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

IJARCCE
International Journal of Advanced Research in Computer and Communication Engineering
ICRITCSA
M S Ramaiah Institute of Technology, Bangalore
Vol. 5, Special Issue 2, October 2016

An Efficient Technique for Optimization of Test Cases with the Aid of HNN-MCS for Assessing Software Reliability

Lakshminarayana P[1], Dr T V Suresh Kumar[2]

Associate Professor, Department of Computer Applications, BMS College of Engineering, Bangalore, India[1]

Professor and Head, Department of Computer Applications, MSRIT, Bangalore, India[2]

Abstract: The advance in software development has resulted in need for efficient and reliable software products. In recent the growth of software demands high reliability and safety, software reliability prediction becomes more and more essential. Software reliability is a key part of software quality. Various techniques for predicting software reliability have been proposed and evaluated in terms of their prediction performance; however, their actual contribution to business objectives such as quality improvement and cost reduction has been rarely assessed. The main aim of this work is to develop an efficient software reliability prediction method where soft computing is utilized. We are proposing a novel method of reliability prediction with the aid of Hybrid Neural network incorporated with optimization algorithm (HNN-MCS). The weight factor is globally optimized using the modified cuckoo search algorithm. Once the training is done the data are then tested in order to check the prediction accuracy of the proposed systems. Researchers considered different factors as inputs for training the network. The execution time is utilized in our proposed system for training the neural network and based on this the testing is done. The results in terms of actual and predicted failure rate are estimated.

Keywords: Technique for Optimization of Test Cases with the Aid of HNN-MCS

1. INTRODUCTION

Modern society is highly engaged with the roles of software. Software engineers and software development organizations seeks great responsibilities on maintaining quality, reliability and consumer satisfaction with the software products. Software development testing phase is generally considered as one of the major quality control techniques [2]. The goal of software engineering is to develop the techniques and tools needed to develop high-quality applications that are more stable and maintainable.

In order to assess and improve the quality of an application during the development process, developers and managers use several metrics [5]. Various business and technical motives such as shorter development cycles, lower development costs, improved product quality, and access to source code, more and more software developers and companies are basing their software products on open source components [6]. To quantify the failure behavior of a software system, software reliability is an important measure to help developers to arrange adequate test activities. By facilitating the prior estimation of software reliability, developers can dynamically reallocate the testing resources, and reduce the cost of fixing bugs after releasing the software [9], and software reliability becomes a very important characteristic of the computer systems.

Software reliability is consequently one of the most important features for a critical software system. According to the ANSI definition, software reliability is defined as the probability of failure-free software operation for a specified period of time under a specified environment. In practice, it is very difficult for the project managers to measure software reliability & quality [7].

In order to calculate and predict the product quality, software reliability is found as a significant attribute [3]. Software reliability can be defined as a probability of zero-failure operation of particular software at a specific instant of time in a specific kind of environment [1]. In order to ensure the cumulative reliability of software, it is important to precisely model the software reliability and to predict the probable trends. Certain, but important metrics such as time period, MTBF, number of faults and MTTFs through SRGMS would be helpful for such circumstances [4].

Predicting software development effort with high precision is still a largely unsolved problem. Consequently, there is an ongoing, high level of activity in this research field. A large number of different prediction models1 have been proposed over the past twenty years. So far, the lack of convergence of studies on software prediction models is poorly understood, and it has been a puzzle to the research community on software prediction

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

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICRITCSA

M S Ramaiah Institute of Technology, Bangalore

Vol. 5, Special Issue 2, October 2016

systems for many years. Clearly, the need is to consolidate the knowledge on software prediction models and research procedures; it is necessary to understand why we have obtained so wildly opposing conclusions on this matter [8]. Being able to predict the number of faults resides in software helps significantly in specifying/ computing the software release day and manage project resources artificial neural network (ANN) with for software reliability prediction aroused more research interest [10].
Neural-network encompasses a noteworthy lead over analytic models, nevertheless, because they necessitate only failure history as input, no postulations. Using that input, the neural-network model automatically develops its own internal model of the failure process and predicts future failures. Similarly Cuckoo search helps in estimating the weight values and that be utilized in neural network for predicting the reliability of software. Based on all these considerations it is proposed modified cuckoo search based neural network classifier in order to predict the reliability of the software since it improves the classification accuracy to a larger extend.

