REVIEW OF COMPUTER ENGINEERING STUDIES
Vol.2, No.3, 2015, pp.1-8
http://dx.doi.org/10.18280/rces.020301

A PRACTICE GUIDE OF PREDICTING RESOURCE CONSUMPTION IN A WEB SERVER

Yongquan Yan

*School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China.

Email:yongquanyan@aliyun.com

ABSTRACT

Software aging is a phenomenon observed in a software application executing continuously for a long period of time, where the state of software degrades and leads to performance degradation, hang/crash failures or both. This gives rise to large costs and requires provisional solutions. Software rejuvenation is a proactive error management technique that involves terminating an application, cleaning up the system internal state, and restarting it to prevent the future failures. Thus, it is important to know system states in advance. In order to find software aging in advance, algorithms of time series and machine learning have been used to forecast resource consumption in various kinds of software systems. In this paper, we use linear regression to find if there is software aging trend in a web server and compare the differences of linear and nonlinear methods in the aspect of resource consumption prediction. Through analysis in the experiment, it is a fact that choosing a proper data set is more important than choosing a method for software resource consumption forecasting.

Keywords: Software aging, Artificial Neural Network, Support vector machine, ARIMA, Web server.

1. INTRODUCTION

There is a report that business revenues affected by application performance problems account for overall business revenues by up to 9% when applications suffer from performance issues such as performance degradation [1]. Even worse, about one third (32 percent) of consumers will abandon slow sites between one and five seconds and some of them will go on to tell others about their experience which will prevent more consumers to visit these sites by [2]. One second page load delay means that it will decrease 11% page views, customer satisfaction, and 7% revenue [3].

Several studies [4], [5] have reported that one of the causes of the performance degradation and unplanned software outages is the software aging phenomena. Software aging is a phenomenon observed in software that the long running software system suffers from abnormal state, performance degradation, even hang, and failure. The reasons of software aging are consumption of operating system resources, data corruption and round error accumulation, which could be accompanied by memory leaks [6], unterminated threads, data fragment, unreleased file locks, unreleased database connections and so on. Software aging problems are not only observed in telecommunication systems [7], Web servers [8], enterprise clusters [9], online transaction processing (OLTP) systems [10], and spacecraft systems [11], but also in military system [5] which means loss of lives. In order to counteract these problems, Huang et al. [12] proposed the technique of software rejuvenation that involves occasionally stopping the software application, removing the cumulative error environments and then rebooting the application in a clean environment. Unlike other technologies, rejuvenation technology is a proactive manner, which means the accumulated errors, or defragments can be removed through this process before occurrence of service degradation and failure. Unlike downtime caused by unexpected failure, the downtime by software rejuvenation can be performed at the discretion of the user or administrator, e.g., at midnight or idle time.

The key issue for software rejuvenation is how to find an optimal time, however, the choice of optimal time is dependent on accuracy of time to failure or prediction accuracy of resource consumption, which means the more the accuracy is, the better the rejuvenation is.

In this paper, we compare the effects of linear and nonlinear methods in the aspect of resource consumption predicting in a web server that is suffered from software aging. The resource usage data are collected from a typical long running software system a web server that is not subjected to artificial load, but a true load.

The contribution of this work is: the captured data from running phase are real data, which means workload is not synthetic; through comparisons between the linear and nonlinear time series methods in resource consumption, the most important thing for resource consumption predicting is how to choose a proper data set for training.

The rest of the paper is organized as follows. In Section 2, some related work on software aging is briefly reviewed. In Section 3, we review the ARIMA, ANN, and SVM modeling approaches. In Section 4, the outputs of models are compared with each other through some measurement methods. Section 5 contains the concluding remarks and discusses possible directions for future research.

2. RELATED WORK

Approaches used for analyzing and solving software aging problem can be grouped into model-based ones and measurement-based ones.

