Journal of Computers Vol. 31 No. 5, 2020, pp. 236-248
doi:10.3966/199115992020103105018

Extreme Learning Machine with Firefly Algorithm for Abnormal Prediction

Yong-Quan Yan*

School of Statistics, Shanxi University of Finance and Economics, Taiyuan, China
yongquanyan@aliyun.com

Received 1 June 2019; Revised 10 September 2019; Accepted 21 November 2019

Abstract. Since software aging problems bring about economic losses, how to find it in advance becomes an important task for software aging treatment. Although many algorithms are proposed to forecast software aging, their performances depend on parameter tuning. In this work, an Extreme learning machine, called ELM, by using an optimal firefly algorithm, is proposed to find the optimal parameter values of ELM to make the software aging prediction more accurately. For ELM, its prediction accuracy depends on the used features and parameters. Therefore, we first use a feature selection algorithm to get a suitable feature set, and then employ the optimized firefly algorithm to get better parameter values for ELM. In the experiments, the presented method is in comparison with SVMs of four different kernels, and the results indicate the presented method gives more exact prediction results than SVMs.

Keywords: abnormity, optimization, prediction, software aging, web server

1 Introduction

Since software aging phenomenon [1] has been observed for about three decades, many researchers devoted to this area. Software aging is a running software problem caused by not released resource, the accumulated error state, round error, and so on, which may cause loss of profit. For the sake of finding software aging in advance and handle it, regression and classification algorithms were utilized to identify software aging occurrence. However, the characteristics of software aging data, such as a nonlinear feature in the sample data, unbalanced data, and small dataset, raise a great challenge for software aging disposition.

Regression methods are often used to forecast software aging by predicting resource consumption of software systems. However, software aging is a complex process accompanied by performance degradation, service delay, wrong return results, so using regression methods to find software aging may be not proper. Therefore, classification methods, for instance, support vector machines [2], also called SVMs, are used to forecast software state by a classifier. For support vector machines, it is essential to find a proper hyper-parameter set.

Evolutionary algorithms [3], as global searching methods, provide a methodology to search optimum global values for a target function. Among them, firefly algorithm [4] is a novel swarm optimization algorithm. It can adaptively update the searching radius and search the target concurrently. For software aging classification, there are so many algorithms, such as SVMs, clustering methods, artificial neural networks (ANNs), and sparse representation, to predict software aging state and improve the prediction accuracy. Nevertheless, how to balance efficiency to the aging data remains a challenge. Extreme learning machine, also called ELM, [5] is a more efficient method [6-7] than SVMs and ANNs because the random weights are employed in the hidden nodes. ELM is a method based on forward neuron network, whose major characteristic is that the node parameters of the hidden layer can be given haphazardly or artificially. Since ELM owns the advantages of high learning efficiency and strong generalization ability, it has been widely applied in all kinds of areas. Recently, like SVMs, kernel-based

* Corresponding Author

ELM was proposed and used with better results [8]. For ELM, hyper-parameters play a key role in prediction performance. For example, the number of the hidden-layer units may increase when the number of training data becomes large [9].

In this paper, to get a suitable hyper-parameter set for ELM to forecast software aging problems, an ELM method with an optimized firefly algorithm is proposed to forecast the aging state in a software system. First, a feature selection algorithm is used to find a suitable feature set. Second, the hyper-parameters of ELM with the kernel are searched by the optimized firefly algorithm. Thirdly, the optimized ELM is used to find software aging in an unseen data set. The ELM's performance is connected with the node number of the hidden-layer, parameters of the kernel function, and regularization parameters. Therefore, in this work, we need to optimize these parameters to find an optimal prediction model.

The key contribution of this work is the development of an effective hyper-parameter searching method for ELM, which can be used to enhance prediction accuracy. The proposed method implements the firefly algorithm to find the optimal hyper-parameter values of ELM.

The remaining parts are organized as follows. In Section 2, related work is illustrated and discussed. We give a detail of our proposed method in Section 3. We give some experiments and discussions in Section 4. In Section 5, we conclude this work.

2 Related Work

Since software aging phenomena had an impact on the performance of running software systems, many researchers have devoted to this area. Software aging problems can be seen as a gradual performance degradation process, which means that software aging phenomena can be found by the software aging state. Araujo et al. [10] employed four algorithms to find software aging in an IaaS-style private cloud，called Eucalyptus. In the experiment, they defined critical memory utilization (CMU) to identify whether there was a software aging problem. When the value reaches 80% of CMU, it means that software aging problems happen and performance decreases. When the value reaches 90% of CMU, it means that rejuvenation operation must be done since the quality of service cannot be accepted. Magalhaes et al. [11] collected data in an online bookstore application based on the TPC-W benchmark program simulation by artificially increasing the load and adding memory leaks. And then they used ANNs to forecast aging state of the online bookstore application. In the experiment, they found that the classification algorithm could be a better manner to find software aging state. However, the response time was used as the one and only aging parameter. And it is not accurate since software aging is a complex process. Li et al. [12] argued that software aging problems of an online system were rarely covered before, and an algorithm, windows over Local Outlier Factor, was used to analyze software aging indicator. In the experiment, they found that CPU was a good indicator compared with memory indicator for aging servers. Also, we use CPU as one of our software aging indicators. In consideration of the defect of time to resource exhaustion (low effectiveness and accuracy), Li et al. [13] utilized a hybrid method to discover software aging problems in the system of data cache and streaming media, and they found their proposed method had better performance on the prediction precision and recall. Liang et al. [14] discussed the software aging problem in the nuclear power control system and provided a life cycle of software aging in the system. However, this work does not provide a practice method to find software aging. Ficco et al. [15] gave a study to inspect software aging manifestation of an Apache Storm and the results showed that Apache Storm experienced abnormal state when it worked for a long time although there is a garbage collection mechanism. Compared to the above method, in this work, a feature selection algorithm is applied to cut down the number of variables used, since the introduced invalid variables can make the prediction effect worse.

