# Self-Organizing Map with False Neighbor Degree between Neurons for Effective Self-Organization

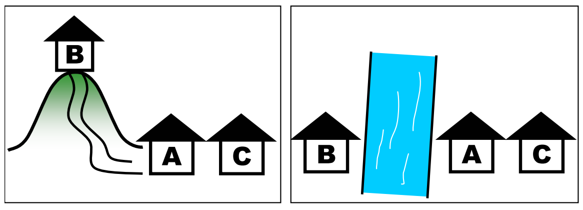
Haruna Matsushita and Yoshifumi Nishio
Department of Electrical and Electronic Engineering, Tokushima University
2-1 Minami-Josanjima, Tokushima 770-8506, JAPAN
email: {haruna, nishio}@ee.tokushima-u.ac.jp

Keywords: Self-Organizing Map (SOM), unsupervised learning, learning

***Abstract*—** In the real world, it is not always true that the nextdoor house is close to my house, in other words, “neighbors” are not always “true neighbors”. In this study, we propose a new Self-Organizing Map (SOM) algorithm, SOM with False Neighbor degree between neurons (called FN-SOM). The behavior of FN-SOM is investigated with learning for various input data. We confirm that FN-SOM can obtain the more effective map reflecting the distribution state of input data than the conventional SOM and Growing Grid.

## 1 Introduction

Since we can accumulate a huge amount of data in recent years, it is important to investigate various clustering methods [1]. Then, the Self-Organizing Map (SOM) has attracted attention for its clustering properties. 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 statistical feature of input data and is applied to a wide field of data classifications. We can obtain the map reflecting the distribution state of input data using SOM. In the learning algorithm of SOM, a winner, which is a neuron with the weight vector closest to the input vector, and its neighboring neuron are updated, regardless of the distance between the input vector and the neighboring neuron. For this reason, if we apply SOM to clustering of the input data which includes some clusters located at distant location, there are some inactive neurons between clusters. Because inactive neurons are on a part without the input data, we are misled into thinking that there are some input data between clusters.



(a) (b)

Figure 1: What are the “neighbors”? The houses B and C is A’s next-door neighbors on the left and on the right, respectively. (a) The house B is at the top of a mountain. (b) The river between A and B does not have a bridge.

Meanwhile, in the real world, it is not always true that the next-door house is close to my house. For example, a case that the next-door house is at the top of a mountain whereas my house is at the foot (as Fig. 1(a)), and another case that there is a river, which does not have a bridge, between my house and my next-door house (as Fig. 1(b)). This means that “neighbors” are not always “true neighbors”.

On the other side, the synaptic strength is not constant in the brain. So far, the Growing Grid network was proposed in 1985 [3]. Growing Grid increases the neighborhood distance between neurons by increasing the number of neurons. However, there are few researches changing the synaptic strength as far as we know even though there are algorithms which increase the number of neurons or consider rival neurons [4], [5].

In our past study, we proposed the algorithm which changes the neighborhood distance between neurons [6]. However, the algorithm used the rank order of the distances between the input data and weight vectors of neurons in addition to changing the neighborhood distance. Thus the algorithm did not work well if the positions of all the weight vectors of the neurons were not taken into consideration. Moreover, the algorithm needs a lot of calculation amount because we have to calculate the rank order at every updating of the weight vector.

In this study, we propose a new SOM algorithm, SOM with False Neighbor degree between neurons (called FN-SOM) without the rank order. False-neighbor degrees are allocated between adjacent rows and adjacent columns of FN-SOM. We find the neuron $q$ which has become the winner least frequently, and the neurons, which is the most distant from $q$ in a set of direct topological neighbors of $q$, are said to be “false neighbors” of $q$. The initial values of all of the false-neighbor degrees are set to zero, however, they are increased with learning, and the false-neighbor degrees act as a burden of the distance between map nodes when the weight vectors of neurons are updated.

We explain the learning algorithm of FN-SOM in detail in Section 4. The learning behaviors of FN-SOM for 2-

dimensional input data, Swiss Roll data and Iris data are investigated. Learning performance is evaluated both visually and quantitatively using two measurements. Furthermore, the results are compared with those obtained by the conventional SOM and Growing Grid. We can confirm that there are few inactive neurons using FN-SOM, and FN-SOM can obtain the most effective map reflecting the distribution state of input data in the three algorithms.

