Enhanced Fuzzy Multi-Layer Perceptron

Kwang-Baek Kim[1], Choong-Sik Park[2] and Abhjit Pandya[3]

[1]Dept. of Computer Science, Silla University, Pusan, Korea
[2]Faculty of Computer Information Eng., Youngdong University, Youngdong, Korea
[3]Department of Computer Science and Engineering, Florida Atlantic University, Florida, USA
E-mail : gbkim@silla.ac.kr, leciel@youngdong.ac.kr, abhi@silla.ac.kr

Abstract— In this paper, we propose a novel approach for evolving the architecture of a multi-layer neural network. Our method uses combined ART1 algorithm and Max-Min neural network to self-generate nodes in the hidden layer. We have applied the proposed method to the problem of recognizing ID number in student identity cards. Experimental results with a real database show that the proposed method has better performance than a conventional neural network.

I. INTRODUCTION

The backpropagation network (BPN), which is a form of a multilayer perceptron (MLP), is currently the most general-purpose and commonly used neural network paradigm[1]. The BPN achieves its generality because of the gradient descent technique used to train the network. Gradient descent is analogous to an error minimization process. The idea is to minimize the network total error by adjusting the weights. Gradient decent, sometimes known as the method of steepest descent, provides a means of doing this. Each weight may be thought of as a dimension in an N-dimensional error space. In error space the weights act as independent variable and the shape of the corresponding error surface is determined by the error function in combination with the training set. However, BPN has a drawback, since, despite its popularity as an optimization tool, not only for neural network training, the gradient descent has several drawbacks such as the possibility of getting trapped in local minima[2,3,4]. Grossberg developed the adaptive resonance theory (ART). ART was developed to solve the learning instability problem suffered by standard feed-forward networks[5,6]. The weights, which have captured some knowledge in the past, continue to change as new knowledge comes in. There is therefore a danger of losing the old knowledge with time. The weights have to be flexible enough to accommodate the new knowledge but not so much as to lose the old. This is called the stability-plasticity dilemma and it has been one of the main concerns in the development of artificial neural network paradigm[7]. Max-Min neural network uses fuzzy logic to update the weights in a multi-layer perceptron rather than the delta rule, which uses multiplication and addition operations [8]. This paper presents a MLP using combined self-generating model of ART1 and Max-Min neural network for enhancing recognition ratio and solving a problem of hidden layer's node numbers.

II. FUZZY MULTI-LAYER PERCEPTRON ARCHITECTURE

BP learning method used widely in multi-layer neural networks has a possibility of getting trapped in local minima due to inadequate weights and insufficient number of hidden nodes.

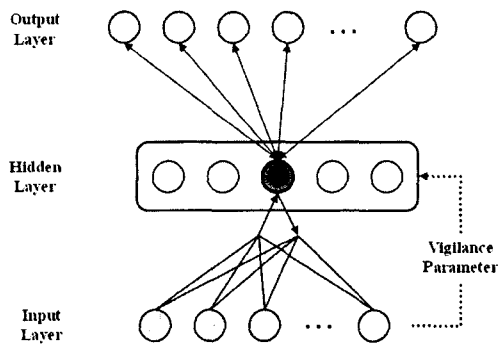

Fig. 1 Fuzzy multi-layer architecture

Therefore, we propose a novel fuzzy multi-layer perceptron using self-organization that self-generates hidden nodes using the compound ART1 algorithm and Max-Min neural network. Fig.1 shows the proposed learning architecture to self-generate nodes of the hidden layer. The proposed network is presented with a large number of patterns and each hidden layer neuron represents the cluster center. The prototype pattern for each cluster is represented by the weights from the hidden neuron to the input neuron. Vigilance criterion is

used to achieve unsupervised learning which determines the actual number of clusters. In the proposed architecture, the connection structure between input layer and hidden layer is similar to structure of the modified ART1. The output layer of the modified ART1 is used as the hidden layer in the proposed structure. A node of hidden layer represents each class. The nodes in hidden layer are fully connected to nodes in input and output layers. In the case of backward propagation by comparing target value with actual output value, we adapt a winner-take-all method to modify weighting factor of only the synapse that is connected to the neuron representing the winner class. The adaptation of weight of synapses between output layer and hidden layer is accomplished by Max-Min neural network.

