A Structure Trainable Neural Network with Embedded Gating Units and Its Learning Algorithm

Kenji Nakayama Akihiro Hirano Aki Kanbe

Dept. of Elec and Comp Eng., Kanazawa Univ., 920-8667, Japan
E-mail:nakayama@t.kanazawa-u.ac.jp

Abstract

Many problems solved by multilayer neural networks (MLNNs) are reduced into pattern mapping. If the mapping includes several different rules, it is difficult to solve these problems by using a single MLNN with linear connection weights and continuous activation functions.

In this paper, a structure trainable neural network has been proposed. The gate units are embedded, which can be trained together with the connection weights. Pattern mapping problems, which include several different mapping rules, can be realized using a single new network. Since, some parts of the network can be commonly used for different mapping rules, the network size can be reduced compared with the modular neural networks, which consists of several independent expert networks.

Keywords !] Multilayer neural networks, Modular neural networks, Pattern mapping, Gate units, Structure learning

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. An essential function used in all applications is pattern mapping. The input data are usually observed data, and the output data are determined based on each application. For example, the latter data include the next coming data in the predictors, symptom in the diagnosis, code of the categories in the pattern classification, and so on. An important point is a rule, which governs the pattern mapping. If different several rules are involved in the same pattern mapping problem, it is difficult to solve this problem by using a single MLNN with linear connections and continuous activation functions. The mapping rule can only be gradually changed with respect to the input data in these neural networks.

Modular neural networks are useful approaches to this kind of problems from performance view point [1],[2]. However, this method requires independent expert network for each rule. One expert network can only be used for one mapping rule. The expert network cannot be shared among different rules. Therefore, from network size view point, this approach is not an elegant approach.

In this paper, we propose a structure trainable neural network with embedded gate units. The gate unit works as switching, whose state is controlled by the input data. A single neural network changes its structure with respect to the input data. It works just as different networks. Some parts of the network can be shared by the different rules, resulting in saving network size.

Computer simulation of discontinuous function approximation will be shown in Sec.4 to confirm usefulness of the proposed method.

2. Network Architecture

2.1. Network Structure

Figure 1 shows the proposed neural network with embedded gate units.

The input data $\boldsymbol{x} = [x_1, x_2, .., x_I]$ are transmitted to the hidden layer, at the same time, to the gate units. The input potential u_j and the output y_j of the hidden units are given by

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

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

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 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

$$v_k = \sum_{j=0}^{J} w_{kj}\hat{y}_j, \qquad \hat{y}_0 = 1 \tag{5}$$
$$z_k = f_o(u_k), \qquad 1 \leq k \leq K \tag{6}$$

$\hat{y}_0$ is a bias unit. $f_o()$ 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.

$$f_{gj}(\boldsymbol{x}) = \frac{1}{1 + e^{b_j(\boldsymbol{x}-\boldsymbol{c}_j)}} \tag{7}$$

An inclination and a switching point are determined by b_j and$\boldsymbol{c}_j$, respectively. To realize switching function, b_j must be large. However, a large b_j will cause some difficulty in a training phase. It will be controlled in the learning process from a small value to a large value gradually. This process is similar to an annealing process.

Since we do not know discontinuous points in mapping rule, that is switching points of the gate units, $\boldsymbol{c}_j$ must be automatically adjusted for each application.
Type of Gate Functions:
Three kinds of gate functions are mixed.
TypeA: Flat gate, that is, $f_{gj}(\boldsymbol{x}) = 1$.
TypeB+: Monotonically increasing function, that is $b_j > 0$.
TypeB-: Monotonically decreasing function, that is, $b_j < 0$.
The flat gate transmit the hidden output in all input range. Therefore, the related hidden units are used in all mapping rules.

3. Learning Algorithm

3.1. Gradient Decent 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.

3.2. Simaltenious Learning

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

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

In this network, the gate function is learned. It is similar to the activation function training [6],[7]. Compared with the above, the hidden unit outputs are replaced by the gate unit outputs in the new structure. From Eqs.(2), (3) and (4), the gate output is expressed by

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

Here, the sigmoid function is used for the activation function $f_h()$. The trainable activation functions proposed in [6],[7], 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 $\boldsymbol{c}_j$.

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

$$\frac{1}{1+e^{b_j(\boldsymbol{x}-\boldsymbol{c}_j)}} \tag{13}$$

can be regarded as a constant. Therefore, the update formula proposed for the trainable activation functions can be basically applied [6],[7].

3.3. Control of Inclination of Sigmoid Functions

In the proposed method, the sigmoid functions are used in the hidden layer and the gate layer. 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 interval. For this purpose, the inclination of both functions are controlled.

Hidden units:

The inclination 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 like the initial guess. This means the learning of the connection weights is restarted using small random numbers at some interval. At this reset, the gate units succeed the previous results.

