International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org
Volume 3, Issue 11, November 2014 **ISSN 2319 - 4847**

Neural Network analysis of Software Reliability Growth Models

Sugam Srivastava[1] and Monika Sharma[2]

[1]University Institute of Engineering and Technology
Panjab University Chandigarh, India

[2]University Institute of Engineering and Technology
Panjab University Chandigarh, India

ABSTRACT

There is a major concern which has come across all the researchers in the field of software-reliability that is how to develop a universal model which can predict the reliability under all circumstances, since reliability-growth models predictions varies across different testing phases. The best way is to develop a model which makes no assumptions about the behavior of software failure. This is where Neural Network model comes into play. Neural Network models have significant advantage over simple parametric reliability models because they only require failure history as input and make no assumptions. Based on those failure inputs neural network model predicts the future failure. This paper shows the reliability prediction through neural network model as well as through reliability growth models.

Keywords:- Neural network model, parametric reliability models, non-parametric reliability models, Hann filter.

1. NTRODUCTION

In today's world, computers are used in almost every application, from banks and hospitals to nuclear reactors and defense. As we know all these areas demand high quality of accuracy and precision and that is the reason why research on software reliability has become all so important. The main area of concern in software reliability has always been how to develop reliable software at a low cost. As the reliability of software increases the probability of future failures decreases. This is where reliability prediction models play an important role. In recent years, researchers have developed many such software reliability models which predict the failure count for any software during testing phase. A reliability model works on the failure data, it takes failure data as an input performs certain computation and predicts future failure occurrences. Software reliability models have always been an area of issue for the researchers because a method to construct one single model which can work for all possible software projects is yet to be defined. The conditions and parameters defined for certain type of models may differ considerably for other class of models due to which a single reliability model is unable to predict future failure for all the software projects. Software reliability models have been broadly classified under two major classes:-Parametric Software Reliability Models and Non-Parametric Software Reliability Models. The main difference between parametric and non-parametric reliability models are that the parametric models are the simple orthodox and traditional models which rely heavily on the assumptions and estimations. The parametric models have a set of unknown parameters which are calculated using the failure data and then these parameters are made to adjust to the failure data so that the model can predict the reliability. However one of the major issues related to parametric models is model inconsistency problem. Since different parametric models have different set of assumptions on which they are based therefore there results vary considerably hence the prediction results of two different models for the same data set will show discrepancies. Therefore the researchers have now focused their area of interest to Non-parametric reliability models. The most important factor which works for non-parametric reliability models is that these models are based on machine learning techniques i.e. these models are able to evolve themselves based on the failure data collected during the initial steps of the testing process. Since these models are more intelligent and advanced and are able to evolve themselves so there reliability over the assumptions and other external parameters are completely eliminated which helps in removing model inconsistency problem up to a large extent. There are number of machine learning techniques used in software reliability models such as Support Vector Machine (SVM), Neural Networks (N.N), genetic algorithm (GA) and boosting techniques (B.T). This paper is organized as follows. In Section II, the brief introduction of the Neural Network is given. Some parametric models have been discussed in section III, IV and V. Results are discussed in the section VI.

2. NEURAL NETWORK MODEL

Neural Network falls under the category of non-parametric reliability models, which means they are able to evolve themselves gradually with the failure input data. A neural network is a simulation of human brain and it works

the same way as our neural system. The basic characteristics of neural networks are [1]:-

1) A neural network consists of many neurons which performs certain computation when input is provided to them to produce certain output.

2) It consists of three layers namely input layer, output layer and a hidden layer. Hidden layer is optional depending on the problem being solved.

3) Each layer has neurons which are connected to each other and each connection has certain weights and bias associated with them.

There are many kind of neural network models and learning techniques such as radial basis function model, learning vector quantization, kohonen self organizing network, recurrent networks but we are more concerned with another important kind of neural network known as feed-forward neural networks[2]. A feed forward neural network starts with an input layer and ends with an output layer as shown in Figure1. It consist of a layer of neurons to which we provide inputs from the outside world, there is an output layer that provides results to the outside world and there is a hidden layer which has no direct contact with the external environment.

