Rolling Prediction Based Software Reliability Model Consideration with Learning Curve

Tian Jie[1, 2*], Wu Ji[1], Yang Haiyan[1], Liu Chao[1]

[1] School of Computer Science and Engineering, Bei hang University, Beijing, China.
[2] Beijing Command College of CPAPF, Beijing, China.

* Corresponding author. Tel.: +86-13811363910; email: happyjie717@126.com
Manuscript submitted July 10, 2016; accepted October 12, 2016.
doi: 10.17706/jsw.11.12.1182-1190

Abstract: Software reliability is an important factor for evaluating software quality in the domain of safety-critical software. The neural network prediction method has been widely used in reliability prediction area. However, Data noise and other issues make this approach easy to falling into local optimum, and reduce the accuracy of the prediction, it also affect the applicability of the model. In this paper, we consider the learning curve effect, and proposed a neural network based reliability prediction, utilize the rolling forecast method to elevate the accuracy and applicability of neural network. The method is validated through three groups of public data sets. And the results show a fairly accurate prediction capability.

Key words: Software reliability, neural network, learning-curve, rolling prediction.

1. Introduction

Software plays a very important role in our modern life and is used in many applications. It has become the core of many safe-critical systems: communications, monitoring systems, automobiles, airplanes, and so on. All of these applications demand high-quality software. As a critical factor for software quality, software reliability has become more and more important in computer system. Software reliability is defined in ANSI, which is the probability of failure-free software operation for a specified period of time in a specified environment [1], [2]. During the last three decades, there are many researches around this area, and many software reliability growth models (SRGMs) have been proposed for evaluating software reliability. SRGMs can be generally divided into two categories: one is the parametric models, such as nonhomogeneous Poisson process (NHPP) models, Schneidewind models and Musa's Basic execution time model; the other is non-parametric models, which is based on neural network, support vector machine (SVM) and so on.

The comparison between the models shows that the strong assumptions of parametric models make them can only be applied in some areas of the software system. It made the model difficult to use, or lead to the prediction result is not satisfactory. With a detailed study, the neural network prediction method has a better effect than the traditional parametric model prediction [3]. It has good adaptability and can improve the prediction accuracy effectively.

However, some problems in the neural network lead it is difficult to apply in the actual project. The problems reflected in the network structure is difficult to determine, the neural network model easy to fall into local optimum, the noise of training data affect the prediction results, and so on. In order to improve the applicability of the neural network prediction method, this paper proposes considering the learning

effect the reliability prediction methods based on rolling forecasts neural network.

In this paper, we considered the learning-curve and data noise. Through eliminating the learning effect of failure data and rolling prediction of the failure time, we can improve the model adaptability and the predictive accuracy effectively.

The rest of this paper is organized as follows: Section 2 introduce some related work about software reliability prediction. Section 3 describe the leaning effect of failure data and the rolling prediction of failure time. Section 4 shows the experimental results and finally we conclude this paper in Section 5.

2. Related Works

In this section, we briefly introduce the related works about software reliability and the reliability prediction based on neural network.

From the software reliability engineering perspective, the research of software reliability around the failure mechanism and reliability assessment. Since 1970, many software reliability growth models have been proposed, these models have an important theoretical foundation for the analysis and assessment of reliability, and to provide practical guidance for the project [4]-[6]. Musa and Okumoto proposed that the software reliability models can be divided into five categories of perspective [7]: (1) reliability analysis of the running time (including the CPU time, calendar time and testing time) point; (2) reliability analysis of the cumulative of the failure numbers; (3) reliability analysis of distribution patterns in the number of failures before time t; (4) reliability analysis of the failure intensity in a limited time (a function of time); (5) reliability analysis of the failure intensity in indefinite period of time under the failure intensity.

Classical software reliability models based on the failure time assumption, which assumes that the software failure time distribution has an exponential distribution pattern. Exponential distribution model has better nature of mathematics, including: the reliability model assumptions based on Markov features, the reliability model assumptions based on the non-homogeneous Poisson process (NHPP), the reliability model assumptions based the localized forecast(Schneidewind model).

