Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 481919, 6 pages
http://dx.doi.org/10.1155/2015/481919

Research Article

Research and Application of Improved AGP Algorithm for Structural Optimization Based on Feedforward Neural Networks

Ruliang Wang, Huanlong Sun, Benbo Zha, and Lei Wang

Computer and Information Engineering College, Guangxi Teachers Education University, Nanning 530023, China

Correspondence should be addressed to Ruliang Wang; wrl@gxtc.edu.cn

Received 31 May 2014; Revised 18 September 2014; Accepted 7 October 2014

Academic Editor: Yiu-ming Cheung

The adaptive growing and pruning algorithm (AGP) has been improved, and the network pruning is based on the sigmoidal activation value of the node and all the weights of its outgoing connections. The nodes are pruned directly, but those nodes that have internal relation are not removed. The network growing is based on the idea of variance. We directly copy those nodes with high correlation. An improved AGP algorithm (IAGP) is proposed. And it improves the network performance and efficiency. The simulation results show that, compared with the AGP algorithm, the improved method (IAGP) can quickly and accurately predict traffic capacity.

1. Introduction

Artificial neural networks have been widely applied in data mining, web mining, multimedia data processing, and bioinformatics [1]. The success of the artificial neural network is largely determined by its structure. The optimization of network structure is usually a trial-and-error process by growing or pruning method. However, many algorithms employ the hybrid algorithm to optimize network structure [2], such as AGP.

Generally speaking, the method of optimizing neural network structure includes growing method, pruning method, and the hybrid algorithm of the two strategies basically. The first is also known as a constructive method. Based on the minimum network, adding new hidden units trains the network by data [3]. We know that the grow-when-required (GWR) algorithm of Marsland adds the hidden nodes based on the network performance requirements [4]. The disadvantages of growing methods are that the initial small network can be easily overfitting and trapped in local minima and it may also increase the training time [1].

The second method is called the destructive method, which deletes the unimportant nodes or weights in the original large network [5]. Lauret et al. put forward the extended Fourier amplitude sensitivity algorithms to prune the hidden neurons. This algorithm quantifies the correlation of neurons in the hidden layer and sorts it. And finally it iterates the most favorable neurons by using quantitative information and prunes the notes that rank late. By this method, however, the output and input of the network hidden neurons are independent [6]. When there are dependencies between them, this method is invalid. Xu and Ho describe a UB-OBS pruning algorithm that prunes the hidden units of feedforward neural network. It uses orthogonal decomposition method to determine the hidden node that needs pruning and then recalculate the weights of the remaining nodes to maintain the network performance. But the biggest drawback of pruning method is to determine the size of the initial network [7].

There will be more problems only by growing or pruning algorithms, so the hybrid algorithm of growing and pruning algorithms is proposed. It does not need to determine the initial network and does not carry out overfitting [8]. And it can be complementarily the two kinds of algorithms by enlarging their respective advantages and narrowing disadvantages [1]. AGP is a kind of growing pruning hybrid algorithm. In the structural design, the algorithm is based on the sigmoidal activation value of the node to adjust the neural network by pruning the little value neurons, merging similar neurons, and increasing the corresponding neurons, so it can adjust

the structure of network self-adaptively [9]. In recent decades, the structure optimization algorithm of neural network has received extensive attention [10–17]. The algorithm could be applied to nonlinear function approximation problems, but it has many times of iteration, complex calculation and needs to set threshold and adjust the parameters frequently.

Therefore, the feedforward neural network structure optimization algorithm still has much room for improvement. So IAGP was presented in this paper. Network pruning is based on the sigmoidal activation value of the node and all the weights of its outgoing connections. Network growing is based on the idea of variance. We directly copy those nodes with high correlation. It can rapidly, accurately, and self-adaptively optimize network structure.

Finally, it is applied to nonlinear function approximation and prediction of traffic capacity, and simulation results show the effectiveness of the improved AGP algorithm.

2. IAGP

2.1. AGP. This algorithm can solve the problem of adjusting the structure of network self-adaptively. First, it creates an initial feedforward neural network and then trains network by using BP algorithm until it reaches the target error. Otherwise, it calculates the sigmoidal activation value of the node to prune all the insignificant neurons and combines a large number of neurons to achieve the purpose of simplifying the network. Then after a certain amount of training, if it still does not reach the target accuracy, we will increase node based on the idea of cell division. It ensures that the growing node is the best. At the same time, it ensures the correlation between the two nodes. Then we retrain the network. If classification accuracy of the network falls below an acceptable level, then stop training; otherwise, continue training [9].