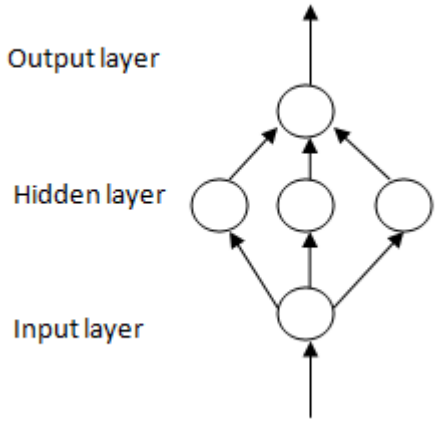

Figure1. A standard feed-forward network

The hidden layer acts as a bridge between the input layer and the output layer. Each neuron in the hidden layer takes input from the input layer, performs certain computation and generates result in the form of activation function. This activation function then goes as an input to the neurons of the output layer. No computation is done by the neurons in the input layer. The major function of input layer is to simply take values from the outside world associate them with weights and pass them to the neurons in the hidden layer. The reason why they are called feed forward network is because they can only propagate the computation in the forward direction unlike the other networks such as Jordan network which can perform both forward and backward computation. Since Neural Networks are machine based learning models, they are trained through some supervised learning algorithms [3]. There are several supervised learning algorithms but the most widely used is known as back propagation algorithm which adjusts the network weight by propagating error back into the network. Training a network involves several steps also called as epochs. During the initial epoch weights are given a set of random values ranging from +1.0 to -1.0 which are added to the output generated from the neurons of the input layer [4].

3.PARAMETRIC MODELS

As discussed earlier in the paper, the parametric models mainly depend upon the assumptions and the statistical theory. These models are based on mathematical theory of statistics and on assumptions about the already defined behavior of software failure [5]. There are many number of parametric reliability models which have been develop through these years such as goel okumoto, yamada s-shaped model, littlewood-verral model, non homogenous Poisson process(NHPP) model etc. Our area of focus in this paper will be on two such models i.e. Jelinski-Moranda model and Musa-Okumoto model. Both these model are the class of parametric models and are some very early developed models. Every model must have certain important characteristics [6]. It must have good future prediction of failure behavior. It must be useful to the organization using the model. It must be simple, not confusing. Both these models possess all these attributes. These models are also based on certain common assumptions such as:-

1) Every failure, fault detected during the process are independent and are not dependent on any factor whatsoever.
2) The probability of occurrence any fault under any class is equally distributed.
3) The data being used is operated in such a manner that it can be fit to reliability prediction.

Now let us discuss these models in some more details:-

4.JELINSKI-MORANDA MODEL

Jelinski-Moranda (J-M) model is considered as the first reliability model to be proposed by the researchers Jelinski and Moranda while working on some navy projects. It is more popular when compared to Musa-Okumoto and its metrics are easily available. Jelinski-Moranda model assumes that finite number of failures are present in the system therefore it is categorized by time between failures [7]. It is a statistical model which generates output in the form of exponential order. It is based on a basic assumption that the process of correction of faults (sometimes detection) only starts when the data has N number of faults and all the faults have the same ratio □ [8].

In addition to the standard assumptions discussed above Jelinski-Moranda has following assumptions too:-

1) Whenever a fault is corrected, no new fault is introduced into the system.
2) Failure rate is proportional to the number of remaining defects.
3) Failure rate does not change over the time between different fault occurrences.
4) Every error occurs unintentionally and are accidently occurred.

Jelinski-Moranda model is sometimes also known as de-eutrophication model [9]. The rate of fault detection is directly proportional to the current fault content [3]. The hazard rate can be seen in the equation (1):-

$$\lambda_i = (N - i) \qquad (1)$$

N - represents total number of faults
is called the proportionality constant
is the failure rate

The equation clearly shows that the failure rate decreases proportionally with the parameter . Therefore, the Jelinski-Moranda model can be thought of in terms of actual failure times instead of interfailure times.

