Predicting Software Reliability with a Novel Neural Network Approach

Shirin Noekhah(✉), Naomie Binti Salim, and Nor Hawaniah Zakaria

Faculty of Computing, Universiti Teknologi of Malaysia, UTM,
81300 Johor, Malaysia
nshirin2@live.utm.my, shirinnoekhah@gmail.com,
{naomie,hawaniah}@utm.my

Abstract. With the application of software systems in variety critical field the complexity level of software has increased, so software reliability has become more and more difficult to guarantee. To overcome human power and time limitation, researchers have focused on a soft computing approach. Nevertheless, the new techniques - especially NN - have some problems, like no solid mathematical foundation for analysis, trap in local minima and a convergence problem. This paper proposed a model to predict software reliability by hybridizing the Multi-Layer Perceptron neural network (MLP) and Imperialist Competitive algorithm (ICA). This model has solved most of the previous problems, such as the convergence problem, requiring a large amount of data, and it can be applied in complex systems. Numerical results show that both the training and testing stages of proposed approach have greater accuracy in predicting the number of software failures compared to the existing approaches.

Keywords: Neural network · Software reliability · MLP · ICA algorithm

1 Introduction

Nowadays, computers have become an inseparable part of human life. They are used in wide areas such as the military, business and industrial sectors. One of the most important parts of each computer is the software part, so, having the proper software which can perform the desired task with a high standard is required. Software reliability is one of the significant factors of software quality, which is applied to quantify the profile of the system operations. Software reliability is the probability of failure-free software operation measured during a precise period of time within an exact environment (American National Standards Institute, ANSI 1991) [1].

Software reliability is a non-linear concept, so researchers have proved that an artificial neural network (ANN) can be more effective than traditional methods. ANN has the capability of non-linear mapping which causes applied in the field of time series prediction.

Neural network-based models have proven that they can be applied universally for any non-linear function with high accuracy. The first neural network model for software reliability prediction was proposed by Karunanithi et al. [2], Adnan et al. [3] and Park et al. [4] have also presented other software reliability prediction models using

F. Saeed et al. (eds.), *Recent Trends in Information and Communication Technology*, Lecture Notes on Data Engineering and Communications Technologies 5, DOI 10.1007/978-3-319-59427-9_93

neural networks, and proved that their results are better than analytical models. Most researchers use single-input single-output neural network architecture to create reliability models. Karunanithi et al., in [2], considered cumulative execution time as input and the number of failures as desired output. In contrast, in [5], they applied the failure number as input and the time of failure as output.

Cai et al., in [6], used the recent 50 inter-failure times as the multiple-delayed-inputs to predict the next failure time. They proved that the architecture of the neural network, especially the number of input layer neurons and hidden layer neurons, has an influence on the performance of the network. Cai et al. selected 20, 30, 40, and 50 input neurons, while Adnan et al. [3] applied 1, 2, 3, and 4 input neurons.

Tian et al. [7] proposed an online adaptive software reliability model by applying an evolutionary connectionist method based on multiple-delayed input single-output. They used current failure time data and a genetic algorithm to optimise the number of neurons in input and hidden layers. Moreover, they used a modified Levenberg-Marquardt (LM) algorithm with Bayesian regularisation to increase the predicting ability of software reliability. In another work, Tian et al., in [8], proposed an evolutionary neural network model for failure time prediction based on multiple-delayed-inputs single-output architecture. Unlike traditional models they model the inter-relationship among software failure data instead of the relationship between the time and number of failure data. In this case, both the input and the output of the neural network are failure time data. They used a genetic algorithm to optimize the neural network architecture. After the optimized neural network configuration was determined, a modified Levenberg-Marquardt (LM) algorithm with Bayesian regulation was applied as the desired neural network schema.

In [9], Jun presented an ensemble method of neural network to create non-parametric models to predict software reliability. He collected multiple predictors and proved that, in comparison with a single predictor, the combination of them brings about a better performance. Jin proposed a Support Vector Regression (SVR) model based on the hybridization of genetic algorithm (GA) and simulated an annealing algorithm (SA) to predict software reliability. A comparison between this model and other models was made. The results illustrated that this model reduces the predicting errors and provides higher relationship and performance prediction than other models [10]. The optimization of SVR's parameters is very difficult, so, many optimization algorithms have been exploited to discover the best combination of parameters, but still most of them do not perform very well. In another work, Jin improved estimation of distribution algorithms (EDA) to retain the variety of the population, and then proposed IEDA-SVR model which is a combination of EDA algorithm and SVR model to optimize SVR's parameters and having the better prediction performance [11]. Selecting the wrong model or weight assignment leads to ineffective software reliability prediction. Park and Baik proposed a software reliability prediction model to combine multiple software reliability models dynamically by exploiting decision trees learning of multi-criteria to reduce error pruning decision tree and increase the average prediction accuracy [12].