## 2 Self-Organizing Map

We explain the learning algorithm of the conventional Self-Organizing Map (SOM). SOM consists of $n \times m$ neurons located at a regular low-dimensional grid, usually a 2-D $n \times m$ rectangular grid. The basic SOM algorithm is iterative. Each neuron $i$ has a $d$-dimensional weight vector $\boldsymbol{w}_i = (w_{i1}, w_{i2}, \cdots, w_{id})$ $(i = 1, 2, \cdots, nm)$. The initial values of all the weight vectors are given over the input space at random. 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.

**(SOM1)** An input vector $\boldsymbol{x}_j$ is inputted to all the neurons at the same time in parallel.
**(SOM2)** Distances between $\boldsymbol{x}_j$ and all the weight vectors are calculated. The winner, denoted by $c$, is the neuron with the weight vector closest to the input vector $\boldsymbol{x}_j$;

$$c = \arg\min_i \{\|\boldsymbol{w}_i - \boldsymbol{x}_j\|\}, \tag{1}$$

where $\|\cdot\|$ is the distance measure, Euclidean distance.
**(SOM3)** The weight vectors of the neurons are updated as

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) + h_{c,i}(t)(\boldsymbol{x}_j - \boldsymbol{w}_i(t)), \tag{2}$$

where $t$ is the learning step. $h_{c,i}(t)$ is called the neighborhood function and is described as a Gaussian function;

$$h_{c,i}(t) = \alpha(t) \exp\left(-\frac{\|\boldsymbol{r}_i - \boldsymbol{r}_c\|^2}{2\sigma^2(t)}\right), \tag{3}$$

where $\|\boldsymbol{r}_i - \boldsymbol{r}_c\|$ is the distance between map nodes $c$ and $i$ on the map grid, $\alpha(t)$ is the learning rate, and $\sigma(t)$ corresponds to the width of the neighborhood function. Both $\alpha(t)$ and $\sigma(t)$ decrease with time, in this study, we use following equations;

$$\alpha(t) = \alpha_0(1 - t/t_{\max}), \quad \sigma(t) = \sigma_0(1 - t/t_{\max}), \tag{4}$$

where $\alpha_0$ and $\sigma_0$ are the initial value of $\alpha$ and $\sigma$, respectively, and $t_{max}$ is the maximum number of the learning.
**(SOM4)** The steps from (SOM1) to (SOM3) are repeated for all the input data.

## 3 Growing Grid

We explain an overview of the Growing Grid. The network of Growing Grid consists of $nm$ neurons located at a rectangular $n \times m$ grid. Each neuron has an $d$-dimensional weight vector $\boldsymbol{w}_i$ as the conventional SOM. A winning frequency $\gamma_i$ is associated with each neuron and is set to zero initially.

An input vector $\boldsymbol{x}_j$ is inputted to all the neurons, and a winner $c$ is found according to Eq. (1). The weight vectors of the neurons are updated according to

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) + h_{Gc,i}(t)(\boldsymbol{x}_j - \boldsymbol{w}_i(t)), \tag{5}$$

where $h_{Gc,i}(t)$ is the neighborhood function of Growing Grid;

$$h_{Gc,i}(t) = \alpha_0 \exp\left(-\frac{{d_g}^2(c,i)}{2{\sigma_0}^2}\right), \tag{6}$$

where $\alpha_0$ is a constant learning rate, and $\sigma_0$ is a constant width parameter. $d_g(c,i)$ is the distance on the grid between a winner $c$ and each neuron $i$ and is calculated by city-block distance (which is also known as $L_1$-norm). At each learning step, the winning frequency of $c$ is incremented by ${\gamma_c}^{\text{new}} = {\gamma_c}^{\text{old}} + 1$.

After $n \times m \times \lambda_g$ number of learning steps have been performed, we determine the neuron $q$ which has become the winner most frequently;

$$q = \arg\max_i \{\gamma_i\}. \tag{7}$$

We find the neuron $f$ which is with the most different weight vector in 1-neighbor of $q$. We insert a new row (or column) between $q$ and $f$. The weight vectors of the new neurons are interpolated from their neighbors which does increase the density of weight vectors in the vicinity of $\boldsymbol{w}_q$. The number $n$ of rows (or $m$ of columns) are increased, then all the winning frequency is reset. We continue with the next round of learning unless $nm \geq nm_{\max}$ is fulfilled.

