# Learning Algorithms of Neural Network for Interval-Valued Data

# Learning Algorithms of Neural Networks for Interval-Valued Data

Mizuho OMAE*, Ryosuke FUJIOKA**, Hisao ISHIBUCHI**
and Hideo TANAKA**

(Received June 15, 1991)

In this paper, we propose a classification method using a multilayer neural network for two-group discriminant problems where attribute values of each sample are given as intervals. First, by extending a real-valued activation function in a neural network to an interval-valued function, we show an architecture of the neural network which can deal with interval-valued data. Next, we derive a learning algorithm of the neural network from a cost function which is defined by the maximum squared error beween an interval-valued output and a target output. The derived algorithm can be viewed as an extension of the back-propagation algorithm to the case of interval-valued data. The proposed method is illustrated by simulation results for numerical examples. Last, we show a general formulation of the cost function and derive a general learning algorithm of the neural network for interval-valued data. The generalized cost function is defined as the weighted sum of the maximum and minimum squared errors between an interval-valued output and a target output.

## 1. Introduction

In conventional discriminant problems, attribute values of each sample are usually given as real numbers. There are, however, many cases where attribute values can not be represented by real numbers since the values are fluctuating or can not be measured precisely. In these cases, it may be appropriate to represent the attribute values by intervals. For a discriminant method of interval-valued data, Ishibuchi, Tanaka and Fukuoka[1] proposed an LP based method using a linear interval function. This method can apply only to linearly separable interval-valued data.

In this paper, we propose a two-group discriminant method for interval-valued data using a feedforward neural network. First, we show an architecture of the neural network which can deal with interval-valued data as input vectors. This network is obtained by extending the real-valued activation function of each unit to an interval-valued function. The proposed network can be viewed as a nonlinear interval system which maps an interval vector into an interval. Next, we define a

* Graduate Student, Department of Industrial Engineering, College of Engineering, Presently, Matsushita Electric Industrial Co.,LTD.

** Department of Industrial Engineering, College of Engineering

cost function, which should be minimized in learning of the neural network, by the maximum squared error between an interval-valued output and a given target output. We derive a learning algorithm for interval-valued data from this cost function. The derived algorithm can be regarded as an extension of the back-propagation algorithm[2] to the case of interval-valued data. Furthermore, to illustrate the proposed algorithm, we show simulation results for numerical examples. Last, we show a general formulation of the cost function and derive a general learning algorithm of the neural network for interval-valued data. The generalized cost function is defined as the weighted sum of the maximum and minimum squared errors between an interval-valued output and a target output.

## 2. Back-Propagation Algorithm

Before we propose a learning algorithm for interval-valued data, we briefly describe a two-group discriminant method for real-valued data using the back-propagation algorithm.

Let us assume that m patterns that are divided into group 1(G1) and group 2(G2) are given. It is also assumed that n attribute values of each pattern are given. Let n attribute values of the $p$-th pattern be $x_p=(x_{p1},...,x_{pn})$. A two-group discriminant problem can be viewed as finding a discriminant rule that divides an $n$-dimensional pattern space into two areas from the given data $\{x_1,x_2,...,x_m\}$ .

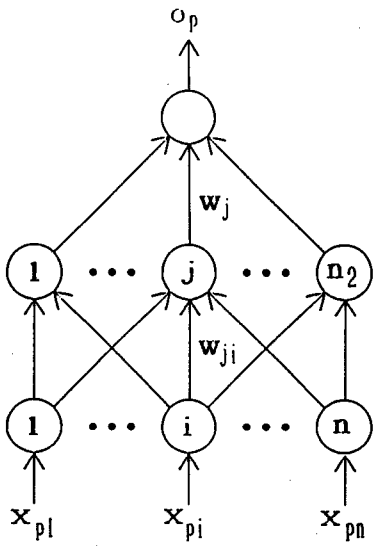


Fig.1 Neural network

To find a nonlinear discriminant rule for a two-group discrimination, we employ a three-layer feedforward neural network with $n$ input units and a single output unit as shown in Fig.1. The number of hidden units, denoted by $n_2$, can be arbitrary.

When the $n$-dimensional input vector $\mathbf{x}_p=(x_{p1},\ldots,x_{pn})$ is presented to the neural network in Fig.1, the input-output relation of each unit is calculated as follows.

Input units : $o_{pi}=x_{pi}, \quad i=1,2,\ldots,n.$ (1)

Hidden units : $o_{pj}=f(net_{pj}), \quad j=1,2,\cdots,n_2,$ (2)

$$net_{pj}=\sum_{i=1}^{n} w_{ji}o_{pi}+\theta_j. \tag{3}$$

Output unit : $o_p=f(net_p),$ (4)

$$net_p=\sum_{j=1}^{n2} w_j o_{pj}+\theta, \tag{5}$$

where $f\ (net)$ is the following activation function.

$$f(net)=1/\ \{1+exp\ (-net)\}\ . \tag{6}$$

The learning of the neural network for the $p$-th input vector $\mathbf{x}_p$ is to minimize the following cost function.

$$e_p=(t_p-o_p)^2/2, \tag{7}$$

where $t_p$ is a target output defined as

$$t_p=\begin{cases}1 & \text{if } p\in G1,\\ 0 & \text{if } p\in G2.\end{cases} \tag{8}$$

Therefore the weights $w_j$ and $w_{ji}$ are changed according to the following rules[2].

$$\Delta_p w_j=\eta(-\partial e_p/\partial w_j)=\eta\delta_p o_{pj}, \tag{9}$$

$$\Delta_p w_{ji}=\eta(-\partial e_p/\partial w_{ji})=\eta\delta_{pj} o_{pi}, \tag{10}$$

where $\eta$ is a learning rate. $\delta_p$ and $\delta_{pj}$ are as follows.

$$\delta_p=(t_p-o_p)o_p(1-o_p)\ , \tag{11}$$

$$\delta_{pj}=o_{pj}(1-o_{pj})\delta_p w_j. \tag{12}$$

Rumelhart et al.[2] introduced the following method with a momentum term in order to accelerate the learning.

$$\Delta w_j(t+1)=\eta\delta_p o_{pj}+\alpha\Delta w_j(t), \tag{13}$$

$$\Delta w_{ji}(t+1)=\eta\delta_{pj} o_{pi}+\alpha\Delta w_{ji}(t), \tag{14}$$

where $t$ is a presentation number and $\alpha$ is a constant corresponding to the momentum term. The biases $\theta$ and $\theta_j$ are changed in the same manner as the weights $w_j$ and $w_{ji}$.

