Adaptive Genetic Algorithm Based Artificial Neural Network for Software Defect Prediction

By Racharla Suresh Kumar & Bachala Satyanarayana

Sri Krishnadevaraya University, India

Abstract- To meet the requirement of an efficient software defect prediction, in this paper an evolutionary computing based neural network learning scheme has been developed that alleviates the existing Artificial Neural Network (ANN) limitations such as local minima and convergence issues. To achieve optimal software defect prediction, in this paper, Adaptive-Genetic Algorithm (A-GA) based ANN learning and weight estimation scheme has been developed. Unlike conventional GA, in this paper we have used adaptive crossover and mutation probability parameter that alleviates the issue of disruption towards optimal solution. We have used object oriented software metrics, CK metrics for fault prediction and the proposed Evolutionary Computing Based Hybrid Neural Network (HENN) algorithm has been examined for performance in terms of accuracy, precision, recall, F-measure, completeness etc, where it has performed better as compared to major existing schemes. The proposed scheme exhibited 97.99% prediction accuracy while ensuring optimal precision, F-measure and recall.

Keywords: software defect prediction, machine learning, genetic algorithm, artificial neural network, object oriented software metrics.

GJCST-D Classification : C.1.3 F.1.1

© 2015. Racharla Suresh Kumar & Bachala Satyanarayana. This is a research/review paper, distributed under the terms of the Creative Commons Attribution-Noncommercial 3.0 Unported License http://creativecommons.org/licenses/by-nc/3.0/), permitting all non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Adaptive Genetic Algorithm Based Artificial Neural Network for Software Defect Prediction

Racharla Suresh Kumar & Bachala Satyanarayana

Abstract- To meet the requirement of an efficient software defect prediction, in this paper an evolutionary computing based neural network learning scheme has been developed that alleviates the existing Artificial Neural Network (ANN) limitations such as local minima and convergence issues. To achieve optimal software defect prediction, in this paper, Adaptive-Genetic Algorithm (A-GA) based ANN learning and weight estimation scheme has been developed. Unlike conventional GA, in this paper we have used adaptive crossover and mutation probability parameter that alleviates the issue of disruption towards optimal solution. We have used object oriented software metrics, CK metrics for fault prediction and the proposed Evolutionary Computing Based Hybrid Neural Network (HENN) algorithm has been examined for performance in terms of accuracy, precision, recall, F-measure, completeness etc, where it has performed better as compared to major existing schemes. The proposed scheme exhibited 97.99% prediction accuracy while ensuring optimal precision, F-measure and recall.

Keywords: software defect prediction, machine learning, genetic algorithm, artificial neural network, object oriented software metrics.

I. INTRODUCTION

As per high pace rise in software applications and major dependency on it, the fault prediction has become one of the inevitable parts of software development life cycle (SDLC) that can play significant role in reducing the probability of software failure.

