TELKOMNIKA, Vol.11, No.11, November 2013, pp. 6547~6555
e-ISSN: 2087-278X

Software Aging Prediction based on Extreme Learning Machine

Xiaozhi Du[1], Huimin Lu*[2], Gang Liu[2]
[1]School of Software Engineering, Xi'an Jiaotong Univeristy, Xi'an 710049, Shaanxi, China
[2]School of Software Engineering, Changchun University of Technology, Changchun, 130012, Jilin, China
*Corresponding author, e-mail: luhm.cc@gmail.com

Abstract

In the research on software aging and rejuvenation, one of the most important questions is when to trigger the rejuvenation action. And it is useful to predict the system resource utilization state efficiently for determining the rejuvenation time. In this paper, we propose software aging prediction model based on extreme learning machine (ELM) for a real VOD system. First, the data on the parameters of system resources and application server are collected. Then, the data is preprocessed by normalization and principal component analysis (PCA). Then, ELMs are constructed to model the extracted data series of systematic parameters. Finally, we get the predicted data of system resource by computing the sum of the outputs of these ELMs. Experiments show that the proposed software aging prediction method based on wavelet transform and ELM is superior to the artificial neural network (ANN) and support vector machine (SVM) in the aspects of prediction precision and efficiency. Based on the models employed here, software rejuvenation policies can be triggered by actual measurements.

Keywords: *software aging, extreme learning machine, prediction*

1. Introduction

The software reliability and availability are increasingly being demanded in present software systems [1]. While recent studies show that when software application is executed continuously for long intervals of time, some error conditions in them are accumulated to result in performance degradation or even a crash failure, which is called software aging [2]. The phenomenon has been observed in many software systems, such as operating system [3], web server [4], SOA server [5], and so on. Because of the effect of software aging, the system reliability decreases. To counteract software aging and its related transient software failures, a preventive and proactive technique, called software rejuvenation, has been proposed and is becoming popular [2]. It involves stopping the running software occasionally, cleaning its internal state and or its environment and restarting it. An extreme but well-known example of software rejuvenation is the hardware reboot [6]. In general, the cost of software rejuvenation is substantially lower than the cost of a system failure followed by a reactive recovery.

Over the recent years, quantitative studies of software aging and rejuvenation have been taken, and many different approaches have been developed and the effects of software rejuvenation have been studied. These studies can be categorized into two kinds, time-based rejuvenation policy and measure-based rejuvenation policy. The time-based rejuvenation policy is characterized by the fact that the software is periodically rejuvenated every time a predefined time constant has elapsed. In these approaches, a certain failure time distribution is assumed and continuous time Markov chain (CTMC) [2], semi-Markov process [7], Markov regenerative process (MRGP) [8] , Markov decision process (MDP) [9] or stochastic Petri net (SPN) model [10] etc is developed to compute and optimize system availability or related measures. The measure-based rejuvenation policy applies statistical analysis to the measured data on resource availability to predict the expected time to resource exhaustion [11], and it provides a window of time during which a rejuvenation action is advised. The basic idea of the measure-based rejuvenation policy is to monitor and collect data on the attributes and parameters, which are responsible for determining the health of the running software system. Garg et al. [3] proposed a methodology to detect and estimate the aging in the UNIX system, they implemented an SNMP based tool to collect data, and adopted non-parametric statistic method to detect and estimate