An example of a two-group discrimination using the neural network is shown in Fig.2, where the patterns of group 1(G1) and group 2(G2) are denoted by closed circles and open circles, respectively. The boundary curve is drawn by plotting all the points in the pattern space $[0,20]\times[0,20]$ for which the output values from the neural network are close to 0.5. This means that the output values corresponding to the input vectors on the boundary curve are close to 0.5. Figure 2 is a simulation result of learning of the neural network with five hidden units. The learning is iterated 2,000 times for each pattern with $\eta=0.5$ and $\alpha=0.9$.

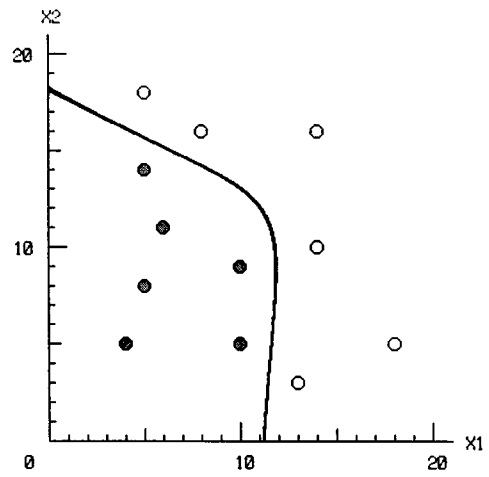


Fig.2 Two-group discriminant analysis by means of a neural network

## 3. Discriminant Analysis of Interval-Valued Data

In Section 2, we briefly mentioned the two-group discriminant method for real-valued data using the feedforward neural network. In this section, we extend the discriminant method to the case of interval-valued data where attribute values of each sample are given as an interval vector. In other words, we generalize the back-propagation algorithm to cope with interval-valued data.

### 3. 1 Interval-valued data

Suppose that $m$ patterns divided into group 1(G1) and group 2(G2) are given. It is assumed that the attribute values of each sample are given as intervals. Let the interval $X_{pi}$ denote the $i$-th attribute value of the $p$-th pattern. Then the attribute values of the $p$-th pattern are denoted by the interval vector $\boldsymbol{X}_p=(X_{p1},...,X_{pn})$. Therefore our discriminant problem is to find a discriminant rule that divides the $n$-dimensional pattern space into two areas from the given interval-valued data $\{\boldsymbol{X}_1,\boldsymbol{X}_2,...,\boldsymbol{X}_m\}$ .

### 3. 2 Real interval arithmetic

In this paper, we denote intervals by uppercase letters $A,B,...,Z$. An interval is represented by its lower (left) and upper (right) limits. For example, $X_{pi}=[x_{pi}{}^L, x_{pi}{}^U]$ .

The generalization of ordinary arithmetic to closed real intervals is known as real interval arithmetic. For complete details, see Alefeld and Herzberger[3]. The basic definition is as follows[3].

**〈definition〉**

Let $*\in\{+,-,\cdot,\div\}$ be binary arithmetic on the set of real numbers. If $A$ and

$B$ are closed real intervals, then

$$A * B = \{a * b \mid a \in A, b \in B\} \tag{15}$$

defines binary arithmetic on the set of all closed real intervals. In the case of division, it is assumed that $0 \notin B$.

The operations on intervals used in this paper can be explicitly calculated from the definition as

$$\begin{aligned} A + B &= [a^L, a^U] + [b^L, b^U] \\ &= [a^L + b^L, a^U + b^U], \end{aligned} \tag{16}$$

$$\begin{aligned} A - B &= [a^L, a^U] - [b^L, b^U] \\ &= [a^L - b^U, a^U - b^L], \end{aligned} \tag{17}$$

$$m \cdot A = m \cdot [a^L, a^U] = \begin{cases} [ma^L, ma^U] & \text{for } m \geqq 0, \\ [ma^U, ma^L] & \text{for } m < 0, \end{cases} \tag{18}$$

where m is a real number.

### 3. 3 Architecture of neural network

For the discriminant analysis of interval-valued data, we use a three-layer feedforward neural network with one output unit as shown in Fig.1. Since attribute values of each pattern are given as intervals, the neural network should be able to deal with the interval vector $\mathbf{X}_p = (X_{p1}, \ldots, X_{pn})$ as an input vector. Therefore we extend the input-output relation of each unit in Section 2 to the interval-valued relation as follows.

Input units : $O_{pi} = X_{pi}, \quad i = 1,2,\ldots,n.$ (19)

Hidden units : $O_{pj} = f(Net_{pj}), \quad j = 1,2,\ldots n_2,$ (20)

$$Net_{pj} = \sum_{i=1}^{n} w_{ji} O_{pi} + \theta_j. \tag{21}$$

Output unit : $O_p = f(Net_p),$ (22)

$$Net_p = \sum_{j=1}^{n_2} w_j O_{pj} + \theta, \tag{23}$$

where the activation function $f(\cdot)$ of the hidden units and the output unit is the logistic function in Eq.(6). Figure 3 shows the interval input-output relation of the hidden units and the output unit. The interval input-output relations in Eqs.(19)–(23) can be explicitly calculated from the interval arithmetic in Section 3.2 as follows.

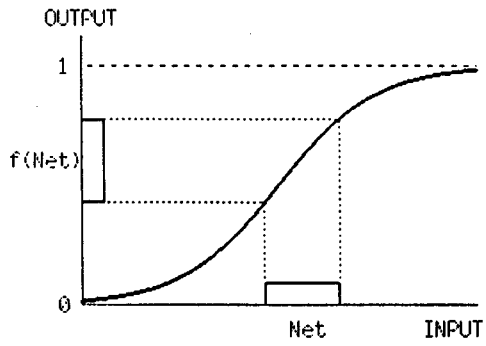


Fig.3 Interval-valued activation function

Input units : $o_{pi}{}^{L}=x_{pi}{}^{L}$, $i=1,2,...,n$, (24)

$o_{pi}{}^{U}=x_{pi}{}^{U}$, $i=1,2,...,n$. (25)

Hidden units : $o_{pj}{}^{L}=f(net_{pj}{}^{L})$, $j=1,2,...,n_2$, (26)

$o_{pj}{}^{U}=f(net_{pj}{}^{U})$, $j=1,2,...,n_2$, (27)

$$net_{pj}{}^{L}=\sum_{\substack{i=1\\ w_{ji}\geqq 0}}^{n} w_{ji}o_{pi}{}^{L}+\sum_{\substack{i=1\\ w_{ji}<0}}^{n} w_{ji}o_{pi}{}^{U}+\theta_j, \tag{28}$$

$$net_{pj}{}^{U}=\sum_{\substack{i=1\\ w_{ji}\geqq 0}}^{n} w_{ji}o_{pi}{}^{U}+\sum_{\substack{i=1\\ w_{ji}<0}}^{n} w_{ji}o_{pi}{}^{L}+\theta_j. \tag{29}$$

