Competing Behavior of Two Kinds of SOMs and its Application to Clustering

Haruna MATSUSHITA and Yoshifumi NISHIO
Tokushima University, JAPAN
{haruna, nishio}@ee.tokushima-u.ac.jp

Abstract - *The Self-Organizing Map (SOM) is an unsupervised neural network introduced in the 80's by Teuvo Kohonen. In this paper, we propose a method of using simultaneously two kinds of SOMs whose features are different. Namely, one is distributed in the area on which input data are concentrated, and the other self-organizes the whole of the input space. The competing behavior of the two kinds of SOMs for nonuniform input data is investigated. Furthermore, we show its application to clustering and confirm the efficiency by comparing with the k-means method.*

Key words - Self-Organizing Maps, clustering, data mining

1 Introduction

In data mining, clustering is essential and important. The k-means method is known as the representative method because of its simplicity [1]. However, when the data contains many noises, it is difficult to extract only the cluster exactly using the k-means method. Then, the Self-Organizing Map (SOM) attracts attentions in recent years. SOM is an unsupervised neural network introduced by Kohonen in 1982 [2] and is a model simplifying self-organization process of the brain. SOM obtains a statistical feature of input data and is applied to a wide field of data classifications. Although many methods to extract clusters by using SOM have been proposed [3][4][5], it seems to be very difficult to construct a simple method using SOM for universal input data. Further, since we can accumulate a huge amount of data including useless information in these years, it is important to investigate various extraction methods of clusters from data including a lot of noises.

In our past study, we have investigated the basic features of using two kinds of SOMs whose features are different [6]. We have confirmed that the two SOMs could extract the features of 2-dimensional nonuniform input data. However, in the study the difference between two SOMs was not completely clear and the clustering ability was not evaluated. In this study, we propose a method using simultaneously two kinds of SOMs whose features are different. Namely, one self-organizes the area on which input data are concentrated, and the other self-organizes the whole of the input space. We explain the difference of the two kinds of SOMs with the learning algorithm and investigate the competing behavior of the two kinds of SOMs. Next, we apply the two kinds of SOMs to clustering. For 2 and 3-dimensional input data including a lot of noises (corresponding to useless information), clustering ability is evaluated both visually and quantitatively using a correct answer rate. Further, the results are compared with those obtained by the k-means method. We also apply the two kinds of SOMs to 5-dimensional input data including a lot of noises and confirm the clustering ability for higher-dimensional data.

2 Two Kind of SOMs

In this study, we propose a method using simultaneously two kinds of SOMs, namely, one self-organizes the area on which input data are concentrated, and the other self-organizes the whole of the input space. We call the former $\mathrm{SOM_L}$ and the latter $\mathrm{SOM_G}$.

2.1 Learning algorithm

We explain the learning algorithm of the two kinds of SOMs. In order to apply the two kinds of SOMs to clustering applications, we use totally n SOMs, that is one $\mathrm{SOM_G}$ and $(n-1)$ $\mathrm{SOM_L}$; namely $\mathrm{SOM_{L1}}$, $\mathrm{SOM_{L2}}$, $\cdots$, $\mathrm{SOM_{L(n-1)}}$. In each SOM, m neurons are arranged as a regular 2-dimensional grid. The range of the elements of d-dimensional input data $\boldsymbol{x}_j = (x_{j1}, x_{j2}, \cdots, x_{jd})$ $(j = 1, 2, \cdots, N)$ are assumed to be from 0 to 1.

[PHASE 1]
(n**SOM1**) The initial values of all the weight vectors $\boldsymbol{w}_{Ll}$ of $\mathrm{SOM}_{\mathrm{L}l}$ $(l = 1, 2, \cdots, n-1)$ are given between 0 and 1 at random. The initial values of all the weight vectors $\boldsymbol{w}_G$ of $\mathrm{SOM_G}$ are given around the center of the input space at random, for example uniform between 0.45 and 0.55.