Since the approaches mentioned above have a good effect on reliability prediction, but we found that many models depend on a priori assumptions and the high demands on data. In recent years, neural network approach has been used to predict software reliability. This method is more flexible and usable, so we introduce some work about neural network based software reliability.

Karunanithi [8], [9] first used some kind of neural networks to estimate the software reliability. They using the execution time as the neural networks' input and predict the cumulative number of the defect numbers. Their study first demonstrated the neural network can model the complexity datasets and improve the predictive quality. There are also some researches based on multiple-delayed-input single-output neural network predict the software reliability. Cai [10] proposed using the recent 50 inter-failure times as the multiple-delayed inputs to predict the next failure time. Tian [11] proposed a software reliability prediction method based on multiple-delayed-input and single-output. But this method requires the network architecture. Su [12] proposed an artificial neural-network-based approach for software reliability prediction. They build a dynamic weighted combinational model (DWCM) which can combine various existing model. Zheng [13] predicted software reliability with neural network ensembles. They found that the ensemble network has better prediction capability than the single network. The reliability model of three kinds of machine learning techniques is analyzed and compared by Pradeep Kumar[19] and Yogita Kansal [20] merges the software reliability model and the neural network model to improve the prediction accuracy of the number of failures.

From existing research, we found the neural network can provide a flexible application for predict software reliability. And the multiple-delayed-input and single-output method can help us considering

complex factor to prediction. In this paper, we proposed considering the learning effect of BP neural network based Software failure time prediction with the rolling prediction.

3. Rolling Forcast Based Failure Time Prediction

Safety-critical software requires a high reliability. However, in the actual project, we found that using the neural networks to do the predictions, the impact of data noise often lead to the prediction results fall into a local optimum, and cannot react the real status of the system status. So we research focus on how to improve the applicability of the neural network prediction method.

Based on the above understanding, we analyze the failure data. In the testing process, test objectives, test strategy and ability of the testing person determine the testing result, such as failure time and failure intensity. Because of different test items having different characteristics, it is difficult to capture this law in a uniform model. Neural network predictive modeling can effectively solve this problem, and will get a better prediction if the sample data selected properly.

In order to give an accurate prediction result, we first eliminate the learning effect, and then determine the network structure; last we use rolling forecast methods to predict the failure time. In this section, we will give the details.

3.1. Software Reliability Prediction Base on BP Neural Network

There are two main characteristics of the neural network make it become a useful tool to solve the prediction problem: Learning and generalization ability. The learning process depends on the training data, the learning algorithm is used to adjust the network weights, making the smallest difference between the results of the neural network output and expected results. When the training process is successful, the neural network will use the new data to compute the prediction results; this result is the generalization result.

These characteristics are very fit for our problems. First, we have a series of process data, and then we use these data to predict the failure numbers.

Neurons are the basic unit of the neural network. A real value vectors is the input of the Neurons, and then calculate the linear combination of these inputs, at last compute of the function which input is this linear combination. We can get the output.

Figure 1 gives the sketch map of the neurons. w is a real constant, which express the weight of the input. w determines the input x on the neuron of the contribution rate for the output.

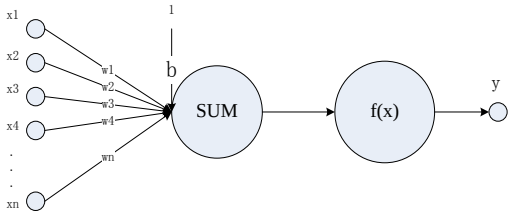


Fig. 1. Neurons.

$x_1 \sim x_n$: Individual component of the input vector

$w_1 \sim w_n$: weight of each input

b: bias

f: transfer function

y: neuron output

Equation 1 and Equation 2 give the mathematical expression of the neuron.

$$y = f(WA' + b) \quad (1)$$

$$y = \sum_{i=1}^{n} w_{ji} x_i + b \quad (2)$$

The neural network approach is often shows its superiority in solving the complex problems, which is difficult to describe by mathematical models, such as prediction problem, etc.