Received April 15, 2013; Revised June 19, 2013; Accepted July 20, 2013

aging. Andrzejak et al. [6] used a spline-based description of the aging profiles and adopted a statistical test to verify its correctness of the SOAP server. Andrzejak et al. [12] also used machine learning methods to model and predict the software aging of a web application. Grottke et al. [4] used non-parametric statistical methods to detect and estimate trends of aging, and adopted AR model to predict the aging of a Web server. Hoffmann et al. [13] gave a practice guide to resource forecasting, they adopted several methods to model and predict the software aging of a Web server, they found that probabilistic wrapper (PWA) was a better method for variable select, and support vector machine (SVM) was a better approach for resource forecasting. For the prediction of time series, artificial neural network (ANN) [14] and support vector machine (SVM) [15] are widely adopted. Artificial neural networks, especially BP networks, are powerful tools for fitting nonlinear time series. However, there are some disadvantages in implementing of artificial neural networks. Firstly it’s hard to determine the parameters of neurons and the network structure. Furthermore, the training process of neural networks is time-consuming and the convergence rate is slow, because the networks often settle in undesirable local minima of the error surface. When support vector machine is used for prediction, it also faces some disadvantages and challenges, such as slow learning rate, multi-parameters to be determined, and so on. In order to overcome some challenges of ANN and SVM, extreme learning machine (ELM) proposed by Huange et al.[16] has attracted the more and more attention recently. ELM is adopted for generalize single-hidden layer feedforward networks (SLFNs), and the hidden layer need not be tuned, which results in better generalization performance, faster learning speed, and least human interaction.

In this paper, we analyze the software aging phenomenon of a real VOD system, and propose a software aging prediction model based on extreme learning machine. The main contributions of this paper are 1) proposing a software aging prediction model based on ELM, 2) applying PCA to reduce the dimension of input variables of ELMs, 3) using the data collected from a real VOD system to evaluate the software aging prediction performance.

2. Software Aging Prediction Model based on ELM

In order to provide support for triggering software rejuvenation actions, we need to predict the system resource utilization state precisely to reflect the software system state in the future. Extreme learning machine, which is a learning algorithm, proposed by Huang et al. [16] was developed for single-hidden layer feedforward neural networks (SLFNs). ELM provides good generalization performance at extremely fast learning speed by choosing hidden nodes randomly and determining the output weights of SFLNs analytically. Therefore, we present a software aging prediction model based on extreme learning machine, shown in Figure 1, which illustrates the k-step prediction procedure.

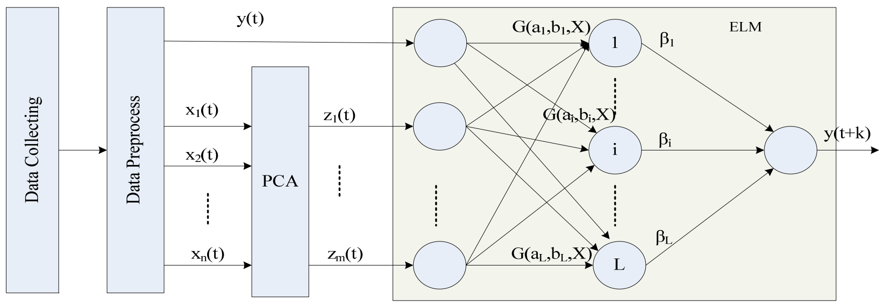


Figure 1. Software Aging Prediction Model

From Figure 1, we see that the data are firstly preprocessed after they are collected. Then the time series of target parameter $y(t)$ is inputted into wavelet transform module, and the detail components $D_i(t), i=1,...,p$ and the approximation component $\mathrm{A_p}(t)$ of $y(t)$ are gotten, where p is the decomposition level assigned by user, t is the sample index. The other parameters $x_i(t), i=1,2,...,n$ are inputted into PCA module, and the first m principals $z_i(t), i=1,2,...,m$ are selected, where $m<n$. Then we construct $p+1$ ELMs to forecast the decomposed components of target parameter separately. Each component ($\mathrm{A}_{\mathrm{p}}(t)$ or $D_i(t), i=1,..., p$) and all the first m principals $z_i(t), i=1,2,...,m$ are the inputs of the respective ELM, and the output of the ELM is the k-step prediction value of the component ($\mathrm{A}*_{\mathrm{p}}(t+k)$ or $D*_i(t+k), i=1,..., p$). Finally, all the outputs of these $p+1$ ELMs are summed to obtain the k-step prediction value of the object parameter $y*(t+k)$.

The detail steps of our approach are as follows:

1) Data Preprocess

Data preprocess includes two phases: (1) Parameter reduction and selection. In this phase, the parameters which are constant values during the monitoring period and the parameters that have the same meanings are excluded. (2) Normalization. In this phase, all the selected parameters are normalized to eliminate dimension influence. After normalization, the range of all these parameters are limited into [-1, 1].

