PAPER Special Section on Nonlinear Theory and its Applications

Tentacled Self-Organizing Map for Effective Data Extraction

Haruna MATSUSHITA[†a)], ***Student Member and*** **Yoshifumi NISHIO**[†b)], ***Member***

SUMMARY Since we can accumulate a large amount of data including useless information in recent years, it is important to investigate various extraction method of clusters from data including much noises. The Self-Organizing Map (SOM) has attracted attention for clustering nowadays. In this study, we propose a method of using plural SOMs (TSOM: Tentacled SOM) for effective data extraction. TSOM consists of two kinds of SOM whose features are different, namely, one self-organizes the area where input data are concentrated, and the other self-organizes the whole of the input space. Each SOM of TSOM can catch the information of other SOMs existing in its neighborhood and self-organizes with the competing and accommodating behaviors. We apply TSOM to data extraction from input data including much noise, and can confirm that TSOM successfully extracts only clusters even in the case that we do not know the number of clusters in advance.

key words: *self-organizing maps, clustering, data extraction, data segmentation*

1. Introduction

Since we can accumulate a large amount of data including useless information nowadays, it is important to investigate various extraction methods of clusters from data including much noise. Then, the Self-Organizing Map (SOM) has attracted attention in recent years. SOM is an unsupervised neural network introduced by Kohonen in 1982 [1] and is a model simplifying self-organization process of the brain. SOM can obtain a statistical feature of input data and can be applied to a wide field of data classifications [2]–[7]. Although many methods to extract clusters by using SOM have been proposed, it seems to be very difficult to construct a simple method of using SOM for universal input data.

In our past study, we have proposed a method of simultaneously using two kinds of SOM whose features are different (*n*SOM method) [8]. In the *n*SOM method, one self-organizes the area where 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 $\mathrm{SOM_G}$. We have investigated competing behavior of *n*SOM caused by the difference of the initial states and the neighborhood functions. Furthermore, we have applied *n*SOM to clustering of data including much noise and have confirmed its efficiency. However, *n*SOM method has two disadvantages in order to obtain an efficient clustering performance.

1) We must set the initial state of $\mathrm{SOM_G}$ avoiding clusters.
2) We must determine the appropriate number of SOMs, namely, we must know the number of clusters in advance.

Regarding the case 1), it is difficult to set the initial state properly in higher-dimensional space. Regarding the case 2), if we use too many SOMs in *n*SOM method, some SOMs compete with each other in the same cluster and the one cluster is extracted as wrong plural clusters.

We have also proposed the Peace SOM algorithm [9] which remedies the above problem 2). PSOM possesses both competing and accommodating abilities and is used after executing the *n*SOM method. This algorithm makes the competing SOMs in the same cluster to accommodate the cluster, while the SOMs situated far keeping to compete with other SOMs. We have investigated the competing and the accommodating behaviors of PSOM with applications to clustering input data including much noise. However, the learning process is a little bit complicated, because the PSOM algorithm must be used after executing the *n*SOM method, namely it is the two-tiered learning.

In this study, we propose a method of using plural SOMs (TSOM: Tentacled SOM) for effective data extraction. TSOM method possesses both abilities of *n*SOM and PSOM, namely, TSOM has context sensitive behaviors of competing and accommodating. The plural SOMs of TSOM consist of the two kinds of SOM, $\mathrm{SOM_L}$ and $\mathrm{SOM_G}$. $\mathrm{SOM_L}$ self-organizes the area where input data are concentrated, while $\mathrm{SOM_G}$ self-organizes the whole of the input space, in the same way as *n*SOM method. In the TSOM algorithm, one $\mathrm{SOM_G}$ and plural $\mathrm{SOM_L}$ are used. Each SOM of TSOM can catch the information of other SOMs existing in its neighborhood — hence the name "tentacled." TSOM has the following three key rules.

1. Winner SOM: All SOMs of TSOM have one winner neuron, respectively. However, neurons of all SOMs are NOT updated. One or more winner SOMs are decided for one input data, and the weight vectors of only the winner SOM are updated.
2. The update direction y_{Gi} of $\mathrm{SOM_G}$: the update direction of $\mathrm{SOM_G}$ is decided by the positional relation with $\mathrm{SOM_L}$.
3. Learning function: $\mathrm{SOM_L}$ are updated according to the learning function which is the same function used in PSOM algorithm.

These key roles make the behavior of TSOM interesting. Namely, $\mathrm{SOM_L}$ and $\mathrm{SOM_L}$ which are located close to each

Manuscript received December 4, 2006.
Final manuscript received May 22, 2007.

[†]The authors are with the Department of Electrical and Electronic Engineering, Tokushima University, Tokushima-shi, 770–8506 Japan.
a) E-mail: haruna@ee.tokushima-u.ac.jp
b) E-mail: nishio@ee.tokushima-u.ac.jp

DOI: 10.1093/ietfec/e90–a.10.2085

other accommodate each other, contrary, $\mathrm{SOM_G}$ and $\mathrm{SOM_L}$ which are located close to each other compete against with each other. $\mathrm{SOM_L}$ and $\mathrm{SOM_L}$, which are located far apart, also compete against with each other.

In Sect. 2, we explain the key points and the learning algorithm of the proposed TSOM algorithm in detail. The competing and the accommodating behaviors of TSOM are investigated in the Sect. 3 with applications to data extraction from input data including much noise. For 2 and 3-dimensional input data, extraction ability of clusters is evaluated both visually and quantitatively using a correct answer rate. We also apply TSOM to 5-dimensional input data and confirm the extraction ability for higher-dimensional data. We can see that TSOM successfully extracts clusters even in the case that we do not know the number of clusters in advance.