Software defect prediction (SDP) can be performed while planning to identify fault-prone modules in software product that as a result can provide the insight to the need for increased quality of monitoring during software development. In addition, it can also facilitate necessary approaches to incorporate certain proper fault verification schemes leading to enhanced software quality [1, 2, 3, 4] and reliability. SDP can be functional based on certain software metrics [3, 4, 5], such as source code changes, previous defects, etc. In fact software metrics are the quantitative data that are employed for characterizing the properties of source code and can be significant for predicting software quality. The efforts made through many generations have facilitated a number of schemes to mitigate defects, but the continuation of researches still indicates towards search for certain optimal SDP solution to ensure optimal performance, reliability, cost optimization and minimal maintenance. A number of efforts have been made for SDP using machine learning and neural network [6, 7, 8, 9, 10], clustering techniques, statistical method, mining and random forest [44, 45, 50] etc. In recent years, majority of software are being developed based on Object-Oriented (OO) paradigm. Thus, the quality of the software can be optimally assessed by employing software metrics, such as Abreu MOOD metric suite [11], QMOOD metrics suite [12], Bieman metric suite [13], Briand et al. [14], Etzkorn et al. [15], Halstead [16], Henderson-sellers [17], Li and Henry [18], McCabe [19], Tegarden et al. [20], Lorenz and Kidd [21] and CK metric [22] suite. These software metrics play significant role in assessing the quality of software such as precision, accuracy, fault-resilience and sensitivity etc. The significance of these object oriented software metrics lies in their capability to predict the software quality in terms of adaptability, functionality, usability, portability, supportability, reliability and cost effectiveness. Predominantly two data driven algorithms, support vector machine (SVM) and artificial neural network (ANN) algorithms have been employed for fault detection. ANN approach functions on the basis of the human brain behavior and possesses neurons and directed edges with certain weights in input-output relationship and can be stated to be a complex nonlinear mapping model between input and output layer. The processes in ANN comprise data sets to enhance the weight parameters, risk minimization scheme for stopping training as soon as the learning error enters in expected margin level. In fact, ANN has been employed in numerous utilities, but still it possesses certain limitations in terms of slow learning ability, local minima etc and hence require further optimization to achieve certain optimal SDP efficiency and performance. Thus, there is the requirement of further optimization of ANN approaches to accomplish a potential SDP solution. Some researches [23, 24] advocate the implementation of evolutionary computing techniques for SDP optimization. This paper proposes a novel evolutionary computing based enhanced ANN algorithm named Hybrid Evolutionary Computation

Author a: Research Scholar, Department of Computer Science, Sri Krishnadevaraya University, Andhra Pradesh, India. e-mail: sureshSKU@yahoo.com

Author b: Professor, Department of Computer Science, Sri Krishnadevaraya University, Andhra Pradesh, India. e-mail: bachalasatya@yahoo.com
based Neural Network (HENN) for defect prediction and classification. HENN system employs Adaptive Genetic Algorithm (A-GA) for optimal weight estimation so as to enhance weight update and learning efficiency of the ANN. In this paper, the object oriented software metrics, CK metrics [22] have been employed as a fault classification data and the respective performance has been analyzed using confusion matrix.

The remaining sections discusses, related work in Section II, problem definition is briefed in Section III, which has been followed by proposed research discussion in Section IV. Section V presents the results and analysis and conclusion has been discussed in Section VI.

II. RELATED WORK

The emergence of software applications and associated need of quality and reliability has motivated software practitioners as well as academia to develop certain novel scheme for defect prediction. With an objective to examine the relation between software metrics and associated faults some initiatives were made in [25, 26, 27, 28, 29, 30] where machine learning mechanism were used for fault detection. With an enthuse to compare the performance of varied other schemes such as decision trees, naïve Bayes, and 1-rule [31] performed fault detection using NASA MDP project. Chug et al [32] performed data mining based fault estimation using conventional J48, Random Forest, and Naïve Bayesian Classifier (NBC) schemes but still couldn’t employ the benefits of advanced classification schemes. With an objective to enhance conventional schemes Pushphavathi et al [33] introduced hybrid scheme of random forest (RF) and Fuzzy C Means (FCM) clustering. Then while, these systems were found limited for unbalanced data sets, which motivated author [34] to propose an approach called AdaBoost.NC that explored varied kinds of class imbalance learning schemes comprising resampling techniques, threshold moving, and ensemble algorithms. With an objective to explore SVM optimization in [35, 36] a dynamic SVM model was proposed for fault detection in source code using with error data and faulty code execution. Researcher in [37] developed an ANN based SDP system. This is the matter of fact that SVM refers the functional paradigm of single layer perceptron’s NN which on addition with kernels behaves like multilayered perceptron’s [38]. Till available systems based neural network with conventional learning and weight estimation suffers from local optima and convergence issue, which has not been discussed dominantly. On contrary, these days the software are developed and examined for faults using object oriented software metrics which even being significant has not been explored in depth to ensure optimal solution for reliability oriented defect prediction.