Based on the forward selection method, Laradji et al. [16] found that feature selection method has a significant impact on the classification effect in the NASA dataset (PC2, PC4, and MC1) and they found that random forest algorithm of integrated greedy forward selection had a better performance than SVM. Wahono [17] used some machine learning methods to forecast the status of a working software system: normal state or aging state. Alonso et al. [18] gathered data of web server by using artificial load, and then used six machine learning methods to forecast the status of the Web server: exception, warning, and normal. In the experiment, they found that the random forest error rate was 1%. Gulenko et al. [19] attempted to find abnormal states in virtual machines through supervised and unsupervised algorithms.

However, the prediction performances of these proposed methods depend on the hyper-parameter selection, which is not discussed in the above studies. In this work, we propose an optimization method by using the firefly algorithm to find better hyper-parameter values to predict software aging occurrences.

3 Method

In this section, a feature selection algorithm is introduced first, and then the firefly algorithm and ELM are shown, respectively. In the last, our proposed method is described in detail.

3.1 Feature Selection

Since the parameters of running software system are numerous, if those unrelated parameters enter into the ELM, the prediction performance drops rapidly. At the same time, Matias et al. [20] pointed out that using only a single aging parameter, such as available memory, swap partition, etc., might cause much software aging false positives, so it is necessary to utilize feature selection method to get a series of appropriate aging parameters. The selection of aging parameters is also called feature selection or dimensionality reduction, which is to get the smallest subset from the input parameter set.

The feature selection method is: for a selected data set, a learning algorithm is used to get an optimizing feature subset by using the optimizing criterion. In this work, we use a stepwise forward selection algorithm, which can be described as:

(1) We need to select the model which is used to fit the target value by aging parameters.

(2) Compute the score statistic for each candidate feature and also compute its significance.

(3) Select the least significant feature. If the significance of this feature is less than the probability value of the specified feature which will be added, then go to step 4, otherwise end the feature selection.

(4) Add the selected feature to the model. If the selected feature by the model is consistent with the feature of the last selection, the feature selection procedure is over. Otherwise, the statistic of the new model need to be calculated, and the calculation process turns to step 5.

(5) Compute the Wald statistic for each variable and then compute the corresponding significance.

(6) Select the most significant feature of step 4. If the significance of the selected feature is less than the removed probability value, go back to step 2. Otherwise, if the feature selection result is consistent with the last selected feature result, end the feature selection procedure, otherwise go to step 7.

(7) Remove the most significant feature from the current model, and compute the parameters of the model after removing the feature, and then go back to step 5.

3.2 Firefly Algorithm

The firefly algorithm (FA) is a novel evolutionary method, which is an intelligent optimization method to simulate the luminescent behavior of fireflies in nature based on group search strategy. The general thought of the firefly method is to make firefly with low luminescence move to another firefly with a high luminescence to get an optimum solution.

Since FA is relatively simple and also has good global optimization ability, it has been used in many areas, such as dynamic path planning, image processing, and economic scheduling.

Each individual in the firefly algorithm is considered as a firefly, and a low-intensity firefly is attracted to a high-intensity firefly. For each firefly, the degree of attraction and brightness of other individuals to themselves vary according to distance changes.

For simplicity, the firefly algorithm has the following assumption:

(1) Assume that all fireflies are of the same gender and are attracted to each other

(2) The attraction is only related to luminous intensity and distance. Fireflies with strong illuminating light will attract fireflies with weak surrounding light, but as the distance increases, the attracting degree will gradually decrease, and fireflies with strong radiance will move randomly

(3) The intensity of the illumination is determined by the objective function and is proportional to the specified function within the defined area

The search process is referred to two kinds of parameters of fireflies: brightness and mutual attraction of the fireflies. The bright fireflies attract those weak fireflies to move toward them. The brightest fireflies represent an optimal result of optimal function. The brighter the firefly is, the higher the

attraction of the surrounding fireflies is.

FA is made up of three parts

(1) Fluorescence brightness

$$I = I_{ini} e^{-\Upsilon r_{i,j}}, \tag{1}$$

where I_{ini} indicates the brightness of the brightest firefly in connection with the objective function value; Υ is a light absorption coefficient, which decreases as distance increases and media absorption, and can be set as a constant; $r_{i,j}$ is the distance between two fireflies.

