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

An Automated Software Testing Approach using Soft Computing Technique

Shweta Srivastava
Computer Science and engineering
AIET, UPTU, Lucknow
pearlshweta710@gmail.com

Masood Ahmad
Computer Science and engineering
AIET, UPTU, Lucknow
ermasood@gmail.com

Abstract--Today, testing is the most challenging and dominating activity used by industry, therefore, improvement in its effectiveness, both with respect to the time and resources, is taken as a major factor by many researchers. Using an automated Agent to support the activities of human testers can reduce the actual cost of the testing process and the related maintenance costs. 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

Almost 50% of the software production development cost is expended in software testing. It consumes resources and adds nothing to the product in terms of functionality. Therefore, much effort has been spent in the development of automatic software testing tools in order to significantly reduce the cost of developing software[6]. A test data generator is a tool, which supports and helps the program tester to produce test data for software. Ideally, testing software guarantees the absence of errors in the software, but in reality it only reveals the presence of software errors but never guarantees their absence. One objective of software testing is to find errors and program structure faults. However, a problem might be to decide when to stop testing the software, e.g. if no errors are found or, how long does one keep looking, if several errors are found. Software testing is one of the main feasible methods to increase the confidence of the programmers in the correctness and reliability of software. Sometimes, programs that are poorly tested perform correctly for months and even years before some input sets reveal the presence of serious errors. Incorrect software that is released to market without being fully tested could result in customer dissatisfaction and moreover it is vitally important for software in critical applications that it is free of software faults which might lead to heavy financial loss or even endanger lives. Nowadays testing tools can automatically generate test data that will satisfy certain criteria, such as branch testing, path testing, etc. However, these tools have problems, when complicated software is tested. A testing tool should be general, robust and generate the right test data corresponding to the testing criteria for use in the real world of software testing. [1]

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.
Two 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

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. Two types of prediction models have been developed.

MODEL-I

One is the development of 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. Here two different prediction models have been developed using two different datasets. 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

MODEL-II

Here interpretation has been carried out with no of failures as a function of cumulative execution time. The nomenclature used for different models are tabulated as below.

Table 3: Structure of Forecasting model both for Model-II

Project Code	Model Nomenclature Used	Input Variables	Output Variable
SYS1	M2-SYS1	CTBF(t-2), CTBF(t-1), CTBF(t)	CTBF(t+1)
CSR1	M2-CSR1	CTBF(t-2), CTBF(t-1), CTBF(t)	CTBF(t+1)

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 4.

Table 4: 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.

4. Results and Discussions:

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

Table 5: Error Values for Model-I

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

Further Table 6 depicts the error values for best developed Model-II obtained using 3-5-1 network configuration for both the datasets.

Table 6: Error Values for Model-II

Model No.	**MAE**	**MSE**	**SSE**
M2-SYS1	0.0202	0.000904	0.0542
M2-CSR1	***0.0128***	***0.000293***	***0.0439***

A comparative error plots of both the models using one and three input variables is given in figures 1 to 3 below.