Gate units:

On the contrary, 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 before hand. Therefore, the gate functions will be set initially on equal spaced points along the input data axis. The switching points are optimized by adjusting $\boldsymbol{c}_j$ in a learning process. In this phase, if the inclination is very high, that is b_j has a very large value, the partial derivative of $f_g()$ is very small in a wide range of the input data. 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 controlled in an annealing way. It is controlled from relatively large value to very large value gradually in a learning process.

4. Simulation and Discussions

Discontinuous Function Apploximation:

Function: The function to be approximated is shown in Fig.2. INPUT and OUTPUT mean $\boldsymbol{x}$ and

z_k, respectively. In this case, one dimensional input and one output are used. It is assumed that training data are given at random, thus, the figure of the function is not known before hand. 500 training data are randomly selected on the lines shown in Fig.2.

Network: The network used here consists of one input unit, 8 hidden units, 8 gate units and one output unit. It has the connection weights from the input unit to the hidden layer $w_{ji}, i = 1$, and from the gate layer to the output unit $w_{jk}, k = 1$.

Initial Guess: The initial guess of the connection weights are uniformly distributed during -0.1 !A0.1. At the reset, the connection weights are set to random numbers distributed in the same way.

The inclination of the gate units is changed so that $b_j = 100, 150, 200, 300$ at the beginning, 5000, 20000 and 40000 iterations in the learning process. The final value is $b_j = 300$. The connection weights are reset at the same iterations. The switching points are initially set at the equally spaced points. However, the just switching points, that is $x = 0.33, 0.66$ are avoided. The date are listed in Table1. Type of the gate functions are also shown in Table1. Besides them, two flat gate units are used.

Simulation Results: Figure 3 shows a learning curve. The mean squared error (MSE), that is E given by Eq.8, suddenly increases at the iterations, where the connection weights are reset. Figure 4 shows the switching point change in the learning process. The gate3 and the gate5 moved and settled down at the optimum switching points, 0.33 and 0.66. The other gates trapped at irregular points. However, as shown in Table1, the connection weights from these gate units to the output unit are small, except for the gate1. This means they can be pruned in the same way as the pruning methods for the hidden units [3],[4],[5].

Figures 5 shows the approximated function. The discontinuous parts are well approximated.

Comparison with Ordinary MLNN: Figures 6 and 7 show the learning curve and the approximated function by using the ordinary MLNN without the gate units. The network size is one input unit, 8 hidden units and one output unit. The convergence is fast, hoever, it cannot approximate around the discontinuous points, resulting in the large error. On the contrary, the proposed method can approximate the discontinuous function more precisely.

Network Reduction: The two flat gate units cover all the input range $0 \leq x \leq 1$, and the gate3 covers two sections. The hidden units and the connection weights related to these gates are commonly used in several intervals, where the mapping rules are different. This common usage of the elements can reduce the network size compared with the modular neural networks [1],[2], where the common usage is impossible.

5. Conclutions

In this paper, a structure trainable neural network has been proposed. The gate units are embedded, which can be trained together with the connection weights. Pattern mapping problems, which include several different mapping rules, can be realized using a single network. Since, some parts of the network can be commonly used for different mapping rules, the network size can be reduced compared with the conventional modular neural network, which consists of several independent expert networks.

References

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

[2] 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.

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

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

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

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

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

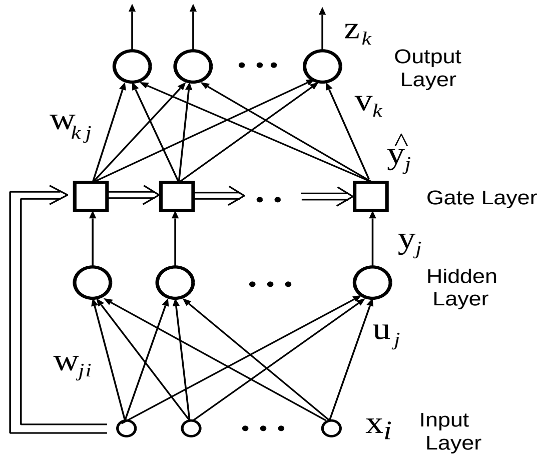

Figure 1: Structure variable neural network with embedded gate units.

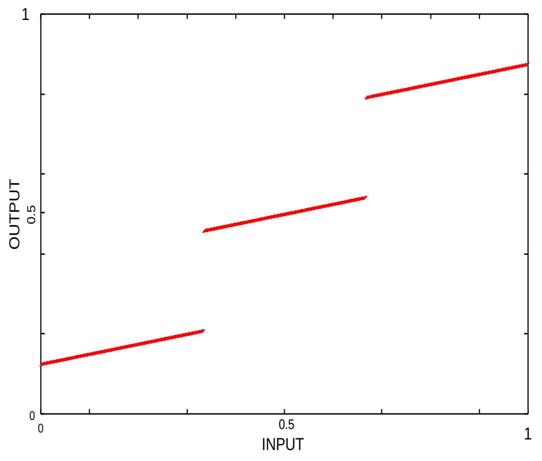

Figure 2: Discontinuous function to be approximated.