2.2. IAGP. In order to improve network performance and efficiency, IAGP was presented in this paper. First, the algorithm creates an initial network based on the actual problem. Here we assume that the initial network is a fully connected multilayer feedforward neural network with L layers, as shown in Figure 1.

In each lth layer, let m_l be the number of neurons where $0 \le l \le L$. Here we let the first layer 0 be an input layer, let the layers between 0 and L be hidden layers, and let the last layer L be an output layer. The ith input neuron of 0th layer is N_{i_0}, $0 \le i_0 \le m_0$, and the m_0th input neuron's bias value is always equal to 1. Let np be the number of patterns, in a dataset, and the value of the ith input neurons of pth pattern is x_{ip}. Among L layers in a network, the jth neuron of lth hidden layer is N_{jl}, where $0 < l < L$ and $1 \le j_l \le m_l$. The weight between input neuron N_{i_0} and hidden neuron N_{j_l} is ω_{ij_1}, $j_l \in \{1, 2, \ldots, m_l\}$. The weight between a neuron N_{jl} and a neuron $N_{k_{l+1}}$ is $v_{j_l k_{l+1}}$, $k \in \{1, 2, \ldots, m_{l+1}\}$, and the initial weights generally take a random value between −1 and 1.

The activation value of neuron N_{j1} is h_{j1}, and the activation value of neuron N_{jl} is h_{jl}. Here let o_k be the output

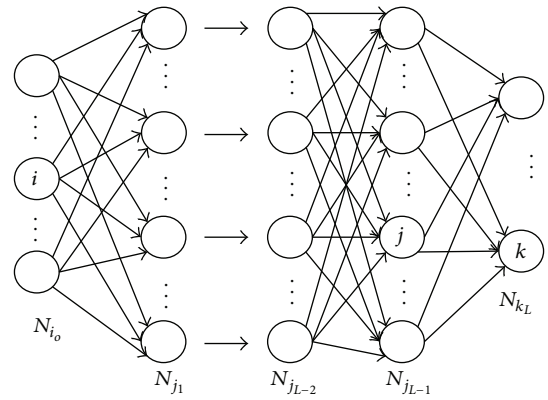

Figure 1: Multilayer feedforward neural network.

of the N_{k_L} th neuron in output layer L, where $1 \le k \le m_L$. BP algorithm is adopted here, and h_{j1} and h_{jl} can be written as

$$h_{j1} = f\left(\sum_{i=0}^{m_0}\left(x_{ip}\cdot\omega_{ij_1}\right)\right),$$
$$h_{jl} = f\left(\sum_{j_{l-1}=1}^{m_{l-1}}\left(h_{j_{l-1}}\cdot v_{j_{l-1}k_l}\right)\right), \tag{1}$$

where $f(x) = 1/(1+e^{-x})$, $1 < l < L$; based on the above, we can get the output o_k:

$$o_k = f\left(\sum_{j_{L-1}=1}^{m_{L-1}}\left(h_{j_{L-1}}\cdot v_{j_{L-1}k_L}\right)\right), \tag{2}$$

where $f(x) = 1/(1+e^{-x})$.

Here the value mean squared error is $E = (1/np)\sum_{p=1}^{np}(\sum_{k=1}^{n}(1/2)(o_k - d_k)^2)$; we know the dataset with K objects and the desired known target value d_k; and we can use BP algorithm to train the dataset. The total net value of the neuron N_{jl} can be written as

$$\text{tnet}_{j_l} = \begin{cases} \sum_{p=1}^{np}\sum_{i=0}^{m_0} x_{ip}\cdot\omega_{ij_1} & l = 1 \\ \sum_{j_{l-1}}^{m_{l-1}} f\left(\text{tnet}_{j_{l-1}}\right)\cdot v_{j_{j-1}j_l} & 1 < l < L. \end{cases} \tag{3}$$

Then the significance measure s_{j_l} can be expressed as

$$s_{j_l} = \sum_{k_{l+1}}^{m_{l+1}}\left|f\left(\text{tnet}_{j_l}\right) + b_l v_{j_l k_{l+1}}\right|, \tag{4}$$

where $f(\text{tnet}_{j_l}) = 1/(1+e^{-\text{tnet}_{j_l}})$, $\sum_{l=0}^{L} b_l = 1$, and $0 \le l \le L$.