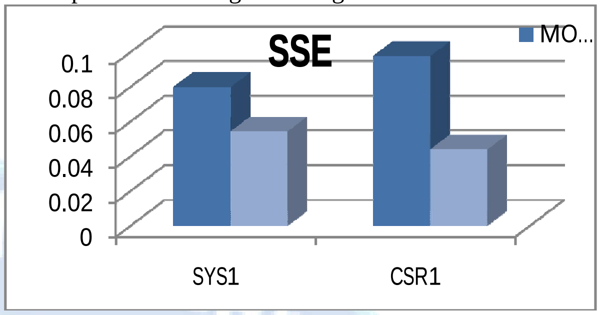

Fig. 1: SSE Plot for M-1 & M-2 models

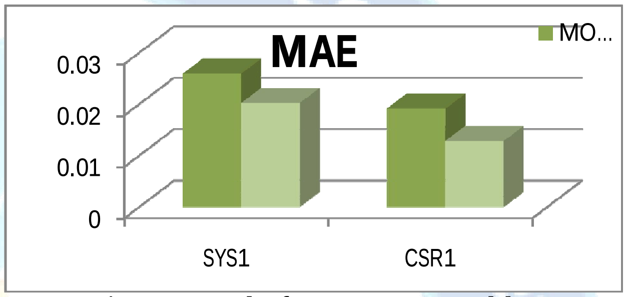

Fig. 2: MAE Plot for M-1 & M-2 models

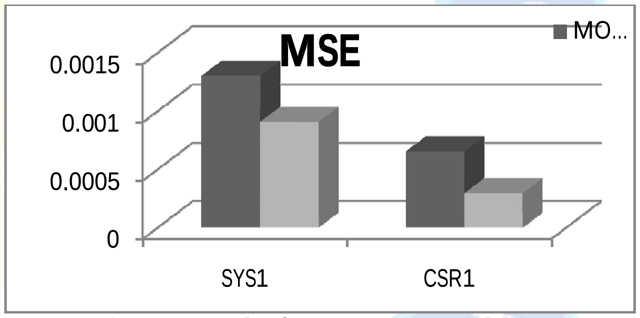

Fig. 3: MSE Plot for M-1 & M-2 models

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.

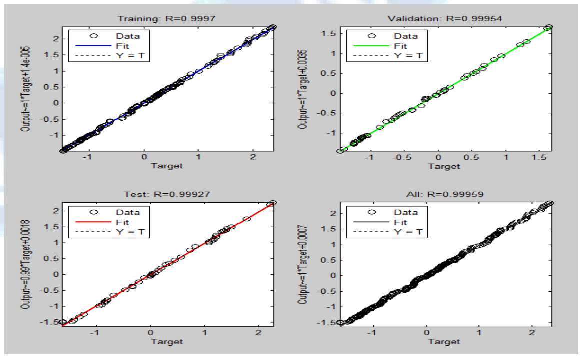

Fig. 4: Regression Plot during training for M1-CSR1 Model

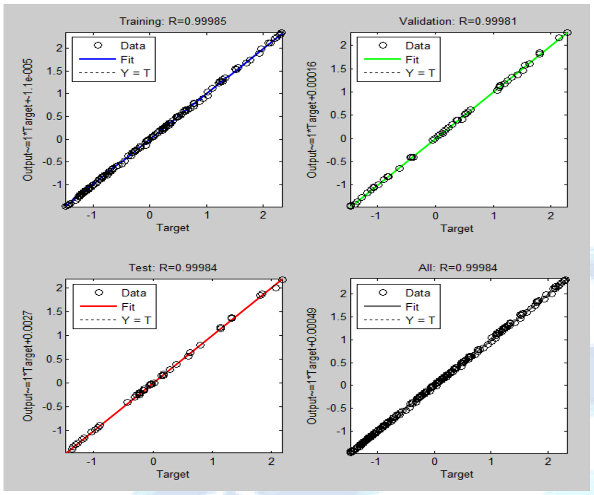

Fig. 5: Regression Plot during training for M2-CSR1 Model

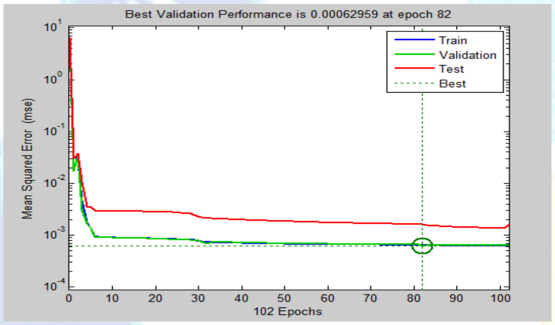

Fig. 6: Depicts the training of NN Model gauged by MSE for N=5 using Levenberg-Marquardt training algorithm with Gradient Descent with momentum as training function for M1-CSR1 Model

