Software Fault Prediction Based on Grey Neural Network

Peng Zhang
The Engineering Institute
Air Force Engineering University
Xi'an, ShaanXi, P. R. China

Yu-tong Chang
Representative Office
AVIC XI'AN AERO-Engine(Group) LTD.
Xi'an, ShaanXi, P. R. China

***Abstract*—Considering determining the number of software fault is an uncertain non-linear problem with only small sample, a novel software fault prediction method based on grey neural network is put forward. Firstly, constructing the grey neural network topological structure according the small sample sequence is necessary, and then the network learning algorithm is discussed. Finally, the grey neural network prediction model based on the grey theory and artificial neural network is proposed. The sample fault sequences of some software project are used to verify the precision of this method. Comparison with GM(1,1), the proposed model can reduce the prediction relative error effectively.**

Keywords- Software; Fault Prediction; Grey Neural Network

I. INTRODUCTION

The purpose of software fault prediction is trying to determine the number of fault according to the elementary software testing, which could provide a significative guidance for the further maintenance of software system as in [1]. For example, it is convenient for the software managers to do a reasonable schedule plan according to the fault number during the various lifecycles. Furthermore, it is necessary to make an exact appropriation budget and focus on the core modules whose fault numbers are in existence.

Because the import and appear of software fault are correlate with several factors, determining the number of software fault is an uncertain non-linear problem. So it is difficult to establish an exact mathematical model for efficient solutions. On the other hand, the given software project's characters are different from the other one, the historical information for prediction method was limited, that is to say, determining the number of software fault is also a small sample problem.

In this paper we proposed a novel software fault prediction method based on grey neural network. The rest is organized as follows. In Section 2 we introduce the basic concepts of the software fault prediction. In Section 3 the model of the grey neural network for the prediction is been proposed and the given workflow-diagram could be used as a practical guidance. In Section 4 and 5 the experimental evaluation and further work are discussed.

II. SOFTWARE FAULT PREDICTION AND GREY PREDICTION

In the process of software maintenance, the main task of the software sustaining plan is trying to locate the faults and repair them as soon as possible. So it would be very helpful if the software engineers can obtain the numbers in the next software test and diagnosis by anticipation as in [2]. Some scholars have studied this problem based on the technique of quantization, such as: case-based reasoning as in [3], bayesian method as in [4], linear programming methodologies as in [5] etc. At present, the most classification and prediction algorithms in KDD have been widely used in this area. However, in the face of small sample, uncertain, non-linear problems of software fault prediction, mentioned in Section 1, so it is difficult to establish an exact mathematical quantization model for efficient solutions as in [6].

Grey System Theory is a novel method for small sample, less information and uncertain data. Grey Prediction can do a quantization analysis of intending status for the given system based on original data pre-processing and Grey Model establishment. For the given original time series $X=(x_1,x_2,\cdots,x_n)$, the original data should be pre-processed by the accumulated generating operation as follows:

$$X^{'}=(X_t^{'},t=1,2,\cdots,n)=(x_1,\sum_{t=1}^{1}x_t,\sum_{t=1}^{2}x_t,\cdots,\sum_{t=1}^{n}x_t) \tag{1}$$

According to the new time series $X^{'}$, the differential equation of GM(1,1) could be established as:

$$\frac{dX^{'}}{dt}+aX^{'}=u \tag{2}$$

So the solution of the above equation is:

$$X^{*'}=(x_1-u/a)e^{-a(t-1)}+u/a \tag{3}$$

$X^{*'}$ is the estimating value of series $X^{'}$, and prediction value X^{*} of series X can be obtained by the inverse accumulated generating operation:

$$X^{*}=X_t^{*'}-X_{t-1}^{*'} \qquad t=2,3,\cdots \tag{4}$$

Chinese scholar Cheng-Ying Mao took advantage of GM(1,1) to propose a new software fault prediction method. The Experimental result showed that this model can achieve a satisfying prediction precision even if less abundant data however. Actually the software fault numbers of the different editions and of the different lifecycle are unstable, these fault series fluctuations are frequently happened. In this situation,

Project supported by the National Science Foundation for Post-doctoral Scientists of China (Grant No. 201150M1551)

the average relative error of Grey Model is perhaps larger which noted in his researches.