(2) Mutual attraction degree

$$\beta(r) = \beta_{ini} e^{-\Upsilon r_{i,j}}, \tag{2}$$

where β_{ini} represents the maximum attractiveness, i.e., the degree of attraction at the light source.

(3) Optimal target iteration

$$x^{(i)}(t+1) = x^{(i)}(t) + \beta(x^{(j)}(t) - x^{(i)}(t)) + \alpha(rand - 1/2), \tag{3}$$

where $x^{(i)}(t)$ represents the position of firefly i, $x^{(j)}(t)$ represents the position of firefly j, α represents a step length factor, and rand represents a random factor with uniform distribution on [0, 1].

Since the firefly algorithm has the following disadvantages: slower convergence speed, poor computational efficiency, and premature convergence, firefly algorithm can be improved by a chaotic idea. In this work, by using the property of the ergodicity and non-repetition of chaos, chaotic optimization can give global search capability with higher speed [21].

The chaotic firefly method is

(1) Initialization. Generate the firefly parameters

(2) Computing the fluorescent light

(3) Weight updating

(4) Updating the firefly position

First, let t be 0, the position of the firefly i need be mapped between 0 to 1 to a chaotic variable:

$$cx_k^{(i)}(t) = \frac{x_k^{(i)}(t) - x_{\min,k}}{x_{\max,k} - x_{\min,k}}, \tag{4}$$

where k is between 1 to n, $cx_k^{(i)}(t)$ is a chaotic variable, the variable $x_{\max,k}$ is the ceiling of the k-th dimension, and the variable $x_{\min,k}$ is the floor of the k-th dimension.

Second, we use Chebyshev map to calculate the next values of the chaotic variable:

$$cx_k^{(i)}(t+1) = \cos(t \cos^{-1}(cx_k^{(i)}(t))), \tag{5}$$

Third, calculate the position information:

$$x_k^{(i)}(t+1) = x_k^{(i)}(\min) + cx_k^{(i)}(t+1)(x_k^{(i)}(\max) - x_k^{(i)}(\min)), \tag{6}$$

(5) Calculating the fluorescent light of every firefly after updating location

(6) Finding 20% with the highest fitness in the population after updating location and searching with a chaotic local method

(7) Calculating the fluorescent light of every firefly

(8) Judging whether termination condition is satisfied or not. If so, end calculation and export the globally optimal solution. Otherwise, continue

(9) Shrinking the search area as follows

$$\begin{aligned} x_k^{(i)}(\min) &= \max\{x_k^{(i)}(\max), x_k^{(i)}(\max light) - rand(x_k^{(i)}(\max) - x_k^{(i)}(\min))\} \\ x_k^{(i)}(\max) &= \min\{x_k^{(i)}(\max), x_k^{(i)}(\max light) - rand(x_k^{(i)}(\max) - x_k^{(i)}(\min))\}. \end{aligned} \tag{7}$$

where *rand* is between 0 and 1, $x_k^{(i)}(\max light)$ is the value of the k-th dimension of the brightest firefly. After shrinking, randomly generate fireflies and go back to step 3.

3.3 ELM

Although ELM was originally designed to supervised learning problems, its applications are extended to almost all machine learning areas, including clustering and feature learning.

A data set can be $\{(x^{(1)},y^{(1)}),.....(x^{(i)},y^{(i)}),.....,(x^{(m)},y^{(m)})\}$, where $x^{(i)}=(x_1^{(i)},.....,x_k^{(i)})^T \in R^k$, $y^{(i)}$ is a vector with n dimensional. The structure of ELM is made up of m dimensionality input nodes, and L nodes of the hidden layer.

The ELM's output is

$$t_L = \sum_{i=1}^{L} \beta_i h(W_i \cdot x + b_i) = \sum_{i=1}^{L} \beta_i g_i(x) = g(x)\beta. \tag{8}$$

where x is an input vector of ELM, β is a weight of output, W_i is a weight of input, b_i is a bias of hidden node I, t_L is a target value, $g(x)$ and $h(W_i \cdot x + b_i)$ are called as the feature map or the activation function, whose purpose is to map the data of the input layer from its original space to the feature space.

And also it can be rewritten as:

$$T = G(x)\beta, \tag{9}$$

where G is a matrix from hidden layer to output layer

$$G=\begin{bmatrix} g(x^{(1)}) \\ \vdots \\ g(x^{(m)}) \end{bmatrix} = \begin{bmatrix} g_1(x^{(1)}) & \cdots & g_k(x^{(1)}) \\ \vdots & \ddots & \vdots \\ g_1(x^{(m)}) & \cdots & g_k(x^{(m)}) \end{bmatrix}. \tag{10}$$

ELM uses L1 loss function to reduce the training error

$$\min \left\| G(x)\beta\text{-}T \right\|, \tag{11}$$

After using L1 regularization, equation (11) is rewritten as

$$\min \frac{1}{2}\left\|\beta\right\|^2 + \frac{C}{2}\left\|G(x)\beta\text{-}T\right\|^2, \tag{12}$$

