Paper

Universal Learning Networks with Branch Control

Non-member	Qingyu XIONG	(Kyushu University)
Member	Kotaro HIRASAWA	(Waseda University)
Member	Jinglu HU	(Kyushu University)
Member	Junichi MURATA	(Kyushu University)
Non-member	Dazi LI	(Kyushu University)

In this paper, Universal Learning Networks with Branch Control (ULNs with BC) is proposed. The point of the paper is to adjust the outputs of the intermediate nodes of the basic network using an additional branch control network. The adjustment means to multiply the nodes outputs by the coefficients ranging from zero to one, which is obtained from the branch control network. Therefore, the following are done in ULNs with BC, (1) the branch is cut when the coefficient of its branch is zero, and (2) multiplication is carried out in the nodes outputs adjustment when the coefficient takes a nonzero value. ULNs with BC is applied to a function approximation problem and a two-spirals problem. The simulation results show that ULNs with BC exhibits better performances than the conventional neural networks with comparable complexity.

Keywords: neural networks, fuzzy networks, universal learning networks, functions localization, multiplication

1. Introduction

In human's brain, it has been recognized that there are many special intellectual parts related to mathematical knowledge, musical knowledge, gymnastic knowledge and so on, and each intellectual part is activated depending on the action of each knowledge. This is called "functions are locally processed in the brain [1]".

But commonly used neural networks, especially layered neural networks, have the structure with all the nodes being connected, that is, the connection between the nodes never changes depending on the input values. Therefore, functions localization of the networks can not be realized in the conventional networks. Here, functions localization means that only parts of the networks are activated depending on the input values of the network.

To obtain such a brain-like model in artificial neural networks for solving the large-scale real-world problems, one of the most important things is how to divide a large problem into smaller and simpler subproblems; and how to guide the training algorithm to realize the functions localization.

In this paper, an artificial functions localized network is proposed using Universal Learning Networks with Branch Control (ULNs with BC) and it is studied why and how much ULNs with BC can improve the performance of the networks compared to the conventional neural networks. ULNs with BC is composed of two kinds of networks. One is a basic network, which is a conventional layered neural network. The other one is a branch control network, which controls the branches from the intermediate nodes to the output node of the basic network in a way that the outputs of the intermediate nodes are multiplied by the coefficients ranging from zero to one, which is calculated by the branch control network.

Therefore, ULNs with BC can realize an artificial functions localized network, where the basic network is divided into smaller and simpler networks by cutting some branches of the basic network, when the coefficients calculated by the branch control network become zero. The connection or disconnection of the branches of the basic network depends on the input values of the network. On the other hand, when the coefficients of the branches do not take a value of zero, the multiplication operation in the nodes outputs adjustment plays an important role for improving the performance of ULNs with BC.

Modular networks [2] [3] have similar module structures to ULNs with BC. But, the structure of the modular networks is different from the one of ULNs with BC. While the modules in modular networks are completely separated, the ULNs with BC has the module structure where the different modules have the identical nodes and branches, in other words, the modules in ULNs with BC are overlapped. In addition, the multiplication operations exist in ULNs with BC in the case of non-zero coefficient values like modular networks. Anyway, the proposed method does not necessarily affirm that the real brain has the branch control mechanism, it is just a model for functions localization found in the real brain.

The remainder of this paper is organized as follows: In Section 2, ULNs are reviewed briefly, because ULNs are used to construct a ULN with BC which uses various kind of node functions including multiplication. In Section 3, we present the basic structure and training al-

gorithm of ULNs with BC. In addition, the reasons why ULNs with BC have better representation and generalization ability than the conventional neural networks are also stated in Section 4. In Section 5, we give simulation conditions and simulation results both for a function approximation problem and a two-spirals problem. Finally, conclusions are presented in Section 6.

2. Universal Learning Networks

In this section, the structure of ULNs are explained briefly [4].

The purpose of ULNs is to provide a general framework for modeling and control of the complex systems widely found in the real world. It is generally recognized that any dynamical system can be described by using a set of related equations. Depending on prior knowledge available about the system, the equations may be fully known, partly known, or totally unknown. To model such dynamical systems, ULNs consisting of two kinds of element, nodes and branches, have been introduced.

As shown in *Fig.*1, Universal Learning Networks consist of a number of nodes and branches for interconnecting the nodes. The nodes may have any continuously differentiable nonlinear functions in them, and each pair of nodes can be connected with each other by multiple branches with arbitrary time delays. Multiple branches and their arbitrary time delays can be effective to model the dynamical systems in a very compact network.

The generic equation that describes the behavior of ULNs is expressed as follows:

$$h_j(t) = f_j(\{h_i(t - D_{ij}(p)) | i \in JF(j), p \in B(i,j)\}, \{\lambda_m | m \in M(j)\}, \{r_n(t) | n \in N(j)\}), \quad (1)$$
$$j \in J, \quad t \in T,$$

where

$h_j(t)$	:	output value of node j at time t;
$r_n(t)$	:	value of nth external input variable at time t;
λ_m	:	value of mth parameter;
f_j	:	nonlinear function of node j;
$D_{ij}(p)$	:	time delay of pth branch from node i to node j;
J	:	set of suffixes of nodes $\{j\}$;
$JF(j)$	:	set of suffixes of nodes which are connected to node j;
N	:	set of suffixes of external input variables $\{n\}$;
M	:	set of suffixes of parameters $\{m\}$
$B(i,j)$	:	set of suffixes of branches from node i to node j;
T	:	discrete set of sampling instants.

