Research and Application of BP Algorithm Based on Genetic Algorithm in System Performance Bottleneck Analysis

Hongman Wang¹, *, Peidian Li², *

¹Institute of Network Technology, Engineering Research Center of Information Network, Beijing University of Posts and Telecommunications, Ministry of Education, Beijing, China
²Institute of Network Technology, Beijing University of Posts and Telecommunications, Beijing, China

*Corresponding author e-mail: lipeidian@qq.com, *wanghm@bupt.edu.cn

Abstract. System performance bottleneck analysis is an essential issue in application performance management. With the development of the neural network, BP neural network has been applied to analyze the performance bottleneck of the system, but because BP algorithm is easy to fall into local optimum, the results obtained by this algorithm lack certain accuracy. In the system performance bottleneck, response time and performance counter can effectively help to analyze the performance bottleneck. In this paper, the server performance counter collected related data as sample data, and the improved BP algorithm based on adaptive genetic algorithm is used to model. This method can solve the shortcomings of traditional BP algorithm very well. The experimental results show that the improved BP algorithm based on genetic algorithm is superior to the conventional BP algorithm in the accuracy of system performance bottleneck analysis, and the learning process takes less time on large datasets.

1. Introduction

System performance bottleneck analysis has always been an important research topic in the field of application performance management. The traditional system performance bottleneck analysis is to monitor the performance counter of the host system. In the literature [1] [2] [3], the author uses software technology to track the indicators of the system, displaying the indicators data in a graphical way, which is convenient for the tester to analyze the system performance. Manual analysis of performance counters makes system performance bottleneck analysis very complicated, and testers need to spend a lot of time to analyze the data recorded by each counter.

With the development of the neural network, the neural network algorithm is used to classify and predict [4]. BP (Back propagation) neural network is one of the most important models in artificial neural networks. It is widely used and has very good distributed storage and fault tolerance. It is very suitable for solving non-linear problems. BP neural network algorithm has been used in software defect prediction [5] [6], but because BP neural network algorithm is a gradient-based steepest descent method, it has the shortcomings of slow convergence in the learning process and easy to fall into local minimum in network training.
The genetic algorithm is a new optimization algorithm, which simulates the genetic mechanism and evolutionary process of organisms in nature and forms an adaptive global search for the optimal solution. It has good parallelism, robustness, and global optimality. It can not only reduce the risk limited to the optimal solution but also optimize the initial weight and threshold of the BP network, which further improves the stability of the BP network. In this paper, the genetic algorithm is used to optimize the traditional BP algorithm.

In performance testing, response time, performance counter can effectively help to analyze the system performance bottleneck [7]. Response time mainly consists of four parts: application software processing time, network transmission time, server processing time and database processing time. Each part of the response time is affected by some indicators. The performance indicators of application software include processor time, user state time, page error speed, etc. The indicators affecting network performance are network throughput, network delay, packet loss rate and so on. The indicators affecting server performance are processor performance, memory performance, disk performance, and network interface performance. The indicators affecting database performance include page reading rate, cache hit rate, transaction rate and so on.

Taking the performance bottleneck on the server as an example, this paper illustrates that using the key performance index of the server [8], the resource of the system performance bottleneck can be found by modeling and analyzing the BP algorithm optimized by the adaptive genetic algorithm, and the validity of the optimized algorithm is verified by experiments.

2. Overview of BP Neural Network and Genetic Algorithms

2.1. BP neural network

BP neural network is a multi-layer forward network with one-way propagation, which consists of three levels: the input layer, hidden layer, and the output layer. The input layer receives external signals, the hidden layer maps and converts the input signals, and the output layer finally outputs the simulation results of the network. There are several implied layers, generally using a three-tier architecture, as shown in Figure 1. BP neural network has a complete learning mechanism, which imitates the response process of human brain neurons to external stimulus signals. By establishing a multi-layer perceptron model, using the forward propagation of signals and the feedback regulation mechanism of errors, the neural network model for learning non-linear information is constructed.

![Figure 1. Three-layer Network Architecture of Neural Network.](image)

The learning process of the BP algorithm includes two stages: the forward propagation of signals and the reverse propagation of errors.

Forward propagation of signals: external signals are processed by the input layer and transmitted to the output layer through the hidden layer. Output signals are generated in the output layer. The weights of the network remain unchanged in the process of forwarding propagation. If the output layer does not get the desired output, it will propagate backward.

Back Propagation of Error Signal: The output signal starts from the output terminal and propagates forward one by one in a certain way. The connection weights of each unit layer are adjusted by error
feedback. The weights of neurons are continuously revised by repeated forward propagation and error back regulation. When the actual output approaches the expected output, the learning can be stopped.