Now the study on the improvement of Grey Prediction are focused upon two types: one is error reduction technique, and the other is error compensation technique. The Grey Neural Network proposed recently is integration model between GM and ANN(Artificial Neural Network) which can improve the prediction precision further.

III. Grey Neural Network Model for Software Prediction

A. *Grey Neural Network Topological Structure*

In an effort to avoid confusion and convenient expression, the original sequence X is also noted as $x(t)$, the accumulated sequence X' is also noted as $y(t)$, and the prediction sequence value noted as $z(t)$.

The differential equation of the Grey Neural Network(GNN) with n-parameters is as follows:

$$\frac{dy_1}{dt}+ay_1=b_1y_2+b_2y_3+\cdots+b_{n-1}y_n \tag{5}$$

$y_2,y_3,\cdots,y_n$ are the system input parameters, y_1 is the system output parameter, $a,b_1,b_2,\cdots,b_{n-1}$ is the coefficient of the differential equation(5).

Then, the time response of the above differential equation can be noted as:

$$z(t)=[y_1(0)-\frac{b_1}{a}y_2(t)-\frac{b_2}{a}y_3(t)-\cdots-\frac{b_{n-1}}{a}y_n(t)]e^{-at}+$$
$$\frac{b_1}{a}y_2(t)+\frac{b_2}{a}y_3(t)+\cdots+\frac{b_{n-1}}{a}y_n(t)$$

If $d=\frac{b_1}{a}y_2(t)+\frac{b_2}{a}y_3(t)+\cdots+\frac{b_{n-1}}{a}y_n(t)$, then the above equation can be transformed as:

$$z(t)=\left[(y_1(0)-d)\bullet\frac{e^{-at}}{1+e^{-at}}+d\bullet\frac{1}{1+e^{-at}}\right]\bullet(1+e^{-at})$$
$$=\left[(y_1(0)-d)\bullet\left(1-\frac{1}{1+e^{-at}}\right)+d\bullet\frac{1}{1+e^{-at}}\right]\bullet(1+e^{-at})$$
$$=\left[(y_1(0)-d)-y_1(0)\bullet\frac{1}{1+e^{-at}}+2d\bullet\frac{1}{1+e^{-at}}\right]\bullet(1+e^{-at}) \tag{6}$$

So equation(6) transformed can be mapped into an extended BP neural network which is constructed by four layers with n-input-parameters and 1-output-parameter[7]. This BP neural network is so called the Grey Neural Network, whose topological structure is shown as Fig.1.

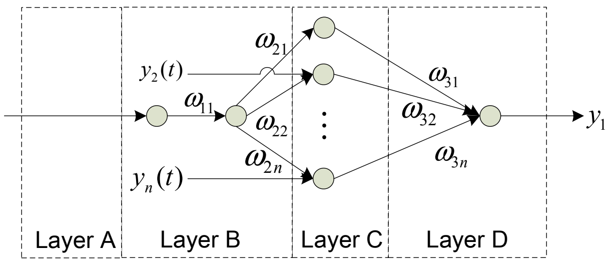

Figure 1. The grey neural network topological structure

t is noted as the sequence number of input parameters. $y_2(t),y_3(t),\cdots,y_n(t)$ are the input parameters of the network. $\omega_{21},\omega_{22}(t),\cdots,\omega_{2n}(t),\omega_{31},\omega_{32}(t),\cdots,\omega_{3n}(t)$ are the weights of the network. y_1 is the prediction value by the grey neural network.

B. *Grey Neural Network Learning Algorithm*

If $\frac{2b_1}{a}=u_1,\frac{2b_2}{a}=u_2,\cdots,\frac{2b_{n-1}}{a}=u_{n-1}$, then the initial value of the network weights can be denoted as:

$$\omega_{11}=a,\omega_{21}=-y_1(0),\omega_{22}=u_1,\omega_{23}=u_2,\cdots,\omega_{2n}=u_{n-1} \tag{7-1}$$

$$\omega_{31}=\omega_{32}=\cdots=w_{3n}=1+e^{-at} \tag{7-2}$$

The threshold value of the node in Layer D is:

$$\theta=(1-e^{-at})[d-y_1(0)] \tag{8}$$

Then the grey neural network learning algorithm can be concluded as follows:

Step1. Initializing the network and its' parameters according to the training samples. In this step parameter u can be calculated both by a and b.

