Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 7, July 2019, pp. 1955-1964
DOI: 10.23940/ijpe.19.07.p23.19551964

Cloud Computing Resource Load Forecasting based on Bat Algorithm Optimized SVM

Yuxia Li*

Beijing Union University, Beijing, 100025, China

Abstract

For the problem of resource load forecasting in cloud computing, the optimized bat algorithm is combined with SVM for forecasting. Firstly, the bat algorithm adopts the reverse learning strategy for population initialization, and secondly, the weighting factor in the particle swarm optimization is used for individual optimization. Finally, the individual is selected using the Gaussian mutation method. Two important parameters in the SVM are optimized using the improved algorithm. In the simulation experiment, the SVM is optimized by the particle swarm optimization and compared with the genetic algorithm optimizing SVM, and a better forecasting effect is obtained.

Keywords: cloud computing; resource load; forecasting; bat algorithm

(Submitted on March 13, 2019; Revised on May 15, 2019; Accepted on June 15, 2019)

1. Introduction

How to effectively forecast the resource load in cloud computing is a research hotspot in cloud computing resources, because it is related to the hardware configuration of cloud computing nodes and the effect of real-time monitoring. It provides decision-making for cloud computing resource optimization [1-2]. The cloud computing resource load has certain uncertainties and nonlinear characteristics [3]. Reference [4] proposed distributed load forecast model mining based on hybrid gene expression programming and cloud computing (DLFMM-HGEPCloud). Reference [5] proposed cloud computing using a PSO-based weighted support vector machine, and the experiment results indicated that the proposed algorithm is superior to the other four prediction algorithms in terms of prediction accuracy and efficiency. Reference [6] proposed a technique that not only makes the data center more energy efficient but also balances the workload more efficiently. Reference [7] proposed a heuristic-based algorithm called the greedy-based load balance (GBLB) algorithm. Simulation results showed that the proposed GBLB algorithm can reduce the number of active servers and achieve the best load balancing level at the cost of a few more migrations. Reference [8] proposed a load balancing algorithm based on the processing capacities of virtual machines (VMs) in cloud computing. Simulation results showed that the proposed algorithm can reduce the number of active servers. In reference [9], the improved GWO algorithm was used to optimize the support vector machine parameters, thereby improving the forecasting effect of cloud computing. Reference [10] proposed a hybrid wavelet packet transform and optimization of the chaotic whale optimization (CSCWOA) algorithm to optimize the short-term cloud computing resource load forecasting method of multi-layer perceptron neural networks (MLP). Reference [11] proposed an improved ant colony algorithm (LBIACO) to improve the load balance of virtual machines, and simulation experiments demonstrated the method's advantages in task execution cost and execution time. Reference [12] proposed a correlation time series model to study resource forecasting, providing a feasible way for cloud computing on-demand use and resource elasticity construction. Reference [13] proposed a load forecasting method based on the parallel random forest algorithm in big data, and the experimental results showed that the scheme improves resource utilization while ensuring the performance of the host.

* Corresponding author.
E-mail address: bjlyx1970@sina.com

2. Related Knowledge

2.1. Bat Algorithm

The bat algorithm (BA) is a new meta-heuristic search algorithm proposed by Yang in 2010 to simulate the bat echolocation behavior. The principle of BA is to use the individual bat as the solution of a search space. Adjusting the individual bat's loudness and pulse rate, follow the optimal bat in the current solution space, so that the entire population produces a search process from disorder to order. The bat algorithm has strong global search ability and convergence speed, but it is also similar to other global optimization algorithms in that it is easy to fall into the local optimal solution and its solution precision is not high.