Wu et al., in [13] demonstrated a novel model based on a General Regression neural network (GRNN) to predict software reliability. In this model, they considered the affect test coverage in software reliability. It led to improvements in the accuracy of

the model. They trained GRNN with extended data to increase the predicting ability when dealing with a small size of data. Liu et al. applied a neural network for software reliability and found that it gave a better result than any other analytical models [14]. Chang et al. [15] proposed a new model based on counter propagation and back propagation neural networks to determine reliability parameters with a small size of data. Their results proved that the accuracy was improved in comparison with previous models. In [16], Vapnik demonstrated a new approach by hybridizing neural network and fuzzy logic. The results showed that a neural fuzzy system enhanced predicting accuracy more than GRNN. In another attempt, Recurrent Neural Network (RNN) in combination with Back-propagation Through Time (RNNBPTT) learning has been exploited by Bhuyan et al. to predict software reliability. The results proved that RNN performs an accurate and consistent behavior in reliability prediction [17]. Bal et al. made an investigation on an ensemble technique by the combination of different artificial neural networks to predict software reliability and compare with other traditional models. They evaluate their ensemble model on three benchmark datasets by using artificial neural network approach along with a mathematical linear model [18].

Different types of learning algorithms can be applied to train neural network. Back Propagation algorithm (BP), Genetic algorithm (GA) and Evolutionary algorithms are the most popular algorithms. These algorithms have some weaknesses which motivate us to exploit more powerful evolutionary algorithm (ICA) which can overcome most of these problems. BP hardly achieves full sampling of final model, traps in the local minima and has slow convergence rate. On the other hand, PSO has other problems such as no solid mathematical foundation for analysis, limitation in real time application and problem for initialize parameters and find best solution.

In comparison with these algorithms ICA covers most of the weaknesses of previous mentioned algorithms. This algorithm not only uses sufficient parameters and mathematical equations, but also, never traps in the local minima and gives the best and optimal solution. In this case we apply ICA for training MLP because of all advantages of ICA which helps MLP to predict reliability more accurate.

2 Software Reliability Concepts

Software is embedded in many devices, so the statistics of software failure grow rapidly. It consists of user interface problems to directly programming errors. In software reliability, the expected outcome compared with the output of software in a specific environment and condition with desirable data processing. In other words, it refers to the correctness of software. There is an intimate relationship between software failures and reliability; more faults cause the reliability to decrease.

There are many factors which affect software reliability. The more popular ones are rate of software failure, the number of failures, mean of failures and software failure intensity function. Software failures usually occur during execution time. As illustrated in Fig. 1, t_i is the execution time of ith software failure. In this figure, t_i is the time interval between ith and $(i - 1)$th failure ($\Delta t_i = t_i - t_{i-1}$).

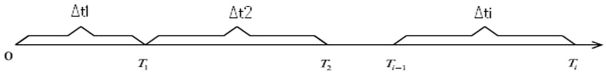

Fig. 1. Software failure process

3 ICA-MLP Software Reliability Prediction Model

In the software engineering field, different types of models have been exploited to estimate different tasks. By applying a model, developers can allocate resources more reasonably and develop software projects in a shorter time. Having a model can help us to reduce the cost and risk of designing and developing software. Figure 2 presents the study framework.