The ULNs operate on a discrete-time basis. Each pair of nodes i and node j may have multiple branches between them, i.e., set $B(i,j)$ may contain more than one elements. Functions $f_j(\cdot)$ that govern the operation of the nodes can be any continuously differentiable functions; typically, sigmoidal functions can be employed. In this case, *Eq.*(1) can be expressed specifically as follows:

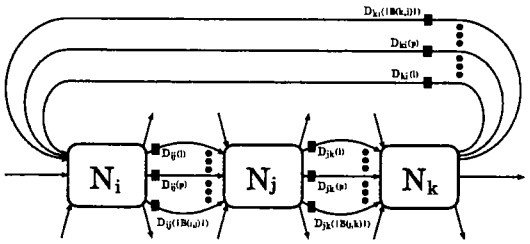


Fig. 1. The structure of Universal Learning Networks (ULNs)

$$h_j(t) = f_j(\alpha_j(t)) = z_j \frac{1 - e^{-\phi_j \alpha_j(t)}}{1 + e^{-\phi_j \alpha_j(t)}}, \quad \cdots\cdots\cdots\cdots \quad (2)$$

$$\alpha_j(t) = \sum_{i \in JF(j)} \sum_{p \in B(i,j)} \omega_{ij}(p) h_i(t - D_{ij}(p)) + \theta_j, \quad (3)$$

where

$\omega_{ij}(p)$	:	weight parameter of pth branch from node i to node j;
θ_j	:	threshold parameter of node j;
ϕ_j	:	slope parameter of node j;
z_j	:	gain parameter of node j.

Therefore, ULNs can be recognized as a framework for modeling more general systems in the sense that after being processed by each function at each node, the output signal of the node is transferred to another node through multiple branches with arbitrary time delays, then it is again processed by another node with its function. The static version of ULNs used in this paper is reduced to a simpler form without feedback loops(see reference 4).

3. Structure of Universal Learning Networks with Branch Control

3.1 Basic Structure of ULNs with BC ULNs with BC is composed of a basic network and a branch control network (see *Fig.*2). The relationship between the outputs of the branch control network and the branches from the intermediate nodes to the output node in the basic network is one to one correspondence. As a result, the output of the branch control network can control the connection of the corresponding hidden branches of the basic network with a coefficient of relative strength. When the coefficient R in *Fig.*2 takes zero value, the corresponding branch is deleted from the basic network, otherwise, the corresponding output of the intermediate node is multiplied by non-zero coefficient R.

In this paper, a commonly used three layered neural network is used as the basic network to study the comparison between ULNs with BC and neural networks, while a fuzzy inference network is used as the branch control network, because the localized property of fuzzy inference networks is used to execute the branch control effectively.

To be more concrete, the following notations are used for the three layered neural network and fuzzy inference

network.

• Neural Network

Supposing a static network whose inputs are h_i $(i \in N)$ and output is h_o, then, h_o can be described as follows according to $Eq.(1)$,

$$h_o = \sum_{j \in JF(o)} \omega_{jo} h_j R(Z_j) + \theta_o, \quad \cdots\cdots\cdots\cdots (4)$$

$$h_j = f_j(\alpha_j) = z_j \frac{1 - e^{-\phi_j \alpha_j}}{1 + e^{-\phi_j \alpha_j}}, \quad \cdots\cdots\cdots\cdots (5)$$

$$\alpha_j = \sum_{i \in JF(j)} \omega_{ij} h_i + \theta_j, \quad \cdots\cdots\cdots\cdots\cdots (6)$$

where $R(Z_j) \in [0, 1]$ is the coefficient between the intermediate node j and the output node, which depends on the output Z_j of the fuzzy inference network.

Here, multiple branches and time delays are not considered because the neural network is a static network. As being stated before, the functions localization can be realized in ULNs with BC by appropriately calculating $R(Z_j)$.

• Fuzzy Inference Network

In this paper, the following fuzzy inference network using Gaussian membership functions is used.

$$Z_j = \frac{\sum_{q \in Q}[f_q(h)\sigma_{jq}\mu_{jq}]}{\sum_{q \in Q}[f_q(h)\sigma_{jq}]}, \quad \cdots\cdots\cdots\cdots (7)$$

$$f_q(h) = exp[-\frac{(h - \mu_{hq})^T \Gamma_{hq}^{-1} (h - \mu_{hq})}{2}], \quad \cdots (8)$$

$$g_q(Z_j) = exp[-\frac{(Z_j - \mu_{jq})^2}{2\sigma_{jq}^2}], \quad \cdots\cdots\cdots\cdots (9)$$

where

Z_j : the output of the fuzzy inference network corresponding to the intermediate node j of the basic network;

$f_q(h)$: membership function of the IF part of the qth fuzzy rule;

$g_q(Z_j)$: membership function of the $THEN$ part of the qth fuzzy rule;

h = $(h_1, h_2, \cdots, h_N)$, input vector;

μ_{hq} : vector representing the center of the IF part of the qth fuzzy rule;

Γ_{hq}^{-1} : matrix representing the width of the IF part of the qth fuzzy rule;

μ_{jq} : parameter representing the center of the $THEN$ part of the qth fuzzy rule, $\mu_{jq} \in [0, 1]$;

σ_{jq}^2 : parameter representing the width of the $THEN$ part of the qth fuzzy rule;

Q : set of suffixes of fuzzy rules.

The most important coefficient R(Relative Strength) is calculated by $Eq.(10)$ as shown in $Fig.3$.

$$R(Z_j) = \begin{cases} \frac{Z_j - Z_j^o}{1 - Z_j^o} & \text{if } Z_j \geq Z_j^o \\ 0 & \text{if } Z_j < Z_j^o, \end{cases} \quad \cdots\cdots\cdots (10)$$

where

Z_j^o : threshold value to determine whether the intermediate node j is connected to the output node of the basic network;