2) Principal Component Analysis

In the experiments, we collect 30 parameters of memory, 15 parameters of CPU, 17 parameters of disk and 27 parameters of application server. Some parameters are constant during the observation interval, such as Commit Limit of memory, C2 Transitions/sec of CPU and so on. Some parameters have the same meaning, such as Available Kbytes and Available Mbytes of memory etc. After all these parameters are excluded, there are still 32 parameters left. The relationship between software aging and these parameters can be expressed as the Equation (1):

$$y = f(x_1, x_2, \cdots, x_n) \tag{1}$$

Where y denotes the available bytes of memory and $x_1, x_2, \cdots, x_n$ are the impact factors of software aging in the VOD system, and here n equals 32.

If we take all these 32 parameters as the inputs of the ELMs directly, the network scale is very large, and its efficiency is very low. PCA [17] is an essential method of multivariate statistical analysis, which selects several representative principle components to explain most of the data changes. Therefore, in order to improve the efficiency and to keep the accuracy of our prediction model, PCA is adopted to reduce the input parameters here.

Let the samples of these factors be $X = (X_1, X_2, \cdots X_n)^T$, then the procedure of PCA is as follows:

(1) Normalize the samples X to remove dimension influence.

(2) Calculate the relative matrix P and covariance matrix C of the sample data, and eigenvalues $\lambda_1, \lambda_2, \cdots \lambda_n$ and eigenvectors are obtained.

(3) Calculate the contribution rate of each component respectively. The choice of principal component is determined based on variance contribution rate and cumulative contribution rate. The variance contribution rates are calculated by the Equation (2):

$$\alpha_k = \lambda_k / (\sum_{i=1}^{n} \lambda_i) \qquad k = (1, 2, \cdots, n) \tag{2}$$

And the cumulative contribution rate for each principle component is gotten by the Equation (3):

$$\rho_m = \sum_{j=1}^{m} \lambda_j / (\sum_{i=1}^{n} \lambda_i) \qquad m = (1, 2, \cdots, n) \tag{3}$$

The higher of the α_1 means the stronger of the ability for the first principal component to abstract the information of $x_1, x_2, \cdots, x_n$. If the accumulation contribution rate of the first m

components is more than a predetermined threshold (such as 85 percent), the first m components are selected as the inputs of the ELM.

After PCA is finished, formula (1) can be reduced to the Equation (4):

$$y = f(z_i, z_2, \cdots z_m) \tag{4}$$

where y denotes the available bytes of memory and $z_1, z_2, \cdots, z_m$ is the principal components of aging impact factors of the VOD system, and here $m < n$.

3) Extreme Learning Machines

Extreme learning machine, which is a learning algorithm, proposed by Huang et al. [16] was developed for single-hidden layer feed forward neural networks (SLFNs). ELM provides good generalization performance at extremely fast learning speed by choosing hidden nodes randomly and determining the output weights of SFLNs analytically. Here we consider a single-hidden layer feed forward network with L hidden neurons. The input $X = (z_1, z_2, ..., z_m, y)$ is a vector with m+1 elements, the output of the i th hidden neuron is $G(a_i, b_i, X)$, where b_i is the bias, and $a_i = (a_{i1}, a_{i2}, ..., a_{im}, a_{iy})$ is the weight vector, $a_{is}(s = 1, 2, ..., m, y)$ is the connection weight between the i th hidden neuron and the s^{th} input neuron. Then the output of the SLFN is given by the Equation (5):

$$y(t+k) = f(X) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, X) \tag{5}$$

Where, $\beta_i = (\beta_{i1}, ..., \beta_{in}, \beta_{iy})'$ is the weight vector connecting hidden layer with output layer, β_{ik} is the connection weight between the i th hidden neuron and the k^{th} output neuron. For the case of additive hidden neurons, $G(a_i, b_i, X)$ takes the following form shown by the Equation (6):

$$G(a_i, b_i, X) = g(a_i' X + b_i) \tag{6}$$

Where $g: R \to R$ is the activation function.

