Conceptual Software Reliability Model using Neural Network

Meenu
Ph.D. Scholar
Mewar University
Chittorgarh, Rajasthan,India

Sumeet
Research Guide
Mewar University Chittorgarh, Rajasthan,India

ABSTRACT
Reliability is the one of the most important attribute of the software for customer satisfaction. It is big challenge for the software development organizations to achieve the reliability of the software. Since research on software reliability is being carried out for last three decades and various software reliability models have been developed. These models calculate the reliability of the software and help to take decision to deploy the software or continue the testing process to meet the reliability objective. The models used so far, worked on the basis of some conventions which must be made before the beginning of the project like software development environment, the nature of software failures, the probability of individual failures. Recent research in the field of the Neural Network can also be applied to calculate the software reliability. The best thing to use Neural Network is to calculate the reliability without making any basic assumptions. In this paper, a conceptual model is proposed to develop the software reliability model using the approach of Neural Network.

Keywords
Software Reliability, Software Reliability Growth Models, Neural Network.

1. INTRODUCTION
Software reliability is defined as probability of failure free operation for a specified period of time in a specified environment. [ANSI 91][1]. A failure is the departure of the external results of system operation from user needs and is dynamic in nature. It relates to the behavior of the system. A fault in software is the defect in the program which when executed under particular conditions causes a failure. Failures may be caused by the different set of failures. A fault can cause more than one failure. Fault is a property of program rather than the property of its execution or behavior. Software fault is a defect, missing instruction or extra instructions or set of related instructions that can cause one or more actual. [2]

Reliability of software is measured in terms of failures which are a departure of program operation from program requirements. The software reliability is characterized as a function of failures experienced [3]. Software reliability is one of the important factor been considered while ensuring the software quality. Software reliability deals with the failure or faults that exist in the system. [7]

The process of finding and removing faults to improve the software reliability can be described by a mathematical relationship called a Software Reliability Growth Model (SRGM). [4]Software Reliability Model is used to determine whether the reliability of the software meets with the desired reliability as per the demand of the user by utilizing the software failure data or software testing data. By using Software Reliability Model, we can easily measure & predict the software reliability and can plot the software reliability growth charts. The charts of software reliability growth depict the trends that are used to forecast software failures as a function of calendar time. Besides, the charts can also us to determine the additional time needed to meet the reliability requirements and the associated costs. [5] The ability to predict the reliability of a software system would enable project management to better perform product assurance and assess reliability for release. [14]

These models on the basis of test data predict the software reliability. These models try to show a relationship between test data and mathematical functions. [6] SRGM shows how software reliability improves as the faults are detected and repaired. SRGM can be used to predict when a particular level of reliability is to be attained. Thus SRGM is used to determine when to stop testing to attain a given reliability level. There are many software reliability growth models like GO model, JM model, S- Shaped model etc. [9]

Existing models are based on the some assumptions about development environments, the nature of software failures, and the probability of individual failures occurring. All these assumptions must be made before the project begins so selecting an appropriate model in a particular environment may be a complicated and challenging task. Using Neural Network, a software reliability model can be developed without making any prior assumptions. This model can use the failure history as input data and will utilize the failure history to predict the future failures, so will help to estimate software reliability and can be applied in any environment. [8]

2. PROPOSED MODEL
2.1 Basic Architecture
The proposed Software Reliability model is based upon the architecture of Neural Network. The neural networks are made up of layers of neurons connected with each other. One layer receives input from the preceding layer of neurons and passes the output on to the subsequent layer. Input Layer, Output Layer & Hidden Layer are the basic layers in the architecture of the model. Input Layer accepts the input from the external world and passes the input to next layer without performing any useful calculations. Each unit in the input layer acts as a distribution point for the external inputs. Output layer presents the output or response to the external world. The units that output the network's response to the external world constitute the output layer. Hidden layers are the intermediate layers i.e. their existence is between the input and output layers. These layers have no direct communication with the external world. The number of layers in a network may vary from a lower limit of two (one input & one output layer) to any higher positive integer. In single layer architecture, there is no hidden layer i.e. the network is constituted with the help of input and output layers only. In Multilayer architecture, there are one or more hidden layers with Input and output layers. [10, 12]

