Applied Soft Computing xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Robust feedforward and recurrent neural network based dynamic weighted combination models for software reliability prediction

Pratik Roy[a], G.S. Mahapatra[b,*], Pooja Rani[b], S.K. Pandey[b], K.N. Dey[a]

[a] Department of Computer Science and Engineering, University of Calcutta, Kolkata 700009, India
[b] National Institute of Technology – Puducherry, Karaikal 609605, India

ARTICLE INFO

Article history:
Received 10 January 2013
Received in revised form 29 March 2014
Accepted 11 April 2014
Available online xxx

Keywords:
Software reliability growth model
Dynamic weighted combination model
Artificial neural network
Genetic algorithm
Reliability prediction

ABSTRACT

Traditional parametric software reliability growth models (SRGMs) are based on some assumptions or distributions and none such single model can produce accurate prediction results in all circumstances. Non-parametric models like the artificial neural network (ANN) based models can predict software reliability based on only fault history data without any assumptions. In this paper, initially we propose a robust feedforward neural network (FFNN) based dynamic weighted combination model (PFFNNDWCM) for software reliability prediction. Four well-known traditional SRGMs are combined based on the dynamically evaluated weights determined by the learning algorithm of the proposed FFNN. Based on this proposed FFNN architecture, we also propose a robust recurrent neural network (RNN) based dynamic weighted combination model (PRNNDWCM) to predict the software reliability more justifiably. A real-coded genetic algorithm (GA) is proposed to train the ANNs. Predictability of the proposed models are compared with the existing ANN based software reliability models through three real software failure data sets. We also compare the performances of the proposed models with the models that can be developed by combining three or two of the four SRGMs. Comparative studies demonstrate that the PFFNNDWCM and PRNNDWCM present fairly accurate fitting and predictive capability than the other existing ANN based models. Numerical and graphical explanations show that PRNNDWCM is promising for software reliability prediction since its fitting and prediction error is much less relative to the PFFNNDWCM.

1. Introduction

Computer plays an essential role in our modern civilization. Computer systems are managed by software and hence software should be reliable and fault free. Perfect measurement of software reliability has become one of the most important tasks for development of good quality software. According to ANSI definition, software reliability is defined as the probability of failure-free software operation for a specified period of time in a specified environment [1,4]. Software reliability is the most important quality metric. One can estimate the duration of software testing period by measuring software reliability. Software reliability prediction plays a vital role in today's rapidly growing complex software development process. In the last three decades, many SRGMs have been proposed in the literature to predict the relationship between software failure and time. The software reliability models can be divides into parametric and non-parametric model. In parametric models, the model parameters are estimated based on the assumptions about the behavior of the software faults, failure processes and development environments. The most well-known parametric models are the nonhomogeneous Poisson process (NHPP) based SRGMs [9–18]. However, it has been shown that none single model can produce accurate prediction results in all circumstances [19]. On the other hand, non-parametric model like the ANN based model comprise the flexibility to predict software reliability based on only fault history data without any assumptions of the parametric model. It has been exposed that ANN based non-parametric models can produce better predictive quality than the parametric models [6–8,24]. Different types of FFNNs and RNNs have been applied to predict cumulative number of detected faults where the software execution time is used as the input to the network. Several approaches have been developed to combine various existing

* Corresponding author. Tel.: +91 9433135327; fax: +91 4368231665.
E-mail addresses: pratik.roy43@gmail.com (P. Roy), g_s_mahapatra@yahoo.com (G.S. Mahapatra), pooja.nitpdy@gmail.com (P. Rani), kndey55@gmail.com (K.N. Dey).

http://dx.doi.org/10.1016/j.asoc.2014.04.012
1568-4946/

software reliability models to produce a dynamic weighted combination model whose prediction accuracy is much better than the component models [7,20,21].

In this paper, firstly we propose a robust FFNN based dynamic weighted combination model for software reliability prediction. The PFFNNDWCM is constructed by combining the traditional SRGMs. The SRGMs are merged based on the dynamically evaluated weights determined by the training algorithm of the FFNN. In this study, we select four well-known traditional parametric SRGMs to develop the PFFNNDWCM. Based on this proposed FFNN architecture, we also propose a robust RNN based dynamic weighted combination model for more accurate prediction of software reliability. We construct the proposed FFNN and RNN by using different activation functions for hidden layer neurons of the ANNs. Cumulative software execution time is the input and the predicted cumulative number of software failures is the output of the proposed networks. We propose the RNN modeling approach to learn the temporal patterns of the failure data dynamically which has a significant impact on network prediction performance. We propose a real-coded GA based learning algorithm to train the proposed ANNs using the software failure data sets by globally optimizing the weights and parameters of the ANNs. We compare the performances of the proposed models with the existing ANN based dynamic weighted combination model [7] and ANN ensemble model [8] in the software reliability literature. The proposed models are also compared with the models that can be derived by using three or two of the selected four models. We explain the fitting and predictability of the different models through three real software failure data sets.

The rest of the paper has been organized in the following way: Section 2 introduces some related works about the application of ANNs in software reliability prediction. Section 3 presents the proposed FFNN and RNN based modeling approach for the development of the dynamic weighted combination models for software reliability prediction. In Section 4, we propose GA based learning algorithm to train the proposed FFNN and RNN. Section 5 describes the nature of the software failure data sets used in the experiments. Section 6 describes the model comparison criteria which are used to compare the prediction performance of the different models. Section 7 shows the experimental results based on the three real software failure data sets and some conclusions are drawn in Section 8.

2. Literature survey

This section describes some related works where ANNs [2] have been applied in software reliability modeling and prediction. Karunanithi et al. [6,22] first proposed ANNs to predict the software reliability by using the execution time and the cumulative number of detected faults as the input and the desired output of the network, respectively. Sitte [23] compared the predictive capability of two different software reliability prediction methods: neural networks and recalibration for parametric models. Khoshgoftaar and Szabo [25] applied ANN to predict the number of faults in programs during testing. Cai et al. [26] proposed the back-propagation algorithm based ANN for software reliability prediction and used the recent 50 inter-failure times as input to predict the next failure time. Ho et al. [27] proposed a modified RNN for modeling and prediction of software failures. Tian and Noore [28,29] proposed an evolutionary ANN modeling approach to predict the software cumulative failure time based on multiple-delayed-input single-output architecture. Su and Huang [7] proposed an ANN based dynamic weighted combination model for software reliability prediction. Hu et al. [30] applied RNNs to model fault detection and fault correction processes together. Kiran and Ravi [31] proposed a non-linear ensemble-based approach for software reliability prediction. Kapur et al. [32] applied ANN methods to build SRGMs considering faults of different complexity. Zheng [8] used the ensemble of ANNs to predict software reliability. Kapur et al. [33] presented a new dimension to build an ensemble of different ANN for complex software architectures to improve the estimation accuracy. Li et al. [20] proposed adaboosting-based approaches for combining the parametric SRGMs to significantly improve the estimating and forecasting accuracy. Wu et al. [21] proposed a dynamically-weighted software reliability combination model to improve the predictive quality. Mohanty et al. [34] proposed novel recurrent architectures for genetic programming and group method of data handling to predict software reliability.

Table 1
Selected SRGMs with mean value function.