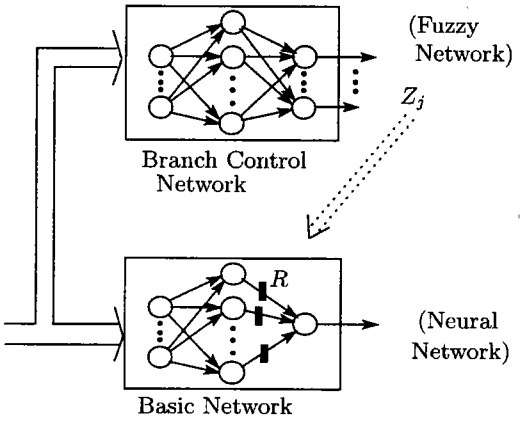


Fig. 2. Structure of ULNs with BC

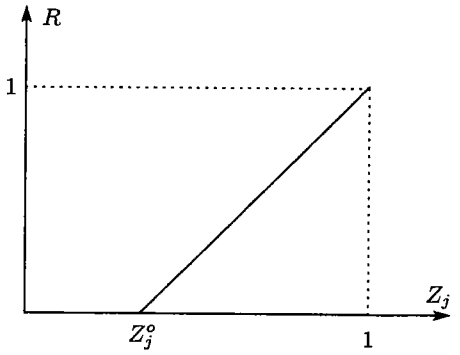


Fig. 3. Relative strength R calculated by Z_j

Therefore, if $Z_j < Z_j^o$, then the connection between the intermediate node j and output node of the basic network is cut, as a result, a functions localized network is realized. On the other hand, if $Z_j \geq Z_j^o$, then the nodes outputs adjustment is done, that is, the output h_j of node j is multiplied by $R(Z_j)$ as shown in $Eq.(4)$.

In this case, as being stated later, the multiplication plays an important role to improve the performance of ULNs with BC.

3.2 Training of ULNs with BC As for training of ULNs with BC, parameters of the basic network could be trained in the same way as the commonly used neural networks with supervised learning.

But, as branch connection with a coefficient of relative strength or disconnection of the basic network of ULNs with BC depends on the input values of the network, it is difficult to train the parameters by using the gradient method like back propagation algorithm.

So, a kind of random search method named RasID [5] was used to train the parameters of the basic network.

RasID is an abbreviation of Random Search with Intensification and Diversification, and executes the search for optimal parameters in a unified manner, where intensified search and diversified search are carried out iteratively, systematically and effectively using the information on success and failure of the past search.

Furthermore, as being explained later, it is noticeable that even though the parameters of ULNs with BC are trained in the following way, ULNs with BC shows distinguished performances: (1) only the parameters of the basic network are trained, (2) the parameters of IF parts of the fuzzy rules such as μ_{hq} and Γ_{hq}^{-1} are set at appropriate values in advance, and the parameters of $THEN$ parts of the fuzzy rules like μ_{jq} are randomly set in $[0,1]$.

As can be seen later, μ_{hq} and Γ_{hq}^{-1} are set in such a way that the whole input space is covered by each IF membership function $f_q(h)$ with an appropriate overlap.

The whole process of the training of ULNs with BC is arranged as follows:

Step 1 : Choose an initial value set for μ_{hq}, Γ_{hq}^{-1}, σ_{jq}, μ_{jq}, z_j and ϕ_j;
Given a threshold value Z_j^o;
Step 2 : Choose an initial value set for ω_{ij}, θ_j, ω_{jo}, and θ_o;
Step 3 : Calculate the output Z_j of the branch control network; (see $Eq.(7)-(9)$)
Compute the relative strength R;(see $Eq.(10)$)
Step 4 : Calculate the output h_o of the basic network; (see $Eq.(4)-(6)$)
Step 5 : Train the parameters of ω_{ij}, θ_j, ω_{jo}, and θ_o using $RasID$ algorithm;
Stop if a certain pre-specified condition is met, else, go to **Step 4**.

4. Theoretical Analyses of ULNs with BC

In this section, theoretical analyses of ULNs with BC are described in term of functions localization and multiplication processing.

4.1 Effects of Functions Localization The branch connectivity ρ of the basic network can be adjusted by setting the threshold Z_j^o at an appropriate value:

$$\rho = \frac{l}{L}, \quad \cdots\cdots\cdots\cdots\cdots\cdots \quad (11)$$

where l is the number of the connected branches with a coefficient of relative strength between the intermediate nodes and output node of the basic network, and L ($i.e.|JF(o)|$ in $Eq.(4)$) is the total number of hidden branches of the basic network. The branch connectivity ρ is increased when Z_j^o takes a small value, and vice versa.

When functions locally processed networks like ULNs with BC are used for many applications, it is generally expected that there exists a connectivity where the criterion function is minimized like $Fig.4$. This means

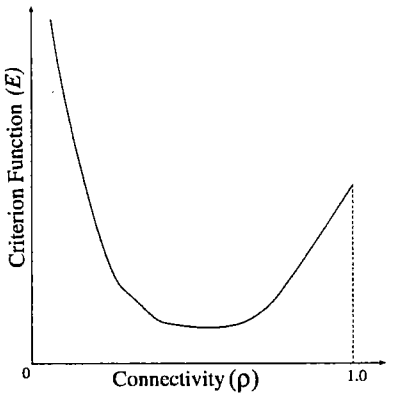


Fig. 4. Relation between connectivity and criterion function of the network

that higher performance can be obtained than the commonly used neural networks by using the network with some branches being cut depending on the input values of the network. This fact has been already verified by using Learning Petri Network [6]-[9] developed in our laboratory.

The reason why the optimal connectivity exists in the networks is that it is appropriate to use the compact partial networks rather than the total big network for the whole problem. In addition, it can be also stated that since the compact partial networks have a small number of parameters, parameter training of functions localized networks is expected to carry out smoothly.

4.2 Effects of Multiplication Operation In this subsection, it is discussed theoretically why the performance of ULNs with BC is improved by the multiplication operation, that is, by multiplying the output h_j of node j by relative coefficient $R(Z_j)$.