First, a N dimensions search space needs to be determined. The frequency range of the bat is $[f_{\min}, f_{\max}]$. The range of the corresponding wavelength is $[\lambda_{\min}, \lambda_{\max}]$. The bat loudness is A, the pulse frequency is r, and the update formula of the first bat at the position l_i^t, the velocity v_i^t, and the moment t is as follows:

$$f_i = f_{\min} + (f_{\max} - f_{\min}) \times \beta \tag{1}$$

$$v_i^t = v_i^{t-1} + (x_i^t - x^*) \times f_i \tag{2}$$

$$l_i^t = l_i^{t-1} + v_i^t \tag{3}$$

Where f_i is the bat frequency, $\beta \in [0,1]$ are the random variables that are uniformly subject to uniform distribution. x^* represents the optimal position obtained after each iteration. For the adjustment of f, when the local search is performed, an optimal solution is generated from the current local search, and each bat randomly generates a new position.

$$l_{new} = l_{old} + \alpha \overline{A^t} \tag{4}$$

Where the value of α is between [-1,1], $\overline{A^t}$ represents the average loudness of all bats at the t moment, while the velocity and position are updated with iteration. The update step in the bat algorithm is similar to that of the standard PSO algorithm. When it is easy to find the prey, the loudness drops and the pulse frequency rises.

$$A_i^{t+1} = \alpha A_i^t \tag{5}$$

$$r_i^{t+1} = r_i^0 \times [1 - \exp(-\kappa t)] \tag{6}$$

In the above formula, assume the values of α and κ are at [0,1], and after testing, the values of α and κ are approximated as 0.925.

$$A^t \to 0 \quad r_i^t \to 0 \tag{7}$$

2.2. SVM

SVM refers to support vector machine, which is a common method of discrimination. In the field of machine learning, it is a supervised learning model that is commonly used for pattern recognition, classification, and regression analysis. Through SVM, a certain load forecasting model can be established in a short period of time. The nonlinear mapping function $\varphi(x)$ is used to map the load sequence $x_1, x_2, \cdots, x_n$ in a short time as a sample to the high-dimensional feature space, where linear regression is performed in high dimension.

The regression function of SVM in a high-dimensional feature space is:

$$f(x) = \omega \times \varphi(x) + b \tag{8}$$

Where ω is the weight vector and b is the offset vector. According to the principle of risk minimization, Equation (8) is transformed into the expression of the following optimization problem:

$$\begin{aligned} &\min J = \frac{1}{2}\| \omega \|^2 + C\sum_{i=1}^{n}(\xi_i^* + \xi_i) \\ &s.t\begin{cases} y_i - \omega \times \varphi(x) - b \le \varepsilon + \xi_i \\ \omega \times \varphi(x) + b - y_i \le \varepsilon + \xi_i^* \\ \xi_i^*, \xi_i \ge 0, i = 1,2,\cdots,n \end{cases} \end{aligned} \tag{9}$$

Where $\| \omega \|$ is the term related to the complexity of function f, ε is the sparse insensitive loss, ξ_i^* and ξ_i represent the relaxation factor, and C represents the penalty factor. The Lagrangian multiplier is introduced, thus turning the problem into a convex quadratic optimization problem:

$$L(w,b,\xi,\alpha,\gamma) = \frac{1}{2}\| w \|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}a_i(y_i(w^T x_i + b) - 1 + \xi_i) - \sum_{i=1}^{n}\gamma\xi_i \tag{10}$$

Where α_i and α_i^* represent the Lagrangian multiplier and γ_i represents the loss factor. Convert Equation (10) to the dual form, as follows:

$$\begin{aligned} &\omega(\alpha,\alpha^*) = -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)\times(\alpha_j - \alpha_j^*)(\varphi(x_i),\varphi(x_j)) + \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)y_i - \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) \\ &s.t\begin{cases} \omega = \sum_{i,j}^{n}(\alpha_i - \alpha_i^*)x_i \\ \sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0 \\ 0 \le \alpha_i, \alpha_i^* \le C \end{cases} \end{aligned} \tag{11}$$

Therefore, the regression function of SVM is:

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)(\varphi(x_i),\varphi(x)) + b \tag{12}$$

To further simplify the use of kernel function $K(x_i, x)$ to replace $(\varphi(x_i),\varphi(x))$, the regression function of SVM is:

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x_i, x) + b \tag{13}$$

The radial basis kernel function is used as the SVM kernel function, and the definition is as follows:

$$K(x_i, x) = \exp\left(-\frac{\| x_i - x_j \|^2}{2\sigma^2}\right) \tag{14}$$

In the formula, σ is the radial basis kernel function width parameter.

