International Journal of Research and Development in Applied Science and Engineering (IJRDASE)
ISSN: 2454-6844

An Automated Software Reliability Testing Approach using Artificial Intelligence Technique

Brijesh Pandey
Computer Science and Engg
Goel Group of Intitutions., Lucknow
brijesh48academics@gmail.com

Ritika Chandel
Computer Science and Engg
Goel Group of Intitutions., Lucknow

Abstract--Software reliability is one of the important factors which decide the quality of the software .Software is thoroughly checked and errors are removed before it is delivered to the client. A key factor in the success of a software project is achieving the best-possible software reliability. Software reliability is a mathematical model which ensures that software development has been done within cost and time and it will not cause failure under specified conditions. Different types of SRMs are used for different phases of the software development life-cycle. The present work is concerned with developing prediction model using a soft computing techniques, viz. Artificial Neural Network (ANN) as an automated agent to evaluate and analyze software testing and quality assessment. A logistic model to be used for testing and evaluating the reliability of a software package has been developed. Real software failure data has been used for the comparison of the proposed logistic models. The models predict the mean time between failure (MTBF) of software packages. These models are fast, having quick computation capability, been able to handle noisy data in the current study under consideration. From the analysis of the above results it is seen that the cumulative time between failure prediction model developed using Feed Forward Neural Network technique with back propagation training algorithm has been able to perform well. Also it is seen that it is better able to handle non linearity in the data. From amongst the training algorithms used, it was concluded that Levenberg-Marquardt algorithm was the best one to achieve the desired results

Keywords: Software Testing, Soft Computing Technique, ANN, MTBF

1. Introduction:

The basic goal of software development is to produce high quality software at low cost and within time. As the size and complexity of the software grows the issues with the reliability of the software also grows. So there is requirement of a software reliability model which ensures that the operation is failure free. It is very desirable to know the probability of software failure or the rate at which software errors will occur. SRGMs are mathematical models which describes how software attains reliability when the faults are detected and removed[1]. It indicates when software is ready to release and it has gain the expected reliability level[7]. Many SRGMs have been proposed in the past to estimate the expected number of total defects (or failures) or the expected number of remaining defects/ failures.

From the paper itself

Artificial neural networks (ANNs) have been used in the past to handle several aspects of software testing. Experiments have been conducted to evaluate the effectiveness of generating test cases capable of exposing faults, to use principle components analysis to find faults in a system, [6] to compare the capabilities of neural networks to other fault-exposing techniques,[5] [7] and to find faults in failure data. Hence prediction model is going to be developed using software failure data. As failure occurrences initiate the removal of faults, engineers reported failure times and time between failures (TBF). Both have been used to find the cummulative time between failures (CTBF), which is then used to investigate the reliability growth. Models that discuss the behaviour of CTBF are called SRMs.

2. Literature Review:

A review of the recent available literature on software testing and quality prediction is presented. It covers literature on software reliability models, reliability-relevant software metrics, software defect prediction model, and software quality prediction models and regression testing models. Deepam Agarwal, (2004), found out that how well the application under test conforms to its specifications. An ROC Analysis was carried out to compare the approaches. John E. Bentley, Wachovia Bank, Charlotte NC (2005), According to them software testing is often less formal and rigorous than it should, and reason for that is because the project staff is unfamiliar with software testing methodologies, approaches, and tools. They said that to overcome it every SAS professional should be familiar with basic software testing concepts, roles, and terminology. Mrs.Agasta Adline, Ramachandran. M(2014) Predicting the fault-proneness of program modules when the fault labels for modules are unavailable is a challenging task frequently raised in the software industry. They attempted to predict the fault–proneness of a program modules when fault labels for modules are not present. Xiaoxing Yang, et.al. (2014) Used the rank performance optimization technique for software forecasting model development. For this rank to learning approach was used. Rakesh Roshan, Rabins Porwal, Chandra Mani Sharma, (2012), reviewed the recent advancements in this field of Search Based Software