2. Tentacled Self-Organizing Map (TSOM)

In this study, we propose a method of using plural SOMs (TSOM: Tentacled SOM) for effective data extraction. The plural SOMs of TSOM contains the two kinds of SOM, namely $\mathrm{SOM_G}$ and $\mathrm{SOM_L}$. Each SOM of TSOM exchanges their own position information each other and self-organizes with the competing and accommodating behaviors.

In the TSOM algorithm, we use totally n SOMs, that is one $\mathrm{SOM_G}$ and $(n-1)$ $\mathrm{SOM_L}$; namely $\mathrm{SOM_{L1}}, \mathrm{SOM_{L2}}, \cdots, \mathrm{SOM_{L(n-1)}}$. Each SOM consists of m neurons located at regular 2-dimensional grid. The lattice of the grid is either hexagonal or rectangle. Each neuron i of $\mathrm{SOM_G}$ has a d-dimensional weight vector $\boldsymbol{w}_{Gi} = (w_{Gi1}, w_{Gi2}, \cdots, w_{Gid})$, and each neuron i of $\mathrm{SOM_{L}}l$ $(l = 1, 2, \cdots, n-1)$ also has the weight vector $\boldsymbol{w}_{Lli} = (w_{Lli1}, w_{Lli2}, \cdots, w_{Llid})$. The input data is denoted by $\boldsymbol{x}_j = (x_{j_1}, x_{j_2}, \cdots, x_{j_d})$ $(j = 1, 2, \cdots, N)$.

TSOM has three key rules, namely, "Winner SOM," "The update direction of $\mathrm{SOM_G}$" and "Learning function." We explain these key points with the learning algorithm in this section.

2.1 Winner SOM Deciding Rule

We use plural SOMs for TSOM algorithm and find winner neurons in every SOM for one input data by calculating the distance between $\boldsymbol{x}_j$ and all the weight vector. In other words, each winner neuron is the neuron with the weight vector closest to $\boldsymbol{x}_j$ in each SOM, namely, all SOMs of TSOM have one winner neuron, respectively. The winner neuron of $\mathrm{SOM_G}$ is denoted by c_G, and the winner neuron of $\mathrm{SOM_{L}}l$ is denoted by c_{Ll}. However, neurons of all SOMs are NOT updated, the weight vectors updated are that of only the winner SOM.

We explain the method for deciding the winner SOM. This plays a very important role in the TSOM algorithm and produces the competing behavior and the accommodating behavior of TSOM. The winner SOM is determined by the position of each winner neuron of each SOM.

[For $\mathrm{SOM_L}$]

(1) The winner SOM is $\mathrm{SOM_{L}}l$ if the winner neuron c_{Ll} of $\mathrm{SOM_{L}}l$ has the weight vector closest to $\boldsymbol{x}_j$ in all the winner neurons (as Fig. 1(a)).

(2) Plural $\mathrm{SOM_L}$ can be the winner SOMs if the distances between $\boldsymbol{x}_j$ and winner neurons of plural $\mathrm{SOM_L}$ are less than $R_L(t)$, namely $\|\boldsymbol{w}_{Ll c_{Ll}} - \boldsymbol{x}_j\| \leq R_L(t)$ (as Fig. 1(b)).

[For $\mathrm{SOM_G}$]

(3) The winner SOM is $\mathrm{SOM_G}$ if the winner neuron c_G of $\mathrm{SOM_G}$ has the weight vector closest to $\boldsymbol{x}_j$ in all the winner neurons, and all winner neurons of $\mathrm{SOM_L}$ are more than $R_L(t)$ away from $\boldsymbol{x}_j$ (as Fig. 1(c)).

(4) $\mathrm{SOM_G}$ can not be the winner if distances between $\boldsymbol{x}_j$ and winner neurons of $\mathrm{SOM_L}$ are not more than $R_L(t)$ although the winner neuron c_G of $\mathrm{SOM_G}$ has the weight vector closest to $\boldsymbol{x}_j$ in all the winner neurons (as Fig. 1(d)).

In the case (4), the weight vectors $\boldsymbol{w}_G$ of $\mathrm{SOM_G}$ are updated to minus direction. The value of deciding the update direction of $\mathrm{SOM_G}$ for the input data $\boldsymbol{x}_j$ can be written as;

$$y_{Gj} = \begin{cases} 1, & \text{if case (3)} \\ -1, & \text{if case (4)} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

