IJARCCE

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

A Hybrid Approach for Software Fault Prediction Using Artificial Neural Network and Simplified Swarm Optimization

Ankit Pahal[1] and R. S. Chillar[2]

Department of Computer Science & Application, M.D. University, Rohtak, Haryana, India[1,2]

Abstract: The major task in the software developing is to provide a software which is free from any kind of defects. But this task is hard to accomplish by the developers. Fault prediction can be classified as one main region to forecast the possibility of the software containing faults. The aim of the fault prediction in software development life cycle is to categorize the software modules in fault-prone and non fault-prone modules as soon as possible. This classification of fault-proneness of a module is actually essential for reducing the cost and increasing the efficiency of the software development process. In this paper, we propose a hybrid model using artificial neural network (ANN) and Simplified Swarm Optimization (SSO) for fault prediction. ANN is used for categorization the software modules in fault-prone and non-fault-prone modules, and SSO is then used to reduce dimensionality of dataset. This approach is easy to implement as no expert knowledge is required. The attained results confirms a preferred performance of this approach for fault prediction and output rate or recognition. The results indicates the prediction rates of proposed method is more than 90 percent in best condition.

Keywords: Software fault proneness, fault prediction, artificial neural network, simplified swarm optimization.

I. INTRODUCTION

Presently, software plays a significant share in numerous areas; consequently software testing [1] is also a necessary task. Though, with the growth of the software business, software are getting larger and larger in size, so it becomes a costly task and consumes a lot of effort and time in the software development. Since we rely on the software systems very much in our day-to-day lives, software faults can effect extremely and even lethally, particularly with the high risk systems. To avoid this situation, software modules' potential faultiness prediction through the development cycle will be a much advantage for scheduling activities. As the studies demonstrate that the most common faults are frequently found in merely a little software modules [2], so developers must need to concentrate on these fault-prone software modules. Alternatively, it is also desired to procedure designs of non fault-proneness modules. These complications can be resolved by using the existing historical data & extracting knowledge and building a model for the prediction of fault-proneness which can be used in future developments. Even though software is developed in accordance with the standard procedures, still, software quality can be affected by many factors. Presently, the main objective of software industry is to make software systems of high quality and eliminate possible software faults.

The objective of software fault prediction is to classify the fault-prone software modules before the testing phase using certain primary characteristics of the software system. Subsequently, this assist in proficient and cost-effective allocation of testing resources. A lot of research to construct and assess the fault prediction models for the fault proneness prediction of the software modules has been done in the past. Regrettably, software faults cannot be easily measured, though it can be assessed through software metrics. Many studies shows first- hand proof that associations occur in certain software metrics and fault-proneness [3]. Classification of software systems with fault-proneness is usually realized with the help of binary classifiers that predict if a module contains fault or not using several software metrics. Initial methods of prediction for software fault-proneness were built using statistics, though the prediction efficiency was insufficient of these methods. For this reason, most modern studies introduce the machine learning systems comprising data mining [4], SVMs [5], ANN [6], naive Bayes algorithm [7], and fuzzy logic, etc. While software faults were explored using these methods, yet there are numerous characteristics of faults continuing vague. We observed that the associations among software metrics and fault-proneness are of tenintricate and nonlinear, the suitability of outdated linear models is conceded, that effects in the building of non-linear models, and higher performance as compared to linear models is expected.

1.1 Artificial neural network

ANN is a dominating supervised learning technique. It pretends the construction of the human brain with the help of artificial neurons network. The two key components of network structure are neurons and weighted-directed relations, which connect one layer of neurons to another

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

layer of neurons (Fig. 1).In the training phase, certain weights of the connections are adjusted. ANN models, without any contribution, can be trained for these features from sample data and this information can be used to predict or categorize data in a dataset. Since ANN executes its job by means of a black box, it is difficult to understand the ANN models. A significant benefit of ANN is that they can resist discrepancy or omitted values in datasets. Also ANN is proficient of understanding complex non-linear input and output conversions, and therefore, is very useful for modeling of software fault-prediction.

