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

Yong Cao¹, Xiaoguang Yue², Fei Xiong³ and Youjie Zhao⁴

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

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

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

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

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

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

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

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 $\Delta w_{ij}$ can be computed as

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

\[ t = \{t_1, t_2, \ldots, t_n\}, \]

where, failure time, of ith times, of software systems is \( t_i \), and \( t_0=0 \). So, failure space time is \( T_i = t_i - t_{i-1}, i \leq N \), 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:

\[ \ln(E_t) = \ln(\hat{F}_t) + \epsilon_t \]  

(5)

where \( \hat{F}_t \) is the forecasting value and estimate it through fractal. \( \epsilon_t \) 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

\[ \hat{E}_t = (\hat{F}_t)\exp(\hat{\epsilon}_t) \]  

(6)

where \( \hat{\epsilon}_t \) is the forecasting value of \( \epsilon_t \) 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 \( m+k-1 \) {

\[ B(i)=\log(A(i)); /*\text{the logarithm of practical failure time in the slide window.} */ \]

\[ C(i)=\log(i); /*\text{the logarithm of failure number in the slide window.} */ \]

}
According to eq.(5) of literature [5] and method of linear regression, compute the slope of linear regression in the slide window $b = d = 1 / \text{fractal dimension}$ 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 =
\[ \frac{1}{n} \sum_{i=1}^{n} \frac{\text{abs} \left( T_i - \overline{T}_i \right)}{\overline{T}_i} \]

and Normal Root Mean Square Error NRMSE = \[ \sqrt{\frac{\sum_{i=1}^{n} (T_i - \overline{T})^2}{\sum_{i=1}^{n} T_i^2}} \]

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, \( i = 1, 2, \ldots \). Unit: second

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

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 \(i\)th failure, \(i = 1, 2, \ldots \). Unit: second
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.

<table>
<thead>
<tr>
<th>Error</th>
<th>FB</th>
<th>Fractal</th>
<th>ARIMA</th>
<th>AK</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAE</td>
<td>0.0250</td>
<td>0.0271</td>
<td>0.0432</td>
<td>0.0425</td>
</tr>
<tr>
<td>NRMSE</td>
<td>0.0248</td>
<td>0.0312</td>
<td>0.0493</td>
<td>0.0481</td>
</tr>
<tr>
<td>MAE</td>
<td>0.0550</td>
<td>0.0574</td>
<td>0.0718</td>
<td>0.0635</td>
</tr>
<tr>
<td>NRMSE</td>
<td>0.0509</td>
<td>0.0645</td>
<td>0.0824</td>
<td>0.0702</td>
</tr>
</tbody>
</table>
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 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 1 and 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