In other words, $\mathrm{SOM_G}$ and $\mathrm{SOM_L}$, which are located close

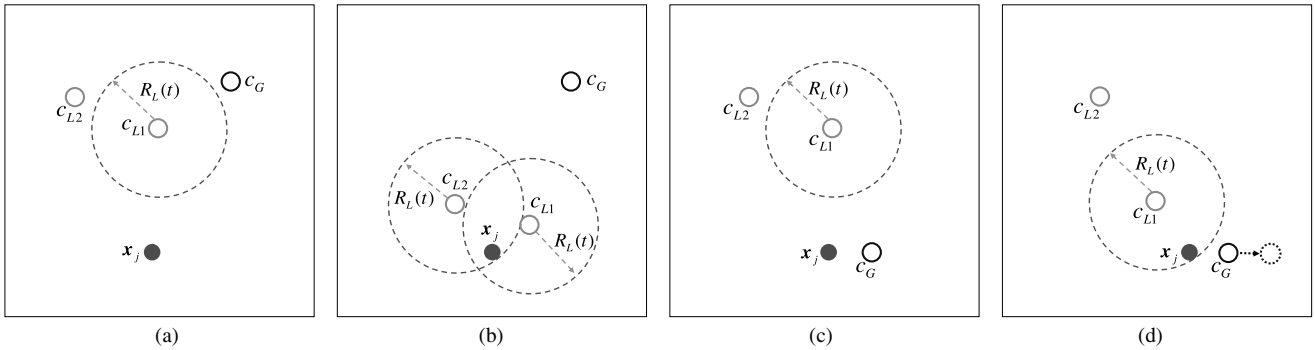

Fig. 1 Example of winner SOM and the location of winner neurons. (a) Winner SOM is $\mathrm{SOM_{L1}}$. (b) Winner SOMs are $\mathrm{SOM_{L1}}$ and $\mathrm{SOM_{L2}}$. (c) Winner SOM is $\mathrm{SOM_G}$. (d) Winner SOM is $\mathrm{SOM_{L1}}$.

to each other, compete against each other.

$R_L(t)$ plays an tentacle of each SOM and increases with time according to;

$$R_L(t) = R_{Lm}\left\{1 - \left(\frac{0.005}{R_{Lm}}\right)^{t/T}\right\}, \quad (2)$$

where R_{Lm} is a maximum value of $R_L(t)$, and T is the maximum number of the learning.

2.2 Learning Algorithm

The range of the elements of the input data $\boldsymbol{x}_j$ are assumed to be from 0 to 1. The initial values of all the weight vector of $\mathrm{SOM_G}$ and $\mathrm{SOM_L}$ are given between 0 and 1 at random.

(TSOM1) An input data $\boldsymbol{x}_j$ is inputted to all the neurons of $\mathrm{SOM_G}$ and $\mathrm{SOM_L}$ simultaneously in parallel.
(TSOM2) Distances between $\boldsymbol{x}_j$ and all the weight vectors of all SOM are calculated. The winner neuron of each SOM is the neuron with the weight vector closest to $\boldsymbol{x}_j$ in each SOM;

$$c_G = \arg\min_i\{\|\boldsymbol{w}_{Gi} - \boldsymbol{x}_j\|\},$$
$$c_{Ll} = \arg\min_i\{\|\boldsymbol{w}_{Lli} - \boldsymbol{x}_j\|\}, \quad (3)$$

where $\|\cdot\|$ is the distance measure, in this study, Euclidian distance.
(TSOM3) One or more winner SOMs are determined by the distance between $\boldsymbol{x}_j$ and the winner neurons, according to II. *A*.
(TSOM4) The weight vectors of neurons of $\mathrm{SOM_G}$ and $\mathrm{SOM_L}$ are updated.

The weight vectors of neurons in $\mathrm{SOM_G}$ are updated according to the following equation;

$$\boldsymbol{w}_{Gi}(t+1) = \boldsymbol{w}_{Gi}(t) + y_{Gj} \cdot h_{Gc_G,i}(t)(\boldsymbol{x}_j - \boldsymbol{w}_{Gi}(t)), \quad (4)$$

where y_{Gj} is the value of deciding the update direction of $\mathrm{SOM_G}$ for $\boldsymbol{x}_j$ determined by Eq. (1), and $h_{G_{c_G},i}(t)$ is the neighborhood function of $\mathrm{SOM_G}$ described as follows;

$$h_{G_{c_G},i}(t) = \alpha(t)\exp\left(-\frac{\|\boldsymbol{r}_i - \boldsymbol{r}_{c_G}\|^2}{2\sigma^2(t)}\right), \quad (5)$$

where $\boldsymbol{r}_i$ is the vectorial locations on the display grid of neuron i, $\alpha(t)$ is the learning rate, and $\sigma(t)$ corresponds to the widths of the neighborhood function. Both $\alpha(t)$ and $\sigma(t)$ decrease monotonically with time according to the following equation;

$$\alpha(t) = \alpha(0)(1 - t/T), \quad \sigma(t) = \sigma(0)(1 - t/T), \quad (6)$$

where T is the maximum number of the learning.

If the winner SOM is $\mathrm{SOM_{L}}l$, the weight vectors of neurons in $\mathrm{SOM_{L}}l$ are updated, according to;

$$\boldsymbol{w}_{Lli}(t+1) = \boldsymbol{w}_{Lli}(t) + h_{Llc_{Ll},i}(t)(\boldsymbol{x}_j - \boldsymbol{w}_{Lli}(t)), \quad (7)$$

The function $h_{Llc_{Ll},i}(t)$ is the neighborhood function of $\mathrm{SOM_L}$ and it is described as follows;

$$h_{Llc_{Ll},i}(t) = p_{c_{Ll}}(t)\exp\left(-\frac{\|\boldsymbol{r}_i - \boldsymbol{r}_{c_{Ll}}\|^2}{2\sigma^2(t)}\right), \quad (8)$$

where $p_{c_{Ll}}(t)$ is the learning function, and it is explained in the next subsection.

(TSOM5) The steps from (TSOM1) to (TSOM4) are repeated for all the input data, namely from $j = 1$ to $j = N$.