Supposing a static network, whose input are h_i $i \in N$, and output is h_o, then, the output deviation Δh_o caused by the changes of the inputs $\Delta h = (\Delta h_1, \Delta h_2, \cdots, \Delta h_N)$ can be calculated approximately by Taylor expansion like $Eq.(12)$.

$$\begin{aligned}\Delta h_0 \approx & \sum_{i1\in N} \frac{\partial^{\dagger} h_0}{\partial h_{i1}} \Delta h_{i1} \\ & + \frac{1}{2} \sum_{i1\in N} \sum_{i2\in N} \frac{\partial^{\dagger 2} h_0}{\partial h_{i1} \partial h_{i2}} \Delta h_{i1} \Delta h_{i2} \\ & + \cdots\cdots \\ & + \frac{1}{l!} \sum_{i1\in N} \cdots \sum_{il\in N} \frac{\partial^{\dagger l} h_0}{\partial h_{i1} \cdots \partial h_{il}} \Delta h_{i1} \cdots \Delta h_{il} \\ & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (12)\end{aligned}$$

Generally speaking, it is recognized that the networks with high representation ability can train the parameters easily so that the output deviation Δh_o can take any values from small to large when the change of the input vector Δh occurs.

Therefore, it is said from $Eq.(12)$ that if the higher order derivatives of the network such as $\frac{\partial^{\dagger} h_0}{\partial h_{i1}}$, $\frac{\partial^{\dagger 2} h_0}{\partial h_{i1} \partial h_{i2}}$,

$\cdots$, and $\frac{\partial^{\dagger l} h_0}{\partial h_{i1} \cdots \partial h_{il}}$ can take any values by training the parameters, the network shows high representation ability.

The higher order derivatives can be calculated by using the forward propagation algorithm of ULNs [4] as follows.

$$P_1(k, h_{i1}) = \frac{\partial^{\dagger} h_k}{\partial h_{i1}} = \sum_{j \in JF(k)} \left[\frac{\partial h_k}{\partial h_j} P_1(j, h_{i1}) \right] + \frac{\partial h_k}{\partial h_{i1}}, \quad (13)$$

where

h_k : the output value of node k

$$\frac{\partial h_k}{\partial h_{i1}} = \begin{cases} 1, & \text{if } k = i1 \\ 0, & \text{otherwise} \end{cases}$$

$$P_2(k, h_{i1}, h_{i2}) = \frac{\partial^{\dagger 2} h_k}{\partial h_{i1} \partial h_{i2}} = \sum_{j \in JF(k)} \left[\frac{\partial^{\dagger}(\frac{\partial h_k}{\partial h_j})}{\partial h_{i2}} P_1(j, h_{i1}) \right] + \sum_{j \in JF(k)} \left[\frac{\partial h_k}{\partial h_j} P_2(j, h_{i1}, h_{i2}) \right], \quad (14)$$

$$\vdots$$

$$P_l(k, h_{i1}, h_{i2}, \cdots, h_{il}) = \frac{\partial^{\dagger l} h_k}{\partial h_{i1} \cdots \partial h_{il}}$$

$$= \sum_{j \in JF(k)} \left[\frac{\partial^{\dagger l-1}(\frac{\partial h_k}{\partial h_j})}{\partial h_{i2} \cdots \partial h_{il}} P_1(j, h_{i1}) \right]$$

$$+ \sum_{j \in JF(k)} \left[\frac{\partial^{\dagger l-2}(\frac{\partial h_k}{\partial h_j})}{\partial h_{i3} \cdots \partial h_{il}} P_2(j, h_{i1}, h_{i2}) + \cdots\cdots + \frac{\partial^{\dagger l-2}(\frac{\partial h_k}{\partial h_j})}{\partial h_{i2} \cdots \partial h_{il-1}} P_2(j, h_{i1}, h_{il}) \right]$$

$$+ \sum_{j \in JF(k)} \left[\frac{\partial^{\dagger l-3}(\frac{\partial h_k}{\partial h_j})}{\partial h_{i4} \partial h_{i5} \cdots \partial h_{il}} P_3(j, h_{i1}, h_{i2}, h_{i3}) + \cdots\cdots + \frac{\partial^{\dagger l-3}(\frac{\partial h_k}{\partial h_j})}{\partial h_{i2} \partial h_{i3} \cdots \partial h_{il-2}} P_3(j, h_{i1}, h_{il-1}, h_{il}) \right]$$

$$+ \cdots\cdots$$

$$+ \sum_{j \in JF(k)} \left[\frac{\partial^{\dagger}(\frac{\partial h_k}{\partial h_j})}{\partial h_{i2}} P_{l-1}(j, h_{i1}, h_{i3}, \cdots, h_{il}) + \cdots\cdots + \frac{\partial^{\dagger}(\frac{\partial h_k}{\partial h_j})}{\partial h_{il}} P_{l-1}(j, h_{i1}, h_{i2}, \cdots, h_{il-1}) \right]$$

$$+ \sum_{j \in JF(k)} \left[\frac{\partial h_k}{\partial h_j} P_l(j, h_{i1}, h_{i2}, \cdots, h_{il}) \right]. \quad \cdots\cdots (15)$$

$\frac{\partial h_k}{\partial h_{i1}}$ in *Eq.*(13) takes a value of 1.0 or 0.0. Therefore, whether P_1 can take any values from small to large or not depends on whether $\frac{\partial h_k}{\partial h_j}$ in *Eq.*(13) can take any values or not. Here after, P_1, P_2, $\cdots$, and P_l etc. are used to denote $P_1(k, h_{i1})$, $P_2(k, h_{i1}, h_{i2})$, $\cdots$, and $P_l(k, h_{i1}, h_{i2}, \cdots, h_{il})$ etc., respectively.

In addition, $\frac{\partial^{\dagger}(\frac{\partial h_k}{\partial h_j})}{\partial h_{i2}}$ in *Eq.*(14) can be expressed by P_1, if $\frac{\partial h_k}{\partial h_j}$ is a function of the nodes [4]. Therefore, in that case whether P_2 in *Eq.*(14) can take any values or not also depends on whether $\frac{\partial h_k}{\partial h_j}$ can take any values or not. Likewise, it is clear from *Eq.*(15) that whether $\frac{\partial h_k}{\partial h_j}$ can take any values or not is essential to study whether P_l can take any values or not.