The growth process is finished, we perform the fine-tune the weight vectors using a decreasing learning rate. We perform $t'_{\max} = n \times m \times \lambda_f$ steps according to Eq. (5) using $\alpha(t') = \alpha_0(\alpha_1/\alpha_0)^{t'/t'_{\max}}$. $t'$ denotes the learning step in the fine-tuning phase which starts after the growth step is finished.

## 4 SOM with False Neighbor Degree (FN-SOM)

We explain a new SOM algorithm, SOM with False Neighbor Degree between neurons (FN-SOM). False-neighbor degrees of rows $R_r$ $(1 \leq r \leq n-1)$ are allocated between adjacent rows of FN-SOM with the size of $n \times m$ grid (as Fig. 2). Likewise, false-neighbor degrees of columns $C_k$ $(1 \leq k \leq m-1)$ are allocated between adjacent columns of FN-SOM. In other words, $R_1$ means the false-neighbor degree between neurons of the 1st row and the 2nd row, and $C_4$ is the false-neighbor degree between neurons of the 4th column and the 5th column. The initial values of all of the false-neighbor degrees are set to zero, and the initial values of all the weight vectors are given over the input space at

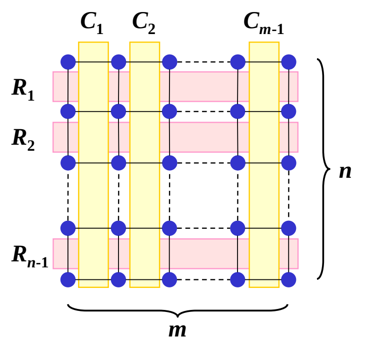


Figure 2: A false-neighbor degree of row $R_r$ $(1 \leq r \leq n-1)$ and column $C_k$ $(1 \leq k \leq m-1)$. Neurons of FN-SOM are located at a $n \times m$ rectangular grid.

random. Moreover, a winning frequency $\gamma_i$ is associated with each neuron and is set to zero initially.

### *Learning Steps*

**(FN-SOM1)** An input vector $\boldsymbol{x}_j$ is inputted to all the neurons at the same time in parallel.
**(FN-SOM2)** Distances between $\boldsymbol{x}_j$ and all the weight vectors are calculated, and a winner $c$ is found according to Eq. (1).
**(FN-SOM3)** Increment the winning frequency of winner $c$ by $\gamma_c{}^{\text{new}} = \gamma_c{}^{\text{old}} + 1$.
**(FN-SOM4)** The neighboring distances between the winner $c$ and the other neurons are calculated. For instance, for two neurons $s_1$, which is located at $r_1$-th row and $k_1$-th column, and $s_2$, which is located at $r_2$-th row and $k_2$-th column, the neighboring distance is defined as the following measure;

$$d_f(s_1, s_2) = (|r_1 - r_2| + \sum_{r=r_1}^{r_2-1} R_r)^2 + (|k_1 - k_2| + \sum_{k=k_1}^{k_2-1} C_k)^2, \tag{8}$$

where $r_1 < r_2$, $k_1 < k_2$, namely, $\sum_{r=r_1}^{r_2-1} R_r$ means the sum of the false-neighbor degrees between the rows $r_1$ and $r_2$, and $\sum_{k=k_1}^{k_2-1}$ means the sum of the false-neighbor degrees between the column $k_1$ and $k_2$.
**(FN-SOM5)** The weight vectors of the neurons are updated as

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) + h_{F\,c,i}(t)(\boldsymbol{x}_j - \boldsymbol{w}_i(t)), \tag{9}$$

where $h_{F\,c,i}(t)$ is the neighborhood function of FN-SOM:

$$h_{F\,c,i}(t) = \alpha(t) \exp\left(-\frac{d_f(c,i)}{2\sigma^2(t)}\right). \tag{10}$$

**(FN-SOM6)** If $\sum_{i=1}^{nm} \gamma_i \geq \lambda$ is satisfied, we find the false-neighbors and increase the false-neighboring degree, according to steps from (FN-SOM7) to (FN-SOM10). If not, we perform step (FN-SOM11). In other words, we consider the false-neighbors every time when the learning steps are performed for $\lambda$ input data.