In model-based approach, some assumptions such as distributions for failure, workload and repair time are made. And based on these assumptions a mathematical model is built. These models are either analytical or stochastic. Garg et al. [13] analyzed software aging in a transaction system and proposed an analytical model to estimate the probability of losing an arriving transaction and the expected response time of a transaction. Dohi et al. [14] built semi Markov reward process models of software aging to solve the same problem. Xie et al. [15] generalized the semi Markov model presented in [14] by introducing the possibility of service level rejuvenation in addition to system level rejuvenation. Wang et al. [16] analyzed performability of clustered systems with deterministic and stochastic Petri nets (DSPN) and homogeneous continuous-time Markov chains (CTMC). There is a problem in model-based approach that is the mathematical assumptions cannot be easily validated in practice and may be not proper for software aging problem in production phase. Meanwhile, the rejuvenation for model-based approach is usually based on a regular time interval, which may be not a good choice for real problems especially for web service as the most popular service of Internet whose rejuvenation is sensitive to the time, which means rejuvenation in this time is proper, but may be not appropriate in other time.

In measurement-based approach, the aging parameters (performance parameters or resource consumption parameters) are monitored through some tools such as Monit [17], Ganglia [18], Munin [19] or Nagios [20]. Based on periodically collected data, measurement-based approaches try to assess current or future state of software system. Garg et al. [21] analyzed the exhaustion of resources like physical memory, and swap space in a network of UNIX workstations and proposed a slope estimation method to quantify the effect of aging in operating system level. Silva et al. [22] collected data from the application server and used Hot Standby technology to implement rejuvenation on a virtualization (XEN virtualization middleware) layer. Shereshevsky et al. [23] chose a completely different approach to the analysis of aging in memory resources. Instead of modeling and predicting memory utilization directly, they monitored the Hölder exponent (a measure of the local rate of fractality) of the system parameters. Their findings indicated that system crashes were often preceded by the second abrupt increase in this measure. Chen et al. [24] used non-linear threshold autoregressive (TAR) model to forecast the resource usage. Michael Grottke et al. [25] collected data from apache web server and use autoregressive model to predict system resources consumption, i.e., free physical memory, and used swap space. They found that there was a periodical pattern in used swap space. Machida et al. [26] that aging detection using the Mann-Kendall test alone was in general unreliable, or might require long measurement times. Li et al. [27] presented a method of nonlinear autoregressive models with exogenous inputs to detect the aging phenomenon of the software system which is a variant for neural network. Hoffmann et al. [28] used some predictive algorithms (liner regression, support vector machines and so on) to predict resource consumption in an industrial telecommunication system. Araujo et al. [29] used four kinds of time series models to forecast resource consumption in Eucalyptus Cloud Computing Infrastructure. Simeonov and Avresky [30] presented a framework for detecting anomalies in servers leading to crash such as memory leaks in aging systems. However, most of above approaches analyze software aging phenomenon by using synthetic load data which neglects the real runtime environment of software system. Meanwhile, the fault injection method or overload method is used in order to get aging data as quickly as possible. However, according to the definition of software aging, the software aging appears when software system is in long running phase, so the method of fault injection or overload may be not proper, especially for the software system in production phase. Although there are some methods for predicting resource consumption, it lacks practice guides that which method should be used when a web server is suffered from software aging.

3. METHOD

There are some different methods for resource consumption forecast, which can be generally categorized as traditional statistical models and nonlinear models. Traditional statistical models including moving average, exponential smoothing, and autoregressive integrated moving average are linear in that predictions of future values are constrained to be linear functions of past observations. The Second category of models is nonlinear models. Several nonlinear models have been proposed to overcome the linear limitation of time series models, which include the threshold autoregressive (TAR) model, general autoregressive conditional heteroscedastic (GARCH), the bilinear model, chaotic dynamics, and artificial neural networks, general regression neural networks (GRNNs), support vector machines, and so on. In this section, the basic concepts and modeling process of autoregressive integrated moving average, artificial neural networks, and support vector machines are reviewed.

3.1 ARIMA model

In the 1970s, Box and Jenkins [31] proposed a set of methods for analysis, prediction and controllable method, called ARIMA or Box-Jenkins method. ARIMA model [32] is, in theory, the most general model for forecasting a time series which can be stabilized by transformations such as differencing. In fact, we can think that ARIMA model is an adjusted model of random walk and random trend: the fine adjustment consists of adding lags of the differenced series and/or lags of the forecast errors for the forecasting formula, where removing any last traces of autocorrelation from the forecast errors is needed.