Let us study the following sigmoidal function:

$$h_k = z_k \frac{1 - e^{-\phi_k \alpha_k}}{1 + e^{-\phi_k \alpha_k}}, \quad \cdots\cdots (16)$$

$$\alpha_k = \sum_{j \in JF(k)} \omega_{jk} h_j + \theta_k. \quad \cdots\cdots (17)$$

In this case, the following holds:

$$\frac{\partial h_k}{\partial h_j} = \frac{\omega_{jk} \phi_k z_k}{2} (1 - (\frac{h_k}{z_k})^2). \quad \cdots\cdots (18)$$

Therefore, even when $|\omega_{jk}|$, ϕ_k and z_k are set at large values, it is difficult to train the sigmoidal network so that $|\frac{\partial h_k}{\partial h_j}|$ takes any value, because $|\alpha_k|$ should be small. Therefore, $\frac{\partial h_k}{\partial h_j}$ can not take any values from small to large in the sigmoidal networks.

On the other hand, when the following multiplication operation is carried out in the nodes,

$$h_k = z_k \prod_{j' \in JF(k)} (h_{j'} - \omega_{j'k}) + \theta_k \quad \cdots\cdots (19)$$

then,

$$\frac{\partial h_k}{\partial h_j} = z_k \prod_{j' \in JF(k)-j} (h_{j'} - \omega_{j'k}) \quad \cdots\cdots (20)$$

holds.

As a result, it is clear from *Eq.*(20) that $\frac{\partial h_k}{\partial h_j}$ can take any values in the network with multiplication.

5. Simulations of ULNs with BC

Simulations were carried out by adjusting the threshold Z_j^o in order to study the fundamental characteristics of ULNs with BC, that is, to study whether there exists an optimal connectivity as shown in *Fig.*4, even when parameters $\mu_{jq} \in [0, 1]$ of the fuzzy inference network are randomly set.

5.1 Simulation Conditions The parameters of the membership functions of the *IF* parts of the qth fuzzy rules $f_q(h)$ are fixed in such a way that the input space of the network is appropriately covered with

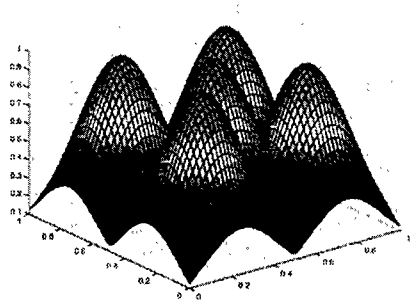


Fig. 5. Membership Functions of *IF* part

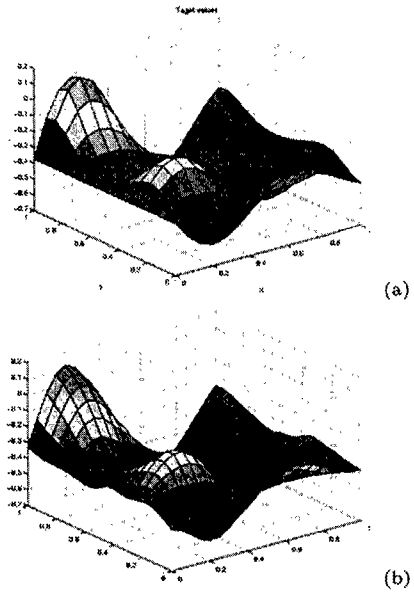


Fig. 6. The training inputs and testing result for function approximation: (a) shows 256(16 × 16) training inputs and (b) shows an output result by using 441(21 × 21) test inputs, which are different from the 256 training inputs.

5 fuzzy rules as shown in *Fig.*5.

Here, two dimensional input space is supposed.

$$\mu_{hq} = (0.25, 0.25), (0.25, 0.75), (0.75, 0.25), (0.75, 0.75), (0.5, 0.5)$$

Γ_{hq}^{-1} is a diagonal matrix with its element being $\frac{1}{0.03}$, σ_{jq} is equal to 1.0, and μ_{jq} is a random value in [0, 1], which will be fixed during the whole training process . Therefore, it is clear from $Eq.(7)-(9)$ that $0 \leq Z_j \leq 1$ holds.

As being stated before, it is a distinguished point of ULNs with BC that ULNs with BC is expected to have higher representation and generalization ability than commonly used neural networks even when $\mu_{jq} \in [0, 1]$ is randomly set.

5.2 Function Approximating Problem In order to study the characteristics of ULNs with BC for a function approximation problem, the simulation is executed by using the following two dimensional function approximation problem which is defined on $0 \leq x \leq 1, 0 \leq y \leq 1$ (see $Fig.6(a)$).

$$f(x,y) = 0.5(0.475((1.35 + e^x sin(13.0(x-0.6)^2) \cdot e^{-y} sin(7y)) - 2.5) - 0.2)$$

The training inputs set contains 256(16 × 16) data patterns, and the testing inputs set are composed of 441(21 × 21) data patterns, which are different from the 256 training inputs.

The basic network is a three layered neural network.

The parameters of the basic network are initialized with random numbers.

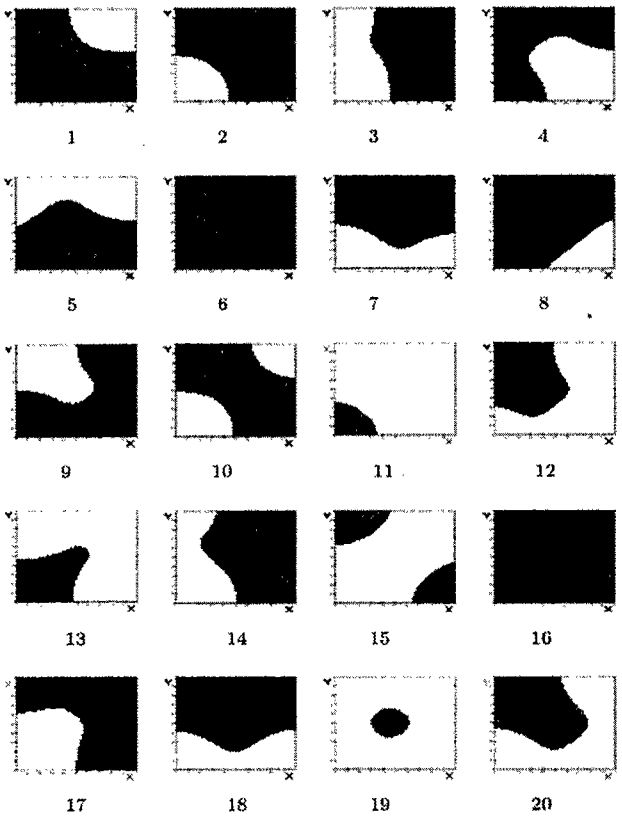


Fig. 7. Connectivity shown by input space with regard to each intermediate node when Z_j^o is set at 0.4 (The number under each small figure represents the number of the intermediate nodes in the basic network)