[PHASE 2]
(n**SOM2**) An input data $\boldsymbol{x}_j$ is inputted to all the neurons of $\mathrm{SOM_G}$ and $\mathrm{SOM}_{\mathrm{L}l}$ at the same time in parallel.
(n**SOM3**) The distance between $\boldsymbol{x}_j$ and the weight vector $\boldsymbol{w}_{Gi} = (w_{Gi1}, w_{Gi2}, \cdots, w_{Gid})$ $(i = 1, 2, \cdots, m)$ of the neuron i of $\mathrm{SOM_G}$, and the distance between $\boldsymbol{x}_j$ and the weight vector $\boldsymbol{w}_{Lli} = (w_{Lli1}, w_{Lli2}, \cdots, w_{Llid})$ of the neuron i of $\mathrm{SOM}_{\mathrm{L}l}$ are calculated. The internal activity degrees $net_{G_i^j}$ and $net_{Ll_i^j}$ are obtained as;

$$net_{G_i^j} = \|\boldsymbol{w}_{Gi} - \boldsymbol{x}_j\|^{-1}, \qquad net_{Ll_i^j} = \|\boldsymbol{w}_{Lli} - \boldsymbol{x}_j\|^{-1}. \tag{1}$$

(n**SOM4**) The winner neuron $c(j)$ for $\boldsymbol{x}_j$ is the neuron with the maximum internal activity degree in all net_G^j and $net_{L_l^j}$.
(n**SOM5**) If the winner neuron $c(j)$ is a neuron in $\mathrm{SOM_G}$, the weight vectors of the all neurons of $\mathrm{SOM_G}$ are updated as;

$$\boldsymbol{w}_{Gi}(t+1) = \boldsymbol{w}_{Gi}(t) + h_{Gc(j),i}(t)(\boldsymbol{x}_j - \boldsymbol{w}_{Gi}(t)), \tag{2}$$

where $h_{Gc(j),i}(t)$ is the neighborhood function of $\mathrm{SOM_G}$.
While, if the winner neuron $c(j)$ is in $\mathrm{SOM}_{\mathrm{L}l}$, the weight vectors of the neurons of $\mathrm{SOM}_{\mathrm{L}l}$ are updated as;

$$\boldsymbol{w}_{Lli}(t+1) = \boldsymbol{w}_{Lli}(t) + h_{Lc(j),i}(t)(\boldsymbol{x}_j - \boldsymbol{w}_{Lli}(t)), \tag{3}$$

where $h_{Lc(j),i}(t)$ is the neighborhood function of $\mathrm{SOM_L}$. The neighborhood functions $h_{Gc(j),i}(t)$ and $h_{Lc(j),i}(t)$ are important functions deciding the behaviors of n SOMs and are explained in the next subsection.

(nSOM6) The steps from (nSOM2) to (nSOM5) are repeated for all the input data, namely from $j = 1$ to $j = N$.

[PHASE 3]
(nSOM7) Furthermore, only SOM$_\mathrm{G}$ learns with the time reset as $t = 0$.
(nSOM8) An input data is inputted to all the neurons similarly to the step (nSOM2).
(nSOM9) The internal activity degree $net_{Ll_i}^{j}$ is calculated similarly to the step (nSOM3).
(nSOM10) If the maximum value of $net_{Ll_i}^{j}$ is smaller than a given threshold value $1/\varepsilon$ (this means that the distance between the input data and SOM$_{\mathrm{L}l}$ is larger than ε), the weight vectors of the neurons of SOM$_\mathrm{G}$ are updated as;

$$\boldsymbol{w}_{Gi}(t+1) = \boldsymbol{w}_{Gi}(t) + h_{Gsc(j),i}(t)(\boldsymbol{x}_j - \boldsymbol{w}_{Gi}(t)). \tag{4}$$

(nSOM11) The steps from (nSOM8) to (nSOM10) are repeated for all the input data.

