SOFTWARE FAULT PREDICTION USING ARTIFICIAL NEURAL NETWORK AND RESILIENT BACK PROPAGATION

Devendra Mundada[#1], Aachal Murade[#2], Omkar Vaidya[#3], Swathi J. N.*
SCSE Department of VIT
Vellore Institute of Technology, Vellore, Tamil Nadu, 632014.
[#1]mundadadevendra.somnath2015@vit.ac.in
[#2]aachalsudhakar.murade2015@vit.ac.in
[#3]vaidyaomkar.ramakant2015@vit.ac.in
*jnswathi@vit.ac.in

ABSTRACT - Software engineering field contains various approaches related to prediction such as test effort prediction, correction cost prediction, fault prediction etc. Among these software fault prediction is the most popular research area and many new projects are started in this area. When there is an error in the computer program, it produces an invalid or false result. Hence prediction of defective modules is necessary to enhance the software quality. Various methods and metric sets are available to find out the false modules that are error prone. In this, Artificial Neural Network based software fault prediction technique is used. To find estimated solutions to optimization and search problems this method is used. Artificial Neural Network is used for finding the faulty elements as well as for predicting the erroneous modules.

KEYWORDS*:*Software fault, Artificial Neural Network, Classification, Defect Prediction, Back Propagation, Fault modules.

I. INTRODUCTION

Software fault prediction methods use previous software parameters and fault data to predict the faulted modules for the next release of software. For achieving the target of a software quality assurance initiative, software quality models are one of the useful tools. This model can be used to recognize program modules that are flawed. Software fault is an error in the coding that may lead to software to act not in the correct way and this may result in error and hence software failure. Software fault which has occurred due to programming error can be a recoverable error. Once such faults are detected they are sent to appropriate handler for performing required steps. Fault prediction will give an opportunity to the team to retest again the modules or files which are faulty or for which the probability of defectiveness is more. Spending more time on the defective modules instead of non defective ones results into proper resource utilization and this leads to maintenance of the project in easier way that is beneficial for both customers as well as project owners. The exact prediction of where the errors are probable to occur in code can help directly to test effort, enhance the software quality and reduce costs. A fault susceptible module is the one in which the quantity of faults is higher than selected threshold. Today the greatest challenge in the software industry is to make any application or software completely fault free to achieve the best software. As the defected modules are recognized, it is easier for the experts to focus only on the development work. Faulty modules can be easily predicted by classifying software modules into groups of faulty and non faulty modules at the beginning of development. Several algorithms have been used to predict the software faults such as k-means clustering and hierarchical algorithms. Bayes Network Classification Algorithm and spam filtering methods are also used for finding flawed modules. Support vector machine (SVM) and module dependency graphs (MDGs) are also helpful in predicting the faults. Among various algorithms Genetic Algorithms are useful for solving classification as well as regression problems. Therefore for predicting the quality it is important to test the ability of this algorithm.

The paper is organized as follows: section 2 discusses the related work, Section 3 explains about the empirical data collection and section 4 describes the GA based methodology. The result of the study is given in section 5. Finally conclusions of the research are presented in section 6.

II. Related Work

A review was carried out on software engineering mainly on software fault prediction. In this study nearly about 90 papers were investigated related to fault prediction between the years 1990 and 2009. Both statistical based methods and machine learning based methods are studied in this paper. After doing the survey it is found that most of the models were based on machine learning techniques. This will benefit researchers to obtain the knowledge about the fault prediction (Catal, Cagatay 2011). A multi-layer feed forward artificial neural network (ANN) based logistic growth curve model (LGCM) is proposed in this paper for software reliability and prediction. An artificial neural network is being developed by representation of different functions of hidden layer neurons. A neuro-genetic is proposed for the ANN based LGCM by reducing the weights of the network. In this the comparison of two learning algorithms is also presented to predict the software reliability (Pratik Roy et al. 2015). Investigation was carried out for designing goal oriented quality model with proper resource utilization using genetic based multi-objective method. It presents a case study which shows that which modules are fault-prone or not. Advantage of using genetic programming is that it does not require additional information regarding size and structure of the problem (Taghi M et al.2007). The main goal of this paper was to find and evaluate the studies regarding the metrics used in fault prediction. An extreme search was also done on digital libraries. From each study ten properties were drawn out and an assessment was done to get new insights. With the help of these studies researchers can find out frequently used metrics (DanijelRadjenovic et al. 2013). Research was done on data mining techniques that are used for searching the rules that show the modules that has the chance of being defective. Datasets are used from repository. First feature selection is applied on defective attributes and then genetic algorithm is used. Like this defective software modules are found out using data mining (J. C. Riquelme et al. 2009).