SRGM	Mean value function
Goel–Okumoto model (G)	$a(1-e^{-bt})$
Yamada delayed s-shaped model (Y)	$a(1-(1+bt)e^{-bt})$
Inflection s-shaped model (I)	$\frac{a(1-e^{-bt})}{1+\beta e^{-bt}}$
Logistic growth curve model (L)	$\frac{a}{1+be^{-ct}}$

3. Proposed ANN based software reliability models

We know that none single SRGM can be trusted to produce accurate prediction results in all circumstances. Here, we propose FFNN and RNN-based dynamic weighted combination models which combine the traditional SRGMs to form a dynamic weighted combination of software reliability models.

We consider four well-known traditional statistical SRGMs namely, Goel–Okumoto model [9], Yamada delayed s-shaped model [16], inflection s-shaped model [14] and logistic growth curve model to develop the dynamic weighted combination models which can combine the SRGMs based on the dynamically assigned weights determined by the training of the proposed ANNs.

The mean value function of the selected four SRGMs are given in Table 1.

3.1. PFFNNDWCM

PFFNNDWCM is constructed as shown in Fig. 1 to develop the dynamic weighted combination model by combining the selected four SRGMs. The proposed FFNN has one hidden layer with single neuron in each of the input and output layers. The hidden layer consists of four neurons representing the four SRGMs to be combined.

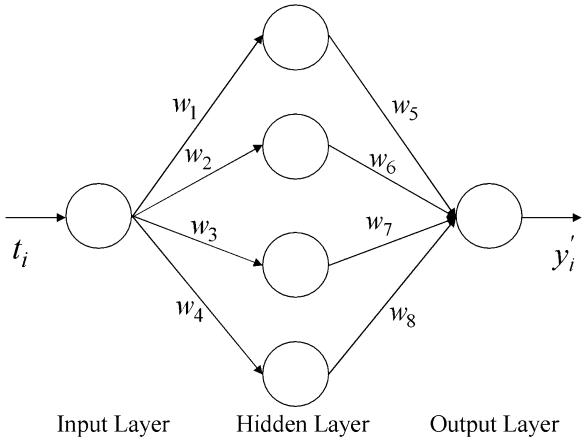

Fig. 1. PFFNNDWCM architecture.

Please cite this article in press as: P. Roy, et al., Robust feedforward and recurrent neural network based dynamic weighted combination models for software reliability prediction, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.012

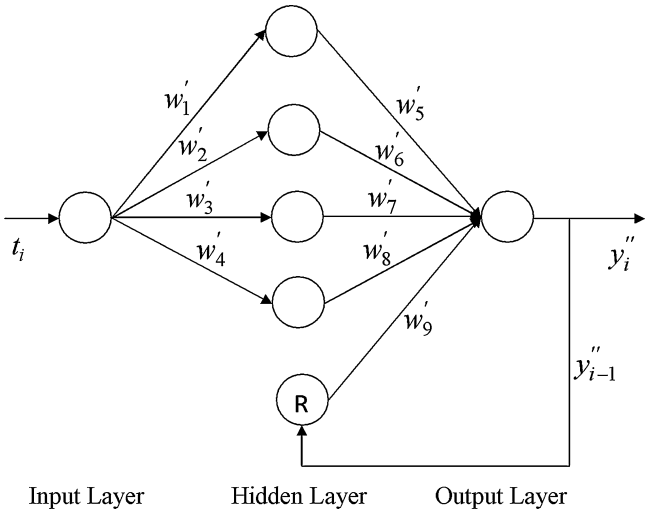

Fig. 2. PRNNDWCM architecture.

The cumulative software execution time (t_i) is the input of the FFNN and output of the FFNN is the predicted cumulative number of software failures (y_i'), where i is the cumulative software execution time sequence index. We use $1-e^{-x}$, $1-(1+x)e^{-x}$, $(1-e^{-x})/(1+a_1e^{-x})$ and $1/(1+b_1e^{-x})$ as the activation functions for the four hidden layer neurons of the proposed FFNN. Here, the activation functions used for hidden layer neurons are developed according to the mean value function of the selected SRGMs. Linear activation function $f(x)=x$ is used in the output layer neuron of the ANN. Hence, the output of the FFNN can be evaluated as follows:

$$y_i' = w_5(1-e^{-w_1t_i}) + w_6(1-(1+w_2t_i)e^{-w_2t_i}) + \frac{w_7(1-e^{-w_3t_i})}{1+a_1e^{-w_3t_i}} + \frac{w_8}{1+b_1e^{-w_4t_i}} \quad (1)$$

where w_j (>0), $j=1, 2, \ldots, 8$, are the weights of the FFNN and their values are determined by the training algorithm. Here, a_1 and b_1 ($a_1, b_1 > 0$) are activation function parameters whose values are also evaluated through the learning of the proposed FFNN.

In this PFFNNDWCM, we merge different traditional exponential and s-shaped SRGMs based on the proposed FFNN approach and the weights of each model are determined dynamically by training the FFNN according to the characteristics of the selected failure data sets.

3.2. PRNNDWCM

RNN is a feedback ANN in which the current output/state is a function of the previous output/state and the current input. RNN includes the dynamic temporal property internally, which have great impact on network prediction performance. RNN has the advantage in feedbacking the data generated by the network to be used in future iterations and the feedback path enables the network to learn temporal patterns/sequences dynamically.

3.2.1. Model architecture

We consider that the cumulative number of failures at time t_i is a function of the cumulative number of failures at time t_{i-1} and the current execution time t_i, i.e. $y_i'' = f(y_{i-1}'', t_i)$. We need to forecast y_i'' by use of y_{i-1}''. We construct the PRNNDWCM which has the feedback path from the output layer to the hidden layer via the recurrent neuron denoted by R as shown in Fig. 2.

Similar to the FFNN modeling, the cumulative software execution time (t_i) is the input of the RNN and the predicted cumulative number of software failures (y_i'') is the output of the RNN. We apply the activation functions $1-e^{-x}$, $1-(1+x)e^{-x}$, $(1-e^{-x})/(1+a_2e^{-x})$ and $1/(1+b_2e^{-x})$ in the four hidden layer neurons and linear activation function in the output layer neuron of the RNN. The RNN has feedback path from the output of the network to the input of the output layer neuron by which it can recognize the cumulative number of failures in execution time t_{i-1}. The feedback path allows the network to employ the previous output of the network in the current state of the network. Here, w_9' is the feedback weight determined by the training algorithm. The output of the RNN is given as follows:

$$y_i'' = w_5'(1-e^{-w_1't_i}) + w_6'(1-(1+w_2't_i)e^{-w_2't_i}) + \frac{w_7'(1-e^{-w_3't_i})}{1+a_2e^{-w_3't_i}} + \frac{w_8'}{1+b_2e^{-w_4't_i}} + w_9'y_{i-1}'' \quad (2)$$

where $w_j' > 0$, for $j=1, 2, \ldots, 9$ and $a_2, b_2 > 0$.

The values of all the weights and parameters of the RNN are determined by the learning algorithm and the four SRGMs are combined based on these dynamically evaluated weights in accordance with the failure data used to train the RNN. This PRNNDWCM has the capability of incorporating the cumulative number of failures in previous state/execution time, which immensely control the predictive power of this model.