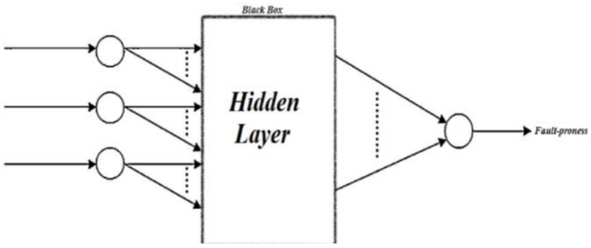

Fig. 1: A general structure of ANN.

Commonly, the efficiency of ANN relies on a suitable choice of the best appropriate input variables since certain inappropriate &repeated input variables normally occur in the input dataset which can make prediction tougher and reduce simplification performance of ANN.

1.2 Simplified swarm optimization

Only the software metrics are not sufficient for training the neural network for predicting fault-proneness and we must reduce dimensionality of input dataset. Furthermore, every software metrics effectthe software fault-proneness predictiondifferently, and certain metrics effect the evaluation of software fault-proneness prediction slightly. Consequently, we musteliminate themetrics which have alittle impact for decreasingbiased input dataset and increase the effectiveness of the prediction model.A sequence of comparatively noveloptimization algorithm is theSwarm intelligence optimization comprising bee colony optimization [8],particle swarm optimization (PSO) [9] andant colony optimization [10]etc.They simulates the swarm behavior of thepopulation of individuals, using the exchange of information and teamworkofswarms to attain the optimization. In 1995, Eberhart and Kennedy developed a population-based optimization technique known as PSO, althoughPSO is a simple algorithm as compared tothe otheralgorithms of swarm intelligenceowing to thebetter convergence rate andless control parameters. Nonlinear complex optimization problems are primarily solved using PSO. Furthermore, it uses the characteristics of the data itself anddo not require any assumptions for software datasetsin the implementation procedure of PSO. As a result PSO appealsseveralresearchers and has arose as the best tool for optimization problems. But PSO likewise has particulardrawbacksfor examplelow convergence rate in the later phases of PSO, poorer convergence efficiency, generally has three parameters and fall into local minima easily etc. Therefore, to overcome the drawbacks of PSO, a new Simplified Swarm Optimization (SSO) [11] is proposed that has superior global search ability. Hence, in this paper, SSO is used for reducing the dimensionality of input dataset and finds certain metrics from the optimal solution of SSO.

Step 1. Initialize the variables.
Step 2. Generate and initialize **pbest** and **gbest** on a random position **x**.
Step 3. Evaluate fitness value for each particle.
Step 4. Update **pbest** and **gbest**.
Step 5. Generate a random no. and check if better **pbest** or **gbest** value is obtained.
Step 6. Repeat until termination criteria is met.
Step 7. End

Fig. 2. A general SSO algorithm.

II. EXPERIMENTAL METHODOLOGY

2.1. Model methodology

To develop the software fault-proneness prediction method we have used the ANN and SSO hybrid approach.This is well-known that, ANN has remainedextensivelyfunctional in pattern recognition because of the reason that ANN-founded classifiers are able toincludestructural andstatistical information together and accomplishsuperior performance as compared to the modest minimum distance classifiers [12], are also broadly used in soft computing. Presently,the widely used ANN model is a feed-forward multi-layer neuralnetwork which is based on error back propagation (BP) algorithm. The neural network model consist of an input layerand an output layer, and also one or more hidden layer in between. The neighboring layers accomplish full connectivitybetween neurons, though no connection is present between neurons withina layer. We used a three-layer networkANN in this paper. Furthermore, SSO is a universal convergence assured search method thatpresents the Exchange Local Search (ELS) into the traditional PSO. SSO outperforms traditional PSO on many problems instances. Therefore, SSO is used for reducing dimensionality. In this paper, on the basis of hybrid ANN and SSO, an improved software fault-proneness prediction methodis suggested. The block diagram ofrecommended softwarefault-prone prediction method is presented in Fig. 3.