In fact, an ARIMA (p, d, q) model uses autoregressive moving average model (ARMA) to fit time series which is stationary. A stationary time series means that the series has no trend in the long run.

Given a set of resource consumption data series (x_1, …, x_t, …, x_m), where t = 1, …, m, the difference transformation can be expressed as

$$\nabla^d x_t = x_t - x_{t-d} = (1-B)^d x_t, \tag{1}$$

Where ∇^d is the difference operator, d is the number of the difference, x_t and x_{t-1} are the actual values of time series at time t and t-1, B is the backward shift operator..

Supposing x_t is the output of a stationary time series, ARMA (*p, q*) model which is a stationary ARIMA can be described by (2)

$$x_t = \varphi_1 x_{t-1} + \ldots + \varphi_p x_{t-p} + \ldots + \varepsilon_t - \theta_1 \varepsilon_{t-1} - \ldots - \theta_q \varepsilon_{t-q} \quad (2)$$

Where ε_t is a white noise which is an uncorrelated random variable with mean zero and a constant variance σ_ε^2, p is the order of non-seasonal autoregressive part of the model, q is the order of non-seasonal moving average part of the model, φ and θ are coefficients that satisfy stationary and invertible conditions, respectively.

Using the backward shift operator B with $Bx_t = x_{t-1}$ and $B\varepsilon_t = \varepsilon_{t-1}$, we can describe (2) as

$$\begin{aligned} &\Psi(B)x_t = \Theta(B)\varepsilon_t \\ &\Psi(B) = (1 - \varphi_1 B - \varphi_2 B^2 - \ldots - \varphi_p B^p) \\ &\Theta(B) = (1 - \theta_1 B - \theta_2 B^2 - \ldots - \theta_q B^q). \end{aligned} \quad (3)$$

When q is equal to zero, the model turns into AR(*p*) model.

$$x_t = \varphi_1 x_1 + \ldots + \varphi_p x_{t-p} + \varepsilon_t. \quad (4)$$

When p is equal to zero, the model turns into MA(*q*) model.

$$x_t = \varepsilon_t - \theta_1 \varepsilon_{t-1} - \ldots - \theta_q \varepsilon_{t-q}. \quad (5)$$

3.2 Artificial neural network

Artificial Neural network (ANN) can be viewed as an attempt by humans to simulate the human brain function and be a stretch computing framework for modeling a broad range of nonlinear problems. The ANNs can process the information from the data in parallel and approximate a large class of functions with a high degree of accuracy over nonlinear data. In the process of model building, there is no prior assumption required. On the contrary, the trained network model is largely determined by the features of the data. Multilayer perceptron (MLP), especially with one hidden layer is one of the most widely used forms of artificial neural networks for modeling and forecasting. The model is characterized by a network of three layers of simple processing units connected by acyclic links. The relationship between the output (x_t) and the inputs (x_{t-1}, …, x_{t-P}) can be expressed as

$$x_t = w_0 + \sum_{j=1}^{q} w_j g(w_{0,j} + \sum_{i=1}^{p} w_{i,j} x_{t-i}) + e_t, \quad (6)$$

Where $w_{i,j}$ (i=0, 1, 2, …, p, j=1, 2, …, q) and w_j (j=0, 1, 2, …, q) are model parameters called connection weights; p is the number of input nodes; and q is the number of hidden nodes. The sigmoid function is often used as the hidden layer transfer function, that is

$$sig(x) = \frac{1}{1 + \exp(-x)}. \quad (7)$$

Hence, the ANN model implements a nonlinear functional mapping from the past observations to the future value x_t, i.e.

$$x_t = f(x_{t-1}, \ldots, x_{t-p}, W) + e_t. \quad (8)$$

Where W is a vector of all parameters and f is a function determined by the network structure and connection weights. Therefore, the neural network is equivalent to a nonlinear autoregressive model. Note that (6) implies one output node in the output layer, which is typically used for one-step-ahead forecasting. In practice, simple network structure that has a small number of hidden nodes often works well in out-of-sample forecasting. This may be due to the over-fitting effect typically found in process of neural network modeling. It occurs when the network has too many free parameters, which allow the network to fit the training data well, but typically lead to poor generalization. In addition, it has been experimentally shown that the generalization ability begins to deteriorate when the neural network has been trained more than necessary, that is when it begins to fit the noise of the training data.