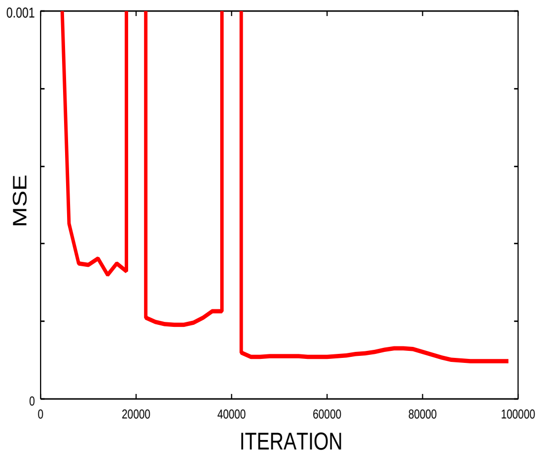

Figure 3: Learning curve in proposed method.

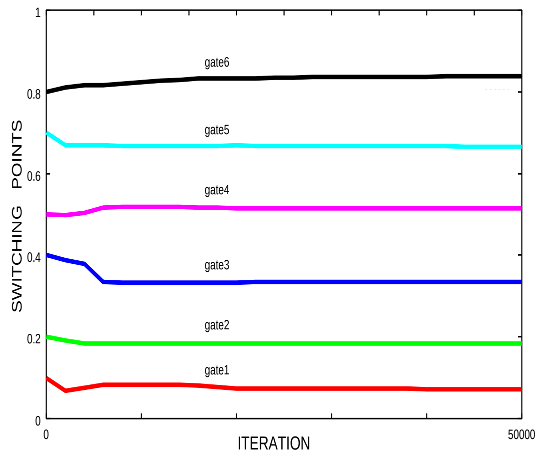

Figure 4: Switching point movement of gate units

Table 1: Type of gate functions, switching points (SP) and w_{kj}.

Gate No.	Type	Initial SP	Final SP	w_{kj}
1	B+	0.1	0.07	2.30
2	B-	0.2	0.18	-0.30
3	B+	0.4	0.33	1.31
4	B-	0.5	0.51	-0.05
5	B+	0.7	0.66	1.26
6	B-	0.8	0.84	-0.13

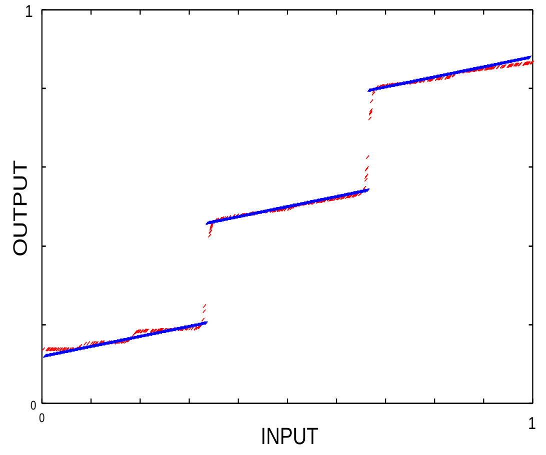

Figure 5: Function approximated by new network with gate units.

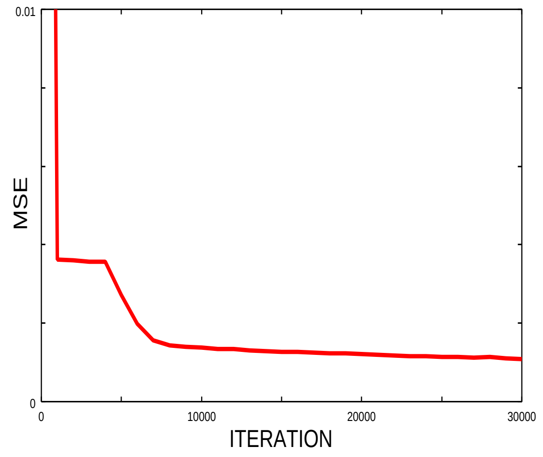

Figure 6: Learning curve in ordinary MLNN without gate units.

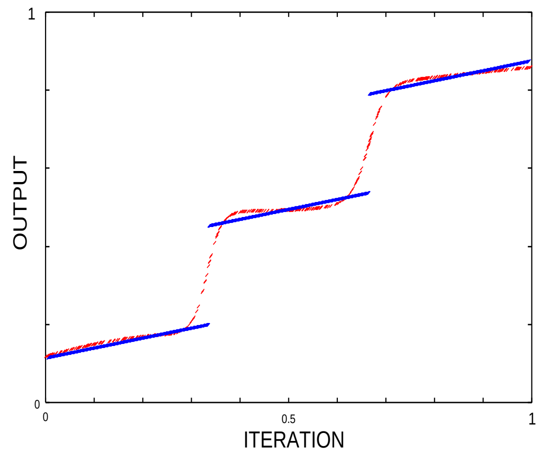

Figure 7: Function approximated by ordinary MLNN without gate units.