Predicting Accumulated Faults in Software Testing Process Using Radial Basis Function Network Models

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

Alaa Sheta
Computer 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 we propose the idea of building a new software reliability models using Radial Basis Function (RBF) network. The RBF network is easy to design and the network structure can be represented in a simple mathematical equation. Our goal is to build a generalized model that can be used for software predication [1]. The RBF network was trained with a set of data collected from the testing process of Military application projects. The RBF model was tested on other sets of projects. The results are promising.*

1. Introduction

Building a reliable model for predicting faults in software testing process is very important in software reliability growth prediction because both the release date and resource allocation decision can be affected by the accuracy of predication. Several solutions have been proposed to address these issues for model selection [2,3].

Recently, computer systems started to play important roles in our society. When failure happens to such systems major damages are expected. This is why computer systems must be very reliable. The current reduction of hardware cost made redundancy techniques feasible so that hardware faults are well tolerated by these redundancy techniques. Due to increasing advance and complexity of the developed software engineering systems most computer systems today is huge and complicated so that it is more likely to suffer from faults. Thus, the reliability of software has recently become one of the major issues in the realization of highly reliable computer systems. The problem of developing efficient and reliable software is still a challenge.

One of the main research direction in software reliability research is how to develop general prediction models [4]. Existing models usually count on *a priori* information and assumptions about the type of expected failures, the probability of individual failures and the probability of individual failures. Most of these models, named as parametric models. The model parameters are adjusted such that it can catch the behavior of the testing to failure response. Statistical techniques are likely to provide models that are linear in the parameters. This type of models can not catch the actual testing to failure characteristics by adjusting two or three parameters.

2. Neural Networks in the Prediction of Software Reliability

One of the most common model-building approach used in the literature as an alternative to least mean squares regression is the feed-forward neural network. It is often simply called back-propagation neural networks.

Although there is a large number of different neural network architectures and training algorithms exist, almost all published studies in software reliability concerned was limited to this type [5,6,7,8,9,10]. This can be seen as a reflection of the way of understanding the neural network techniques by many practitioner.

3 Structure of RBF Network

The problem of interpolation of real multivariable time series can be expressed as follows. Let us consider that a set of N data points input space R^d, together with their associated desired output values in R:

$$D = \{(x_i, y_i) \in R^d \times R, 1 \le i \le N \,|\, f(x_i) = y_i\} \qquad (1)$$

This data set can be used to characterize a function with one-dimensional output values multi-dimensional interpolation can be done by generalizing the following equations and algorithms, while; considering separately each component of the output vectors. We will consider only dimensional output in the following. The RBF approach to approximate function f use M functions ϕ_φ. ϕ_φ is, the radial basis function, described as follows:

$$\phi_j(u) = \phi_j(\| u-c_j \|) \qquad (2)$$

c_j are the locations of the centroids i.e., the centers of the radial basis functions, while |…| denotes as the Euclidean norm and $j= 1, 2 \ldots,N$. u is the network input vector. The approximation of the function f may be expressed as a linear combination of the radial basis functions:

$$\hat{f}(u) = \sum_{i=1}^{M} w_j \phi_j(\| u-c_j \|) \qquad (3)$$

The most common radial basis function, in practices, is a Gaussian kernel given by:

$$\phi_j(\| u-c_j \|) = exp(\| u-c_j \|/rj)^2 \qquad (4)$$

r_j is the width factor of the kernel j (j= 1,2,..,N). Once the general shape of the ϕ_j function is chosen, i.e. c_j and r_j, thus the purpose of a RBF algorithm is to find the weights w_j, to best fit the function f. fitting means that global mean square errors between the desired outputs y_i far all data input points x_i, $1 \leq i \leq N$ and the estimated outputs $\hat{y}(k)$ is minimized. This error is given by:

$$MSSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2 \qquad (5)$$

4. Measure of Performance

The evaluation criterion was defined as the Mean of the Sum Square of the Error (MSSE). The equation which, governs the MSSE is as follows:

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

N is the number of measurements used. y_i is the observed faults and $\hat{y}$ is the predicted faults for the given model structure.

5. Prediction Using RBF Network

5.1 Data Used

John Musa of Bell Telephone Laboratories compiled a software reliability database contains data for 16 projects from various applications. He collected failure interval data to assist software managers in monitoring test status, predicting schedules and to assist software researchers in validating software reliability models.

5.2 Network Structure

The architecture of the network used for modeling the real time control program is a multi-layer feed-forward network. It consists of 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 input signals to the RBF network. They are $y(k-1),y(k-2),y(k-3),$and $y(k-4)$. $y(k-1)$ is the observed faults per day before the current day. The hidden layer consists of four nonlinear neurons. The activation functions for these neurons are Gaussian function. The hidden units are fully connected to both the input and output. The hidden and output layers node has a linear activation function.

