Software Fault Prediction Model Based on Adaptive Dynamical and Median Particle Swarm Optimization

Cong Jin, En-Mei Dong, Li-Na Qin
Department of Computer Science,
Centual China Normal University,
Wuhan 430079, P.R.China
jincong@mail.ccnu.edu.cn

***Abstract*—Software quality prediction can play a role of importance in software management, and thus in improve the quality of software systems. By mining software with data mining technique, predictive models can be induced that software managers the insights they need to tackle these quality problems in an efficient way. This paper deals with the adaptive dynamic and median particle swarm optimization (ADMPSO) based on the PSO classification technique. ADMPSO can act as a valid data mining technique to predict erroneous software modules. The predictive model in this paper extracts the relationship rules of software quality and metrics. Information entropy approach is applied to simplify the extraction rule set. The empirical result shows that this method set of rules can be streamlined and the forecast accuracy can be improved.**

Keywords- Data mining; Classification; Fault Prediction; ADMPSO; Rule simplification

I. INTRODUCTION

The modeling technique of software defect prediction is the critical technology in the software quality prediction systems. Studies have shown that the software fault prediction can find the relationship between software metrics and quality, and help developers focus mainly on the defect-prone modules, thus reduce development and maintenance cost. Accordingly the effective software fault prediction can also improve reliability and quantity of the software.

Data mining means the extraction of hidden predictive information from large databases; it is a powerful new technology with great potential to help peoples focus on the most important information in their data warehouses. Data mining techniques have been successfully applied in many different domains. Well-known examples are market basket analysis in the retail, breast-cancer detection in the biomedical sector. Recently, however, researchers began applying data mining techniques on software quality prediction so as to improve the development of software.

This paper mainly focuses on create defect prediction models in software modules by using of the data mining techniques. The model applies the adaptive dynamic and median particle swarm optimization (ADMPSO) to extract the relationship rules between the decision and metrics attribute of the software, and utilized their entropy of information to reduce the relationship rules.

This paper is structured as follows. First, the importance of software prediction is depicted in Section 1, and then the ADMPSO is described in Section 2. In Section 3 the ADMPSO model is analyzed and experiment results are shown in Section 4. The final Section concludes the paper.

II. RELATED WORDS

A. Standard PSO

The PSO[1] algorithm was initially introduced by Kennedy and Eberhart in 1995 based on the evolutionary programming and the genetic algorithms, it is an algorithm which modeled on swarm intelligence that finds a solution to an optimization problem in a search space, or modeled and predicted social behavior in the presence of objectives. The particle position and velocity equations form are given by:

$$v_{id} = \omega \cdot v_{id}^{'} + c_1 rand(t)(p_{id} - x_{id}) + c_2 rand(t)(g - x_{id}) . \quad (1)$$

$$x_{id} = x_{id} + v_{id} . \quad (2)$$

where c_1 and c_2 are constants, $rand(t)$ is random numbers. Changing velocity enables the particle to search the best individual position and the best global position of itself.

At each update step of PSO, the velocity of each particle is calculated according to (1) and the position is updated according to (2). Generally speaking, a maximum velocity vector is defined in order to control the v_i . Wherever v_{id} exceeds the defined limit, its velocity will be set to be $v_{\max}$. If a particle finds a better position than the previously best position, it will be stored in the memory. The algorithm goes on until a satisfactory solution is found or the predefined number of iterations is met.

B. ADMPSO

During the search process, some particles do not move for several iterations. The reason is usually that the best particle has fallen into a local minimum or the velocity of the particle has been reduced to a very small value. To ensure the diversity of response between the individual and group values, measures must be taken to overcome this problem to prevent premature convergence. We try to apply the average Pbest position of particles to generate the current Pbest position of every particle [2]. The particle position update equation form is given by (3):

978-0-7695-4008-5/10 $26.00 © 2010 IEEE
DOI 10.1109/MMIT.2010.10

$$v_{id} = \omega \cdot v_{id}^{'} + c_1 rand(t)(p_{id,ave} - x_{id}) + c_2 rand(t)(g - x_{id}) \quad . (3)$$

where $p_{id,ave} = (p_{i,1} + p_{i,2} + \Lambda + p_{i,n})/n$ is the particle median position of the passing particles. Then the particle is not searched between the Pbest and the Gbest but between the Gbest and the central of the Pbest.