BP neural network is a multi-layer feed-forward neural network, the main features of the BP neural network is the error back to pass. The neural network can approximate any nonlinear continuous function to arbitrary precision.

Karunanithi, N. and Malaiya first use the neural network model for predicting the software reliability in 1991. Recently, more and more researchers began to concerning about this prediction method. Current research focuses on the following categories in Table 1.

Table 1. Neural Network Based Prediction Method Category

Single input	Single output	Multiple input	Single output
Cumulative execution time	Cumulative number of failures	Failure numbers detected in recent days	Remaining numbers of failures
Series of failure time	Cumulative failure time interval	Failure times	Next failure time

Neural network model has a versatile, easy to use, and predict the effect of characteristics, in recent years has been the focus of research in the field of reliability prediction. How to take full advantage of the neural network more accurate prediction is a very critical issue [14]. Multilayer feed-forward neural network has been proved to have multi-dimensional function mapping capability, but it also has shortcomings:

- Hidden layer neuron numbers is difficult to determine.
- Easy to fall into local optimum.

In order to minimize the impact of such issues on the prediction accuracy, our prediction method is considering the learning effect of the rolling prediction based on BP neural network.

3.2. Rolling Prediction

The basic idea of rolling forecast is through the “recent past” to predict the “future”. We think this method can eliminate the local optimum effectively.

Rolling prediction algorithm is derived from the rolling budget law. The rolling budget law, also known as continuous budget or sustainable budget. In accordance with the principle of nearly far fine and coarse, according to a budget, adjusting and preparation of the next budget, and the preparation of the budget period by period of continuous scroll onward, the budget is always to maintain a certain margin of time. Simply put, the completion is based on a budget targets, adjust and prepare the next budget and the budget period for rolling forward the passage of a budgeting method.

Based on the idea of rolling forecasts, when we design the BP neural network, we consider using the recent result to predict the next set of data. Though rolling prediction, we can effectively eliminate the problem of local optimal and improve the prediction accuracy.

3.3. Eliminate the Learning Curve Effect

The learning curve (Learning curve) reflects the relationship between the total number of the unit production time and the products. Learning curve, sometimes called exercise curve (practice curves), in order to know the details of the phenomenon in the process and pace of progress, as the future efforts of the pointer.

In the entire testing process, the cumulative number of failures shows the characteristics of S-shape, the first smooth region shows the learning-curve effect, this phenomenon derived from testers may be not familiar with the SUT, and it will leading to the failure number is small of long time intervals in the initial stage, and these data will impact the predict results, We think that the early data played a negative role in the failure time prediction. Based on neural networks, these data also will lead the prediction results into a local optimum.

Schneidewind [15] proposed the updated software reliability metrics, Its basic idea is the current software failure rate can do the better prediction than the earlier data. They analyze the bug or failure number and failure time to determine the initial interval. In this paper, we considered to use this method to determine the initial failures and eliminate the learning curve effect.

3.4. Determine the Network Structure and Predict the Cumulative Failure Time

We choose a most common type of neural network architecture which named feed-forward network. This network contains three layers: an input layer, a hidden layer and an output layer. Our input is the recent 10 failures cumulative failure time, and output is the cumulative failure time for the next failure. Fig. 2 depicted the construction of the network.

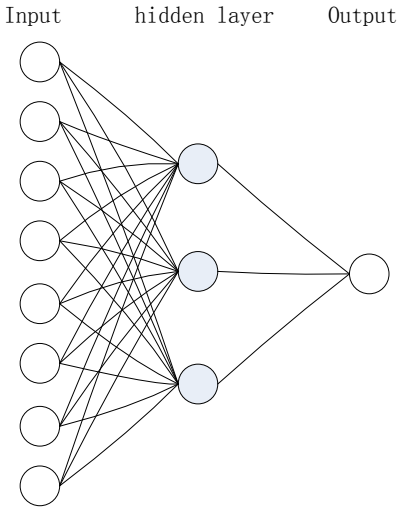


Fig. 2. Multi-perspective of testing process.