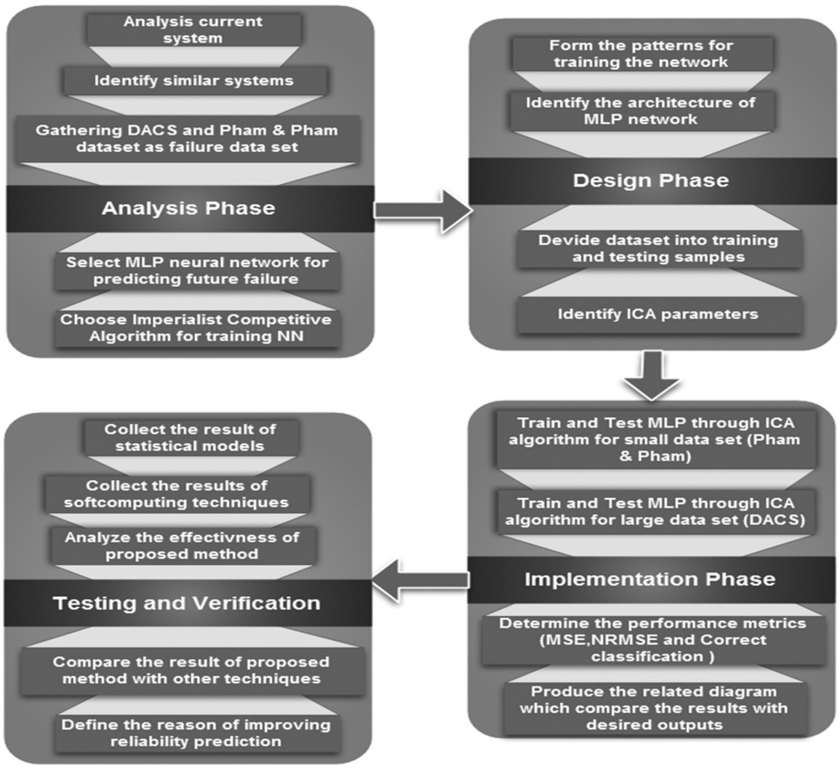

Fig. 2. Research framework

In this paper, a hybridised model for software reliability prediction has been proposed. This model is a data driven model which uses time series analysis to predict the number of failures which will occur in the software system. This model is called ICA-MLP as it is a hybridisation of the Imperialist Competitive Algorithm (ICA) and

the Multi-Layer Perceptron (MLP). In this model, ICA algorithm has been used for training MLP to improve the ability of the neural network to predict the number of failures in future. To achieve this goal, two types of different datasets (DACS and Pham & Pham) have been exploited to train and test the neural network. Each neural network needs some mathematical function to update the weights of the connections between two layers, so, tangent hyperbolic function has been used to upgrade them.

As Pham & Pham dataset is not large enough, 80 countries along with 100 decades have been allocated. On the other hand, as DACS dataset is large, the algorithm is initialized with 200 countries and 250 decades to test proposed model. These numbers have been selected according to literatures and trial and error tests. The details of the algorithm have been explained comprehensively in [19].

3.1 ICA-MLP Software Reliability Prediction Model

Most researchers apply two specific datasets, which are DACS and Pham & Pham datasets. In this study, same datasets have been exploited. The main reason for applying these datasets is the difference in their size, so the flexibility and reliability of proposed model in terms of failure prediction can be tested more accurately.

Pham and Pham presented a software reliability dataset based on the times of inner failure of software. In this dataset, i is the failure number and is the inner failure time. On the other hand, DACS (Data and Analysis Centre for Software) is based on the failure interval time to help the developers to control testing, predicting and modelling software reliability. This dataset contains the failure data of 16 different projects. In this study, military dataset (No. 40) has been selected to propose the software reliability model. It includes 180,000 instructions, 101 software failure numbers and was collected during the system testing phase. The dataset values have been changed to the proper values in order to improve the accuracy and convergence speed of proposed model. To perform this task, first, the model normalizes the data in a range [−1, 1] before sending it into the input layer. Then the data from output layer is denormalized into the original value before normalising.

3.2 ICA-MLP Experimental Results

The results of ICA-MLP model performed on two mentioned datasets have been illustrated in Table 1.

Table 1. ICA-MLP results (PHAM & PHAM and DACS datasets)

Parameter	Pham & Pham	DACS
MSEtrain (BestCost)	0.0133	0.0020
MSEtest	0.0078	0.0516
Correct Classification Train (%)	0.9824	0.9987
Correct Classification Test (%)	0.9914	0.9988

The results prove that there is an inverse relation between dataset size and the system accuracy. Increasing dataset size and training samples help to increase the software accuracy by minimizing the error value (MSE). Increasing the number of training data samples can make the training process of the neural network to be more efficient. So, the network's tolerance will be gradually increased and can classify (memorize) the testing pattern.

The Pham & Pham result shows the fact that, by having a small sized data set, the correction classification of the neural network will be decreased and consequently the amount of MSE will increase. The neural network ability for training patterns classification can be calculated by correction classification metrics. In contrast, during the testing phase these values determine to what extent the network can memorize the training patterns and, according to this value, it classifies the input test patterns. Figures 3, 4, 5 and 6 present the neural network output for training and testing input patterns of DACS and Pham & Pham datasets.