### *Considering False-Neighbors*

**(FN-SOM7)** We find the neuron $q$ which has become the winner least frequently:

$$q = \arg\min_i \{\gamma_i\}, \tag{11}$$

where, if more than one $\gamma_i$ is minimum, the neuron $i$ with the smallest index is chosen.
**(FN-SOM8)** A false-neighbor $f$ of $q$ is chosen from the set of direct topological neighbors of $q$ denoted as $N_{q1}$. $f$ is the neuron whose weight vector is most distant from $q$:

$$f = \arg\max_i \{\|\boldsymbol{w}_i - \boldsymbol{w}_q\|\}, \quad i \in N_{q1} \tag{12}$$

**(FN-SOM9)** A false-neighbor degree between $q$ and its false neighbor $f$, $R_r$ or $C_k$, is increased. If $q$ and $f$ are in the $r$-th row and in the $k$-th and $(k+1)$-th column (as Fig. 3(a)), the false-neighbor degree $C_k$ between columns $k$ and $k+1$ is increased according to

$$C_k{}^{\text{new}} = C_k{}^{\text{old}} + \left\{1 - \exp\left(-\frac{\|\boldsymbol{w}_f - \boldsymbol{w}_q\|^4}{2\sigma_F{}^2}\right)\right\}, \tag{13}$$

where $\sigma_F$ is the constant width parameter of the Gaussian function.

In the same way, if $q$ and $f$ are in the $k$-th column and in the $(r+1)$-th and $r$-th row (as Fig. 3(b)), the false-neighbor degree $R_r$ between rows $r$ and $r+1$ is also increased according to

$$R_r{}^{\text{new}} = R_r{}^{\text{old}} + \left\{1 - \exp\left(-\frac{\|\boldsymbol{w}_f - \boldsymbol{w}_q\|^4}{2\sigma_F{}^2}\right)\right\}. \tag{14}$$

**(FN-SOM10)** The winning frequency of all the neurons are reset to zero:

$$\gamma_i = 0. \tag{15}$$

**(FN-SOM11)** The steps from (FN-SOM1) to (FN-SOM10) are repeated for all the input data.

## 5 Experimental Results

We apply FN-SOM to various input data and compare FN-SOM with the conventional SOM and Growing Grid.

### 5.1 For 2-dimensional data

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 $N$ is 1200, and the input data include three clusters. 400 data are distributed within a range from 0.1 to 0.9 horizontally and from 0.05

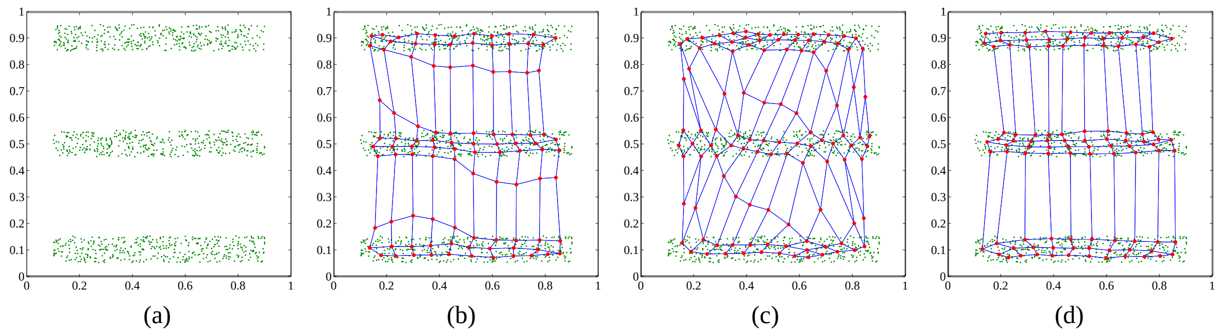


Figure 4: Learning results of three algorithms for 2-D data. (a) Input data. (b) Conventional SOM. (c) Growing Grid. (d) FN-SOM.

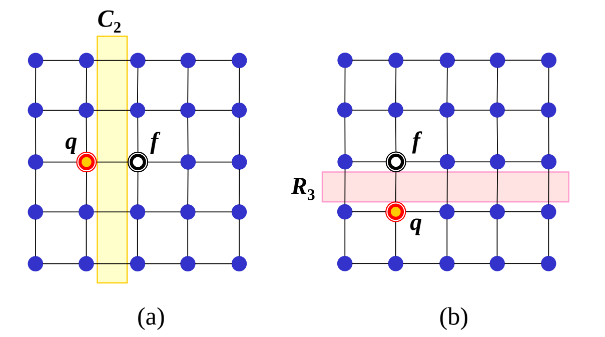


Figure 3: Increment the false-neighbor degree. (a) $q$ and its false-neighbor $f$ are in the 3rd row and in the 2nd and 3rd column, respectively. Then, the false-neighbor degree $C_2$ between columns 2 and 3 is increased by Eq. (13). (b) $q$ and $f$ are in the 2nd column and in the 4th and 3rd row, respectively. Then, the false-neighbor degree $R_3$ between rows 3 and 4 is increased by Eq. (14).