2. RELATED WORK

A number of researches have been proposed by researchers for the prediction of Software reliability. Following are few literatures applied for assessment of the state-of-art work on the reliability prediction models. Software reliability prediction was very important for minimizing cost and improving the effectiveness of the software development process. As an important method, relative data during software lifecycle was used to analyse and predict software reliability. However, predicting the variability of software reliability with time was very difficult. Recently, support vector regressions (SVR) have been widely applied to solve non-linear predicting problems in many fields such as software reliability prediction and have obtained well performance in many situations, and it was still difficult to select its parameters.

Previously, intelligence optimization algorithms, such as genetic algorithm (GA), are mostly used for finding better parameters of SVR, however existing methods of selecting parameters require usually has some disadvantages. Jin [11] have proposed a technique to overcome weaknesses of GA, such as the local minima and the premature convergence problems, GA and simulated annealing (SA) are integrated into an optimize algorithm, called GA-SA, it is then applied to SVR for predicting software reliability. Then the proposed GA-SA-SVR model was compared with other software reliability models through real software failure data. The experimental result showed that the proposed GA-SA-SVR model could obtain better predictions results than the other models and has a fairly accurate prediction capability. Although many algorithms and techniques have been developed for estimating the reliability of component-based software systems (CBSSs), much more research was needed. Accurate estimation of the reliability of a CBSS was difficult because it depends on two factors: component reliability and glue code reliability. Moreover, reliability was a real-world phenomenon with many associated real-time problems. Soft computing techniques could help to solve problems whose solutions are uncertain or unpredictable. A number of soft computing approaches for estimating CBSS reliability have been proposed. These techniques learn from the past and capture existing patterns in data. The two basic elements of soft computing are neural networks and fuzzy logic. Tyagi and Sharma [12] have proposed a model for estimating CBSS reliability, known as an adaptive neuro fuzzy inference system (ANFIS), that was based on these two basic elements of soft computing, and they compared its performance with that of a plain FIS (fuzzy inference system) for different data sets.
Early prediction of software reliability might be used to evaluate design feasibility, compare design alternatives, identify potential failure areas, trade-off system design factors, track reliability improvements, and identify the cost overrun at an early stage and to provide optimal development strategies. Many researchers have proposed different approaches to predict the software reliability based on Markov model but the uncertainty associated with these approaches is to find the transition probabilities in between the two states of the Markov chain. Singh et al. [13] have proposed an approach to address this problem by modeling the software system through Petri Net, converting it into Markov chain and solving the linear system mathematically. The validation of the proposed approach have also been shown by comparing the predicted reliability, based on predicted transition probability, with computed reliability, based on operational profile of safety critical software of Nuclear Power Plant.

Software reliability prediction plays a very important role in the analysis of software quality and balance of software cost. The data during software lifecycle was used to analyze and predict software reliability. However, predicting the variability of software reliability with time was very difficult. Recently, support vector regression (SVR) have been widely applied to solve nonlinear predicting problems in many fields and has obtained good performance in many situations; however it was still difficult to optimize SVR's parameters. Previously, some optimization algorithms have been used to find better parameters of SVR, but these existing algorithms usually are not fully satisfactory. Cong Jina and Shu-Wei Jin [14] have first improved estimation of distribution algorithms (EDA) in order to maintain the diversity of the population, and then a hybrid improved estimation of distribution algorithms (IEDA) and SVR model, called IEDA-SVR model, was proposed. IEDA was used to optimize parameters of SVR, and IEDA-SVR model was used to predict software reliability. They compared IEDA-SVR