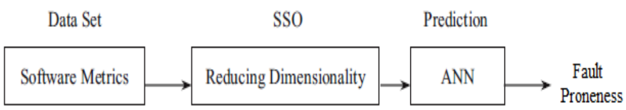

Fig. 3 Block diagram of proposed approach.

2.2. Software metrics

Severalstudies nowadaystake software metrics as important parameter to recognizethemodules which are

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

fault-prone and their studiesrevealed that software metrics are actually useful in predicting the fault-pronemodules. Previous research suggestedvarious software product metrics containing both static and dynamic for the prediction of fault proneness and determining the testing persistence. Code structure measurement, a static metric, is used to recognize software complexity, for examplethe Halstead's Software Science [13] and the McCabe's Cyclomaticnumber. Whereas Dynamic metrics measure the testing persistence using structural and data flow coverage.In the paper, McCabe [14], Halstead [15] metrics, etc. are used for fault-proneness prediction. Figure 4describesexplanation of selected metrics.In this paper, each software module is denoted by 21 metrics for software fault-pronenessprediction in experiments.They all are eminent software metrics in the software fault-pronenessprediction perspective and consequently we do not offer the explanation of these metrics. The detail of the metrics can be found in [16].

No.	Software Metric	Symbol
1	Number of blank lines	LOCb
2	Number of lines which contain both code and comment	LOCec
3	Total number of lines	LOC
4	Number of branches	BR
5	Number of unique operators	n_1
6	Total number of operators	N_1
7	Number of unique operands	n_2
8	Total number of operands	N_2
9	McCabe cyclomatic complexity.	$v(G)$
10	McCabe design complexity.	$iv(G)$
11	Halstead program length, $N = N_1 + N_2$	N
12	Halstead program difficulty, $D = (n_1/2) \times (N_2/n_2)$	D
13	Halstead program level, $L = 1/D$	L
14	Halstead program effort, $E = D \times V$	E

Fig. 4 Selected Metrics.

2.3. Data standardization

In the ANN, the input dataset is (x(i), y(i)) (i = 1, 2, . . ., q), where x(i)∈ Rdis software metric valuesvectorthat enumerate the metric ofthe ith class, and the total number of data samples is denoted by q. the output neuron y(i) of ith class is expected to give outputvalue "1" conforming the fault-proneness and "0" conforming the non fault-proneness. Each input to the similar range is usually normalize while training the ANN.

Performance of the training process is thus enhanced, ensuring the equality of each initial input asimportant.In the case of software metrics the upper bound is typicallyunrestricted in value range. Therefore, so as to normalize it is essential to attain upper andlower bounds of thevalue range of software metrics.According to datasetswe can obtain thevalue of every metric for definite datasets and software metrics. The minimumand maximum values can be obtain easily as the value of every single metric has been givenin these datasets. Letthe minimum and maximum values be min(x(j)) and max(x(j))respectively ofthe jthsoftware metric in the dataset for each software metric. Then the scaledvalue X(j)is

$$X^{(j)} = \frac{x^{(j)} - \min(x^{(j)})}{\max(x^{(j)}) - \min(x^{(j)})}$$

Therefore, everynoted value is drawn to the closed interval [0,1]. (X(i), y(i)) (i = 1, 2, . . ., q) is the normalized dataset and X(i)∈ Rd is the normalized metrics vector.