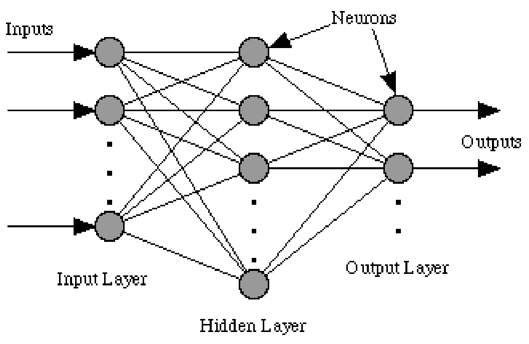

Fig 1: Neural Network

Basically Neural Network Provides the two types of connectivity options. Feed Forward Network and recurrent Network. In this proposed model, Feed forward Network will be used which either can be single Layer Network or Multilayer. The basic property of feed forward network is to propagate activations only in the forward directions.

For the conceptual Software Reliability Model, we can use Multilayer Model. The Network have one Input, one Output and Number of hidden layers. For the input layer, Cumulative execution time will be considered as the Input Data and Cumulative faults may be considered as output from the output layer or Target Layer. Hidden Layers may have n number of neurons. The output of Input Layer serves as the input of the Hidden Layer & the output of Hidden Layer is passed to the output Layer.

Further, to reduce the complexity of the network i.e. if initially we don't want to declare the number of hidden layers then Cascade - Correlation Network may be used. In this type of Network first we take only Input & Output layer is network and then the network dynamically creates the hidden layers according to the requirement.

2.2 Learning Algorithm

To solve a problem using neural network, the network must first be taught solutions using a set of typical instances of input-output pairs known as training set. The procedure by which the network is taught is known as Learning Algorithm. During the training or learning phase, the strength of the interconnection links of the network are adjusted to reduce the residual error resulting from the training set.

There are various learning algorithms. To train the conceptual model, Back Propagation Learning method will be adopted.

Back Propagation Learning Algorithm – This algorithm is a class of supervised learning algorithm in which network weights are iteratively adapted using errors propagated back from the output layer.

1. Initialization Phase - Initialize the network weights with a set of random values. A set of random values drawn from a small interval is used and even the network weights can be initialized with a set of values from a known distribution.

2. Weight Adjustment Phase – adjust the network weights incrementally over several iterations. For each iteration, adjust the weights in the direction of steepest decreasing gradient of the error surface i.e. the surface formed by the sum of the square of the error between the desired output & the actual output for all patterns in the training set. This iterative adjustment continues until either a minimum is reached in which the error is less than a pre-specified tolerance limit or until a set number of iterations has been reached. During each epoch, the algorithm presents the network with a sequence of training pairs.

The algorithm then calculates a sum squared error between the desired outputs and the network's actual outputs. It uses the gradient of the sum squared error (with respect to weights) to adapt the network weights so that the error measure is smaller in future epochs. Training terminates when the sum squared error is below a specified tolerance limit. [10-13]

2.3 Training Data

The network is trained with the help of training data which may be collected during the testing.During training, each input it at time t is associated with the corresponding output Ot. Thus the network learns to model the actual functionality between the independent (or input) variable and the dependent (or output) variable.