2.3 Learning Function

In our past study, we have proposed the learning function. The learning function produces the competing behavior and the accommodating behavior of $\mathrm{SOM_L}$. The value of the learning function is determined by the distance between the input vector $\boldsymbol{x}_j$ and the winner neuron c_{Ll} of $\mathrm{SOM_{L}}l$, according to;

$$p_{c_{Ll}}(t) = \alpha(t)\exp\left(-\frac{\|\boldsymbol{w}_{Llc_{Ll}} - \boldsymbol{x}_j\|^2}{2\sigma_P^2}\right), \quad (9)$$

where $\boldsymbol{w}_{Llc_{Ll}}$ is the weight vector of the winner neuron c_{Ll} of $\mathrm{SOM_{L}}l$, σ_P corresponds to the width of the learning function, and $\alpha(t)$ corresponds to the maximum value of the learning function according to Eq. (6).

It is desirable to choose σ_P as a small value such as less than 0.1, because this is a key point to decide their competing behavior and accommodating behavior. Namely, when plural $\mathrm{SOM_L}$ are winner SOM, some winner $\mathrm{SOM_L}$ whose winner neurons are near the input data are significantly updated and this means that these $\mathrm{SOM_L}$ tend to accommodate the input data each other. Conversely, some winner $\mathrm{SOM_L}$ whose winner neurons are far apart from the input data are updated very little and this means that these $\mathrm{SOM_L}$ keep to compete with other $\mathrm{SOM_L}$.

In other words, some $\mathrm{SOM_L}$, which are located close to each other, accommodate each other, contrary, some $\mathrm{SOM_L}$, which are located far apart, compete against each other.

3. Application to Data Extraction

3.1 2-Dimensional Input Data

First, we consider 2-dimensional input data as shown in Fig. 2(a). The input data is generated artificially as follows. Total number of the input data N 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.

Because the input data include 2 clusters, we use one

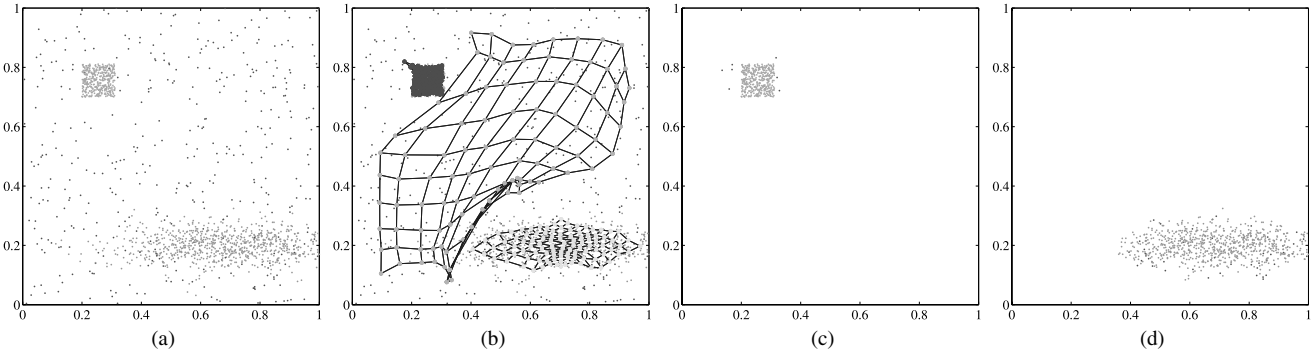

Fig. 2 Clustering of 2-dimensional input data using TSOM. (a) Input data. (b) Simulated result of TSOM ($n = 3$). (c) Clusters extracted by $\mathrm{SOM_{L}}1$. (d) Clusters extracted by $\mathrm{SOM_{L}}2$.

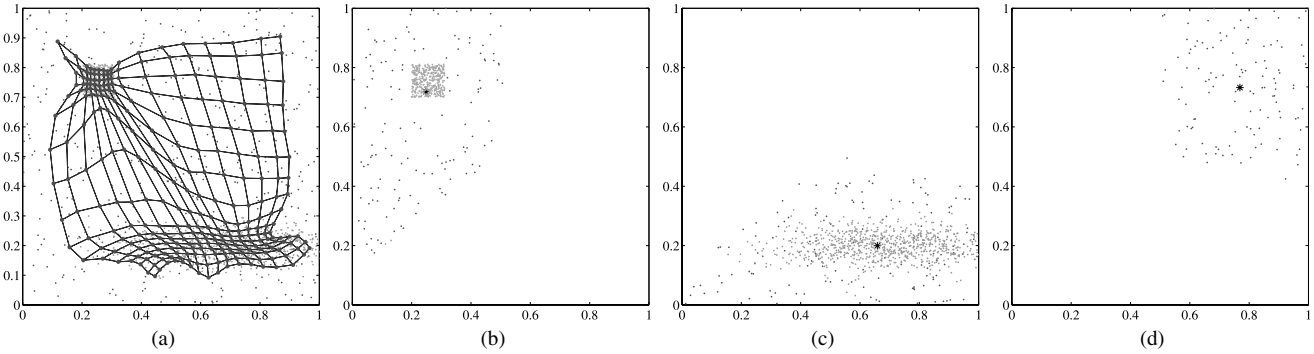

Fig. 3 Comparison of simulation results. (a) Learning result of conventional SOM. (b) Extraction of Cluster1 by k-means method. (c) Extraction of Cluster2 by k-means method. (d) Extraction of noise by k-means method.