5.2 Training with single project and Testing with two projects

The neural network was trained with different set of initial weights until the best set of weights were calculated. The MSSE was minimized to small value. We used the NNs weights developed from the training case to test the network performance. The NNs model has been tested with rest of the collected data, which represents two other projects. The mean of the sum square error training and testing cases is described in Table 1. In Figures 1 to 3 we are showing the results for project 40 in the training case and projects 17 and 27 in the testing cases. The prediction error in each case is provided in the lower figures.

Project Number	MSSE
Training Project 40	17.9261
Testing Project 17	9.7997
Testing Project 27	15.8263

Table 1: Results for the RBF Network when training with single project and testing with two projects.

5.4 Training with two projects and Testing with one project

The RBF network was trained with a data collected from two projects, project 40 and project 17. The MSSE was computed. The used the NNs weights developed from training case to test the network performance. The NNs model has been tested with the test data collected from project 27.

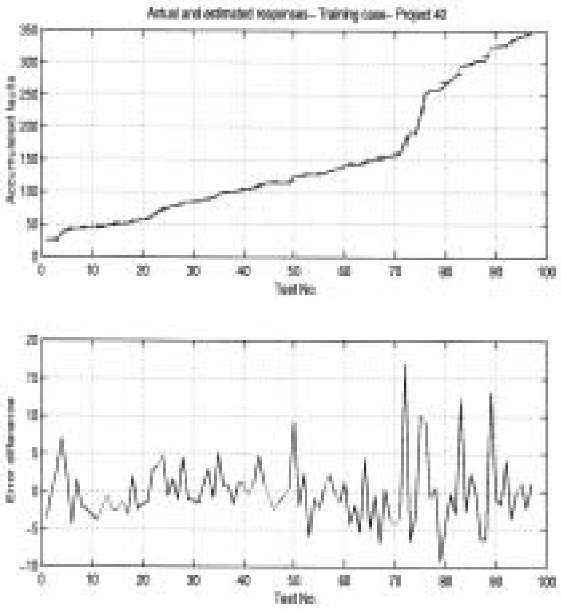

Figure 1: a) Actual and Estimated Faults
(b) Prediction error: Military Application

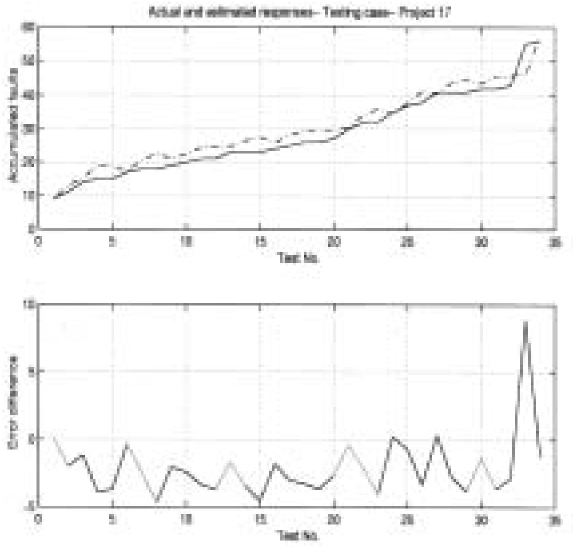

Figure 2: a) Actual and Estimated Faults
(b) Prediction error: Military Application

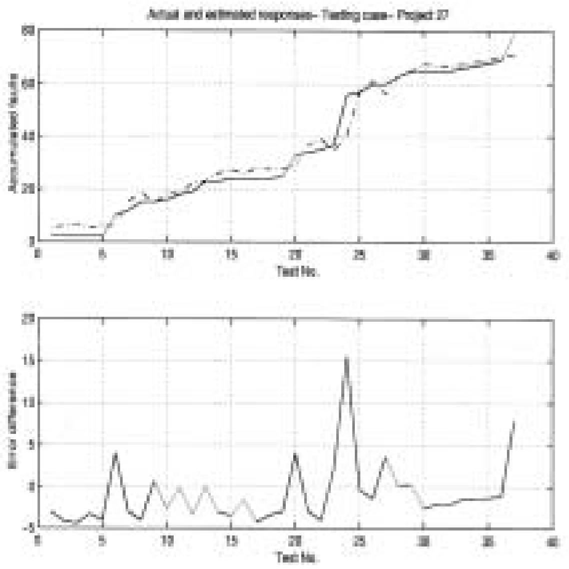

Figure 3: a) Actual and Estimated Faults
(b) Prediction error: Military Application

All projects have same nature since they were collected from Military application programs. The MSSE of the training and testing, in NN case is given in Table 2. In Figures 4 to 5, we are showing the training and testing results for various projects using NNs. Also, the prediction error in each case is provided.

