Prediction of Software Reliability: A Comparison between Regression and Neural Network Non-Parametric Models

Sultan H. Aljahdali
School of Information Tech.
George Mason University
Fairfax, VA 22030, USA
saljahdali@gmu.edu

Alaa Sheta
Computers and Systems Dept.
Electronics Research Institute
Cairo, Egypt
asheta1@eri.sci.eg

David Rine
Computer Science Dept.
George Mason University
Fairfax, VA 22030, USA
Drine@cs.gmu.edu

Abstract- In this paper neural networks have been proposed as an alternative technique to build software reliability growth models. A feed-forward neural network was used to predict the number of faults initially resident in a program at the beginning of a test/debug process. To evaluate the predictive capability of the developed model data sets from various projects were used [1]. A comparison between regression parametric models and neural network models is provided.

1 Introduction

The problem of developing reliable software at a low cost remains as an open challenge. To develop a reliable software system, we must address several issues. These include specification of reliable software, reliable development methodologies, testing methods for reliability, reliability growth prediction modeling, and accurate estimation of reliability [2, 3]. The issue of finding a common model for all possible software projects is yet to solved. Selection of a particular model is very important in software reliability growth prediction because both the release date and the resource allocation decision can be affected by the accuracy of prediction. Existing analytic models describe the failure process as a function of execution time (or calendar time) and a set of unknown parameters [4, 5, 6].

Artificial Neural Networks (ANNs) have been used both to estimate parameters of a formal model and to learn to emulate the process model itself to predict future outcomes. Recently, the use of neural networks in software reliability prediction have been explored in many publications [7, 8]. In this paper we focus on the problem of developing non-parametric software reliability models using NNs. The main goal is to develop a linear neural network model structure that can provide an efficient prediction capability than traditional parametric models. Most of the published literature used the a single input and single output NNs to build growth models. In our case we are using multiple delayed input of the observed faults in a test/debug process as input to the NN. A feedforward neural network trained using backpropagation learning algorithm was used.

2 Software Reliability Data Set

John Musa of Bell Telephone Laboratories compiled a software reliability database [1]. 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. In our case, we used data from three different projects. They are Military, Real Time Control and Operating System.

3 Regression Models

Linear least squares regression analysis is still the most common technique used to estimate linear models as observed in the literature [9]. Much of the appeal of this technique lies with its simplicity and also its easy accessibility from many of the popular statistical packages. The AR model can be described by the the following equation:

$$\beta(k) \quad = \quad a_0 + \sum_{i=1}^{n} a_i \beta(k-i)$$

where $\beta(k-i)$ is the previous observed number of faults and $(i = 1, 2, .., n)$. The value of n is referred to as the "order" of the model. a_0 and $a_i, (i = 1, 2, .., n)$ are the model parameter.

4 Evaluation Criterion

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

$$SSE = \sum_{i}^{m} (\beta(i) - \hat{\beta}(i))^2$$

β_i is the observed faults and $\hat{\beta}$ is the predicted faults for the given model structure. m is the number of observations.

5 Prediction Using LSE

We developed an AR 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.

$$\begin{aligned}\hat{\beta}(k) &= a_0 + a_1\beta(k-1) + a_2\beta(k-2) \\ &+ a_3\beta(k-3) + a_4\beta(k-4)\end{aligned}$$

LSE has been used to identify the values of the parameters a. A data set represents 70% of the collected data were used in the training phase. To verify the results of the parameter estimation process, the model has been tested with another set that represents 30% of the collected data for various projects. The results of parameter estimation procedure are given in Table 1. The sum square error of the training and testing, in regression model case, is given in Table 2.

6 Prediction Using NNs

6.1 Network Structure

The architecture of the network used for modeling the real time control program is a multi-layer feedforward 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 the network model. In our case, there are four inputs to the network. They are $\beta(i-1)$, $\beta(i-2)$, $\beta(i-3)$ and $\beta(i-4)$. $\beta(i-1)$ is the observed faults per day before the current day. The hidden layer consists of two nonlinear neurons connected to $\beta(i-1)$ and $\beta(i-2)$ and two linear hidden neurons connected to $\beta(i-3)$ and $\beta(i-4)$. The output layer consists of one linear output neuron producing the estimated value of the fault. The hidden units are fully connected to both the input and output. The hidden and output layers nodes have linear activation functions.