It is found from Equations (13) and (14) that the purpose of establishing a cloud computing load forecasting model is to find the two parameters C and σ of the optimal support vector and thus obtain the corresponding value of the function $f(x)$. Through a large number of experiments, it is proven that C and σ determine the advantages and disadvantages of

the models established by the support vector machine to an extent. Therefore, further optimization of the two parameters C and σ relies on intelligent algorithms for processing.

3. Improved Bat Algorithm

3.1. Introduce a Reverse Learning Strategy

According to the principle of the bat algorithm, individual bats in the population continuously update their velocity and position to ensure they are close to the bat individual with the best fitness value. In the process of approaching, there may be a possibility of missing the global optimum. Therefore, it is necessary to further improve the optimization ability of the algorithm. Reverse learning [14] is a method that can expand the scope of population search and increase the diversities of populations. By adopting a reverse learning strategy for different bat individuals, the probability of getting close to the optimal solution is greatly improved. Because elite individuals' adapting degree value will be better than that of ordinary individuals, elites with better fitness can be close to the optimal solution by applying reverse learning.

Set N as the inverse solution $x'_{best} = (x'_1, x'_2, \cdots, x'_n)$ of the elite individual $x_{best} = (x_1, x_2, \cdots, x_n)$ in the group G of the search space, so the elite's inverse solution is defined as $x'_i = k \times (da_i + db_i) - x_i$, and $k \in [0,1]$ is a random number that obeys a uniform distribution. $x_i \in [\, a_i \,, b_i \,]$, $[\, da_i \,, db_i \,]$ is the dynamic boundary in the dimension i in the group G, $da_i = \min_{1 \le i \le G}(x_i)$, and $db_i = \min_{1 \le i \le G}(x_i)$. Because k is a random value, the individuals after inversion have different positions, which can effectively improve the mining ability of the algorithm.

3.2. Introducing Inertia Weighting Factor for Individual Optimization

The bat algorithm is a swarm intelligence optimization algorithm. In the process of solving, the individual adjusts its position and follows the optimal individual through the improvement of loudness and pulse rate. Therefore, this algorithm has the advantage of fast convergence, but it has the disadvantage of falling easily into low solution accuracy and a local optimum. In Equation (2), each bat moves with a fixed step size, so the speed change is relatively simple. When the step value is small, the algorithm falls into the local optimum prematurely; when the step size is higher, the bat flies away from the optimal solution, leading to oscillations in the later stage of the algorithm. This situation is very similar to the particle swarm algorithm, so the inertia weight factor is added to the bat algorithm for improvement. This paper sets according to the weighting factor in the particle swarm optimization algorithm.

$$\omega = \begin{cases} \omega_{\min} + \dfrac{(\omega_{\max} - \omega_{\min})(f - f_{\min})}{f_{avg} - f_{\min}}, & \text{if } f \le f_{avg} \\ \omega_{\max}, & \text{otherwise} \end{cases} \tag{15}$$

Where f is the fitness value corresponding to the current bat individual, f_{avg} represents the average function fitness value, $f_{\min}$ represents the minimum function fitness value, and $\omega_{\max}$ and $\omega_{\min}$ represent the maximum value and the minimum value of ω, respectively. When the distance between individual bats is relatively scattered, the inertia factor is relatively small. When the fitness function values of bat individuals tend to be consistent, the inertia factor will become larger. In the solution process, when the fitness function value corresponding to the bat individual is lower than the average fitness function value, a larger inertia factor can be obtained, thereby accelerating the bat individual's optimization speed. When the bat individual corresponding fitness degree function value is greater than the average fitness value, a smaller inertia factor can be obtained, which is beneficial to the balance of the entire algorithm. Therefore, Equation (2) is transformed as follows:

$$v_i^t = v_i^{t-1} + \omega \times (x_i^t - x^*) \times f_i \tag{16}$$

3.3. Gaussian Mutation Difference Algorithm

The Gaussian mutation differential algorithm is a global search algorithm. Its main idea is to use real-coded methods to take individuals in a population as a variable in D dimensional space. Set $x_i(k)$ as the i^{th} individual in the k-order evolution, where $i \le NP$, k is less than the maximum evolution number, and $x_i^{\min} \le x_i(k) \le x_i^{\max}$, that is,