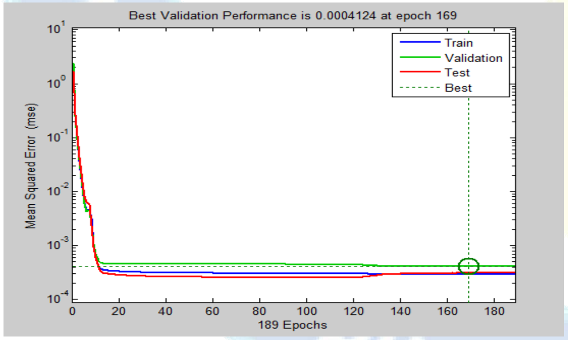

Fig. 7: Depicts the training of NN Model gauged by MSE for N=5 using Levenberg-Marquardt training algorithm with Gradient Descent with momentum as training function for for M2-CSR1 Model

Fig 4 and Fig. 5 shows the regression plot of training, testing and validation results, whereas on the other hand fig 6 and Fig. 7 demonstrates the plot of MSE for the bet developed ANN model, i.e. 1-5-1 and 3-5-1, using Levenberg Marquardt training algorithm for best developed M1-CSR1and M2-CSR1 Models. For finding the accuracy of the models during training, testing and validation stage, MSE criteria is used. As seen from figures 4.6 and 4.7 the performance in case of both the models 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, while in case of M2-CSR1 Model, though the initial fall in error values was the same but around epoch 10 it got saturated and the best validation result obtained was 0.0004124 at epoch number 169. The network model 1-5-1 and 3-5-1, which are 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.

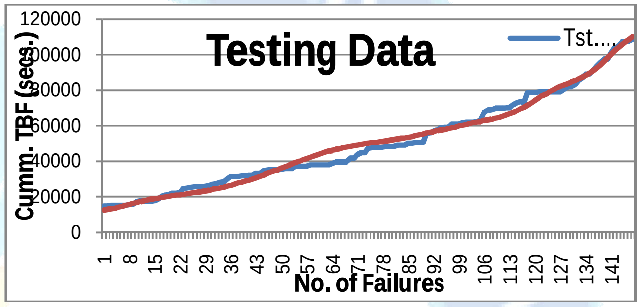

Fig. 8: Plot of Observed Vs. Predicted CTBF during testing for M1-CSR1 Model

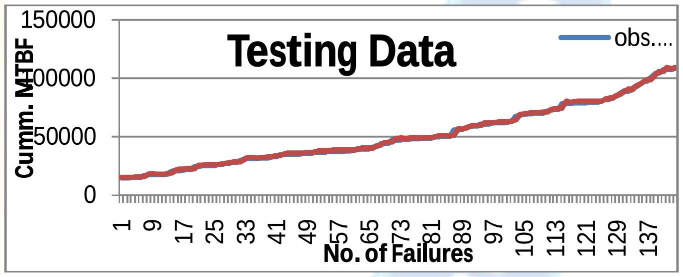

Fig. 9: Plot of Observed Vs. Predicted CTBF during testing for M2-CSR1 Model

Figures 4.8 and 4.9, for Models M1-CSR1 and M2-CSR1 respectively 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 two models M1 and M2, using one and three input variables respectively, it can be seen that in both the cases data CSR1 has resulted in the development of better prediction model. Further, the comparative plots of both the models with respect to MSE, SSE and MAE, as given in Figs. 4.1 to 4.3, clearly demonstrates that the model developed with three input variables, considering only the CTBF as data and output as one step ahead CTBF has shown greater accuracy in terms of various performance criteria, as against Model-I, which considers number of failures as input variable and CTBF as predictor variable.

5.Conclusion

In this study, applicability and suitability of ANN technique for the development of better cumulative time between failure prediction model has been demonstrated. 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. The work has been carried out using MATLAB environment. From amongst the training algorithms used, it was concluded that Levenberg-Marquardt algorithm was the best one to achieve the desired results. Here also it was observed that varying the number of hidden layer neurons from 2 to 10 showed variation in the prediction accuracy of the model and the best model architecture obtained was 1-5-1 and 3-5-1 for both the models. Further, comparative study of Models MI-SYS1, MI-CSR1 with that of MII-SYS1, MII-CSR1 showed that model MII -CSR1 ha better prediction accuracy than the rest of the models. Also, dataset CSR1 has resulted in better accuracy than SYS1 dataset.

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" sci-int.com.

[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.