$\mathrm{SOM_{G}}$ and two $\mathrm{SOM_{L}}$ ($n = 3$). Each SOM has 100 neurons (10×10), namely 3 SOMs have totally 300 neurons, and these neurons are arranged on a 2-dimensional grid. We repeat the learning four times for all input data. The parameters of the learning are chosen as follows;

$$\alpha(0) = 0.5, \sigma(0) = 3, \sigma_P = 1/16, R_{Lm} = 0.2.$$

The simulation result is shown in Fig. 2(b). We can see that two $\mathrm{SOM_{L}}$ stay around the two clusters by the competing behavior of TSOM method.

In order to extract only clusters of the input data including much noise, we calculate the distance between the input data $\boldsymbol{x}_j$ and the weight vectors in $\mathrm{SOM_{L}}l$ after learning of TSOM. If the calculated distance is smaller than a threshold value R, the input data $\boldsymbol{x}_j$ is classified into the cluster corresponding to $\mathrm{SOM_{L}}l$. Figures 2(c) and (d) show the input data classified into the clusters corresponding to $\mathrm{SOM_{L}}1$ and $\mathrm{SOM_{L}}2$, respectively ($R = 0.05$). As we can see from the figures, $\mathrm{SOM_{L}}$ can successfully extract the clusters, and the noise are removed by $\mathrm{SOM_{G}}$. This is the result of TSOM when we know the number of the clusters in advance.

We carry out the conventional SOM and the k-means method for the same input data for the comparison. From Fig. 3(a), we can confirm that the conventional SOM self-organizes the input data the influence of noise. On the other hand, Fig. 3(b)–(d) shows the results obtained by using the k-means method, where the number of the cluster is set as $k = 3$ (namely, two clusters and noise). We can see that the clusters obtained by the k-means method include much noise.

Next, we assume that we do not know the appropriate number of clusters in advance. We simulate for the same input data using one $\mathrm{SOM_{G}}$ and three $\mathrm{SOM_{L}}$, namely the number of $\mathrm{SOM_{L}}$ is more than the number of clusters. We use the same parameters as Fig. 2. Figures 4(a)–(h) show the learning process of TSOM, and the simulation result is Fig. 4(h). We can see that all $\mathrm{SOM_{L}}$ stay around the two clusters and $\mathrm{SOM_{G}}$ self-organizes only noise. We observe this in more detail, the two $\mathrm{SOM_{L}}$, which stay around the same cluster C_2, are accommodating in one cluster without competing although one $\mathrm{SOM_{L}}$ is redundant. On the other hand, $\mathrm{SOM_{L}}$ situated on the other cluster C_1 is competing with other two $\mathrm{SOM_{L}}$ without accommodating.

Let us consider the learning process. In the early stage of learning as Figs. 4(a) and (b), $\mathrm{SOM_{G}}$ is self-organizing the cluster C_1. However, $\mathrm{SOM_{L}}$ edges out $\mathrm{SOM_{G}}$ and is self-organizing this cluster as Fig. 4(d), because $\mathrm{SOM_{G}}$ are updated to the minus direction, according to Eq. (1). In the meanwhile, two $\mathrm{SOM_{L}}$ are competing in the same cluster C_2 as Figs. 4(a)–(d), however, these two $\mathrm{SOM_{L}}$ are accommodating in one cluster as the learning progressed. This is because $\mathrm{SOM_{L}}$ and $\mathrm{SOM_{L}}$, which are located close to

each other, accommodate each other, according to Eq. (8), namely, some SOM_L whose winner neurons are close to the input data are significantly updated.

We carry out the extraction of clusters from Fig. 4(h). Figures 5(a)–(c) show that the input data are classified into the clusters corresponding to SOM_{L1}, SOM_{L2} and SOM_{L3}, respectively ($R = 0.05$).

In order to investigate the ability of the TSOM algorithm, we carry out the simulations with excessive number of SOMs. The parameters are the same as Fig. 2. Figure 6 shows the result using six SOMs for the same input data, namely 5 SOM_L just for 2 clusters. As we can see from the figures, TSOM can extract 2 clusters as if they know the number of the clusters.

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

$$R_{CI} = \frac{N_{rI} - N_{eI}}{N_{CI}}, \quad (I = 1, 2, \cdots), \tag{10}$$

where N_{CI} is the true number of the input data within the cluster C_I, N_{rI} is the obtained number of the desired input data within C_I, and N_{eI} is the obtained number of undesired input data out of C_I. The calculated results are summarized in Table 1. When the number of SOM_L is equal to the number of the clusters (n=3), we can evaluate the effectiveness of the method of using TSOM method by this index value.