The difficulty is determining the number of hidden layer nodes. Through the experimental comparison, we calculate the network mean square error of the MSE and determine the number of hidden nodes. During the training process, we use the Levenberg-Marquardt (LM) algorithm and Gaussian-Newton algorithm to achieve fast convergence.

Our work supposes that the ability of testers is stable, and the quality of different test case is equality.

4. Case Study

4.1. Data Sets

Our method has been tested using three real-time control application and flight dynamic application data sets. These data sets are all from public data sets:

- DATA-1[4]: Musa's dataset--Software reliability measurement prediction application consisting of 136 failures.
- DATA-2[16]: flight dynamic application consisting of 118 failures.
- DATA-3[17]: Changjie Ma's embedded system data--A Neural Network Based Approach for Reliability Analysis of Software-intensive Equipment.

4.2. Implementation Steps

- Analyze the sample data, compute the learning curve interval.
- Select the sample data sets: 70% of the data for training and 30% of the data for prediction;
- Data pre-processing: the data is normalized by:

$$y=\frac{x-min\ (x)}{max(x)-min\ (x)} \tag{3}$$

- Determine the network structure: a three-layer network architecture, show as Fig. 2.
  1) The difficulty is determining the number of hidden layer nodes. Through the experimental comparison, we calculate the network mean square error of the MSE and determine the number of hidden nodes.
  2) In the training process of neural network, we use the rolling predict ideas, and gradually adjust the training error and offset.
- Predict the failure time
  1) The performance of prediction result is evaluated by average relative prediction error (AE), AE is defined as:

$$\mathrm{AE} = \frac{1}{(n-i_{min}+1)}\sum_i^n \left|\frac{\hat{x}_i-x_i}{x_i}\right| \times 100 \tag{4}$$

$\hat{x}_i$ is the predicted value of cumulative failure time, and x_i is the actual value of cumulative failure time and n is the number of failure time data.

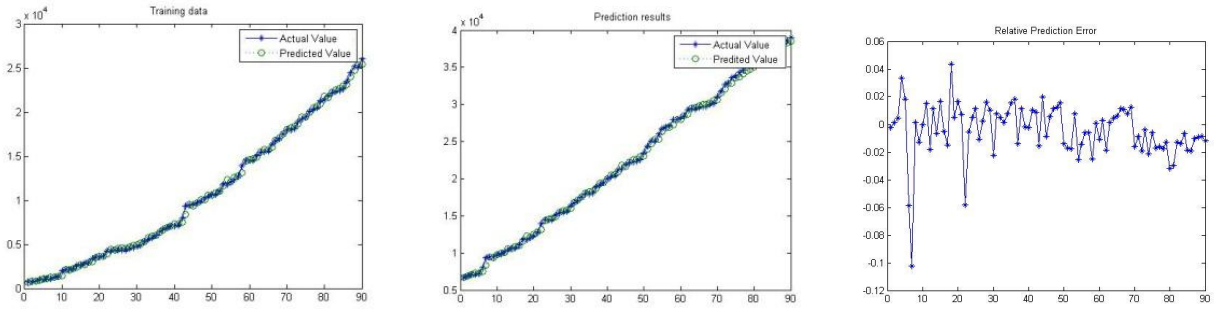
4.3. Results Analysis

Table 2 gives the number of hidden layer nodes. Through comparing the training effect of different hidden nodes (from one node to 20 nodes), calculating the MSE, we select the MSE minimum number of hidden nodes for the final network of hidden, and determine the structure of network.

