Journal of Database Management
Volume 27 • Issue 2 • April-June 2016

Predicting Software Abnormal State by using Classification Algorithm

Yongquan Yan, School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China
Ping Guo, School of System Science, Beijing Normal University, Beijing, China

ABSTRACT

Software aging, also called smooth degradation or chronics, has been observed in a long running software application, accompanied by performance degradation, hang/crash failures or both. The key for software aging problem is how to fast and accurately detect software aging occurrence, which is a hard work due to the long delay before aging appearance. In this paper, two problems about software aging prediction are solved, which are how to accurately find proper running software system variables to represent system state and how to predict software aging state in a running software system with a minor error rate. Firstly, the authors use proposed stepwise forward selection algorithm and stepwise backward selection algorithm to find a proper subset of variables set. Secondly, a classification algorithm is used to model software aging process. Lastly, t-test with k-fold cross validation is used to compare performance of two classification algorithms. In the experiments, the authors find that their proposed method is an efficient way to forecast software aging problems in advance.

KEYWORDS

Corrected T-Test, Feature Selection, Smooth Degradation, Software Aging

1. INTRODUCTION

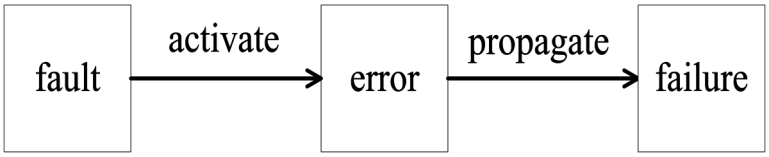
It has been reported that 9% of overall business revenues (Bixby, 2010) is affected by application performance problems. Taking the most popular web service as an example, when the web server runs in a very long time, the system cannot respond quickly for the upcoming request. Even worse, it may not respond any requests even if the workload level is modest. In generally, this performance degradation does not show an instantaneous failure. This performance degradation phenomenon may last a few days or even several weeks, if there is no any manual intervention. This phenomenon of performance degradation, unplanned software outages, or suddenly failure is called software aging (Cotroneo, Natella, Pietrantuono, & Russo, 2014), smooth degradation (Alencar, Santos, Santana, & Fernandes, 2014), or chronics (Kavulya et al., 2012). In this work, we call it as software aging.

In fact, software aging is a consequence of problems with the software itself. When a fault is activated, an error will appear as part of the internal state of the software system. When multiple errors sufficiently accumulate and the proliferation of errors reaches the service interface of the software system, the system will incur performance degradation, or even failure. Fig. 1 gives chain of software failure.

In order to counteract problems caused by software aging, Huang, Kintala, Kolettis, and Fulton (1995) proposed the technique of software rejuvenation, including occasionally ceasing software application, removing accumulated error environments and then rebooting application. However, execution of software rejuvenation can cause both direct and indirect costs during the downtime of

DOI: 10.4018/JDM.2016040103

Figure 1. Chain of software failure



the system. In order to minimize the loss caused by rejuvenation, the implementation of software rejuvenation need be executed based on the current state of the system. There are two core problems which need be solved to identify system state and execute rejuvenation.

Problem 1: How to accurately find proper system and application variables to represent system state for running software.
Problem 2: How to predict software aging state in a running software system with a minor error rate.

This paper, which gives a practice framework for forecasting software aging using a classification algorithm, focuses on these two problems. Firstly, we propose two feature selection algorithms to choose a subset of variables of operating system and application system. Secondly, a classification algorithm called support vector machine (SVM) is carefully analyzed and used to model the software aging process through the collected variables of an IIS web server that is a running commercial server. Lastly, statistical analysis is used to analyze the performance between SVM and artificial neural network (ANN).

2. RELATED WORK

Since software aging and rejuvenation have been proposed, performance degradation problems has been reported on a variety of software systems, such as web server (Rahme & Xu, 2015), database system (Cotroneo, Natella, & Pietrantuono, 2013), OLTP server, operating system (Kim, Chan, & Lee, 2014), middleware, and military system (Avritzer, Cole, & Weyuker, 2010). In order to get degradation trend of running software system, Sen's Slope Estimator was used in many researches. Zheng, Qi, Zhou, and Zhang (2014) proposed a method called Modified Cox-Stuart Test and Iterative Hodrick-Prescott Filter to overcome drawbacks of Sen's Slope Estimator, such as over simple linear assumption, sensitivity to noise. However, the method proposed by Zheng still cannot deal with drawbacks of Sen's Slope Estimator, since the proposed method is still linear method and software aging is only judged by a single variable, which is not accurate. Jia et al. (2015) analyzed the relationship between workload and available memory by using ANN. Yakhchi, Alonso, Fazeli, and Bitaraf (2015) used some machine learning algorithms to predict time to resource exhaustion and found that artificial neural network has the best forecast accuracy. However, time to resource exhaustion as a unique indicator to software aging is too simple, since it is prone to outlier and fluctuation of data.

In addition, some scholars tried to predict software aging according to software state: normal or abnormal. Magalhaes and Silva (2010) collected online bookstore data based on a TPC-W benchmark using artificial workload and memory leak. Then ANN was used to predict software state. However, it is not proper that labeling software aging state is only by a single variable response time. Meanwhile, response time can not be accurately obtained since it is affected by many factors, such as the state of network. Su, Chen, Qi, and Wu (2013) analyzed software aging problems in a HelixServer. In order to find proper features to train SVM with radial basis function kernel, they use principal component analysis (PCA) to choose features from a controlled dataset. However, PCA is limited to the correlation of variables, which means that if the correlations of variables are weak, it cannot