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

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICRITCSA

M S Ramaiah Institute of Technology, Bangalore

Vol. 5, Special Issue 2, October 2016

model with other software reliability models using real software failure datasets. In spite of much research efforts to develop software reliability models, there was no single model which was appropriate in all circumstances. Accordingly, some recent studies on software reliability have attempted to use existing models more effectively in practice (e.g., model selection and combination). However, it was not easy to identify which model was likely to make the most trustworthy predictions and to assign appropriate weights to models for the combination. The improper model selection or weight assignment often causes unsuccessful software reliability prediction in practice, which leads to cost/schedule overrun. Park and Baik [15] have proposed a systematic reliability prediction framework which dynamically selects and combines multiple software reliability models based on the decision trees learning of multi-criteria. For the model selection, the proposed approach uses the empirical patterns of multi-criteria derived from multi reliability models. Reduced error pruning decision tree selects the models with the best predictive patterns and automatically assign a weight to each model. Then, the likelihood of over- or under-prediction of the identified models was examined, and the competitive models in both tendency groups are combined by their given weights.

Roy [16] has proposed a multi-layer feedforward artificial neural network (ANN) based logistic growth curve model (LGCM) for software reliability estimation and prediction. They developed the ANN by designing different activation functions for the hidden layer neurons of the network. They explained the ANN from the mathematical viewpoint of logistic growth curve modeling for software reliability. They also proposed a neuro-genetic approach for the ANN based LGCM by optimizing the weights of the network using proposed genetic algorithm (GA). They first trained the ANN using back-propagation algorithm (BPA) to predict software reliability. After that, they used the proposed GA to train the ANN by globally optimizing the weights of the network. The proposed ANN based LGCM was compared with the traditional Non-homogeneous Poisson process (NHPP) based software reliability growth models (SRGMs) and ANN based software reliability models. They presented the comparison between the two training algorithms when they are applied to train the proposed ANN to predict software reliability. The applicability of the different approaches was explained through three real software failure data sets.

3. PROPOSED METHODOLOGY:

Basic Concept:

Since software products are rising rapidly in size and complexity, software reliability prediction has become a vital task in the software development process. Software reliability prediction is very important for minimizing cost and improving the effectiveness of the software development process. As an important method, relative data during software lifecycle is used to analyse and predict software reliability. Subsequently, considering the importance of proactive action on important factors like early identification of the cost overrun, to provide optimal development strategies, track reliability improvements, and also to overcome the stated problem, the field of early prediction of software reliability has taken on boost.

The modified cuckoo search is an optimization technique which is utilized to optimize the weight factor in the neural network classifier. The optimization of the weight factor can aid in improving the classification rate thereby making the neural network more efficient. The modified cuckoo search is chosen prior to other optimization techniques as it provide better optimized outcome.