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

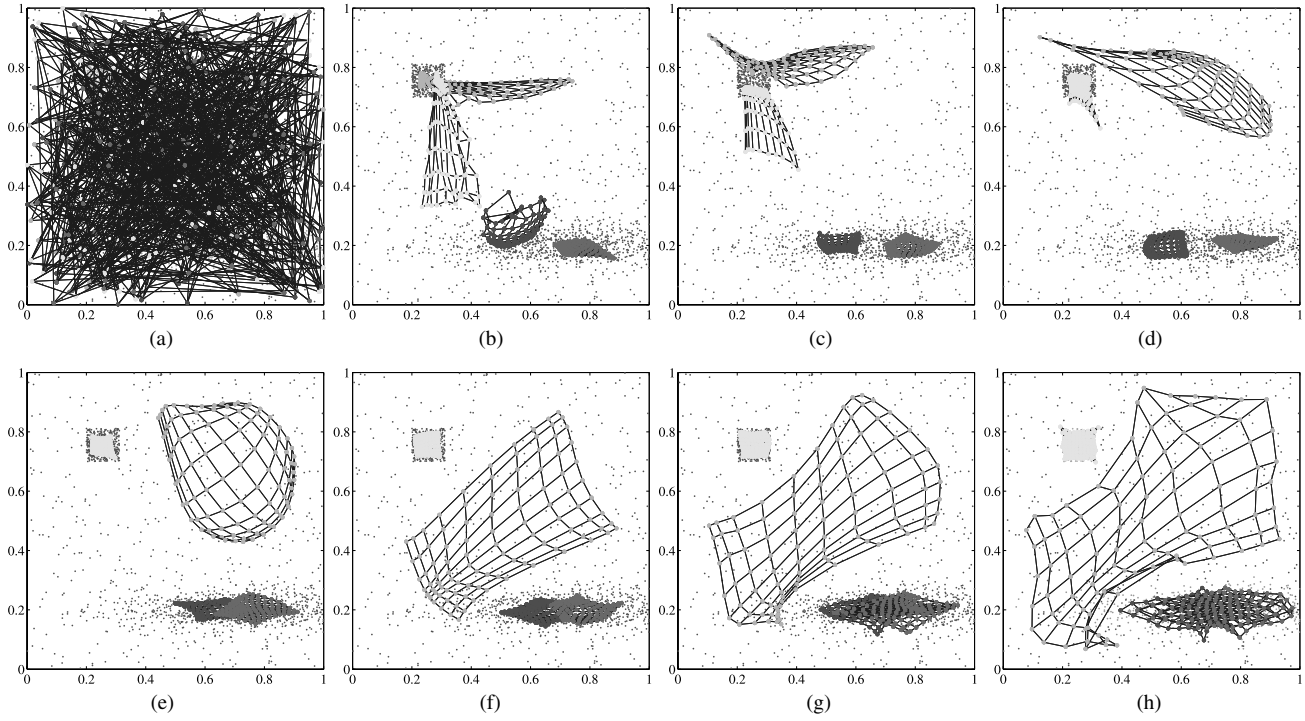

Fig. 4 Clustering using TSOM with inappropriate number of SOM_L. (a) Initial state ($t = 0$). (b) $t = 200$. (c) $t = 400$. (d) $t = 600$. (e) $t = 1000$. (f) $t = 2000$. (g) $t = 3000$. (h) Simulation results ($t = 6400$).

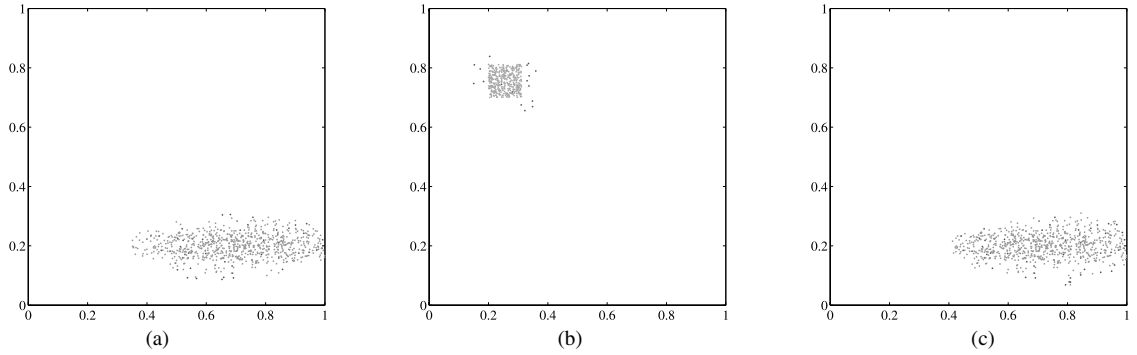

Fig. 5 Input data extracted by SOM_L of TSOM. (a) SOM_{L1}. (b) SOM_{L2}. (c) SOM_{L3}.

