A Structure Trainable Neural Network with Embedded Gate Units: Multi-Dimensional Input/Output and Its Learning Algorithm

Kenji NAKAYAMA Akihiro HIRANO Makoto KOURIN

Dept of Information and Systems Eng., Faculty of Eng., Kanazawa Univ.
2–40–20 Kodatsuno, Kanazawa, 920–8667, Japan
e-mail: nakayama@t.kanazawa-u.ac.jp

Abstract

In this paper, a synthesis and learning method for the neural network with embedded gate units and a multi-dimensional input is proposed. When the input is multi-dimensional, gate functions are controlled in a multi-dimensional space. In this case, a hypersurface, on which the gate function is formed should be optimized. Furthermore, the switching points should be considered on the unit input. An initialization and a control methods for gate functions, which optimize the hypersurface, the switching point and the inclination, are proposed. The stabilization methods, already proposed, are further modified to be applied to the multi-dimensional environment. The gate functions can be trained together with the connection weights. Discontinuous function approximation is demonstrated to confirm usefulness of the proposed method.

1 Introduction

Multilayer neural networks (MLNNs) have been applied to a wide variety of fields. They include prediction, diagnosis, analysis, pattern classification, function approximation and so on [1]. An essential function used in all applications is pattern mapping. In order to realize the neural networks, several design methods have been proposed to minimize network size [2]-[6].

An important point is a rule, which governs the pattern mapping. If different rules are involved in a problem, it is difficult to solve it by using only a single MLNN, which consists of linear connections and continuous activation functions.

Modular neural networks are useful approaches to this kind of problem from performance view point [7],[8]. However, this method requires an independent expert network for each rule. One expert network is only used for a single mapping rule, and cannot be shared by different rules. Therefore, from network size view point, this approach is not elegant. In order to share the same network by different mapping rules, gate units are embedded in the neural networks [9]. A single neural network changes its structure with respect to the input data, and works just as different networks.

A simultaneous learning algorithm for connection weights and gate functions has been proposed [10]. The gate function is formed with a sigmoid function, with sharp inclination. The switching point of the gate functions are trained together with the connection weights. Some stabilization techniques have been proposed. However, the input is limited to one dimensional.

In this paper, the above method is expanded to multi-dimensional input neural networks. In this case, the gate functions should be optimized in the multi-dimensional space. New methods for determining the initial guess, controlling the slope and the hypersurface of the gate functions are proposed. Computer simulation of discontinuous function approximation will be shown in Sec.7 to confirm usefulness of the proposed method.

2 Network Architecture

2.1 Network Structure

Figure 1 shows the neural network with embedded gate units and a multi-dimensional input. The input potential u_j and the output y_j of the hidden units are given by

$$u_j = \sum_{i=0}^{I} w_{ji} x_i, \quad x_0 = 1 \tag{1}$$

$$y_j = f_h(u_j), \quad 1 \le j \le J \tag{2}$$

x_0 is bias and $f_h()$ is an activation function in the hidden layer. Transmission of the output y_j is controlled by the

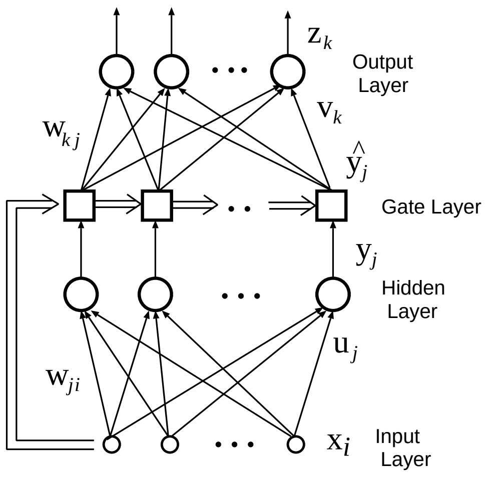


Figure 1: Structure variable neural network with embedded gate units.

gate units as follows:

$$g_j = f_{gj}(\boldsymbol{x}) \tag{3}$$

$$\hat{y}_j = g_j y_j \tag{4}$$

The gate unit output g_j takes 1 or 0 depending on the input $\boldsymbol{x}$. Therefore, the hidden units are selected based on the input data. The input potential v_k and the output of the output units are given by

$$v_k = \sum_{j=0}^{J} w_{kj}\hat{y}_j, \quad \hat{y}_0 = 1 \tag{5}$$

$$z_k = f_o(u_k), \quad 1 \le k \le K \tag{6}$$

$\hat{y}_0$ is bias. $f_o()$ is an activation function in the output layer.