*Fig.*6(b) shows one of the output results of ULNs with BC after 500 000 epochs training of the basic network with twenty intermediate nodes, when the threshold value Z_j^o is set at 0.2, in this case the connectivity $\rho = 86.31\%$ is obtained by counting the number of connected branches.

Comparing *Fig.*6(a) with *Fig.*6(b), it can be seen that the function is well approximated by the proposed network. The Mean Squared Errors(MSE) of training is 3.48×10^{-4} and the MSE of testing is 3.80×10^{-4}.

*Fig.*7 shows an example of the states of the connection and disconnection of the branches from twenty intermediate nodes to the output node in the basic network respectively, when Z_j^o is set at 0.4, in this case the connectivity $\rho = 59.38\%$ is obtained by counting the

Table 1. Mean Squared Errors (MSE) for Different Thresholds Z_j^o

(The number of hidden nodes: 20)

Z_j^o	ONN	0.0	0.1	0.2
ρ	1.00	1.00	0.95	0.88
MSE(x10^{-4})(Training)	3.96	3.20	3.31	3.80
MSE(x10^{-4})(Testing)	4.32	3.39	3.47	3.99
Z_j^o	0.3	0.4	0.5	0.6
ρ	0.79	0.63	0.51	0.35
MSE(x10^{-4})(Training)	4.16	5.00	5.82	10.24
MSE(x10^{-4})(Testing)	4.36	5.27	6.39	10.83

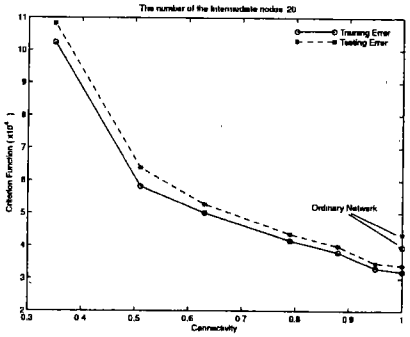


Fig. 8. Mean Squared Errors for different thresholds Z_j^o

(The number of hidden nodes: 20)

Table 2. Mean Squared Errors (MSE) for Different Thresholds Z_j^o

(The number of hidden nodes: 50)

Z_j^o	ONN	0.0	0.1	0.2
ρ	1.00	1.00	0.94	0.89
MSE(x10^{-4})(Training)	3.21	2.27	2.12	2.14
MSE(x10^{-4})(Testing)	3.51	2.43	2.25	2.33
Z_j^o	0.3	0.4	0.5	0.6
ρ	0.77	0.64	0.52	0.35
MSE(x10^{-4})(Training)	2.36	2.27	2.93	3.54
MSE(x10^{-4})(Testing)	2.56	2.43	3.38	3.83

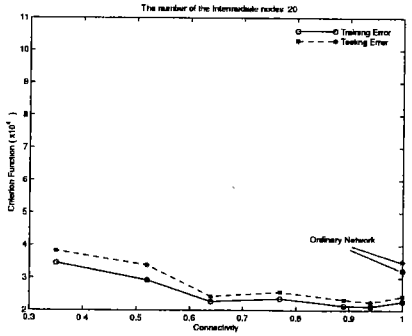


Fig. 9. Mean Squared Error for different thresholds Z_j^o

(The number of hidden nodes: 50)

number of connected branches.

The dotted parts stand for the connection, i.e., $R \neq 0.0$, while the white parts show the disconnection shown by input space (x, y).

It is shown in *Fig.*7 that some areas in the input space (x, y) correspond to the connection of the branches from the intermediate nodes to the output node, while other areas correspond to the disconnection, although the shape of areas is different branch by branch. According to this, one can say that the functions localization is realized by ULNs with BC.

*Tab.*1 and *Fig.*8, *Tab.*2 and *Fig.*9 show how the performance and the generalization ability of the proposed network change as the connecting rate ρ varies by adjusting the threshold Z_j^o (They are the average results over 10 times for each threshold value). Here, the threshold value Z_j^o described as ONN in *Tab.*1 and *Tab.*2 corresponds to an ordinary neural network, that is, ULNs without branch control meaning $R(Z_j) = 1.0$ and the threshold value Z_j^o ranging from 0.0 to 0.6 corresponds to ULN with BC, i.e., the relative strength $R(Z_j)$ is calculated by using *Eq.*(10).

From the results, we can see that ULNs with BC owns better performance and generalization ability than the ordinary neural network.