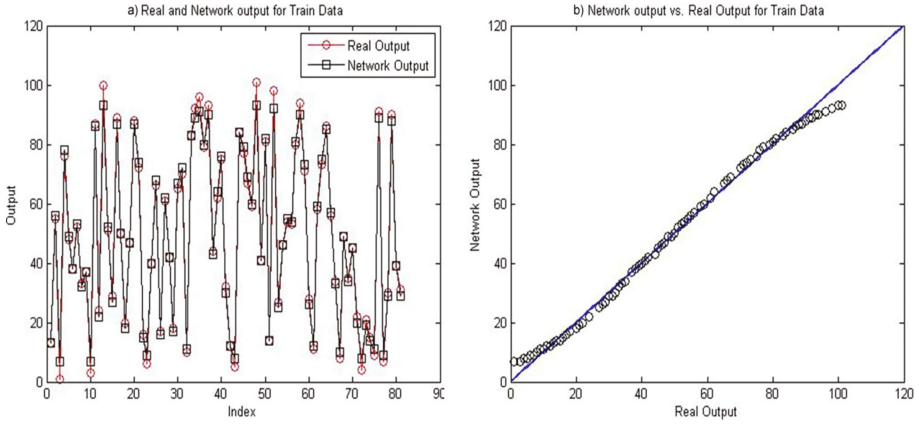

Fig. 3. (a) Outputs comparison for training data samples (DACS), (b) Benchmark evaluation (DACS)

Randomization technique usually has been used for training of neural network to initialize the connections' weights within a network. Consequently, there are varieties of weights convergence for each training phase. Despite of the fact that the prediction results are different, they are so close to each other. The training procedure is run for few times, the results are taken and, finally, the outputs' average is computed. The investigations have proved that soft computing techniques, such as neural network, give more accurate prediction than analytical methods which are no longer effective.

During analysing the results of the neural network, this fact should be notice that output results will be different during each training procedure. As the training methods initialise weights randomly, even though the training method, network architecture and dataset are same, but the results will be different. Still with this weak point their performance and accurateness are more significant than other techniques. In this section, the proposed model will be compared with other neural network techniques based on their learning algorithm and neural network architecture.

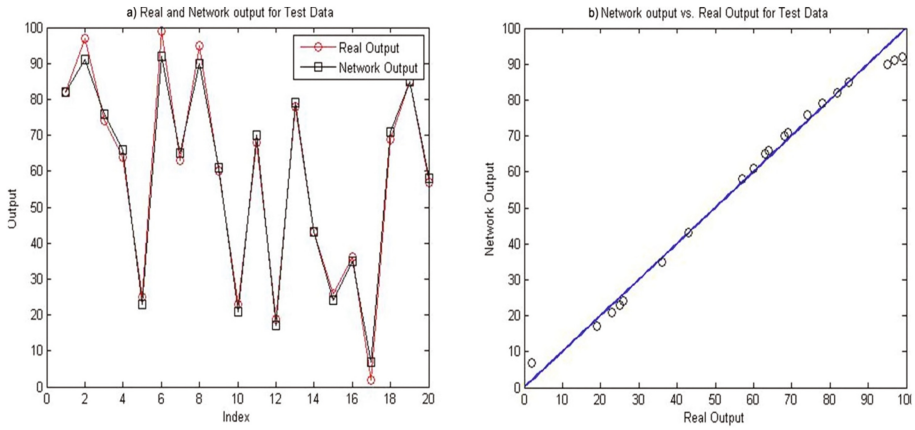

Fig. 4. (a) Outputs comparison for testing data samples (DACS), (b) Benchmark evaluation (DACS).

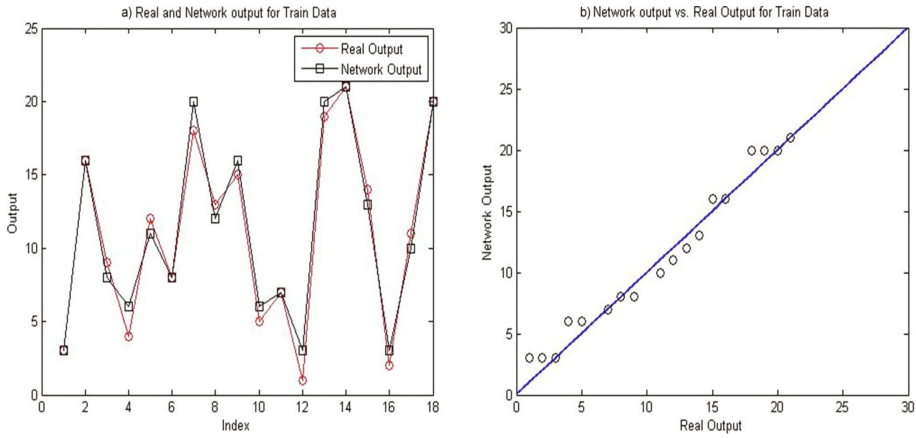

Fig. 5. a) Outputs comparison for training data samples (Pham & Pham), (b) Benchmark evaluation (Pham & Pham)