ω is the inertia weight, if it is large, the particle tends to keep its current search direction for a longer time and therefore to prevent premature convergence and increase the chance of finding out the global solution. So by inducting rate of particle evolution (h) and converging factor (s), inertia weight can be changed dynamically [3], which also leads to a reduction in the number of iterations. We have defined this equation in (4).

$$\omega = \omega - h \cdot \omega_h + s \cdot \omega_s \, . \qquad (4)$$

and where $h = \min(f(g_{T-1}), f(g_T)) / \max(f(g_{T-1}), f(g_{Tr}))$, the smaller the value of h, the faster the tempo of evolution. $s = \min(f(g_T), \overline{f_T}) / \max(f(g_T), \overline{f_r})$, $\overline{f_T}$ is the average value of the fitness function; s can reflect the multiplicity of the particle.

The ADMPSO algorithm has a high probability of finding local and the global optimum solution for the problems, and enhances the capability of the PSO algorithm to convergence for complicated multimodal problems in this paper. Firstly, we use the uniform design to determine the locations of the initial particles in the design space, and that ensures the distribution of particles in the whole design space. Using the median value of particles to generate the velocity can avoid the particle being trapped in local minima. Secondly, the ADMPSO algorithm can be used to change the velocity weighting factor and restrain the maximum velocity automatically in order to switch the search from the global search to the local search and thus accelerating the convergence. Then the ADPOS can be ensured that the search area is increased and capacity is optimized.

Algorithm1. Pseudo-code of ADMPSO algorithm
Input: datasets
Output: exacted rules
While the training set is empty
I) Initialize particle
II) Do:
For each particle:
1) Calculate fitness value
2) If the fitness value is better than the best fitness value (pbest) in history, update the pbest and the gbest
3) Calculate particle velocity according to the velocity equation (3)
4) Update particle position according to the position equation (2)
End
While maximum iterations or minimum error criteria is not attained
III) Extract rules corresponding to the optimized particle
End while

III. THE RESEARCH MODEL OF PROPOSED METHOD

In this Section, the model is described, which is combining the features of the ADMPSO and the Information entropy. Figure 1 shows the research model of proposed method.

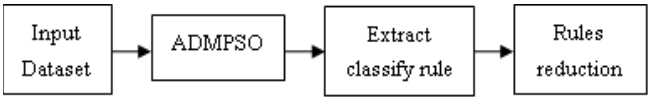

Figure 1. Predict Model.

A. Fitness Function

The fitness value and candidate solutions to the problem are employed to act as particles or individuals, and each of which adjusts its flying based on the flying experiences of itself and its companion. It keeps track of its coordinates in hyperspace that associated with its previous best fitness solution, and also of its counterpart corresponding to the overall best value acquired from any other particle in the population.

To acquire an optimal combination, it should be noted that choosing proper fitness function is very important in synthesis procedure. For our optimization problem, different fitness functions promote different PSO behaviors. The bigger the fitness function value, the more precise of predict. The fitness function is proposed as following:

$$fit(x_i) = (\alpha_1 \cdot nfp_i / nfp) \times (\alpha_2 \cdot fp_i / fp) \, . \qquad (5)$$

where nfp_i is the non Fault-Prone(NFP) covered by x_i, fp_i is the Fault-Prone(FP) model covered by x_i. nfp_i and nfp respectively stand of NFP and FP. α_1 and α_2 are weighting parameters which are constrained in the range of (0,1) and $\alpha_1 > \alpha_2$.