Output unit : $o_p{}^{L}=f(net_p{}^{L})$, (30)

$o_p{}^{U}=f(net_p{}^{U})$, (31)

$$net_p{}^{L}=\sum_{\substack{j=1\\ w_j\geqq 0}}^{n2} w_j o_{pj}{}^{L}+\sum_{\substack{j=1\\ w_j<0}}^{n2} w_j o_{pj}{}^{U}+\theta, \tag{32}$$

$$net_p{}^{U}=\sum_{\substack{j=1\\ w_j\geqq 0}}^{n2} w_j o_{pj}{}^{U}+\sum_{\substack{j=1\\ w_j<0}}^{n2} w_j o_{pj}{}^{L}+\theta. \tag{33}$$

### 3. 4 Learning algorithm

Using the interval output $O_p$ corresponding to the input vector $\boldsymbol{X}_p$, we define a cost function $e_p$ for the pattern $p$ as follows.

$$e_p = \max \{(t_p - o_p)^2/2 \mid o_p \in O_p\}$$
$$= \begin{cases} (t_p - o_p^L)^2/2 & \text{if } t_p = 1, \\ (t_p - o_p^U)^2/2 & \text{if } t_p = 0. \end{cases} \quad (34)$$

This cost function is defined as the maximum squared error between the interval-valued output $O_p$ and the target output $t_p$. The learning of the neural network is to minimize the cost function of Eq.(34). The weights $w_j$ and $w_{ji}$ are changed according to the following rules.

$$\Delta_p w_j = \eta(-\partial e_p/\partial w_j), \quad (35)$$

$$\Delta_p w_{ji} = \eta(-\partial e_p/\partial w_{ji}), \quad (36)$$

where $\partial e_p/\partial w_j$ and $\partial e_p/\partial w_{ji}$ are calculated as follows.

(1) $\partial e_p/\partial w_j$

① If $t_p = 1$ and $w_j \geqq 0$, then

$$\frac{\partial e_p}{\partial w_j} = \frac{\partial}{\partial w_j} \{(t_p - o_p^L)^2/2\}$$
$$= \frac{\partial}{\partial o_p^L} \{(t_p - o_p^L)^2/2\} \frac{\partial o_p^L}{\partial net_p^L} \frac{\partial net_p^L}{\partial w_j}$$
$$= -(t_p - o_p^L) o_p^L (1 - o_p^L) o_{pj}^L. \quad (37)$$

② If $t_p = 1$ and $w_j < 0$, then

$$\frac{\partial e_p}{\partial w_j} = -(t_p - o_p^L) o_p^L (1 - o_p^L) o_{pj}^U. \quad (38)$$

③ If $t_p = 0$ and $w_j \geqq 0$, then

$$\frac{\partial e_p}{\partial w_j} = \frac{\partial}{\partial w_j} \{(t_p - o_p^U)^2/2\}$$
$$= \frac{\partial}{\partial o_p^U} \{(t_p - o_p^U)^2/2\} \frac{\partial o_p^U}{\partial net_p^U} \frac{\partial net_p^U}{\partial w_j}$$
$$= -(t_p - o_p^U) o_p^U (1 - o_p^U) o_{pj}^U. \quad (39)$$

④ If $t_p = 0$ and $w_j < 0$, then

$$\frac{\partial e_p}{\partial w_j} = -(t_p - o_p^U) o_p^U (1 - o_p^U) o_{pj}^L. \quad (40)$$

(2) $\partial e_p/\partial w_{ji}$

① If $t_p = 1$, $w_j \geqq 0$ and $w_{ji} \geqq 0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=\frac{\partial}{\partial w_{ji}}\{(t_p-o_p{}^L)^2/2\}$$

$$=\frac{\partial}{\partial o_p{}^L}\{(t_p-o_p{}^L)^2/2\}\frac{\partial o_p{}^L}{\partial net_p{}^L}\frac{\partial net_p{}^L}{\partial o_{pj}{}^L}\frac{\partial o_{pj}{}^L}{\partial net_{pj}{}^L}\frac{\partial net_{pj}{}^L}{\partial w_{ji}}$$

$$=-(t_p-o_p{}^L)o_p{}^L(1-o_p{}^L)w_j o_{pj}{}^L(1-o_{pj}{}^L)o_{pi}{}^L. \tag{41}$$

② If $t_p=1$, $w_j\geqq 0$ and $w_{ji}<0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=-(t_p-o_p{}^L)o_p{}^L(1-o_p{}^L)w_j o_{pj}{}^L(1-o_{pj}{}^L)o_{pi}{}^U. \tag{42}$$

③ If $t_p=1$, $w_j<0$ and $w_{ji}\geqq 0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=\frac{\partial}{\partial w_{ji}}\{(t_p-o_p{}^L)^2/2\}$$

$$=\frac{\partial}{\partial o_p{}^L}\{(t_p-o_p{}^L)^2/2\}\frac{\partial o_p{}^L}{\partial net_p{}^L}\frac{\partial net_p{}^L}{\partial o_{pj}{}^U}\frac{\partial o_{pj}{}^U}{\partial net_{pj}{}^U}\frac{\partial net_{pj}{}^U}{\partial w_{ji}}$$

$$=-(t_p-o_p{}^L)o_p{}^L(1-o_p{}^L)w_j o_{pj}{}^U(1-o_{pj}{}^U)o_{pi}{}^U. \tag{43}$$

④ If $t_p=1$, $w_j<0$ and $w_{ji}<0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=-(t_p-o_p{}^L)o_p{}^L(1-o_p{}^L)w_j o_{pj}{}^U(1-o_{pj}{}^U)o_{pi}{}^L. \tag{44}$$

⑤ If $t_p=0$, $w_j\geqq 0$ and $w_{ji}\geqq 0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=\frac{\partial}{\partial w_{ji}}\{(t_p-o_p{}^U)^2/2\}$$

$$=\frac{\partial}{\partial o_p{}^U}\{(t_p-o_p{}^U)^2/2\}\frac{\partial o_p{}^U}{\partial net_p{}^U}\frac{\partial net_p{}^U}{\partial o_{pj}{}^U}\frac{\partial o_{pj}{}^U}{\partial net_{pj}{}^U}\frac{\partial net_{pj}{}^U}{\partial w_{ji}}$$

$$=-(t_p-o_p{}^U)o_p{}^U(1-o_p{}^U)w_j o_{pj}{}^U(1-o_{pj}{}^U)o_{pi}{}^U. \tag{45}$$

