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

Ankit Pahal¹ and R. S. Chiller²

Department of Computer Science & Application, M.D. University, Rohtak, Haryana, India¹²

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 tenintrinsic 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.
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 effect the software fault-proneness prediction differently, and certain metrics effect the evaluation of software fault-proneness prediction slightly. Consequently, we musteliminate themetrics which have alittle impact for decreasing biased input dataset and increase the effectiveness of the prediction model. A sequence of comparatively noveloptimization algorithm is the swarm intelligence optimization comprising bee colony optimization [8], particle swarm optimization (PSO) [9] and ant colony optimization [10] etc. They simulates the swarm behavior of the population of individuals, using the exchange of information and teamwork of swarms to attain the optimization. In 1995, Eberhart and Kennedy developed a population-based optimization technique known as PSO, although PSO is a simple algorithm as compared to the other algorithms of swarm intelligence oweing to the better convergence rate and less control parameters. Nonlinear complex optimization problems are primarily solved using PSO. Furthermore, it uses the characteristics of the data itself and do not require any assumptions for software datasets in the implementation procedure of PSO. As a result PSO appeals several researchers and has arose as the best tool for optimization problems. But PSO likewise has particular drawbacks for example low 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 number 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 remained extensively functional in pattern recognition because of the reason that ANN-founded classifiers are able to include structural and statistical information together and accomplish superior 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 neural network which is based on error back propagation (BP) algorithm. The neural network model consist of an input layer and an output layer, and also one or more hidden layer in between. The neighboring layers accomplish full connectivity between neurons, though no connection is present between neurons withina layer. We used a three-layer network ANN in this paper. Furthermore, SSO is a universal convergence assured search method that presents 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 method is suggested. The block diagram of recommended software fault-proneness prediction method is presented in Fig. 3.

Fig. 3 Block diagram of proposed approach.

2.2. Software metrics

Several studies nowadays take software metrics as important parameter to recognize the modules which are
fault-prone and their studies revealed that software metrics are actually useful in predicting the fault-pronemodules. Previous research suggested various 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 example the Halstead’s Software Science [13] and the McCabe’s Cyclomatic number. 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 4 describes explanation of selected metrics. In this paper, each software module is denoted by 21 metrics for software fault-proneness prediction. They all are eminent software metrics in the software fault-proneness prediction perspective and consequently we do not offer the explanation of these metrics. The detail of the metrics can be found in [16].

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

Therefore, every noted value is drawn to the closed interval [0,1]. \((X(i), y(i))\) \((i = 1, 2, \ldots, q)\) is the normalized dataset and \(X(i) \in R^d\) is the normalized metrics vector.

2.4. Reducing dimensionality

Suppose that dataset, \(D = (S, M)\) where \(M = \{m_1, \ldots, m_q\}\) is the d metrics and \(S = \{s_1, \ldots, s_q\}\) is q samples sets, respectively. \(C = \{c_1, \ldots, c_l\}\) denotes type set. We convert the solution X of SSO into a binary string to reduce dimensionality, as the following operation: In place of some known random number, rand, in the interval [0,1],

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

where \(S(x) = \frac{1}{1 + e^{-x}}\) is sigmoidal function. In recommended reducing dimensionality method, a binary (0 or 1) string characteristic the position of particle i. Anda selected metric is denoted by 1, whereas a non-selected metric is denoted by 0. In SSO, whenever get the absolute solution Pglobal, the corresponding metric is followed with respect to Pglobal position. Assume \(M_i < M\) is the final selected metric, and let 1 < d be the total selected metrics.

### III. PROPOSED ALGORITHM

As the input & output dataset for ANN are normalized, thus the value will be in the range [0,1]. ANN is considered for a representation of “0” or “1” from the data space. These selected metrics set \(M_i\) represents the number of input neurons of ANN 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 X is 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 projects from the NASA repository are used as datasets that are openly available 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 datasets are KC2 and KC3 are implemented in C++ and Java languages respectively where a method is module in this instance. Every dataset has their software metrics and the associated variable that tells if the module has any fault-proneness or not. The modules that have \(v(G) > 10\) in the four selected datasets, classify to be fault-proneness as...
stated by the regular McCabes rules. The main characteristics of the datasets are shown in Table 1.

<table>
<thead>
<tr>
<th>Project</th>
<th># of modules</th>
<th># of faulty modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC1</td>
<td>1107</td>
<td>76</td>
</tr>
<tr>
<td>JM1</td>
<td>10,878</td>
<td>2102</td>
</tr>
<tr>
<td>KC1</td>
<td>2107</td>
<td>265</td>
</tr>
<tr>
<td>KC3</td>
<td>458</td>
<td>43</td>
</tr>
</tbody>
</table>

Table 1. Selected datasets.

This approach operates based on the number of incorrect or correct answers. The data in the Confusion matrix demonstrates the performance of the proposed algorithm for two-class problem that is shown in Table 2.

<table>
<thead>
<tr>
<th>Error in module</th>
<th>Reality</th>
<th>Forecast</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Positive</td>
<td>Negative</td>
</tr>
<tr>
<td>Positive</td>
<td>True positives (TP)</td>
<td>False-positives (FP)</td>
</tr>
<tr>
<td>Negative</td>
<td>False-negatives (FN)</td>
<td>True-negatives (TN)</td>
</tr>
</tbody>
</table>

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 determine the performance of the proposed method are shown in Fig 5.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of particles Q</td>
<td>60</td>
</tr>
<tr>
<td>Max iteration Gmax</td>
<td>100</td>
</tr>
<tr>
<td>In SSQ, the unique parameter ( \alpha )</td>
<td>0.55</td>
</tr>
<tr>
<td>In PSO, inertial weight ( w )</td>
<td>0.45</td>
</tr>
<tr>
<td>In PSO, acceleration constant ( c_1 = c_2 )</td>
<td>2</td>
</tr>
</tbody>
</table>

Fig. 5 Selected Parameters

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

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

The results demonstration accuracy of the proposed approach’s performance is 92.03%.
The results demonstration accuracy of the proposed approach’s performance is 89.13%.

V. CONCLUSIONS

This paper examined the usage of hybrid ANN and SSO to improve the software fault-proneness prediction method. The presented method can predict the software modules’ fault-proneness simply using software metrics. The key characteristics of presented prediction method is that dimensionality of themetrics is reduced using SSO and software modules’ fault-proneness is predicted using ANN. The reduction dimensionality method is actually effectiveness since there are 3 control parameters in PSO, whereas SSO have single parameter only, consequently SSO performs accurately. Results from experimental use of the simplifying dimensionality procedure which rely on SSO can reduce ANN model. The hybrid model of ANN & SSO has superior performance than current best other prediction methodologies and also it has verified to be much operational for founding association 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 the software 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