4. Network learning through proposed GA

The PFFNNDWCM and PRNNDWCM of software reliability modeling are trained by a proposed real-coded GA [3] which is a robust evolutionary optimization search technique modeled from natural genetics to find global optimal solution.

We apply real-coded GA to encode the weights and parameters of the proposed FFNN and RNN as chromosomes. The chromosomal encoding of the weights and parameters of the FFNN and RNN are given by $[w_1\ w_2\ w_3\ w_4\ w_5\ w_6\ w_7\ w_8\ a_1\ b_1]$ and $[w_1'\ w_2'\ w_3'\ w_4'\ w_5'\ w_6'\ w_7'\ w_8'\ w_9'\ a_2\ b_2]$, respectively. The hidden layer activation function parameters of the proposed FFNN and RNN are also combined with the weights of the ANNs in the chromosomal representation to be evaluated by the proposed GA. Each weight and parameter of the ANNs is represented by a gene of the chromosomes.

Here, the fitness function of the proposed GA is the network error function which is to be minimized. We use normalized root mean square error (*NRMSE*) generated by the ANN as the error function as follows:

$$NRMSE = \sqrt{\frac{\sum_{i=1}^{n}(m_i-y_i)^2}{\sum_{i=1}^{n}y_i^2}} \quad (3)$$

where n is the number of data points used to train the ANN, y_i is the actual value and m_i is the predicted value generated by the ANNs for the ith data point in the software failure data set. The values of *NRMSE* are minimized by the proposed GA during learning of the each network.

Tournament selection process is used for selection of the parent chromosomes from the population to be involved in reproduction processes such as crossover, mutation. After selection, arithmetic crossover and uniform mutation operations are performed on the selected parent chromosomes to reproduce more fit individuals/chromosomes, i.e. offsprings with higher fitness value. More better solution can be achieved as the generation number of the GA increases. The GA will be stopped when the desired stopping criteria has been satisfied.

Because of the stochastic nature of GA, 100 trials have been performed for 1000 generations as the maximum number of

Please cite this article in press as: P. Roy, et al., Robust feedforward and recurrent neural network based dynamic weighted combination models for software reliability prediction, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.012

generations in each trial and the best solution from among the 100 trials is considered as the final optimal solution.

GA implementation for training of FFNN and RNN for software reliability prediction:

The proposed real-coded GA to train the PFFNNDWCM and PRNNDWCM is described below:

Step 1: Initialize the parameters of the GA as follows: *population _size* = 40, *crossover _rate* = 0.9, *mutation _rate* = 0.01 and *max _gen* = 1000, where *max _gen* represents the maximum number of generations.
Step 2: Set *gn* = 1, where *gn* denotes the current generation number.
Step 3: Encode the weights and parameters of the ANN into chromosome.
Step 4: Generate the initial population by initializing the chromosomes for the population.
Step 5: Evaluate the fitness value of each chromosome in the population by considering the fitness function (3).
Step 6: Select parent chromosomes from the population by tournament selection process of size 3.
Step 7: Apply arithmetic crossover and uniform mutation operations on the selected parent chromosomes with the *crossover _rate* and *mutation _rate*, respectively, to produce offsprings with higher fitness value.
Step 8: If the termination criteria is satisfied, go to step 11.
Step 9: Increase *gn* by unity.
Step 10: Go to step 5.
Step 11: Evaluate the fitness value of each offspring in the population.
Step 12: Return the chromosome with best fitness value (which is the optimal setting of the weights and parameters for the ANN).
Step 13: Stop.

5. Software failure data

Software failure may occur during the testing process due to the hidden faults in the software. The software failure data are normally arranged in pairs $\{t_i, y_i\}$ where y_i is the cumulative number of failures in the software execution time t_i and i is the cumulative software execution time sequence index. The objective of software reliability prediction is to accurately predict the number of failures in the future execution time based on the historical software failure data. In this experiment, three real software failure data sets *DS*1, *DS*2 and *DS*3 are used to check the performance and validity of the PFFNNDWCM and PRNNDWCM. The description of the data sets are given below.

- Data Set-1(*DS*1): This data set was reported by Musa [5] based on the 136 failures observed from a real-time command and control system with 21,700 assembly instructions.
- Data Set-2(*DS*2): This data set was collected (Musa et al. [5]) from a military application with 61,900 instructions and 38 failures.
- Data Set-3(*DS*3): This data set by Lyu [1] was collected from a single-user workstation with 397 failures.

The cumulative software execution time (t_i) and the cumulative number of failures (y_i) of each data set are normalized in the range of [0, 1] before feeding to the ANNs.

6. Model performance measures

Some meaningful performance comparison criteria are needed to compare the fitting and predictive powers of the different ANN based software reliability models under consideration. In this experiment, we adopt the variable-term prediction and end-point prediction approaches.

6.1. Fitting performance

The fitting performance of an ANN based software reliability model demonstrates how much fit the model to the software failure data. To evaluate the fitting performance, first the ANN is trained using a part of the failure data (training data). The trained ANN is then used to estimate the same failure data which was used to train the ANN. The estimated failure number m_i at the execution time t_i is compared with the actual failure number y_i from the data set. The fitting performance of an ANN based model is measured in terms of the relative error (*RE*) and average error (*AE*) as follows:

$$RE_i = \left| \frac{m_i - y_i}{y_i} \right| \times 100 \quad (4)$$

$$AE = \frac{1}{n} \sum_{i=1}^{n} RE_i \quad (5)$$

where n is the number of data points used to train the ANN and to be estimated.

6.2. Variable-term prediction

In this approach, only part of the failure data is used to train the ANN. Then, the trained ANN is used to predict the rest of the failure data. The predicted failure number m_i at the execution time t_i is compared with the actual failure number y_i from the data set. The variable-term predictability of a model is measured in terms of the relative error (*RE*) and average error (*AE*) as follows:

$$AE = \frac{1}{p} \sum_{i=1}^{p} RE_i \quad (6)$$

where p is the number of data points to be predicted.

6.3. End-point prediction

The end-point prediction is determined by assuming that x failures have been observed at the end of the testing time t_x and using the available failure data upto time t_e ($t_e \leq t_x$) to predict the number of failures at the time t_x. We apply different sizes of training patterns to train the ANN. In the trained ANN, we employ the end of the testing time t_x as input to predict the number of failures at the time t_x. The predicted value is compared with the actual value x. Predictive validity can be checked by *RE* for different values of t_e [4].

7. Performance analysis

The performances of the PFFNNDWCM and PRNNDWCM are compared with the ANN based dynamic weighted combinational model (DWCM) [7] and the neural network ensembles model (NNEM) [8] in the software reliability literature. We also compare the fitting performance and predictability of the proposed models with the software reliability models that can be developed by combining three or two of the selected four SRGMs. The number of possible combinations is ${}^4C_3 + {}^4C_2 = 10$. Here, for performance analysis, we only choose the following five combinations from the 10 combinations:

GYI, *YIL*, *GY*, *YI*, *IL*

There are FFNN and RNN based models possible for each of the above combinations. We represent FFNN and RNN based such

Please cite this article in press as: P. Roy, et al., Robust feedforward and recurrent neural network based dynamic weighted combination models for software reliability prediction, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.012

models as FFNNM and RNNM, respectively. All of these models are trained by the proposed GA.