$x_i(k) = (x_{i1}(k), x_{i2}(k), \cdots, x_{in}(k))$. The individual evolutionary overexpression is shown in Equations (17) and (18), where λ is generally 0.8. $rand(k)$ is a random function in k dimensional space, where $\rho 1$, $\rho 2$, $\rho 3 \in [1, NP]$:

$$v_{ij}(\alpha) = \begin{cases} x_{\rho 1 j} + \lambda \times (x_{\rho 2 j} - x_{\rho 3 j}), & j = rand(k) \\ x_{ij}, & \text{otherwise} \end{cases} \tag{17}$$

$$x_i(a+1) = \begin{cases} v_i(\alpha + 1), & i \le rand(k) \\ x_i(a), & \text{otherwise} \end{cases} \tag{18}$$

It is found from Equations (17) and (18) that the candidate solution depends on a certain relationship among the selected three vectors $\rho 1$, $\rho 2$, and $\rho 3$. When the value $(x_{\rho 2 j} - x_{\rho 3 j})$ tends to 0, the candidate solution generated by Equation (17) tends to be $x_{\rho 1}$. This will make the evolutionary process stagnate at a later stage, so the individual diversity will decline, causing the algorithm to fall into a local optimum. To avoid this, Gaussian difference variability is performed according to Equation (19):

$$v_i^t = \begin{cases} G(v_i^{t-1} + (1-p) \times (x_{ij}^t - x_{best}^*) f_i, \left| x_{ij} - x_{bestj} \right|), & x_{ij} \ne x_{bestj} \\ G(v_i^{t-1} + (1-p) \times (x_{ij}^t - x_{best}^*) f_i, \left| x_{i\max} - x_{i\min} \right|), & x_{ij} = x_{bestj} \end{cases} \tag{19}$$

In Equation (19), p represents a uniformly distributed random number, $v_i^{t-1} + (1-p) \times (x_{ij}^t - x_{best}^*) f_i$ represents a value between the current individual bat i and the optimal value, and $x_{ij} - x_{bestj}$ represents the distance of two individuals in the j dimensional space. If the current solution is not the optimal solution, then the current individual i approaches the optimal value. If the current solution is the optimal solution, the solution is taken as the center point position for difference mutation. In this way, the individual's evolutionary behavior can be corrected in time to ensure the diversity of the bat population and avoid falling into a local optimum.

4. Improved Bat Algorithm Optimizes SVM Resource Scheduling Algorithm

4.1. Selection of Kernel Function

The SVM kernel function is an important part of SVM theory and determines the running result to an extent. This paper uses the Gaussian radial basis function as the kernel function of SVM. It is the most widely used among all kernel functions. Regardless of the situation, it can demonstrate the best performance. At the same time, it has fewer parameters and is easier to improve than other kernel functions. The final results of SVM are jointly affected by the parameter g of the RBF function and the penalty factor C. The penalty factor is a compromise between the proportion of complex sample misclassification and the complexity of the algorithm. It adjusts the confidence range of the learning machine and the proportion of empirical risk in the determined feature subspace and has a great impact on the generalization ability of SVM [15]. This paper uses the strong global optimization ability of ICS to search for the optimal combination of SVM parameters and feature vectors in order to avoid the occurrence of local optimal solutions. This paper refers to this SVM as a support vector machine based on the improved cuckoo algorithm.

4.2. Algorithm Flow

Step 1 Select the data flow under cloud computing in a period to make sample data, randomly divide the sample data into training samples and test samples, and normalize the samples;

Step 2 Set the relevant parameters in the bat algorithm, the maximum number of evolution iterations to 200, the maximum number of populations to 20, the range of C and σ to (0,100), and λ to 0.8;

Step 3 Within the range of values of C and σ, (C, σ) is randomly generated. Use the data in step 1 for training and testing, and make each (C, σ) correspond to an individual of the bat algorithm;

Step 4 The bat individual performs reverse learning according to Section 2.1. Integrate the bat individuals and their corresponding reverse learning into a new set, and arrange them according to the fitness value from high to low. The individuals are selected according to a certain ratio;

Step 5 Optimize the speed of the bat individual using Equations (15) and (16);

Step 6 Adjust the position of the bat using Equation (3);

Step 7 Optimize the individual selection of bats using Equations (17) to (19);

Step 8 Determine whether the algorithm termination condition is satisfied. If the termination condition is not met, skip to step 5; otherwise, terminate the condition and find the optimal bat individual, that is, C and σ represent the best individual.