Testing. It covered the area of modern Software Testing. They showed that search based software test has many advantages including reduced efforts and improved reliability over state-of-the-art approaches of Software Testing. Animesh Kumar Rai, Rana Majumdar (2014), analysed different types of software reliability models and calculated failure rate of the software product. K.Venkata Subba Reddy and Dr.B.Raveendra Babu (2013) we propose a software reliability growth model, which relatively early in the testing and debugging phase, provides accurate parameters estimation, gives a very good failure behavior prediction and enable software developers to predict when to conclude testing, release the software and avoid over testing in order to cut the cost during the development and the maintenance of the software. Najia Saher, Dost Muhammad Khan, Faisal Shahzad, Ayesha Karim, analysed two points, that is at what point when ought to a test be automated and when it ought to be manual.

3. Data Used

Failure data during system testing phase of various projects collected at Bell Tele-phone Laboratories, Cyber Security and Information Systems In-formation Analysis Centre(CSIAC) by John D. Musa are considered.

Five numbers of application software testing data set for demonstration of predictive performance and prediction accuracy as shown in Table 1 has been considered. 70% of each dataset is used for training the model and the rest failure data is used for validating the model. The datasets are downloaded from [12].

Table 1: Different software failure datasets used

Project Code	Project Name	No. of Failures	Development Phases
SYS1	Real Time Command & Command System	136	System Test Operations
CSR1	Real Time Command & Command System	397	System Test Operations

4. Model Inputs and Structure

The modeling approach used to develop prediction model along with details on input and output parameters is given in this section. One among the foremost important steps within the development of any prediction model is that the choice of suitable input variables that may enable any classification model to successfully produce the specified results. Sensible understanding of the system into consideration is a crucial prerequisite for successful application of data driven approaches. Physical understanding of the method being studied ends up in more sensible choice of the input variables.

The model developed has Cumulative Time Between Failure (CTBF) prediction model, using FFNN algorithm, with Cumulative number of Failures is taken as input and Cumulative Time Between Failure (CTBF) is taken as output variable. The nomenclature used are as follows;

Table 2: Structure of Forecasting model both for Model-I

Project Code	Model Nomenclature Used	Input Variables	Output Variable
SYS1	M1-SYS1	No. of Failures	CTBF
CSR1	M1-CSR1	No. of Failures	CTBF

5. Artificial Neural Network (ANN) Model Development:

Here optimal network geometry was investigated, using trial and error approach in an attempt to create more optimum model. The number of hidden nodes was used to guide the trial and error search approach for the optimal geometry [8]; however since an optimum model has been sought, only models containing less than ten hidden nodes were considered. Thus to minimise the number of networks that required training and testing, ANN's containing 1 to 10 nodes were considered in order to narrow down the search. Once this range was determined, the trial and error approach was repeated, with the number of hidden nodes increasing in increment of one from minimum nodes onwards. Finally the optimum nodes were found for the best developed network and the networks on either side of the best developed network were also tested. The models developed in this research contained a single hidden layer with sigmoid logistic activation function in the hidden nodes and output node. The number of nodes in the input layer has been kept fixed to three in all the six models considered [4][8]. The learning rate was also initially kept to minimum and slowly increased. Further in order to resolve the contradiction of using slow or high learning rate (η), a momentum (φ) term was introduced which provided a built in inertia allowing a slow learning rate but faster learning. Thus various permutation and combinations of both these factors were used during the training process. The fixed period stops of 1000 cycles was used for training the network and the target error was set to stop during training when the average error reaches below 0.00999. Model parameter values for Back Propagation Algorithm are given below in Table 3.