6.2 Training and Testing

The neural network was trained with different sets of initial weights until the best set of weights were calculated. The SSE was minimized to a small value. We used the NNs weights developed from the training case to test the network performance. The NNs model has been tested with the rest of the collected data which represents 30% of the collected data set. The sum square error of the training and testing, in NN case, is given in Table 2. In Figures 1 to 6 we are showing the training and testing results for various projects using NNs. Also, the prediction error in each case is provided.

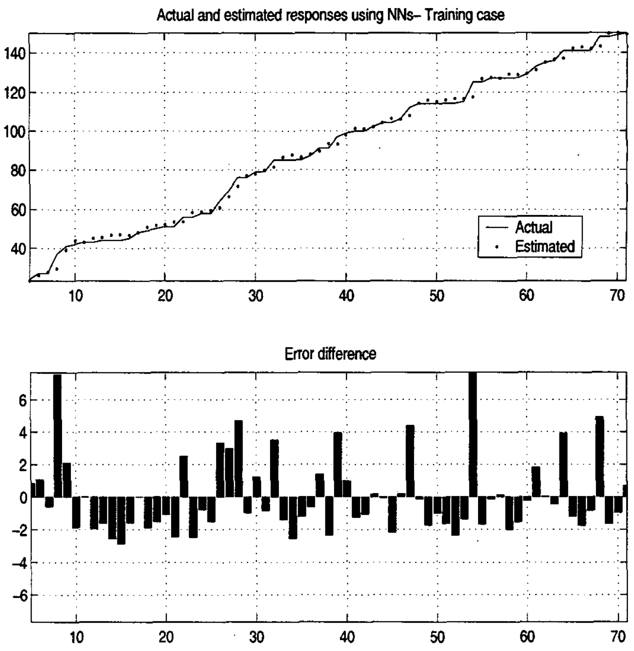

Figure 1: (a) Actual and Estimated Faults (b) Prediction error: Military Application

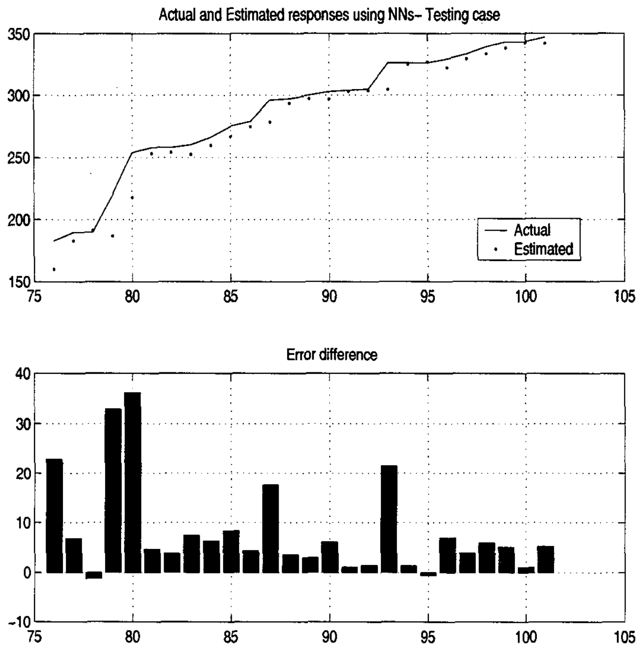

Figure 2: (a) Actual and Estimated Faults (b) Prediction error: Military Application

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

Table 1: Results for the estimation of a's using LSE

	Training		Testing	
Project Name	SSE-AR4	SSE-NNs	SSE-AR4	SSE-NNs
Military	5.7092	5.5409	168.541	160.3887
Real Time Control	2.4745	1.7988	1.3680	1.2542
Operating System	4.3482	4.2928	10.8850	9.9623