DWCM has one hidden layer with three neurons. NNEM contains 19 component ANNs with median combination rule and each ANN is a three-layer single-input single-output FFNN with five hidden neurons. There is no activation function parameter for the hidden layer neurons of the DWCM and NNEM.

7.1. Model validation for DS1

Table 2 demonstrates the fitting performance of the different models under comparison in terms of *AE* for different normalized execution time (NET) values of *DS*1.

From Table 2, we observe that PFFNNDWCM has smallest *AE* value only at NET 0.8. So PFFNNDWCM has best fitting power in only that execution time. It means that PFFNNDWCM do not provide good fitting performance in *DS*1. Again, PRNNDWCM provides relatively better fitting performance than PFFNNDWCM as PRNNDWCM has lowest *AE* values for NETs 0.2, 0.6, 0.9 and 1.0. Hence, PRNNDWCM provides best fitting capability at the NETs 0.2, 0.6, 0.9 and 1.0. We can also see that DWCM has best fitting power in the NETs 0.1, 0.3, 0.4, 0.5 and 0.7. The fitting power of the PFFNNDWCM improves when the execution time increases from 0.1 to 0.8. Fitting capability of the PRNNDWCM increases with the growth of execution time from 0.4. It is very clear from Table 2 that the software reliability models that are developed by combining three of the four SRGMs, i.e. $RNNM_{GYI}$, $FFNNM_{GYI}$, $RNNM_{YIL}$ and $FFNNM_{YIL}$ have better fitting performance than the models that are developed by combining two of the four SRGMs, i.e. $RNNM_{GY}$, $FFNNM_{GY}$, $RNNM_{YI}$, $FFNNM_{YI}$, $RNNM_{IL}$ and $FFNNM_{IL}$. Hence, higher fitting performance is achieved by increasing the number of models to be combined. It is also noted that the fitting power of the RNN based model is always better than the FFNN based model for each of the combinations under consideration. The fitting performance of the PRNNDWCM and PFFNNDWCM are always much better than the $RNNM_{GYI}$, $FFNNM_{GYI}$, $RNNM_{YIL}$, $FFNNM_{YIL}$, $RNNM_{GY}$, $FFNNM_{GY}$, $RNNM_{YI}$, $FFNNM_{YI}$, $RNNM_{IL}$ and $FFNNM_{IL}$ for all NET values.

Table 3 shows the variable-term prediction in terms of *AE* and Table 4 shows the end-point prediction in terms of *RE* for different NET values that varies from 0.1 to 0.9.

Table 3 demonstrates that the proposed models have better predictability than the DWCM and NNEM. PFFNNDWCM has the smallest *AE* value at the NET 0.5. So PFFNNDWCM has the best predictive power at the NET 0.5. The *AE* values of the PRNNDWCM are smallest among the *AE* values of the models under comparison except at NET 0.5. Hence, it is obviously noted that PRNNDWCM has the overall better predictive capability than the PFFNNDWCM which proves the outstanding ability of the RNN in software reliability prediction. The predictive powers of the proposed models are improved with the growth of execution time from 0.3. From Table 3, it is also clear that the predictive capability of the $RNNM_{GYI}$, $FFNNM_{GYI}$, $RNNM_{YIL}$ and $FFNNM_{YIL}$ are better than the $RNNM_{GY}$, $FFNNM_{GY}$, $RNNM_{YI}$, $FFNNM_{YI}$, $RNNM_{IL}$ and $FFNNM_{IL}$. So higher predictability can be achieved by increasing the number of models to be combined. It is obvious that for all of the combinations under consideration, the RNN based model has always better predictive power than the FFNN based model. For all NETs, the PRNNDWCM and PFFNNDWCM demonstrate enhanced predictive accuracy than the models that can be developed by combining three or two of the four SRGMs.

From Table 4, we observe that the proposed models have much better end-point prediction capability than the DWCM and NNEM. PFFNNDWCM has the best end-point predictive power at the NET 0.2. PRNNDWCM has the overall better end-point prediction ability than the PFFNNDWCM as the *RE* values of the PRNNDWCM are smallest among the *RE* values of the other models under consideration except at NET 0.2. The end-point predictive powers of the proposed models are increased with the growth of execution time. Table 4 also proves that the proposed RNN has the best software reliability prediction capability than the FFNN. It is clearly visible that the models that are developed by combining three of the four models have better end-point predictive capability than the models that are developed by combining two of the four models. Hence, higher end-point predictability can be achieved by increasing the number of component models to be combined. It is also clear that the RNN based model has better end-point predictive power than the FFNN based model for all of the combinations under consideration. It is obvious that for all NETs, the PRNNDWCM and PFFNNDWCM have better end-point predictive capability than the $RNNM_{GYI}$, $FFNNM_{GYI}$, $RNNM_{YIL}$, $FFNNM_{YIL}$, $RNNM_{GY}$, $FFNNM_{GY}$, $RNNM_{YI}$, $FFNNM_{YI}$, $RNNM_{IL}$ and $FFNNM_{IL}$.

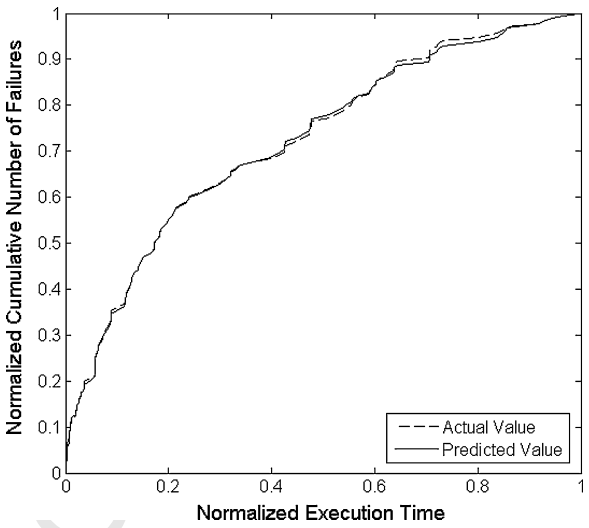

Fig. 3. Prediction curve of PRNNDWCM for *DS*1.

Fig. 3 shows the prediction curve of the PRNNDWCM which has the best predictive capability among the models under consideration for *DS*1.

From Fig. 3, we find that the PRNNDWCM has the excellent prediction ability compared to the observed values from the data set *DS*1.

The relative prediction error (*RPE*) curves of the different models under comparison are shown in Fig. 4. Here, we only compare the proposed models with the DWCM and NNEM as we have already seen from Tables 2–4 that the performances of the PRNNDWCM and PFFNNDWCM are always much better than the models that can be constructed by combining three or two of the selected four SRGMs.

From Fig. 4, we see that the proposed models have smaller *RPE*s than the DWCM and NNEM. It is also noted that the PRNNDWCM has much lesser *RPE* than the other models which again establishes the excellent software reliability predictive power of the proposed RNN.

7.2. Model validation for DS2

Now the PRNNDWCM and PFFNNDWCM are only compared with the DWCM and NNEM as it is obvious from the experimental results for *DS*1 that the proposed models have always superior fitting and predictive accuracy (for all NET values) than the software reliability models that can be developed by combining three or two of the selected four SRGMs.

Table 5 shows the fitting performance of the different software reliability models under comparison in terms of *AE* for different NET values of *DS*2.