The choice of q is dependent on the characteristic of data, which has no systematic rule to decide this parameter. Also, there is no theory that can be used to decide the selection of p. Hence, experiments are often conducted to select an appropriate p as well as q.

3.3 Support vector machine

In a time series, data set can be divided into training set and testing set. A set of training data (x_1, y_1), …, (x_m, y_m) are given, where $x_i \in R^n$, i = 1, …, m, and $y_i \in R$. Each y_i is the target value for the input vector x_i and y_i also comes from the same time series as x_i comes. Generally, a time series function f(x) can be written as

$$f(x) = \langle w, x \rangle + b, \quad (9)$$

Where w and b are, respectively, the weight vector and intercept of the model for the optimal regression. Also, $\langle \cdot, \cdot \rangle$ means the inner product of the involved arguments.

Support vector regression (SVR) is one of SVM models, which is an application of SVMs in time series problem. SVR method tries to locate a low-dimensional hyperplane with small risk in high-dimensional feature space based on nonlinear kernel regression method. Among numerous of support vector regression methods, the most commonly used method is ε-SVR which finds a nonlinear hyperplane with an ε-insensitive band [33], [34]. In order to make the method more robust, the mapping from the input data to the target data does not need to lie rigorously inside or on the ε-insensitive band. The mappings outside the ε-insensitive band are punished and slack variables are imported for the mappings. For convenience, in the following, the term SVR is used to present ε-SVR. The objective function with constraints for SVR is as follows

$$\min_{w,b} \frac{1}{2}\langle w,w\rangle + C\sum_{i=1}^{m}(\xi_i + \xi_i^*)$$
$$\text{subject to } (\langle w,\phi(x_i)\rangle + b) - y_i \le \varepsilon + \xi_i, \quad (10)$$
$$y_i - (\langle w,\phi(x_i)\rangle + b) \le \varepsilon + \xi_i^*,$$
$$\xi_i, \xi^*_i \ge 0,$$

Where i=1, …, m, m is the number of training set, C is a parameter which gives a trade-off between model training error and complexity, ξ_i and ξ_i^* are slack variables which are used when the predictable value is above or below the target value more than ε.

One thing to note is that $\phi(x)$ is a nonlinear mapping function from the input space to a feature space. Therefore, we can rewrite the general function (9) into the derived hyperplane which is

$$f(x) = \langle w,\phi(x)\rangle + b. \quad (11)$$

To solve (10), one can introduce the Lagrangian multipliers, take partial derivatives regard to the primal variables and set the results to be zero, and turn the Lagrangian into the following Wolfe dual form

$$\max_{\alpha,\alpha^*} \sum_{i=1}^{m} y_i(\alpha_i - \alpha_i^*) - \varepsilon\sum_{i=1}^{m}(\alpha_i + \alpha_i^*)$$
$$-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i(\alpha_i^* - \alpha_i) - \varepsilon\sum_{i=1}^{m}(\alpha_j^* - \alpha_j)K(x_i,x_j), \quad (12)$$
$$\text{subject to } \sum_{i=1}^{m}(\alpha_i^* - \alpha_i) = 0,$$
$$0 \le \alpha_i, \alpha_i^* \le C,$$

where i is equal to 1, …, m, α_i and α_i^* are Lagrange multipliers. $K(x_i,x_j)$ is a kernel function which represents the inner product of $\langle \phi(x_i),\phi(x_j)\rangle$.

If $0 < \alpha_i, \alpha_i^* < C$, the corresponding training sample is a free support vector.

If $\alpha_i, \alpha_i^* = C$, the corresponding training sample is a boundary support vector.

If $\alpha_i, \alpha_i^* = 0$, the corresponding training sample is a non-support vector, which doesn't affect the classification or regression result.

Free support vectors and boundary support vectors are called support vectors.