This paper intends to provide an optimal solution for software defect prediction using evolutionary computing based neural network for efficient fault classification.

III. PROBLEM DEFINITION

In software development life cycle the reliability assurance is of great significance and to achieve it, the defect prediction is an inevitable need. The defect prediction can be performed using software metrics data, in which either it is predicted whether the code is defective or not or the magnitude of the probable defect and its severity is examined. In this research work, the predominant questions are whether evolutionary computing schemes, specifically GA can optimize neural network based artificial intelligence (AI) to achieve optimal software defect prediction. An another question that this research paper considers is that whether the conventional Genetic Algorithm can be further enhanced to deal with a scenario where multiple chromosomes are having similar fitness, and how this enhancement would perform classification or fault prediction?. In order to explore the answers of this significant question, in this paper it has been intended to optimize ANN learning and respective optimal weight estimation using GA, which has further being optimized to behave as an Adaptive GA (A-GA) scheme that ensures adaptive GA parameters (Crossover and mutation) estimation. Here, considering requirements of object oriented software metrics, CK metrics [22] have been been considered that characterizes overall features of software in terms of varied component features. In this paper, the key software metrics considered are WMC, NOC, DIT, CBO, RFC, LCOM, which can be considered for defect prediction in certain class or data model. Based on the proposed model, the defect can be predicted which can be useful for ensuring quality and reliability of the software product. Given a training data, certain learning model can be developed that can classify the data for its faulty or non-faulty status. The artificial intelligence technique neural network has been used extensively so far for classification utilities, but being conventional these approaches do suffer from local minima and weight update issues. Thus, to enhance the systems, certain global optimization schemes like evolutionary computing can be considered. Since Particle Swarm Optimization suffers due to optimal minima and convergence issues, here we proposed an adaptive GA (A-GA) for ANN weight estimation where the weights are estimated dynamically in each iteration. Here, mean square error has been considered as the fitness value for A-GA. Further, the GA parameters such as crossover probability and mutation probability can be adaptively updated to make the overall system more robust and efficient. The optimization of ANN with A-GA can make it more effective and can be a potential candidate for fault detection in SDLC applications.
performance evaluation for these two approaches can be done in terms of accuracy, precision, recall, specificity etc.

IV. Proposed System

This section discusses the proposed evolutionary computing based hybrid neural network (HENN) for software defect prediction. HENN: Evolutionary Computing Based Neural Network for Software Defect Prediction.

Neural networks (NN) have seen an explosion of interest over the years, and are being successfully applied across a range of problem domains. Indeed, anywhere dealing with the problem of classification and prediction, neural networks are being used. For software defect prediction, ANN can be employed with learning approaches such as Gradient Descent (GD), Gauss Newton, and Levenberg Marquardt (LM) etc. Unlike conventional approach, in this paper, we have proposed an evolutionary computing technique called Adaptive Genetic Algorithm for ANN learning optimization and weight estimation, which has been further employed for fault prediction. Here, we intend to find relation between object oriented software metrics and fault prone classes and six CK metrics; WMC, NOC, DIT, RFC, CBO, LCOM. In this paper we have considered eight hidden layers. Since, in the proposed SDP model, only FAULTY and NON-FAULTY are the results expected for prediction, therefore only one output node. The overall design of the proposed ANN model can be presented as follows:

![Figure 1: ANN model for Defect prediction](image)

For input layer i.e., the output of the output layer is treated as input of the input layer\(O_0 = I_1\). Further, the sigmoid function has been employed for hidden layer\(O_h\). Hence, the result of the hidden nodes \(O_h\) with the feed input of \(I_h\) is estimated mathematically as \(O_h = \frac{1}{1+e^{-I_h}}\) and final outcome of output nodes \(O_o\) is presented mathematically by \(O_o = \frac{1}{1+e^{-O_h}}\).