Please cite this article in press as: P. Roy, et al., Robust feedforward and recurrent neural network based dynamic weighted combination models for software reliability prediction, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.012

Table 2
Q3 Comparison of fitting performance results for *DS*1.

Model	NET									
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
PRNNDWCM	5.5776	**2.1342**	2.7225	4.2255	3.2424	**2.1110**	2.1033	2.0300	**1.2534**	**1.1128**
PFFNNDWCM	8.7797	7.1735	5.8202	5.3108	3.2771	2.5720	2.1604	**1.2443**	1.4096	1.3312
DWCM	**3.5816**	3.0041	**2.3167**	**2.6459**	**2.2821**	3.2455	**1.9021**	1.7956	1.5037	1.6453
NNEM	65.0522	40.9960	34.5643	32.0253	28.4014	26.0588	23.3445	22.0056	21.1367	20.7896
$RNNM_{GYI}$	12.1464	8.2065	7.7503	9.9996	8.5725	8.0473	5.8695	5.4728	5.8621	5.2431
$FFNNM_{GYI}$	12.9512	9.6344	8.8697	10.0521	8.6243	9.0381	8.1934	6.0427	7.4602	5.9809
$RNNM_{YIL}$	11.6787	9.3328	8.8897	6.7947	9.0466	8.1380	8.1921	7.4822	7.1161	5.9436
$FFNNM_{YIL}$	12.0325	9.7186	11.9431	9.6813	9.1921	9.0623	8.7358	8.2084	8.0526	7.4586
$RNNM_{GY}$	19.1590	10.3520	14.3502	10.4281	10.2726	9.4545	9.4031	10.3025	8.8348	7.7932
$FFNNM_{GY}$	25.5181	11.4169	19.8922	10.4337	10.4497	10.0284	9.7521	13.9143	12.4502	9.4925
$RNNM_{YI}$	21.3797	14.6847	16.6009	12.0338	9.5251	10.6287	10.2973	13.9443	9.6897	8.0845
$FFNNM_{YI}$	31.3714	19.0491	18.7143	12.1408	9.8850	12.7744	16.8856	14.0859	12.1359	8.6374
$RNNM_{IL}$	14.2678	11.6417	13.5112	13.4395	10.8849	14.0632	12.1947	13.0562	11.9345	9.5206
$FFNNM_{IL}$	18.9679	13.2086	14.3028	13.6889	11.6052	15.0933	18.6781	14.0740	13.6135	10.2724

Table 3
Comparison of variable-term predictability for *DS*1.

Model	NET								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
PRNNDWCM	**1.9144**	**2.0188**	**2.1940**	**2.0038**	1.7728	**0.7294**	**0.5100**	**0.2823**	**0.2010**
PFFNNDWCM	3.3212	2.1905	2.3590	2.3162	**1.4560**	0.9735	0.5519	0.4378	0.3131
DWCM	31.1805	25.9846	19.1255	17.2881	13.1873	7.3045	3.4416	2.9249	1.7614
NNEM	24.1533	22.2280	20.9049	15.1608	8.9837	4.3946	1.5777	0.8479	1.3386
$RNNM_{GYI}$	8.0016	3.5261	2.5122	2.3483	1.8406	1.9649	1.0981	0.8084	0.4776
$FFNNM_{GYI}$	8.2103	4.5563	3.3776	2.5555	2.2445	2.1938	2.1792	0.8983	0.5377
$RNNM_{YIL}$	5.8694	2.9078	3.7543	2.8498	2.5644	1.2347	0.6856	0.6711	0.6163
$FFNNM_{YIL}$	6.7924	3.2045	4.3819	2.9854	2.7416	1.5221	1.0084	0.8282	0.7778
$RNNM_{GY}$	25.0405	6.9065	4.9319	3.7304	2.8814	2.7723	2.6104	1.3050	0.9169
$FFNNM_{GY}$	32.3466	8.2041	6.1765	3.8442	3.6086	2.8878	2.7872	1.3428	1.1415
$RNNM_{YI}$	17.3169	5.9523	7.2264	6.8516	3.8857	3.5879	2.1869	1.5134	0.9998
$FFNNM_{YI}$	18.8365	6.1093	9.4835	7.1780	3.9034	3.7798	2.3000	1.7333	1.2603
$RNNM_{IL}$	14.4560	8.4623	13.6215	4.8548	4.8161	3.9447	3.5591	1.7160	1.2195
$FFNNM_{IL}$	20.6097	11.9966	17.2859	5.9576	5.4996	4.1352	6.8183	2.1043	1.4584

Table 4
End-point prediction of software reliability for *DS*1.

Model	NET								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
PRNNDWCM	**0.2275**	0.1958	**0.0731**	**0.0316**	**0.0234**	**0.0163**	**0.0054**	**0.0030**	**0.0022**
PFFNNDWCM	0.2358	**0.0953**	0.0817	0.0633	0.0610	0.0455	0.0272	0.0212	0.0165
DWCM	48.0263	36.1886	30.9267	25.7216	19.5971	8.3143	6.1137	3.1769	1.5932
NNEM	34.9391	30.3986	20.1151	16.6727	9.6483	5.4012	2.0059	1.3250	1.4618
$RNNM_{GYI}$	3.9363	3.5290	2.1862	0.7617	0.2230	0.3983	0.3864	0.3565	0.1267
$FFNNM_{GYI}$	6.8821	5.4517	3.6088	2.3016	0.3709	0.4229	0.5736	0.6453	0.1605
$RNNM_{YIL}$	6.2929	0.4812	0.0868	2.7021	0.4352	0.5618	0.0967	0.4187	0.0306
$FFNNM_{YIL}$	6.7978	1.3338	1.5698	2.7487	0.5324	1.0539	0.5053	0.2772	0.1181
$RNNM_{GY}$	10.3031	9.6127	5.6254	4.1543	2.0928	2.7745	1.0866	0.8934	0.7624
$FFNNM_{GY}$	10.7199	11.6561	6.9043	4.5827	2.6775	2.8503	1.8889	1.7668	0.9730
$RNNM_{YI}$	14.9407	7.1085	11.4412	4.8045	1.5550	1.5931	2.0276	1.8158	1.4809
$FFNNM_{YI}$	16.3377	8.2594	11.6877	6.9619	1.7000	1.7876	3.1412	2.0835	1.6698
$RNNM_{IL}$	28.1815	18.7682	14.3435	10.0676	5.2256	4.2669	5.0749	3.4596	1.0028
$FFNNM_{IL}$	29.8990	19.6143	14.4609	21.2308	6.4048	11.1081	6.2983	5.2671	1.2388

Table 5
Comparison of fitting performance results for *DS*2.

NET	PRNNDWCM	PFFNNDWCM	DWCM	NNEM
0.1	**1.5555**	14.9605	18.2784	23.2674
0.2	**1.0327**	12.9713	14.6173	20.5464
0.3	**1.0305**	7.2449	9.9645	16.9635
0.4	**1.5711**	7.3575	7.6027	13.2474
0.5	**0.2779**	6.2920	6.4377	14.3375
0.6	**0.1873**	5.2373	6.5621	11.4631
0.7	**0.5540**	4.0945	5.6794	12.6297
0.8	**0.3689**	3.8838	5.2756	10.5903
0.9	**0.3969**	3.1937	4.7710	10.5048
1.0	**0.1069**	2.2358	4.3043	10.3753

From Table 5, we can observe that the proposed models have better fitting capacity than the DWCM and NNEM in *DS*2. Again, all *AE* values of the PRNNDWCM are smallest among the *AE* values of the different models under consideration. Hence, PRNNDWCM has the best fitting power in *DS*2. The fitting capability of PFFNNDWCM are improved with the growth of execution time from 0.4.