By using KKT theory, the original optimization problem can be expressed as a dual optimization problem. And by calculating partial derivation, the weight is

$$\beta=(GG^T + \frac{1}{C})^{-1}G^T G, \tag{13}$$

So the ELM's output can be:

$$t_L = g(x)\beta = g(x)H^T(GG^T + \frac{1}{C})^{-1}T, \tag{14}$$

Also, by using the kernel method, we can rewrite the ELM's output as

$$\begin{aligned} f(x) = t_L &= g(x)\beta = g(x)G^T(GG^T + \frac{1}{C})^{-1}T \\ &= \begin{bmatrix} K(x,x^{(1)}) \\ \vdots \\ K(x,x^{(m)}) \end{bmatrix} (GG^T + \frac{1}{C})^{-1}T. \end{aligned} \tag{15}$$

In the paper, a radial base function is used as kernel function

$$K(x, x^{(i)}) = \exp(-\frac{\left\| x - x^{(i)} \right\|^2}{2\sigma^2}). \quad \textbf{(16)}$$

3.4 Proposed Method for Optimizing ELM

To get a more suitable model to forecast software aging problems, hidden nodes, regularization coefficient, and kernel are three key parameters for ELM. If the hidden-layer is assigned, the matrix of the hidden-layer can be determined. Once the kernel of RBF is selected as kernel function, the data mapping will be decided by σ.

In this work, a firefly optimization method is used to seek a better solution to ELM. The structure of presented method is shown in Fig. 1. In Fig. 1, firefly algorithm is used to choose the hidden unit, regularization coefficient, and kernel width. The value of objective target function is chosen with the brightest firefly by the firefly algorithm and the values of the best-optimized parameters of the objective target function can be seen as the position of the brightest firefly. The procedure can be described as:

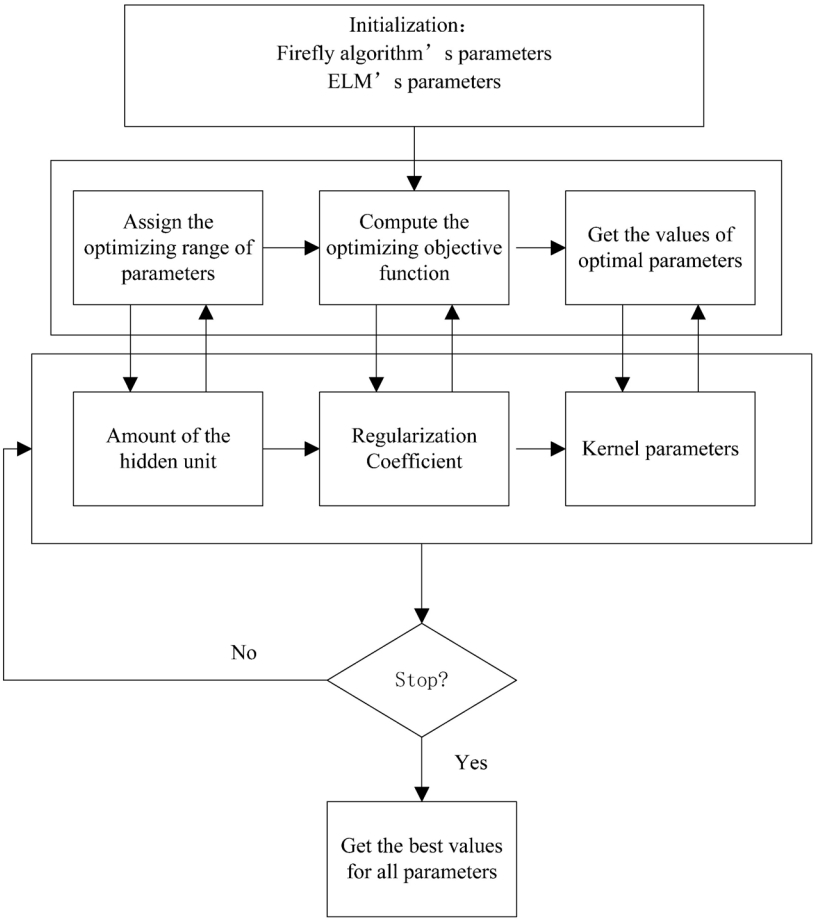

Fig. 1. The proposed method for ELM optimization

(1) Initialization
The initial values of the hidden unit, regularization coefficient, and kernel width are assigned.
(2) Optimization of the hidden unit.
Optimize the amount of the hidden unit of ELM in a specified range.
(3) Optimization of the regularization coefficient
Optimize the regularization coefficient of ELM in a specified range.
(4) Kernel function optimization
Optimize the kernel function of ELM in a specified range.
(5) If the brightest firefly is found, end the searching process. Otherwise, go back to step 2.

4 Experiments

The experimental environment is based on a real commercial web service, which is a key application in cloud computing. The web service is made up of web server and database management server, and the running applications are hospital applications, such as online booking system, online querying system. The data-collection time spans about half a year. In these collected data sets, two of them are selected as the used data sets for the experiment. The number of the first data, which is used to train and validate our proposed method, is 17,099. The number of the second data, which was utilized to measure the predictive ability in the unknown situation, is 13,715.

