IETI Transactions on Computers, 2015, Volume 1, Issue 1, 11-21.
http://www.ieti.net/tc
An International Open Access Journal

The Software Reliability Model Using Hybrid Model of Fractals and BP Neural Network

Yong Cao[1, a], Xiaoguang Yue[2], Fei Xiong[3] and Youjie Zhao[4]

[1] School of Computer and Information, Southwest Forestry University, Kunming, Yunnan, China

[2]Wuhan University, China

[3] School of Computer and Information, Southwest Forestry University, Kunming, Yunnan, China

[4] School of Computer and Information, Southwest Forestry University, Kunming, Yunnan, China

[a]cn2caoyong@gmail.com

Keywords: Software Reliability Model, Fractals, BP Neural Network and Power Law

Abstract. The software reliability is the ability of the software to perform its required function under stated conditions for a stated period of time. In this paper, a hybrid methodology that combines both BP neural network and fractal models is proposed to take advantage of unique strength of BP neural network and fractal in modeling. Based on the experiments performed on the software reliability data obtained from literatures, it is observed that our method is effective through comparison with past methods and a new idea for the research of the software failure mechanism is presented.

Introduction

Software reliability, namely the capability that a given component or system within a specified environment will operate correctly for a specified period of time, has been one of the most important qualities [1, 2, 3]. In general, the probability of correct operation is inversely related to the length of time specified; the longer a system operates, the greater the chance of failure. The software reliability model is used not only to estimate reliability, but also to measure and control the

software test. The important problem of the software reliability model is to calculate and predict the next failure time in advance [2].

The term fractal, which means broken or irregular fragments are mathematical or natural objects that are made of parts similar to the whole in certain ways. It belongs to geometrical category. If time series also follow the laws of fractal geometry we can use fractal to analyze time series. According to [5] self-similarity exists in time series of software failure. Cao and Zhu [5] have applied fractal to foresting software failures and provided software prediction model based on fractals. Please see [5] for a detailed exposition.

Some forecasting techniques have been developed, each one with its particular advantages and disadvantages compared to other approaches. This motivates the study of hybrid model combining different techniques and their respective strengths. Different forecasting models can complement each other in capturing patterns of data sets, and both theatrical and empirical studies have concluded that a combination of forecast outperforms individual forecasting models [3, 4, 7]. Terui and Dijk [4] presented a linear and nonlinear time series model for forecasting the US monthly employment rate and production indices. Their results demonstrated that the combined forecasts outperformed the individual forecasts. Xiao and Tadashi [8] apply the wavelet-based techniques to estimate software intensity functions in non-homogeneous Poisson process based software reliability models. They show that their wavelet-based estimation method can provide higher goodness-of-fit performances than the conventional maximum likelihood estimation and the least squares estimation in some cases.

The outline of this paper is as following: Section 2 presents the fractal software reliability model Using Hybrid Model of Fractals and BP Neural Network; Section 3 validates the model through analyzing the empirical failure data; Section 4 concludes this paper and describes the future research.

Software Reliability Hybrid Model of Fractals and BP Neural Network

A neural network model takes an input vector X and produces output vector Y. The relationship between X and Y is determined by the network architecture. The neural network generally consists of at least three layers: one input layer, one output layer, and one or more hidden layers. It is widely accepted that a three-layer back propagation neural network with an identity transfer function in the output unit and logistic functions in the middle-layer units can approximate well any continuous function arbitrarily, given a sufficient amount of middle-layer units.

The back-propagation algorithm consists of two phases. Suppose we have s samples. Each is described by

$$X_i = (x_{i1}, x_{i2}, \ldots, x_{im}) \tag{1}$$

$$T_i = (t_{i1}, t_{i2}, \ldots, t_{in}) \tag{2}$$

Where X_i is an input vector, T_i is a target output vector and $1 \le i \le s$.