Table. 3: Model parameter values for Back Propagation Algorithm for both the models

Parameters	Range of values
Training Function	'trainlm'
Adaptation Learning Function	'learnGD'
Training mode	Supervise
Gradient mode	Jacobian
Performance Function	MSE, SSE, MAE
Transfer Function	For Hidden layer – tansigmoid For output layer - linear
Number of Hidden nodes	2-5
Learning Rate (η)	0.1 to 0.9
Momemtum (φ)	0.1 to 0.9
No. of epochs	100

Various network architectures were investigated in order to determine the optimal MLP architecture (i.e. the lowest mean square error and the optimum regression value) for the given combination of sixteen input variables. Different training algorithms were used with changes in the number of neurons in the hidden layers. In addition, the effect of transfer functions i.e. tangent sigmoid in the hidden layer were also investigated.

6. Results and Discussions:

Given below in Table 4 are the error values for best developed Model obtained using 1-5-1 network configuration for both the datasets.

Table 4: Error Values for Model

Model No.	MAE	MSE	SSE
M1-SYS1	0.0259	0.0013	0.0798
M1-CSR1	0.0191	0.000651	0.0976

Once the training process is complete and the results are obtained, then their accuracy is ascertained by using the testing data such that the predicted results are very close to the observed ones.

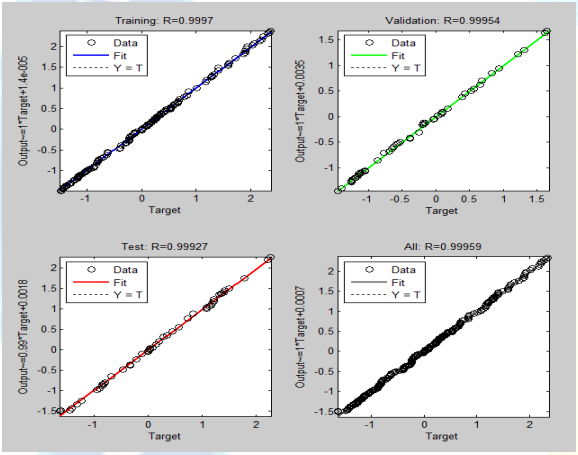

Fig. 1. Regression Plot during training for M1-CSR1 Model

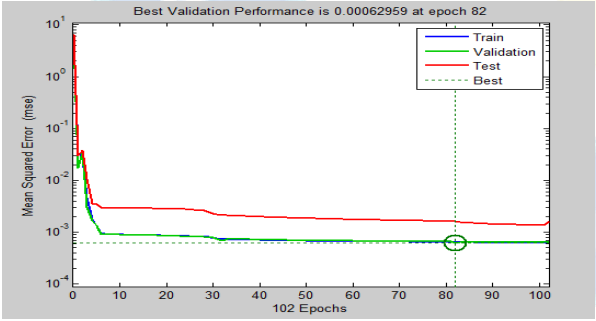

Fig. 2. Depicts the training of NN Model gauged by MSE for N=5 using Levenberg-Marquardt training algorithm with Gradient Descent with momentum as training function for M1-CSR1 Model

Fig 1 shows the regression plot of training, testing and validation results, whereas on the other hand and Fig. 2 demonstrates the plot of MSE for the bet developed ANN model, i.e. 1-5-1, using Levenberg Marquardt training algorithm for best developed M1-CSR1Model. For finding the accuracy of the models during training, testing and validation stage, MSE criteria is used. As seen from figures 4.2 the performance for the model improved even when the network error was low. It was noticed that in case of M1-CSR1 initially there was sharp fall in the nerwork error and at epoch 5 errors for training, testing and validation datasets got saturated and best validation result was obtained as 0.00062959 at epoch number 82. The network model 1-5-1, which is the best can be considered as the best selection of network topology. This topology is able to maintain the number of layers, processing elements, generalization characteristics. This also can be seen that the training time elapsed has also been reduced due to less iteration required as each time.