Table 2: A Comparison between AR4 and NN models in both training/testing cases

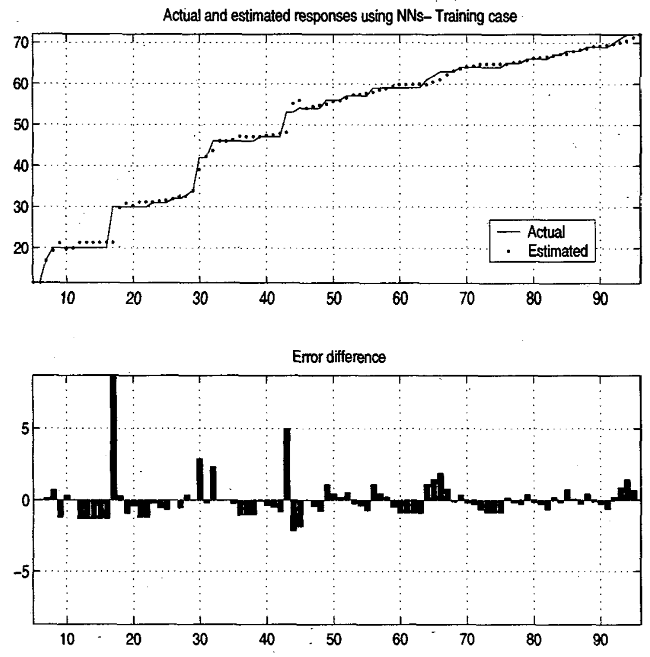

Figure 3: (a) Actual and Estimated Faults (b) Prediction error: Real Time Control Application

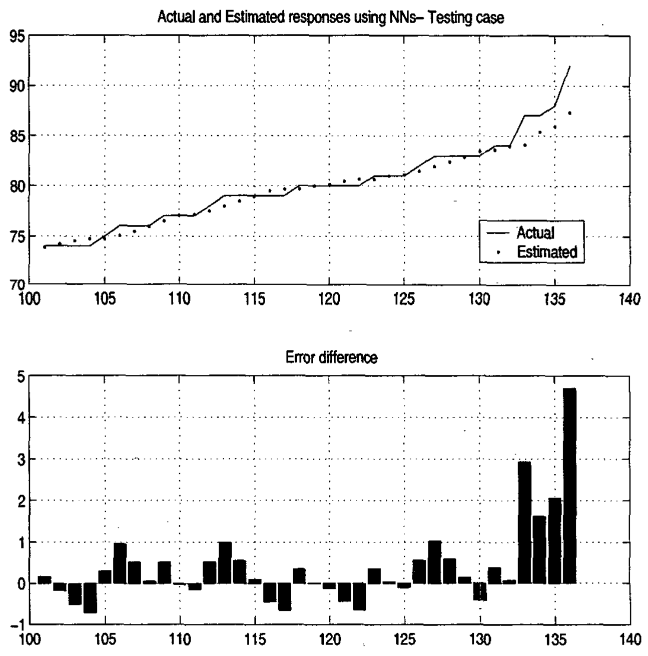

Figure 4: (a) Actual and Estimated Faults (b) Prediction error: Real Time Control Application

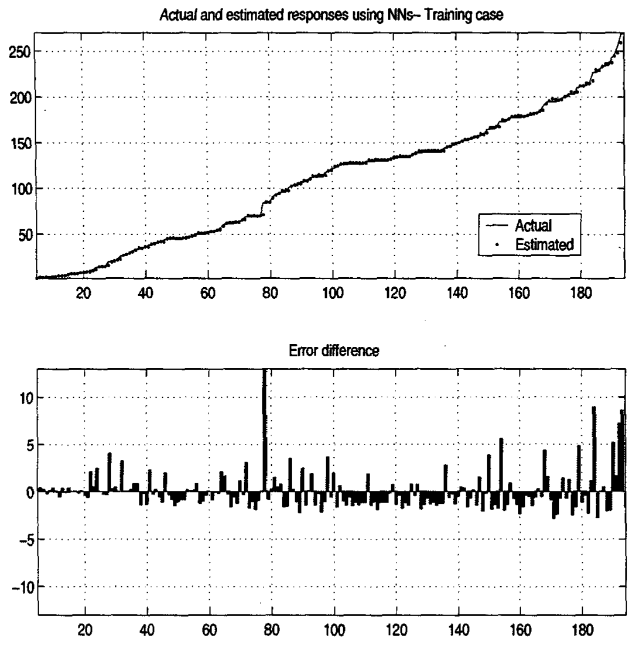

Figure 5: (a) Actual and Estimated Faults (b) Prediction error: Operating System Application

