Evolutionary Neural Network Prediction for Software Reliability Modeling

Sultan Aljahdali
College of Computer Sciences
Taif University, Saudi Arabia
aljahdali@pca.gov.sa

Khalid A. Buragga
College of Computer Sciences
King Faisal University, Saudi Arabia
kburagga@kfu.edu.sa

Abstract: *Software Reliability is a key concern of many users and developers of softwares. Demand for high software reliability requires robust modeling techniques for software quality prediction. This paper presents a new approach to software reliability assessment by using neural network.*
The neural network model has been applied to three different applications and normalized root mean of the square of error as an evaluation criterion. Results show that the neural network model adopted has good predictive capability.

Keywords: Neural Network, Software Quality, Software Reliability, and Auto Regression

1. Introduction

Software reliability is becoming more and more important in software industry various techniques are required to discover faults in the development of software. Software reliability can defined as the probability of a software system to perform its specified functions correctly over a long period of time or for different input set under the usage environments similar to that of its target customer [15]. However; as reliability of software is measured in terms of failure it is impossible to measure reliability before the software is developed completely, software reliability is the most extensively studied quality among all the quality attributes [10].

In the past few years a number of software reliability assessment models have been developed to solve software reliability models. These software models have been developed in response to the urgent need for software engineers, system engineers and managers to quantify the concept of software quality prediction. Software reliability models were useful in cases like managing reliability, managing project changes and monitoring test programs. Some of the models that have been developed for software quality prediction are: exponential order statistical model, logic regression Case based reasoning, Artificial Neural Networks, and Optimal Set reduction. The main objective of these models are to help predict which modules are error prone which in turn can help developer to focus on many aspects of maintenance cycle [2].

In this paper we propose a new approach towards software reliability assessment using neural networks and normalized root of mean of the square of error *(NRMSE)* criterion as an evaluation criterion. The rest of the paper is organized as follows: After a brief examination of the existing techniques for software reliability using prediction using auto regression model, in section 2.1, a new approach using neural network for software reliability section 2.2, then we will discuss about data set in section 3, in section 4 we cover the architecture of the neural networks used for modeling software reliability. Section 5 we cover the experiment setup by observation of data for test/debug for one of the program for real-time control, in addition to evaluation criterion for each developed model to measure its performance.
Section 6 presents the results of the prediction of software reliability model using auto regression and the prediction of the software reliability model, using neural networks in the training and in the testing cases. Finally a summary of the work done and future research directions for the proposed strategy are discussed in section 7.

2.1 Prediction Using Auto Regression Model

Auto Regression models are the most popular method for building models and are used to calibrate almost all of the models. Linear least square regression analysis is still the most common technique used, as observed in the literature [1]. Much of the appeal of this technique lies with its simplicity and also its easy accessibility from many of the popular statistical packages.

One of the most famous regression models is the Auto-Regressive model. This model has been used in many applications. The auto-regressive model can be described in the following form:

$$y(k) = \theta_0 + \sum_{i=1}^{n} \theta_i y(k - \tau_i)$$

$y(k-\tau_i)$ is the past system output and $(k = 1,2,..n)$. θ_i is the tuning parameter for the auto-regressive model; n is referred to as the "order" of the model. This model can work as a reliability growth model if the model variables are redefined as follows:

$$C(k) = \theta_0 + \sum_{i=1}^{n} \theta_i C(k - \tau_i)$$

C(k-τ_i) is defined as the previous observed number of faults (k=*1,2,...,n*). This, way the auto-regressive regression model can be used in this research.