⑥ If $t_p=0$, $w_j\geqq 0$ and $w_{ji}<0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=-(t_p-o_p{}^U)o_p{}^U(1-o_p{}^U)w_j o_{pj}{}^U(1-o_{pj}{}^U)o_{pi}{}^L. \tag{46}$$

⑦ If $t_p=0$, $w_j<0$ and $w_{ji}\geqq 0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=\frac{\partial}{\partial w_{ji}}\{(t_p-o_p{}^U)^2/2\}$$

$$=\frac{\partial}{\partial o_p{}^U}\{(t_p-o_p{}^U)^2/2\}\frac{\partial o_p{}^U}{\partial net_p{}^U}\frac{\partial net_p{}^U}{\partial o_{pj}{}^L}\frac{\partial o_{pj}{}^L}{\partial net_{pj}{}^L}\frac{\partial net_{pj}{}^L}{\partial w_{ji}}$$

$$=-(t_p-o_p{}^U)o_p{}^U(1-o_p{}^U)w_j o_{pj}{}^L(1-o_{pj}{}^L)o_{pi}{}^L. \quad (47)$$

⑧ If $t_p=0$, $w_j<0$ and $w_{ji}<0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=-(t_p-o_p{}^U)o_p{}^U(1-o_p{}^U)w_j o_{pj}{}^L(1-o_{pj}{}^L)o_{pi}{}^U. \quad (48)$$

The following momentum terms can be introduced in the same manner as Rumelhart et al[2].

$$\Delta w_j(t+1)=\eta(-\partial e_p/\partial w_j)+\alpha\Delta w_j(t), \quad (49)$$

$$\Delta w_{ji}(t+1)=\eta(-\partial e_p/\partial w_{ji})+\alpha\Delta w_{ji}(t). \quad (50)$$

The biases $\theta$ and $\theta_j$ are changed in the same manner.

## 4. Numerical Examples

In this section, we illustrate the proposed discriminant method of the interval valued data using numerical examples.

**[Example 1]**

Suppose that attribute values of six patterns are given as shown in Table 1.

Table 1 Interval-valued data in Example 1

| Group 1 (G1) | | | Group 2 (G2) | | |
|---|---|---|---|---|---|
| No. | $X_1$ | $X_2$ | No. | $X_1$ | $X_2$ |
| 1 | [1, 5] | [8, 14] | 4 | [8, 14] | [16, 20] |
| 2 | [4, 10] | [1, 5] | 5 | [13, 19] | [1, 3] |
| 3 | [6, 12] | [9, 11] | 6 | [14, 18] | [5, 15] |

First, we apply the learning algorithm proposed in Section 3.4 to this example using the neural network with five hidden units. The algorithm is iterated 2,000 times for each pattern with $\alpha=0.9$ and $\eta=0.5$. By plotting all the points in the pattern space for which the output values from the neural network are close to 0.5, we can draw the boundary curve as Fig.4. We also show the interval-valued outputs corresponding to the given interval input vectors together with the target outputs in Table 2. We can see from Fig.4 and Table 2 that the neural network trained by the

proposed algorithm can discriminate all the given interval-valued data correctly.

Next, in order to compare the original back-propagation algorithm with the proposed algorithm, we apply the back-propagation algorithm to this example. Since the original back-propagation algorithm can not deal with interval-valued data, we use the four vertexes of each of the given six interval input vectors. The neural network with five hidden units is trained 2,000 times using the back-propagation algorithm for each vertex with the same parameterization as the proposed algorithm.

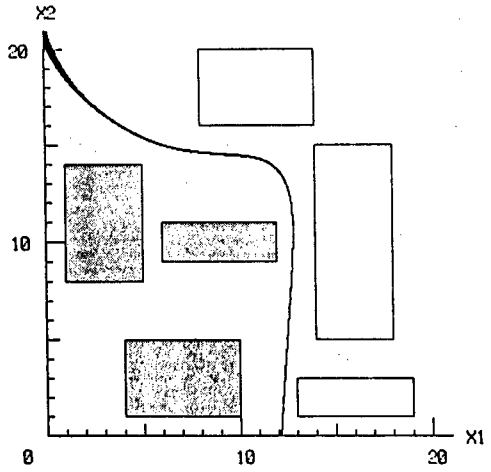


Fig.4 Simulation result after 2,000 iterations of the proposed algorithm for the interval-valued data in Example 1

Table 2 Interval-valued outputs corresponding to the input vectors given in Table 1

| Group 1 (G1) Target output=1 | | Group 2 (G2) Target output=0 | |
|---|---|---|---|
| No. | Interval-valued output | No. | Interval-valued output |
| 1 | [0.99, 1.00] | 4 | [0.00, 0.02] |
| 2 | [1.00, 1.00] | 5 | [0.00, 0.02] |
| 3 | [0.96, 1.00] | 6 | [0.00, 0.02] |

The boundary curve corresponding to the output values close to 0.5 is shown in Fig.5. We can see from Fig.5 that the neural network trained by the back-propagation algorithm discriminates all vertexes of interval vectors correctly, but does not discriminate all the given interval vectors. Of course, if we use many other points included in the interval vectors for the learning of the neural network, all the given interval-valued data may be discriminated correctly. But in such a method, many points are needed for each interval vector. Actually, if we use the vertexes and the midpoints of intervals as shown in Fig.6, the neural network should learn $3^n$ points for an n-dimensional interval vector. Therefore the learning using many points in each of the given interval vectors may require enormous learning time especially in

the case where the dimension of the pattern space, i.e., the number of attributes is large.

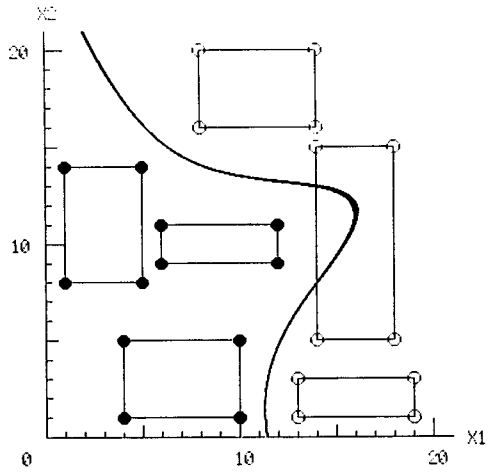


Fig.5 Simulation result after 2,000 iterations of the back-propagation algorithm for all vertexes of interval-valued data in Example 1