2.2 Neighborhood function

In the learning algorithm of n SOMs, the difference between the neighborhood functions of SOM$_\mathrm{G}$ and SOM$_\mathrm{L}$ plays a key role to decide their behaviors.
The neighborhood functions for SOM$_\mathrm{G}$ and SOM$_\mathrm{L}$ are described as follows;

$$h_{Gc(j),i}(t) = \alpha_G(t) \exp\left(-\frac{\|\boldsymbol{r}_i - \boldsymbol{r}_{c(j)}\|^2}{2\sigma_G^2(t)}\right), \quad h_{Lc(j),i}(t) = \alpha_L(t) \exp\left(-\frac{\|\boldsymbol{r}_i - \boldsymbol{r}_{c(j)}\|^2}{2\sigma_L^2(t)}\right), \tag{5}$$

where $\alpha_G(t)$ and $\alpha_L(t)$ are the learning rate, $\boldsymbol{r}_i$ and $\boldsymbol{r}_{c(j)}$ are the vectorial locations on the display grid, and $\sigma_G(t)$ and $\sigma_L(t)$ correspond to the widths of the neighborhood functions.
In order to give different features to SOM$_\mathrm{G}$ and SOM$_\mathrm{L}$, we set the following schedule functions for $\alpha_G(t)$, $\sigma_G(t)$, $\alpha_L(t)$, and $\sigma_L(t)$.

$$\begin{aligned} \alpha_G(t) &= \alpha_G(0)\,(1 - t/T)\,, & \sigma_G(t) &= \sigma_G(0)\,(1 - t/T)^2\,, \\ \alpha_L(t) &= \alpha_L(0)\left\{1 - (t/T)^{\frac{1}{2}}\right\}, & \sigma_L(t) &= \sigma_L(0)\,(1 - t/T)\,, \end{aligned} \tag{6}$$

where T is the maximum number of the learning.
Figure 1 shows an example of the neighborhood functions for $\alpha_G(0) = 0.9$, $\alpha_L(0) = 0.5$, $\sigma_G(0) = \sigma_L(0) = 4$, and $T = 6400$. The lines $G1$ and $L1$ show the case that the value of $\|\boldsymbol{r}_i - \boldsymbol{r}_{c(j)}\|$ is zero, while $G2$ and $L2$ show the case of $\|\boldsymbol{r}_i - \boldsymbol{r}_{c(j)}\| = 4\sqrt{2}$.
The neighborhood function for SOM$_\mathrm{G}$ in the [PHASE 3] is given as follows;

$$h_{Gsc(j),i}(t) = \alpha_{Gs}(t) \exp\left(-\frac{\|\boldsymbol{r}_i - \boldsymbol{r}_{c(j)}\|^2}{2\sigma_{Gs}^2(t)}\right). \tag{7}$$

$\alpha_{Gs}(t)$ and $\sigma_{Gs}(t)$ decrease with time according to the following equations;

$$\alpha_{Gs}(t) = \alpha_{Gs}(0)\,(1 - t/T_s)\,, \qquad \sigma_{Gs}(t) = \sigma_{Gs}(0)\,(1 - t/T_s)\,, \tag{8}$$

where T_s is the maximum number of the learning since the step (nSOM7).

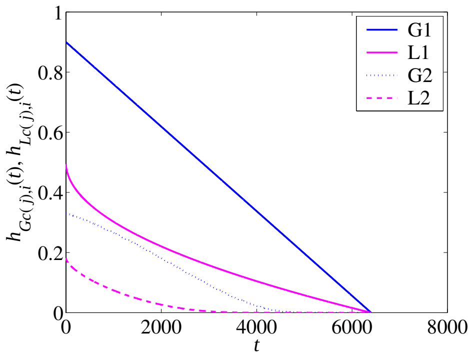

Figure 1: An example of neighborhood functions.