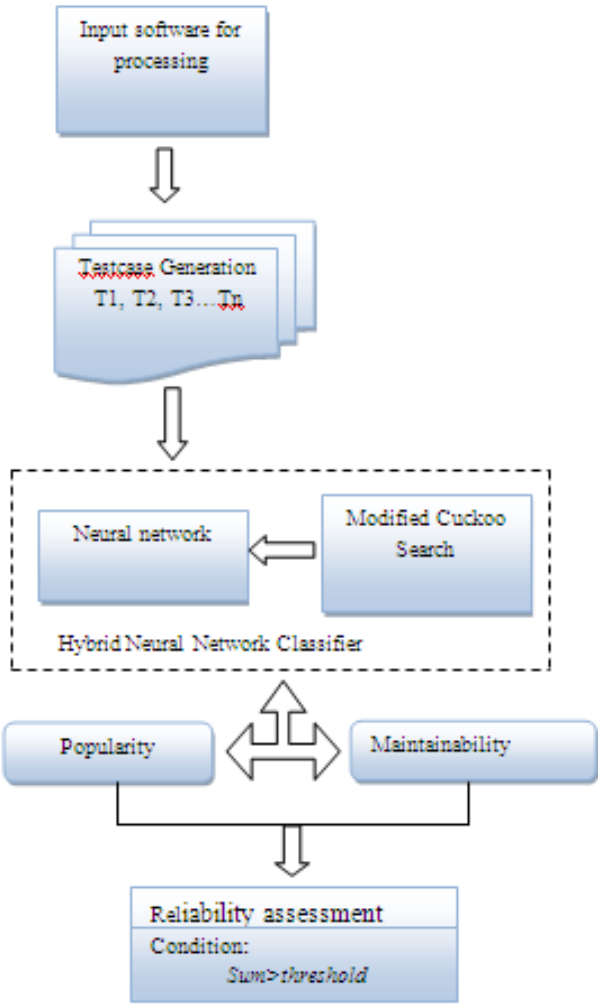

Fig 1: Proposed software reliability assessing model.

3.1 Steps involved in the Software reliability prediction:

The reliability estimation basically utilizes different software parameter inorder to calculate the quality of particular software. The figure 1 shows the steps involved in our proposed method for the software reliability

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

IJARCCE
International Journal of Advanced Research in Computer and Communication Engineering
ICRITCSA
M S Ramaiah Institute of Technology, Bangalore
Vol. 5, Special Issue 2, October 2016

prediction process. The Proposed technique can help to make the system more reliable by assisting software architects in evaluating the impact of their design decisions on the system reliability. This can help to save costs, time, and efforts significantly by avoiding implementing software architectures that do not meet the reliability requirements. However, existing reliability prediction approaches for component-based software systems suffer from the following drawbacks and therefore are limited in their applicability and accuracy. Here we will calculate the application reliability which is estimated based on the reliability of the individual components and their interconnection mechanisms.

We have proposed a novel method of reliability prediction with the aid of Hybrid Neural network incorporated with optimization algorithm (HNN-MCS). The weight factor is globally optimized using the modified cuckoo search algorithm. Once the training is done the data are then tested inorder to check the prediction accuracy of the proposed systems.

3.2. Test case generation

Test cases are employed to test all feasible combinations in the application and as well it offers the user to simply replicate the steps that were assumed to expose a defect that as identified during test. Test cases can be charted directly and obtained from use cases examples. Moreover, when the test cases are produced early, Software Engineers can frequently discover ambiguities and inconsistencies in the requirements specification and design documents. The generated test cases will be fed to the advanced neural network for classification based on which the software reliability will be predicted.

3.3 Classification using HNN with MCS:

The Improved Artificial Neural Network is utilized to ascertain the license plate classification and it is trained by employing the features values which are extorted from each and every image. The improved artificial neural network is well trained by means of the extorted features. The innovative HNN is home to three input units, n hidden units and one output unit. The input of the neural network is the feature vector, which is extracted from the images. The network is trained under a large set of different license plate images in order to enable them to effectively classify the exact query image in the testing phase of neural network. The neural network works making use of two phases, one is the training phase and the other is the testing phase.

A. TRAINING PHASE

In the training phase, the input image is feature extracted and this feature vector is given as the input to the neural network. Initially, the nodes are given random weights. As the output is already known in the training phase, the output obtained from the neural network is compared to the original and weights are varied so as to reduce the error. This process is carried for a large number of images so as to yield a stable system having weights assigned in the nodes.

Multilayer feed forward neural network is utilized in our methodology. The structure is depicted in Fig. 2. The input layer has $|M|$ neurons i.e. number of matrix elements, the hidden layer has N_g neurons and the output layer has N neurons i.e. the number of characters ranging from A to Z and letters 0 to 9. Back propagation algorithm is used to train the neural network, which is described below.