Step2. Calculating the network weights according to equation(7-1) and (7-2).

Step3. Calculating the output of the four-different-layer for every input sequence $(t,y(t))$, $t=1,2,3,\cdots,N$.

Layer A: $a=\omega_{11}t$

Layer B: $b=f(\omega_{11}t)=\frac{1}{1+e^{-\omega_{11}t}}$

Layer C: $c_1=b\omega_{21},c_2=y_2(t)b\omega_{22},\cdots,c_n=y_n(t)\omega_{2n}$

Layer D: $d=\omega_{31}c_1+\omega_{32}c_2+\cdots+\omega_{3n}c_n-\theta_{y1}$

Step4. Calculating the network output error between the prediction value and desired output.

Error of Layer D: $\delta=d-y_1(t)$

Error of Layer C: $\delta_1=\delta_2=\cdots=\delta_n=\delta(1+e^{-\omega_{11}t})$

Error of Layer D:

$$\delta_{n+1} = \frac{1}{1+e^{-\omega_{11}t}}\left(1-\frac{1}{1+e^{-\omega_{11}t}}\right)(\omega_{21}\delta_1 + \omega_{22}\delta_2 + \cdots + \omega_{2n}\delta_n)$$

Step5. Regulating the network weights and threshold value according to the prediction error.

$$\omega_{21} = -y_1(0), \omega_{22} = \omega_{22} - \mu_1\delta_2 b, \cdots, \omega_{2n} = \omega_{2n} - \mu_{n-1}\delta_n b$$

$$\omega_{11} = \omega_{11} + at\delta_{n+1}$$

$$\theta = (1+e^{-\omega_{11}t})\left(\frac{\omega_{22}}{2}y_2(t) + \frac{\omega_{23}}{2}y_3(t) + \cdots + \frac{\omega_{2n}}{2}y_n(t) - y_1(0)\right)$$

Step6. If the network training steps or the output error reaches desired value, then the learning process is over, else the algorithm goes back to Step3.

C. Grey Neural Network Prediction Model

The proposed software fault prediction model based on grey neural network is shown as Figure2. The network structure should be established according to the dimensionalities of the I/O fault data.

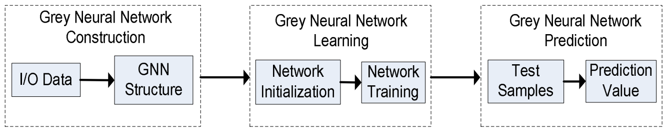
Grey Neural Network Construction
I/O Data
GNN Structure
Grey Neural Network Learning
Network Initialization
Network Training
Grey Neural Network Prediction
Test Samples
Prediction Value

Figure 2. Grey neural network prediction model for software fault prediction

IV. EXPERIMENTAL EVALUATION

Some software project fault numbers is shown as Table 1. Both GM(1,1) and GNN models are used to predict this small sample sequence as in [8].

TABLE I. FAULT NUMBERS IN SOME SOFTWARE PROJECT

Weeks	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
Fault Numbers	18	24	23	20	16	12	13	10

From Talbe1, both the original sequence $x(t) = (18, 24, 23, 20, 16, 12, 13, 10)$ and accumulated sequence $y(t) = (18, 42, 65, 85, 101, 113, 126, 136)$ can be obtained obviously. Then the GNN can be determined with whose structure is SISO(Single Input and Single Output).Following the network learning algorithm referring to 3.2 whose learning rate $u = 0.001$ and desired training error $\varepsilon = 0.005$, the network converged at the 168 iterations where whose output performance error is 0.00493. The training process of the grey neural network is shown as Fig.3.