According to the above formula, we can see that the significance measure s_{j_l} of a neuron N_{jl} is computed by adding its aggregated activation value over all the patterns with all its outgoing connections.

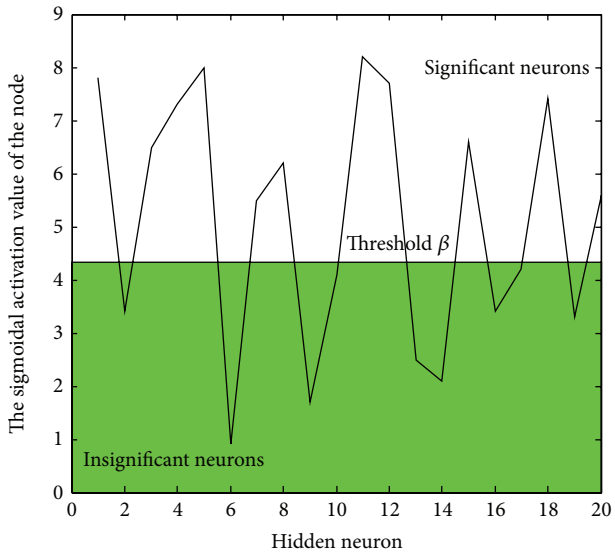

FIGURE 2: Hidden neuron N_{j_l}.

In order to achieve the purpose of pruning neural network, we should combine similar neurons, and the weight of the new neurons can be expressed as

$$\omega_{\text{new}} = P\omega_1 + (1 - P)\,\omega_2, \tag{5}$$

where ω_1 and ω_2 are 2 initial neuron weights and P is their similarity, where $P = ((\omega_1 * \omega_2)^2 - [(\omega_1 + \omega_2)/2]^2)/|\omega_1^2 - \omega_2^2|$.

When the neural network needs pruning, network adjustment for hidden layer neurons is based on the following formula:

$$N_{j_l} = \left\{ \begin{array}{ll} \text{insignficant} & s_{j_l} \le \beta,\ \beta = \sum_{j_l=1}^{m_l} \dfrac{\left(s_{j_l}\right)}{m_l} \\ \text{significant} & \text{otherwise} \end{array} \right\}, \tag{6}$$

where β is the threshold value, $0 \le l \le L$, and s_{j_l} is the neuronal contribution value; if it is less than the threshold, the neuron is meaningless; if it is more than the threshold, it is significant.

The process of identifying insignificant hidden neurons is shown in Figure 2.

Similarly, we can get the rule of pruning the input layer as follows:

$$s_i = \sum_{j_1}^{m_1} \left| f\left(tx_{ip}\right) + b_i \omega_{ij_1} \right|, \tag{7}$$

where $0 \le i < m_0$, $\sum_{i=0}^{m_0} b_i = 1$, and $tx_{ip} = \sum_{p=1}^{np} x_{ip}$.

So

$$N_{i_0} = \left\{ \begin{array}{ll} \text{insignificant} & s_i \le \alpha,\ \alpha = \sum_{i=0}^{m_0} \dfrac{\left(s_i\right)}{m_0} \\ \text{significant} & \text{otherwise} \end{array} \right\}. \tag{8}$$

Here the threshold is obtained by calculating the average of all contributions based on the sigmoidal activation value of the node and all the weights of its outgoing connections. It only eliminates neurons below the threshold and less number of iterations. Because it inherits the weight of the previous network, it reduces the amount of pruning step and does not make any complicated calculations, sets thresholds, and adjusts parameters.