2.2 Gate Units

The gate function $f_{gj}()$ is trained together with the connection weights w_{ji} and w_{kj}. For this purpose, the following function is employed.

$$f_{gj}(\boldsymbol{x}) = \frac{1}{1 + e^{(b_j h(\boldsymbol{x}) + c_j)}} \tag{7}$$

An inclination and a switching point are determined by b_j and c_j/b_j, respectively. To realize a switching function, b_j must be large. However, a large b_j will cause s ome difficulty in a learning phase, that is slow convergence or local minimum. It will be controlled in the learning process from a small value to a large value gradually. Since we do not know discontinuous points in mapping rule, that is switching points of the gate functions, c_j must be automatically adjusted for each application.

3 Simultaneous Learning Algorithm

The learning algorithm is based on the gradient decent method. A cost function is given by

$$E = \frac{1}{K}\sum_{k=1}^{K}(d_k - z_k)^2 \tag{8}$$

d_k is the target. The correction term is determined by the partial derivative. Letting $p(n)$ be a parameter at the nth iteration, it is updated by

$$p(n+1) = p(n) - \eta\frac{\partial E}{\partial p(n)} \tag{9}$$

Learning the gate function is similar to the activation function training [5],[6]. Compared with the above, the hidden unit outputs are replaced by the gate unit outputs in the new structure. From Eqs.(33), (3) and (4), the gate output is expressed by

$$\hat{y}_j = f_{gj}(\boldsymbol{x}) f_h(u_j) = \frac{1}{1 + e^{(b_j h(\boldsymbol{x}) + c_j)}} \frac{1}{1 + e^{-u_j}} \tag{10}$$

Here, the sigmoid function is used for the activation function $f_h()$. The trainable activation functions proposed in [5],[6] have the following form.

$$y = f(u) = \sum_{l=1}^{L}\{\frac{a_l}{1 + e^{-b_l u - c_l}} + d_l\} \tag{11}$$

A learning method for adjusting both the connection weights and this activation function was also proposed. The parameters a_l, b_l, c_l, d_l are trained together with the connection weights w_{ji} and w_{kj}. Compared with Eq.(11), Eq.(10) has another function

$$\frac{1}{1 + e^{-u_j}}, \tag{12}$$

which can be regarded as a constant in adjusting b_j and c_j.

On the other hand, in calculating the partial derivative of E with respect to w_{ji}, the other function

$$\frac{1}{1 + e^{(b_j h(\boldsymbol{x}) + c_j)}} \tag{13}$$

can be regarded as a constant. Therefore, the update formula proposed for the trainable activation functions can be basically applied [5],[6].

Connection Weight Update

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n+1) \tag{14}$$

$$\Delta w_{kj}(n+1) = \eta\phi_k y_j + \alpha\Delta w_{kj}(n) \tag{15}$$

$$\phi_k = (d_k - z_k) z_k(n)[1 - z_k(n)] \tag{16}$$

$$w_{ji}(n+1) = w_{ji}(n) + \Delta w_{ji}(n+1) \tag{17}$$

$$\Delta w_{ji}(n+1) = \eta\phi_j x_i + \alpha\Delta w_{ji}(n) \tag{18}$$

$$\phi_j = \hat{y}_j(n)[1 - \hat{y}_j(n)]\sum_{k=1}^{K}\phi_k w_{kj}(n) \tag{19}$$

Swtching Point Update

$$
\begin{aligned}
e_k &= d_k - z_k & (20)\\
\xi_j &= \sum_{k=1}^{K} e_k w_{kj}(n) z_k (1-z_k) & (21)\\
c_j(n+1) &= c_j(n) + \Delta c_j(n+1) & (22)\\
\Delta c_j(n+1) &= \eta_c \xi_j g_j (1-g_j) + \alpha_c \Delta c_j(n) & (23)
\end{aligned}
$$

η_c and α_c are a learning rate and a momentum term.

