and its ramifications, IEEE Transactions on Software Engineering SE-11 (1985) 1456–1464.
[32] P.F. Pai, W.C. Hong, Software reliability forecasting by support vector machines with simulated annealing algorithms, Journal of Systems and Software 79 (6) (2006) 747–755.