to 0.15 vertically. The other 400 data are distributed within a range from 0.1 to 0.9 horizontally and from 0.45 to 0.55 vertically. The remaining 400 data are distributed within a range from 0.1 to 0.9 horizontally and from 0.85 to 0.95 vertically. All the input data are sorted at random.

Both the conventional SOM and FN-SOM has $nm = 100$ neurons ($10 \times 10$). Growing Grid starts learning with a $2 \times 2$ neurons, and new rows and columns are inserted as long as the number of neurons is less than $nm_{\max} = 100$. We repeat the learning 15 times for all input data, namely $t_{\max} = 18000$. The parameters of the learning are chosen as follows;

(For SOM)

$$\alpha_0 = 0.3,\ \sigma_0 = 4,$$

(For Growing Grid)

$$\alpha_0 = 0.1,\ \sigma_0 = 0.9,\ \lambda_g = 20,\ \alpha_1 = 0.005,\ \lambda_f = 100,$$

(For FN-SOM)

$$\alpha_0 = 0.3,\ \sigma_0 = 4,\ \sigma_F = 0.05,\ \lambda = 500,$$

where we use the same $\alpha_0$ and $\sigma_0$ to SOM and FN-SOM for the comparison and the confirmation of the false-neighbor degree effect.

The learning results of the conventional SOM and Growing Grid are shown in Figs. 4(b) and (c), respectively. We can see that there are some inactive neurons between three clusters. The other side, the result of FN-SOM is shown in Fig. 4(d). We can see from this figure that there are no inactive neurons between three clusters, and FN-SOM can obtain the more effective map reflecting the distribution state of input data than SOM and Growing Grid.

Furthermore, in order to the learning performance of FN-SOM in comparison with the conventional SOM and Growing Grid, we use the following two measurements to evaluate the training performance of the three algorithms.

**Quantization Error $Q$:** This measures the average distance between each input vector and its winner;

$$Q = \frac{1}{N}\sum_{j=1}^{N} \|\boldsymbol{x}_j - \bar{\boldsymbol{w}}_j\|^2, \tag{16}$$

where $\bar{\boldsymbol{w}}_j$ is the weight vector of the corresponding winner of the input vector $\boldsymbol{x}_j$. Therefore, the small value $Q$ is more desirable.

**Neuron Utilization $U$:** This measures the percentage of neurons that are the winner of one or more input vector in the map [5];

$$U = \frac{1}{nm}\sum_{i=1}^{nm} u_i, \tag{17}$$

where $u_i = 1$ if the neuron $i$ is the winner of one or more input data. Otherwise, $u_i = 0$. Thus, $U$ nearer 1.0 is more desirable.

The calculated two measurements are shown in Table. 1. The quantization error $Q$ of FN-SOM is the smallest value in the three algorithms, and by using FN-SOM, the quantization error $Q$ has improved 18.9% from using the conventional SOM. This is because the result of FN-SOM has no inactive neurons, therefore, the more neurons can self-organize the input data. This is confirmed by the neuron utilization $U$. The neuron utilization $U$ of FN-SOM is the largest value in the three algorithms and is 1.0 which is the maximum value. It means that all the neurons of FN-SOM are the winner of one or more input data, namely, no neurons are inactive neurons.

Table 1: Quantization error $Q$ and Neuron utilization $U$ for 2-dimensional input data.

| | SOM | Growing Grid | FN-SOM |
|---|---|---|---|
| $Q$ | $6.2756 \times 10^{-4}$ | $7.1131 \times 10^{-4}$ | $5.0902 \times 10^{-4}$ |
| $U$ | 0.8200 | 0.8056 | 1.0 |

## 5.2 For Swiss Roll data

Next, we consider "Swiss Roll" data used by Tenenbaum *et al.* [7], as shown in Fig. 5. Total number of the input data $N$ is 1000, and the input data are normalized and are sorted at random.