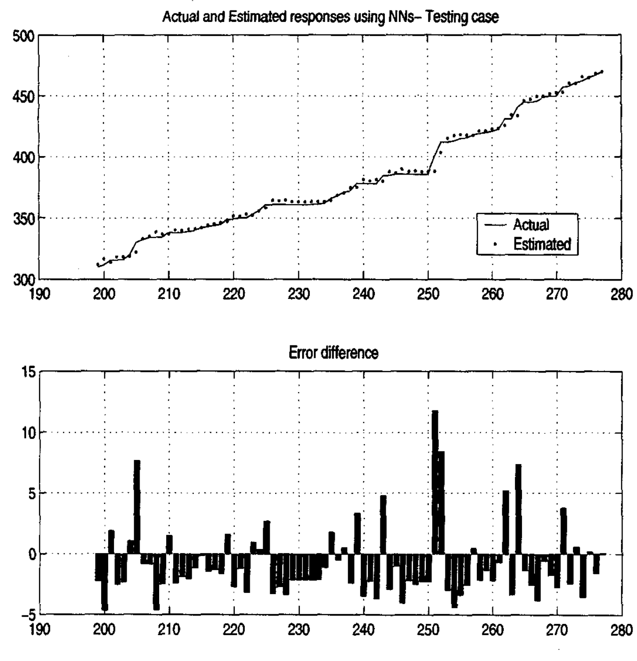

Figure 6: (a) Actual and Estimated Faults (b) Prediction error: Operating System Application

7 Conclusions and Future Work

We have shown that neural network can be used for building software reliability growth models. NNs were able to provide models with small SSE than the regression model in all considered cases. If a regression model with higher order have been considered probably less SSE is obtained. However, the number of the regression model parameters will be increased. This will require more observations for providing reliable estimate of the parameters. At present, we are investigating the use of evolutionary computations in to solve the software reliability growth modeling problem.

8 Acknowledgment

This research was developed while Dr. Sheta was a Visiting Scientist from the Electronics Research Institute (ERI), Cairo, Egypt to the ECE Dept., George Mason University, VA, USA under the NSF Grant No. INT-0000109.

Bibliography

[1] J. Musa, "Data analysis center for software: An information analysis center," *Western Michigan University Libraray, Kalamazoo, Michigan*, 1980.

[2] S. Brocklehurst, P. Y. Chan, B. Littlewood, and J. Snell, "Recalibrating software reliability models," *IEEE Trans. Software Engineering*, vol. 16, pp. 458–470, 1990.

[3] M. R. Lyu, *Handbook of Software Reliability Engineering*. IEEE Computer Society Press, McGraw Hill, 1996.

[4] B. Littlewood and J. L. Verall, "A bayesian reliability model with a stochastically monotone failure rate," *IEEE Trans. Relibaility*, vol. 23, pp. 108–114, 1974.

[5] J. D. Musa, "A theory of software reliability and its application," *IEEE Trans. Software Engineering*, vol. 1, pp. 312-327, 1975.

[6] J. D. Musa and K. Okumoto, "A logarithmic poisson execution time model for software reliability measurement," in *Proceedings of the 7 th Inter. Conf. Software Engineering*, pp. 475–478, 1984.

[7] N. Karunanithi, D. Whitely, and M. K., "Prediction of software reliability using connectionist models," *IEEE Transactions on Software Engineering*, vol. 18, no. 7, pp. 563–574, 1992.

[8] R. Sitte, "Comparison of software reliability growth predictions: Neural networks vs parametric recalibration," *IEEE Transactions on Reliability*, vol. 48, no. 3, pp. 285-291, 1999.

[9] M. Tummala, "Efficient iterative methods for FIR least squares identification," *IEEE Transaction Acoust., Speech, Signal Processing*, vol. 38, no. 5, pp. 887-890, 1990.