4 Gate Functions for Multi-Dimensional Input

4.1 Hypersurface of Gate Function

The gate function $f_{gj}()$ is defined by Eq.(7). How to determine $h(\boldsymbol{x})$ is discussed here. Since the gate units are assigned to the hidden units, one way is to set the hypersurface, on which the gate function is formed, to be the same as that of the activation function given by

$$u_j = \sum_{i=0}^{I} w_{ji} x_i \qquad (24)$$

$h(\boldsymbol{x})$ is formulated as

$$
\begin{aligned}
\hat{w}_{ji} &= \frac{w_{ji}}{\sum_{i=1}^{I} |w_{ji}|} & (25)\\
\hat{u}_j &= \sum_{i=1}^{I} \hat{w}_{ji} x_i & (26)\\
f_{gj}(\hat{u}_j) &= \frac{1}{1+e^{-(b_j \hat{u}_j + c_j)}} & (27)\\
g_j &= f_{gj}(\hat{u}_j) & (28)\\
\hat{y}_j &= g_j y_j & (29)
\end{aligned}
$$

A ratio among w_{ji} determine the hypersurface, and absolute value of w_{ji} affects inclination of the gate function, which should be controlled by b_j. Therefore, w_{ji} are normalized in the above formulation, which determine only the hypersurface. One example is shown in Fig.2. The activation function and the gate function are formed on the the same hypersurface. The inclination of the gate function is very sharp.

4.2 Compensation of Switching Point

The switching point of the gate function locates on the $\hat{u}_j$ axis, and should be located within the range, where $\hat{u}_j$ is distributed. However, the range of $\hat{u}_j$ changes as the connection weights are adjusted. Therefore, the switching point is compensated for as follows:

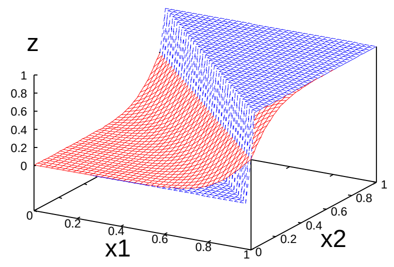


Figure 2: Relation between activation function and gate function. They are formed on the same hyperplane.

$$\hat{u}_{jmax}(n) = \sum_{i=1}^{I} \hat{w}_{ji}, \qquad \hat{w}_{ji} \geq 0 \qquad (30)$$

$$\hat{u}_{jmin}(n) = \sum_{i=1}^{I} \hat{w}_{ji}, \qquad \hat{w}_{ji} < 0 \qquad (31)$$

$$\frac{\hat{u}_{sp}(n) - \hat{u}_{jmin}(n)}{\hat{u}_{jmax}(n) - \hat{u}_{jmin}(n)} = \frac{\hat{u}_{sp}(n-1) - \hat{u}_{jmin}(n-1)}{\hat{u}_{jmax}(n-1) - \hat{u}_{jmin}(n-1)} \qquad (32)$$

This process is illustrated in Fig.3. The updating process is as follows: The connection weights $\boldsymbol{w}(n-1)$ and the switching points $\hat{u}_{sp}(n-1)$ are updated at $n-1$ iteration. $\hat{u}_{jmin}$ and $\hat{u}_{jmax}$ are calculated, and the new switching point denoted $\hat{u}_{sp}(n)$ is obtained above. $\hat{u}_{sp}(n-1)$ are replaced by $\hat{u}_{sp}(n)$, which are further updated at n iteration.

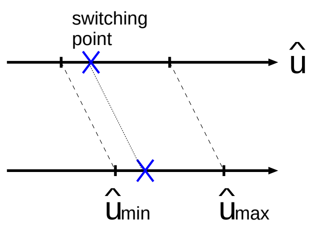


Figure 3: Compensation of switching point tracking change of $\hat{u}_j$ rang due to connection weight adjustment.

5 Learning Process

5.1 Multi-Stage Learning Process

In the multi-dimensional input network, determining the initial guess for the gate functions is rather difficult. So, the following multi-stage learning process is proposed.