III. FUZZY MULTI-LAYER PERCEPTRON USING SELF-GENERATION

Use The creation for organizing the clustering layer nodes is based on the number of determining classes based on input patterns. Based on ART1, we assume the number of maximum initial nodes of the hidden layer as the number of classes.

Starting with one node, we allocate related classes to the initially suggested pattern from the input layer of the node. Next input patterns choose a winner from the nodes in the present state. If all the existing nodes fail to choose a winner, one node is added and allocated to the class for the presented pattern. In this way, patterns are sequentially presented and the nodes for the class are created dynamically. If the stored pattern of the winner node is similar to the input pattern, it becomes the winner. Otherwise, classification is repeated until we get a winner. If an existing node is found to be a winner node, all the weights linking that node to the input layer are updated to reflect the accommodation of that input pattern into the representative class.

The proposed algorithm uses a winner-take-all method instead of the conventional error backpropagation learning to change weights. When we classify the connection between the input layer and the clustering layer, and the connection between the clustering layer and the target layer, the winner node chosen from the clustering layer becomes the representative class of input pattern. Therefore, we should adjust the weights connected to the winner node from the clustering layer to the input layer. To reflect target value for the input pattern to the actual output value by the representative class, we change only the connection weights related to the target layer node and its representative class.

The proposed fuzzy multi-layer perceptron is trained using self-generation as follows:

Step 1. Initialize weights, bias term and vigilance threshold.

$$b_{ji} = 1,\ w_{ji} = \frac{1}{m+1}$$

set ρ, where $0 < \rho \le 1$, w_{kj}, θ_k : small random value,

$0 \le i \le m-1,\ 0 \le j \le n-1,\ 0 \le k \le p-1$

where b_{ji} is the value of the top-down weight from neuron i in the input layer to neuron j in the hidden layer and w_{ji} is the value of the bottom-up weight from neuron i in the input layer to neuron j in the hidden layer. w_{kj} is the value of a weight from neuron j in the hidden layer to neuron k in the output layer. θ_k is a bias term in the output layer. ρ is the vigilance threshold, which determines how close an input has to be to correctly match a stored pattern.

Step 2. Set target value t_k and train each input data x_i.

Step 3. Calculate output o_{j^*} in hidden layer.

$$o_j = \sum_{j=0}^{n-1} w_{ji} \times x_j \tag{1}$$

Step 4. Select a winner node O_{j^*}.

$$o_{j^*} = Max[o_j] \tag{2}$$

The method that selects winner node for input data is that the winner node maximizes output value o_j in hidden layer.

Step 5. Compare the similarity.

If $\frac{\| T \bullet X \|}{\| X \|} \ge \rho$, go to step 7.

Else, go to step 6.

where ρ is the vigilance parameter.

Step 6. Reassign zero to o_{j^*} in the winner node and go to step 4.

Step 7. Update the connection weights of the winner node between hidden layer and input layer.

$$t_{j^*i}(n+1) = t_{j^*i}(n) \times x_i \tag{3}$$

$$w_{j^*i}(n+1) = \frac{t_{j^*i}(n+1) \times x_i}{0.5 + \sum_{i=1}^{m} w_{j^*i} \times x_i} \tag{4}$$

Step 8. Calculate node's NET for output layer using the winner node's output o_{j^*} in hidden layer and the connection weight w_{kj^*} between hidden layer and output layer. And then calculate the output o_k of output layer using fuzzy max($\vee$) operator.

$$NET = \{o_{j^*} \circ w_{kj^*}\} \quad (5)$$

$$o_k = NET \vee \theta_k \quad (6)$$

where "$\circ$" denotes max-min composition.

Step 9. Update the connection weights between output layer and hidden layer and bias term.

$$w_{kj^*}(n+1) = w_{kj^*}(n) + \alpha\Delta w_{kj^*}(n+1) + \beta\Delta w_{kj^*}(n) \quad (7)$$

$$\theta_k(n+1) = \theta_k(n) + \alpha\Delta\theta_k(n+1) + \beta\Delta\theta_k(n) \quad (8)$$

where α is the learning rate and β is the momentum.

$$\Delta w_{kj^*} = \sum_{k=1}^{p}(t_k - o_k)\frac{\partial o_k}{\partial w_{kj^*}}, \Delta\theta_k = \sum_{k=1}^{p}(t_k - o_k)\frac{\partial o_k}{\partial \theta_k} \quad (9)$$

$$\frac{\partial o_k}{\partial w_{kj^*}} = 1, \text{ where } o_k = w_{kj} \quad \frac{\partial o_k}{\partial w_{kj^*}} = 0, \text{ otherwise.} \quad (10)$$

$$\frac{\partial o_k}{\partial \theta_k} = 1, \text{ where } o_k = \theta_k \quad \frac{\partial o_k}{\partial \theta_k} = 0, \text{ otherwise.} \quad (11)$$

Step 10. For all training pattern pair, if (*TSS<Error Criteria*) then stop learning.