In the first phase (forward-propagation), X_i is fed into the input layer, and an output $Y_i = (y_{i1}, y_{i2}, \ldots, y_{in})$ is generated based on the current weight vector W. The objective is to minimize an error function E, which is defined as

$$E = \sum_{i=1}^{s} \sum_{j=1}^{n} \frac{(y_{ij} - t_{ij})^2}{2} \tag{3}$$

In the second phase (back-propagation), a gradient descent in the weight space, W, is performed to locate the optimal solution. The direction and magnitude change Δw_{ij} can be computed as

$$\Delta w_{ij} = \frac{\partial E}{\partial w_{ij}} \varepsilon \tag{4}$$

Because software reliability prediction has only one dependent variable and no explanatory variable in strict sense. If we have a time series, we followed the general time series predicting model in this paper, while is represented in the following form:

$$t= \{t_1, t_2, \quad \dots , t_N\},$$

where, failure time, of ith times, of software systems is t_i, and t_0=0. So, failure space time is $T_i=t_i-t_{i-1}$, $i \leqslant N$, and, N is maximum observation time domain. Thus, t and T are random sequence.

We focus on value of random sequence t, since it reflects evolving regularity of failure time of software systems. When a software system is tested, it is modified immediately whenever an error is found in software. Because software is changing irregularly, the sequence t is a non-stationary and nonlinear random sequence.

We applied software reliability hybrid Model of fractals and BP neural network to software failure time series. Suppose we model time series with E_t, and it can be represented as follow:

$$\ln(E_t) = \ln(\hat{F}_t) + \varepsilon_t \tag{5}$$

where $\hat{F}_t$ is the forecasting value and estimate it through fractal. ε_t is residual and it contains noise. Therefore, we can use wavelet to shrink noise and reconstruct signal to make our prediction more accurate . Therefore, the combined forecast is

$$\hat{E}_t = (\hat{F}_t)\exp(\hat{\varepsilon}_t) \tag{6}$$

where $\hat{\varepsilon}_t$ is the forecasting value of ε_t and $\hat{E}_t$ is the forecasting value of E_t.

Algorithm 1:

Begin

Initialization: suppose the size of slide window m, k=1 and A is a array of the number of failure corresponding failure time;

Repeat for $i=k$ *to* $m+k$-1 {

$B(i)$=log($A(i)$);/*the logarithm of practical failure time in the slide window.*/

$C(i)$=log(i);/*the logarithm of failure number in the slide window.*/

}

(1) According to eq.(5) of literature [5] and method of linear regression, compute the slope of linear regression in the slide window $b=d=1/fractal\ dimension$ and constant $a=\log(s)=-d\log(C)$;

(2) Using the above a and b and equation $c=b*\ln(C(i))+a$ compute c of each point in sliding window;

(3) Compute difference of c and actual failure time of each point to produce a difference series in sliding window;

(4) According to BP neural network method to compute the difference series;

(5) According to eq.(8), eq.(9) and linear regression method predict next failure time;

(6) Add the practical failure time of the next point to A;

k++; /*the slide window move backwards.*/

Until test over

End

Experiments

The forecasting algorithm and one-step-ahead forecasting policy are applied in Musa's data set 1 and 2 [6] (Table 1 and Table 2). The performance of the proposed model is compared with fractal model [5], adaptive Kalman filter [5], and ARIMA [5] forecasting methods. The experimental results are shown in Fig.1, Fig.2, Fig.3, Fig.4 and Table 3. In Fig.2 60% of the forecasting errors using the fractal prediction model based on wavelet are less than 2% and in Fig.4 50% of the forecasting errors using the fractal prediction model based on wavelet are less than 6%. Obviously our method is effective. In the investigation, the values of Mean Absolute Error MAE =

$\frac{1}{n}\sum_{i=1}^{n} abs\frac{T_i - \overline{T}_i}{T_i}$ and Normal Root Mean Square Error NRMSE = $\sqrt{\frac{\sum_{i=1}^{n}(T_i - \overline{T}_i)^2}{\sum_{i=1}^{n} T_i^2}}$, where T_i is the *i*th actual failure time and $\overline{T}_i$ is prediction time (Table 3).