2.4. Reducing dimensionality

Suppose thata dataset, D = (S, M) where M = {m_1, . . ., m_d} is d metrics and S = {S_1, . . ., S_q} is q samples sets, respectively. C = {c_1, . . .,c_t} denotes type set.We convert the solution X of SSO intoa binary string to reduce dimensionality, as the following operation: In place ofsomeknown random number, rand, in the interval [0,1],

$$\text{if}\left(\text{rand} < S(\mathrm{x})\right)\text{then X} = 1, \text{else X} = 0$$

where$S(\mathrm{x}) = \frac{1}{(1+e^{-x})}$ is sigmoidal function.In recommended reducing dimensionality method, a binary (0 or 1) stringcharacterizedthe position ofparticle i. Anda selected metric is denoted by 1, whereas a non-selected metric is denoted by 0. In SSO, whenwe get the absolute solution Pglobal, the corresponding metricisfollowedwith respect to Pglobal position. Assume M_1⊂ M is the final selected metric,and let l <d bethe total selected metrics.

III. PROPOSED ALGORITHM

As the input & output dataset for ANNare normalized thusthe value will be in the range [0,1]. ANN is considered for a representation of "0" or "1"from the data space. Theselected metrics set M_1represents the number of input neurons ofANN and there is a single output neuron in the proposed fault prediction method. So we can obtain the prediction algorithm as follows.

Step 1. The input metrics Xis a normalized metric which constantly lies in the range [0,1].

Step 2. The dataset is divided into testing and training subset randomly.

Step 3. ANN is modelled on the training subset & trained ANN is obtained.

Step 4. The dimensionality of M is reduced using SSO to obtain M_1 and the input dataset X is reduced to X'.

Step 5. On the basis of new reduced dataset, trained ANN is developed.

Step 6. Fault-proneness module is predicted using the ANN.

IV. RESULTS & DISCUSSIONS

In this study, four projectsfrom the NASA repository are used as datasets that are openlyavailable from the NASA Metrics Data Program [17]. Two of the selected datasets are PC1 and JM1, which are implemented in C language where a function is considered as module. The further two selected datasetsare KC1 and KC3 are implemented inC++ and Java languages respectively where a method is module in this instance. Everydataset hastheir software metrics and the associated variable that tells if the modulehas any fault-pronenessor not. The modules that havev(G) > 10 in the four selected datasets, classify to be fault-proneness as

ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

stated by the regularMcCabes rules. The main characteristicsof the datasets are shown in Table1.

Project	# of modules	# of faulty modules
PC1	1107	76
JM1	10,878	2102
KC1	2107	265
KC3	458	43

Table 1. Selected datasets.

This approachoperates based on the number of incorrect or correct answers.The data in the Confusion matrix demonstrates the performance of the proposedalgorithm for two-class problem[18] that is shown in Table 2.

Error in module		Reality	
		Positive	Negative
Forecast	Positive	True positives (TP)	False-positives (FP)
	Negative	False- negatives (FN)	True-negatives (TN)

Table 2. Confusion Matrix

The accuracy of the proposed software fault-proneness detection approach is calculated as

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

The results of simulation of software fault-proneness detection approach are provided, using MATLAB software and ANN toolkit. The parameters used to determinethe performance of the proposed method are shown in Fig 5.

Parameter	Value
Number of particles Q	60
Max iteration Gmax	100
In SSO, the unique parameter, α	0.55
In PSO, inertia weight w	0.45
In PSO, acclrn constant $c_1 = c_2$	2

Fig. 5 Selected Parameters

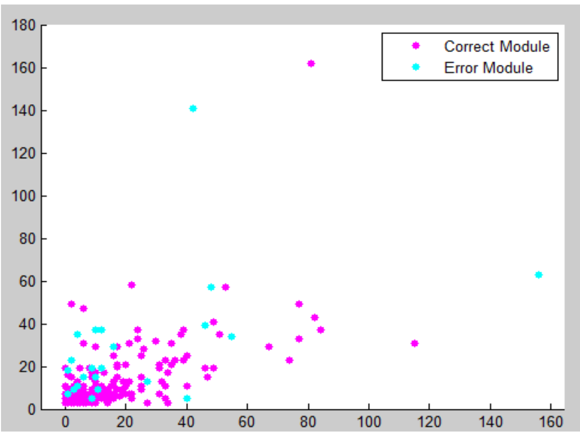

Fig. 7 Detection of fault-prone modules in the PC1 data set

The results demonstration accuracy of the proposed approach's performance is 90.08 %.