This paper focuses on software quality modelling by using multiple data repositories. It uses genetic-programming based approach for constructing optimal models with the help of datasets. Baseline Classifier, Validation Classifier, and Validation-and-Voting Classifier are the three techniques presented in this paper. The results show that last method is much better than the others and has the less probability for over fitting (Liu, Yi Cathy et al. 2010). Improper data is an important factor when constructing prediction models for high reliable systems. With skewed data this paper uses Roughly Balanced Bagging algorithm for fault prediction. For solving the problem it then merges data sampling and bagging. Then a comparison is done with the models without bagging. This is needed to explain these problems clearly (Seliya, Naeem et al. 2010). The effectiveness of Quad Tree based

K-Means clustering algorithm is compared with original K-means algorithm in predicting faulty modules. The goal of this paper is twofold. First, Quad trees are applied to find cluster centers that are given as input to K-means algorithm. Then for identifying faults algorithm is used (ParthaSarathiBishnu et al. 2012). This paper introduced a new genetic programming algorithm in the context of reliability modeling. After evaluating this new one the results show that it is less costly as compared to classical GP (Eduardo Oliveira Costa et al. 2010). Here predictability of reliability is measured by using group of models trained by GA. This study is applied to three sets and results show that models are linear in future (Sultan H. Aljahdali et al. 2009).

In this, near about 15 Bayesian network learners are compared. Among all the results shows that Augmented Naive Bayes classifiers obtains better performance as compared to others (Karel Dejaegeret al.2013). In a software company to minimize costs, fault and effort prediction are main tasks. However predictive performance is sacrificed. In this paper to solve this issue rule extraction is used (Julie Moeyersomsa et al. 2015). A hybrid IEDA-SVR is proposed in this paper. It is used to predict the software reliability. Here two data sets are used and a comparison with previous models is done (Cong J et al. 2014). Here a method is proposed that aims at avoiding the model-generalization problem. The plan is to reuse the existing the models to generate new ones having both general and specific knowledge (Salah Bouktif et al. 2010). This paper solves the problem of predicting effort that is needed to fix a defect. New four enhancements are proposed. It also shows the improvement over the results showed in literature and gives new ways for research under this area (AlaaHassounaand LadanTahvildari 2010).

III. Proposed Work

A. Artificial Neural Network

Artificial Neural network is defined as a system where data can be processed through a number of nodes similar to neurons in brain. Each node is assigned with a function and it determines the node output with the help of some parameters available locally to it for a set of given input. By adjusting weight of these parameters the node function can be altered as intended.

B. Neural Network Modeling

Like a brain, a neural network also performs a similar fashion. It has some learning mechanism designed within it for modeling the reliability. A number of nervous constitute Neural Network which is simple processing elements. These neurons are connected to each other directly through communications links associated with some weight. Supervised learning method is used to train the Neural Network with a series of sample input and to compare the resource overall for the pre specified period of time with the expected sample output. The training procedure is carried out until expected and convincing responses are provided by the network. The neurons are arranged layer by layer and the connection patterns within and in-between layers make the network architecture. He network can be either single-layered or multi-layered; layers of interconnected links between the neuron slabs determine it.