Since the software aging problems are affected by multiple factors, the peaks of different parameters appear differently. It is difficult to determine the state of the software through a single aging parameter, so we try to mark software status: normal state or aging state, by the ratio of the collected data. For example, the states of the first dataset are labeled on the basis of the different proportions: the last 10% of the dataset is marked as the aging state dataset, and all previous data is used as normal status data; the last 15% is used as aging status data, and all previous data is used as normal status data; the last 20% is used as aging status data, and all previous data is used as normal status data. After using feature selection algorithm, the features used are Usage Peak, Processor Time and so on.

And we also plot these features in Fig. 2 to Fig. 10.

In Fig. 2, usage peak attains a peak value around the 9,000-th minute and remains this trend until server restarts. And also, usage peak is not high during the operation of the server.

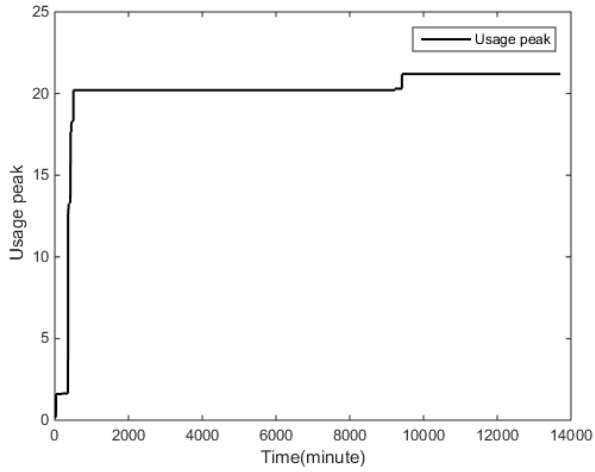

Fig. 2. The values of Usage peak

In Fig. 3, process time fluctuates before the 10,000-th minute and remains high level after the 12,000-th minute. And process time is high in the last running of the server.

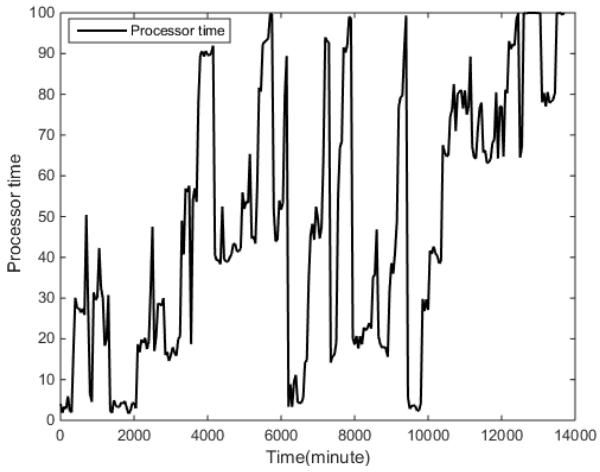

Fig. 3. The values of Processor time

In Fig. 4, we see that there is a downward trend from beginning to end and there are many large fluctuations.

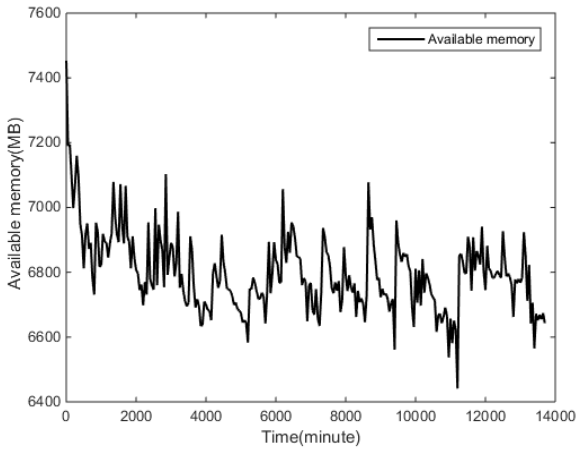

Fig. 4. The values of Available Memory

In Fig. 5, context switches presents a download trend on the whole. And also the fluctuations remain until server goes down.

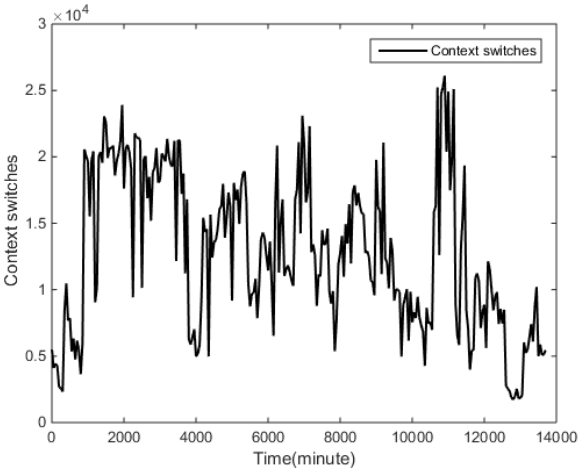

Fig. 5. The values of Context switches

In Fig. 6, we see that the number of errors that the server could not process reaches 300 around the 12,000-th minute.