2.2. Adaptive genetic algorithm
Genetic algorithm (GA) is an iterative algorithm for global search, which does not depend on gradient information. Different from single-point search, the genetic algorithm has strong adaptability. It is suitable for large-scale, highly non-linear optimization of discontinuous and multi-peak functions. Because of its many search trajectories, it is easy to parallelize, thus further improving the efficiency of the algorithm.

The evolutionary process of genetic algorithm begins with the initial population representing the possible solution set of the problem, evaluates the fitness of each individual according to the predetermined fitness function, and then uses the basic operators of selection, crossover and mutation to transform the individual, preserving the individual with high fitness into the next evolutionary cycle until the individual closest to the expected value comes out.

The crossover probability and mutation probability of general genetic algorithm are fixed, which quickly leads to stagnation of evolution after entering an optimal local state, leading to the premature phenomenon. For this reason, Srinivas M et al. proposed an adaptive genetic algorithm [9] [10]. The crossover probability \( P_c \) and mutation probability \( P_m \) changed adaptively with the fitness of chromosomes. For chromosomes with lower fitness than the average fitness, larger crossover and mutation probability was used to speed up their evolution, and for chromosomes with higher fitness than the average fitness, smaller probability of crossover and mutation, try to retain excellent chromosomes. The adaptive formulas of crossover rate and mutation rate are as follows:

\[
P_c = \begin{cases} 
P_{c1} = \frac{(P_{c1} - P_{c2})(f' - f_{avg})}{f_{max} - f_{avg}}, & f' \geq f_{avg} \\
P_{c1}, & f' < f_{avg} 
\end{cases} \tag{1}
\]

\[
P_m = \begin{cases} 
P_{m1} = \frac{(P_{m1} - P_{m2})(f' - f_{avg})}{f_{max} - f_{avg}}, & f' \geq f_{avg} \\
P_{m1}, & f' < f_{avg} 
\end{cases} \tag{2}
\]

\( P_{c1} \) and \( P_{c2} \) are the crossover probabilities of the first and second generations at the time of initialization; \( P_{m1} \) and \( P_{m2} \) are the mutation probabilities of the first and second generations at the time of initialization; \( f_{avg} \) is the average fitness of the current population; \( f_{max} \) is the greater fitness of the two chromosomes that need to be crossed, and \( f \) is the fitness of the chromosomes that need to be mutated at present.

3. BP Neural Network Based on Genetic Algorithms

3.1. Algorithm model overview
The basic idea of BP neural network algorithm based on genetic algorithm optimization [11] [12] is to combine BP neural network algorithm with the genetic algorithm. When the convergence speed of BP algorithm training network is slow, the threshold and weight of each hidden layer node of BP neural network are used as input information of genetic algorithm. Then, they are encoded to generate chromosomes, and the selection operator, crossover operator and mutation operator of the genetic algorithm are used to generate new offspring as the initial value of the BP algorithm. The BP algorithm is continued to train the network, which repeatedly runs until the error accuracy required by the problem is achieved.
From BP neural network algorithm based on the genetic algorithm, its method is that genetic algorithm searches the solution space of target information extensively. BP algorithm locates when genetic algorithm searches for a better network form and finds the optimal solution of the problem demand.

The process of optimization of BP neural network by genetic algorithm includes three parts: the determination of the connection structure of BP neural network, the weight and threshold of BP neural network optimized by genetic algorithm, and the prediction of BP neural network, as shown in Figure 2. The structure of BP neural network is determined by inputting predictive data, and the layers and nodes of BP neural network are further determined. The length of individual coding of the genetic algorithm is determined by the number of parameters needed by BP neural network. The error of BP neural network training is used to optimize the fitness value. After many operations, the optimal weights and thresholds are obtained. Finally, the genetic algorithm assigns the optimal weights and thresholds to the corresponding BP neural network, and the optimized BP neural network is used for prediction.

Figure 2. BP Neural Network Based on Genetic Algorithms.

The BP neural network flow based on genetic algorithm optimization is as follows.

1) The weights and thresholds of BP neural network are cascaded in sequence, i.e., the weights of the input layer and hidden layer, hidden layer and output layer, input layer and hidden layer, hidden layer and output layer, and N chromosomes are generated randomly.

2) Taking the mean square error function as the fitness function, the fitness of each chromosome is further calculated, and whether it meets the optimization criteria, if it meets step 4, is judged.

3) Select individuals that meet the fitness requirements, and apply replication, crossover, and mutation to produce new individuals.

4) Check whether the new individual meets the fitness requirements of the best individual. If the new individual meets the criteria of the next step and is not satisfied, it will return to step 2.