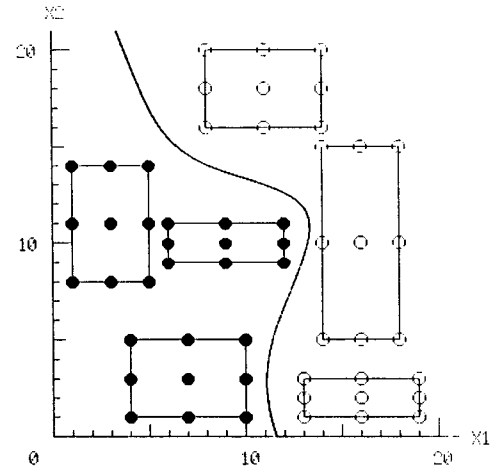


Fig.6 Simulation result after 2,000 iterations of the back-propagation algorithm for all vertexes and midpoints of inteval-valued data in Example 1

**[Example 2]**

Since the proposed algorithm is the generalization of the back-propagation algorithm to the case of interval-valued data, it can deal with interval-valued data where some attribute values are given as real numbers. An example of such data is shown in Table 3.

We apply the proposed algorithm to the example in Table 3 using the neural network with five hidden units. In this case, we regard a real number $x$ as a degenerated interval $[x, x]$. The algorithm is iterated 2,000 times for each pattern with $\alpha=0.9$ and $\eta=0.5$. The boundary curve corresponding to the output values

close to 0.5 is shown in Fig.7. We can see from Fig.7 that the neural network can discriminate both interval-valued data and real-valued data correctly.

Table 3 Interval-valued data and real-valued data in Example 2

| Group1 (G1) | | | Group2 (G2) | | |
|---|---|---|---|---|---|
| No. | $X_1$ | $X_2$ | No. | $X_1$ | $X_2$ |
| 1 | [2, 10] | [2, 10] | 4 | 3 | 15 |
| 2 | 13 | 5 | 5 | 11 | 16 |
| 3 | 13 | 11 | 6 | [16, 20] | [2, 20] |

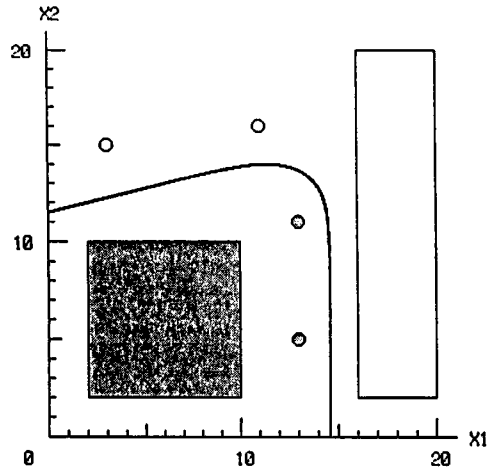


Fig.7 Simulation result after 2,000 iterations of the proposed algorithm for the inteval-valued data and the real-valued data in Example 2

## 5. General Formulation of Cost Function

In Section 3, we define the cost function as the maximum squared error between the target output and the interval-valued output from the neural network. In this section, we define a general cost function and derive a learning algorithm. The generalized cost function is defined as the weighted sum of the maximum and minimum squared errors between the target output and the interval-valued output from the neural network. We define the following cost function.

$$e_p=\beta \cdot \max \{(t_p-o_p)^2/2 \mid o_p \in O_p\}$$

$$+(1-\beta) \cdot \min \{(t_p-o_p)^2/2 \mid o_p \in O_p\}$$

$$=\begin{cases} \beta(t_p-o_p^L)^2/2+(1-\beta)(t_p-o_p^U)^2/2 & \text{if } t_p=1, \\ \beta(t_p-o_p^U)^2/2+(1-\beta)(t_p-o_p^L)^2/2 & \text{if } t_p=0, \end{cases} \tag{51}$$

where $\beta$ is the constant within the closed interval $[0,1]$ . This cost function is coincident with the cost function in Section 3 in the case of $\beta=1.00$. In the case of $\beta=0.00$, Eq.(51) is the minimum squared error between the target output $t_p$ and the interval-valued output $O_p$ from the neural network. We show the learning algorithm based on the cost function defined by Eq.(51) in Appendix.

We apply the learning algorithm based on Eq.(51) to the data in Example 1 using various values of $\beta$ in order to investigate the effect of $\beta$ on the learning speed of the neural network. Using the neural network with five hidden units, the learning algorithm with $\alpha=0.9$ and $\eta=0.5$ is iterated until the sum of cost functions:

$$e_{SUM}=\sum_{p=1}^{m} e_p \tag{52}$$

is less than 0.01.

We show the number of iterations for $e_{SUM}<0.01$ in Table 4. From Table 4, we can see that the value of $\beta$ has large effect upon the learning speed of the neural network. Since the cost function in the case of $\beta=1.00$ is the same as in Section 3, we obtain a similar result as Fig.4 for $\beta=1.00$. For the cases of $\beta=0.75$, $\beta=0.50$, $\beta=0.25$ and $\beta=0.00$, we show the results in Figs.8, 9, 10 and 11, respectively. These results show that the neural network after learning can discriminate the given interval-valued data correctly in the case of $\beta>0.00$. In the case of $\beta=0.00$, although the sum of the cost functions is very small (i.e., 0.01), the neural network can not discriminate the given interval-valued data correctly (see Fig.11). From these results, we can see that it is proper to set $\beta=1.00$ for the data in Table 1. We obtain a similar result for the data in Table 3, that is, it is also proper to set $\beta=1.00$.

Table 4. Number of iterations for $e_{SUM}<0.01$

| $\beta$ | 1.00 | 0.75 | 0.50 | 0.25 | 0.00 |
|---|---|---|---|---|---|
| Iterations | 2,049 | 4,546 | 82,890 | 5,145 | 41 |