Table 6 shows the variable-term prediction in terms of *AE* and Table 7 shows the end-point prediction in terms of *RE* for different NET values of *DS*2.

Table 6 demonstrates that the proposed models have better predictive power than the DWCM and NNEM. The PRNNDWCM has the best software reliability prediction ability because of its minimum

Please cite this article in press as: P. Roy, et al., Robust feedforward and recurrent neural network based dynamic weighted combination models for software reliability prediction, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.012

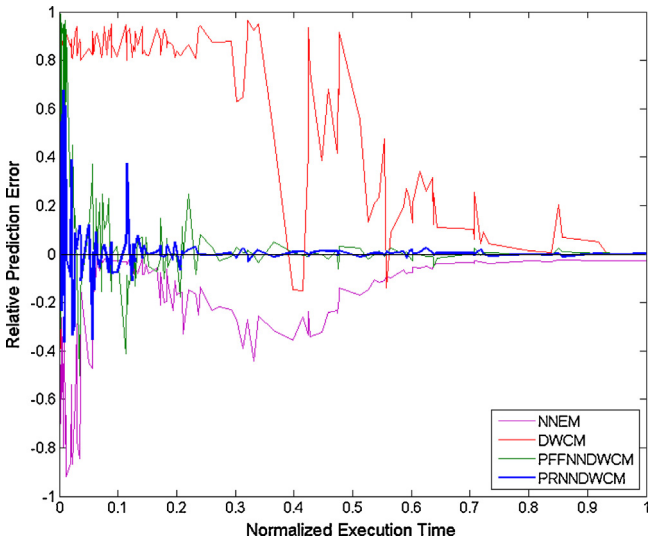

Fig. 4. *RPE* curves of different software reliability models for *DS*1.

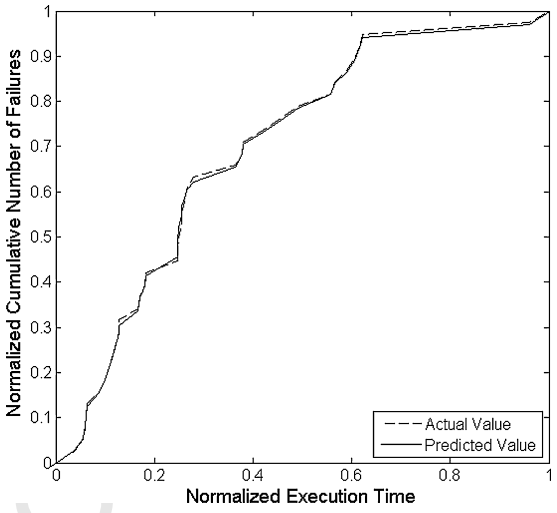

Fig. 5. Prediction curve of PRNNDWCM for *DS*2.

Table 6
Comparison of variable-term predictability for *DS*2.

NET	PRNNDWCM	PFFNNDWCM	DWCM	NNEM
0.1	**3.4073**	6.0156	10.8748	8.2058
0.2	**0.3691**	4.7483	7.2511	10.5845
0.3	**1.0075**	2.7003	3.6476	6.0227
0.4	**0.9890**	2.3971	9.5089	4.3772
0.5	**0.8897**	2.6346	6.7021	5.9463
0.6	**0.1051**	2.5704	5.5145	6.0819
0.7	**0.2220**	1.0521	3.6977	1.2171
0.8	**0.1445**	0.2019	1.0138	1.1987
0.9	**0.0108**	0.1087	0.2210	1.2242

AE values compared to the other software reliability models. Hence, it is also proved for *DS*2 that the proposed RNN has the best software reliability predictive power than the FFNN. The predictive power of the PFFNNDWCM is increased as the execution time is raised from 0.5. PRNNDWCM shows increasing prediction capability from the NET 0.7.

From Table 7, we observe that the proposed models have much better end-point predictive power than the DWCM and NNEM. PFFNNDWCM has the best end-point prediction capability at the NET 0.1. The *RE* values of the PRNNDWCM are smallest among the *RE* values of the models under comparison except at NET 0.1. Hence, it is easily seen that the PRNNDWCM has the overall best end-point predictive performance than the other models.

Fig. 5 shows the prediction curve of the PRNNDWCM for *DS*2.

From Fig. 5, we find that the PRNNDWCM has excellent prediction capability compared to the observed values from the data set *DS*2.

The *RPE* curves of the different models under consideration are shown in Fig. 6.

From Fig. 6, we see that the proposed models have smaller *RPE*s than the DWCM and NNEM. The PRNNDWCM has the smallest *RPE* than the other models which again establishes that the proposed RNN has the best software reliability prediction capability than the FFNN.

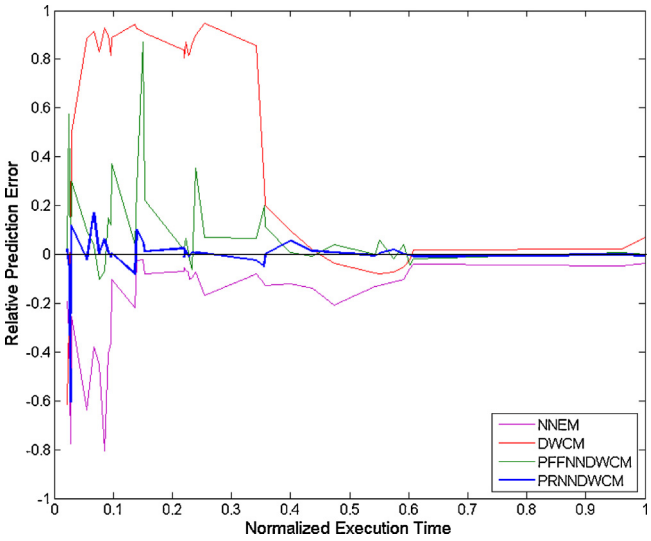

Fig. 6. *RPE* curves of different software reliability models for *DS*2.

Table 7
End-point prediction of software reliability for *DS*2.

NET	PRNNDWCM	PFFNNDWCM	DWCM	NNEM
0.1	0.7702	**0.5769**	10.8623	5.7650
0.2	**0.2409**	0.5761	6.1479	5.8947
0.3	**0.1352**	0.5906	4.6289	4.6501
0.4	**0.0534**	0.1793	12.5512	11.4849
0.5	**0.0440**	0.0744	7.1566	10.7432
0.6	**0.0498**	0.0721	7.1354	6.2275
0.7	**0.0091**	0.0202	4.2067	1.5212
0.8	**0.0192**	0.0294	0.3680	0.9588
0.9	**0.0064**	0.0107	0.2834	1.0578

Table 8
Comparison of fitting performance results for *DS*3.

