Multi-Layered Neural Networks with Learning of Output Functions

Lixin Ma†, *Hiromi Miyajima*†† *and Noritaka Shigei*††
† Shanghai University of Electric Power, China
†† Dept. of Electrical and Electronics Engineering, Kagoshima University, Japan

Summary
This paper proposes multi-layered neural networks with learning of output functions like RBF and fuzzy models. Various models that differ in the number of trained output functions are compared in two types of simulations: XOR problem and functional approximation. As a result it is shown that the proposed models are faster in learning time than the conventional one. Further, based on the simulation results, an effective heuristic model is proposed.

Key words:
Multi-layered neural networks, backpropagation, output function, XOR problem, function approximation

Introduction

Multi-layered neural networks perform well pattern recognition and nonlinear function by using backpropagation (BP) algorithms and have been successfully applied to many applications [1-3]. When the number of hidden units in the network is not restricted, it has been theoretically proven that such a network has an ability to approximate any continuous input-output relation with any accuracy [1,2]. On the other hand, the BP learning is guaranteed to converge to a local minimum but not the global minimum. In order to converge to a better local minimum, it is important to break the symmetry in the network components. From this point of view, we pay attention to output functions, that is, the conventional model has the same output function for each neuron. This paper proposes multi-layered neural networks with not only all weights but output functions like RBF and fuzzy models as learning parameters. Without loss of generality, three-layered neural networks are used. Various models that differ in the number of trained output functions are compared in two types of simulations: XOR problem and functional approximation. As a result it is shown that the proposed models are faster in learning time than the conventional one. Further, based on the simulation results, an effective heuristic model is proposed.

2. Multi-Layered Neural Networks and the Proposed Algorithm

Let us consider three-layered neural networks each of which consists of I, J and K neurons for input, hidden and output layers, respectively as shown in the

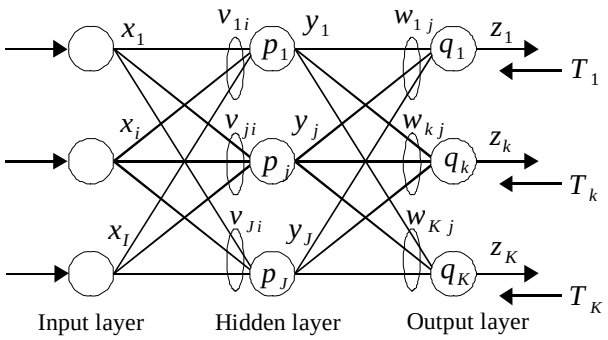


Fig. 1. The three layered neural network.

Fig.1, where x_i, y_j and z_k are the values of the i-th, j-th and k-th neurons, v_{ji} for x_i and y_j and w_{kj} for y_j and are weights and $\boldsymbol{T} = (T_1, \cdots, T_K)$ is the supervised output for input $\boldsymbol{x} = (x_1, \cdots, x_I)$.
Further, p_j and q_k for $j = 1, \cdots, J$ and $k = 1, \cdots, K$ are learning parameters of output functions for hidden and output layers. For input data $\boldsymbol{x}$, y_j and z_k ($j = 1, \cdots, J$, $k = 1, \cdots, K$) are obtained as follow:

$$y_j = \frac{1}{1 + \exp(-p_j \cdot g_j)} \qquad (1)$$

$$g_j = \sum_{i=1}^{I} (x_i \cdot v_{ji}) + v_{j0} \qquad (2)$$

$$z_k = \frac{1}{1 + \exp(-q_k \cdot h_k)} \qquad (3)$$

$$h_k = \sum_{j=1}^{J} (y_j \cdot w_{kj}) + w_{k0} \qquad (4)$$

where v_{j0} and w_{k0} are threshold values. The difference E between the supervised output $\boldsymbol{T}$ and the output $\boldsymbol{z} = (z_1, \cdots, z_K)$ of the network for input $\boldsymbol{x}$ is defined as follows:

$$E = \frac{1}{2} \sum_{k=1}^{K} (T_k - z_k)^2 \qquad (5)$$

Then the BP learning based on the gradient descent method is represented as follows [1,2]:

$$\begin{aligned} \Delta w_{kj} &= -\alpha \frac{\partial E}{\partial w_{kj}} \\ &= -\alpha \frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial h_k} \frac{\partial h_k}{\partial w_{kj}} \\ &= \alpha (T_k - z_k) z_k (1 - z_k) y_j \end{aligned} \qquad (6)$$