After the above steps, if it still cannot reach the target, here we assume that the algorithm cannot fully learn the sample. So we need to add nodes with the idea of inheritance and significance measure's variance. We directly copy those nodes with high correlation g (select the intensity broad point and then average them):

$$\begin{aligned} c &= \frac{\sum_{i=1}^{n} \left(S_i - \overline{S}\right)^2}{n}, \qquad a = c_{\min}, \\ b_i &= a \pm d, \qquad g = \frac{\sum_{i=1}^{n} B_i}{n}, \end{aligned} \tag{9}$$

where $i = 1, 2, \ldots, n$; $d \in (0, 0.01)$.

c is significance measure's variance, S_i is the significance measure of the i neuron, $\overline{S}$ is the average value of the significance measure of all neurons from 1 to n, a is the smallest variance, b_i is an intensity variance near a, and B_i is the node whose density is wide. Let the hidden neuron g be a parent node, and copy it into R parts. The input weight of the new node is $\omega_{i\text{new}} = \omega_{i\text{old}}$ and the output weight is $\omega_{o\text{new}} = h_n \omega_{o\text{old}}$, $\sum_{n=1}^{R} h_n = 1$, $n = 1, 2, \ldots, R$.

$\omega_{i\text{new}}$ and $\omega_{i\text{old}}$ are, respectively, the input weights of old and new neuron and $\omega_{o\text{new}}$ and $\omega_{o\text{old}}$ are, respectively, the output weights of old and new neuron. The direct "copy," thought to add new nodes, can retain the relevance between nodes, greatly reduce the error, prevent overfitting, and quickly converge, and be fewer iterations.

2.3. The Algorithm. IAGP is based on the sigmoidal activation value of the node and all the weights of its outgoing connections. We optimize the neural network structure by increasing or decreasing the neurons. We can use BP algorithm to train network until it reaches the target error. It can quickly and effectively achieve the target error.

Compared with AGP algorithm, the improved AGP algorithm has the following advantages.

(1) Because the growth method and pruning method are adopted, the training time is greatly reduced and the training step is relatively short.

(2) Although the structure of neural network that is optimized by IAGP algorithm is more simple, it also keeps the overall performance of the original network.

(3) It does not need to set parameters in advance and these parameters are directly obtained by calculation.

(4) The IAGP has better fitting accuracy and generalization ability than the original algorithm.

(5) It can achieve network performance requirements faster and better.

The pseudocode of IAGP is as follows.

Step 1. Create a small initial network, and then use BP algorithm to train network.

Step 2. If classification accuracy of the network falls below an acceptable level, then stop pruning and go to Step 6; otherwise, go to Step 3.

Step 3. Calculate the sigmoidal activation value of the node and combine a large number of neurons to achieve the purpose of simplifying the network.

Step 4. Using the improved pruning method to train dataset, if it met the requirement of network performance, go to Step 6; otherwise, go to Step 4.

Step 5. After the above steps, if it still does not reach the target accuracy, at that time, we use the improved growing method to train the dataset; as we know, if it met the network performance, go to Step 6; otherwise, go to Step 2.

Step 6. End the neural network training.

Research indicates that IAGP can quickly and efficiently adjust the network structure accurately, reduce a large number of steps, and improve the efficiency.

3. Simulation Experiments

In this paper, considering the effectiveness of IAGP, it is applied in the prediction of nonlinear function approximation and the transportation capacity. The algorithm is proven to be effective according to simulation result.

3.1. Approximation of the Nonlinear Function. Consider the following nonlinear function:

$$f(x) = 1 - e^{-0.2x} \cdot \cos 0.8x + \cos 0.2x, \tag{10}$$

where $x \in [1, 6\pi]$. There are 70 groups of experimental data as the training samples and 30 groups as test samples. There are 15 initial hidden neurons, and we use improved AGP algorithm to train the network. Then 7 hidden nodes are left.

Figure 3 shows the effect of nonlinear function's approximation by neural network. Compared with AGP, we easily find that AGP can approximate the function effective better, faster and more effective. In Figure 4, it is a training error.

3.2. Application for Transport Capacity. We all know that the transportation has the nonlinearity complexity and randomness [18]. This paper adopts the IAGP algorithm to predict the transportation capacity. In order to be able to handle the transport demand and well predict the transportation capacity, we need to get some parameters based on the analysis of the factors influencing the freight volume. These parameters maybe include *GDP, industrial output, the length of railway line*, the proportion of double track mileage, the length of road transport routes, the proportion of grade highway, the number of railway train, and the number of laden civilian vehicles. These parameters can be used as the input vectors of the artificial neural network and the output vectors are total volumes of cargo transportation, rail freight, and highway freight. The neural network structure of experiment is 8-24-3, the model is shown in Figure 5, and the experimental data comes from China yearbook.