Table 1. The Musa's data set 1 of software failure time series, and from left to right the time in each cell denotes the cumulate time of the *i*th failure, *i*=1, 2, …… .Unit: second

3	33	146	227	342	351	353	444	556	571
709	759	836	860	968	1056	1726	1846	1872	1986
2311	2366	2608	2676	3098	3278	3288	4434	5034	5049
5085	5089	5089	5097	5324	5389	5565	5623	6080	6380
6477	6740	7192	7447	7644	7837	7843	7922	8738	10089
10237	10258	10491	10625	10982	11175	11411	11442	11811	12559
12559	12791	13121	13486	14708	15251	15261	15277	15806	16185
16229	16358	17168	17458	17758	18287	18568	18728	19556	20567
21012	21308	23063	24127	25910	26770	27753	28460	28493	29361
30085	32408	35338	36799	37642	37654	37915	39715	40580	42015
42045	42188	42296	42296	45406	46653	47596	48296	49171	49416
50145	52042	52489	52875	53321	53443	54433	55381	56463	56485
56560	57042	62551	62651	62661	63732	64103	64893	71043	74364
75409	76057	81542	82702	84566	88682				

Table 2. The Musa's data set 2 of software failure time series, and from left to right the time in each cell denotes the interval between the (i-1)th failure and the ith failure, i=1, 2, …… . Unit: second

320	1439	9000	2880	5700
21800	26800	113540	112137	660
2700	28493	2173	7263	10865
4230	8460	14805	11844	5361
6553	6499	3124	51323	17010
1890	5400	62313	24826	26335
363	13989	15058	32377	41632
4160	82040	13189	3426	5833
640	640	2880	110	22080
60654	52163	12546	784	10193
7841	31365	24313	298890	1280
22099	19150	2611	39170	55794
42632	267600	87074	149606	14400
34560	39600	334395	296015	177395
214622	156400	166800	10800	267000

Table 3. Prediction results of different models of Musa's data set 1 and 2. Ak stands for adaptive Kalman filter, FW stands for hybrid model of fractals and BP Neural Network.

	Error	FB	Fractal	ARIMA	AK
Musa 1	MAE	0.0250	0.0271	0.0432	0.0425
	NRMSE	0.0248	0.0312	0.0493	0.0481
Musa 2	MAE	0.0550	0.0574	0.0718	0.0635
	NRMSE	0.0509	0.0645	0.0824	0.0702

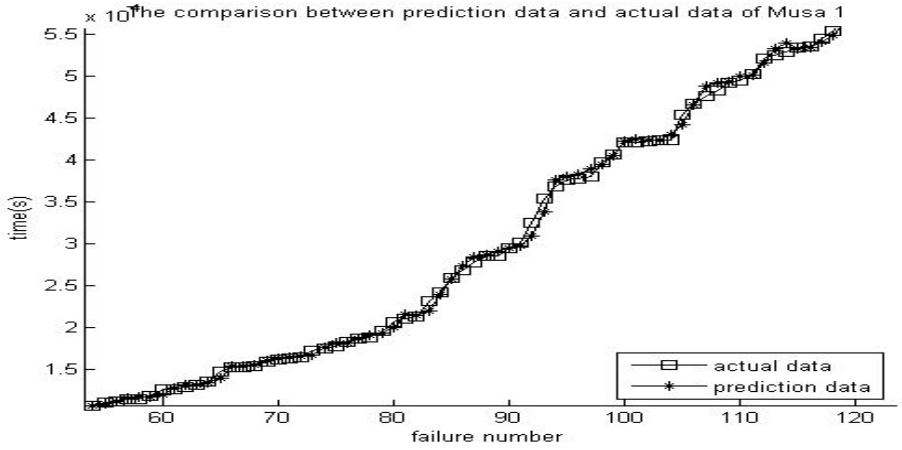

Fig.1. The comparison between prediction data and actual data of Musa's data set 1 (sliding window size m=16).