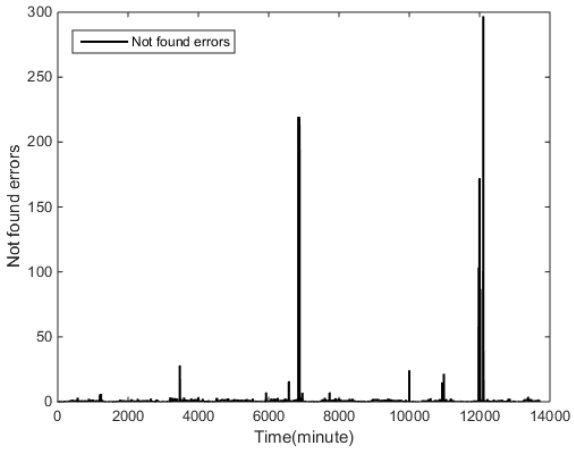

Fig. 6. The values of Not found errors

In Fig. 7, the number of http requests reaches a peak after the 9,000-th minute.

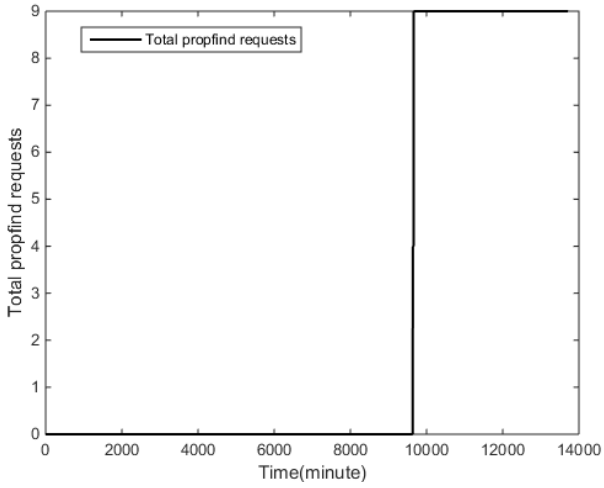

Fig. 7. The values of Total propfind requests

In Fig. 8, the request number reaches a peak when the server crashes after that moment.

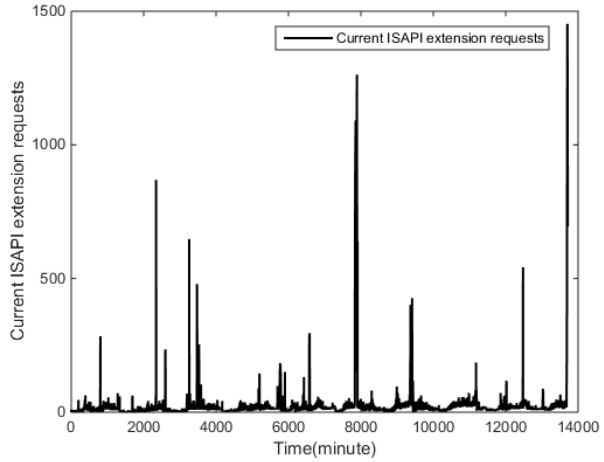

Fig. 8. The values of Current ISAPI extension requests

In Fig. 9, we see that the values of heap memory in the application level have an upward trend.

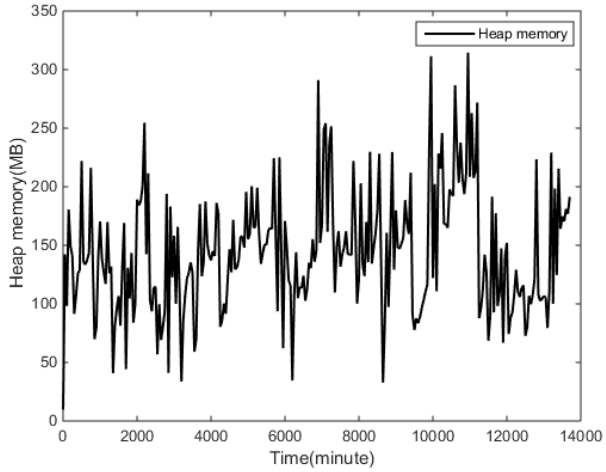

Fig. 9. The values of Heap memory

In Fig. 10, Bytes received by web server reveals a trend of fluctuations.

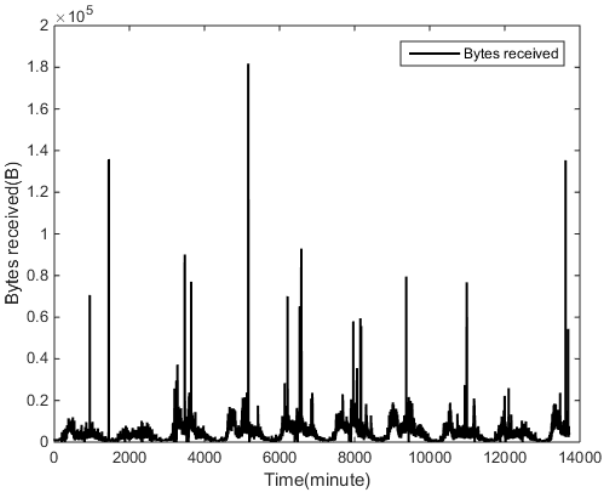

Fig. 10. The values of Bytes received

The parameters of the initialization process for ELM with firefly algorithm are given in Table 1.