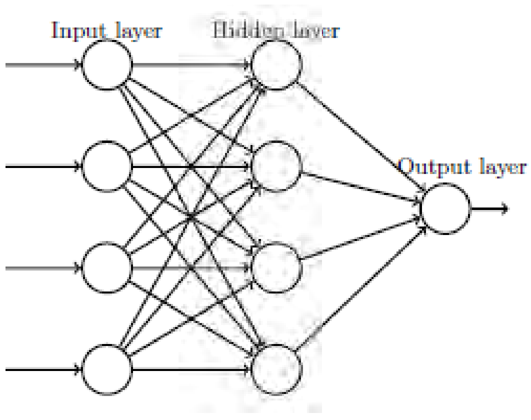

Fig-1: Neural Network with 3 Layers

IV. Predicting Models

In the past few years, hundreds of predicting models were introduced to estimate the fault tolerance of software. The issue of building growth models was the subject of many research works which helps in estimating the fault tolerance capacity of a software system before it is handover to the client or released. This paper mainly focuses on the fault tolerance of the software using artificial neural network using back propagation algorithm.

Algorithm:

1. Initialize the weights.
2. Repeat
   a. For each training pattern
   b. Train on that pattern
   c. Find fault for pattern and mean square error for total number of patterns
   d. Update the connecting weights by calculating fault layer by layer backward
3. End
4. Until the fault is acceptably low.

A. **Back Propagation Learning Algorithm**

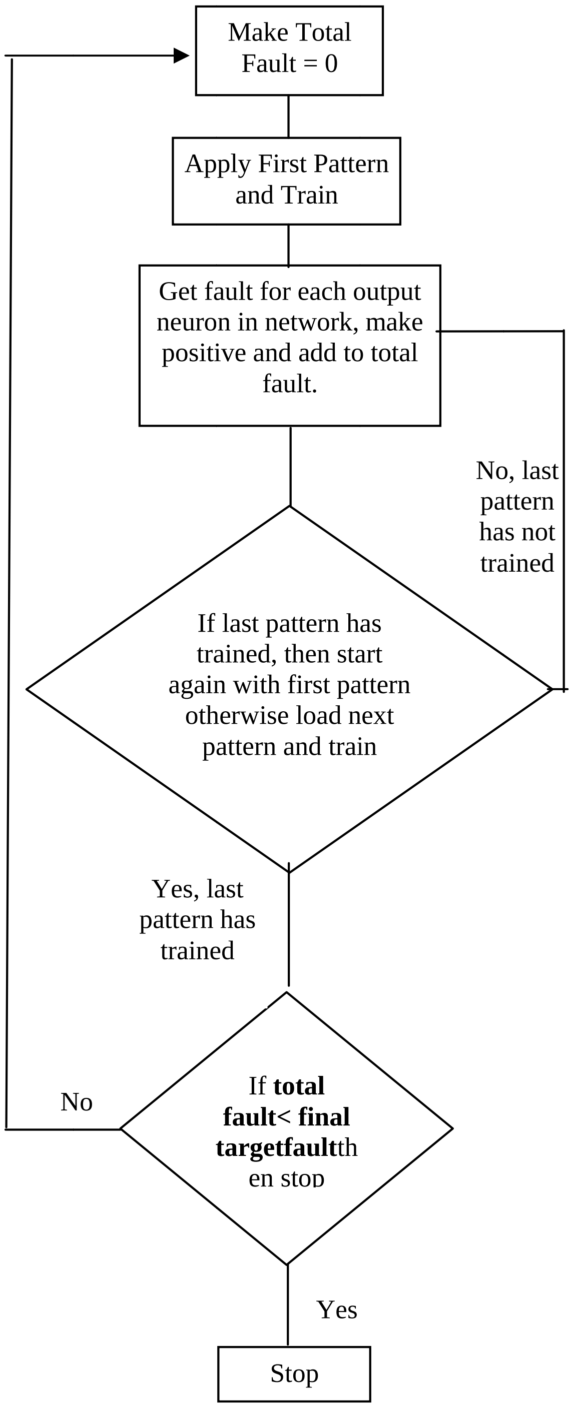
Make Total Fault = 0
Apply First Pattern and Train
Get fault for each output neuron in network, make positive and add to total fault.
No, last pattern has not trained
If last pattern has trained, then start again with first pattern otherwise load next pattern and train
Yes, last pattern has trained
No
If **total fault< final targetfault**then stop
Yes
Stop

Fig-2: Flowchart for Back Propagation Algorithm

B. **Gradient Descent:**

Gradient descent is one of the method for updating the weighted during learning phase. In this method first order derivate of the total error is used to find out the minima in the error space.