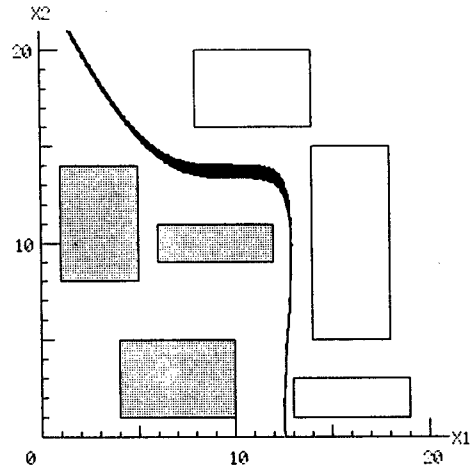


Fig.8 Simulation result with $\beta=0.75$

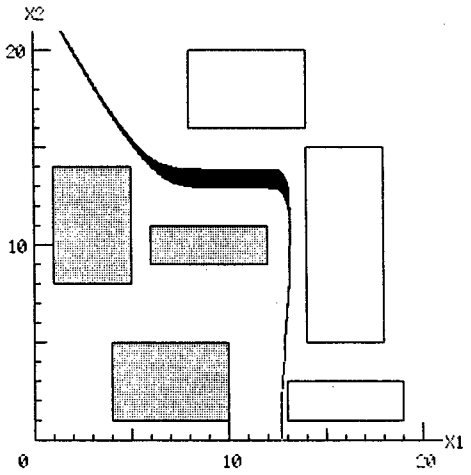


Fig.9 Simulation result with $\beta=0.50$

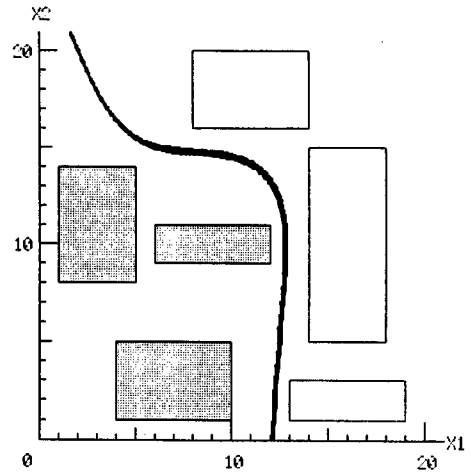


Fig.10 Simulation result with $\beta=0.25$

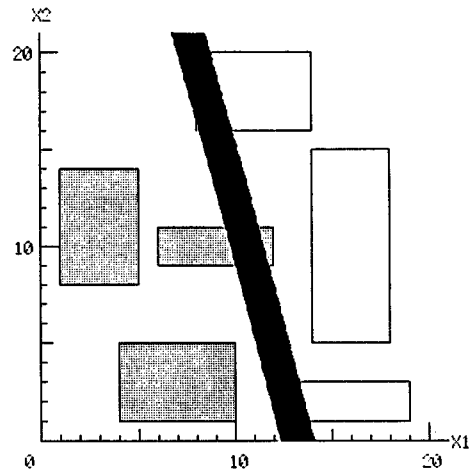


Fig.11 Simulation result with $\beta=0.00$

Next, we apply the same algorithm to the data in Table 5. In Table 5, it should be noted that the given data have an overlap between two groups. Accordingly, the neural network can not discriminate all the given data correctly.

Table 5 Overlapping data

| Group 1 (G1) | | | Group 2 (G2) | | |
|---|---|---|---|---|---|
| No. | $X_1$ | $X_2$ | No. | $X_1$ | $X_2$ |
| 1 | [4, 8] | [12, 20] | 5 | [8, 12] | [6, 10] |
| 2 | [2, 4] | [2, 8] | 6 | [16, 20] | [8, 10] |
| 3 | [6, 10] | [4, 8] | 7 | [18, 20] | [12, 20] |
| 4 | [12, 18] | [2, 4] | 8 | [12, 16] | [16, 20] |

We show the results after learning using various values of $\beta$ in Figs. 12,13,14,15 and 16. All of these figures are obtained after 20,000 iterations of the learning algorithm. From Figs. 12,13,14,15 and 16, we can see that the discrimination result with $\beta=0.5$ is more proper than that with $\beta=1.00$.

From these simulation results mentioned above, we can see that $\beta=1.00$ may be proper for the data with no overlap and may not be proper for those with overlaps.