5) The optimal individuals are segmented sequentially as the weights and thresholds of BP neural network.
6) BP neural network carries out forward propagation, calculates global error, adjusts network parameters (weights and thresholds), repeats learning training, and ends network training when the required accuracy or the number of times to reach the upper learning limit is reached.

3.2. Coding Design of Chromosomes
In the traditional genetic algorithm, binary coding is generally used. When binary coding processes more independent variables, it will lead to the shortcomings of larger search space and lower search efficiency. Also, there are quantization errors in the process of conversion from binary to decimal, which affect the search accuracy. In the process of system bottleneck analysis, the data collected by performance counter is regarded as an independent variable. Because of the large number of independent variables and the need to combine with the neural network, real value coding is adopted. This can effectively improve the speed and accuracy of operation, and avoid additional problems in coding.

Considering a neural network whose input node is i, hidden node is j and output node is k, the initial weight threshold of BP neural network will consist of four matrices: the initial weight matrix IW from input layer to hidden layer, the weight matrix LW from hidden layer to output layer, the threshold matrix B1 of hidden layer, and the threshold matrix B2 of output layer. All the weight thresholds together constitute a chromosome.

3.3. Design of fitness function
The fitness function is the most important part of the whole genetic algorithm. In this paper, the fitness function uses the sum of squares of network errors as follows:

$$RMSN = \sqrt{\sum_{r=1}^{N} (y_r - \hat{y}_r)^2}$$  \hspace{1cm} (3)

Among them, $y_r$ is the observation value, $N$ is the sample length, $\hat{y}_r$ is the output value of the input sample through the hidden layer node and the output layer node, and the output of the sample through the hidden node is as follows:

$$Z_i = f(\sum_j w_{ij}x_i - \theta_i)$$  \hspace{1cm} (4)

The output value of the hidden layer node through the output layer node is:

$$y_r = f(\sum_j T_{jk}Z_i - \delta_k)$$  \hspace{1cm} (5)

The transfer functions of hidden layer nodes and output layer nodes are respectively:

$$f(x) = \frac{1}{1-e^{-x}}$$  \hspace{1cm} (6)

$$f(x) = x$$  \hspace{1cm} (7)

After substituting the output value of the hidden layer into the output formula of output layer nodes, $y_{- R}$ is calculated, and the sum of square error of the network is calculated.

3.4. Neural network computation
After calculating by genetic algorithm, the most suitable individuals are selected, and the chromosome coding is assigned to BP network as the initial weight threshold. BPLM algorithm is used for local optimization by global optimization of the genetic algorithm. Use formula:
\[ x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T e \]  \hspace{1cm} (8)

Among them, \( \mu \) is the weight of the neural network, \( J \) is the Jacobian matrix, the matrix element is the first derivative of the network error with the weight, \( e \) is the error vector of the net, and \( \mu \) is the scalar of the control learning process.

4. Implementation and verification

4.1. Implementation

In the previous chapter, we described the neural network model based on the genetic algorithm. In this section, we describe the implementation of the algorithm.

The classical three-layer network is adopted in the neural network model. There are six nodes in the input layer, which represent the six main factors affecting the system performance: CPU load, disk reading speed, disk writing speed, network receiving speed, network sending speed and memory utilization. Hidden layer nodes are derived from empirical formulas:

\[ L_h \leq \sqrt{P(O + 3)} + 1 \]  \hspace{1cm} (9)

Among them, \( P \) is the number of nodes in the input layer, \( O \) is the number of nodes in the output layer and \( L \) is the upper limit of nodes in the hidden layer. In this paper, the number of hidden nodes is 5. The number of output nodes is 1, which corresponds to the system performance bottleneck.

4.2. Verification

The experimental data are derived from pressure measurements of multiple services. The experimental data are shown in Table 1.

| Column names | Training file size/KB | Number of learning samples | Experimental file size/KB | Sample size |
|--------------|-----------------------|----------------------------|--------------------------|-------------|
| PC1          | 47                    | 323                        | 42                       | 310         |
| PC2          | 732                   | 9597                       | 587                      | 7782        |
| PC3          | 27                    | 200                        | 21                       | 194         |
| PC4          | 171                   | 1586                       | 90                       | 745         |
| PC5          | 145                   | 1127                       | 139                      | 1077        |

With the same experimental data, the traditional BP algorithm model and the improved BP algorithm model based on genetic algorithm are used to model, and the results of the two models in predicting the performance bottleneck of the system are analyzed and compared. The accuracy comparison of the algorithms is shown in Table 2.