Step 1: Generate arbitrary weights within the interval [0, 1] and assign it to the hidden layer neurons as well as the output layer neurons. Maintain a unity value weight for all neurons of the input layer.

Step 2: Input the training dataset I to the classifier and determine the BP error as follows

$$BP_{err} = C_{tar} - C_{out} \quad (1)$$

In Eq. (1), I_{tar} is the target output and I_{out} is the network output, which can be determined as $C_{out} = [Y_2^{(1)} \; Y_2^{(2)} \ldots Y_2^{(N)}]$, $Y_2^{(1)}, Y_2^{(2)}, \ldots, Y_2^{(N)}$ are the network outputs. The network outputs can be determined as

$$Y_2^{(l)} = \sum_{r=1}^{N_H} w_{2r1} Y_1(r) \quad (2)$$

where,

$$Y_1(r) = \frac{1}{1 + \exp(-w_{11r} \cdot C_{in})} \quad (3)$$

Eq. (2) and Eq. (3) represents the activation function performed in the output layer and hidden layer respectively.

Step 3: Adjust the weights of all neurons as $w = w + \Delta w$, where, Δw is the change in weight which can be determined as

$$\Delta w = \gamma . Y_2 . BP_{err} \quad (4)$$

In Eq. (4), γ is the learning rate, usually it ranges from 0.2 to 0.5.

Step 4: Repeat the process from step 2, until BP error gets minimized to a least value. Practically, the criterion to be satisfied is $BP_{err} < 0.1$.

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

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICRITCSA

M S Ramaiah Institute of Technology, Bangalore

Vol. 5, Special Issue 2, October 2016

B. TESTING PHASE

In the testing phase, the input image is fed to the trained neural network having particular weights in the nodes and the output is calculated so as classify the images based on the trained dataset. In ordinary neural network the process will be stopped after testing. In the proposed modified neural network, for testing process the optimization algorithm is incorporated in order to optimize the weight used for testing.

In our proposed method the weights are optimized with the help of the MCS (Modified Cuckoo Search Algorithm). By incorporating optimization process the classification accuracy will be improved there by providing better recognition of the images. The structure of the artificial neural network is illustrated in Fig.2.

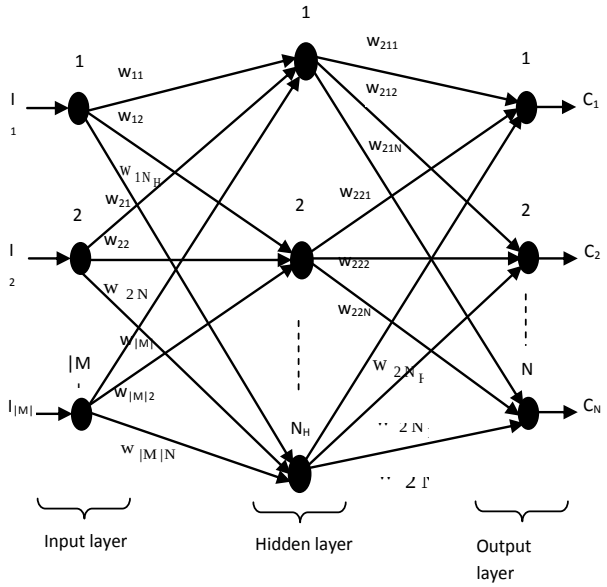

Fig 2: Structure of Artificial Neural Network

3.3 Weight Optimization Using Modified cuckoo search algorithm:

The Cuckoo search algorithm represents a meta-heuristic algorithm which owes its origin to the breeding conduct of the cuckoos and it is easy of implementation. There is a multitude of nests in the cuckoo search. Each egg signifies a solution and an egg of cuckoo corresponds to a novel solution.