Assume that N arbitrary samples $(X_i, Y_i) \in R^m \times R^n$ are given, the weight vectors a_i and bias b_i are randomly assigned. Then the SLFN with L hidden neurons can approximate the N samples with zero error if and only if there exists β_i, so that we get Y_j by Equation (7):

$$Y_j = \sum_{i=1}^{L} \beta_i G(a_i, b_i, X_j), \; j = 1, 2, ..., N \tag{7}$$

The above N equations can be rewritten in the following compact form shown by Equation (8):

$$H\beta = Y \tag{8}$$

where $H = \begin{bmatrix} G(a_i, b_i, X_1) & \cdots & G(a_L, b_L, X_1) \\ \vdots & \cdots & \vdots \\ G(a_i, b_i, X_N) & & G(a_L, b_L, X_N) \end{bmatrix}_{N \times L}$, $\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m}$, $Y = \begin{bmatrix} Y_1^T \\ \vdots \\ Y_N^T \end{bmatrix}_{N \times M}$.

Once the hidden node parameters (a_i, b_i) are generated randomly, they remain fixed. Then training the SLFN is equivalent to finding the minimum norm least-squares solution β^*, which is given by the Equation (9):

$$\beta^* = H^+ Y \tag{9}$$

where H^+ is the Moore–Penrose generalized inverse of matrix H.

For the training data set $\{(X_i, Y_i)\}_{i=1,2,...,N}$, the ELM algorithm [16] is described as follows:

Step 1: Assign the hidden node number L, and the activation function $g(.)$;

Step 2: For $i = 1, 2, ..., L$, randomly generate the input weight vector a_i and the bias b_i

Step 3: Calculate the hidden layer output matrix H and H^+;

Step 4: According Equation (9), calculate the output weight vector β^*.

For any input sample $x \in R^n$, the output value y^* is calculated by using the Equation (10):

$$y^* = \sum_{i=1}^{L} \beta_i^* g(a_i x + b_i) \tag{10}$$

Then, we get the k-step prediction value of the object parameter, and the procedure is finished.

3. Results and Analysis

3.1. Experimental Setup

The experimental environment is a real VOD system, and its structure is shown in Figure 2. The VOD system consists of a web server, an application server (or video server) and a disk array. The web server acts as the presentation layer, which provides media data to clients. When it receives the order request from a client, the web server redirects this request to the application server. Then the client makes connection with the application server and receives the media data from the application server directly if the client gets the permission. The disk array is used to store the media data. In our experiments, we adopt *Apache* as the web server, and *Helix server* as the application server.

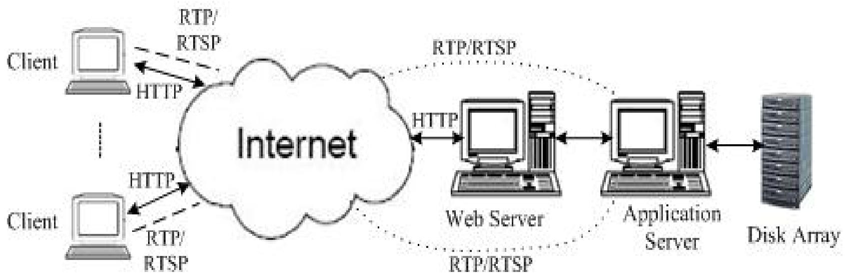


Figure 2. Structure of the VOD System

3.2. Data Collections

During the experiments, we collect 30 parameters of memory, 15 parameters of CPU, 17 parameters of disk and 27 parameters of application server, the sampling interval is 3 minutes, 7500 samples of the system parameters are collected for 375 hours.

The number of client access of the VOD system is illustrated in Figure 3, and the time series of system available memory is shown in Figure 4.