In general, ANN is represented by a function \(Y = f(W, X)\) where \(Y\) represents the output vector and \(W\) and \(X\) are the weight vector and the input vector respectively. In process, the weight factor \(W\) is updated in each iterations to reduce Mean Square Error (MSE), which is estimated as follows:

\[
MSE = \frac{1}{n} \sum_{i=1}^{n} (y_i' - y_i)^2
\]

(1)

Where \(y\) represents the actual output while the expected output is given by \(y_i'\).

In order to process the datasets using ANN, at first the normalization of data is required. A discussion of the proposed data normalization technique in this paper is given as follows:

a) Data normalization

In the proposed model, initially the normalization has been performed before data processing that strengthens the system for better readability and defect prediction. Here the data normalization has been done over the range of \([0, 1]\) for adjusting the defined range of input feature value and avoid the saturation of neurons. A number of schemes such as Min-Max normalization, Z-Score normalization and decimal scaling can be employed for the purpose of data normalization. In this paper, Min-Max normalization approach has been used that performs a linear transformation on the original data and maps each of the actual data \(x_i\) of attribute \(X\) to normalized value \(x_i'\) that exists in the range of \([0, 1]\). The Min-Max based normalized data has been obtained by the following expression:

\[
\text{Normalized}(x_i) = x_i' = \frac{x_i - \min(X)}{\max(X) - \min(X)}
\]

(2)

Where \(\max(X)\) and \(\min(X)\) represent the maximum and minimum value of the attribute \(X\) respectively. Performing data normalization the ANN model has been employed for fault classification and SDP functions.

In ANN based artificial intelligence systems, the efficient weight estimation is of great significance and till existing approaches have explored techniques such as Gauss Newton, Gradient descent, Levenberg Marquardt etc. Unfortunately these approaches couldn’t be enhanced by scientific society to make weight estimation effective by means of certain global optimization techniques such as Genetic Algorithm. Efficient weight estimation during ANN learning can
make classification optimal. This requirement motivated us to employ genetic algorithm for dynamic weight estimation during ANN learning. A brief discussion of the proposed Adaptive Genetic Algorithm (A-GA) is given in the following section.

b) Adaptive Genetic Algorithm (A-GA)

Genetic Algorithm (GA) is an adaptive search method for finding optimal or near optimal solutions, premised on the evolutionary ideas of natural selection. The fundamental concept of GA is emphasized on simulating processes in the natural system required for evolution, distinctively those that consider the Charles Darwin principles representing the terms of the survival of the fittest. Considering procedural flow, GA at first generates the initial population arbitrarily, where population refers a set of solutions. These solutions are nothing else but a chromosome that possesses a form of binary strings where all the comprising parameters are supposed to be encoded. Generating the population, GA estimates the fitness function of individual chromosome. Here the fitness function states toward a user-defined function that returns the evaluation results of each chromosome, thus a higher fitness value means its chromosome is a dominant gene. As per retrieved fitness values, offspring are generated using genetic operators—crossover and mutation. Applying these genetic operators the generations of the population are repeated iteratively until the stopping criteria are satisfied and an optimal solution is achieved. As illustrated in Figure-1, in this paper, the proposed HENN model comprises 1-h-0 network configuration with i input layer, h hidden layer and 0 output layer or neurons. In the proposed ANN model, all the six considered CK metrics or feature vector are fed as input to the individual input node, where each feature vector metrics accompanies the number of classes available in datasets. Considering Figure-1 and relevant network configuration, there is N weight required to be estimated. Mathematically, the number of weight vectors is:

\[ N = (i + O) \times h \]  

(3)

Here, the individual weight, which is considered as gene in the chromosomes of the A-GA, is a real number. Considering the gene length or the number of digits bel. Then the length of the chromosome is:

\[ L_{Chrom} = N \times l = (i + O) \times h \times l \]  

(4)