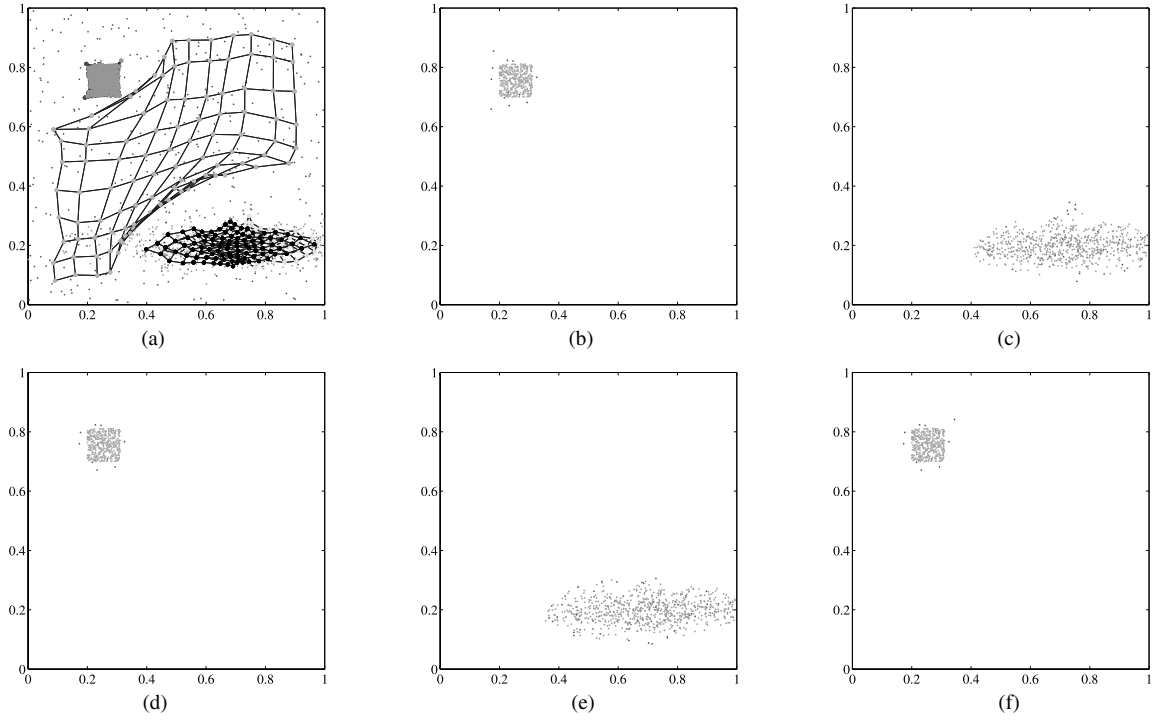

Fig. 6 Cluster extraction using excessive number of SOMs. (a) Learning result of TSOM ($n = 6$). (b) $\mathrm{SOM_{L}}1$. (c) $\mathrm{SOM_{L}}2$. (d) $\mathrm{SOM_{L}}3$. (e) $\mathrm{SOM_{L}}4$. (f) $\mathrm{SOM_{L}}5$.

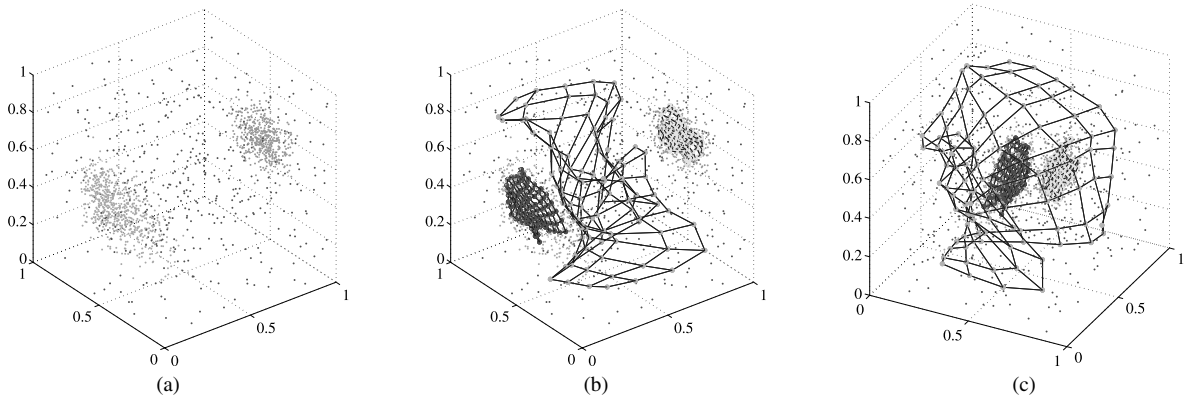

Fig. 7 Clustering of 3-dimensional input data using TSOM. (a) Input data. (b) Learning result of TSOM ($n = 3$) (c) Result from another angle.

much noise out of the clusters.

Figure 7(b) shows the learning result of TSOM ($n = 3$), and the same result from another angle is shown in Fig. 7(c). We can see that two $\mathrm{SOM_{L}}$ stay around the two clusters and $\mathrm{SOM_{G}}$ self-organizes the area of noise. The input data classified into the clusters corresponding to each $\mathrm{SOM_{L}}$ are shown in Figs. 8(a) and (b), respectively ($R = 0.15$). We can confirm that the noise are removed by $\mathrm{SOM_{G}}$ and only the cluster part can be extracted very well.

Furthermore, we carry out this simulation for the same input data using TSOM ($n = 5$) and k-means method ($k = 3$). The correct answer rates are summarized in Table 2. As we can see from this table, the clustering ability of using TSOM method is effectual whether or not we know the number of clusters.

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

n	Method	$\mathrm{SOM_{L}}l$				
		1	2	3	4	5
3	k-means	81.78	66.00	-	-	-
	TSOM	87.36	97.73	-	-	-
4	TSOM	85.48	95.62	83.08	-	-
6	TSOM	96.50	84.01	96.97	86.09	96.74

3.3 5-Dimensional Input Data

Furthermore, we perform the simulation for 5-dimensional input data of 1600 points. This data include four clusters and

much noise, and the four clusters are generated by random Gaussian data as shown in Table 3.

We carry out simulations for the cases of $n = 5$ (which is the same number of clusters) and $n = 7$ (we do not know the number of cluster). The parameters of the learning for $n = 5$ and $n = 7$ are chosen as follows;

(For $n = 5$)

$$\alpha(0) = 0.7, \sigma(0) = 4, \sigma_P = 1/16, R_{Lm} = 0.2,$$

(For $n = 7$)

$$\alpha(0) = 0.7, \sigma(0) = 4, \sigma_P = 1/16, R_{Lm} = 0.25.$$

The results of the calculated correct answer rates ($R = 0.3$) are summarized in Table. 4. We can confirm that the correct answer rates are high.