Table 1. Range of initial parameter values

Parameters	
Firefly algorithm	ELM
Iterations = [300, 1000]	$L = [1, 100]$
Population size = [80, 100]	$C = [0.01, 100]$
Light absorbance = 1	$\sigma = (0, 1]$
Attractiveness = 1	
Step size = [0.2, 0.5]	

In Table 1, we set the range of hyper-parameter values to discover an optimal solution of the firefly method and ELM. And our proposed method's results are compared with SVMs [22], which contain four different kernels: linear, polynomial, sigmoid, and Radial Basis Function (RBF), are described in Table 2 to Table 4.

In Table 2, the consequences are shown with two states, normal state and aging state, and two stages, validation stage and testing stage. SVM with RBF kernel has the highest prediction accuracy than the other three methods in the validation stage. And SVM with sigmoid kernel doesn't correctly forecast aging states in column 3. In the testing stage, our method can correctly predict aging states with 1,301, and SVM with RBF kernel doesn't predict aging states completely although it has the best prediction result in the validation stage. In a word, our presented algorithm is better than SVMs with sigmoid kernel and RBF kernel, and has a similar result with SVM with linear kernel in the validation phase.

Table 2. Output results of 10%

	Normal state (validation stage)	Aging state (validation stage)	Normal state (testing stage)	Aging state (testing stage)
Normal state (our method)	4,662	9	12,084	72
Aging state (our method)	2	510	258	1,301
Normal state (linear kernel)	4,660	6	10,453	202
Aging state (linear kernel)	4	513	1,889	1,171
Normal state (polynomial kernel)	4,663	2	11,983	1,219
Aging state (polynomial kernel)	1	517	359	154
Normal state (sigmoid kernel)	4,661	519	11,983	1,219
Aging state (sigmoid kernel)	3	0	359	154
Normal state (RBF kernel)	4,661	6	11,761	1,373
Aging state (RBF kernel)	3	513	581	0

In Table 3, the results contain two parts: the validation and testing stage. In the validation stage, SVM with sigmoid kernel doesn't identify the state of software aging, and our proposed method can correctly forecast s the state of software aging with 770. In the testing stage, SVM with sigmoid kernel also does not have a correct result for aging state prediction. Moreover, SVM with RBF kernel cannot give a right answer for aging state prediction. As same for the validation stage, our proposed method has the highest accuracy for software state prediction. In a word, our presented method owns higher accuracy than SVMs with four kernels. And SVM with sigmoid kernel owns a poor manifestation since it cannot distinguish normal state and aging state no matter in the validation stage or testing stage.

Table 3. Output results of 15%

	Normal state (validation stage)	Aging state (validation stage)	Normal state (testing stage)	Aging state (testing stage)
Normal state (our method)	4,398	11	11,423	128
Aging state (our method)	4	770	233	1,931
Normal state (linear kernel)	4,384	25	11,297	1,902
Aging state (linear kernel)	18	756	359	157
Normal state (polynomial kernel)	4,384	21	11,230	1,880
Aging state (polynomial kernel)	18	760	426	179
Normal state (sigmoid kernel)	4,402	781	11,656	2,059
Aging state (sigmoid kernel)	0	0	0	0
Normal state (RBF kernel)	4,382	24	11,652	2,059
Aging state (RBF kernel)	20	757	4	0

In Table 4, SVM with polynomial kernel has the best prediction result for aging state and our method has the second place in the validation stage. In the testing stage, SVM with polynomial kernel only predicts the state of software aging rightly with 157, but our method gives the best prediction for the aging state with 2682. And we can see that SVM with RBF kernel doesn't forecast the state of software aging correctly although it could foreast the state of software aging in the validation stage. In short, our proposed method has a better performance than SVMs with RBF kernel, sigmoid kernel, and linear kernel.

Table 4. Output results of 20%

	Normal state (validation stage)	Aging state (validation stage)	Normal state (testing stage)	Aging state (testing stage)
Normal state (our method)	4,110	37	10,798	61
Aging state (our method)	35	1,001	174	2,682
Normal state (linear kernel)	4,100	58	10,613	2,589
Aging state (linear kernel)	45	980	359	154
Normal state (polynomial kernel)	4,112	25	10,600	2,586
Aging state (polynomial kernel)	33	1,013	372	157
Normal state (sigmoid kernel)	4,132	1,038	10,609	2,589
Aging state (sigmoid kernel)	13	0	363	154
Normal state (RBF kernel)	4,098	66	10,972	2,743
Aging state (RBF kernel)	47	972	0	0

In the above three experiments, although our proposed method has similar results as SVM with polynomial kernel in the validation stage, it has the best performance in the testing phase no matter for 10%, 15%, and 20%, which means that our presented method may be more suitable for software aging prediction in web service.