These all chromosomes are considered as the population of the genetic algorithm. In the proposed model to estimate the fitness value of the individual chromosome, the weights are required to be extracted from the individual chromosome. In our proposed model, the weights \( W_k \) are estimated by the following expression:

\[ W_k = \begin{align*}
&\text{if } 0 \leq x_{k,l+1} < 5 \\
&-x_{k,l+2} \times 10^{-2} + x_{k,l+3} \times 10^{-3} + \cdots + x_{k,l+10} \\
&+ x_{k,l+11} \times 10^{-2} + x_{k,l+12} \times 10^{-3} + \cdots + x_{k,l+10} \\
&\text{if } 5 \leq x_{k,l+1} \leq 9
\end{align*} \]  

(5)

In order to process the Adaptive Genetic Algorithm (A-GA), the fitness values for each chromosome are required to be estimated. The fitness generation algorithm for the proposed A-GA system is given in Figure-2. The fitness values of each chromosome is estimated on the basis of the derived fitness function and the considered algorithm for deriving fitness function are \( T_i = (t_{i1}, t_{i2}, t_{i3}, \ldots, t_{in}) \) and \( I_i = (i_{11}, i_{12}, i_{13}, \ldots, i_{ih}) \) for each chromosome \( C_i, i = 1, 2, 3, \ldots, p \), belonging to the current population \( P_i \) of size \( p \).

**Algorithm for Fitness Estimation**

**Input:** \( I_i \) (integers) 
**Output:** \( T_i \) (integers) 

Where \( I_i \), \( T_i \) represent the input and output pairs of the \( i - h - 0 \) configuration of neural network.

**Phase-1:** Weights \( W_i \) from \( C_i \) can be estimated by

\[ W_k = \begin{cases} 
- x_{k,l+1} \times 10^{-2} + x_{k,l+2} \times 10^{-3} + \cdots + x_{k,l+10} \\
+ x_{k,l+11} \times 10^{-2} + x_{k,l+12} \times 10^{-3} + \cdots + x_{k,l+10} 
\end{cases} \]  

(6)

**Phase-2:** Considering \( W_i \) as a constant weight, train the network for \( N \) input instances and estimate output \( O_i \).

**Phase-3:** Estimate error \( E_j \) for each input instance \( j \),

\[ E_i = \sqrt{\frac{1}{N} \sum_{j=1}^{N} E_j} \]  

Where \( N \) is the total number of training data set

**Phase-4:** Estimate Root mean square error (RMSE) of chromosome \( C_i \)

**Phase-5:** Estimate the fitness value for chromosome \( C_i \)

\[ F_i = \frac{1}{N} \sum_{j=1}^{N} E_j \]  

\[ F_i \]  

**Figure 2:** Algorithm for Fitness generation using A-GA

This is the matter of fact that the evolutionary computing scheme named Genetic Algorithm has established itself as a potential optimization technique for various application scenarios, still this approach possess scopes for further optimization that specifically depends on the working environment. In this paper, there might be the possibility that after every generation to achieve optimal fitness, certain new population would be generated and thus the processing data might be increased after each iterations, thus resulting into certain
restraints such as premature convergence caused due to local optima and low convergence speed, which is common in other evolutionary techniques such as Particle Swarm Optimization. In order to alleviate these issues, the parameters like cross over probability \( P_c \) and mutation probability \( P_m \) can be made dynamic and weight adaptive. In addition, such novelty can deal with a common scenario, where there is the possibility of multiple chromosomes having similar fitness value, causing degraded classification accuracy. Taking into consideration of these all factors and motivations, in this paper a weight adaptive genetic algorithm (A-GA) has been developed where the genetic parameters (Crossover and mutation) are updated dynamically. In the proposed approach the parameters \( P_c \) and \( P_m \) have been dynamically updated by means of the following mathematical model:

\[
(P_c)_{k+1} = (P_c)_k - \frac{C_1 * n}{5}
\]

\[
(P_m)_{k+1} = (P_m)_k - \frac{C_2 * n}{5}
\]