Step1

To find the initial guess for the gate functions, the network without the gate units is used to learn the given problem. The sigmoid functions are used to emulate the gate functions. After convergence, information about the gate functions are extracted from the activation functions, that is the switching points, the hypersurface and the inclination as will be described in Sec.5.2. Random gate units and flat gate units are also added to make the network has more freedom.

Step2

The connection weights and the gate functions are simultaneously updated. The inclination b_j is adjusted by the annealing way, which will be described later. The network shown in Fig.1 is used.

Step 3

In this step, the hypersurface of the gate functions are assumed to be optimized, and are fixed. Since the connection weights are adjusted more, the hypersurface of the activation functions can move toward different direction. The switching points are further slightly adjusted.

5.2 Initial Guess for Gate Functions

Initial Guess of Inclination

In Step1, the activation functions are expressed as

$$y_j = f_h(u_j) = \frac{1}{1+e^{-u_j}} \tag{33}$$

Here, the following is defined,

$$\tilde{u}_j = \sum_{i=1}^{I} w_{ji} x_i \tag{34}$$

Equation (33) is rewritten using $\tilde{u}_j$,

$$y_j = f_h(u_j) = \frac{1}{1+e^{-\tilde{u}_j - w_{j0}}} \tag{35}$$

Furthermore, $\tilde{u}_j$ is related to $\hat{u}_j$ as,

$$\tilde{u}_j = \sum_{i=1}^{I} |w_{ji}| \hat{u}_j \tag{36}$$

From these relations, Eq.(33) is finally expressed using $\hat{u}_j$ as follows:

$$y_j = f_h(u_j) = \frac{1}{1+e^{-\sum_{i=1}^{I}|w_{ji}|\hat{u}_j - w_{j0}}} \tag{37}$$

Comparing Eq.(7), the inclination of $f_h(u_j)$ on the $\hat{u}_j$ axis is determined by $\sum_{i=1}^{I}|w_{ji}|$. Therefore, the initial inclination of the gate function is determined using $\sum_{i=1}^{I}|w_{ji}|$ after Step1.

Initial Guess for Switching Points

From Eq.(7), the switching point is given by $-c_j/b_j$. Comparing Eq.(37), the following relation can be held.

$$-\frac{c_j}{b_j} = -\frac{w_{j0}}{\sum_{i=1}^{I}|w_{ji}|} \tag{38}$$

Therefore, c_j satisfying the above relation is used for the initial guess of the switching point.

However, if the initial switching point obtained above locates outside the range of $\hat{u}_j$, it does not work well. In this case, the switching point, which randomly located or the center of the $\hat{u}_j$ range, is used instead.

6 Stabilization of Learning Process

6.1 Control of Gate Function Inclination

The inclination of the gate function $f_{gj}()$ must be sharp, in order to realize discontinuous function. However, the optimum switching points are not known exactly before hand, even though the initial guess is estimated in Step1. The switching points are optimized by adjusting c_j in a learning process. In this phase, if the inclination is very sharp, that is b_j has a very large value, the partial derivative of $f_{gj}()$ is very small in a wide range of u_j. This causes very slow convergence, and the gate function cannot move toward the optimum switching point.

For this reason, in the proposed method, the inclination b_j is adjusted in an annealing way. It is controlled from a relatively large value to a very large value gradually in a learning process. Change of the inclination is shown in Fig.4.

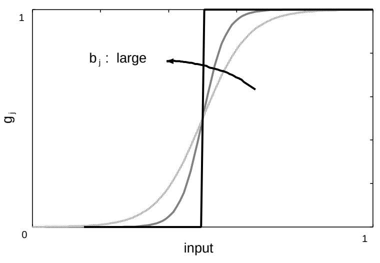


Figure 4: Annealing of inclination of gate functions in learning process.

6.2 Re-initialization of Connection Weights

In the proposed method, the sigmoid functions are used in the hidden units and the gate units. This means the hidden units can play a role of the gate units instead. However, the role of the hidden units is to approximate the mapping rule in each section. For this purpose, the inclination of both functions are controlled.