5. Simulation Experiment

This article builds a hardware platform consisting of a Core i3 processor, 4GDDR3, and hard disk with 1T capacity. The software platform is the Linux operating system, and data processing is performed through the Cloudsim cloud computing simulation platform. In order to better illustrate the effect of the proposed algorithm in optimizing SVM, the particle swarm optimization SVM model (PSO-SVM) and the optimized SVM model by genetic algorithm (GA-SVM) are selected as comparison algorithms. The learning factor and social factor in the PSO algorithm are set to 1, the crossover probability and mutation probability in the genetic algorithm are 0.5, the population of the three algorithms is 100, the maximum number of iterations is 1000, C is [1,50], σ is [1,50], λ is 0.8, the maximum number of iterations is 1000, and the dimensions are from 10 to 50.

5.1. Data Source and Processing

This paper selects the CPU load in Google and Clarknet network load data as the research data of the cloud computing resource load in the experiment. It extracts the CPU load in Google and the network load in Clarknet as the experimental objects. The time series recorded in the three sets are all 1s, and 600,000 historical data (time span of about one week) are compiled as training data to predict the load data for the next 300s. The following equation is adopted:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{20}$$

Where x represents the data traffic in cloud computing, and $\max(x)$ and $\min(x)$ represent the maximum and minimum values, respectively. Use the (average absolute error) MAE value as the criterion for judging the flow effect forecasting, and set y_t as the actual load data, $\hat{y}_t$ as the forecasting data, and T as the time series:

$$MAE = \frac{1}{T}\sum_{t=1}^{t=T} | y_t - \hat{y}_t | \tag{21}$$

5.2. Analysis of the Algorithm's Performance

The SVM based on the intelligent algorithm can effectively improve the accuracy of prediction. Therefore, the performance of the intelligent algorithm is key to determining the accuracy of SVM prediction. The algorithm of this paper is compared with PSO-SVM and GA-SVM. The sphere function and Schwefel function are selected as comparison functions. The sphere function is mainly used to verify the convergence performance of the algorithm, and the Schwefel function is mainly used to verify the search effect of the algorithm, as shown in Table 1.

Table 1 shows a comparison of the performance of the three algorithms in different dimensions of basic test functions. The results show that the performance of BA-SVM algorithm is better than PSO-SVM and GA-SVM algorithm in both test functions, indicating that the algorithm has better effect on convergence and global optimal solution. Figure 1 depicts the three algorithms in the convergence of the algorithm. It is found from the figure that the algorithm has reached convergence when the number of iterations is 60, which is much lower than the PSO-SVM algorithm at 80 and the GA-SVM algorithm at 100, which shows that the performance of the algorithm is much better than the other two. The algorithm is mainly because

the reverse learning is used for the initialization operation, which improves the efficiency of the initial solution of the algorithm, and the bat individual's search ability is improved by the inertia weight factor. Figure 2 shows the global search ability under the three algorithms. It is found from the figure that the curve of the algorithm is gradually stable after the initial oscillation, and the curve decreases when the number of iterations is half. This shows that the global convergence ability of the algorithm is gradually stable, mainly because the improvement of the individual avoids the local optimization through Gaussian mutation, and improves the efficiency of the global optimal solution of the algorithm.

Table 1. Comparison results of the three algorithms

Function	Dimension	Parameter	ACO-SVM	PSO-SVM	BA-SVM
Sphere function	10	Optimal value	1.941E-8	1.94E-9	1.71E-10
		Worst value	8.16E-10	3.41E-15	9.17E-20
	20	Optimal value	12.73E-14	17.78E-15	10.32E-16
		Worst value	15.83E-16	15.13E-17	12.73E-17
	50	Optimal value	23.18E-20	22.18E-21	20.41E-25
		Worst value	35.17E-21	33.28E-21	30.69E-22
Schwefel function	10	Optimal value	1.41E+02	1.47E+02	0.92E+02
		Worst value	3.91E+06	8.41E+06	3.48E+04
	20	Optimal value	21.18E+09	20.78E+09	16.83E+09
		Worst value	28.17E+10	26.72E+10	22.83E+09
	50	Optimal value	34.78E+13	30.18E+12	26.83E+11
		Worst value	46.18E+15	43.79E+14	40.98E+13