A widely adopted kernel function also used in this paper is the radial basis function (RBF) which is defined as follows

$$k(x_i,x_j) = \langle \phi(x_i),\phi(x_j)\rangle = \exp(-\gamma \|x_i - x_j\|^2), \quad (13)$$

Where γ is the width parameter of the RBF kernel. Now, equation (12) can be solved by Sequential Minimal Optimization (SMO) algorithm [35]. Suppose $\hat{\alpha}_i$ and $\hat{\alpha}_i^*$ are the optimal values by computation, the hyperplane for the time series problem can be shown as

$$f(x) = \sum_{i=1}^{m}(\hat{\alpha}_i^* - \hat{\alpha}_i)K(x_i,x) + \hat{b}. \quad (14)$$

4. RESULT AND DISCUSSION

In this section, we describe the setup of data acquisition environment used in this work. The environment is an actual running web environment, which is composed of a web application server and a database server.

The web application server is Internet Information Services 6.0 (IIS) developed by Microsoft and SQL Server 2005 as database.

In web server aging analysis, most of studies focused at aging of Apache web server that is a popular web server for web application. However, the aging problem for IIS, which is as the second web server software [36], is neglected by most of scholars. In this paper, we want to find whether the IIS has aging phenomenon, and how to more accurately predict resource consumption when web application suffers aging problem. The applications on the IIS web server are composed of a set of health care applications which contain hospital websites, health government websites and some applications about health care such as online registration system. The application system does not use a hot standby system that means the only way to handle the performance degradation or failure is to reboot IIS or operating system.

There are many tools for capturing parameters from application system such as Nagios, but we use built-in windows counter which can get operating system parameters and other parameters such as IIS parameters or database parameters in running phase without disturbing the running system. The collected data that are between two reboots due to performance or failure problem contains: web service, process, .net common language runtime memory, etc. But in this work, we only use available memory and .net common language runtime memory in all heaps (heap memory, for short) as well as Java heap memory in Apache web server, because of their representation for system level (operating system level) and application level (web service level provided by IIS web server).

The resource consumption data are collected every 1 minute. In this work, the data collection time between two reboots is more than 13,000 minutes, which means that the number of the data is more than 13,000. Before using the collected data set to train these models, preprocessing need to be done. For example, the gathered data through windows counter contain some empty values that need to be removed before using data to train model.

4.1 Software aging trend test

In order to investigate software aging problem of IIS web server, we first need to analyze the collected data to find whether software aging trend exists, which is indicated by degradation in performance of the web server and/or exhaustion of resources.

We use a linear regression model to fit the monitored resource data to find if there is an aging trend in IIS web server system.

$$y = at + g \quad (15)$$

Where *y* denotes the consumed resource, *t* is the time from beginning of start of the operating system and IIS web server, a denotes the degradation rate and g is the constant bias.

To obtain appropriate values of a and g, the least squares method is adopted to minimize the summed square of errors.

The summed square of errors is given by

$$E = \sum_{t=1}^{m} (y_t - \hat{y}_t)^2, \quad (16)$$

Where y_t denotes observed value at time t, $\hat{y}_t$ is the estimated value at time t and m is the number of training data set.

Figure 1 gives curves of observed data and linear regression. In Figure 1, it is easy to look down the slope of this trend, so the slope of linear regression model is negative, which means that the available memory gradually decreases as time goes on.