NET	PRNNDWCM	PFFNNDWCM	DWCM	NNEM
0.1	**8.9606**	13.8850	25.7359	47.0375
0.2	**8.3661**	10.6617	24.9427	37.5333
0.3	**7.8941**	8.9816	22.1860	35.2792
0.4	**7.5021**	8.1000	23.8398	30.8718
0.5	**7.6505**	8.9565	21.2922	29.4203
0.6	**7.0797**	8.0227	20.0959	28.0453
0.7	**6.8224**	9.3592	18.1551	26.2632
0.8	**4.1726**	8.6939	19.8555	24.3311
0.9	**0.4883**	7.7570	16.8434	29.9379
1.0	**0.1792**	7.6700	15.2561	27.9877

Please cite this article in press as: P. Roy, et al., Robust feedforward and recurrent neural network based dynamic weighted combination models for software reliability prediction, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.012

Table 9
Comparison of variable-term predictability for *DS*3.

NET	PRNNDWCM	PFFNNDWCM	DWCM	NNEM
0.1	**7.0158**	10.2244	18.7082	18.5530
0.2	**2.1472**	2.6195	10.6521	17.4699
0.3	1.2199	**1.1148**	6.2989	15.4188
0.4	**1.0072**	1.1152	3.4242	2.9721
0.5	**0.9061**	1.0001	3.3188	2.5781
0.6	0.7550	**0.6037**	0.8505	2.3684
0.7	0.6617	**0.5608**	1.7783	2.0781
0.8	**0.2731**	0.5252	0.5333	1.5566
0.9	**0.1279**	0.1643	0.3957	1.3222

7.3. Model validation for DS3

Table 8 shows the fitting performance of the various software reliability models under comparison in terms of *AE* for different NET values of *DS*3.

From Table 8, we observe that the proposed models have better fitting accuracy than the DWCM and NNEM in *DS*3. All *AE* values of the PRNNDWCM are smallest among the *AE* values of the different models under consideration. Hence, the PRNNDWCM has the best fitting power in *DS*3 also. The fitting power of the PFFNNDWCM are improved with the growth of execution time from 0.7. PRNNDWCM shows increasing fitting accuracy with the growth of NET from 0.5.

Table 9 shows the variable-term predictability in terms of *AE* and Table 10 shows the end-point prediction of software reliability in terms of *RE* for different NET values of *DS*3.

Table 9 demonstrates that the proposed models have better software reliability prediction capability than the DWCM and NNEM. PFFNNDWCM has the smallest *AE* values in the NETs 0.3, 0.6 and 0.7. So PFFNNDWCM has the best predictive power at the NETs 0.3, 0.6 and 0.7 of *DS*3. The PRNNDWCM has the best predictive power at the NETs 0.1, 0.2, 0.4, 0.5, 0.8 and 0.9 because of its lowest *AE* values in that execution times. We can find that the PRNNDWCM has the overall better predictive capability than the PFFNNDWCM. The predictive power of the PFFNNDWCM is improved with the growth of NET from 0.4. PRNNDWCM shows increasing prediction accuracy with the rising execution time.

From Table 10, we observe that the proposed models have much better end-point prediction capability than the DWCM and NNEM. PFFNNDWCM has the best end-point predictive power at the NETs 0.4 and 0.8. PRNNDWCM has the overall better end-point prediction ability than the PFFNNDWCM as the *RE* values of the PRNNDWCM are smallest among the *RE* values of the models under consideration except at NETs 0.4 and 0.8. The end-point predictive power of the PFFNNDWCM is increased with the growth of execution time.

Fig. 7 shows the prediction curve of the PRNNDWCM which has the best software reliability prediction capability among the models under consideration for *DS*3.

From Fig. 7, we conclude that the PRNNDWCM has the outstanding prediction ability compared to the observed failure data from *DS*3.

Table 10
End-point prediction of software reliability for *DS*3.

NET	PRNNDWCM	PFFNNDWCM	DWCM	NNEM
0.1	**0.2866**	2.4396	22.0868	15.4520
0.2	**0.2515**	0.8627	20.9103	14.7059
0.3	**0.2162**	0.2452	11.0076	11.7712
0.4	0.2089	**0.1581**	8.5497	4.7663
0.5	**0.1054**	0.1471	9.0611	3.4339
0.6	**0.0129**	0.0806	5.8323	4.5190
0.7	**0.0167**	0.0481	2.4122	2.6376
0.8	0.1710	**0.0184**	2.9903	2.6065
0.9	**0.0020**	0.0111	0.3239	2.1024

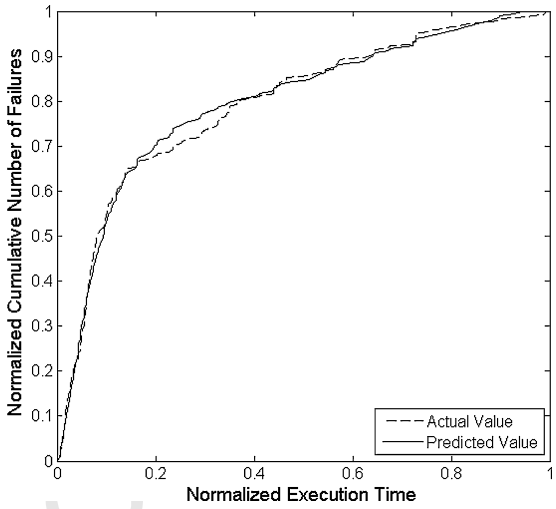

Fig. 7. Prediction curve of PRNNDWCM for *DS*3.

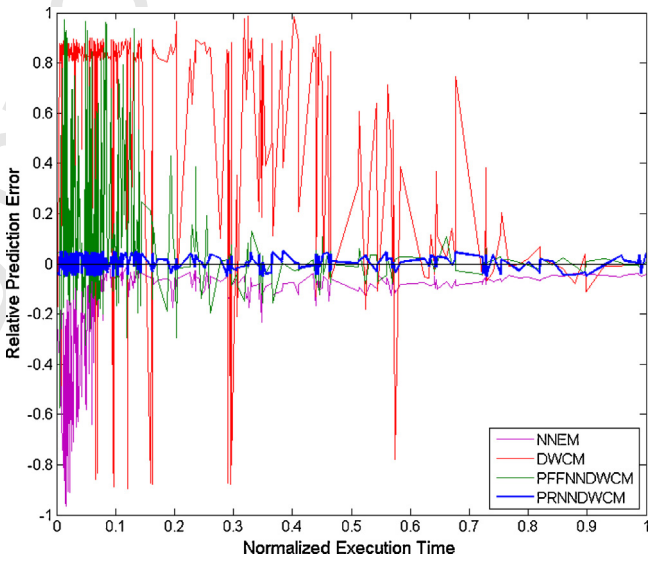

Fig. 8. *RPE* curves of different software reliability models for *DS*3.

The *RPE* curves of the different models under comparison are shown in Fig. 8.

From Fig. 8, we conclude that the proposed models have lower *RPE*s than the DWCM and NNEM. We also conclude that the PRNNDWCM has much lower *RPE* than the other models which conforms the fact that the proposed RNN can be a better software reliability predictor than the FFNN.