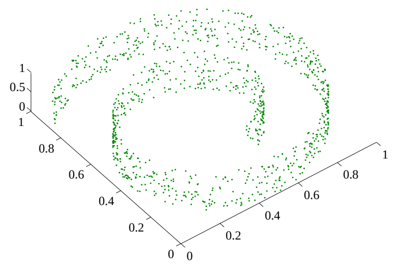


Figure 5: Swiss Roll data for 3-dimensional input data.

We repeat the learning 15 times for all input data, namely $t_{\max} = 15000$. The learning conditions are the same used in Fig. 4 except $\sigma_F = 0.03$ and $\lambda_g = 300$ for FN-SOM.

The learning results of the three algorithms are shown in Figs. 6(a)-(c), respectively. Furthermore, Figs. 6(d)-(f) show the results of Figs. 6(a)-(c) in 2-D (namely, X-Y coordinate), respectively. We can see from these figures that FN-SOM can obtain the most effective map reflecting the distribution state of input data.

The calculated the quantization error $Q$ and the neuron utilization $U$ are shown in Table. 2. We confirm that the quantization error $Q$ of FN-SOM is the smallest value in the three algorithms, and $Q$ of FN-SOM has improved $14.6\%$ from using the conventional SOM. Moreover, the neuron utilization $U$ of FN-SOM is the largest value in the three algorithms and is 1.0, which is the maximum value, as in the case of the 2-dimensional input data. From this table and Fig. 6, we can say that the result of FN-SOM has the fewest inactive neurons.

Table 2: Quantization error $Q$ and Neuron utilization $U$ for Swiss Roll data.

| | SOM | Growing Grid | FN-SOM |
|---|---|---|---|
| $Q$ | 0.0121 | 0.0123 | 0.0104 |
| $U$ | 0.9400 | 0.9821 | 1.0 |

## 5.3 For Iris data

Furthermore, we apply FNN-SOM to the real world clustering problem. We use the Iris plant data [8] as real data. This data is one of the best known databased to be found in pattern recognition literatures [9]. The data set contains three clusters of 50 instances respectively, where each class refers to a type of iris plant. The number of attributes is four as the sepal length, the sepal width, the petal length and the petal width, namely, the input data are 4-dimension. The three classes correspond to *Iris setosa*, *Iris versicolor* and *Iris virginica*, respectively. *Iris setosa* is linearly separable from the other two, however *Iris versicolor* and *Iris virginica* are not linearly separable from each other.

We repeat the learning 100 times for all input data, namely $t_{\max} = 15000$. The input data are normalized and are sorted at random. The learning conditions are the same used in Fig. 4 except $\sigma_F = 0.02$ for FN-SOM.

The calculated quantization error $Q$ and the neuron utilization $U$ are shown in Table. 3. We confirm that the quantization error $Q$ of FN-SOM is the smallest value in the three algorithms, and $Q$ of FN-SOM has improved $16.7\%$ from using the conventional SOM. This is because the result of FN-SOM hardly has inactive neurons between *Iris setosa* and the other two, therefore, the more neurons can self-organize the data of *Iris versicolor* and *Iris virginica*. The neuron utilization $U$ of FN-SOM is the largest value in the three algorithms. From these results, we can confirm the efficiency of FN-SOM.

Table 3: Quantization error $Q$ and Neuron utilization $U$ for Iris data.

| | SOM | Growing Grid | FN-SOM |
|---|---|---|---|
| $Q$ | 0.0018 | 0.0027 | 0.0015 |
| $U$ | 0.7300 | 0.7315 | 0.7800 |

## 6 Conclusions

In this study, we have proposed a new SOM algorithm, SOM with False Neighbor degree between neurons (called FN-SOM). False-neighbor degrees are allocated between adjacent rows and adjacent columns of FN-SOM. The initial values of all of the false-neighbor degrees are set to zero, however, they are increased with learning, and the false-neighbor degrees act as a burden of the distance be-