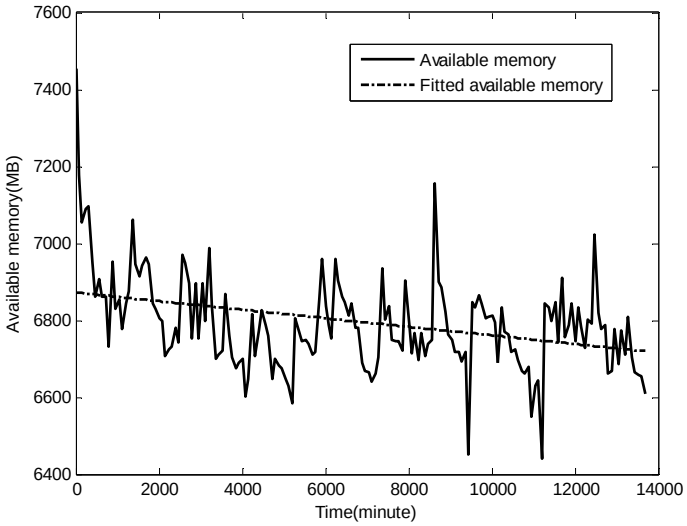

Figure 1. Available memory estimation

Figure 1 shows aging trend in operating system level, also we want to know whether application level as same as web service provided by web server is subjected to aging or not. Therefore, we train linear regression model of heap memory in the collected data. In Figure 2, we can see that the line shows an upward tendency. Therefore, the slope of linear regression model is positive, which indicates that the consumption by web server will increase over time and aging phenomenon does exist in application level.

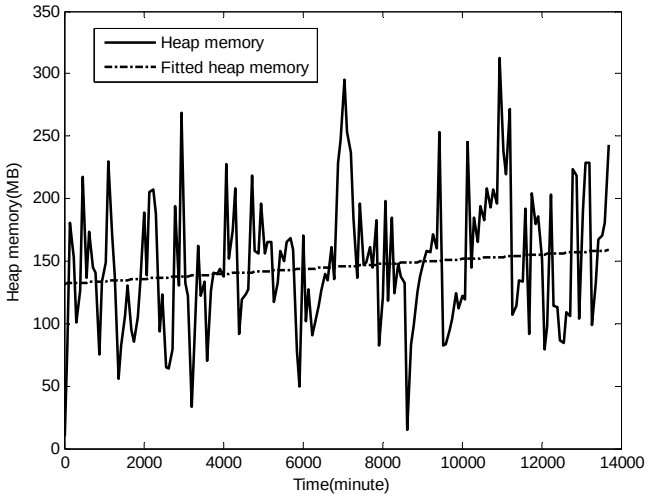

Figure 2. Heap memory estimation

4.2 Available memory forecasting

In order to find which of three models is appropriate in predicting available memory based on different situations, we split the collected data set into two parts: training set and testing set. In this paper, we use 5%, 10%, 50%, 70% of the whole data set as training data set and the remainder data set is used as testing set.

The MAE (Mean Absolute Error) which is computed from the following equation is employed as performance indicator to measure forecasting performance.

$$MAE = \frac{1}{n}\sum_{t=1}^{n} | y_t - \hat{y}_t |. \quad (17)$$

In Table 1, the training results of different models for available memory prediction are given.

Table 1. Comparison of the performance of different models in training set (available memory)

Rate of the training set	5%	10%	50%	70%
ARIMA	27.362	28.334	24.632	24.915
ANN	27.782	29.365	25.56	25.742
SVM	26.566	28.889	24.082	24.167

In Table 2, the testing results of different models for available memory prediction are given.

Table 2. Comparison of the performance of different models in testing set (available memory)

Rate of the testing set	95%	80%	50%	30%
ARIMA	25.329	24.794	25.198	24.913
ANN	42.547	45.501	25.897	25.562
SVM	63.624	52.084	24.963	23.734

In the testing phase of available memory, ARIMA has a steady performance in predicting available memory. ANN and SVM have weak performances in the testing set of 95%, 80%, and SVM has worse results than ANN's. In Table 3, we find that the linear correlations of the training and testing phases have bigger differences in the rate of 5% and 10% than others through comparisons of linear correlations between collected data and fitted data by SVM. Therefore, it is difficult to fit the collected data of the testing phase, when the model of training phase can hardly fit the testing phase.

Table 3. Linear correlation between collected data and fitted data by SVM (available memory)

Rate of the training set	5%	10%	50%	70%
Linear correlation in the training phase	0.959	0.944	0.946	0.941
Linear correlation in the testing phase	0.748	0.752	0.897	0.902

Through the analysis for ARIMA, ANN, and SVM, we can see that nonlinear methods such as ANN and SVM do not do better than linear method in available memory predicting. The reason may be that the gap of the linear correlation between the training data and testing data is large in small data set and for SVM and ANN, the initial 5% and 10% data as training data have a large linear correlation.

4.3 Heap memory forecasting

In Table 4, the training results of different models for heap memory prediction are given.