From the above-described results, it can be seen that RBF network was able to generalize the results provided from training cases when test by other projects. All projects have the same nature. Our intention was to show it is possible to build a prediction model that can be used for predicting accumulated faults for other projects have data collected in the same environment. In our case it is the Military application.

Project Number	MSSE
Training Project 40 and 17	15.738
Testing Project 27	12.4759

Table 2: Results for the RBF Network when training with two projects and testing with one project.

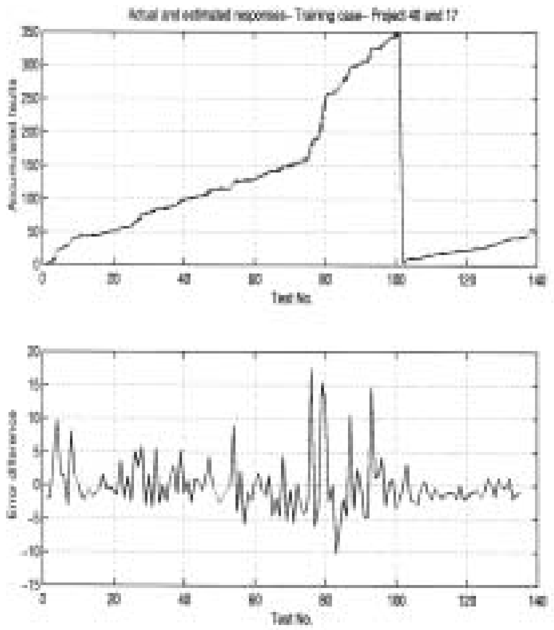

Figure 4: a) Actual and Estimated Faults
(b) Prediction error: Military Application

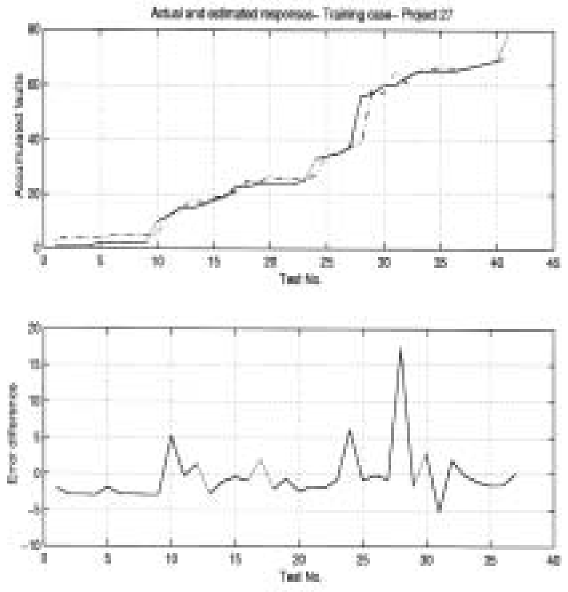

Figure 5: a) Actual and Estimated Faults
(b) Prediction error: Military Application

6. 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 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.

7. Acknowledgment

This research is supported by the US-Egypt Board under Grant No. INF4-001-019

Bibliography

[1] J. Musa, “Data analysis center for software: An information analysis center,” *Western Michigan University library, Kalamazoo, MI* 1980.

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

[3] K.M Atsumoto and K. Inoue, “Experimental evaluation of software reliability growth models,” *in Proceeding of the IEEE of FTCS18*, pp.148-153, 1988.

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

[5] Y. K. Malaya, N. Karunanithi, and P. Verman, “ predictability measures for software reliability models” *in proceeding of the 14 the IEEE inter. Conf. Computer Software Applications*, pp. 7-12, 1990.

[6] B. Littlewood and J.L. Verall, “A Bayesian reliability model with a stochastically monotone failure rate,” *IEEE Trans. Reliability*, vol.23, pp.108-114, 1974.

[7] N. Karunanithi, D. Whitely, and M.K., “ Prediction of software reliability using connectionist models,” *IEEE Trans. on software Engineering*, Vol. 18, no 7, pp. 563-574, 1992.

[8] R. Sitte, “Comparison of software reliability growth prediction: Neural Networks VS Parametric recalibration,” *IEEE Trans. on Reliability*, vol. 48, no.3, pp. 285-291, 1999.

[9] A. L. Goel, “ Software reliability models: Assumptions, Limitations, and applicability,” *IEEE transactions on software Engineering*, vol. 11, no. 12, pp. 1411-1434, 1985.

[10] P.B. Moranda, “predictions of software reliability during debugging,” *in proceeding of Annual reliability and Maintainability symposium*, pp. 327-332, 1975