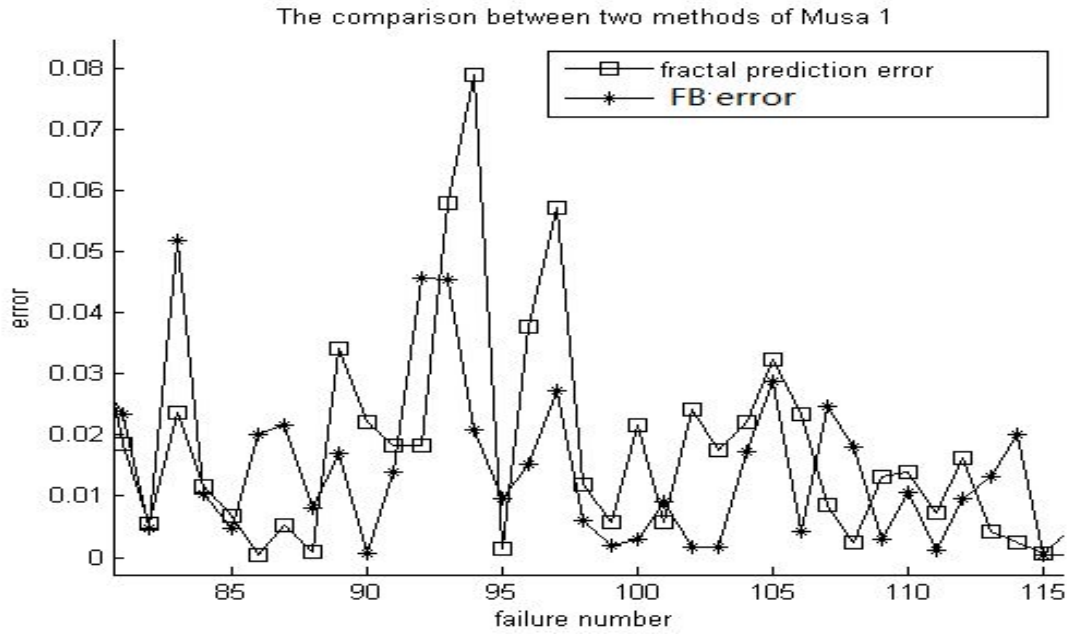

Fig.2. The comparison of relative error between fractal prediction and Hybrid Model of Fractals and BP Neural Network of Musa's data set 1 (sliding window size m=16).

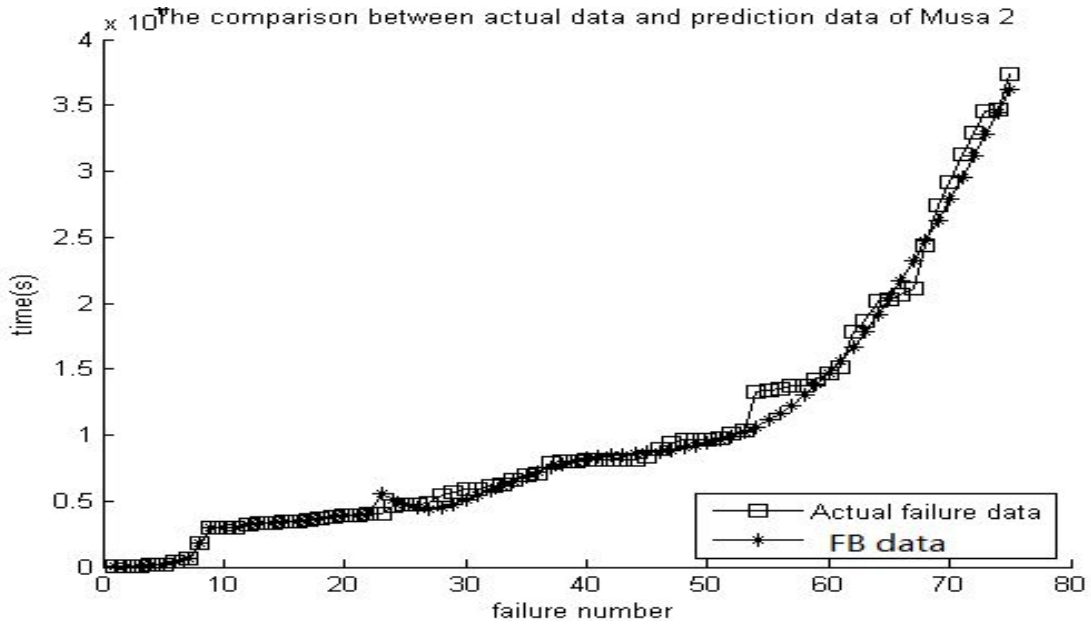

Fig.3. The comparison between prediction data and actual data of Musa's data set 2 (sliding window size m=15).