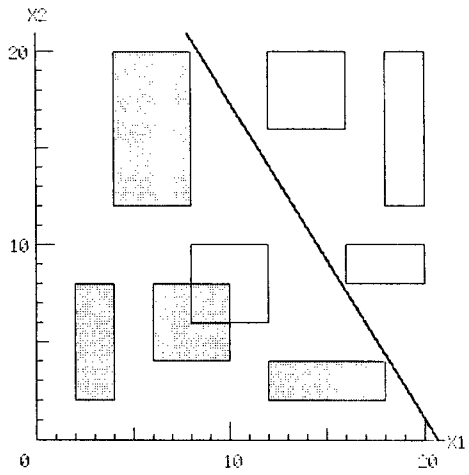


Fig.12 Simulation result for overlapping data with $\beta=1.00$

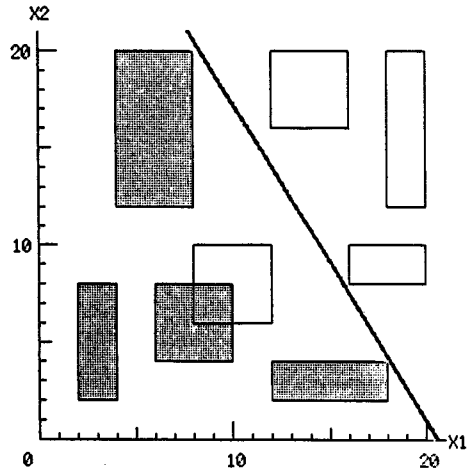


Fig.13 Simulation result for overlapping data with $\beta=0.75$

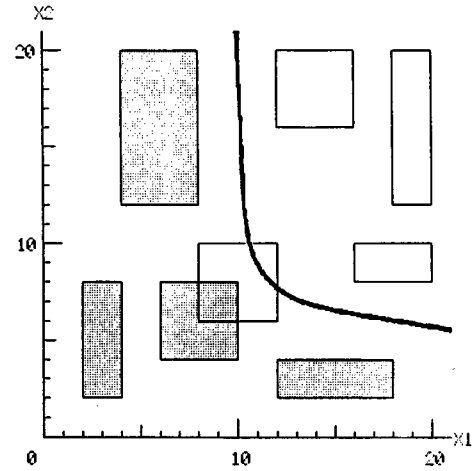


Fig.14 Simulation result for overlapping data with $\beta=0.50$

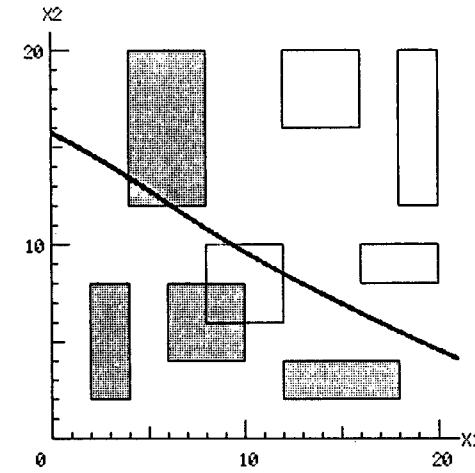


Fig.15 Simulation result for overlapping data with $\beta=0.25$

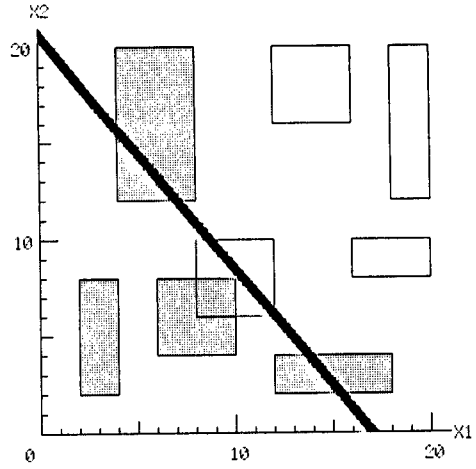


Fig.16 Simulation result for overlapping data with $\beta=0.00$

## 6. Conclusion

In this paper, we proposed a classification method using a multilayer neural network for two-group discriminant problems where attribute values of each sample are given as intervals. The derived algorithm can be viewed as an extension of the back-propagation algorithm to the case of interval-valued data. Furthermore, we illustrated the proposed method and demonstrated its classification power using simulation results for numerical examples. Last, we showed a general formulation of the cost function defined by the weighted sum of the maximum and minimum squared errors between the interval-valued output and the target output.The learning algorithm based on the generalized cost function was also derived.

## Appendix: Generalized learning algorithm

The learning of neural network is to minimize the cost function of Eq.(51). The weights $w_j$ and $w_{ji}$ are changed according to Eqs.(49) and (50). $\partial e_p/\partial w_j$ and $\partial e_p/\partial w_{ji}$ are calculated as follows.

(1) $\partial e_p/\partial w_j$

① If $t_p=1$ and $w_j \geqq 0$, then

$$\frac{\partial e_p}{\partial w_j}=\beta\frac{\partial}{\partial w_j}\{(t_p-o_p{}^L)^2/2\}+(1-\beta)\frac{\partial}{\partial w_j}\{(t_p-o_p{}^U)^2/2\}$$

$$=\beta\frac{\partial}{\partial o_p{}^L}\{(t_p-o_p{}^L)^2/2\}\frac{\partial o_p{}^L}{\partial net_p{}^L}\frac{\partial net_p{}^L}{\partial w_j}$$

$$+(1-\beta)\frac{\partial}{\partial o_p{}^U}\{(t_p-o_p{}^U)^2/2\}\frac{\partial o_p{}^U}{\partial net_p{}^U}\frac{\partial net_p{}^U}{\partial w_j}$$

$$=-\beta(t_p-o_p{}^L)o_p{}^L(1-o_p{}^L)o_{pj}{}^L$$

$$-(1-\beta)(t_p-o_p{}^U)o_p{}^U(1-o_p{}^U)o_{pj}{}^U. \qquad (A-1)$$

② If $t_p=1$ and $w_j<0$, then

$$\frac{\partial e_p}{\partial w_j}=-\beta(t_p-o_p{}^L)o_p{}^L(1-o_p{}^L)o_{pj}{}^U$$

$$-(1-\beta)(t_p-o_p{}^U)o_p{}^U(1-o_p{}^U)o_{pj}{}^L. \qquad (A-2)$$

③ If $t_p=0$ and $w_j \geqq 0$, then

$$\frac{\partial e_p}{\partial w_j}=\beta\frac{\partial}{\partial w_j}\{(t_p-o_p{}^U)^2/2\}+(1-\beta)\frac{\partial}{\partial w_j}\{(t_p-o_p{}^L)^2/2\}$$

$$=\beta\frac{\partial}{\partial o_p{}^U}\{(t_p-o_p{}^U)^2/2\}\frac{\partial o_p{}^U}{\partial net_p{}^U}\frac{\partial net_p{}^U}{\partial w_j}$$

$$+(1-\beta)\frac{\partial}{\partial o_p{}^L}\{(t_p-o_p{}^L)^2/2\}\frac{\partial o_p{}^L}{\partial net_p{}^L}\frac{\partial net_p{}^L}{\partial w_j}$$

$$=-\beta(t_p-o_p{}^U)o_p{}^U(1-o_p{}^U)o_{pj}{}^U$$

$$-(1-\beta)(t_p-o_p{}^L)o_p{}^L(1-o_p{}^L)o_{pj}{}^L. \qquad (A-3)$$

④ If $t_p=0$ and $w_j<0$, then

$$\frac{\partial e_p}{\partial w_j}=-\beta(t_p-o_p{}^U)o_p{}^U(1-o_p{}^U)o_{pj}{}^L$$

$$-(1-\beta)(t_p-o_p{}^L)o_p{}^L(1-o_p{}^L)o_{pj}{}^U. \qquad (A-4)$$

(2) $\partial e_p/\partial w_{ji}$

① If $t_p=1$, $w_j \geqq 0$ and $w_{ji} \geqq 0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=\beta\frac{\partial}{\partial w_{ji}}\{(t_p-o_p^L)^2/2\}+(1-\beta)\frac{\partial}{\partial w_{ji}}\{(t_p-o_p^U)^2/2\}$$

$$=\beta\frac{\partial}{\partial o_p^L}\{(t_p-o_p^L)^2/2\}\frac{\partial o_p^L}{\partial net_p^L}\frac{\partial net_p^L}{\partial o_{pj}^L}\frac{\partial o_{pj}^L}{\partial net_{pj}^L}\frac{\partial net_{pj}^L}{\partial w_{ji}}$$

$$+(1-\beta)\frac{\partial}{\partial o_p^U}\{(t_p-o_p^U)^2/2\}\frac{\partial o_p^U}{\partial net_p^U}\frac{\partial net_p^U}{\partial o_{pj}^U}\frac{\partial o_{pj}^U}{\partial net_{pj}^U}\frac{\partial net_{pj}^U}{\partial w_{ji}}$$

$$=-\beta(t_p-o_p^L)o_p^L(1-o_p^L)w_j o_{pj}^L(1-o_{pj}^L)o_{pi}^L$$

$$-(1-\beta)(t_p-o_p^U)o_p^U(1-o_p^U)w_j o_{pj}^U(1-o_{pj}^U)o_{pi}^U. \quad (A-5)$$

② If $t_p=1$, $w_j \geqq 0$ and $w_{ji}<0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=-\beta(t_p-o_p^L)o_p^L(1-o_p^L)w_j o_{pj}^L(1-o_{pj}^L)o_{pi}^U$$

$$-(1-\beta)(t_p-o_p^U)o_p^U(1-o_p^U)w_j o_{pj}^U(1-o_{pj}^U)o_{pi}^L. \quad (A-6)$$