2.3 Simulation Results

As explained in the previous subsections, the main differences between SOM$_\mathrm{G}$ and SOM$_\mathrm{L}$ are only the initial states and the neighborhood functions. However, these differences cause interesting behaviors of n SOMs.

Figure 2 shows an example of input data and initial states of SOM$_\mathrm{G}$ and SOM$_\mathrm{L}$ for the case of $n = 2$ (namely the number of SOM$_\mathrm{L}$ is only one). Input data in Fig. 2(a) include 2-dimensional 1600 points (j=1600). Each SOM has 100 neurons (10 × 10).

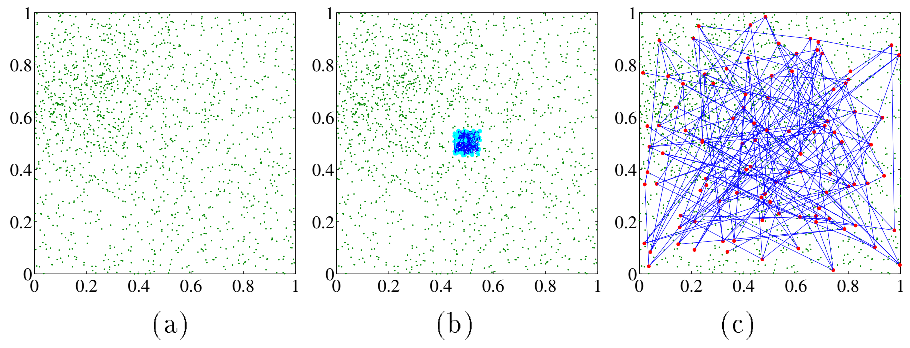

Figure 2: (a) Input data. (b) Initial state of SOM$_\mathrm{G}$. (c) Initial state of SOM$_\mathrm{L}$.

The parameters of the learning are chosen as follows;

$\alpha_G(0) = 0.9,\ \alpha_L(0) = 0.5,\ \sigma_G(0) = 2,\ \sigma_L(0) = 4,\ \alpha_{Gs}(0) = 0.5,\ \sigma_{Gs}(0) = 2,\ \varepsilon = 0.02.$

After repeating [PHASE 2] four times, furthermore [PHASE 3] are repeated four more times. The simulation result is shown in Fig. 3. Figures 3(a) and (b) show the states after [PHASE 2] and [PHASE 3], respectively. The network at the upper left corner is SOM$_\mathrm{L}$ and the other network is SOM$_\mathrm{G}$.

Because only one neuron of all the neurons in the both SOMs can be a winner for one input data according to (nSOM4), the two networks compete each other in [PHASE 2] as shown in Fig. 3(a). In the early stage of [PHASE 2], only SOM$_\mathrm{L}$ actively moves according to the input data, because the initial state of SOM$_\mathrm{L}$ covers the whole input space. Hence, SOM$_\mathrm{L}$ tends to move to the area where the input data are dense. In the late stage of [PHASE 2], SOM$_\mathrm{L}$ will not make a large move any more, because the learning rate $\alpha_L(t)$ decreases rapidly according

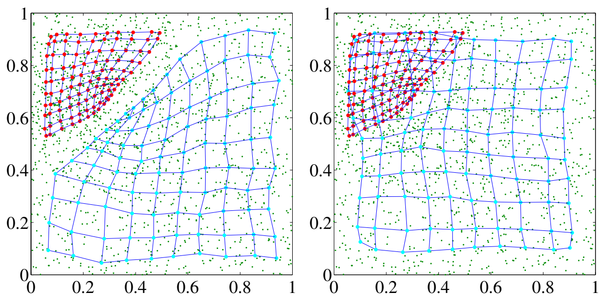
(a) (b)

Figure 3: Simulation results. (a) After [PHASE 2]. (b) After [PHASE 3].

to Eq. (6). While $\mathrm{SOM_G}$ actively moves all over the whole space except the area occupied by $\mathrm{SOM_L}$, because $\mathrm{SOM_L}$ stays in the limited area and the width of the neighborhood function $\sigma_G(t)$ decreases slowly according to Eq. (6).
In the [PHASE 3], $\mathrm{SOM_G}$ covers the whole input space beyond the area occupied by $\mathrm{SOM_L}$ as shown in Fig. 3(b).