Table 2. Hidden Layer Nodes Numbers

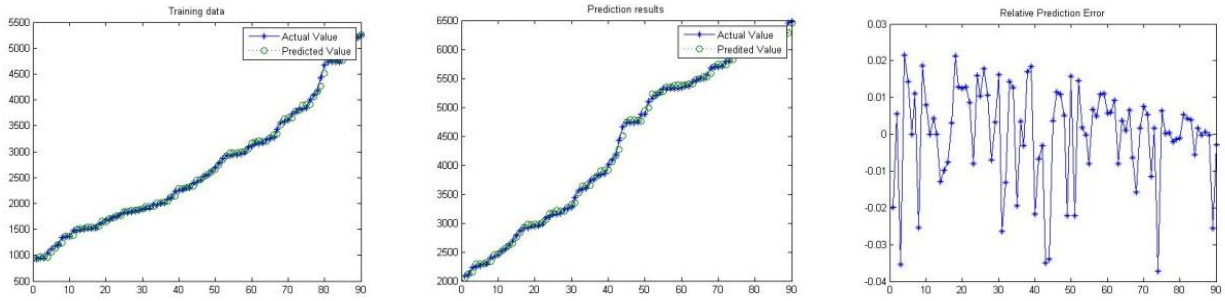
Dataset	Number of Hidden layer	MSE
Data-1	9	4.1693e-005
Data-2	13	7.0303e-005
Data-3	7	1.0536e-005

Fig. 3- Fig. 5 show the predict result of datasets.



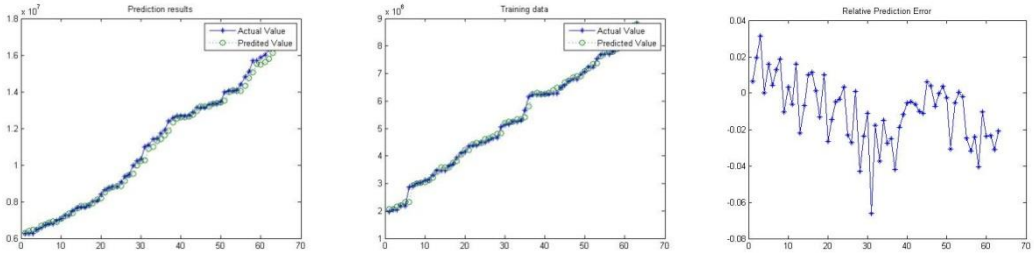
(a) (b) (c)

Fig. 3. Training and Prediction results of data-1. (a) Shows the training result, (b) Shows the prediction result, (c) Shows the relative prediction error



(a) (b) (c)

Fig. 4. Training and Prediction results of Data-2. (a) shows the training result, (b) shows the prediction result, (c) shows the relative prediction error.



(a) (b) (c)

Fig. 5. Training and Prediction results of Data-2. (a) shows the training result, (b) shows the prediction result, (c) shows the relative prediction error

Table 3. Average Relative Prediction Error in Data Sets

AE in data sets(%)				
Data sets	Proposed approach	L.Tian[18]	Park	Karunanithi
Data-1	1.32	1.79	2.28	2.50
Data-2	0.017	0.88	1.51	4.76

Table 3 compares our approach to other neural network prediction approach and NHPP model prediction results by average. The prediction results show that our approach has a better performance with the cumulative failure time prediction.

5. Conclusion

In this paper, we proposed a neural network based cumulative failure time prediction method which is depending on the rolling forecast. Through eliminate the learning curve effect and utilized the rolling forecast method, we can improve the prediction accuracy and the application of the model. In the future, we will consider to utilize the information of testing process to do the prediction, such an number of test cases, etc.

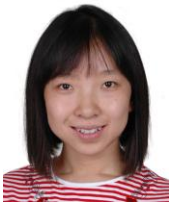
References

[1] American institute of aeronautics and astronautics, Aiaa software reliability engineering recommended practice.

[2] Lyu, M. R. (1996). *Handbook of Software Reliability Engineering*. McGraw-Hill, New York.