③ If $t_p=1$, $w_j<0$ and $w_{ji} \geqq 0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=\beta\frac{\partial}{\partial w_{ji}}\{(t_p-o_p^L)^2/2\}+(1-\beta)\frac{\partial}{\partial w_{ji}}\{(t_p-o_p^U)^2/2\}$$

$$=\beta\frac{\partial}{\partial o_p^L}\{(t_p-o_p^L)^2/2\}\frac{\partial o_p^L}{\partial net_p^L}\frac{\partial net_p^L}{\partial o_{pj}^U}\frac{\partial o_{pj}^U}{\partial net_{pj}^U}\frac{\partial net_{pj}^U}{\partial w_{ji}}$$

$$+(1-\beta)\frac{\partial}{\partial o_p^U}\{(t_p-o_p^U)^2/2\}\frac{\partial o_p^U}{\partial net_p^U}\frac{\partial net_p^U}{\partial o_{pj}^L}\frac{\partial o_{pj}^L}{\partial net_{pj}^L}\frac{\partial net_{pj}^L}{\partial w_{ji}}$$

$$=-\beta(t_p-o_p^L)o_p^L(1-o_p^L)w_j o_{pj}^U(1-o_{pj}^U)o_{pi}^U$$

$$-(1-\beta)(t_p-o_p^U)o_p^U(1-o_p^U)w_j o_{pj}^L(1-o_{pj}^L)o_{pi}^L. \quad (A-7)$$

④ If $t_p=1$, $w_j<0$ and $w_{ji}<0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=-\beta(t_p-o_p^L)o_p^L(1-o_p^L)w_j o_{pj}^U(1-o_{pj}^U)o_{pi}^L$$

$$-(1-\beta)(t_p-o_p^U)o_p^U(1-o_p^U)w_j o_{pj}^L(1-o_{pj}^L)o_{pi}^U. \quad (A-8)$$

⑤ If $t_p=0$, $w_j \geqq 0$ and $w_{ji} \geqq 0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=\beta\frac{\partial}{\partial w_{ji}}\{(t_p-o_p{}^U)^2/2\}+(1-\beta)\frac{\partial}{\partial w_{ji}}\{(t_p-o_p{}^L)^2/2\}$$

$$=\beta\frac{\partial}{\partial o_p{}^U}\{(t_p-o_p{}^U)^2/2\}\frac{\partial o_p{}^U}{\partial net_p{}^U}\frac{\partial net_p{}^U}{\partial o_{pj}{}^U}\frac{\partial o_{pj}{}^U}{\partial net_{pj}{}^U}\frac{\partial net_{pj}{}^U}{\partial w_{ji}}$$

$$+(1-\beta)\frac{\partial}{\partial o_p{}^L}\{(t_p-o_p{}^L)^2/2\}\frac{\partial o_p{}^L}{\partial net_p{}^L}\frac{\partial net_p{}^L}{\partial o_{pj}{}^L}\frac{\partial o_{pj}{}^L}{\partial net_{pj}{}^L}\frac{\partial net_{pj}{}^L}{\partial w_{ji}}$$

$$=-\beta(t_p-o_p{}^U)o_p{}^U(1-o_p{}^U)w_j o_{pj}{}^U(1-o_{pj}{}^U)o_{pi}{}^U$$

$$-(1-\beta)(t_p-o_p{}^L)o_p{}^L(1-o_p{}^L)w_j o_{pj}{}^L(1-o_{pj}{}^L)o_{pi}{}^L. \qquad (A-9)$$

⑥ If $t_p=0$, $w_j \geqq 0$ and $w_{ji}<0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=-\beta(t_p-o_p{}^U)o_p{}^U(1-o_p{}^U)w_j o_{pj}{}^U(1-o_{pj}{}^U)o_{pi}{}^L$$

$$-(1-\beta)(t_p-o_p{}^L)o_p{}^L(1-o_p{}^L)w_j o_{pj}{}^L(1-o_{pj}{}^L)o_{pi}{}^U. \qquad (A-10)$$

⑦ If $t_p=0$, $w_j<0$ and $w_{ji} \geqq 0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=\beta\frac{\partial}{\partial w_{ji}}\{(t_p-o_p{}^U)^2/2\}+(1-\beta)\frac{\partial}{\partial w_{ji}}\{(t_p-o_p{}^L)^2/2\}$$

$$=\beta\frac{\partial}{\partial o_p{}^U}\{(t_p-o_p{}^U)^2/2\}\frac{\partial o_p{}^U}{\partial net_p{}^U}\frac{\partial net_p{}^U}{\partial o_{pj}{}^L}\frac{\partial o_{pj}{}^L}{\partial net_{pj}{}^L}\frac{\partial net_{pj}{}^L}{\partial w_{ji}}$$

$$+(1-\beta)\frac{\partial}{\partial o_p{}^L}\{(t_p-o_p{}^L)^2/2\}\frac{\partial o_p{}^L}{\partial net_p{}^L}\frac{\partial net_p{}^L}{\partial o_{pj}{}^U}\frac{\partial o_{pj}{}^U}{\partial net_{pj}{}^U}\frac{\partial net_{pj}{}^U}{\partial w_{ji}}$$

$$=-\beta(t_p-o_p{}^U)o_p{}^U(1-o_p{}^U)w_j o_{pj}{}^L(1-o_{pj}{}^L)o_{pi}{}^L$$

$$-(1-\beta)(t_p-o_p{}^L)o_p{}^L(1-o_p{}^L)w_j o_{pj}{}^U(1-o_{pj}{}^U)o_{pi}{}^U. \qquad (A-11)$$

⑧ If $t_p=0$, $w_j<0$ and $w_{ji}<0$, then

$$\frac{\partial e_p}{\partial w_{ji}}=-\beta(t_p-o_p{}^U)o_p{}^U(1-o_p{}^U)w_j o_{pj}{}^L(1-o_{pj}{}^L)o_{pi}{}^U$$

$$-(1-\beta)(t_p-o_p{}^L)o_p{}^L(1-o_p{}^L)w_j o_{pj}{}^U(1-o_{pj}{}^U)o_{pi}{}^L. \qquad (A-12)$$

## References

1) H. Ishibuchi, H. Tanaka and N. Fukuoka, International Journal of General Systems, **16,** (4) 311, (1990)
2) D.E.Rumelhart, J.L.McClelland and the PDP Research Group, "Parallel Distributed Processing Vol.1", MIT Press, Cambridge, USA, (1989)
3) G.Alefeld and J.Herzberger, "Introduction to Interval Computations", Academic Press, New York, USA, (1983)