The novel and superior solution replaces the most horrible solution in the nest. Similar to modified neural network, the ordinary cuckoo search algorithm is modified by including the Gauss distribution in the updation phase where levy flight equation is used. The gauss distribution adds better results of optimization when compared to the normal process. The modus operandi of the clustering procedure is shown as follows:

Step 1: Initialization Phase

The population (m_i, where i=1, 2, n) of host nest is initiated arbitrarily.

Step 2: Generating New Cuckoo Phase

With the help of the levy flights a cuckoo is selected randomly which generates novel solutions. Subsequently, the engendered cuckoo is evaluated by employing the objective function for ascertaining the excellence of the solutions.

Step 3: Fitness Evaluation Phase

The fitness function is evaluated in accordance with Equations 9 and 10 shown hereunder, followed by the selection of the best one.

$$P_{\max} = \frac{P_S}{p_T} \qquad (5)$$

$$fitness = \max imum \quad popularity = P_{\max} \qquad (6)$$

Where,

P_S - signifies the selected population

P_T - represents the total population

Step 4: Updation Phase

At the outset, the solution is optimized by the levy flights by employing the cosine transform. The quality of the novel solution is evaluated and a nest is selected arbitrarily from among them. If the quality of novel solution in the selected nest is superior to the previous solution, it is replaced by the novel solution (Cuckoo). Otherwise, the previous solution is treated as the best solution. The levy flights employed for the general cuckoo search algorithm is expressed by the Equation 7 shown below:

$$m_i^* = m_i^{(t+1)} = m_i^{(t)} + \alpha \oplus Levy\ (n) \qquad (7)$$

By suitably adapting Equation 7, levy flight equation using the gauss distribution is exhibited in Equation 8 here under:

$$m_i^* = m_i^{(t+1)} = m_i^{(t)} + \alpha \oplus \sigma_s \qquad (8)$$

Where,

$$\sigma_s = \sigma_0 \exp(-\mu\kappa) \qquad (9)$$

σ_0, μ - represents the constants

K – Symbolizes the current generation

Step 5: Reject Worst Nest Phase

In this section, the worst nests are ignored, in accordance with their possibility values and novel ones are constructed. Subsequently, depending upon their fitness function the best solutions are ranked. Thereafter, the best solutions are detected and marked as optimal solutions.

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

IJARCCE
International Journal of Advanced Research in Computer and Communication Engineering
ICRITCSA
M S Ramaiah Institute of Technology, Bangalore
Vol. 5, Special Issue 2, October 2016

Step 6: Stopping Criterion Phase
Till the achievement of the maximum iteration, the procedure is continued. By deftly employing the above-mentioned classification technique, it has been able to achieve superlative classification accuracy as the number of images classified is highly accurate in relation to certain modern techniques.

As an example consider a test case t1, which is at first generated from the input software. The testcase t1 is then applied to the classifier (hybrid neural network). The neural network is modified here in our proposed technique using MCS for optimization. So let us consider a weight value w1 as the input to the MCS algorithm. The fitness of the weight value w1 will be calculated by MCS using the levy flight equation and based on this fitness value we select the required weight values. The procedure is followed for all the generated testcases and number of weight values and finally we predict the reliability of the particular input software.

3.4 Software popularity and Maintainability

Once the optimizations of the testcases are done the next process in our proposed method is to estimate the software popularity and maintainability. The popularity is based on the amount of usage of the software among the customers. The maintainability of the software is the manner by which software can be adapted and it is regarded to be the major software quality feature.