5.3 Two-Spirals Problem Learning so that two spirals apart should be classified is a neural network benchmark task[10]. Each of the two spirals of the benchmark task makes three complete turns in the plane, with 32 points per turn plus an end point, totaling 97. During one epoch, the outermost circle point is presented first, then the outermost star point, and so on, working into the center of each spiral. Specifically, the training set exemplar sequences $(\mathbf{a}^{(1)}, \mathbf{b}^{(1)})$, $(\mathbf{a}^{(2)}, \mathbf{b}^{(2)})$ $\cdots$, with $\mathbf{a}^{(t)} = (a_1^{(t)}, a_2^{(t)}) \in \mathcal{R}^2$ and $\mathbf{b}^{(t)} = (b^{(t)}) \in \mathcal{R}^1$, are given by the following equations. For $n = 1, 2, \cdots, 97$,

$$a_1^{(2n-1)} = 1 - a_1^{(2n)} = r_n \sin \alpha_n + 0.5, \quad \cdots\cdots \quad (21)$$

$$a_2^{(2n-1)} = 1 - a_2^{(2n)} = r_n \cos \alpha_n + 0.5, \quad \cdots\cdots \quad (22)$$

where

$$r_n = 0.4\left(\frac{105 - n}{104}\right), \quad \cdots\cdots\cdots\cdots\cdots \quad (23)$$

$$\alpha_n = \frac{\pi(n-1)}{16}, \quad \cdots\cdots\cdots\cdots\cdots\cdots \quad (24)$$

$$b^{2n-1} = 1 \quad \text{[circle]}, \quad \cdots\cdots\cdots\cdots\cdots\cdots \quad (25)$$

and

$$b^{2n} = 0 \quad \text{[star]}. \quad \cdots\cdots\cdots\cdots\cdots\cdots \quad (26)$$

The data set consists of 194 points belonging to two interspersed spiral-shaped classes, with 97 samples for each class (*Fig.*10).

The task is to construct a classifier able to distinguish between the two classes. Lang and Witbrock[11] constructed a back-propagation system that learned to distinguish points on the two intertwined spiral. They reported this task could not be accomplished using a standard three layered network, and had to use additional connections to achieve convergence.[10][12]~[16]

Here, the proposed ULNs with BC is used to complete this task. The simulation condition is the same as the function approximating problem.

Also, the basic network is a three layered neural network but with the number of the intermediate nodes

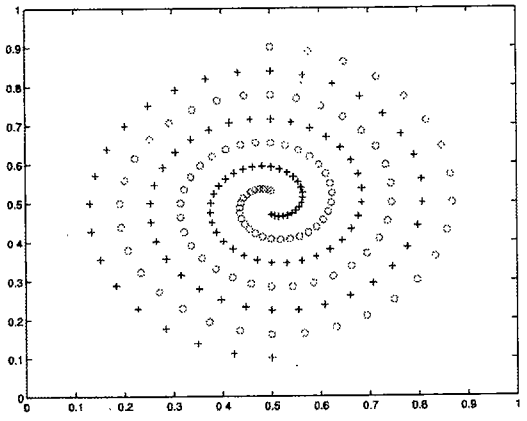


Fig. 10. The training inputs for two-spirals problem

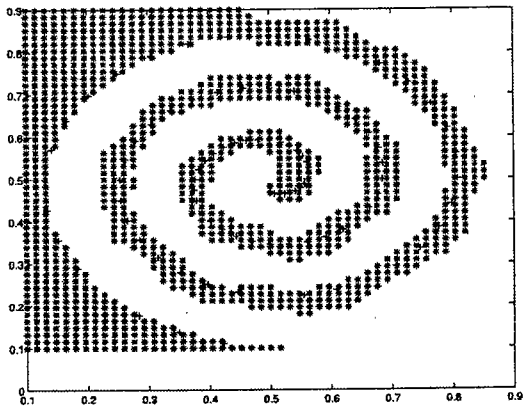


Fig. 11. An output result by using test inputs for the two-spirals problem

being 100. And the testing inputs set consists of 2601(51×51) data patterns, which is different from the 194 training inputs.

*Fig.*11 shows one of the output results of ULNs with BC for the two-spirals problem after 10^6 epochs training, when the threshold value Z_j^o is set at 0.3, in this case the connectivity $\rho = 81.43\%$ is obtained by counting the number of connected branches in the basic network. The Mean Squared Errors(MSE) of training is equal to 0.0173.

From *Fig.*11, we can see that the two intertwined spirals can be clearly separated, while this result is hard to obtain by an ordinary neural network with full connections [11].

*Tab.*3 and *Fig.*12 show how the performance of ULNs with BC changes as the connectivity ρ varies by adjusting the threshold Z_j^o (They are the average results over 6 times for each threshold value).

From our results, one can see that ULNs with BC has better performance and generalization ability than an ordinary neural network, because it can fulfill the classification task for the two-spiral problem even if there is only one intermediate layer in the network. It is clearly stated that ULNs with BC has the tendency to have optimal connectivity, around which the best performance

Table 3. Mean Squared Errors (MSE) for Different Thresholds Z_j^o

(Two-spirals Problem)

Z_j^o	ONN	0.0	0.1	0.2
ρ	1.00	1.00	0.99	0.94
MSE	0.0419	0.0333	0.0295	0.0306
Z_j^o	0.3	0.4	0.5	0.6
ρ	0.84	0.68	0.54	0.35
MSE	0.0174	0.0151	0.0198	0.0218

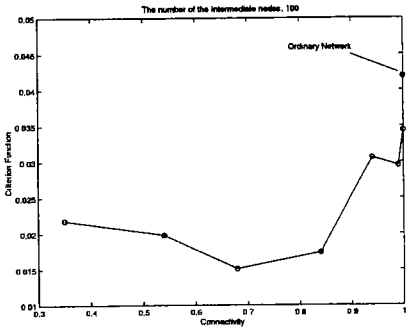


Fig. 12. Mean Squared Error for different thresholds Z_j^o

(Two-spirals problem)

is obtained like *Fig.*4.

6. Conclusions

In this paper, a new type of functions locally processed network named ULNs with BC is proposed. It consists of two kinds of networks such as basic network and branch control network. The branch control network can control the branch connection and disconnection of the basic network with a coefficient of relative strength, depending on the input patterns in the input space. From our simulation results, it has been clarified that ULNs with BC owns better performance and generalization ability than an ordinary neural network and also there exists an optimal connectivity where the best performance of the network can be obtained. Even for the well-known "two-spirals" benchmark problem, the samples can be classified correctly by ULNs with BC only with one intermediate layer in the basic network, although the same results can hardly be obtained by an ordinary neural network as many scientific researchers showed. It means that commonly used neural networks are not always the best in terms of their architecture. This fact is not well known in the neural network community.