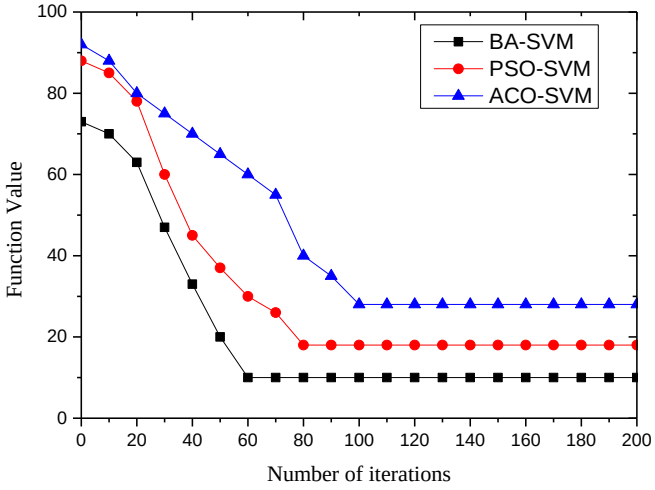

Figure 1. Comparison of sphere function

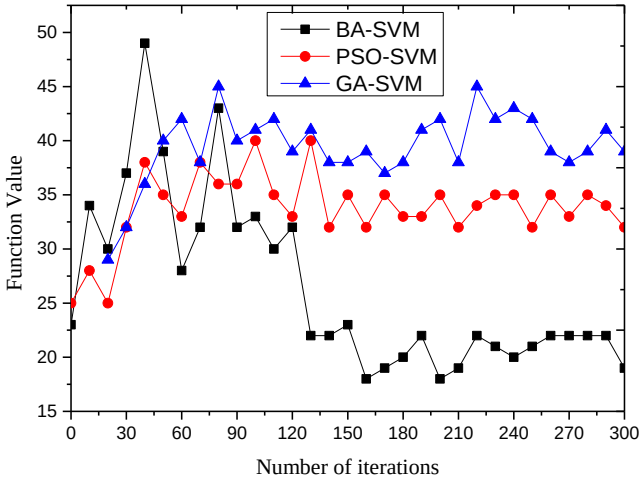

Figure 2. Comparison of Schwefel function

5.3. Experimental Analysis

Figure 3 shows the load forecast of the CPU in Google in this paper. Figure 4 shows the network load in the algorithm Clarknet. Figure 5 shows the CS-SVM, PSO-SVM, and GA-SVM algorithms in the CPU. The negative MAE value, and Figure 6 shows the network load MAE values of the CS-SVM, PSO-SVM, and GA-SVM algorithms in Clarknet.