IV. EXPERIMENTS AND PERFORMANCE ANALYSIS

To analyze proposed method's performance, we simulated the algorithm using Intel Pentium-1GHz PC and Visual C++ 6.0. HP Scanjet 4200C scanner obtains the BMP files (600x400) of student ID card, and we compared the proposed algorithm with conventional backpropagation algorithm using 10 ID codes extracted from student ID cards.

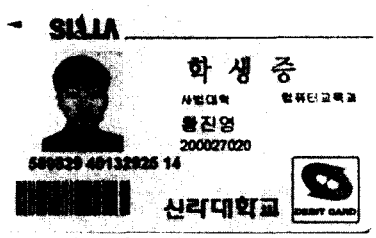

Fig. 2 Sample of test image

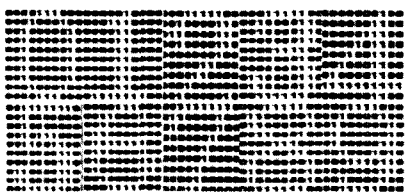

Fig. 3 ID code training data

The region of ID code was extracted by two procedures. First the region containing the ID numbers was extracted from the ID card image. Then the individual numbers were extracted from that field. The individual ID code was extracted as follows: The presented scheme sets up an average brightness as a threshold, based on the brightest pixel and the least bright one for the source image of the ID card. After converting to a binary image, a horizontal histogram was applied and the ID code was extracted from the image. Then the noise was removed from the ID code region using mode smoothing with a 3x3 mask. After removing noise from the ID-code region, the individual ID code was extracted using a vertical histogram[9]. For training, we used 10 ID patterns as shown in Fi.3.

Table 1 presents the comparison of epoch's number and TSS using the proposed method and the conventional backpropagation algorithm with error limit 0.01.

Table 1 Epoch Number & TSS according to Momentum

	Momentum	# of epoch	TSS
Conventional BP	0.1	6320	0.01
	0.5	6034	0.01
	0.9	5410	0.01
The Proposed Method	0.1	204	0.009861
	0.5	57	0.002735
	0.9	57	0.002735

In backpropagation algorithm, when we set 5~10 nodes of hidden layer, we obtained the fact that the case of 10 nodes had good performance (fast training time, high convergence). Therefore, Table 1 presents the result of training with 10 hidden nodes. In the proposed method, since we applied ART1 to the structure of between input and hidden layer, it produced 10 hidden nodes after training. Fig.4 presents the change procedure for the epoch number and TSS based on the momentum in conventional backpropagation algorithm. Fig.5 presents a change procedure for epoch number and TSS based on the momentum in proposed method.

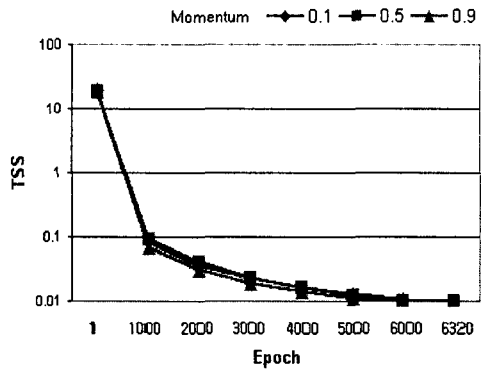

Fig. 4 Change procedure of epoch number and TSS based on momentum in conventional backpropagation algorithm

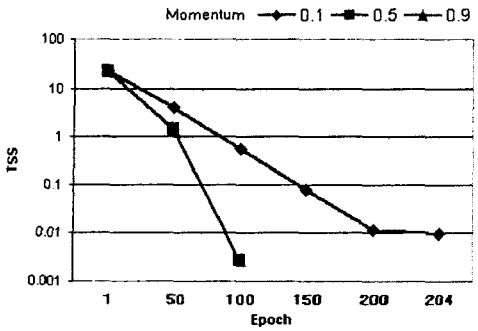

Fig. 5 Change procedure of epoch number and TSS base on momentum in proposed method

Table 2 presents the result of training through vigilance variable's change (0.5~0.9) in the proposed method. We set a momentum with 0.95 in Table 2.