3.MUSA-OKUMOTO MODEL

Musa-Okumoto (M-O) model are those kinds of parametric models which are widely accepted in the telecommunication industry. Musa-Okumoto model allows significant changes to the software while testing and reliability evaluation takes place. Musa-Okumoto model comes under the category of geometric family and are basically Poisson in nature[10]. The Musa-Okumoto model assumes that an infinite number of failures permanently exist in the system. The failure intensity is described best in terms of the expected number of failures experienced i.e. failures per time period. In addition to the standard assumptions, the Musa-Okumoto model also has some additional assumptions as well:-

1) The cumulative number of failures follows a Poisson process.
2) Failure intensity decreases exponentially with the expected number of failures experienced. Unlike Jelinski-Moranda model, the Musa-Okumoto model demonstrates a hazard rate that is exponential in nature, and is a continuous function as shown in equation 2:-

$$\lambda_i = \lambda_o e^{-R} \qquad (2)$$

In this the hazard rate is not constant, but decreases exponentially as failures occur. The Musa-Okumoto model is also called the logarithmic model since the expected number of failures is a logarithmic function [11]. One of the major differences between J-M Model and M-O model is that in J-M model whenever a fault is fixed it is reflected in the overall reliability of the system, whereas in M-O model, they do not have any direct effect on the reliability of the system.

4.SIMULATION AND RESULT

The performance of the models is simulated on Matlab R2009b as well as in CASRE tool. For Neural Network model we will be using Matlab for simulation and for parametric models we will be using CASRE tool.

Table1: Data set used for both neural networks and the parametric models.

Test Interval	Number of failures	Length of test interval(seconds)	Severity
1	1.4000000E+02	5.6000000e+001	1
2	1.9000000E+02	5.6000000e+001	1
3	2.3000000E+02	5.6000000e+001	1
4	1.2000000E+02	5.6000000e+001	1

5	2.2000000E+02	5.6000000e+001	1
6	1.2000000E+02	5.6000000e+001	1
7	1.3000000E+02	5.6000000e+001	1
8	1.9000000E+02	5.6000000e+001	1
9	1.0000000E+02	5.6000000e+001	1
10	5.0000000E+01	5.6000000e+001	1
11	5.0000000E+01	5.6000000e+001	1
12	5.0000000E+01	5.6000000e+001	1
13	7.0000000E+01	5.6000000e+001	1
14	7.0000000E+01	5.6000000e+001	1
15	1.0000000E+01	5.6000000e+001	1
16	3.0000000E+01	5.6000000e+001	1
17	1.0000000E+01	5.6000000e+001	1
18	2.0000000E+01	5.6000000e+001	1

We have used the same data set as given in TABLE I for both neural network as well as parametric models so that a proper evaluation can be done between all the models.

1) The first column gives a sequential test interval number.
2) The second column specifies the number of failures that were observed during a given test interval.
3) The third column gives the length of the test interval
4) The fourth column specifies the severity of the failure, on a scale of 1 to 9.