Where \( (P_c)_{k+1} \) and \( (P_m)_{k+1} \) represent the updated probability of cross over and mutation, \( (P_c)_k \) and \( (P_m)_k \) are the current probability of cross over and mutation, \( C_1 \) and \( C_2 \) can be the positive constant and \( n \) is the number of chromosome having same fitness value. Thus, implementing these discussed approaches, if the final output estimated is greater than 0.5, then the class is labeled as FAULTY otherwise NON-FAULTY. Figure-3 represents the overall process of software defect prediction using Adaptive Genetic Algorithm (A-GA).

**Figure 3:** Proposed HENN Scheme for Software Defect Prediction

The discussion of Figure-3 for HENN simulation is given as follows:

i. **HENN-SDP Simulation**

As illustrated in Figure-1, in the proposed HENN model, three layers of neural network has been considered comprising six input nodes, eight hidden nodes and one output node. The overall process of HENN based fault estimation is discussed as follows:

**Chromosome Initialization**: Initial chromosome selection with population size is 50 has been considered using random selection process.
**Weight Estimation:** Obtained the weight vector $W_{\text{kt}}$ for each chromosome as the input to hidden layer and hidden layer to output layer and thus the weight of input to hidden node and hidden node to output are estimated using equation 5.

**Fitness Estimation:** On the basis of weights retrieved, the fitness value is estimated for each chromosome, where the proposed HENN intends to minimize the mean square error as defined in Figure 2.

**Ranking of Chromosomes:** Perform the ranking of each chromosomes based on respective fitness value and substitute the chromosomes with minimum fitness value by the chromosomes with highest fitness value chromosome.

**Crossover:** Perform two point crossover process dynamically vary the GA parameters $P_c$ and $P_m$ till reaching optimal criteria using equation (6).

In the simulation model, the initial $P_c$ and $P_m$ are 0.6 and 0.1 respectively and a signifies the number of chromosome having similar fitness value.

**Stopping Criteria:** The developed system terminates once the 95% chromosomes in the gene pool accomplishes its unique fitness value and beyond this the fitness level of chromosomes gets saturated.

**Classify Faults:** If the final weight is greater than 0.5, then the class is labeled as non-faulty otherwise FAULTY. Confusion Matrix Generate the confusion matrix for each classes of OO-SM and classify fault/non-fault distribution for performance evaluation.

Thus, employing the proposed HENN model, the fault classification and prediction has been done. The simulation, results and discussion is provided in the following section.

**V. Result and Analysis**

This section discusses the research variables, simulation setups, results obtained and respective performance analysis.

a) **Data collection**

In this paper, the CK metric suites have been employed which have been defined for varied objectives such as software fault detection/prediction, effort evaluation, re-usability and maintenance. Considering the robustness of CK metric suite [27], it has been used as object oriented software metrics which has been processed using Chidamber and Kemerer Java Metrics tool (CKJM) tool that extracts software metrics by executing byte code of compiled Java cases and assigns a definite weight of the comprising classes having feature vectors. In this paper, PROMISE fault benchmark data [39] and NASA MDP datasets [40] and PROMISE repository to evaluate the performance of the proposed fault prediction scheme. We intended to establish the relationship between Object-Oriented software metrics (OO-SM) and the fault proneness at the class level. In order to perform defect prediction using regression analysis paradigm, we have considered fault as a dependent variable while the CK metric as the independent variable. The predominant OO-SM metrics are given in Table-1.