2.2 Neural Network for Software Reliability

The most popular training algorithm for feed-forward neural networks is the back-propagation algorithm; the back-propagation learning algorithm provides a way to train multi-layered feed-forward neural networks. In this research, we use the back-propagation learning algorithm to explore the development of a suitable model for software reliability prediction problem.
Neural Networks, consists of a number of elements called neurons. These neurons are grouped together to form a layer. Each neuron has a number of inputs and a single output. Each input has an assigned factor or parameter called the weight. A neuron works in the following way: the input signal to each neuron is first multiplied by the corresponding weight; then the result from the multiplication is summed and passes through a transfer function. The neuron output will not be activated unless the summation exceeds certain threshold. This operation can be represented by the Following:

$$o_j = f(|\sum_{i=1}^{m} w_{ij} o_i - \theta_j|)$$

Where O_i is the output of the unit *i*, O_j is the output of the unit *j*, *f(..)* is a transfer function, w_{ij} is the weight of the link between *i* and *j*, and θ_j is the threshold of the neuron *j*. *m* is the number of neurons in the input layer, and neurons in a neural network are arranged into layers. The structure of the feed-forward neural network in presented in the following figure: 1

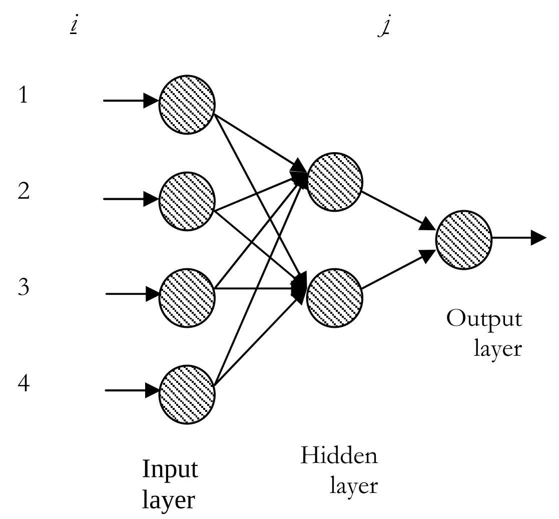

Figure 1: Feed-forward neural networks structure

3. Data set

The DACS Services at the Department of Defense (D.O.D.) Software Information Clearinghouse provides an authoritative source for the state of the art software information, supplying technical support for the software community. John Musa of Bell Telephone Laboratories compiled a software reliability database. His objective was to collect failure interval data to assist software managers in monitoring test status, predicting schedules and to assist software researchers in validating software reliability models. These models are applied in the discipline of Software Reliability Engineering. The dataset consists of software failure data on 16 projects. Careful controls were employed during data collection to ensure that the data would be of high quality. The data was collected throughout the mid 1970s. It represents projects from a variety of applications including real time command and control, word processing, commercial, and military applications.

4. Neural Networks Structure

The architecture of the neural networks used for modeling software reliability problem in this research is a three-layer feed-forward neural network. It consists of an input layer, one hidden layer, and an output layer. The input layer contains a number of neurons equal to the number of delayed measurements allowed to build neural networks model

In our case, there are four inputs to the network, They *are C(k-1), C(k-2), C(k-3), C(k-4).*
C (k-1) is the observed faults one-day before the current day. The hidden layer consists of linear hidden units. The output layer consists of one output neuron producing the estimated value of the fault. There is no direct connection between the network input and output. Connections occur only through the hidden layer. The hidden units are fully connected to both the input and output. The hidden and output layer nodes have linear activation functions.

5. Experiment Setup

5.1. Test/Debug data for Real-Time Control

Observation of data for test/debug of a program for real-time control was used. The size of the program is 870 kilo-steps of FORTRAN and a middle level language. Since the test data is recorded day by day, the test operations performed in a day are regarded to be a test instance.

5.2. Evaluation Criteria

We used an evaluation criterion for each developed model to measure its performance. The criterion of evaluation (i.e. performance) was defined as the Normalized Root Mean of the Square Error (*NRMSE*). The equation, which governs the *NRMSE*, is as follows:

$$\frac{1}{n-1}\sqrt{\frac{\sum_{k=1}^{n}(y(k)-\hat{y}(k))^2}{\sum_{k=1}^{n}(y(k))^2}}$$

Where $C(k)$ is the observed faults and $\hat{C}(k)$ is the predicted faults for the given model structure. To explore the possibility of using neural networks for software reliability prediction, we developed some experiments to show the advantages of neural networks.