[3] Sitte, R. (1999). Comparison of software-reliability-growth predictions: Neural networks vs parametric-Recalibration. *IEEE Trans.Reliability*, 1999, *48(3)*, 285-291.

[4] Musa, J. D., Iannino, A., & Okumoto, K. (1989). *Software Reliability, Measurement, Prediction and Application*. McGraw-Hill.
[5] Xie, M. (1991). Software reliability modeling. *World Scientific Publishing Company*.
[6] Pham, H. (2000). *Software Reliability*. Springer-Verlag,
[7] Iannino, A., Musa, J. D., & Okumoto, K. (1984). Criteria for software reliability model comparisons. *IEEE Transacions on Software Engineering*.
[8] Karunanithi, N., & Malaiya, Y. K. (1992). The scaling problem in neural networks for software reliability prediction. *Proceedings of the third International IEEE Symposium of Software Reliability Engineering*.
[9] Karunanithi, N., & Malaiya, Y. K. (1996). *Neural Networks for Software Reliability*. McGraw-Hill, NewYork, 699-728.
[10] Cai, K. Y., Cai, L., Wang, W. D., Yu, Z. Y., & Zhang, D. (2001). On the neural network approach in software reliability nodeling. *The Journal of Systems and Software*, 47-62.
[11] Tian, L., & Noore. A. (2005). Evolutionary neural network modeling for software cumulative failure time prediction. *Reliability Engineering and System*.
[12] Su, Y. S., & Huang, C. Y. (2007). Neural-network-based approaches for software reliability estimation using dynamic weighted combinational model. *The Journal of Systems and Software*.
[13] Jun, Z. (2009). Predictiong software reliability with neural network ensembles. *Expert Systems with Applications*.
[14] Yin, Q. (2006). The Study on software reliability model based on neural network. PhD Thesis. Beijing Normal University.
[15] Norman, S. (2009). Updated software reliability metrics. *Journal of Aerospace Computing, Information, and Communication*.
[16] Park, J. Y., Lee, S., & Park. J. H. (1999). Neural network modeling for software reliability prediction from failure time data. *J Electr Eng Inform Sci*.
[17] Ma, C. J., Gu, G. C., Jing, Z., & Jun, N. (2010). A neural network based approach for reliability analysis of software-intensive equipment. *Proceedings of the 2010 Fifth International Conference on Internet Computing for Science and Engineering*.
[18] Liang, T., & Afzel, N. (2005). Evolutionary neural network modeling for software cumulative failure time prediction. *Reliability Egineering and System Safety*.
[19] Pradeep, K., & Yogesh, S. (2012). An empirical study of software reliability prediction using machine learning techniques. *International Journal of System Assurance Engineering and Management*.
[20] Kansal, Y., & Choudhary, S. (2014). Forecasting the Reliability of Software via Neural Networks. *International Journal of Computer Science & Information Technology*.



Tian Jie is currently is a Ph.D. student studying at School of Computer Science and Engineering, Beihang University. Her research interests include software reliability analysis and software testing.

Wu Ji is an associate professor, an assistant dean of School of Computer Science and Engineering (SCSE), Beihang University. He received his Ph.D. degree from Beihang University in 2003, and a M.S. degree from Second Research Institute of the China Aerospace Science and Industry Group in 1999. He focuses on the industry-oriented researches, and the main research interests include embedded system and software modeling and

verification,software requirement and architecture modeling and verification, safety and reliability assessment, and software testing. He was invited to visit Simula Research Laboratory for one year in 2012.

Yang Haiyan is a master and a lecturer in Beihang University. Her main research interests include software engineering, software safety and requirement engineering.

Liu Chao is a professor, the director of Software Engineering Institute (SEI), Beihang University (BUAA), Beijing, China. His research interests include software quality engineering, software testing, as well as software process improvement. He received his Ph.D. degree and M.S. degree in computer software and theory at Beihang University, and his B.S. degree in mathematics at Beijing University of Posts and Telecommunication.