<table>
<thead>
<tr>
<th>Table 1: Object Oriented Software Metrics (CK Metrics [22])</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMC</td>
</tr>
<tr>
<td>NOC</td>
</tr>
<tr>
<td>DIT</td>
</tr>
<tr>
<td>CBO</td>
</tr>
<tr>
<td>RFC</td>
</tr>
<tr>
<td>LCOM</td>
</tr>
<tr>
<td>NOM</td>
</tr>
<tr>
<td>NOA</td>
</tr>
<tr>
<td>NOAI</td>
</tr>
<tr>
<td>NOMI</td>
</tr>
<tr>
<td>Fan-in</td>
</tr>
<tr>
<td>Fan-out</td>
</tr>
<tr>
<td>NOPM</td>
</tr>
<tr>
<td>NOPA</td>
</tr>
<tr>
<td>NOPM</td>
</tr>
<tr>
<td>NOPA</td>
</tr>
<tr>
<td>NLOC</td>
</tr>
</tbody>
</table>

In our work, we have developed a function to explore the relation between Object-Oriented software metrics (OO-SM) (WMC, NOC, DIT, RFC, CBO and LCOM) and faults existing in class under consideration. The minimization of faults can be of great significance towards optimization of software equality, and to ensure optimal defect prediction, the fault has been derived as the function of software metrics as illustrated as follows:

$$\text{Faults} = f(\text{WMC}, \text{NOC}, \text{DIT}, \text{CBO}, \text{RFC}, \text{LCOM})$$

We used four public domain defect datasets from the PROMISE repository [9][39]. The considered data sets are JEdit, IVY, Ant and Camel which contain static code measures along with varied modules sizes, defective modules and defect rates. In our simulation model, the respective extracted weights and features of the data classes are taken as input. The datasets with respective classes or modules are given in Table-2.

<table>
<thead>
<tr>
<th>Table 2: PROMISE Data and modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROMISE</td>
</tr>
<tr>
<td>Number of modules</td>
</tr>
</tbody>
</table>
In this paper, HENN algorithm has been developed for simulation using MATLAB 2012b software tool having artificial intelligence and ANN toolboxes. The proposed models examined defect datasets and the FAULTY and NON FAULTY data have been classified. Here on the basis of FAULT distribution by proposed model, a confusion matrix has been generated that encompasses two rows and columns comprising true negatives, true positive, false negative and false positive variables. The respective values of True negatives (TN) refer the modules which are NON FAULTY or fault-free on the other hand, true positives (TP) represents for those modules which are classified as FAULTY. False negatives (FN) are those modules which are FAULTY and are classified incorrectly as NON FAULTY. Similarly, false positives (FP) modules are those modules which are faultless but are classified incorrectly as FAULTY. A matrix presentation of confusion matrix is given in Table 3.

Table 3: Confusion Matrix

<table>
<thead>
<tr>
<th>Predicted Defective</th>
<th>Predicted Defect Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAULTY (True Positive)</td>
<td>False Negative</td>
</tr>
<tr>
<td>NON-FAULTY (False Positive)</td>
<td>True Negative</td>
</tr>
</tbody>
</table>

Generally, the meanings of the values of the binary variables are not needed to be defined, however, in our work, especially for performance assessment the variables have been labeled as positive and negative. The positive levels refer towards the results as FAULTY in that specific simulation scenario. In this paper, we have measured the performance of the proposed HENN SDP in terms of correctness, precision, F-measures, accuracy, recall, specification and cost factor analysis. A brief mathematical definition of these variables is given as follows:

Table 4: Performance Parameters

<table>
<thead>
<tr>
<th>Construct</th>
<th>Mathematical Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recall</td>
<td>TP/(TP+FN)</td>
<td>Proportion of defective units correctly classified</td>
</tr>
<tr>
<td>Precision</td>
<td>TP/(TP+FP)</td>
<td>Proportion of Units correctly predicted as defective</td>
</tr>
<tr>
<td>Specification</td>
<td>TN/(TN+FP)</td>
<td>Proportion of correctly classified non defective units</td>
</tr>
</tbody>
</table>

Table 5: Performance analysis for proposed HENN SDP model and other existing models