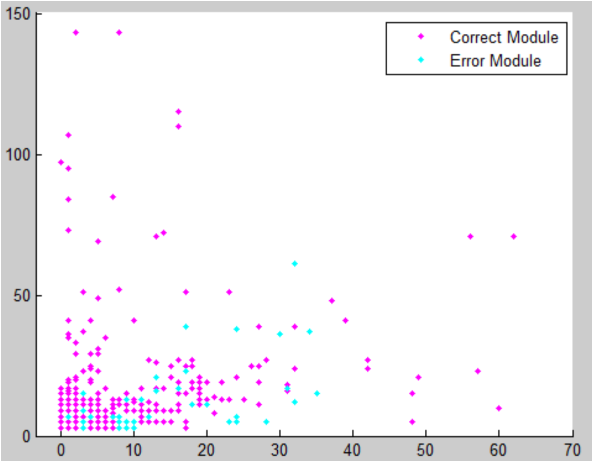

Fig. 8 Detection of fault-prone modules in the JM1 data set

The results demonstration accuracy of the proposed approach's performance is 86.06 %.

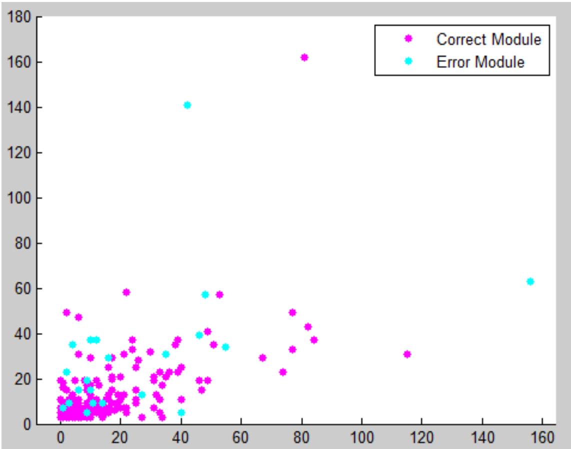

Fig. 9 Detection of fault-prone modules in the KC1 data set

The results demonstration accuracy of the proposed approach's performance is 92.03 %.

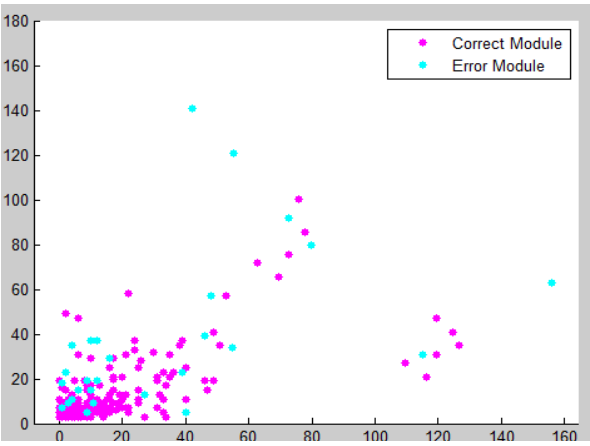

Fig. 10 Detection of fault-prone modules in the KC3 data set

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 3, March 2017

The results demonstration accuracy of the proposed approach's performance is 89.13 %.

V. CONCLUSIONS

This paper examined the usage of hybrid ANN and SSOto improve the software fault-proneness prediction method.The presentedmethod can predict the software modules' fault-pronenesssimply using software metrics. The keycharacteristicsof presented prediction methodisthat dimensionality of themetrics is reduced using SSO and software modules' fault-proneness is predicted using ANN. The reduction dimensionality method is actually effectivesincethere are 3 control parameters in PSO, whereasSSO have single parameter only, consequently SSO performs accurately. Results from experimentsensure the simplifying dimensionalityprocedure which rely on SSO can reduce ANN model. The hybrid model of ANN & SSO has superior performance than currentbest other prediction methodologies and also it has verified to be muchoperational for foundingassociation between fault-proneness and software metrics. In the fault-proneness of software modules,the non-selected metrics from metric space have less significance over the few selected metrics.Therefore instead of all metrics, developers must concentrate on these selected metrics in thesoftware development process.The results, at best, recommend a greater detection rate of the proposed approach higher than 90%.