6. Experimental Results

6.1. Prediction Using an Auto Regression

We developed an Auto Regression model of order four to predict the software reliability for test/debug data of a program for real-time control. The model structure is given by the following equation:

$$\hat{C}(k) = a_0 + a_1C(k-1) + a_2C(k-2) + a_3C(k-3) + a_4C(k-4)$$

Where $C(k)$ is the observed fault and $\hat{C}(k)$ is the predicted fault, an auto regression has been used to identify the values of the parameters a_i (k=1,2,3,4). A data set represents 70% of the collected data that was used in the training phase. To verify the results of the parameter estimation process, the model has been tested with whole data set that represents 100% of the collected data for various projects. The results of the parameter estimation procedure are given in Table 1. The *NRMSE* of the testing, in regression model case also, is given in Table 2.

6.2. Prediction Using Neural Networks

In the following section, we will show the prediction of the software reliability model, using neural networks in the training and in the testing cases.

6.2.1. Training and Testing

The neural networks were trained with a different set of initial weights until the best sets of weights were calculated and the *NRMSE* was reduced to a small value. We used the neural networks weights developed from the training case to test the neural networks performance. The neural networks model has been tested with the rest of the collected data, which represents 100% of the collected data set. The *NRMSE* of the testing in neural networks case is given in Table 2. In Figures 1 to 3 we are showing the training and testing results for various projects using regression model, also, in Figure 4 to 6 we are showing the training and testing result for the same projects using neural networks.

7. Conclusions and Future Work

We have shown that the neural networks can be used for building software reliability growth models. Neural networks were able to provide models with smaller *NRMSE* than the regression model in all considered cases. If a regression model with a higher order had been considered then probably a smaller *NRMSE* would have been obtained. However, the number of the regression model parameters will be increased. This will require more observations for providing a reliable estimate of the parameters. The entire system of software reliability research is considered useful for software development and testing industry. At the present we are investigating the use of genetic programming to solve the software reliability problem.

Project Name	Military	Real Time Control	Operating System
a_0	3.7427	2.3977	0.4034
a_1	1.0087	0.8898	1.0621
a_2	-0.0181	0.0730	-0.0841
a_3	-0.2301	-0.1549	0.2673
a_4	0.2249	0.1612	-0.2392

Table 1: Results for the Estimation of a's using Least Square Estimation.

Project Name	Military	Real Time Control	Operating System
Number of Faults	101	136	277
Training Data	71	96	194
Testing Data	101	136	277
Regression Model	3.1434	1.7086	1.0659
Neural Networks	1.0755	0.5644	0.7714

Table 2: A Comparison between Regression model order 4 and neural networks model in testing case (NRMSE).

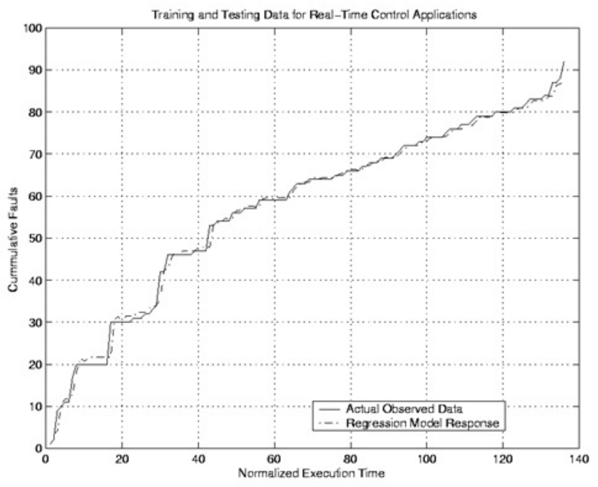

Figure 1: Actual and Estimated Faults Prediction error using LSE: Real-Time and Control Applications