(Manuscript received April 20, 2001, revised June 7, 2002)

References

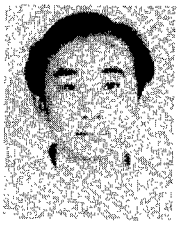
(1) T. Sawaguchi, "*Brain Structure and Evolution*", Kaimeisha, 1989 (in Japanese).

(2) R. A. Jacobs and M. I. Jordan, "Learning piecewise control strategies in a modular neural network architecture", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 23, pp.337-345, 1993.

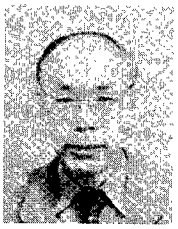
(3) M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of ex-

perts and the EM algorithm", *Neural Computation*, Vol. 6, pp.181-214, 1994.
(4) K. Hirasawa, X. Wang, J. Murata, J. Hu and C. Jin, "Universal Learning Networks and its Application to Chaos Control", *Neural Networks*, Vol.13, pp.239-253, March, 2000.
(5) J. Hu, K. Hirasawa and J. Murata, "RasID - Random Search for Neural Network Training", *Journal of Advanced Computational Intelligence*, Vol. 2, No. 4, pp.134-141, 1998.
(6) K. Hirasawa, M. Ohbayashi, S. Sakai and J. Hu, "Learning Petri Network and Its Application to Nonlinear System Control", *IEEE Transactions on Systems, Man, and Cybernetics Part B*, Vol. 28, No. 6, pp.781-789, 1998.
(7) K. Hirasawa, S. Oka, M. Ohbayashi, S. Sakai and J. Murata, "Locally Functions Distributed Network -Learning Petri Network- based on Petri Network", *Transactions of the Society of Instrument and Control Engineers of Japan*, Vol. 32, No. 6, pp.241-250, 1996 (in Japanese).
(8) M. Ohbayashi, K. Hirasawa, S. Sakai and J. Hu, "An Application of Learning Petri Network to Nonlinear System Control", *Transactions of the Institute of Electrical Engineers of Japan*, Vol. 115-C, No. 6, pp.875-881, 1998 (in Japanese).
(9) K. Hirasawa, M. Ohbayashi, J. Murata, C. Jin, S. Oka, S. Sakai, Y.Shitamura, "Modeling of Brain's Functions Distribution by Petri Network", *Transactions of the Institute of Electrical Engineers of Japan*, Vol. 115-C, No. 5, pp.719-727, 1995 (in Japanese).
(10) G. A. Carpenter, S. Grossberg, N. Markuzon, J. H. Reynold, and D. B. Rosen, "Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps", *IEEE Trans. Neural Networks*, Vol. 3, No. 5, pp. 698-713, Sept. 1992.
(11) K. J. Lang and M. J. Witbrock, "Learning to tell two spiral apart" in *Proc. 1988 Connectionist Models Summer School*, pp. 52-59, 1989.
(12) S. Ridella, S. Rovetta, and R.Zunino, "Circular Backpropagation Networks for Classification", *IEEE Trans. Neural Networks*, Vol.8, No.1, pp.84-97, Jan.1997.
(13) S. Ridella, S. Rovetta, and R. Zunino, "Representation and Generalization Properties of Class-Entropy Networks", *IEEE Trans. Neural Networks*, Vol.10, No.1, pp.31-47, Jan.1999.
(14) N. K. Treadgold and T. D. Gedeon, "Exploring Constructive Cascade networks", *IEEE Trans. Neural Networks*, Vol.10, No.6, pp.1335-1350, Nov.1999.
(15) Q. Xiong, K. Hirasawa, J. Hu, and J. Murata, "Growing RBF Structures Using Self-organizing Maps", *Proc. 2000 IEEE International Workshop on Robot and Human Interactive Communication*, pp. 107-111, 2000
(16) S. E. Fahlman and C. Lebiere, "The Cascade-Correlation Learning Architecture", *Advances in NIPS 2. San Mateo, CA: Morgan Kaufman*, pp. 524-532, 1990.

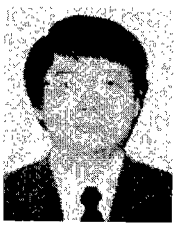
Qingyu XIONG (Non-member) received the Master degree in 1991 from Chongqing University, China. Since April, 1999, he has been a doctor course student in Kyushu University of Japan.



Kotaro HIRASAWA (Member) received the M.S. degree in Electrical Engineering from Kyushu University in 1996. From April 1966, he served in Hitachi Lab. of Hitachi Ltd. and in 1989 he was a vice president of Hitachi Lab.. From August 1991 to November 1992, he served in Omika Factory of Hitachi Ltd.. From December 1992, he was a professor of the Graduate School of information Science and Electrical Engineering, Kyushu University. Since September 2002, he has been a professor of the Graduate School of Information , Production and Systems, Waseda University. Dr. Hirasawa is a member of the Society of Instrument and Control Engineers, a member of IEEE.



Jinglu HU (Member) received the M.S. degree in 1986 from Zhongshan University, China and the PhD degree in 1997 from Kyushu Institute of Technology. From 1986 to 1993, he was a Research Associate and then a Lecturer in Zhongshan University. Since 1997, he has been Research Associate at Kyushu University. His current research interests are learning network theory, system identification and their applications. Dr.Hu is a member of the Society of Instrument and Control Engineering.



Junichi MURATA (Member) received the M.S. and Ph.D. degree from Kyushu University in 1983 and 1986, respectively. He then became a Research Associate and an Associate Professor at Graduate School of Information Science and Electrical Engineering, Kyushu University. His current research interests are neural networks, self-organizing systems and their applications to control and identification. Dr. Murata is a member of SICE, ISCIE, IEEJ and IEEE.

Dazi LI (Non-member) She received the B.S and M.S degree in Industrial Control from Beijing University of Chemical Technology, Beijing, China, in 1992 and 1995 respectively. From 1995 to 2000, she worked in Beijing University of Chemical Technology, where she was a Research Associate and then lecture. Since April 2001, she has been a doctor course student of Kyushu University.