3.2.1 Experimental Results of Comparison

Many different neural network architectures have been proposed by researchers to improve the prediction of software reliability, but four of them include MLP (Multi-Layer Perceptron, RBF (Radial Basis Function), Elman Recurrent and Fuzzy Neural Network have become more popular compared with others. So, the proposed technique is compared with these approaches. Table 2 presents the experimental results of MSE for the software reliability prediction by implementing these techniques on exploited dataset.

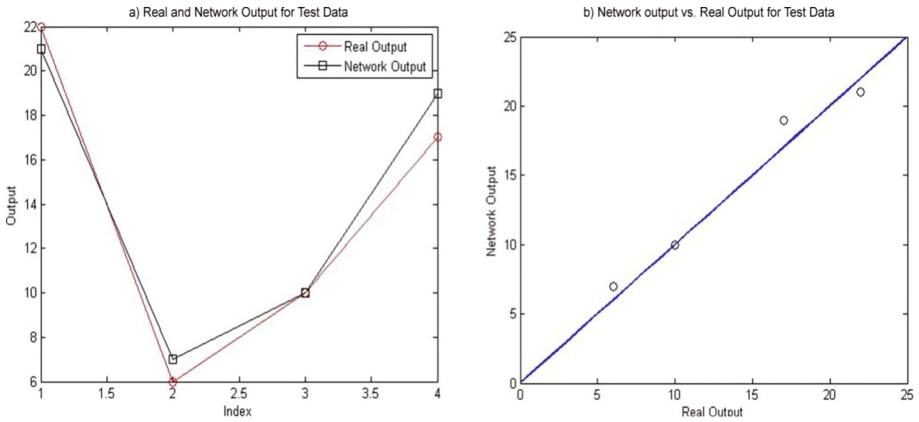

Fig. 6. (a) Outputs comparison for testing data samples (Pham & Pham), (b) Benchmark evaluation (Pham & Pham)

Table 2. MSE comparison of ICA-MLP and other neural network architectures.

Method	Training data	Testing data
ICA-MLP	0.0020	0.0516
RBFN	1.6465	0.1591
Elman	0.1625	0.1394
ANFIS	1.3364	0.9079

From this table, it can be observed that ICA-MLP model outperforms other techniques for both training and testing data samples due to efficient mechanism of algorithm in terms of weights' adjustment and convergence speed improvement. Elman network results the second best answer since it considers the dynamic behaviour of both system and software reliability datasets. As RBFN gives almost the same priority to all input training samples, the convergence rate will be reduced. On the other hand, the run time of this algorithm is longer which increase the cost and risk of the model. The worst result belongs to ANFIS for both training and testing data samples due to its complex architecture since it combines two heavy models named Neural network and Fuzzy Logic. This structure can be applied and be efficient in other domains but not for software reliability. According to these results, ICA-MLP gives the best result compared with other techniques. From another point of view, ICA-MLP model is compared with other techniques which exploit different hybridized techniques for training neural network. NRMSE experimental results have been presented in Table 3 which illustrates the differences among these techniques.

Table 3. NRMSE comparison of ICA-MLP and other neural network techniques.

Method	NRMSE
BPNN	0.145541
TANN	0.150355
PSN	0.157922
MARS	0.15267
GRNN	0.166883
MLR	0.147881
TreeNet	0.161121
DENFIS	0.147641
ICA-MLP	0.08574

4 Conclusion

In this study, the hybridisation of MLP neural network with ICA algorithm is processed. This model is used to predict the future number of software failures to increase the reliability of the system. Moreover, the effectiveness and performance of the hybrid neural network model on software reliability were analysed. ICA algorithm is suitable for both a small and a large amount of datasets, while most of the other models are suitable for one type of dataset. Based on the findings of the study, it can be seen that the MLP-ICA model has less complexity among other methods, especially in terms of computation formulas, since it applied the evolutionary pattern by applying the systematic finding optimal solution. The results of the study also indicated that this model has the lowest MSE and NRMSE in comparison with other models.