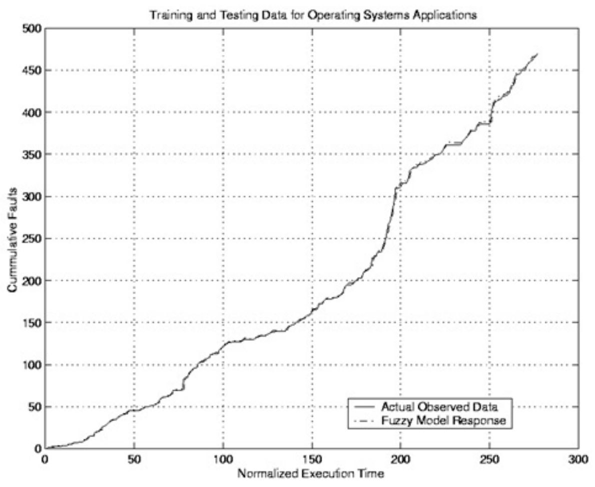

Figure 3: Actual and Estimated Faults prediction error using LSE: Operating Systems Applications.

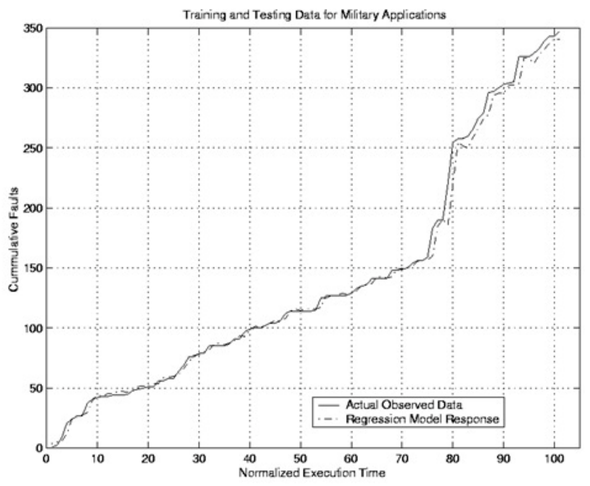

Figure 2: Actual and Estimated Faults prediction error using LSE: Military Applications.

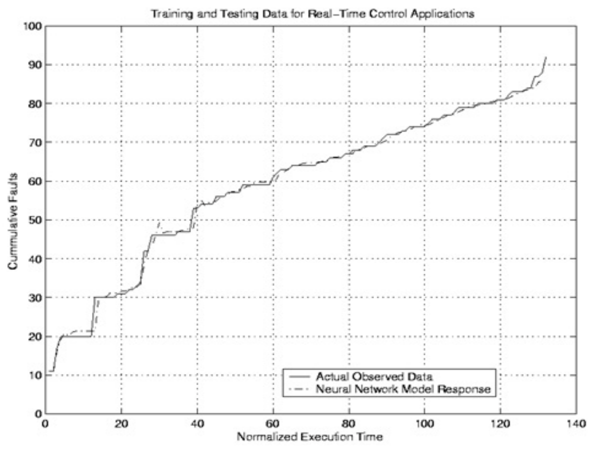

Figure 4: Actual and Estimated Faults prediction error using NNs: Real- Time and Control Applications

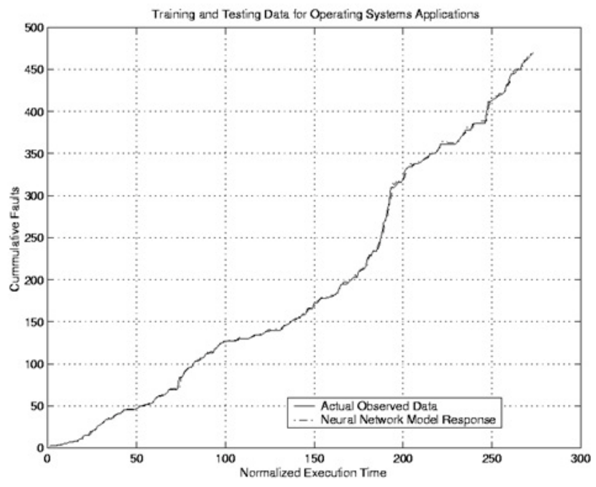

Figure 6: Actual and Estimated Faults prediction error using NNs: Operating Systems Applications.