In this method the gradient vector that is to be added to weights is given as:

$$G = \frac{d}{dW}(E_k) = \frac{d}{dW}\left(\frac{1}{2}(y'_k - y_k)^2\right)$$

This gradient vector is added to every weight to get new, updated weight so that learning process will be easier.

The new updated weights are given by following equation:

$$W_{k+1} = W_k - aG_k$$

C. **Resilient Back Propagation Learning**

Resilient Back Propagation algorithm is used to train the neural network. The resilient back propagation algorithm does training of the network faster than the conventional back propagation training algorithms. In the resilient back propagation there is no need to specify the learning rate or any other parameter as conventional back propagation algorithms required for their implementation. In the RPROP, the size of the size of the weight-step is not depended on the magnitude but on the sequence of signs. Because of this layer in the network has equal chance of learning and to grow.

V. IMPLEMENTATION SETUP

In this paper the artificial neural network with back propagation algorithm is implemented using Python. The dataset used to implement the proposed algorithm is JM1/software defect prediction. This dataset is taken from http://promise.site.uottawa.ca/SERepository/datasets-page.html which is public repository.

The dataset consists of total of 498 tuples. Each tuple has 22 attributes.

Table-1: Dataset Attributes

No.	Attribute	Meaning
1	loc	line count
2	v(g)	Cyclomatic Complexity
3	ev(g)	Essential Complexity
4	iv(g)	Design Complexity
5	n	Total operators and oprands
6	v	Volume
7	l	Program Length
8	d	Difficulty
9	i	Intelligence
10	e	Effort
11	b	
12	t	Time estimator
13	loCode	Line Count
14	loComment	Count of line of Comments
16	loBlank	Count of Blank lines
17	unique_Op	Unique Oprators
18	unique_Opand	Unique Operands
19	total_op	Total Oprator
20	total_opand	Total Operands
21	branchCount	Flow graph
22	defects	{True, False}

VI. EXPERIMENTAL RESULT

We are using python as a programming language. *Numpy* and neurolab framework is used for implementation of the neural networks. The network consists of Input neurones, hidden layer and output neurones. Each neurone has some weight. This weight is adjust durning the training phase. We train the network using dataset. The network is train for 500 epochs. Sigmoid is used as activation function for the network.

The below figure shows error calculated at some epochs

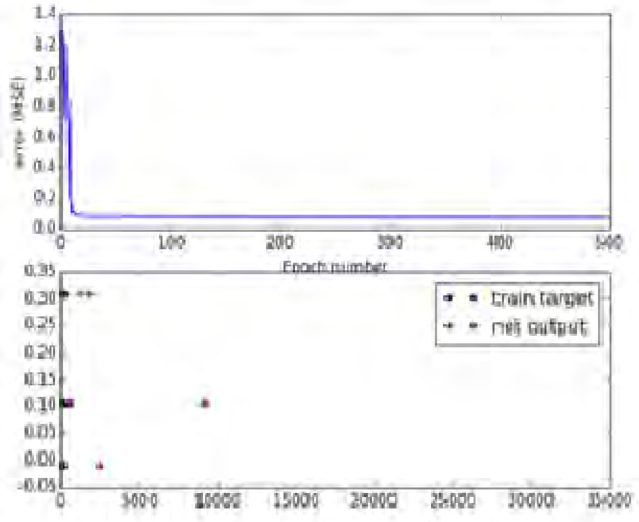

Fig-3: Graph of Error calculated at some epoch

If there is some error then weights of the each node is adjusted so as to minimize this error. From this graph we can say that after some epoch error become constant. So we can stop the training of the data. After the training is done we then test the network.