<table>
<thead>
<tr>
<th>SDP Techniques</th>
<th>Accuracy (%)</th>
<th>Precision (%)</th>
<th>F-Measure (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LLE-SVM [41]</td>
<td>81.1</td>
<td>82.5</td>
<td>80.4</td>
</tr>
<tr>
<td>SVM [41]</td>
<td>69.4</td>
<td>68.1</td>
<td>69.7</td>
</tr>
<tr>
<td>SVM [42]</td>
<td>55.3</td>
<td>88.0</td>
<td>83.2</td>
</tr>
<tr>
<td>Natural Gas [48]</td>
<td>94.25</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Symbolic Regression [48]</td>
<td>89.50</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>RBP-NN [46]</td>
<td>80.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MLP [42]</td>
<td>86.6</td>
<td>86.6</td>
<td>87.4</td>
</tr>
<tr>
<td>Naive Based [42]</td>
<td>85.6</td>
<td>83.1</td>
<td>83.9</td>
</tr>
<tr>
<td>CPSO[43]</td>
<td>69.2</td>
<td>67.6</td>
<td>-</td>
</tr>
<tr>
<td>T-SVM [44]</td>
<td>75.8</td>
<td>84.1</td>
<td>80.91</td>
</tr>
<tr>
<td>GANN[43]</td>
<td>73.4</td>
<td>81.6</td>
<td>-</td>
</tr>
<tr>
<td>AdaBoost [43]</td>
<td>79.1</td>
<td>82.3</td>
<td>-</td>
</tr>
<tr>
<td>Random Forest [50]</td>
<td>91.4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>k-NN [47]</td>
<td>91.8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>C4.5 [47]</td>
<td>88.39</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>J 48 [47]</td>
<td>90.90</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Levenberg-Marquardt-NN [47]</td>
<td>88.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>NNEP-Evolutionary [43]</td>
<td>88.8</td>
<td>81.2</td>
<td>-</td>
</tr>
<tr>
<td>PSO [46]</td>
<td>78.78</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PSO-NN [48]</td>
<td>97.75</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>HENN SDP *</td>
<td>97.9</td>
<td>1</td>
<td>98.9</td>
</tr>
</tbody>
</table>

* The best performance of HENN

Thus, the results obtained exhibit that the optimization made by means of Adaptive Genetic Algorithm has enhanced ANN learning for fault detection. The ultimate results obtained for HENN represents the most effective and optimal results as compared to other existing approaches, especially neural network based SDP models. The performance analysis for the proposed systems is given in Table-6.
VI. Conclusion

Software defect prediction has become an inevitable need for organizations to ensure quality and reliability of software products. The early defect prediction can facilitate managers to rectify and enrich reliability of product. Approaches such as machine learning and neural network have become eminent solution for training and classification of data and can be significant for defect prediction. However, these approaches need optimization in terms of weight update, parametric enhancement while performing defect prediction. The local minima and convergence issue of ANN can be significantly dealt with employing evolutionary computing schemes and the implementation of genetic algorithm can be the dominant candidate. In this paper, Adaptive Genetic Algorithm (A-GA) has been used for ANN optimization, where A-GA functions for optimal weight estimation. The proposed HENN model has been tested with PROMISE data sets, where the average accuracy for HENN was retrieved as 87.23% while the best classification performance was observed with JEdit datasets where HENN exhibited 97.99% accuracy while ensuring 100% precision. Performance in terms of F-measure using HENN was obtained as 98.97%. The results also depicted that A-GA based ANN can outperform Particle Swarm Optimization based defect prediction schemes, regression techniques, AdaBoost, and other conventional weight estimation based ANN models. In future, the efficiency of the proposed HENN model can be examined in comparison with optimized machine learning such as SVM with varied kernel functions.

References Références Referencias


41. Chun Shan, Boyang Chen, Changzhen Hu, Jingfeng Xue1, Ning Li, "SOFTWARE DEFECT PREDICTION MODEL BASED ON LLE AND SVM" Communications Security Conference; pp 1-5, 22-24 May 2014.