This study has only focused on using neural network techniques for software reliability prediction. It is recommended that further studies use techniques to cover this range of uncertainty. The best approach which can consider and simulate the uncertainty phenomena is fuzzy logic. As there are some limitations to fuzzy logic, such as not having the proper level of complexity of computation formula, a combination of fuzzy logic and the neural network approach are suggested.

Acknowledgments. This work is supported by Ministry of Higher Education (MOHE) and Research Management Centre (RMC) at the Universiti Teknologi Malaysia (UTM) under Research University Grant Category (R.J130000.7828.4F719).

References

1. http://webstore.ansi.org/RecordDetail.aspx?sku=R-013-1992
2. Karunanithi, N., Whitley, D., Malaiya, Y.K.: Prediction of software reliability using connectionist models. IEEE Trans. Software Eng. **18**(7), 563–574 (1992)
3. Adnan, W.A., Yaacob, M.H.: An integrated neural-fuzzy system of software reliability prediction. In: First International Conference on Software Testing, Reliability and Quality Assurance, Conference Proceedings, pp. 154–158. IEEE (1994)

4. Park, J.Y., Lee, S.U., Park, J.H.: Neural network modeling for software reliability prediction from failure time data. J. Electr. Eng. Inf. Sci. **4**(4), 533–538 (1999)
5. Karunanithi, N., Whitley, D., Malaiya, Y.K.: Using neural networks in reliability prediction. IEEE Softw. **9**(4), 53–59 (1992)
6. Cai, K.Y., Cai, L., Wang, W.D., Yu, Z.Y., Zhang, D.: On the neural network approach in software reliability modeling. J. Syst. Softw. **58**(1), 47–62 (2001)
7. Tian, L., Noore, A.: On-line prediction of software reliability using an evolutionary connectionist model. J. Syst. Softw. **77**(2), 173–180 (2005)
8. Tian, L., Noore, A.: Evolutionary neural network modeling for software cumulative failure time prediction. Reliab. Eng. Syst. Saf. **87**(1), 45–51 (2005)
9. Jun, Z.: Prediction of software reliability using connectionist models. Expert Syst. Appl. **36**, 2116–2122 (2009)
10. Jin, C.: Software reliability prediction based on support vector regression using a hybrid genetic algorithm and simulated annealing algorithm. IET Softw. **5**(4), 398–405 (2011)
11. Jin, C., Jin, S.W.: Software reliability prediction model based on support vector regression with improved estimation of distribution algorithms. Appl. Soft Comput. **15**, 113–120 (2014)
12. Park, J., Baik, J.: Improving software reliability prediction through multi-criteria based dynamic model selection and combination. J. Syst. Softw. **101**, 236–244 (2015)
13. Wu, Y., Yang, R.: Study of software reliability prediction based on GR neural network. In: 9th International Conference on Reliability, Maintainability and Safety (ICRMS), pp. 688–693. IEEE, June 2011
14. Liu, M.C., Kuo, W., Sastri, T.: An exploratory study of a neural network approach for reliability data analysis. Qual. Reliab. Eng. Int. **11**(2), 107–112 (1995)
15. Chang, P.T., Lin, K.P., Pai, P.F.: Hybrid learning fuzzy neural models in forecasting engine system reliability. In: Proceeding of the Fifth Asia Pacific Industrial Engineering and Management Systems Conference, pp. 2361–2366 (2004)
16. Vapnik, V.N., Vapnik, V.: Statistical Learning Theory, vol. 1. Wiley, New York (1998)
17. Bhuyan, M.K., Mohapatra, D.P., Sethi, S.: Prediction strategy for software reliability based on recurrent neural network. In: Behera, H.S., Mohapatra, D.P. (eds.) Computational Intelligence in Data Mining—Volume 2, pp. 295–303. Springer, New Delhi (2016)
18. Bal, P.R., Jena, N., Mohapatra, D.P.: Software reliability prediction based on ensemble models. In: Singh, R., Choudhury, S. (eds.) Proceeding of International Conference on Intelligent Communication, Control and Devices, pp. 895–902. Springer, Singapore (2017)
19. Noekhah, S., Hozhabri, A.A., Rizi, H.S.: Software reliability prediction model based on ICA algorithm and MLP neural network. In: 7th International Conference on e-Commerce in Developing Countries: With Focus on e-Security (ECDC), pp. 1–15. IEEE, April 2013