| Column names | Ac_BP | Pc_BP | Ac_improved BP | Pc_improved BP |
|--------------|-------|-------|----------------|----------------|
| PC1          | 74.51 | 80.95 | 79.49          | 88.10          |
| PC2          | 73.32 | 78.95 | 75.29          | 66.99          |
| PC3          | 85.04 | 83.33 | 90.24          | 72.22          |
| PC4          | 87.86 | 81.26 | 93.77          | 81.25          |
| PC5          | 86.64 | 77.61 | 90.73          | 71.13          |

By comparing the data in the table, it can be seen that the accuracy of the improved BP neural network model based on genetic algorithm is better than that based on traditional BP algorithm, and it can avoid the optimal local solution very well.
By comparing the accuracy of the two algorithms, the efficiency of the two models is also studied. Table 3 shows the time comparison of the learning process of the two models.

Table 3. The time required for the learning process

| Column names | Number of learning samples | Traditional BP algorithm/ms | Improved BP algorithm/ms |
|--------------|-----------------------------|-----------------------------|--------------------------|
| PC1          | 323                         | 29812                       | 766624                   |
| PC2          | 9597                        | 1064862                     | 220944                   |
| PC3          | 200                         | 26151                       | 936987                   |
| PC4          | 1586                        | 210589                      | 117097                   |
| PC5          | 1127                        | 137316                      | 1135785                  |

The experimental results show that when there are fewer data sets, the traditional BP algorithm can achieve higher efficiency when there are fewer data sets because the improved genetic algorithm needs to spend a certain amount of time to evolve the legacy algorithm to calculate the initial weights. When there are many data sets, the learning time based on traditional BP algorithm is longer than that based on improved genetic algorithm, mainly because of the optimization of the genetic algorithm, it can approach the preset precision faster.

5. Conclusion

In this paper, the method of system performance bottleneck analysis based on improved BP algorithm is studied, and the feasibility of the algorithm in system performance bottleneck analysis is verified. The accuracy and efficiency of the algorithm are compared with traditional BP algorithm. The traditional BP algorithm is avoided to fall into local position by genetic algorithm. The optimal condition can achieve better processing effect, and in the case of large data sets, learning time is less than the traditional BP network. However, due to the introduction of the genetic algorithm, in the case of small data sets, it takes more time to learn.

Acknowledgments

This work is supported by the Fundamental Research Funds of Beijing University of Posts and Telecommunications (No.2018XKJC07).

References

[1] Brewer, E.A., et al., “PROTEUS: A HIGH-PERFORMANCE PARALLEL-ARCHITECTURE SIMULATOR,” Acm Sigmetrics Performance Evaluation Review, vol.20, no.1, pp.247-248, 1991.
[2] Goldberg, A.J., and Hennessy J.L., “Mtool: an integrated system for performance debugging shared memory multiprocessor applications,” IEEE Transactions on Parallel and Distributed Systems, vol.4, no.1, pp.28-40, 1993.
[3] Zhao Li, Wang Hongman, and Yang Fangchun, “JUnit Based Intelligent Network Automatic Test System,” Computer Engineering, vol.31, no.2, pp.59-60, 2005.
[4] Gruen R, and Kubota T., “A neural network approach to system performance analysis,” in Proceedings IEEE. IEEE, SoutheastCon, 2002, pp.349-354.
[5] Arar, et al., “Software defect prediction using cost-sensitive neural network”, Applied Soft Computing, vol.33, pp.263-277, 2015.
[6] Wang Hongman, et al., "Finding Main Causes of Elevator Accidents via Multi-Dimensional Association Rule in Edge Computing Environment," China Communications, vol.14, pp.39-47, Nov. 2017.
[7] Jiang Changhua, Zhu Min, and Chen Youguang, “Web-based Application Stress Testing,” Computer Applications, 2003.
[8] Duan Nian, Detailed Explanation of Software Performance Testing Process and Case Analysis, Beijing: Tsinghua University Press, 2006.
[9] Lin Mingyu, et al., “Self-adaptive Genetic Algorithm Based on Evolvability,” Computer Engineering, vol.36, no. 20, pp.173-175, 2010.

[10] Srinivas M, and Patnaink L M., “Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms,” IEEE Transactions on Systems, Man and Cybernetics, vol. 24, no. 4, pp. 656-667, 1994.

[11] Li Weizhao, et al., “Artificial neural network based on a genetic algorithm,” Computer Engineering and Design, vol. 27, no. 2, pp. 142-144, 2006.

[12] Wen Shaochun, et al., “Survey on the Application of Genetic Algorithms in the Artificial Neural networks,” Computer Technology and Automation, vol. 20, no. 2, pp. 1-5, 2001.