The software fault-proneness prediction work should be further improved by using combination of other evolutionary machine learning algorithms for discovering the most significant feature for fault-proneness prediction and obtaining the susceptible modules and metrics.

REFERENCES

[1] A.K. Pandey, N.K. Goyal, Prediction and ranking of fault-prone software modulesSpringer Series: Early Software Reliability Prediction, 303, 2013, pp. 81–104.
[2] Porter, R. Selby, Empirically guided software development using metric-based classification trees, IEEE Softw. 7(2)(1990) 46–54.
[3] S. Lessmann, B. Baesens, C. Mues, S. Pietsch, Benchmarking classification modelsfor software defect prediction: a proposed framework and novel findings, IEEETrans. Softw. Eng. 34(4)(2008)485–96.
[4] G. Czibula, Z. Marian, I.G. Czibula, Software defect prediction using relationalassociationrule mining, Inf. Sci. 264 (2014 April) 260–278.
[5] C. Jin, Software reliability prediction based on support vector regression usinga hybrid genetic algorithm and simulated annealing algorithm, IET Softw. 5 (4)(2011) 398–405.
[6] S. Kanmani, V.R. Uthariaraj, V. Sankaranarayanan, P. Thambidurai, Object-oriented software fault prediction using neural networks, Inf. Softw. Technol.49 (5) (2007) 483–492.
[7] C. Cagatay, S. Ugur, D. Banu, Practical development of an eclipse-based soft-ware fault prediction tool using naive Bayes algorithm, Expert Syst. Appl. 38(3) (2011) 2347–2353.
[8] F. Kang, J.J. Li, Z.Y. Ma, Rosenbrock artificial bee colony algorithm for accurateglobal optimization of numerical functions, Inf. Sci. 181 (16) (2011) 3508–3531.
[9] Alfi, H. Modares, System identification and control using adaptive particleswarm optimization, J. Appl. Math. Model. 35 (3) (2011) 1210–1221.
[10] S. Khan, A.R. Baig, W. Shahzad, A novel ant colony optimization based singlepath hierarchical classification algorithm for predicting gene ontology, Appl.SoftComput. 16 (2014 Mar) 34–49.
[11] Changseokbae, Wei-Chang yeh, Noorhanizawahid, Yuk yingchung and Yao liu, "A new simplified swarm optimization (sso) using exchange local search scheme". ICIC International @ 2012 issn 1349-4198. Volume 8, number 6, June 2012.
[12] Y.Q. Yang, G.J. Wang, Y. Yang, Parameters optimization of polygonal fuzzy neural networks based on GA-BP hybrid algorithm, Int. J. Mach. Learn. Cybern. 5(5) (2014) 815–822.
[13] N. E. Fenton and S. L. Pfleeger. Software metrics: A rigorous and Practical Approach. London: Intl Thomson Computer Press, 1997
[14] T.J. Mccabe, C.W. Butler, Design complexity measurement and testing, Com-mun. ACM 32 (12) (1989) 1415–1423.
[15] M.H. Halstead, Elements of Software Science, Elsevier, New York, 1977.
[16] M. Jureczko. Significance of different software metrics on defect prediction. An international Journal of Software Engineering, 2011, 1(1), p. 86-95.
[17] http://mdp.ivv.nasa.gov/index.html Date accessed 23/01/2017
[18] R. Burduk, P. Trajdos, Construction of Sequential Classifier Using ConfusionMatrix, Lecture Notes in Computer Science, 2013, pp. 401–407.