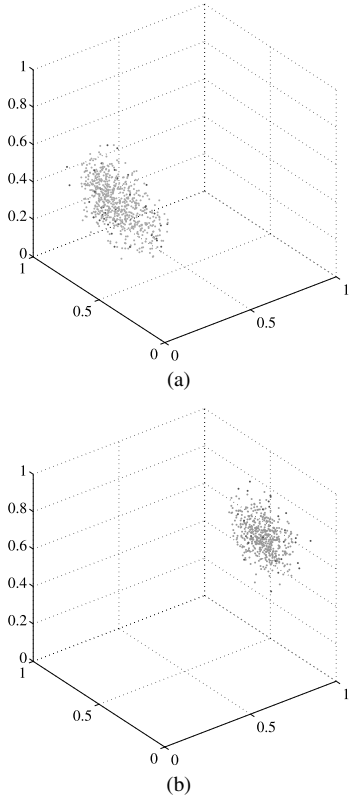

Fig. 8 3-D input data extracted by SOM_L. (a) SOM_{L1}. (b) SOM_{L2}.

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

n	Method	SOM_{Ll}			
		1	2	3	4
3	k-means	65.98	59.29	-	-
	TSOM	89.75	94.26	-	-
5	TSOM	93.85	87.67	90.49	94.47

4. Conclusions

In this study, we have proposed the Tentacled SOM (TSOM) algorithm which is a method of using plural SOMs. TSOM possesses both abilities of nSOM and PSOM. Each SOM of TSOM can catch the information of other SOMs existing in its neighborhood and self-organizes with the competing and accommodating behaviors. We have investigated its behaviors. Furthermore, we have applied TSOM to extract clusters in the case that we do not know the number of the clusters in advance, and have confirmed its efficiency.

From these results, we can say that the proposed method using TSOM can be applied to not only data extraction, but also data segmentation in the future. In addition, by proposing new visualization methods for TSOM, TSOM has the possibility of opening up completely new ways to visualize high-dimensional data.

References

[1] T. Kohonen, Self-organizing Maps, vol.30, Springer, Berlin, 1995.
[2] Y. Cheng, "Clustering with competing self-organizing maps," Proc. International Joint Conference on Neural Networks, vol.IV, pp.785–790, 1992.
[3] W. Wan and D. Fraser, "M2dSOMAP: Clustering and classification of remotely sensed imagery by combining multiple kohonen self-organizing maps and associative memory," Proc. International Joint Conference on Neural Networks, vol.III, pp.2464–2467, 1993.
[4] L. Xu, A. Krzyzak, and E. Oja, "Rival penalized competitive learning for clustering analysis, RBF net, and curve detection," IEEE Trans. Neural Netw., vol.4, no.4, pp.636–649, 1993.
[5] B. Fritzke, "Growing Grid — A self-organizing network with constant neighborhood range and adaptation strength," Neural Process. Lett., vol.2, no.5, pp.9–13, 1995.
[6] J. Vesanto and E. Alhoniemi, "Clustering of the self-organizing map," IEEE Trans. Neural Netws., vol.11, no.3, pp.586–600, 2002.

Table 4 Correct answer rate [%] for 5-dimensional input data.

n	Method	SOM_{Ll}					
		1	2	3	4	5	6
5	TSOM	80.00	97.93	96.35	93.69	-	-
7	TSOM	96.77	95.15	63.52	95.79	70.39	98.37

Table 3 5-dimensional Gaussian data.

No.		Coordinate axes					Probability [%]
		x_1	x_2	x_3	x_4	x_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	

[7] I. Lapidot, H. Guterman, and A. Cohen, "Unsupervised speaker recognition based on competition between self-organizing maps," IEEE Trans. Neural Netw., vol.13, no.4, pp.877–887, 2002.
[8] H. Matsushita and Y. Nishio, "Competing behavior of two kinds of self-organizing maps and its application to clustering," IEICE Trans. Fundamentals, vol.E90-A, no.4, pp.865–871, April 2007.
[9] H. Matsushita and Y. Nishio, "Competing and accommodating behaviors of peace SOM," RISP J. Signal Processing, vol.10, no.5, pp.371–376, 2006.

Haruna Matsushita was born in Tokushima, Japan, in 1982. She received the B.E., M.E. degrees from Tokushima University, Tokushima, Japan, in 2005 and 2007. She is currently working towards Ph.D. degree at the same university. Her research interests include theory and application of Self-Organizing Maps. She is a student member of the IEEE.

Yoshifumi Nishio received B.E., M.E., and Ph.D. degrees in electrical engineering from Keio University, Yokohama Japan, in 1988, 1990, and 1993, respectively. In 1993, he joined the Department of Electrical and Electronic Engineering at Tokushima University, Tokushima Japan, where he is currently an Associate Professor. From May 2000 he spent a year in the Laboratory of Nonlinear Systems (LANOS) at the Swiss Federal Institute of Technology Lausanne (EPFL) as a Visiting Professor. His research interests include analysis and application of chaos in electrical circuits, analysis of synchronization in coupled oscillatory circuits, development of analyzing methods for nonlinear circuits, theory and application of cellular neural networks, and neural network architecture. He was the Chair of the IEEE CAS Technical Committee on Nonlinear Circuits and Systems (NCAS) during 2004–2005 and is currently the Steering Committee Secretary of the IEICE Research Society of Nonlinear Theory and its Applications (NOLTA). He is serving as an Associate Editor for the IEEE Transactions on Circuits and Systems Part I and the RISP Journal of Signal Processing. He is a senior member of the IEEE, and a member of the RISP.