5 Conclusion

Although ELM is an effect method to do regression analysis and classification prediction in several fields, its prediction performance depends on the selection of hyper-parameter. In this work, an ELM with optimized firefly method is presented for the software aging forecast. The proposed method contains

three parts: feature selection, parameter optimization, and model prediction. First, a feature selection method is applied to get a proper feature set. Second, an ELM method is used to train software state data. Third, a firefly algorithm is used to tune the parameters of ELM. In the experiments, our proposed method outperforms than other methods with higher accuracy and better aging state discovery. For example, in Table 4, our method can predict software aging state with 2682 compared to SVMs with polynomial kernel with 157. Moreover, there are some things need to be done in the future: how to find a proper strategy to decrease search time, how to make the algorithm run concurrently, and how to optimize the firefly algorithm.

Acknowledgements

The authors want to appreciate the reviewers for their detailed reviews to help us ameliorate the quality of the paper. This work is supported by the China Statistical Science Research Project (2018LY78), Program for the Philosophy and Social Sciences Research of Higher Learning Institutions of Shanxi (201803082), and Research project 1331 of the 13th five-year plan for education science in Shanxi Province (ZX-18031).

References

[1] Y. Huang, C. Kintala, N. Kolettis, N.D. Fulton, Software rejuvenation: Analysis, module and applications, in: Proc. Twenty-Fifth International Symposium on Fault-Tolerant Computing, 1995.

[2] A. Andrzejak, L. Silva, Using machine learning for non-intrusive modeling and prediction of software aging, in: Proc. 2008 NOMS 2008-2008 IEEE Network Operations and Management Symposium, 2008.

[3] A. Trivedi, D. Srinivasan, K. Sanyal, A. Ghosh, A survey of multiobjective evolutionary algorithms based on decomposition, IEEE Transactions on Evolutionary Computation 21(2016) 440-462.

[4] A.H. Gandomi, X.-S. Yang, S. Talatahari, A.H. Alavi, Firefly algorithm with chaos, Communications in Nonlinear Science and Numerical Simulation 18(2013) 89-98.

[5] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Transactions on Systems, Man, and Cybernetics 42 (2011) 513-529.

[6] G.-B. Huang, X. Ding, H. Zhou, Optimization method based extreme learning machine for classification, Neurocomputing 74(2010) 155-163.

[7] L. Duan, H. Zhong, J. Miao, Z. Yang, W. Ma, X. Zhang, A voting optimized strategy based on ELM for improving classification of motor imagery BCI data, Cognitive Computation 6(2014) 477-483.

[8] W. Li, C. Chen, H. Su, Q. DU, Local binary patterns and extreme learning machine for hyperspectral imagery classification, IEEE Transactions on Geoscience and Remote Sensing 53(2015) 3681-3693.

[9] J. Li, Y. Qi, L. Cai, A hybrid approach for predicting aging-related failures of software systems, in: Proc. 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE), 2018.

[10] J. Araujo, R. Matos, P. Maciel, F. Vieira, R. Matias, K.S. Trivedi, Software rejuvenation in eucalyptus cloud computing infrastructure: A method based on time series forecasting and multiple thresholds, in: Proc. 2011 IEEE Third International Workshop on Software Aging and Rejuvenation, 2011.

[11] J.P. Magalhães, L.M. Silva, Prediction of performance anomalies in web-applications based-on software aging scenarios, in: Proc. 2010 IEEE Second International Workshop on Software Aging and Rejuvenation, 2010.

[12] L. Li, H. He, Q. Wang, H. Zhang, Aberrant software-aging server detection and analysis using sliding window over LOF, in: Proc. 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS), 2017.

[13] J. Li, X. Zhao, Y. Li, Q. Du, B. Xi, J. Hu, Classification of hyperspectral imagery using a new fully convolutional neural network, IEEE Geoscience and Remote Sensing Letters 15(2018) 292-296.

[14] H. Liang, P. Gu, J. Tang, W. Chen, F. Gao, Discussion on software aging management of nuclear power plant safety digital control system, SpringerPlus 5(2016) 2092.

[15] M. Ficco, R. Pietrantuono, S. Russo, Aging-related performance anomalies in the apache storm stream processing system, Future Generation Computer Systems 86(2018) 975-994.

[16] I.H. Laradji, M. Alshayeb, L. Ghouti, Software defect prediction using ensemble learning on selected features, Information and Software Technology 58(2015) 388-402.

[17] R.S. Wahono, Integrasi Bagging dan Greedy Forward Selection pada Prediksi Cacat Software dengan Menggunakan Naïve Bayes, Journal of Software Engineering 1(2015) 101-108.

[18] J. Alonso, L. Belanche, D.R. Avresky, Predicting software anomalies using machine learning techniques, in: Proc. 2011 IEEE 10th International Symposium on Network Computing and Applications, 2011.

[19] A. Gulenko, M. Wallschläger, F. Schmidt, O. Kao, F. Liu, A system architecture for real-time anomaly detection in large-scale NFV systems, Procedia Computer Science 94(2016) 491-496.

[20] R. Matias, B.E. Costa, A. Macedo, Monitoring memory-related software aging: An exploratory study, in: Proc. 2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops, 2012.

[21] L. Dos Santos Coelho, V.C. Mariani, Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization, Expert Systems with Applications 34(2008) 1905-1913.

[22] Y. Yan, P. Guo, Predicting software abnormal state by using classification algorithm, Journal of Database Management 27(2016) 49-65.