3 Application to Clustering

The concept using n SOMs can be exploited to extract the data only in clusters of the input data including a lot of noises, because $\mathrm{SOM_L}$ can find such areas by themselves.

3.1 2-dimensional input data

At first, we consider 2-dimensional input data as shown in Fig. 4(a). The input data is generated artificially as follows. Total number of the input data is 1600. 25% of the input data are distributed within a range from 0.2 to 0.3 horizontally and from 0.7 to 0.8 vertically, and these data are called the cluster C_1. 50% of the input data are distributed in another cluster C_2, whose horizontal-values follow the normal distribution $N(0.7,\ 0.04)$, and the vertical-values $N(0.2,\ 0.0016)$. The remaining 25% of the input data are distributed between 0 and 1 at random.

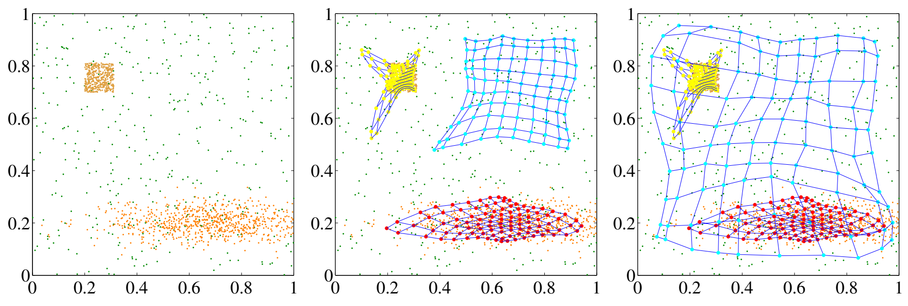
(a) (b) (c)

Figure 4: Clustering of 2-dimensional input data. (a) Input data. (b) Simulated result after [PHASE 2]. (b) Simulated result after [PHASE 3].

We use one $\mathrm{SOM_G}$ and two $\mathrm{SOM_L}$. Each SOM has 100 neurons (10×10), namely 3 SOMs have totally 300 neurons. The parameters of the learning are chosen as follows;

$\alpha_G(0) = 0.9$, $\alpha_L(0) = 0.5$, $\sigma_G(0) = \sigma_L(0) = 4$, $\alpha_{Gs}(0) = 0.7$, $\sigma_{Gs}(0) = 2$, $\varepsilon = 0.05$.

After repeating [PHASE 2] four times, furthermore [PHASE 3] are repeated four more times. The simulation results after [PHASE 2] and [PHASE 3] are shown in Figs. 4(b) and (c), respectively. We can see that two $\mathrm{SOM_L}$ stay around the two clusters.

In order to extract the input data only in the clusters, we calclate the distance between the input data $\boldsymbol{x}_j$ and $\boldsymbol{w}_{Ll}$ in $\mathrm{SOM}_{\mathrm{L}l}$ after [PHASE 2]. (Actually, for the purpose of the clustering, we do not need [PHASE 3].) If the calculated distance is smaller than R, the input data $\boldsymbol{x}_j$ is classified into the cluster corresponding to $\mathrm{SOM}_{\mathrm{L}l}$. Figures 5(a) and (b) show the input data classified into the clusters corresponding to $\mathrm{SOM_{L1}}$ and $\mathrm{SOM_{L2}}$, respectively, $R = 0.05$. As we can see from the figures, $\mathrm{SOM_L}$ can successfully extract the clusters.