The packages and packages with efferent coupling (Ce=0) and instability (I=0) are grouped under the dependable packages. The dependable and the Non-Dependable packages in any software are responsible for improving or reducing the maintainability of the software. Using the expression specified beneath the abstractness for the software is computed,

$$A = N_A / N_C$$

Where, N_A = total number of abstract class in the application, N_C = total number of classes in the application , Likewise the instability is computed by means of the beneath expression,

$$Instabilit \quad y(I_n) = \frac{c_e}{(c_e + c_t)} \qquad (11)$$

Where, c_e =Efferent Coupling, c_t = Total Coupling

4. RESULTS AND DISCUSSION

The proposed quality prediction technique is performed using the Hybrid neural network with optimization algorithm involving modified Cuckoo Search (HCS).The implementation done in JAVA platform with the Netbeans tool. The implementation through Netbeans is simple and provides better understanding of the code. The proposed system of quality prediction is done with the aid of Hybrid neural network technique. The major advantage of our proposed technique is the improved rate of predicting accuracy. Aldo by utilizing the optimization technique in the proposed classifier, the efficiency of the classifier is improved as the selection of weight factor is optimized leading to better classification.

The table 1 given below shows the fitness value of our proposed method with MCS method using different iterations. The fig 3 given below shows the comparison of the fitness value. The graph shows that our proposed method has delivered better fitness value which aids in improving the reliability of the software.

No of iterations	Fitness value	
	Before optimization	Optimization using MCS
5	9.625	12.658
10	8.256	11.354
15	7.568	10.235
20	6.235	10.121
25	5.521	9.654

Table.1 Fitness value for different iteration.

The fig 3 shows the comparison of the fitness value before optimization and our proposed method where HNN with MCS is used. The graph shows that our proposed method has delivered better fitness value which aids in improving the reliability of the software.

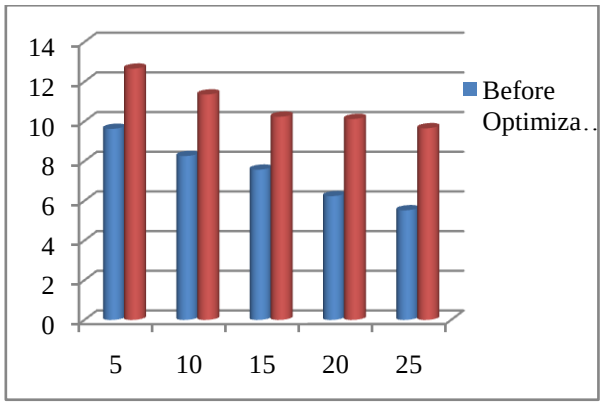

Fig 3: Graphical representation for iteration before and after optimization.

The performance of the proposed method is compared using the classification accuracy with that of some existing methods [16]. The table 3 given below shows the accuracy values for proposed and existing method.

Methods	Classification Accuracy (%)
SVM	89
QDA	85.49
Proposed HNN-MCS	91.3

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

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering

ICRITCSA

M S Ramaiah Institute of Technology, Bangalore

Vol. 5, Special Issue 2, October 2016

Table 3: Comparison of prediction accuracy measures for our proposed and existing method. The graphical representation for prediction accuracy measure for the proposed HNN-MCS algorithm and existing methods are shown in the below fig 4,

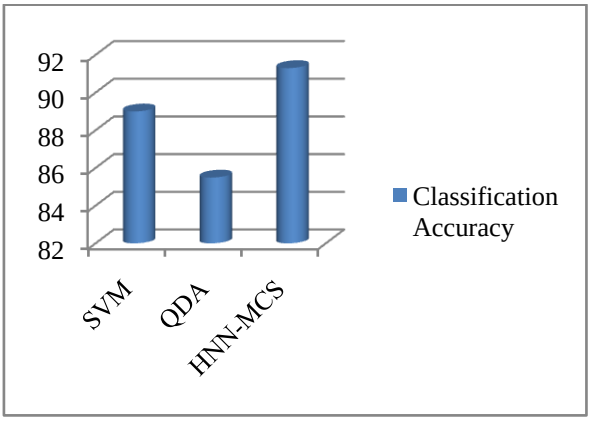

Fig 4: Graphical representation of prediction accuracy measures for our proposed and existing method

5. CONCLUSION AND FUTURE SCOPE

In this paper, it is proposed a system to predict better software reliability using HNN-MCS. The test cases are applied to the classifier which is incorporated with hybrid cuckoo search algorithm in order to make the classification process more accurate. Next, the reliability of the software is measured with the aid of popularity and maintainability.