VII. CONCLUSION

This paper emphasis on the software fault prediction technique which is based on artificial neural network with back propagation learning algorithm. The observation and results conclude that the neural network model performs better in terms of less error in prediction as compared to existing analytical models and hence it is better to do software fault prediction tests using neural networks. However it can be seen that the neural Network method proposed in this paper using back propagation algorithm provide a good fit. As the connection weights are randomly initialized, thus the neural network gives different results for dataset and thus the performance of network varies. The usefulness of neural network is method is dependent on the nature of dataset up to greater extent.

REFERENCES

[1] Narayanan, Swathi Jamjala, et al. "Induction of fuzzy decision trees and its refinement using gradient projected-neuro-fuzzy decision tree." International Journal of Advanced Intelligence Paradigms 6.4 (2014): 346-369. (SCOPUS)

[2] Swathi J. Narayanan, RajenB.Bhatt, IlangoParamasivam, Khalid M, Tripathy B K, 2014, Gradient Projected Neuro Fuzzy Decision Tree, in Proceedings of Elsevier International conference on Advances in Communication, Network and Computing, Chennai, TamilNadu, Feb 21-22, pp. 126-132.

[3] Catal, Cagatay. "Software fault prediction: A literature review and current trends." Expert systems with applications 38.4 (2011): 4626-4636.

[4] Roy, Pratik, G. S. Mahapatra, and K. N. Dey. "Neuro-genetic approach on logistic model based software reliability prediction." Expert Systems with Applications 42.10 (2015): 4709-4718.

[5] Khoshgoftaar, Taghi M., and Yi Liu. "A multi-objective software quality classification model using genetic programming." Reliability, IEEE Transactions on 56.2 (2007): 237-245.

[6] Radjenović, Danijel, et al. "Software fault prediction metrics: A systematic literature review." Information and Software Technology 55.8 (2013): 1397-1418.

[7] Riquelme, J. C., et al. "Finding Defective Software Modules by Means of Data Mining Techniques." IEEE Latin America Transactions 7.3 (2009): 377-382.

[8] Liu, Yi Cathy, Taghi M. Khoshgoftaar, and NaeemSeliya. "Evolutionary optimization of software quality modeling with multiple repositories."Software Engineering, IEEE Transactions on 36.6 (2010): 852-864.

[9] Seliya, Naeem, Taghi M. Khoshgoftaar, and Jason Van Hulse. "Predicting faults in high assurance software." High-Assurance Systems Engineering (HASE), 2010 IEEE 12th International Symposium on. IEEE, 2010.

[10] Bishnu, ParthaSarathi, and VandanaBhattacherjee. "Software fault prediction using quad tree-based k-means clusteringalgorithm." Knowledge and Data Engineering, IEEE Transactions on 24.6 (2012): 1146-1150.

[11] Hiroki Costa, Eduardo Oliveira, Aurora Trinidad Ramirez Pozo, and Silvia Regina Vergilio. "A genetic programming approach for software reliability modeling."Reliability, IEEE Transactions on 59.1 (2010): 222-230.

[12] Aljahdali, Sultan H., and Mohammed E. El-Telbany. "Software reliability prediction using multi-objective genetic algorithm." Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on. IEEE, 2009.
[13] Dejaeger, Karel, Thomas Verbraken, and Bart Baesens. "Towardcomprehensible software faultpredictionmodelsusingbayesian network classifiers." Software Engineering, IEEE Transactions on 39.2 (2013): 237-257.
[14] Moeyersoms, Julie, et al. "Comprehensible software fault and effort prediction: A data mining approach." Journal of Systems and Software 100 (2015): 80-90.
[15] Jin, Cong, and Shu-Wei Jin. "Software reliability prediction model based on support vector regression with improved estimation of distribution algorithms." Applied Soft Computing 15 (2014): 113-120.
[16] Bouktif, Salah, et al. "A novel composite model approach to improve software quality prediction." Information and Software Technology 52.12 (2010): 1298-1311.
[17] HASSOUNA, ALAA, AND LADANTAHVILDARI. "AN EFFORT PREDICTION FRAMEWORK FOR SOFTWARE DEFECT CORRECTION." INFORMATION AND SOFTWARE TECHNOLOGY 52.2 (2010): 197-209.