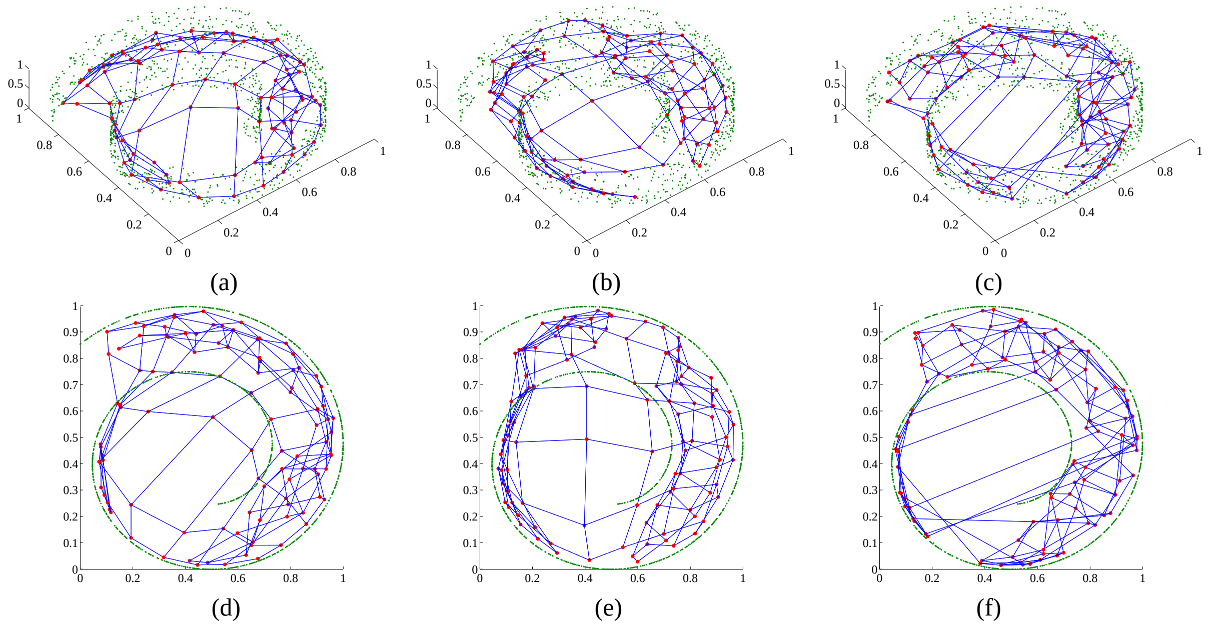


Figure 6: Learning results of three algorithms for Swiss Roll data. (a) Conventional SOM. (b) Growing Grid. (c) FN-SOM. (d) Result of SOM shown in 2-D (X-Y coordinate). (e) Result of Growing Grid shown in 2-D. (f) Result of FN-SOM shown in 2-D.

tween map nodes when the weight vectors of neurons are updated. We have applied FN-SOM to 2-dimensional data, Swiss Roll data and Iris data, and we have investigated the learning behaviors of FN-SOM. Furthermore, the results were compared with those obtained by the conventional SOM and Growing Grid. We have confirmed that the quantization error of FN-SOM was the smallest value in the three algorithms. Moreover, the neuron utilization of FN-SOM was the largest value in the three algorithms. From these results, we have confirmed the efficiency of FN-SOM.

In the future we intend to investigate FN-SOM in more detail in particular its use for high-dimensional data.

## References

[1] J. Vesanto and E. Alhoniemi, "Clustering of the Self-Organizing Map," *IEEE Trans. Neural Networks*, vol. 11, no. 3, pp. 586–600, 2002.

[2] T. Kohonen, *Self-organizing Maps*, Berlin, Springer, 1995.

[3] B. Fritzke, "Growing Grid – a self-organizing network with constant neighborhood range and adaptation strength," *Neural Processing Letters*, vol. 2, no. 5, pp. 9–13, 1995.

[4] L. Xu, A. Krzyzak and E. Oja, "Rival penalized competitive learning for clustering analysis, RBF net, and curve detection," *IEEE Trans. Neural Networks*, vol. 4, no. 4, pp. 636–649, 1993.

[5] Y. Cheung and L. Law, "Rival-Model Penalized Self-Organizing Map," *IEEE Trans. Neural Networks*, vol. 18, no. 1, pp. 289–295, 2007.

[6] H. Matsushita and Y. Nishio, "Self-Organizing Map Considering False Neighboring Neuron," *Proc. of ISCAS'07*, 2007.

[7] J. B. Tenenbaum, V. de Silva and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[8] D. J. Newman, S. Hettich, C. L. Blake and C. J. Merz, UCI Repository of Machine Learning Database, 1998, [http://www.ics.uci.edu/ mlearn/MLRepository.html].

[9] R. A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Annual Eugenics*, no.7, part II, pp. 179–188,1936.