List of cumulative execution times (e1,…ek) belongs to Ek(t) and the corresponding observed accumulated faults (f1, ..., fk) belongs to Fk(t) up to the present time t, and the cumulative execution time at the end of a future test session k+h,,ek+h(t+d) predict the corresponding cumulative faults Fk+h(t+d). [11]

Table 1. Data Set of Project Failure

Hour	Cumulative Faults	Hour	Cumulative Faults	Hour	Cumulative Faults
1	27	10	93	19	128
2	43	11	97	20	129
3	54	12	104	21	131
4	64	13	106	22	132
5	75	14	111	23	134
6	82	15	116	24	135
7	84	16	122	25	136
8	89	17	122		
9	92	18	127		

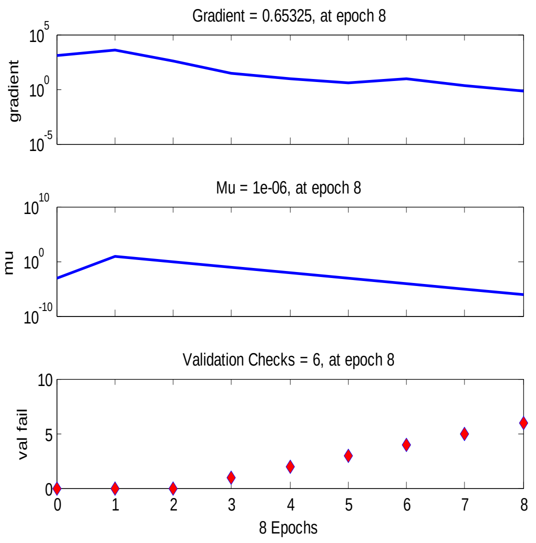

Fig 2: Training States

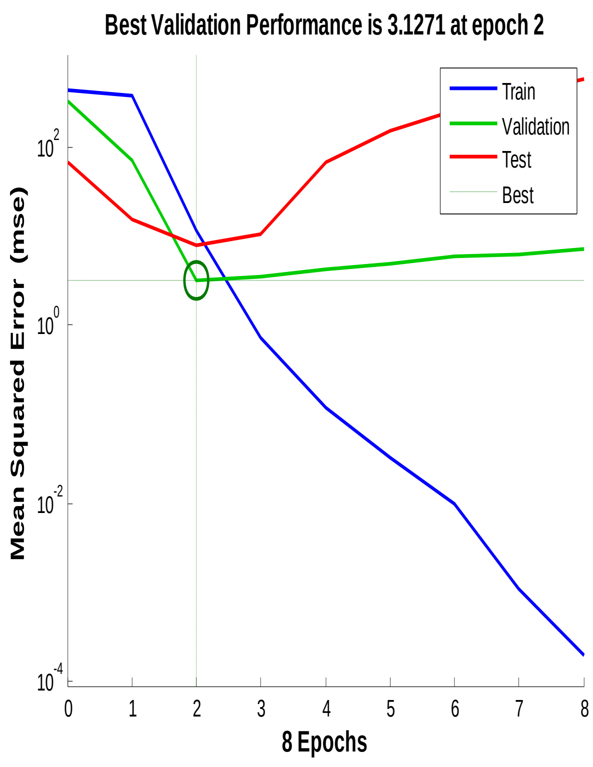

Fig 3: Best Validation Performance

2.4 Neural Network Software Reliability Growth Model

The Predictive capabilities of Neural Network depends upon the failure data, the input and output data given to the model and the sequence in which the input & output values are presented to the network during training and the complexity of the Network. Software Reliability growth prediction can be expressed in terms of neural network mapping as:

This reliability-prediction problem can be stated in terms of a neural network mapping:

P: {(Ek(t), Fk(t)), ek+h(t+d))} -> Fk+h(t+d)

where (Ek(t),Fk(t)) represents the failure history of the software system at time t used in training the network and Fk+h(t+d) is the network's prediction.

For the prediction horizon h=1, the prediction is called the next-step prediction (also known as short-term prediction), and for h=n (> 2) consecutive test intervals, it is known as the n-step-ahead prediction, or long-term prediction. A type of long-term prediction is endpoint prediction which involves predicting an output for some future fixed point in time.