From Figure 3, it is found that the number of client access shows a certain periodicity, however it has a large randomness and it fluctuates frequently. From Figure 4, we see that the available memory shows a frequent and large fluctuation, which is caused by the stochastic arrival of client access and the software aging. According to the time series of the collected system available memory and the number of client access, we use Mann-Kendall method [4] to test whether there is software aging phenomenon in the VOD system. Table 1 shows the results of trend test for the available memory and the number of client access. From Table 1, we find that there is a downward trend in the time series of the available memory, and the time series of the number of client access also has a downward trend. That is, the downward trend of the system available memory is not caused by the clients. Therefore, it can be concluded that there exists software aging phenomenon in the VOD system.

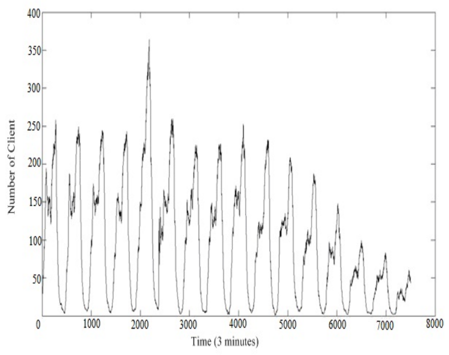


Figure 3. Number of Client Access

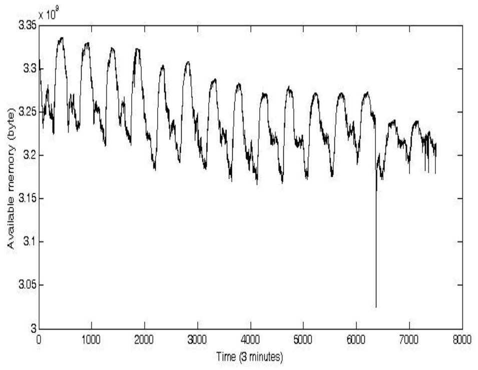


Figure 4. System Available Memory

Table 1. Trend Test

data	Zstatistic	comment
available memory	-41.1374	Downward trend detected
number of client access	-30.7877	Downward trend detected

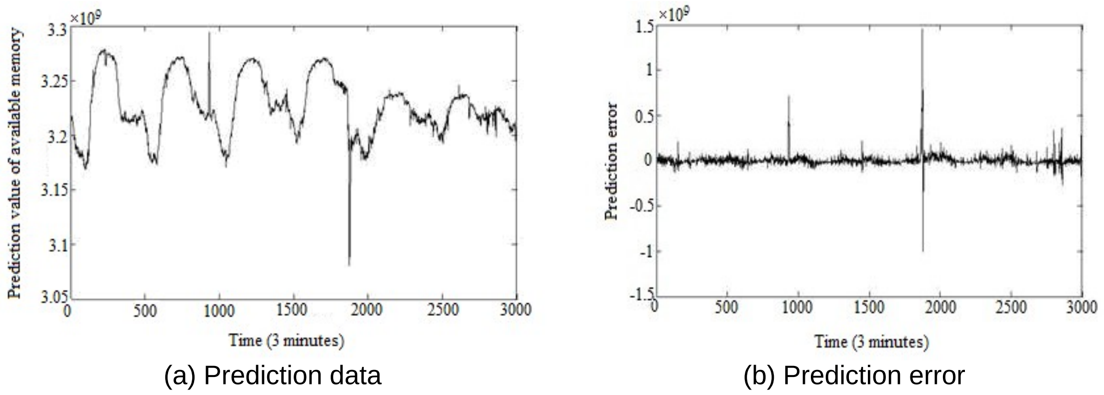
3.3. Software Aging Prediction

In order to evaluate the performance of software aging prediction, root mean square error (RMSE) is adopted as indicator. RMSE is the square root of the variance of the residuals, and it can be interpreted as the standard deviation of the unexplained variance. The lower the values of RMSE, the better the prediction result. RMSE is defined by the Equation (11):\

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(y(i)-\hat{y}(i))^2}{N}} \tag{11}$$

Where, $y(i)$ denotes the actual value of the time series of the available memory, $\hat{y}(i)$ is the respective prediction value, and N is the points of data set.