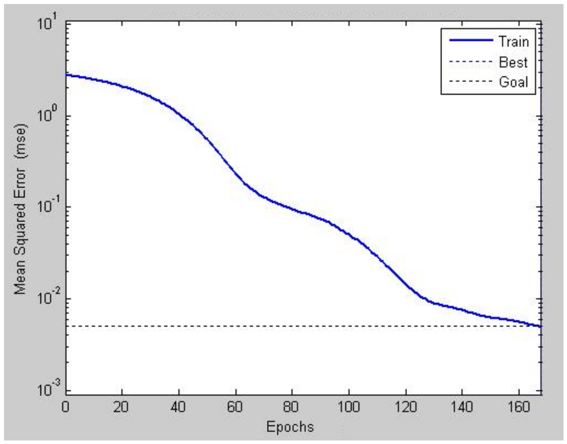

Figure 3. Grey neural network training process

Then we can take use of the proposed GNN model to get the prediction values showed in Fig.4.

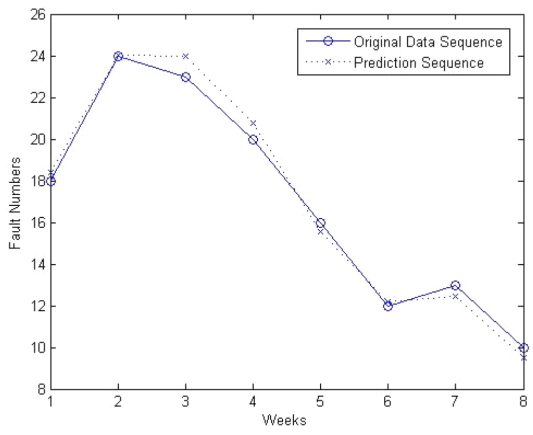

Figure 4. Grey neural network prediction values

From Fig4, the relative prediction error of GNN is only 2.96% less than 6.37% predicted by GM(1,1) as in [8], which indicates that GNN is applicable to the prediction problem of a small sample. Figure5 shows the comparison of the relative error between GNN and GM(1,1).

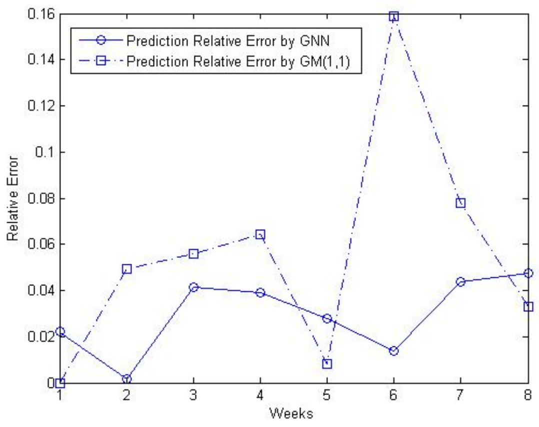

Figure 5. Comparison of the relative error

V. CONCLUSION

The grey neural network prediction model includes network structure construction and learning algorithm. The software fault prediction is an uncertain non-linear problem with only small sample, therefore the proposed method is one of proper solutions according to the experimental results. On the other hand, the GNN training process may have a fast convergence rate, which would make it get into local optimum point. To solve this problem, the other optimization algorithms should be applied perhaps. And of course, it would be our further research work.

REFERENCES

[1] Sherer S A. Software Fault Prediction. Journal of System and Software, 1995, 29, pp. 97-105.

[2] Luo Yun-feng, Ben Ke-rong. BBNs-Based Software Fault Prediction Method. ACTA ELECTRONICA SINICA, 2006, 24(12A), pp. 2380-2383.

[3] KHOSHGOFTAAR T M, SELIYA N, SUNDARESH. An empirical study of predicting software faults with case-based reasoning. Software Quality Journal, 2006, 14, pp. 85-111.

[4] PAI G J, DUGAN J B. Empirical analysis of Software fault content and fault proneness using Bayesian methods. IEEE Trans. On Software Engineering, 2007, 33(10) , pp. 675-686.

[5] PIGHIN M, PODGORELEC V, KOKOL P. Fault-threshold prediction with linear programming methodologies. Empirical Software Engineering, 2003, 8, pp. 117-138.

[6] Catal C, Diri B. A Systematic Review of Software Fault Prediction Studies. Expert Systems with Applications, 2009, 36(4) , pp. 7346-7354.

[7] www.iLoveMatlab.cn. 2011.

[8] Mao Cheng-ying. Study on Risk Analysis and Fault Management Strategy during Software Maintenance, Hefei: Chinese Science and Technology University Press, 2010.