$$\begin{aligned} \Delta v_{ji} &= -\alpha \frac{\partial E}{\partial v_{ji}} \\ &= -\alpha \sum_{k=1}^{K} \left(\frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial h_k} \frac{\partial h_k}{\partial y_j} \right) \frac{\partial y_j}{\partial g_j} \frac{\partial g_j}{\partial v_{ji}} \\ &= \alpha \sum_{k=1}^{K} ((T_k - z_k) z_k (1 - z_k) w_{kj}) y_j (1 - y_j) x_i \end{aligned} \qquad (7)$$

where α is the learning number. In the proposed BP learning, parameters p_k and q_j are also updated as follows:

$$\begin{aligned} \Delta q_k &= -\alpha \frac{\partial E}{\partial q_k} \\ &= -\alpha \frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial q_k} \\ &= \alpha (T_k - z_k) z_k (1 - z_k) h_k \end{aligned} \qquad (8)$$

$$\begin{aligned} \Delta p_j &= -\alpha \frac{\partial E}{\partial p_j} \\ &= -\alpha \sum_{k=1}^{K} \left(\frac{\partial E}{\partial z_k} \frac{\partial z_k}{\partial h_k} \right) \frac{\partial y_j}{\partial q_j} \\ &= \alpha \sum_{k=1}^{K} ((T_k - z_k) z_k (1 - z_k) q_k w_j) y_j (1 - y_j) g_j \end{aligned} \qquad (9)$$

The proposed algorithm is shown as follows:

[Step 1] Initial Assignment:

Input and output data: $(\boldsymbol{x}^{(s)}, \boldsymbol{T}^{(s)})$ for $s = 1, \cdots, S$ are prepared. $t = 1$. $s = 1$.

[Step 2]

For $\boldsymbol{x} = \boldsymbol{x}^{(s)}$, by using Eqs. (1), (2), (3) and (4), the output $\boldsymbol{z}$ is computed as $\boldsymbol{z}^{(s)} = (z_1^{(s)}, \cdots, z_N^{(s)})$.

[Step 3]

The error $E^{(t)}$ at the step t is computed as follows:

$$E^{(t)} = \frac{1}{2} \sum_{k=1}^{N} (T_k^{(s)} - z_k^{(s)})^2 \qquad (10)$$

[Step 4]

The parameters p_k, w_{kj}, q_j and v_{ji} are updated based on the Eqs. (6), (7), (8) and (9) for $i = 1, \cdots, I$, $j = 1, \cdots, J$ and $k = 1, \cdots, K$.

[Step 5]

If $t = T_{\mathrm{MAX}}$, then the algorithm is terminated. Otherwise go to **Step 6**.

[Step 6]

$t = t + 1$, $s = (t - 1) \bmod S + 1$ and go to **Step 3**.

(End of algorithm)

Remark that the proposed method is different from the Boltzmann machine[1].

3. Numerical Simulation

In order to show the effectiveness of the proposed algorithm, two types of simulation are performed.

3.1 XOR Problem

To implement the XOR function, for each input $(x_1, x_2) \in \{(0,0), (0,1), (1,0), (1,1)\}$ we use the supervised data T_1 given in Table 1. The number of neurons in the hidden layer is $M = 2$. We compare the following six models:

- Model 1: $p_1 = p_2 = q_1 = 1$, that is, the conventional model.
- Model 2: parameters p_1, p_2 and q_1 are trained.
- Model 3: $q_1 = 1$, and parameters p_1 and p_2 are trained.
- Model 4: $p_1 = q_1 = 1$, and a parameter p_2 is trained.
- Model 5: $p_1 = 1$, and parameters p_2 and q_1 are trained.
- Model 6: $p_1 = p_2 = 1$, and a parameter q_1 is trained.

Fig. 2 shows the learning error for the XOR problem. Models 2, 3, 4, 5 and 6 converge faster than the conventional one. For example, the proposed models 2 and 6 reach $E = 0.001$ in less than half the time of the conventional one.

Table 1: Supervised data for XOR problem.

Input		Ideal output	Supervised data
x_1	x_2		T_1
0	0	0	0.1
0	1	1	0.9
1	0	1	0.9
1	1	0	0.1

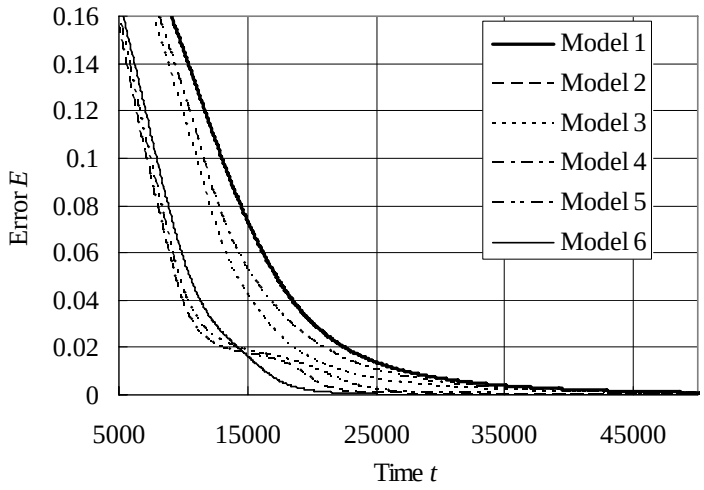


Fig. 1: Transition of learning error for EXOR problem

3.2 Function Approximation

Neural networks are trained so as to approximate the following functions:

$$z = x_1 \cdot x_2 \tag{11}$$

$$z = \frac{\sin(3\pi {x_1}^3) \cdot x_2 + 1}{2} \tag{12}$$

$$z = \frac{\sin(3\pi x_1) + \cos(3\pi x_2) + 2}{4} \tag{13}$$

In each function, x_1 and x_2 are input variables and z is the output variable. The number of learning data sets is $S = 200$, where input data are randomly chosen from $[0,1]$. The number of test data is 2500, where input data are uniformly chosen from $[0,1]$.

The numbers of neurons for hidden layers are $J = 3$ for Eq.(11) and $J = 10$ for Eqs.(12) and (13).

- Model 1: $p_1 = \cdots = p_J = 1$ and $q_1 = 1$, that is, the conventional method.
- Model 2: all the parameters p_j's and q_1 are trained.

The comparison between the conventional and the proposed models is made. In the proposed models, all the parameters p_j's and q_1 are trained. Figs. 3, 4 and 5 show the transitions of learning error for Eqs. (11), (12) and (13). In all cases, the proposed model is superior in learning time to the conventional one. Table 2 shows the result for test data. There is no difference of the generalization ability between two models. Though some other models with learning a part of parameters p_j's and q_1 are tested, the model 2 is fastest of all. Then can we find all the parameters p_j's and q_1 without learning? We have tried some cases such as random selecting and in order. As a result, we found an effective rule as follows.

Table 2: Learning error and test error for function approximation.

Function	T_{MAX}	Model	Learning Error	Test Error
Eq.(11)	8000	1 (Conventional)	0.0376	0.0395
		2	0.0307	0.0321
Eq.(12)	12000	1 (Conventional)	0.0349	0.0426
		2	0.0339	0.0433
Eq.(13)	12000	1 (Conventional)	0.0360	0.0402
		2	0.0315	0.0397

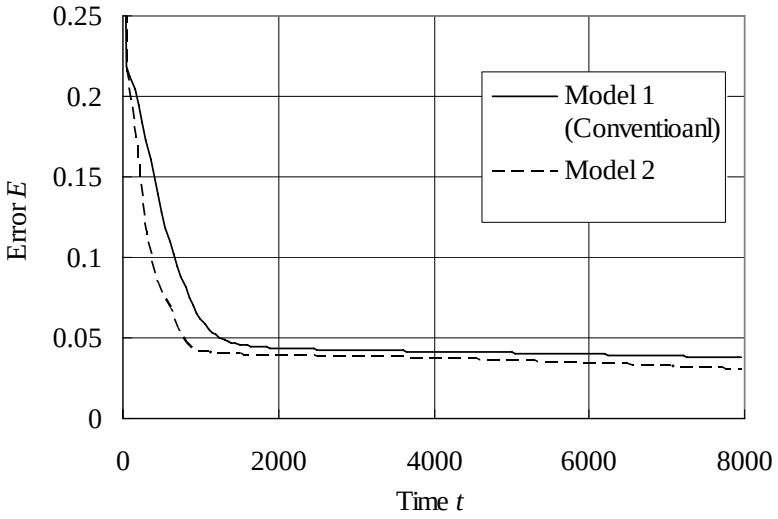


Fig. 2: Transition of learning error for Eq.(11).

- Model A: Parameters p_j 's in hidden layer are assigned equally spaced values from a reasonable range.

In the model, the range of p_j should be appropriately determined. The lower limit should not be too small and the upper limit should not be too large, because of the following reasons.

- Too small p_j such that $p_j \cdot g_j \approx 0$ converts a neuron into a mere bias unit, because $\frac{1}{1+\exp(-p_j \cdot g_j)} \approx \frac{1}{2}$.
- Too large upper limit produces many large p_j 's. For any large p_j, $\frac{1}{1+\exp(-p_j \cdot g_j)}$ takes a similar shape of the Heaviside step function.

For Eqs. (12) and (13), model A adopts $1.0 \le p_j \le 5.5$, that is, $p_1 = 1.0$, $p_2 = 1.5$, $p_3 = 2.0$, …, $p_{10} = 5.5$. In Figs. 4 and 5, the curves for model A are shown. For these cases, we can see that model A is fastest of all. However, it is unknown whether model A is effective or not in the general case.