B. Extracted Rule Reduction

Rule reduction is one of the most important steps in the proposed model. It is a process of crossing out the dispensable rules in the classify system while keeping the capability of classifying. This process provides possibility for us to extract rules and provides us with supports of making decision. In this paper, we apply the information entropy to simplify rules. The bigger the conventional entropy value, the higher the rule priority. This method extracts rule which has the fewest condition or contains the most core-value in the rule core-set. We can gain shortest and discredited rules based on the selecting policy. The estimated entropy of the condition and decision attribute is calculated as follows:

$$I(a,c) = \sum_i p(c_i \mid a_i) \log_2 (p(c \mid a_i) / p(c)) \, . \qquad (6)$$

where a is the condition attribute, c is the decision attribute. $p(c_i \mid a_i)$ is the conditional probability of a. If

$I(a,c)$ is too small, we will delete the rule. In this paper, threshold value of $I(a,c)$ is set at 0.3.

C. Pattern Matching

Pattern matching is the act of checking whether the test data and the constituents of a given pattern are matched or not. This matching process concludes the testing of whether the test data have a desired structure, the finding of relevant rule, and the retrieving of the aligning parts. The fitness function ($f(r)$) and the weights pattern is described in this paper. The weight of condition attribute in the test data is computed based on the dependence of decision attribute. If the pattern matching value is bigger and the test sample is closer to the rule, then the correct rate is higher. The algorithm of pattern matching technique is listed in the following:

Step 1: Sorting $f(r)$ of each rule;

Step 2: The matching value of each rule and the test data is calculated by (7) ;

Step 3: According to the value of $M(r,a)$, classify the test sample

$$M(r,a) = f(r) / \sum_i a_i \quad (7)$$

where a is the dependence weigh. Details of the dependence weight algorithm can be found in [4]. Use $M(r,a)$ as the basis of pattern matching, we can obtain the better correct rate.

IV. Empirical evaluations

This section discusses the conducted empirical study that evaluates the capability of the proposed model in predicting defect-prone software modules.

A. Datasets

Our experiment used data sets from two NASA software projects, labeled CM1 and KC1. CM1 is written in C Code with approximately 20 KSLOC. KC1 describes a 43 KSLOC software project written in C++ and is a subsystem of a large ground control system. The independent variables are 21 static metrics at the module-level including McCabe, Halstead (basic and derived), Line Count and Branch Count. The number of software modules and Boolean variables (Whether the module has any defects or not) for each of these datasets and the number of faulty and non-faulty modules are listed in Table 1.

TABLE I. Dataset Properties

	CM1	KC1
Size	496	1571
FP	48	1252
NFP	448	319

B. Data Preprocessing

Since some independent variables might be highly correlated, discretization based feature selection for SVM [2] which is a correlation-based feature selection technique, was applied to simplify selection of the best predictors from 21independent variables in the datasets. In this manner, The level of consistency, coined from the rough sets theory, is introduced to measure the information for classification loss during discretization so that irrelative or redundant attributes are eliminated while the necessary information for classification is preserved. The retained attributes are listed in Table 2.

In the retained attributes the span of data value is great, such as the value of LOComment and V(*g*), the former is [0, 48], and the latter is [1,136]. The greater the span of data values, the larger the particles search space. To solve this problem efficiently and to attain more powerful search, we went through mapping input data into interval [0, 1] using a mapping function m in the form of $x \alpha\ m(x)$, where $m = (m - m_{\min})/(m_{\max} - m_{\min})$ is the value map. Then the particle will search in a hypercube space with radius of 1.

In search case, because the value of particle position that calculated by equation (2) may be out of range [0, 1], we apply the following equation (8) to confine this value.

$$x_{id} = \begin{cases} 0, if x_{id} < 0 \\ 1, if x_{id} > 1 \end{cases} \quad (8)$$

TABLE II. The Metrics Retained After Input Selection, For The Different Datasets.