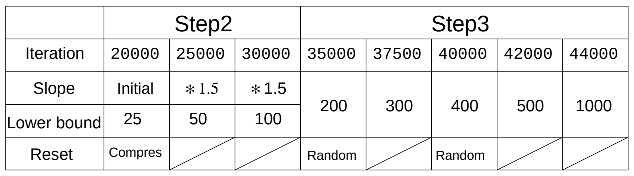
The inclination of the sigmoid function with fixed coefficients is determined by absolute value of the connection weights $|w_{ji}|$. If they have a large value, the inclination, that is, $\partial y_j/\partial \boldsymbol{x}$ becomes sharp, which can work as the gate function. If the hidden units play as a role of switching, the gate units cannot move toward the optimum switching points. In order to avoid this problem, the connection weights w_{ji} are reset to small random numbers or scaled down to small numbers. This means the learning of the connection weights is restarted using small numbers at some intervals.

In Step2, the connection weights are compressed so as to maintain the hypersurface of the gate functions. Like this, the connection weights used for the gate functions are normalized, therefore, this re-initialization of the connection weights does not affect the gate functions. On the other hand, in Step3, the hypersurface of the activation functions and the gate functions are independent, then random small numbers are used for re-initializing the connection weights.

7 Simulation and Discussions

Problem
Discontinuous function approximation is taken into account to evaluate the proposed method. The target is generated using the same network shown in Fig.1 with random parameters.

Control and Reinitialization of Gate Functions
Figure 5 shows scheduling of inclination control and reinitialization of the gate functions.



	Step2			Step3				
Iteration	20000	25000	30000	35000	37500	40000	42000	44000
Slope	Initial	$*1.5$	$*1.5$	200	300	400	500	1000
Lower bound	25	50	100					
Reset	Compres			Random		Random		

Figure 5: Schedule of inclination control and re-initialization of gate functions.

Parameters
Number of input units including bias: 3 units
Number of hidden units: 4 units in Step1, 6 units in Step2 and Step3
Number of output units: 1 unit
Learning rate for connection weights: $\eta = 0.1$
Learning rate for gate functions: $\eta = 0.1$
Number of training data: 1000
Number of maximum iterations: 50,000
Initial guess for connection weights: Random numbers in [-0.1, 0.1]
Scaling factor of compression reset: 0.1

Simulation Results
Figure 6 shows the target. The result of Step1 is shown in Fig.7, which roughly approximates the target. Using this result, the gate functions are initialized. Figure 8 shows the learning curve. The connection weights are reinitialized at 20,000, 35,000 and 40,000 iterations. The mean square error (MSE) defined by the following equation is well reduced.

$$E = \frac{1}{N_p}\sum_{p=1}^{N_p}\left(\frac{1}{K}\sum_{k=1}^{K}(d_k - z_k)^2\right) \qquad (39)$$

Furthermore, transition of switching points and inclination of the gate functions are shown in Figs.9 and 10, respectively. In Fig.9, the switching points are adjusted from 20,000 iterations. Before that, the initial guess are only shown as constant, which has no meaning. The inclination control is different for each gate function. Finally, the approximation result is shown in Fig.11. It is almost the same as the target.

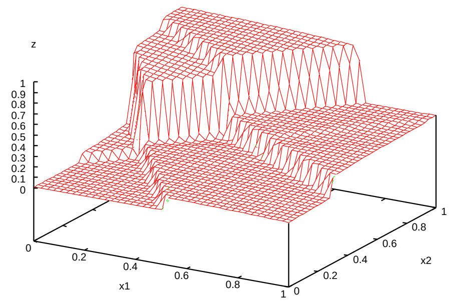


Figure 6: Target of discontinuous function approximation.

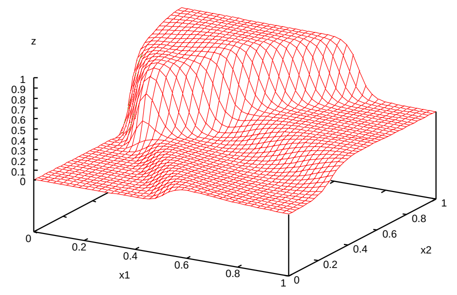


Figure 7: Approximation result in Step1.