In the experiments, the number of principal components m is set to 2. For the ELM, we use a sigmoid activation function and the number of hidden neurons is 47. The first 4500 samples are adopted to train these ELM, and the residual 3000 samples are used to test whether our method is effective. Figure 5 shows the one-step forward prediction value of the available memory.



(a) Prediction data (b) Prediction error

Figure 5. Prediction Results of System Available Memory

From Figure 5, we see that the error between the prediction data and the actual data of the available memory resource is very low. Then we conclude that the prediction model proposed by us is suitable for software aging forecasting. We also find that the prediction error tends to be larger at the valley of the resource consumption. The reason is that there are more clients in the system at the valley of the resource consumption, which results in more memory resource consumption and more fluctuation, so the prediction precise decreases.

Table 2 shows the approximation performance of our software aging prediction model compared with support vector machine (SVM), artificial neural network model (ANN) with BP algorithm. The simulations for our model, ANN are carried out in MATLAB R2007a environment running in a Core2 Duo CPU, 3GHz. The simulation for SVM is carried out by using the LIBSVM [18] implemented in C code running in the same PC. The number of hidden neurons of ANN is set 42, and the kernel function used in SVM is radial basis function. In our experiments, all the inputs and the outputs have been normalized into [-1, 1]. 20 trials have been conducted for all the methods and the average results are adopted.

Table 2. Prediction Results of Various Models

Model	Training data		Testing data	
	RMSE	Time (s)	RMSE	Time(s)
Our method	0.0033	0.084	0.0406	0.021
SVM	0.0186	3.797	0.0445	3.734
ANN	0.0214	139.629	0.0698	0.0391

From Table 2, it can be seen that the prediction precision of our method is superior to that of ANN and SVM. The training time and the testing time of our method are far lower than that of SVM, and the training time of our model is far lower than that of ANN. For the training data, the RMSE of our method is 0.0033, and for the testing data, the RMSE is 0.0406. the training time and the testing time of our method are 0.084 seconds and 0.021 seconds, which show that the efficiency of our method is a very high. Therefore, the method we presented is effective to forecast the software aging process.

3.4. Sensitivity Analysis

We perform sensitivity analysis for our presented model that predicts the available memory of the VOD system by adding a principal component at a time, and calculate the model's change in RMSE. The result is shown in Figure 6.

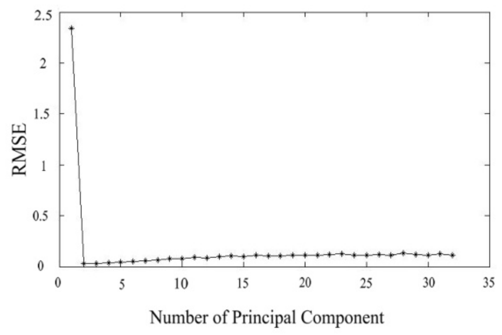


Figure 6. RMSE versus the Number of Principal Components

From Figure 6, it's shown that with the increase of the number of principal components, the RMSE decreases, when the number of principal components arrives to a certain value, the minimum RMSE is gotten. Then with the continuous increase of the number of principal components, the RMSE increases. The reason is that with the increase of the number of principal component, the more information of input variables is included, but except the first several principal components, the residual components have trivial information, and with more principal components, the ELMs are becoming more complex, which results in the increase of RMSE. Therefore, we should select the appropriate number of principal components based on requirement.

4. Conclusion

In this paper, we have investigated the software aging phenomenon of a real VOD system, and have proposed a software aging prediction model base on extreme learning machine. The experimental results have showen that the proposed software aging prediction model is effective to forecast the aging progress, and the PCA is an important and useful method to reduce the redundancy of data. Compared with SVM and ANN prediction model, our model is more effective on precision than ANN and SVM. And the time for training and testing of our model is far lower than that of ANN and SVM.