Prediction of the number of accumulated faults can be done after some specified amount of testing. From the predicted accumulated faults, both the current reliability and how much testing may be needed to meet the particular reliability criterion. [1, 15]

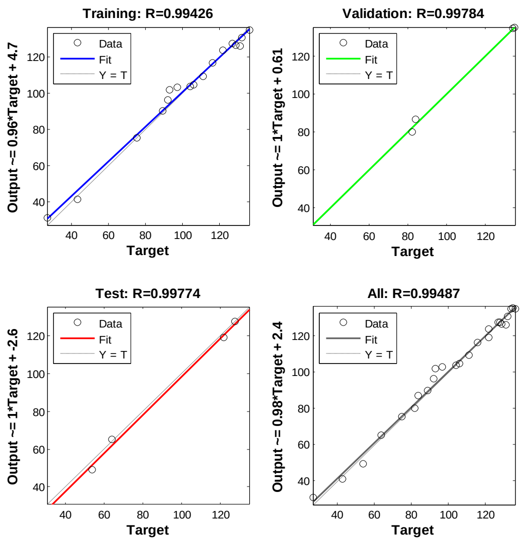

Fig 4: Regression

3. CONCLUSION

Thesteps of developing a neural network for reliability prediction are specifying suitable network architecture, choosing the training data, and training the network. Neural Network models require only failure history data as input with no prior assumptions. Using input, the neural-network model automatically develops its own internal model of the failure process and predicts future failures. Because it adjusts model complexity to match the complexity of the failure history, it can be more accurate than some commonly used analytical models.

4. REFERENCES

[1] Lyu, M. R.,Handbook of Software Reliability Engineering.

[2] Musa, John D., Software Reliability Engineering: More Reliable Software Faster and Cheaper, 2nd Edition.

[3] Rita G. Al gargoor, Nada N. Saleem , Software Reliability Prediction Using Artificial Techniques, IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 4, No 2, July 2013

[4] Almering V., Genuchten M. V, and Cloudt G., "Using Software Reliability Growth Models in Practice" IEEE computer society, 2007, pp. 82 – 88.

[5] Changjie Ma, Guochang Gu, Jing Zhao, Improved Neural Network based on Dynamic Predication Model of Software Reliability, Journal of Convergence Information Technology, Volume6, Number7, July 2011.

[6] Al-Rahamneh Z., Reyalat M. Sheta A. F., Bani-Ahmad S., and Al-Oqeili S., "A New Software Reliability Growth Model: Genetic-Programming-Based Approach" , Journal ofSoftware Engineering and Applications, 2011, pp. 476-481

[7] Haque F., and Bansal, S. "Software Reliability estimationModels: A Comparative Analysis" , InternationalJournal of Computer Applications, Vol. 43,2012, pp. 27 – 31

[8] Karunanithi, N., Whitley, D., & Malaiya, Y. K. (1992a), Using neural networks in reliability prediction.IEEE Software, 9, 53–59.

[9] Quadri S. M., Ahmad N. ,and Farooq S. U. " SoftwareReliability Growth Modeling With GeneralizedExponential Testing –Effort And Optimal Software ReleasePolicy" , Global Journal of Computer Science andTechnology , 2011, pp.27 – 42.

[10] Lyu, M. R.,Handbook of Software Reliability Engineering.

[11] Karunanithi, N., Whitley, D., & Malaiya, Y. K. (1992a), Using neural networks in reliability prediction. IEEE Software, 9, 53–59.

[12] Karunanithi, N, et al., "Prediction of software reliability using neural networks," Proceedings IEEE International Sym. Software Reliability Engineering, pp. 124-130, 1991.

[13] Karunanithi, N., Whitley, D., & Malaiya, Y. K. (1992b). Prediction of software reliability using connectionist models, IEEE Transactions on Software Engineering, 18, 563–574.

[14] Khalaf Khatatneh, Thaer Mustafa, Software Reliability Modeling using Soft Computing Technique, European Journal of Scientific Research, Vol 26 No 1(2009), pp 154-160.

[15] Karunanithi, N, et al., "Prediction of software reliability using neural networks," Proceedings IEEE International Sym. Software Reliability Engineering, pp. 124-130, 1991