From the comparative analysis it is clear that our proposed method achieved better reliability compared to other existing methods. In future, the work can be refined by suggesting some other classifier so that the defect predicting accuracy can be improved to some better extend.

REFERENCES

1. Chin-Yu Huang, Sy-Yen Kuo and Michael R. Lyu, "An Assessment of Testing-Effort Dependent Software Reliability Growth Empirical Software Engineering, Vol. 12, No. 2, pp. 161–182, Apr 2007.Models," IEEE Transactions on Reliability, Vol. 56, No. 2, pp. 198-211, Jun 2007.
2. Carina Andersson, "A replicated empirical study of a selection method for software reliability growth models," Journal of
3. Khurshid Ahmad Mir, "A Software Reliability Growth Model," Journal of Modern Mathematics and Statistics, Vol. 5, No. 1, pp. 13-16, 2011.
4. S. M. K. Quadri, N. Ahmad and Sheikh Umar Farooq, "Software Reliability Growth modeling with Generalized Exponential testing –effort and optimal Software Release policy," Global Journal of Computer Science and Technology, Vol. 11, No. 2, pp. 27-42, Feb 2011.
5. Jehad Al Dallal ,"Mathematical Validation of Object-Oriented Class Cohesion Metrics", International Journal Of Computers , Vol. 4,No.2, 2010.
6. Heikki Orsila, Jaco Geldenhuys, Anna Ruokonen and Imed Hammouda," Update Propagation Practices in Highly Reusable Open Source Components",In.proc.of 20th World Computer Congress on Open Source Software,Milano, Italy,Vol. 275 ,pp.159-170, Sep 7-10, 2008.
7. Chin-Yu Huang and Michael R. Lyu, "Optimal Release Time for Software Systems Considering Cost, Testing-Effort, and Test Efficiency", IEEE Transactions On Reliability, Vol.54, No.4, 2005.
8. Ingunn Myrtveit, Erik Stensrud and Martin Shepperd, "Reliability and Validity in Comparative Studies of Software Prediction Models", IEEE Transactions On Software Engineering, Vol.31, No.5, 2005.
9. Chao-Jung Hsu and Chin-Yu Huang, "An Adaptive Reliability Analysis Using Path Testing for Complex Component-Based Software Systems", IEEE Transactions On Reliability, Vol. 60, No. 1,2011.
10. Rita G. Al gargoor and Nada N. Saleem, "Software Reliability Prediction Using Artificial Techniques", IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 4, No 2,2013.
11. C. Jin, "Software reliability prediction based on support vector regression using a hybrid genetic algorithm and simulated annealing algorithm", IET Software, Vol.5, No.4, pp. 398–405, 2011.
12. Kirti Tyagi and Arun Sharma, "An adaptive neuro fuzzy model for estimating the reliability of component-based software systems", Applied Computing and Informatics, Vol.10, No.2, pp.38–51, 2014.
13. Lalit K. Singh, Gopika Vinod and Anil K. Tripathi, "Approach for parameter estimation in Markov model of software reliability for early prediction: a case study", IET Software, Vol. 9, No.3, pp.65–75,2015.
14. Cong Jina and Shu-Wei Jin, "Software reliability prediction model based on support vector regression with improved estimation of distribution algorithms", Applied Soft Computing, Vol.15, pp.113–120, 2014.
15. Jinhee Park and Jongmoon Baik, "Improving software reliability prediction through multi-criteria based dynamic model selection and combination", Journal of Systems and Software, Vol.101, pp.236-244, 2015.
16. Pratik Roy, G.S. Mahapatra and K.N. Dey, "Neuro- genetic approach on logistic model based software reliability prediction", Expert Systems with Applications, Vol.42, No.10, 2015.