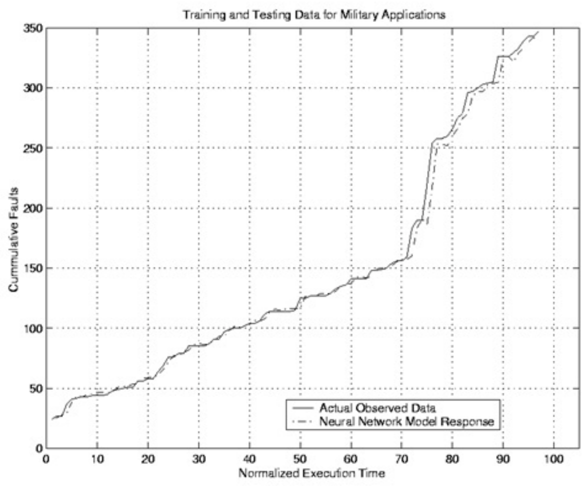

Figure 5: Actual and Estimated Faults prediction error using NNs: Military Applications.

Acknowledgment:

Dr. Aljahdali, and Dr. Buragga, would like to thank Taif University, and King Faisal University for their support during this research.

8. Bibliography

1. Aljahdali, S. "Prediction of Software Reliability Using Neural Network and Fuzzy logic", Ph.D. Dissertation presented to the faculty of College of Graduate Studies., Dept. of the Software Engineering and Info. System, George Mason University, Fairfax, Virginia, U.S.A, May 2003.

2. Aljahdali, S., Sheta, A., and Habib, M. "Software Reliability Analysis Using Parametric and Non-Parametric Methods", Proceedings of the ISCA 18th International Conference on Computers and their Application, March 26-28, 2003, pp. 63-66.

3. Aljahdali, S., Sheta, A., and Rine, D., "Predicting Accumulated Faults in Software Using Radial Basis Function Network", Proceedings of the ISCA 17th International Conference on Computers and their Application, 4-6, April 2002, pp. 26-29.

4. Aljahdali, S., Sheta, A., and Rine, D., "Prediction of Software Reliability: A Comparison between regression and neural network non-parametric Models", Proceeding of the IEEE/ACS Conference, 25-29, June 2001,pp.470-471.

5. Ganesan, K , Khoshgoftaar ,T.M., Allen, E.B. ," Case based Software quality prediction", International Journal of Software Engineeering and Knowledge Engineering, 1999, 9(6), Forthcoming.

6. Khoshgoftaar , T.M.,Allen, E.B, Jones, W.D., Hudepohl, J.P., " Which Software modules have faults that will be discovered by Customers?", Journal of Software Maintenance: Research and Practice, , Jan 1996,11(1):1-18.

7. Khoshgoftaar, T.M. , Pandya A.S., Lanning, D.L. ,"Application of Neural Networks for Predicting Faults", Annals of Software Engineering, 1995,1:141-154.

8. Khoshgoftaar, T.M., Allen, E.B., "Logistic Regression Modeling of Software Quality", International Journal of Reliability, Quality and Safety Engineering, Dec. 1999, 6(4), Forthcoming.

9. Liang T., Afzel, N. "Evolutionary neural network modeling for software cumulative failure time prediction ", Journal of Reliability Engineering and System Safety, 2005, Vol.87, pp. 45–51.

10. Liang T., Afzel, N. "On-line prediction of software reliability using an evolutionary" connectionist model", Journal of Systems and Software, 2005, Vol. 77, pp. 173–180.

11. Musa, D. J. "Software Reliability Engineering: More Reliable Software Faster and Cheaper" Author house, Indiana, 2004.

12. Srinivasan, K., and Fisher, D., "Machine learning approaches to estimating software development effort", IEEE Trans, Software Engineering, 21 (1995) pp.126-137.

13. Stewart, W., "Collinearity and least squares regression", Statistical Science, 2(1987) pp. 68-100.

14. Tian, J, " Better Reliability Assessment and Prediction through Data Clustering", IEEE Transactions on Software Engineering, October 2002, Vol. 28, No. 10.

15. Tian, J, " Software Quality Engineering", John Wiley and Sons Inc. 2005.