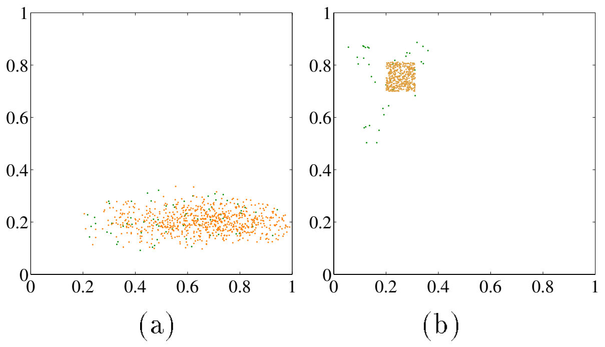
Figure 5: Extraction of clusters by n SOMs method. (a) Cluster 1. (b) Cluster 2.

Although, the k-means method is known to be not useful for the data including a lot of noises, we carry out the k-means method for the same input data for the comparison. Figure 6 shows the results obtained by using the k-means method, where the number of the cluster is set as $k = 3$. We can see that the clusters obtained by the k-means method include a lot of noises.

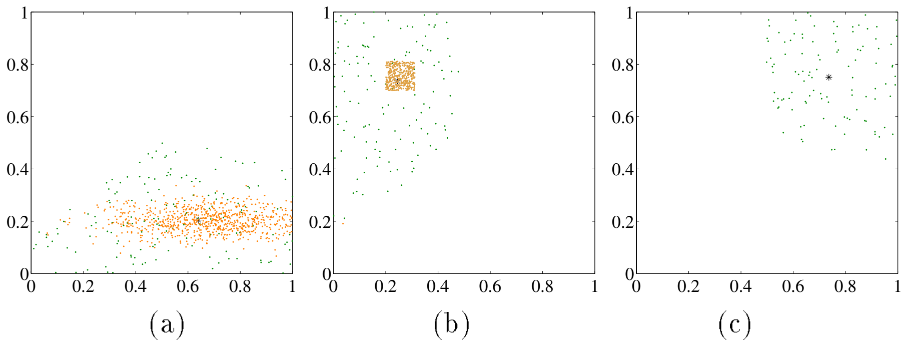
Figure 6: Extraction of clusters by k-means method. (a) Cluster 1. (b) Cluster 2. (c) Noises.

In order to evaluate the clustering ability of n SOMs quantitatively, we define the correct answer rate R_{CI} as follows;

$$R_{CI} = \frac{N_r - N_e}{N_{CI}}, \tag{9}$$

where N_{CI} is the true number of the input data within the cluster C_I, N_r is the obtained number of the desired input data within C_I, and N_e is the obtained number of undesired input data out of C_I. The calculated results are summarized in Table 1. We can evaluate the effectiveness of the method using n SOMs by this index value.

Table 1: Correct answer rate [%] for 2-dimensional input data.

	C_1	C_2
n SOMs method	86.80	91.28
k-means method	79.95	70.18

3.2 3-dimensional input data

Next, we carry out simulation for 3-dimensional input data shown in Fig. 7(a). The input data include two clusters and a lot of noises out of the clusters.

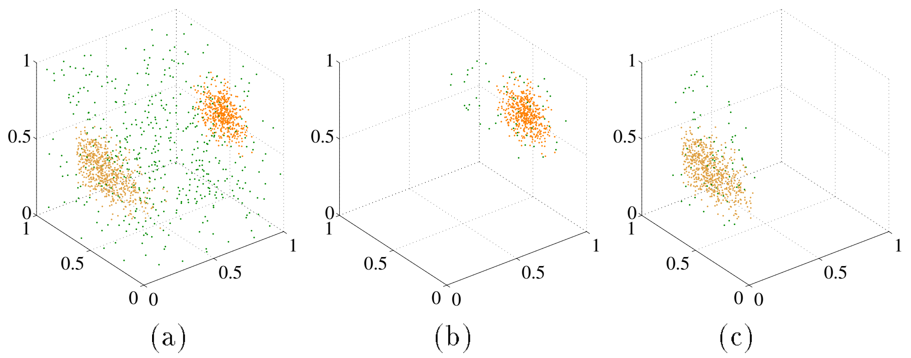

Figure 7: Clustering of 3-dimensional input data. (a) Input data. (b) Extracted cluster by $\mathrm{SOM_{L1}}$. (c) Extracted cluster by $\mathrm{SOM_{L2}}$.