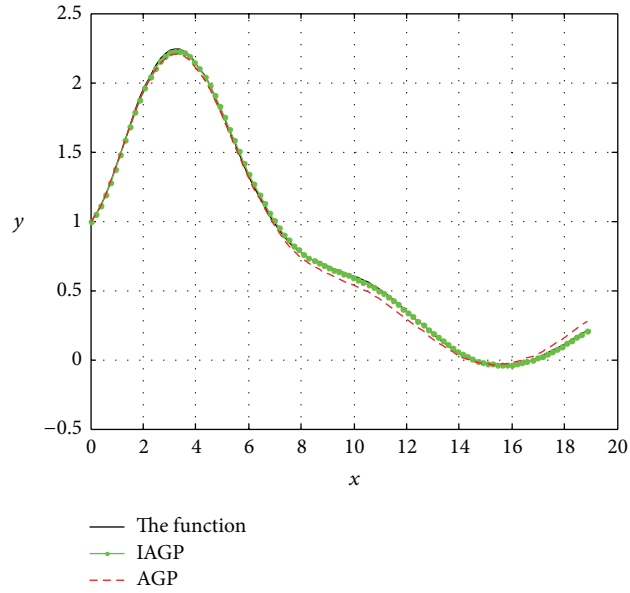

Figure 3: Approximation of the nonlinear function by IAGP.

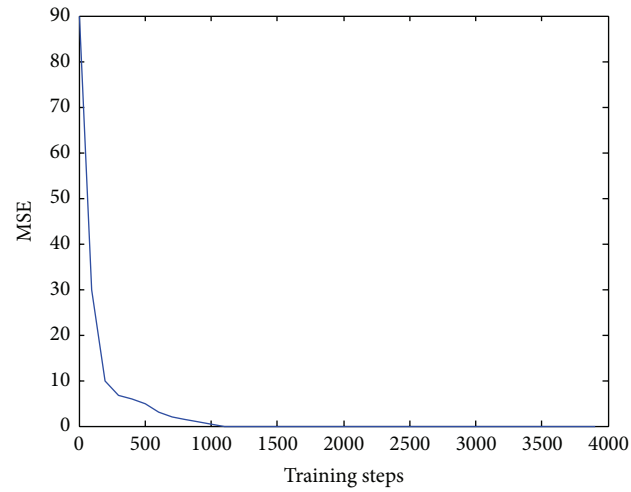

Figure 4: The training error curve of IAGP.

It selects the statistical data of 2002 to 2009 as training sample of the experiment and the statistical data of 2009 to 2013 as test sample of the experiment. Let the initial number of neurons in the hidden layer of new AGP and AGP be 10, and the network training error is 0.01.

It is shown in Table 1 that 2 kinds of optimization algorithm performance are compared. With IAGP, the number of neurons in hidden layer of neural network is 6, training error is 0.031, training steps are 246, and training time is 23.8. By contrast, the improved algorithm has the corresponding

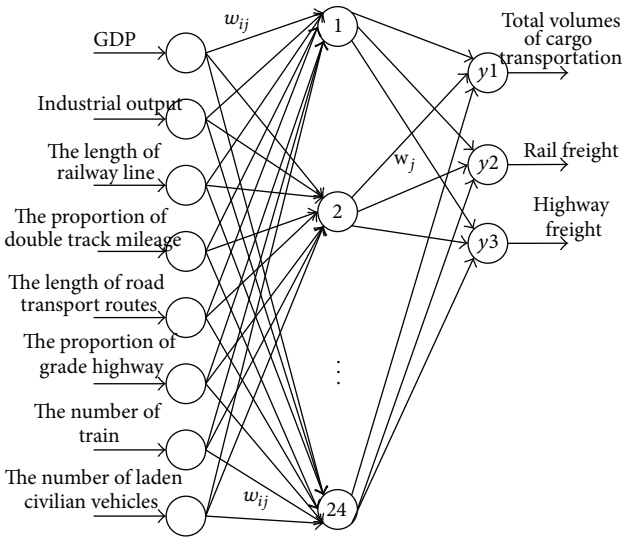

Figure 5: Feedforward neural network structure.