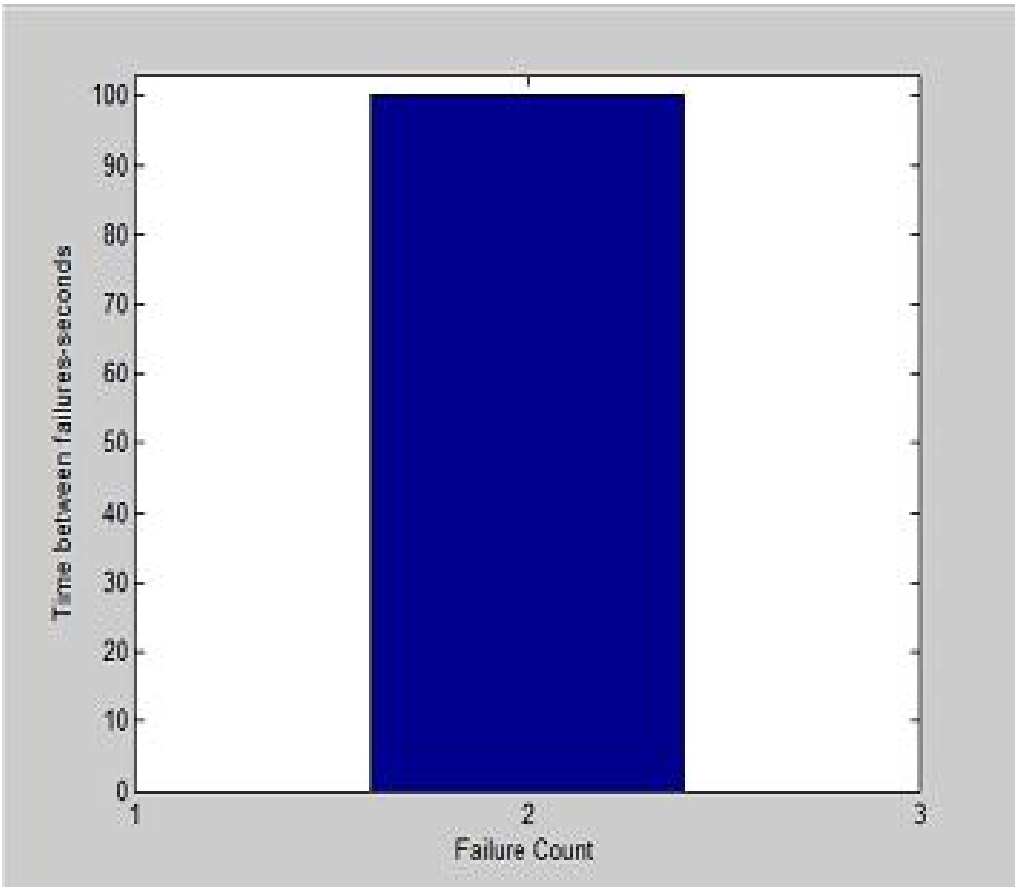

Figure2 Reliability shown after estimation by neural network

We have a constructed a feed forward neural network with the help of Matlab. 70% of the data set shown above has been used for the training purpose and the rest 30% has been used for testing. From Figure 2 we can see that the ability of a neural network to predict the next failure count is 98% which means the probability of the software without failure is 0.98. The X-label in the above bar graph shows the failure count and the Y-label shows the Time between failures (seconds). Figure 3 shows the reliability of the two parametric models as a function of time. X-label indicates cumulative time between failures (seconds) and y-label indicates reliability for next 20,000 seconds. We have specified the length of the time interval for which the reliability will be calculated as 20,000 seconds, indicating that we want to know the probability of software running for 20,000 seconds without failure.