Figures 7(b) and (c) show the extracted clusters by the n SOMs method. We can confirm that the noises are removed by $\mathrm{SOM_G}$ and only the cluster part can be extracted vely well. The correct answer rates are summarized in Table 2. We can confirm the clustering ability using n SOMs.

Table 2: Correct answer rate [%] for 3-dimensional input data.

	C_1	C_2
n SOMs method	89.00	86.15
k-means method	61.91	60.19

3.3 5-dimensional input data

Furthermore, we performed the simulation for 5-dimensional input data of 1600 points. This data include four clusters and a lot of noises, and the four clusters are generated by random

Table 3: 5-dimensional Gaussian data.

No.		Dimension					Probability [%]
		1	2	3	4	5	
C_1	Mean value	0.8	0.3	0.7	0.2	0.7	15
	Variance	0.0016	0.0081	0.0036	0.0009	0.0081	
C_2	Mean value	0.35	0.8	0.3	0.7	0.2	20
	Variance	0.0036	0.0004	0.01	0.0081	0.0036	
C_3	Mean value	0.6	0.9	0.1	0.1	0.9	15
	Variance	0.0004	0.0016	0.0009	0.0025	0.0009	
C_4	Mean value	0.2	0.1	0.8	0.4	0.3	20
	Variance	0.0016	0.01	0.0004	0.0225	0.0025	

Table 4: The correct answer rate [%] of 5-dimensional input data.

	C_1	C_2	C_3	C_4
n SOMs method	89.84	94.22	98.39	96.33
k-means method	54.69	73.86	85.48	73.33

Gaussian data as shown in Table 3. The parameters of the learning are chosen as follows; $\alpha_G(0) = 0.9$, $\alpha_L(0) = 0.6$, $\sigma_G(0) = \sigma_L(0) = 4$, $\alpha_{Gs}(0) = 0.7$, $\sigma_{Gs}(0) = 2$, $\varepsilon = 0.2$, $R = 0.2$.

The correct answer rate is summarized in Table 4. We can say that the method of using n SOMs are effective for higher-dimensional input data.

4 Conclusions

In this study, we have propose the method using simultaneously two kinds of SOMs whose features are different. We have investigated its competeing behavior caused by the difference of the initial states and the neighborhood functions. Further, we have applied the two kinds of SOMs to clustering of data including a lot of noises and have confirmed the efficiency.

References

[1] Y. Linde, A. Buzo and R. Gray (1980), An Algorithm for Vector Quantizer Design, *IEEE Transactions on Communications*, **vol. 28,** no. 1, pp. 84-85.

[2] T. Kohonen (1995), *Self-Organising Maps*, Berlin, Springer, vol. 30.

[3] Y. Cheng (1992), Clustering with Competing Self-Organizing Maps, *Proc. of IJCNN'92*, **vol. IV,** pp. 785-790.

[4] W. Wan and D. Fraser (1993), M2dSOMAP: Clustering and Classification of Remotely Sensed Imagery by Combining Multible Kohonen Self-Organizing Maps and Associative Memory, *Proc. of IJCNN'93*, **vol. III,** pp. 2464-2467.

[5] J. Vesanto and E. Alhoniemi (2000), Clustering of the Self-Organizing Map, *IEEE Transactions on Neural Networks*, **vol. 11,** no. 3, pp. 586-600.

[6] H. Matsushita, Y. Uwate and Y. Nishio (2005), Research on Improvement in Self-Organization Capability Using Two SOMs, *Proc. of NCSP'05*, pp. 307-310.