4. Conclusions

This paper proposed multi-layered neural networks with learning parameters for output functions and it was shown that the proposed model was faster in learning time than the conventional one in compared with two types of simulations: XOR problem and functional approximation. Further, based on the results, we presented an effective heuristic model. The theoretical analysis is the future works.

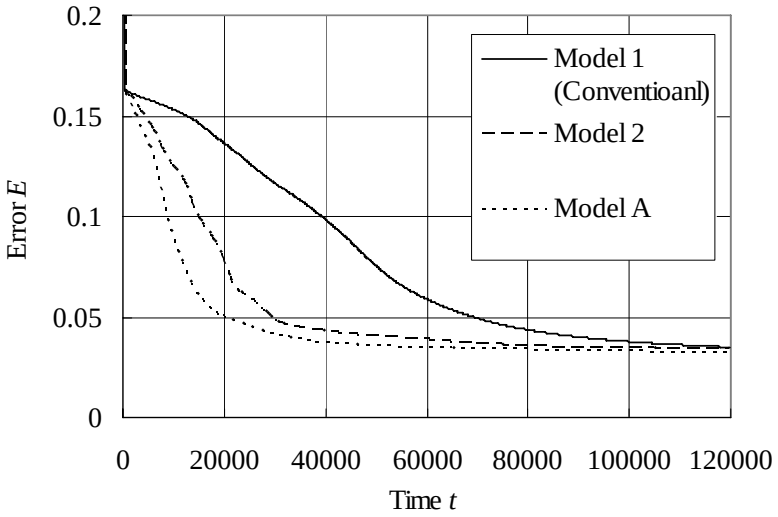


Fig. 3: Transition of learning error for Eq.(12).

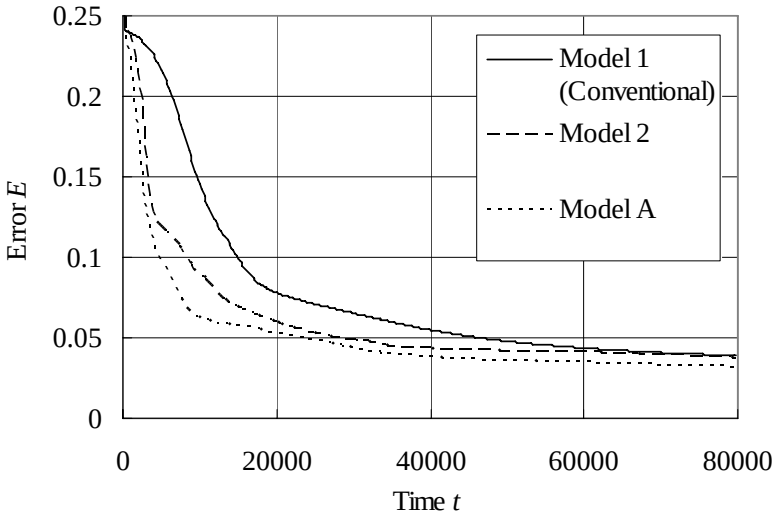


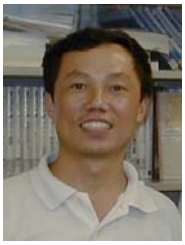
Fig. 4: Transition of learning error for Eq.(13).

References

[1] M.M. Gupta, L. Jin and N. Homma, *Static and Dynamic Neural Networks*, IEEE Press, 2002.

[2] J. Hertz, A. Krogh and R.G. Palmer, *Introduction to the Theory of Neural Computation*, Perseus Books Publishing, 1991.

[3] R. Reed, "Pruning Algorithms-A Survey", IEEE Trans. on Newural Networks, vol.4, no.5, pp.740-747, 1993.



Lixin Ma received the D.E. degree in System Information Engineering from Kagoshima University, Japan, in 1999. He is currently a Professor in Shanghai University of Electric Power, China. His research interests are neural networks and its industrial applications.

Hiromi Miyajima received the B.E. degree in Electrical Engineering from Yamanashi University, Japan, in 1974, and the M.E. and D.E. degrees in Electrical and Communication Engineering from Tohoku University, in 1976 and 1979, respectively. He is currently a Professor in the Department of Electrical and Electronic Engineering at Kagoshima University. His research interests are neural networks, fuzzy modeling, agent systems and parallel computing.

Noritaka Shigei received the B.E., M.E. and D.E. degrees from Kagoshima University, Japan, in 1992, 1994, and 1997, respectively. He is a Research Associate in the Department of Electrical and Electronic Engineering at Kagoshima University. His research interests are neural networks, genetic algorithms, digital circuit design, and parallel computing systems.