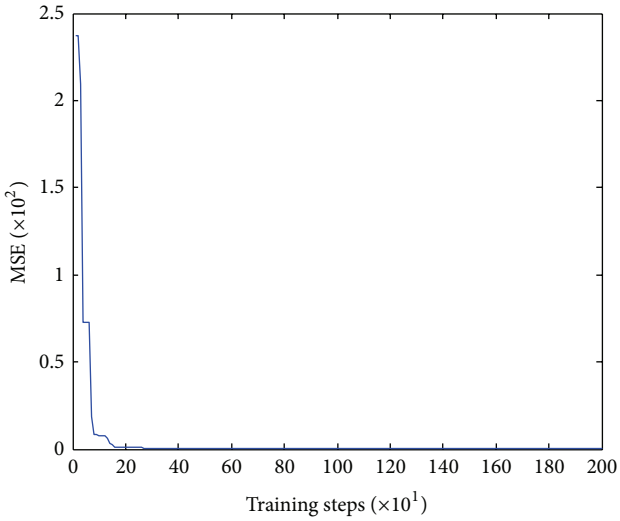

Figure 7: The training error curve of IAGP.

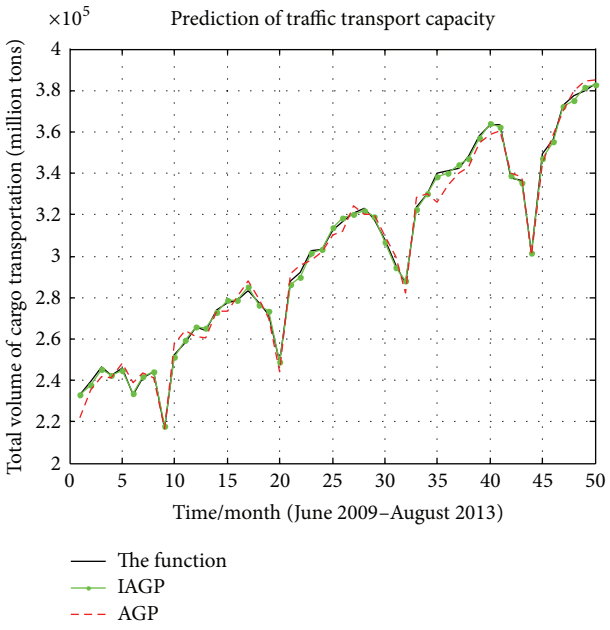

Figure 6: Two kinds of algorithm in contrast to traffic prediction.

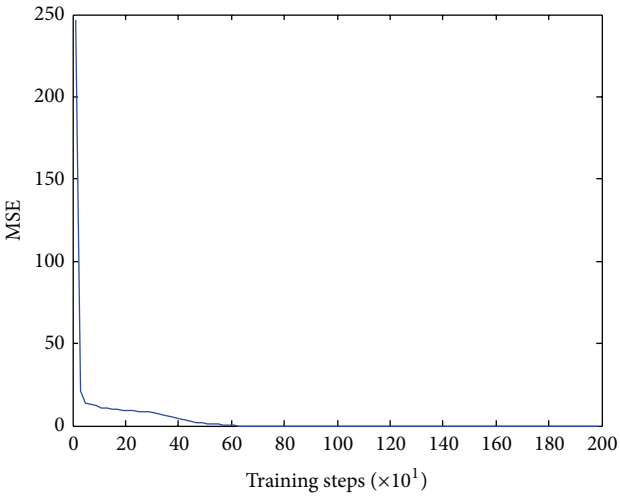

Figure 8: The training error curve of AGP.

Table 1: Comparison of 2 kinds of optimization algorithm performance.

Algorithm	Neuron number	Training error	Training steps	Training time
IAGP	6	0.031	246	23.8
AGP	8	0.047	953	26.6

improvement in the four aspects, and the improved AGP algorithm does not change the overall structure of neural network. So this method is very practical.

Figure 6 shows the performance of the improved AGP algorithm and AGP algorithm in the traffic prediction. It can be seen that the improved AGP algorithm results are basically consistent with the actual situation. Although AGP algorithm generally can keep up with the actual traffic forecast, it has a little lag error or a little gap. Network training step is shown in Figure 7, probably in iteration 250; the network gradually stabilized. While Figure 8 is about 610 iterations, the network became more stable. So the use of IAGP algorithm is faster than the AGP algorithm. The training error is as shown in Figure 8.

Simulation results show that IAGP can well predict the transportation capacity, can be very good to follow the actual output, and has little error. The approximation speed of AGP is slower and maybe has a bigger error.