8. Conclusion

In this paper, we have developed robust feedforward and recurrent neural network based dynamic weighted combination models to improve the software reliability prediction accuracy. Four traditional software reliability growth models have been combined based on the dynamically evaluated weights determined by the learning algorithm of the proposed feedforward and recurrent neural networks. We construct the recurrent neural network architecture based on the proposed feedforward neural network to predict software reliability more precisely by learning the dynamic temporal patterns of the failure data. We propose a real-coded genetic algorithm based learning algorithm to train the proposed artificial neural networks using the software failure data. The

Please cite this article in press as: P. Roy, et al., Robust feedforward and recurrent neural network based dynamic weighted combination models for software reliability prediction, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.012

experimental results from the applications to three real software failure data sets demonstrate that the proposed feedforward and recurrent neural network based dynamic weighted combination models have better software reliability predictive quality than the other artificial neural network based software reliability models. Proposed recurrent neural network based dynamic weighted combination model achieves significantly lower prediction error relative to the proposed feedforward neural network based dynamic weighted combination model, which establishes that the proposed recurrent neural network architecture has the best predictive power and is optimistic for software reliability prediction.

Acknowledgements

The authors are heartily thankful to the editor-in-chief, Prof. R. Roy and reviewers for their detailed and constructive valuable comments that help us to improve the quality of the paper. This research work is supported by the Council of Scientific and Industrial Research of India under the research Project No. 25(0191)/10/EMR-II.

References

[1] M.R. Lyu, Handbook of Software Reliability Engineering, McGraw-Hill, 1996.
[2] S. Haykin, Neural Networks and Learning Machines, Prentice Hall, 2012.
[3] S. Rajasekaran, G.A.V. Pai, Neural Networks, Fuzzy Logic, and Genetic Algorithms Synthesis and Applications, Prentice Hall, 2011.
[4] J.D. Musa, Software Reliability Engineering: More Reliable Software, Faster Development and Testing, McGraw-Hill, 2004.
[5] J.D. Musa, A. Iannino, K. Okumoto, Software Reliability Measurement, Prediction and Application, McGraw-Hill, 1987.
[6] N. Karunanithi, D. Whitley, Y.K. Malaiya, Prediction of software reliability using connectionist models, IEEE Trans. Softw. Eng. 18 (1992) 563–574.
[7] Y.S. Su, C.Y. Huang, Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models, J. Syst. Softw. 80 (2007) 606–615.
[8] J. Zheng, Predicting software reliability with neural network ensembles, Expert Syst. Appl. 36 (2009) 2116–2122.
[9] A.L. Goel, K. Okumoto, Time-dependent error-detection rate model for software reliability and other performance measures, IEEE Trans. Reliab. 28 (3) (1979) 206–211.
[10] M. Xie, Software Reliability Modeling, World Scientific, 1991.
[11] H. Pham, System Software Reliability, Springer, 2006.
[12] C.Y. Huang, S.Y. Kuo, Analysis of incorporating logistic testing effort function into software reliability modeling, IEEE Trans. Reliab. 51 (3) (2002) 261–270.
[13] C.Y. Huang, M.R. Lyu, S.Y. Kuo, A unified scheme of some nonhomogeneous Poisson process models for software reliability estimation, IEEE Trans. Softw. Eng. 29 (3) (2003) 261–269.
[14] M. Ohba, Inflection S-Shaped Software Reliability Growth Models, Stochastic Models in Reliability Theory, Springer, 1984, pp. 44–162.
[15] P. Roy, G.S. Mahapatra, K.N. Dey, An S-shaped software reliability model with imperfect debugging and improved testing learning process, Int. J. Reliab. Saf. 7 (4) (2013) 372–387.
[16] S. Yamada, M. Ohba, S. Osaki, S-shaped software reliability growth models and their applications, IEEE Trans. Reliab. R-33 (4) (1984) 289–292.
[17] Y.K. Malaiya, M.N. Li, J.M. Bieman, R. Karcich, Software reliability growth with test coverage, IEEE Trans. Reliab. 51 (2002) 420–426.
[18] H. Pham, L. Nordmann, X.M. Zhang, A general imperfect software-debugging model with s-shaped fault detection rate, IEEE Trans. Reliab. 48 (2) (1999) 169–175.
[19] S.M. Li, Q. Yin, P. Guo, M.R. Lyu, A hierarchical mixture model for software reliability prediction, Appl. Math. Comput. 185 (2007) 1120–1130.
[20] H. Li, M. Zeng, M. Lu, X. Hu, Z. Li, Adaboosting-based dynamic weighted combination of software reliability growth models, Qual. Reliab. Eng. Int. 28 (1) (2012) 67–84.
[21] W. Wu, K. Han, C. He, S. Wu, A dynamically-weighted software reliability combination model, in: International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (ICQR2MSE), 2012, pp. 148–151.
[22] N. Karunanithi, Y.K. Malaiya, Neural networks for software reliability engineering, in: Handbook of Software Reliability Engineering, McGraw-Hill, 1996, pp. 699–728.
[23] R. Sitte, Comparison of software-reliability-growth predictions: neural networks vs parametric recalibration, IEEE Trans. Reliab. 48 (3) (1999) 285–291.
[24] T.M. Khoshgoftaar, R.M. Szabo, Predicting software quality, during testing, using neural network models: a comparative study, Int. J. Reliab. Qual. Saf. Eng. 1 (1994) 303–319.
[25] T.M. Khoshgoftaar, R.M. Szabo, Using neural networks to predict software faults during testing, IEEE Trans. Reliab. 45 (3) (1996) 456–462.
[26] K.Y. Cai, L. Cai, W.D. Wang, Z.Y. Yu, D. Zhang, On the neural network approach in software reliability modeling, J. Syst. Softw. 58 (2001) 47–62.
[27] S.L. Ho, M. Xie, T.N. Goh, A study of the connectionist models for software reliability prediction, Comput. Math. Appl. 46 (2003) 1037–1045.
[28] L. Tian, A. Noore, On-line prediction of software reliability using an evolutionary connectionist model, J. Syst. Softw. 77 (2005) 173–180.
[29] L. Tian, A. Noore, Evolutionary neural network modeling for software cumulative failure time prediction, Reliab. Eng. Syst. Saf. 87 (2005) 45–51.
[30] Q.P. Hu, M. Xie, S.H. Ng, G. Levitin, Robust recurrent neural network modeling for software fault detection and correction prediction, Reliab. Eng. Syst. Saf. 92 (2007) 332–340.
[31] N.R. Kiran, V. Ravi, Software reliability prediction by soft computing techniques, J. Syst. Softw. 81 (4) (2008) 576–583.
[32] P.K. Kapur, S.K. Khatri, M. Basirzadeh, Software reliability assessment using artificial neural network based flexible model incorporating faults of different complexity, Int. J. Reliab. Qual. Saf. Eng. 15 (2) (2008) 113–127.
[33] P.K. Kapur, V.S.S. Yadavalli, S.K. Khatri, M. Basirzadeh, Enhancing software reliability of a complex software system architecture using artificial neural-networks ensemble, Int. J. Reliab. Qual. Saf. Eng. 18 (3) (2011) 271–284.
[34] R. Mohanty, V. Ravi, M.R. Patra, Hybrid intelligent systems for predicting software reliability, Appl. Soft Comput. 13 (1) (2013) 189–200.

Please cite this article in press as: P. Roy, et al., Robust feedforward and recurrent neural network based dynamic weighted combination models for software reliability prediction, Appl. Soft Comput. J. (2014), http://dx.doi.org/10.1016/j.asoc.2014.04.012