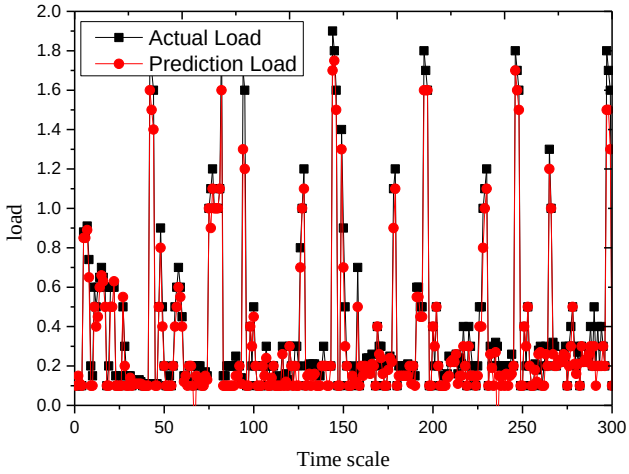

Figure 3. CPU load forecasting results in Google

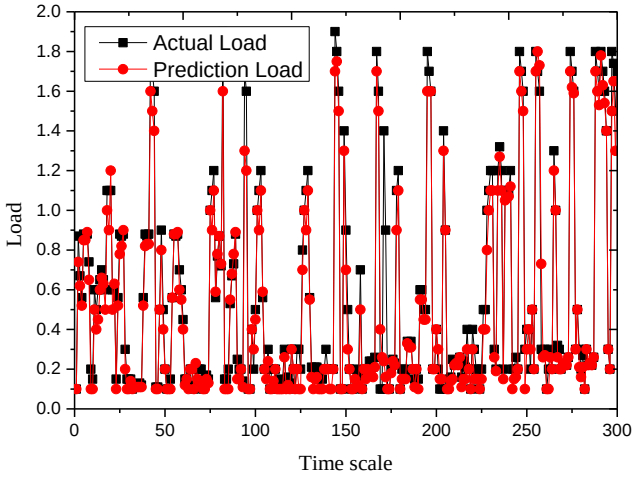

Figure 4. Network load forecasting in Clarknet

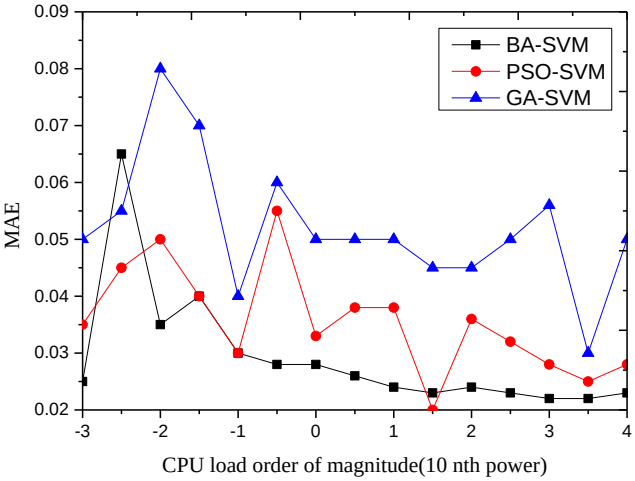

Figure 5. Comparison of MAE values of CPU load under three algorithms

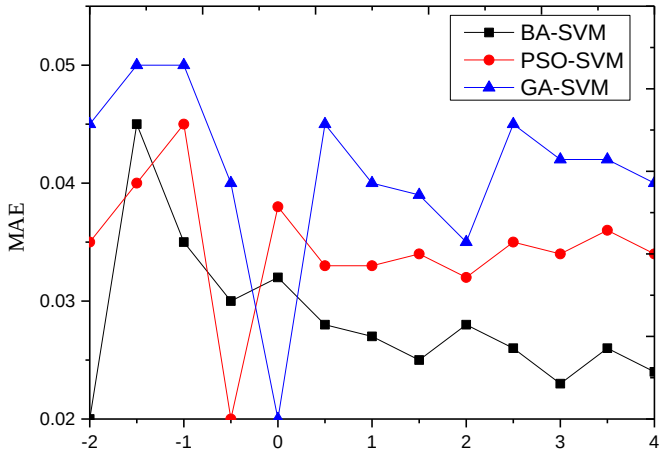

Figure 6. Comparison of MAE values of network loads under three algorithms

As shown in Figures 3 and 4, the proposed algorithm can approach the actual resource load in the forecasting of cloud computing, which indicates that the algorithm can effectively forecast the value of cloud computing. This is mainly because the optimization of the bat algorithm achieves good effects. The reverse learning simplifies the generation of the initial solution. The inertia weight is applied to the individual to make the bat individual's exploration ability stronger, and the algorithm is prevented from falling into a local optimum. The MAE values of CPU Load of these three algorithms in Figure 5 are different and increase as the CPU load increases, but the value of the algorithm is always lower than that of PSO-SVM and GA-SVM, which shows that the algorithm in this paper is superior. The curve rises smoothly, which means that the overall algorithm is relatively stable. This is mainly because the Gaussian mutation method is used in the global search to ensure the global optimal solution, so the CPU load is reduced to some extent and the stability of the algorithm is enhanced. The MAE values of the three algorithms in Figure 6 increase as the network load increases, and the value of the algorithm in this paper is lower than that of the other two algorithms. However, the curve oscillating amplitude of these three algorithms is relatively large, which indicates that the network load has certain influence on these three algorithms. The curve oscillation of the algorithm in this paper is very high, mainly because the complexity of the algorithm in the initialization and individual selection process is increased. The network load consumption of the algorithm is thus higher. From the above results, the algorithm in this paper demonstrates certain advantages in cloud computing load forecasting.