Table 4. Comparison of the performance of different models in training set (heap memory)

Rate of the training set	5%	10%	50%	70%
ARIMA	27.815	28.186	26.149	26.002
ANN	29.497	28.993	26.564	26.265
SVM	27.476	26.88	23.556	23.419

In Table 4, the entire models do well in fitting original data in the training set of 5%, 10%, 50%, 70%, especially for SVM.

In table 5, the testing results of different models for heap memory prediction are given.

Table 5. Comparison of the performance of different models in testing set (heap memory)

Rate of the testing set	5%	10%	50%	70%
ARIMA	26.505	26.165	26.543	27.118
ANN	28.721	27.962	27.778	27.074
SVM	26.465	25.623	24.796	25.103

Through the analysis for ARIMA, ANN, and SVM, we can see that SVM as one of nonlinear methods does better than the other two methods in heap memory prediction. Table 6 gives linear correlation for SVM.

Table 6. Linear correlation between collected data and fitted data by SVM (heap memory)

Rate of the training set	5%	10%	50%	70%
Linear correlation in the training phase	0.64	0.673	0.659	0.69
Linear correlation in the testing phase	0.655	0.689	0.73	0.729

In Table 6, the differences of linear correlations in the training and testing phases are small no matter for small training data set or large training data set.

Through analysis of linear correlations in predictions of available memory and heap memory, we find that finding a proper data set to train and test is more important than to find a model.

6. CONCLUSIONS

Software aging analysis and forecasting is an active research area over the last few decades. In this paper, we compare the performances of three models in resource consumption forecasting of a web server. In section 4, we first identify whether the system has degradation trend by using linear regression model. After that, we use the collected data to train three models and use the remainder data to test effect of the models. Through the results of available memory and heap memory prediction, it is shown that nonlinear methods do not better than linear method in some situations. Through two kinds of resource consumption, we believe that choosing a proper data set for training is more important than choosing a linear or nonlinear method.

There are some topics for future research. The relationships between different resource parameters or performance parameters need to be analyzed and new systems, such as platform of Cloud Computing, are also need be considered, when subjected to software aging.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their detailed reviews and constructive comments, which have helped improve the quality of this paper. This work is supported by the grants from Natural Science Foundation of China (Project No. 61375045) and Beijing Natural Science Foundation (4142030).