CM1	Definition	KC1	Definition
LOC	Total lines of code	*V*	Volume on minimal
IV(g)	Design Complexity	*D*	Difficulty = 1/L
D	Difficulty = 1/L	*V(g)*	Cyclomatic Complexity
I	Intelligent count	*LOCode*	Number of lines of statement
LOComment	Number of lines of comment	*LOComment*	Number of lines of comment
LOBlank	Number of lines of blank	*LOBlank*	Number of lines of blank
		UniqOpnd	Number of Unique operands

C. Experimental Setup and Result

A 10-fold cross-validation [5] was used to evaluate the performance of the prediction models. Each dataset was randomly partitioned into 10 bins of equal size. For 10 times, 9 bins were picked to train the models and the remaining bin was used to test them, each time leaving out a different bin. This cross-validation process was run 50 times, using different randomization seed values for cross-validation shuffling to ensure low bias. We then computed the mean value and the standard deviation of each software project over these 50 different runs. As for the parameters of experiment, the inertia weight was initialized to be 0.9, the value of c1, c2 were set at 2, the maximum number of iterations was set at 1000, the total number of particles was set at 50 and the maximum speed v_{max} of the particle was set at 4.0.

To compare the results of ADMPSO, the normal PSO is performed. The result of comparison is shown in Table 3.

TABLE III. TEST RESULTS OF TWO ALGORITHMS.

	CM1		KC1	
	ADMPSO	PSO	ADMPSO	PSO
The mean	96.33%	92.55%	97.2%	92.79%
The deviation	±0.019	±0.024	±0.016	±0.029

V. CONCLUSION

The possibility to predict faults in software modules can be very valuable for software management. The quality of the software that obtained in a more efficient way improves could make efforts focused mainly on fault-prone software modules that were found by the classification during the testing phase. In this paper, we proposed the adaptive dynamic and median particle swarm optimization model to predict error-prone modules. The capability of ADMPSO in predicting defect-prone software modules and its prediction performance in the context of two NASA datasets shows the advantages in several ways. First, the dimensionality of data was reduced by the discretization based feature selection for SVM, and the selected data redundancy is lower. Second, in the searching process, the median optimum particle replaced individual optimum particle, thus maintained the individual diversity and avoided the degenerate phenomenon. Finally, the search space is constrained to range [0, 1], therefore reduced the search space. Empirical result of the proposed model demonstrated the effectiveness of our approach.

Acknowledgements

This work was financially supported by self-determined research funds of CCNU from the colleges' basic research and operation of MOE (Grant No.CCNU09Y01013), the Natural Science Foundation of Hubei Province of China (Grant No. 2008CDB349), the Key Project from Chinese Ministry of Education (Grant No.108166), and the State Key Laboratory of Software Engineering of Wuhan University (SKLSE) (Grant No.SKLSE3008-07-05).

References

[1] Kennedy J, Eberhart R, and Shi Y, Swarm Intelligence, Morgan Kaufman Publishers, San Francisco, 2001.

[2] Wang Cunlui, Duan Xiaodong, Liu xiangdong, and Zhou Fucai. The median particle swarm optimization algorithm. The second session of the Fifth Annual Conference-cum-Youth. 714-718, 2005.

[3] Zhang Xuanping, Du Yuping, Qin Guoqiang, and Qin Zheng. Adaptive Particle Swarm Algorithm with Dynamically Changing Inertia Weight. Journal of Xi'an Jiaotong University. 10: 1039-1042, 2005.

[4] Zhong Bo, Xiao Zhi, and Zhou Jiaqi. Determination to Weighting Coefficient of Combination Forecast Based on Rough Set Theory. Journal of Chongqing University (Natural Science Edition). 07: 127-130, 2002.

[5] Kohavi, R.. A study of cross-validation and bootstrap for accuracy estimation and model selection. Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI). 1137–1143 , 1995.