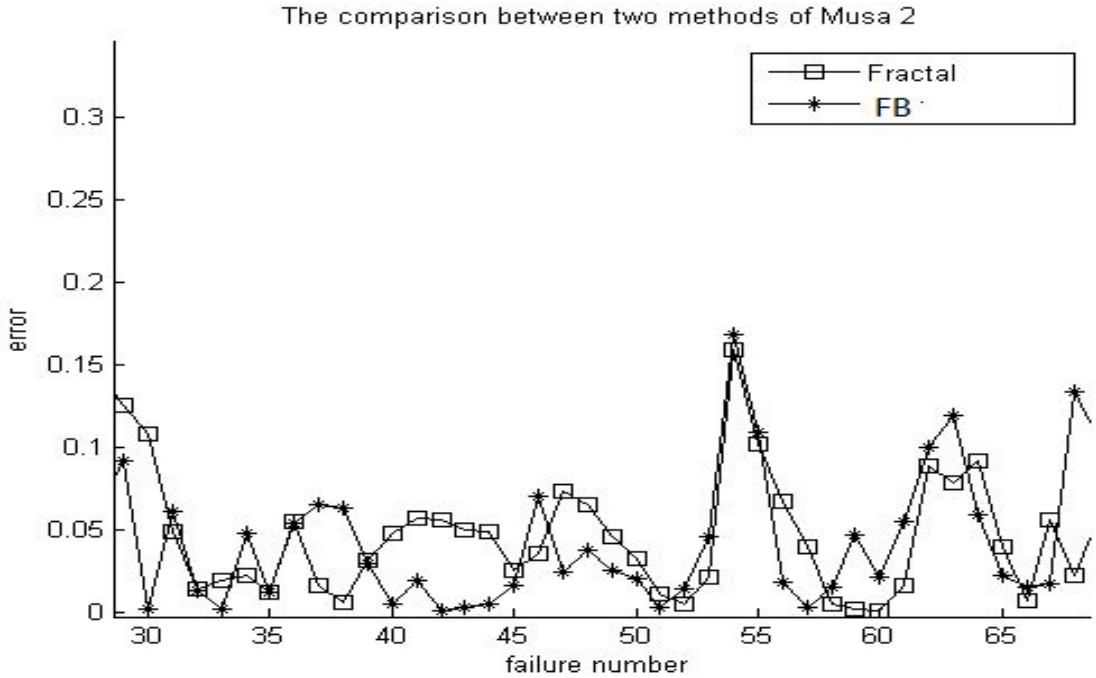

Fig.4. The comparison of relative error between fractal prediction and f hybrid model of fractals and BP neural network of Musa's data set 2 (sliding window size m=15).

Conclusion

Reliability is one of the most important software qualities, and failure analysis is an important part of the research of software reliability. The important problem of the software reliability model is to calculate and predict the next failure time in advance. This paper analyzes the empirical failure data and proposes the Hybrid Model of Fractals and BP Neural Network to predict the next software failure time which almost fit the practical failure time. Studying the empirical data (Musa's failure data set 1and 2) and comparison with the classical models validate the proposed model. A new idea for the research of the software failure mechanism is provided. In the future, some other factors which affect the software reliability can be considered in the model to predict software reliability to improve forecasting accuracy. We will also research the mechanism behind fractals further and draw a clear conclusion.

References

[1] Huo_wang Chen, Ji Wang and Wei Dong. High Confidence Software Engineering Technologies. J.ACTA ELECTRONICA SINICA, 12(2003)

[2] Scott Dick, Cindy L. Bethel, and Abraham Kandel. Software-Reliability Modeling: The Case for Deterministic Behavior. J. IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—PART A: SYSTEMS AND HUMANS, VOL. 37, NO. 1(2007)

[3] E. E. Lewis, Introduction to Reliability Engineering, 2nd ed. New York: Wiley(1996)

[4] Su, Y. S., Huang, C. Y. Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models. Journal of Systems and Software., vol.80, no.4, pp.606-615, April 2006.

[5] Yong Cao, Qingxin Zhu. The software reliability model based on fractals. IEICE Trans. Inf. & Syst., E93-D, 2**(2010)**

[6] Musa J.D. Software Reliability Data, technical reportavailable from Data Analysis Center for Software. Rome Air Development Center, New York(1979)

[7] Nobuhiko Terui, Dijk Herman K. Combined forecasts from linear and nonlinear time series models. International Journal of Forecasting 18,pp.421-38, 2002

[8] Xiao XIAO and Tadashi DOHI. Wavelet-based Approach for Estimating Software Reliability. 20th International Symposium on Software Reliability Engineering **(2009)** November 16-19; Higashi-Hiroshima, Japan