REFERENCES

1. Web Performance Impacts, http://www.webperformancetoday.com/2010/06/15/everything-you-wanted-to-know-about-web-performance/.
2. Website Response Times, http://www.nngroup.com/articles/website-response-times/.
3. Benchmark, http://www.aberdeen.com/Aberdeen-Library/5136/RA-performance-web-application.aspx
4. T. Dohi, K. GoHeva-Popstojanov, K.S. Trivedi, Analysis of Software Cost Models with Rejuvenation, Proc. Fifth IEEE Symp, *High Assurance Systems Engineering*, pp.25-34, 2000.
5. M. Grottke, R. M. Jr, K. S.Trivedi, The Fundamentals of Software Aging, *Proc. Workshop Software Reliability Engineering*, pp.1-6, 2008.
6. Z. Dai, X. G. Mao, Light-Weight Resource Leak Testing Based On Finalisers, *IET Software*, vol.7. pp.308- 316, 2013.
7. Avritzer, E. Weyuker, Monitoring Smoothly Degrading Systems for Increased Dependability, *Empirical Software Eng. J.*, vol. 2. pp.59-77, 1997.
8. Apache, http://httpd.apache.org/docs/.
9. V. Castelli, R. Harper, P. Heidelberg, Proactive Management of Software Aging, *IBM J. Research and Development*, vol. 45. pp.311-322, 2001.
10. K. Cassidy, K. Gross, A. Malekpour, Advanced Pattern Recognition for Detection of Complex Software Aging Phenomena in Online Transaction Processing Servers, *Proc. 2002 Int'l Conf. Dependable Systems and Networks*, pp.478-482, 2002.
11. E. Marshall, Fatal Error: How Patriot Overlooked a Scud, *Science*, vol. 255. pp.1344-1347, 1992.
12. Y. Huang, C. Kintala, N. Kolettis, et al., Software Rejuvenation: Analysis, Module and Applications, Proc. Twenty-Fifth Symp., *Fault-Tolerant Computing*, pp.381-390, 1995.
13. S. Garg, A. Puliafito, M. Telek, et al., Analysis of Software Rejuvenation Using Markov Regenerative Stochastic Petri Net, *Proc. Sixth IEEE Symp. Software Reliability Engineering*, pp.180-187, 1995.
14. T. Dohi, K. Goševa-Popstojanova, K. Trivedi, Estimating Software Rejuvenation Schedules in High-assurance Systems, *The Computer Journal*, vol. 44. pp.473-485, 2001.
15. W. Xie, Y. Hong, K. Trivedi, Analysis of a Two-level Software Rejuvenation Policy, *Reliability Engineering and System Safety*, vol. 87. pp.13-22, 2005.
16. D. Wang, W. Xie, K. S. Trivedi, Performability Analysis of Clustered Systems with Rejuvenation under Varying WorkLoad, *Performance Evaluation*, vol. 64. pp.247-265, 2007.
17. Monit, http://mmonit.com/monit/
18. Ganglia, http://ganglia.sourceforge.net/
19. Munin, http://munin-monitoring.org/
20. Nagios, http://www.nagios.org/
21. S. Garg, A. van Moorsel, K. Vaidyanathan Trivedi, et al., A Methodology for Detection and Estimation of Software Aging, *Proc. Ninth Int'l Conf. Software Reliability Engineering*, pp.283-292, 1998.
22. L. M. Silva, J. Alonso, J. Torres, Using Virtualization to Improve Software Rejuvenation, *Proc. Sixth IEEE Symp. Network Computing and Applications*, pp.33-44, 2007.
23. M. Shereshevsky, J. Crowell, B. Cukic, et al., Software Aging and Multifractality of Memory Resources, *Proc. 2003 Int'l Conf. Dependable Systems and Networks*, pp.721-730, 2003.
24. X. E. Chen, Q. Quan, Y. F. Jia, et al., A Threshold Autoregressive Model for Software Aging, *Proc. Workshop Ser-vice-Oriented System Engineering*, pp.34-40, 2006.
25. M. Grottke, L. Li, K. Vaidyanathan, et al., Analysis of Software Aging in a Web Server, *IEEE Transactions on Re-liability*, vol. 55. pp.411-420, 2006.
26. F. Machida, A. Andrzejak, R. Matias, et al., On the effectiveness of Mann-Kendall test for detection of software aging, *IEEE International Symposium on Software Reliability Engineering Workshops*, pp.269-274, 2013.
27. S. Li, Q. Yong, Software Aging Detection Based on NARX Model, *Proc. 2012 Conf. Web Information Systems and Applications*, pp.105-110, 2012.
28. G. A. Hoffmann, K. S. Trivedi, M. Malek, A Best Practice Guide to Resource Forecasting for Computing Systems, *IEEE Transactions on Reliability*, vol. 56. pp.615-628, 2007.

29. J. Araujo, R. Matos, P. Maciel, et al., Software Rejuvenation in Eucalyptus Cloud Computing Infrastructure: A Method Based on Time Series Forecasting and Multiple Thresholds, *Third Int'l Workshop Software Aging and Rejuvenation*, pp.38-43,2011.
30. V. D. Simeono, D. R. Avresky, Proactive Software Rejuvenation Based on Machine Learning Techniques, Institute for Computer Sciences, *Social Informatics and Telecommunications Engineering*, vol. 34, pp.186-200, 2010.
31. G. E. P. BOX, G. M. JENKINS, G. C. REINSEL, Time Series Analysis, *Forecasting and Control*, 2011.
32. Introduction to ARIMA: Nonseasonal Models, http://people.duke.edu/~rnau/411arim.htm
33. N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, 2000.
34. V. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, 1995.
35. J. C. Platt, Fast Training of Support Vector Machines Using Sequential Minimal Optimization, MIT Press, 1999.
36. Netcraft,http://news.netcraft.com/archives/2013/04/02/april-2013-web-server-survey.html.

.