ELM is a quick and efficient method for resolving the prediction problem, but how to determine the number of hidden neurons is an open issue, though several methods have been presented, it is still a challenge for the ELM. Therefore, in the future we will study this issue. Though in this paper we have study the software aging prediction problem, the cause resulting in software aging is still pending, so next we will explore this problem.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant No. 60933003, the National Natural Science Foundation of China under Grant No. 61240029, the National Postdoctoral Science Foundation of China under Grant No. 2011M500611, the Industrial Technology Research and Development Special Project of Jilin Province under Grant No. 2011006-9; the Fundamental Research Funds for the Central Universities; the National College Students' Innovative Training Program of China under Grant No. 201210190037.

References

[1] Paulson LD. Computer System, Heal Thyself. IEEE Computer. 2002; 35(8): 20-22.

[2] Huang Y, Kintala C, Kolettis N, Fulton D. *Software Rejuvenation: Analysis, Module and Applications. Proceedings of the 25th Symp*. On Fault Tolerant Computing. Pasadena, USA. 1995; 381-390.

[3] Garg S, Puliafito A, Telek M, Trivedi KS. *A Methodology for Detection and Estimation of Software Aging. Proceedings of the Intl. Symp.* On Software Reliability Engineering. NJ, USA. 1998; 283-292.

[4] Grottke M, Li L, Vaidyanathan K, Trivedi KS. Analysis of Software Aging in a Web Server. *IEEE Transactions on Reliability*. 2006; 55(3): 411-420.

[5] Silva L, Madeira H, Silva JG. *Software Aging and Rejuvenation in a Soap-based Server.* Proceedings of the Fifth IEEE International Symposium on Network Computing and Applications. Massachusetts, USA. 2006; 56-65.

[6] Alonso J, Matias R, Vicente E, et al. A Comparative Experimental Study of Software Rejuvenation Overhead. *Performance Evaluation.* 2013; 70: 231-250.

[7] Bao Y, Sun X, Trivedi KS. A Workload-based Analysis of Software Aging and Rejuvenation. *IEEE Transactions on Reliability.* 2005; 54(3): 541-548.

[8] Garg S, Puliafito A, Telek M, et al. Analysis of Preventive Maintenance in Transactions Based Software Systems. *IEEE Trans. on Computers*. 1998; 47(1): 96-107.

[9] Okamura H, Dohi T. Dynamic Software Rejuvenation Policies in a Transaction-based System under Markovian Arrival Processes. *Performance Evaluation*. 2013; 70: 197-211.

[10] Wang D, Xie W, Trived KS. Performability Analysis of Clustered Systems with Rejuvenation under Varying Workload. *Performance Evaluation.* 2007; 247-265.

[11] Vaidyanathan K, Trivedi KS. A Comprehensive Model for Software Rejuvenation. *IEEE Transactions on Dependable and Secure Computing.* 2005; 124-137.

[12] Andrzejak A, Silva L. *Using Machine Learning for Non-intrusive Modeling and Prediction of Software Aging.* Proceedings of the IEEE Network Operations and Management Symposium. Salvador, Brazil. 2008; 25-32.

[13] Hoffmann GA, Trivedi KS, Malek M. A Best Practice Guide to Resource Forecasting for Computing Systems. *IEEE Transactions on Reliability*. 2007; 56: 615-628.

[14] Goh CK, Teoh EJ, Tan KC. Hybrid Multiobjective Evolutionary Design for Artificial Neural Networks. *IEEE Transactions on Neural Networks*. 2008; 19: 1531-1548.

[15] Cao LJ, Tay FEH. Support Vector Machine with Adaptive Parameters in Financial Time Series Forecasting. *IEEE Transactions on Neural Networks*. 2003; 14(6): 1506-1518.
[16] Huang GB, Zhu QY, Siew CK. Extreme Learning Machine: Theory and Applications. *Neurocomputing.* 2006; 70(1-3): 489-501.
[17] Wang X, Kruger U, Irwin GW, McCullough G, McDowel N. Nonlinear PCA with the Local Approach for Diesel Engine Fault Detection and Diagnosis. *IEEE Transactions on Control System Technology*. 2008; 16: 122-129.
[18] Ferrari S, Stengel RF. Smooth Function Approximation Using Neural Networks. *IEEE Transactions on Neural Networks*. 2005; 16(1): 24-38.