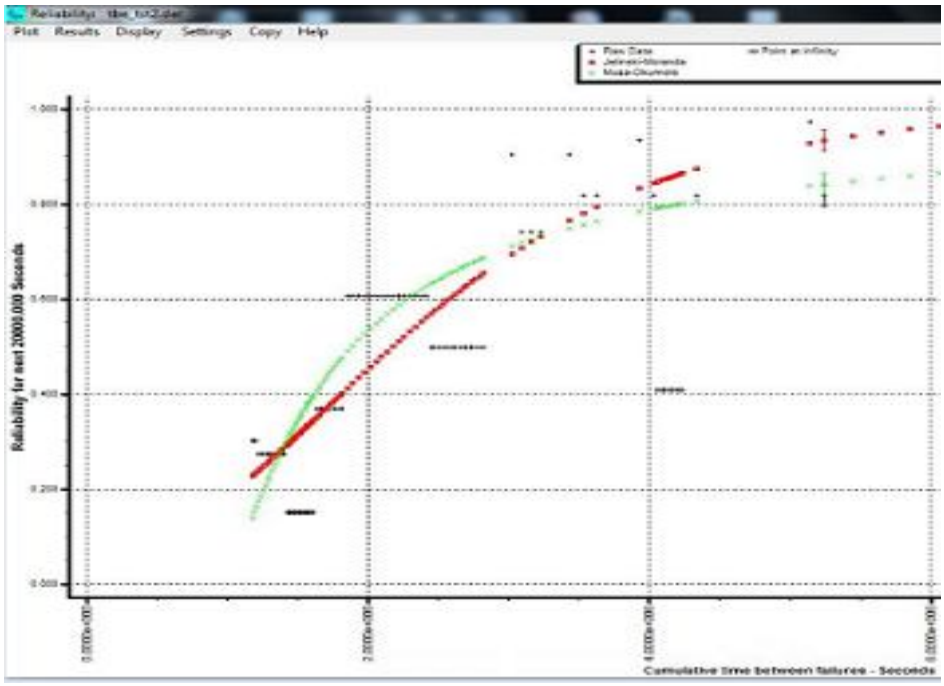

Figure3 Reliability shown by the parametric models

The graph shows that after 2,000,000 seconds, the probability of the software running for 20,000 seconds without failure is 0.45 for Jelinski-Moranda model and 0.5 for Musa-Okumoto model. But if we continue to test further then we can see that after 4,000,000 seconds, the probability for Jelinski-Moranda Model becomes 0.85 and for Musa-Okumoto model it is 0.8.

5. CONCLUSION

Developing reliable software requires the best knowledge of software reliability techniques. Since till now there is no single model which can predict accurate reliability of any software based on different data sets whether open or closed , therefore comparison between different parametric and non-parametric models have been made. From the graphs shown above in the results section we can see that that the neural network model has the highest probability for predicting the reliability of software. The probability for neural network is 0.98 whereas for Jelinski-Moranda it is 0.85 and for Musa-Okumoto it is 0.8.Although Jelinski-Moranda model predicts better reliability when compared to Musa-Okumoto model it is unable to predict better reliability than Neural-Network model. Therefore we can conclude that neural network model is a better reliability model when compared to parametric models.

Refrences

[1] Yu-Shen, Chin-Yu Huang, Yi-Shin Chen “An Artificial Neural-Network-Based Approach to Software Reliability Assessment”IEEE trans Software Eng, Vol 29,No.3,2003, pp 261-269
[2] Nachimuthu Karunanithi, Darrell Whitely, Yashwant K. Malaiya “Using Neural Networks in Reliability Prediction”, Software, IEEE vol.9, Issue:4, 1992, pp 53-59.
[3] Ang Li, Qing Gu, Guang-Cheng Feng “SNN-A Neural Network based combination of Software Reliability Growth Models” TENCON 2005.
[4] N.Karunanithi, D.Whitely, and Y.K.Malaiya, “Prediction of software reliability Using Neural Networks”,IEEE software Int Symp on Soft Eng, May 1991, pp, 124-130
[5] Yumei Wu, Risheng Yang, “Software Reliability Modelling Based on SVM and Virtual Sample”, Reliability and Maintainability Symposium, 2013, pp 1-6.
[6] Sultan Aljahldi, Alaa Sheta Muhammad Habib “Software Reliability Analysis using Parametric and Non-Parametric Methods” 18th International Conference Computers and Their Applications, Honolulu, Hawaii, USA, March 26-28, 2003
[7] Franciso J.Samaniego, Simon P.Wilson “Estimation problems with the Jelinski-Moranda software reliability model ”, IEEE transc. On Reliability, vol.R-34, NO.3,1985
[8] Sona Ahuja, Guru Saran Mishra, Agam Prasad Tyagi, “Jelinski-Moranda Model for Software reliability Prediction and its G.A. based Optimised Simulation Trajectory”,

[9] H.Joe N.Reid "On the Software Reliability Models of Jelinski-Moranda and Littlewood" IEEE trans.on reliability, Vol ,R-34,No.3,1985
[10] P.A Keiller, T.A Mazzuchi "Enhancing the predictive performance of the Musa-Okumoto software reliability growth model", Reliability and Maintainability Symposium, 2000, pp 106-112.
[11] Shigeru Yamada, Mitsuru Ohba, Shunji Osaki, "Musa-Okumoto Software Reliability Growth Models and Their Applications", Reliability, IEEE transc on (Volume:R-33, Issue:4), 1984, pp 289-292
[12] Musa-Okumoto model homepage-http:// home.comcast.net