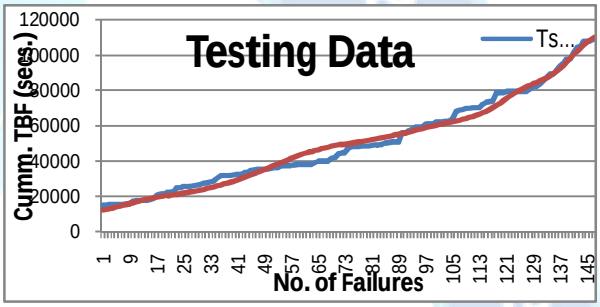

Fig. 3. Plot of Observed Vs. Predicted CTBF during testing for M1-CSR1 Model

Fig. 3 for Model M1-CSR1 shows the comparative analysis of the observed and forecasted values of the Cummulative TBF, for testing datasets, by the optimum model developed. This model used LM training algorithm and GDM training function for N=5. Here it is seen that the observed values and predicted values are almost similar, except for few instances. This demonstrates the validity of the ANN technique for the development of a suitable prediction model. Hence the inference drawn from the obtained results are that ANN models have been able to achieve the desired output.

From detailed analysis of the model using one input variable it can be seen that in this case data CSR1 has resulted in the development of better prediction model.

7. Conclusion:

In the present work feed forward neural network with back propagation learning algorithm, as an automated agent has been successfully implemented. The observations conclude that neural network model performs better in terms of less error in prediction as compared to existing analytical models and hence it is a better alternative to do software testing and reliability assessment. As the connection weights are randomly initialized, thus the neural network gives different results for the same datasets and thus the performance of the network varies. The neural network is shown to be a promising method of testing a software application provided that the training data have a good coverage of the input range. The back propagation method of training the neural

network is a relatively rigorous method capable of generalization, and one of its properties ensures that the network can be updated by learning new data. As the software that the network is trained to simulate is updated, so too can the trained neural network learn to classify the new data. Thus, the neural network is capable of learning new versions of evolving software.

The work involves application of ANN for the development of cumulative time between failures (CTBF) prediction models using six different datasets collected during system testing phase of various projects at Bell Tele-phone Laboratories, Cyber Security and Information Systems In-formation Analysis Centre(CSIAC). The model development phase involves the development of the model using only one input variable, number of failures, has been considered. Overall analysis of the results as given above demonstrates that Neural Network method is dependent on the nature of dataset up to a greater extent. Neural Network model gives better result for larger datasets than smaller datasets. These models are easily compatible with different smooth trend data set and projects. The program has been implemented in MATLAB.

References:

[1]. Hassoun, m.h.,(2002), fundamentals of artificial neural network, printice-hall of india,pp.599-606.
[2]. Zhang, g. P., (2003), time series forecasting using a hybrid arima and neural network model, neurocomputing,50,pp 159–175.
[3]. Huang, y.,(2009),advances in artificial neural networks – methodological development and application, algorithms, pp. 973-1007.
[4]. Macleod, c. An introduction to practical neural networks and genetic algorithms for scientists and engineers.
[5]. Maier, h.r., (1995), a review of artificial neural network., research report no. R131. Dept. Of civil and env.engg. The university of adelaide.
[6]. Jogi john, (2011), “a performance based study of software testing using artificial neural network”, international journal of logic based intelligent systems, vol. 1, no. 1,, pp. 45-60.
[7]. Abdelelah m. Mostafa, (2006), “regression approach to software reliability models”, graduate theses and dissertations, university of south florida.
[8]. Shaik nafeez umar, (2013), “ software testing defect prediction model-a practical approach”, international journal of research in engineering and technology, volume: 02 issue: 05, pp. 741-745.
[9]. Najia saher, dost muhammad khan, faisal shahzad, ayesha karim, “ the quality assessment of software testing procedure and its effects”.
[10]. Animesh kumar rai, rana majumdar (2014), “software reliability models: failure rate estimation”, international journal of latest trends in engineering and technology (ijltet) vol. 4, vol. 4 issue 1, pp. 20-25.
[11]. Voas jm, mcgraw g. Software fault injection; 1998.
[12]. http://www.cse.cuhk.edu.hk/~lyu/book/reliability/data.html.