6. Conclusions

To address the problem that current cloud computing resource load prediction is inadequate, an improved bat algorithm is proposed to optimize the SVM load forecasting model. The forecasting related parameters in the SVM are forecasted by the improved bat algorithm, and a better result is obtained. In the simulation experiment, compared with other intelligent algorithms, significant improvements are seen. As the next step, the impact of virtual machine loads on the forecasting accuracy of cloud computing resources needs to be considered.

References

1. D. R. Mittal and M. Sharma, "A Comparative Study of Different Load Balancing Algorithms in Cloud Computing," *Mody University International Journal of Computing and Engineering Research*, Vol. 1, No. 1, pp. 51-55, 2017
2. A. Tripathi and S. Singh, "A Literature Review on Algorithms for the Load Balancing in Cloud Computing Environments and Their Future Trends," *Mathematical and Computer Modelling*, Vol. 21, No. 1, pp. 64-73, 2017
3. C. Reiss, A. Tumanov, G. R. Ganger, and R. H. Katz, "A Towards Understanding Heterogeneous Clouds at Scale: Google Trace Analysis," in *Proceedings of the 3rd ACM Symp on Cloud Computing*, 2012
4. S. Deng, C. G. Yuan, L. C. Yang, and L. P. Zhang, "Distributed Electricity Load Forecasting Model Mining based on Hybrid Gene Expression Programming and Cloud Computing," *Pattern Recognition Letters*, No. 109, pp. 72-80, 2018
5. W. Zhong, Y. Zhuang, J. Sun, and J. J. Gu, "A Load Prediction Model for Cloud Computing using PSO-based Weighted Wavelet Support Vector Machine," *Applied Intelligence*, pp. 1-12, 2018
6. S. B. Shaw, C. Kumar, and A. K. Singh, "Use of Time-Series based Forecasting Technique for Balancing Load and Reducing Consumption of Energy in a Cloud Data Center," in *Proceedings of 2017 International Conference on Intelligent Computing and Control*, pp. 1-6, 2017
7. H. Y. Xu and B. Yang, "Energy-Aware Resource Management in Cloud Computing Considering Load Balance," *Journal of Information Science & Engineering*, Vol. 33, No. 1, pp. 1-16, 2017

8. R. Kaur and N. S. Ghumman, "A Load Balancing Algorithm based on Processing Capacities of VMs in Cloud Computing," *Big Data Analytics*, pp. 63-69, Springer, Singapore, 2018
9. D. Y. Xu and S. Ding, "Research on Improved GWO-Optimized SVM-based Short-Term Load Prediction for Cloud Computing," *Computer Engineering and Applications*, Vol. 53, No. 7, pp. 68-73, 2017
10. J. Q. Xie, Y. J. Liu, and S. Li, "Application of Improved Whale Algorithm in Load Forecasting of Cloud Computing Resources," *Computer Engineering and Applications*, Vol. 54, No. 13, pp. 73-77, 2018
11. Q. B. Nie, "Optimization Management of Task Scheduling for Cloud Resource Load Balance," *Computer Engineering and Design*, Vol. 38, No. 1, pp. 18-21, 2017
12. D. G. Li, L. Wu, and L. Li, "Research of Load Forecasting and Elastic Resources Scheduling of Openstack Platform based on Time Series," *Journal of Chongqing University of Posts and Telecommunications*, Vol. 28, No. 4, pp. 560-566, 2016
13. D. Wang and Z. Sun, "Big Data Analysis and Parallel Load Forecasting of Electric Power User Side," *Proceedings of the CSEE*, Vol. 35, No. 3, pp. 527-537, 2015
14. S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama, "Opposition-based Differential Evolution," *IEEE Transactions on Evolutionary Computation*, Vol. 12, No. 1, pp. 64-79, 2008
15. Q. J. Liu, G. M. Chen, and X. F. Liu, "Genetic Algorithm based SVM Parameter Composition Optimizaiton," *Computer Applications and Software*, Vol. 29, No. 4, pp. 94-96, 2012

Yuxia Li is an associate professor at Beijing Union University. She received her master's degree from Beijing Institute of Technology. Her research focuses on cloud computing.