Table 2 Epoch Number & Produced Node Number through Vigilance Parameter's Change

	Vigilance parameter	# of epoch	# of the nodes of hidden layer
Proposed Method	0.5	46	5
	0.6	52	7
	0.7	56	8
	0.75	56	9
	0.8	57	10
	0.85	57	10
	0.9	57	10
	0.95	57	10

We could confirm the fact that the proposed method changed the number of hidden node according to vigilance variable's value, but did not affect training time.

Table 3 shows the number of the success convergence by applying 50 number patterns to the error backpropagation and proposed method. In the error backpropagation algorithm, the initial learning rate is 0.3 and the momentum is 0.5. In the proposed algorithm, the vigilance parameter is 0.9. In the criterion of success convergence, the number of epoch is limited to 20,000 and TSS is 0.04.

Table 3 Comparison of Learning Convergence between BP and the Proposed Method

	# of trials	# of success	# of the nodes of hidden layer	# of epoch (average)
BP	10	4	4	10952
Proposed Method	10	10	15	571

As shown Table 3, the number of success convergence of the proposed method is larger and the average number of epoch of that is smaller than the error backpropagation algorithm.

Through experimental results, we found that the proposed method spent less time for training compared with the conventional training method, and had good convergence ability. This is based on the fact that winner-take-all method is adapted to the connection weight adaptation, so that a stored pattern for some pattern gets updated. Moreover, the proposed method reduced the possibility of local minima due to the inadequate weights and the insufficient number of hidden nodes. The reason is that adjustment of weights by the winner-take-all method decreases the amount of computation, and adjusting only the related weights decreases the competitive stages as a premature saturation. Therefore, there is less possibility of the paralysis and local minima in the proposed method.

V. CONCLUSIONS

This paper proposed an enhanced fuzzy multi-layer perceptron learning algorithm using self-organization that self-generates hidden nodes by the compound Max-Min neural network and modified ART1. From the input layer to hidden layer, a modified ART1 was used to produce nodes. Moreover, winner-take-all method was adapted to the connection weight adaptation, so that a stored pattern gets updated. To analyze proposed method's performance, we tested student ID-card images. Through experiments, we found that the proposed method was quite robust with respect to minor change in the momentum parameter. Moreover, it had good convergence ability, and took less training time than conventional backpropagation algorithm.

REFERENCES

[1] James and Freeman, Neural Networks: Algorithm, Application and Programming Techniques, Addison-Wesley, 1991.

[2] R. Hecht-Nielsen, "Theory of Backpropagation Neural Networks," Proceedings of IJCNN, Vol.1, pp.593-605, 1989.

[3] Y. Hirose, K. Yamashita, S. Hijihya, "Backpropagation Algorithm Which Varies the Number of Hidden Units," Neural Networks, Vol.4, pp.61-66, 1991.

[4] K. B. Kim, M. H. Kang and E. Y. Cha, "Fuzzy Competitive Backpropagation using Nervous System," Proceedings of WCSS, pp.188-193, 1997.

[5] S. N. Kavuri, V. Ventatasubramanian, "Solving the Hidden Node Problem in Neural Networks with Ellipsoidal Units and Related Issues," Proceedings of IJCNN, Vol.1, pp.775-780, 1992.

[6] M. Georipoulos, G. L. Heileman and J. Huang, "Properties of Learning Related to Pattern Diversity in ART1," Neural Networks, Vol.4, pp.751-757, 1991.

[7] K. B. Kim and K. C. Kim, "A Study on Face Recognition using New Fuzzy ART," Proceedings of ITC-CSCC, Vol.2, pp.1057-1060, 1998.
[8] T. Saito and M. Mukaidono, "A Learning algorithm for Max-Min Network and its Application to Solve Relation Equations," Proceedings of IFSA, pp.184-187, 1991.
[9] T. K. Kim, H. G. Yun, Y. W. Lho, K. B. Kim, "An Educational Matters Administration System on The Web by Using Image Recognition.," Proceedings of Korea Intelligent Information Systems, pp.203-209, 2002.