As you can see from Figure 6, the traffic freight volume of China is increasing every year. This algorithm plays an important role in forecasting transport ability in our economy and can be reasonably optimization related traffic resources.

4. Conclusions

This paper researches and improves AGP. First of all, we use BP algorithm to train network. Then pruning neurons are based on the sigmoidal activation value of the node and all the weights of its outgoing connections, and growing neurons are based on the correlation of significance measure's variance. Then it trains the neural network until reaching the target accuracy. With the IAGP, we change little network structure, have a few training steps, and have short time, and network structure is more simple. The experimental results show that the method improves the efficiency and accuracy of the traffic prediction.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was jointly supported by the Guangxi Key Laboratory Foundation of University and Guangxi Department of Education Foundation.

References

[1] Z. Zhang, *The Study of Self-Organization Modular Neural Network Architecture Design*, Beijing University of Technology, 2013.

[2] Z. Zhang, J. Qiao, and G. Yang, "An adaptive algorithm for designing optimal feed-forward neural network architecture," *CAAI Transactions on Intelligent Systems*, vol. 6, no. 4, 2011.

[3] X. Yu, *The structural optimization research for FNN controller based on the combination of pruning method and growth method [Master thesis]*, Southwest Jiao tong University, 2009.

[4] S. Marsland, J. Shapiro, and U. Nehmzow, "A self-organising network that grows when required," *Neural Networks*, vol. 15, no. 8-9, pp. 1041–1058, 2002.

[5] J.-F. Qiao, M. Li, and J. Liu, "A fast pruning algorithm for neural network," *Acta Electronica Sinica*, vol. 38, no. 4, pp. 830–834, 2010.

[6] P. Lauret, E. Fock, and T. A. Mara, "A node pruning agorithm based on a fourier amplitude sensitivity test method," *IEEE Transactions on Neural Networks*, vol. 17, no. 2, pp. 273–293, 2006.

[7] J. Xu and D. W. C. Ho, "A new training and pruning algorithm based on node dependence and Jacobian rank deficiency," *Neurocomputing*, vol. 70, no. 1–3, pp. 544–558, 2006.

[8] H.-Z. Yang, W.-N. Wang, and F. Ding, "Two structure optimization algorithms for neural networks," *Information and Control*, vol. 35, no. 6, pp. 700–704, 2006.

[9] M.-N. Zhang, H. Han, and J. Qiao, "Research on dynamic feed-forward neural network structure based on growing and pruning methods," *CAAI Transactions on Intelligent Systems*, vol. 6, no. 2, 2011.

[10] M. Gethsiyal Augasta and T. Kathirvalavakumar, "A novel pruning algorithm for optimizing feedforward neural network of classification problems," *Neural Processing Letters*, vol. 34, no. 3, pp. 241–258, 2011.

[11] J.-J. Tu, Y.-Z. Zhan, and F. Han, "Neural network correlation pruning optimization based on improved PSO algorithm," *Application Research of Computers*, no. 9, pp. 3253–3255, 2010.

[12] Y. Wang and C. Dang, "An evolutionary algorithm for global optimization based on level-set evolution and latin squares," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 579–595, 2007.

[13] J. Qiao and Y. Zhang, "Fast unit pruning algorithm for multilayer feed-forward network design," *CAAI Transactions on Intelligent Systems*, vol. 3, no. 2, 2008.

[14] H.-R. Yan and L.-J. Ma, "Design and realization of intelligent prediction model based on fuzzy neural network," *Modern Electronic Technique*, no. 2, pp. 84–88, 2008.

[15] W. Wang and H. Yang, "Pruning algorithm for neural networks based on pseudo-entropy of weights," *Computer Simulation*, vol. 23, no. 3, 2006.

[16] Y. Li, Y. Wang, P. Jiang, and Z. Zhang, "Multi-objective optimization integration of query interfaces for the Deep Web based on attribute constraints," *Data and Knowledge Engineering*, vol. 86, pp. 38–60, 2013.

[17] Q.-K. Song and M. Hao, "Structural optimization of BP neural network based on correlation pruning algorithm," *Control Theory and Applications*, vol. 12, 2006.

[18] X. Xu, "A forecast model of freight capacity based on RBF network," *Aeronautical Computing Technique*, vol. 37, no. 5, 2007.