8 Conclutions

In this paper, a synthesis and learning method for the neural network with embedded gate units and a

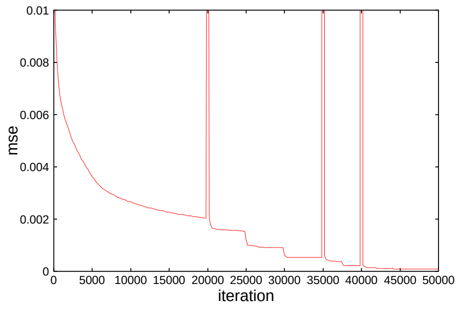


Figure 8: Learning curve.

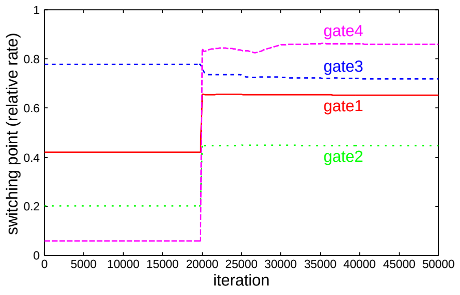


Figure 9: Transition of switching point of gate functions. They are adjusted after 20,000 iterations.

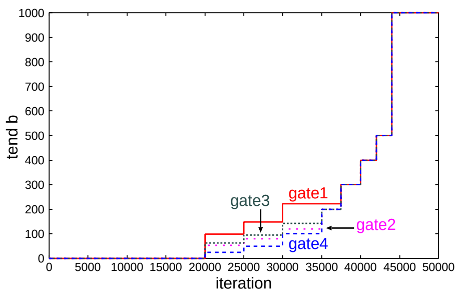


Figure 10: Transition of inclination of gate function.

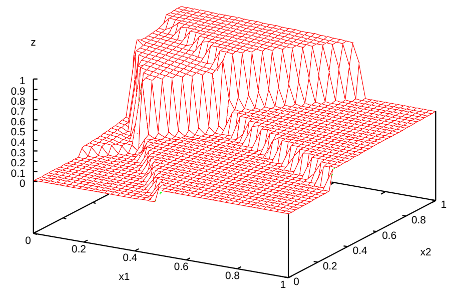


Figure 11: Approximation result in Step3.

multi-dimensional input has been proposed. In multi-dimensional input, gate functions are controlled in a multi-dimensional space. The hypersurface, on which the gate function is formed, is optimized. The switching points should be considered on the u_j axis, which is the input potential of the hidden units. The initialization and stabilizing methods for the gate functions including the hypersurface, the switching point and the inclination, have been proposed. The gate units can be trained together with the connection weights. Discontinuous function approximation has been demonstrated to confirm usefulness of the proposed method.

References

[1] S.Haykin, Neural Networks, Macmillan College Publishing Co., New York 1994.

[2] J.Sietsma and R.J.F.Dow, "Neural net pruning-Why and how," Proc. IEEE ICNN'88, pp.325-333, 1988.

[3] J.Sietsma and R.J.F.Dow, "Creating artificial neural networks that generalize," INNS Neural Networks, vol.4, pp.67-79, 1991.

[4] K.Nakayama and Y.Kimura, "Optimization of activation functions in multilayer neural network," Proc. IEEE ICNN'94, Orlando, pp.431-436, June 1994.

[5] K.Nakayama and M.Ohsugi, "A simultaneous learning method for both activation functions and connection weights of multilayer neural networks", Proc. IEEE INNS IJCNN'98, Anchorage, pp.2253-2257, May 1998.

[6] K.Nakayama, A.Hirano and I.Ido, "A multilayer neural network with nonlinear inputs and trainable activation functions: Structure and simultaneous learning algorithm", Proc. IEEE INNS IJCNN'99, Washington,DC, pp.1657-1661, July 1999.

[7] R.A.Jacobs et al, "Adaptive mixture of local exparts", Neural Computation, vol.3, pp.79-87, 1991.

[8] R.A.Jacobs, M.I.Jordan and A.G.Barto, "Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks", Cognitive Science vol.15, pp.219-250, 1991.

[9] J.Murata, T.Nakazono and K.Hirasawa, "Neural Networks with Node Gates", Proc. 1999 Int. Conf. Electrical Engineering, pp.358-361, 1999.

[10] K.Nakayama, A.Hirano and A.Kanbe, "A structure trainable neural network with embedded gating units and its learning algorithm", IEEE INNS, Proc. IJCNN'2000, Como, Italy, pp.III-253-258 July, 2000.