社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

Batch-Learning Self-Organizing Map with False-Neighbor Degree 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
E-mail: †{haruna,nishio}@ee.tokushima-u.ac.jp

Abstract This study proposes a Batch-Learning Self-Organizing Map with False-Neighbor degree between neurons (called BL-FNSOM). False-neighbor degrees are allocated between adjacent rows and adjacent columns of BL-FNSOM. 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. BL-FNSOM changes the neighborhood relationship more flexibly according to the situation and the shape of data although using batch learning. We apply BL-FNSOM to some input data and confirm that FN-SOM can obtain a more effective map reflecting the distribution state of input data than the conventional Batch-Learning SOM.

Key words self-organizing maps (SOM), clustering, unsupervised learning

1. Introduction

Since we can accumulate a huge amount of data in recent years, it is important to investigate various clustering methods [1]. The Self-Organizing Map (SOM) is an unsupervised neural network [2] and has attracted attention for its clustering properties. In the learning algorithm of SOM, a winner, which is a neuron closest to the input data, and its neighboring neurons are updated, regardless of the distance between the input data and the neighboring neurons. For this reason, if we apply SOM to clustering of the input data which includes some clusters located at distant locations, there are some inactive neurons between clusters where without the input data. Because inactive neurons are on a part without the input data, we are misled into thinking that there are some input data between clusters.

Then, what are the "neighbors"? In the real world, it is not always true that the next-door house is close to my house. In other words, "neighbors" are not always "true neighbors". In addition, the relationship between neighborhoods is not fixed, but keeps changing with time. It is important to change the neighborhood relationship flexibly according to the situation.

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 is not much research changing the synaptic strength even though there are algorithms which increase the number of neurons or consider rival neurons [4].

In our past study, we proposed the SOM with False-Neighbor degree between neurons (called FN-SOM) [5]. The false-neighbor degrees act as a burden of the distance between map nodes when the weight vectors of neurons are updated. We have confirmed 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.

Meanwhile, there are two well-known learning algorithms for SOM: sequential learning and batch learning. In the sequential learning, the winner for an input data is found and the weight vectors are updated immediately. However, the sequential learning has some problems as a large calculation amount and a dependence on order of the input data. In the batch learning, the updates are deferred to the end of a leaning, namely the presentation of the whole dataset. Batch-Learning SOM (BL-SOM) [2] is used to speed up the calculating time and to remove the dependence on order of the input data.

In this study, we propose a Batch-Learning Self-Organizing Map with False-Neighbor degree between neurons (called BL-FNSOM). We improve the previously proposed FN-SOM and apply the batch learning to 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 between map nodes when the weight vectors of neurons are updated. BL-FNSOM changes the neighborhood relationship more flexibly according to the situation and the shape of data although using batch learning.

We explain the learning algorithm of BL-FNSOM in detail in Section 3. We apply BL-FNSOM to a uniform input data to investigate an effect of the false-neighbor degree. The learning behaviors of BL-FNSOM for 2-dimensional in-

put data and 3-dimensional data, which have some clustering problem, are investigated in Section 4. In addition, we apply BL-FNSOM to a real world data set, Iris data. Learning performance is evaluated both visually and quantitatively using three measurements and is compared with the conventional BL-SOM. We confirm that FN-SOM can obtain the more effective map reflecting the distribution state of input data than BL-SOM.

2. Batch-Learning Self-Organizing Map (BL-SOM)

We explain a batch learning algorithm of SOM in detail. SOM consists of $n \times m$ neurons located at 2-dimensional rectangular grid. Each neuron i is represented by a d-dimensional weight vector $\boldsymbol{w}_i = (w_{i1}, w_{i2}, \cdots, w_{id})$ $(i = 1, 2, \cdots, nm)$, where d is equal to the dimension of the input vector $\boldsymbol{x}_j = (x_{j1}, x_{j2}, \cdots, x_{jd})$ $(j = 1, 2, \cdots, N)$. Also batch learning algorithm is iterative, but instead of using a single data vector at a time, the whole data set is presented to the map before any adjustments are made. The initial weight vectors of BL-SOM are given at orderly position based on the first and second principal components of the d-dimensional space by the Principal Component Analysis (PCA) [6].

(BLSOM1) An input vector $\boldsymbol{x}_j$ is inputted to all the neurons at the same time in parallel.
(BLSOM2) Distances between $\boldsymbol{x}_j$ and all the weight vectors are calculated. The input vector $\boldsymbol{x}_j$ belongs to a winner neuron which is with the weight vector closest to $\boldsymbol{x}_j$.

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

denotes the index of the winner of the input vector $\boldsymbol{x}_j$, where $\|\cdot\|$ is the distance measure, Euclidean distance.
(BLSOM3) After repeating the steps (BLSOM1) and (BLSOM2) for all the input data set, all the weight vectors are updated as

$$\boldsymbol{w}_i^{\text{new}} = \frac{\sum_{j=1}^{N} h_{c_j,i} \boldsymbol{x}_j}{\sum_{j=1}^{N} h_{c_j,i}}, \tag{2}$$

where t denotes a training step and $h_{c_j,i}$ is the neighborhood function around the winner c_j:

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

where $\boldsymbol{r}_i$ and $\boldsymbol{r}_{c_j}$ are coordinates of the i-th and the winner c unit on competitive layer, namely, $\|\boldsymbol{r}_i - \boldsymbol{r}_{c_j}\|$ is the distance between map nodes c_j and i on the map grid. $\sigma(t)$ decreases with time, in this study, we use following equation;

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

where σ_0 is the initial value of σ, and t_{max} is the maximum number of the learning.
(BLSOM4) The steps from (BLSOM1) to (BLSOM3) sufficient time.

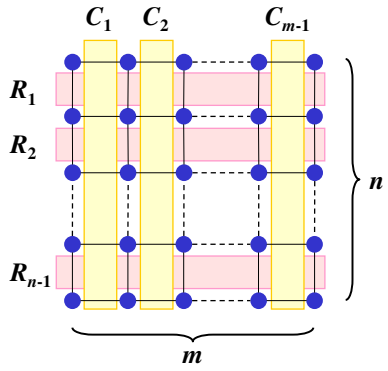


Fig. 1 A false-neighbor degree of row R_r $(1 \leqq r \leqq n-1)$ and column C_k $(1 \leqq k \leqq m-1)$. Neurons of FN-SOM are located at a $n \times m$ rectangular grid.

3. BL-SOM with False-Neighbor Degree (BL-FNSOM)

We explain a Batch Learning SOM with False-Neighbor Degree between neurons (BL-FNSOM) in detail. False-neighbor degrees of rows R_r $(1 \leqq r \leqq n-1)$ are allocated between adjacent rows of BL-FNSOM with the size of $n \times m$ grid (as Fig. 1).

Likewise, false-neighbor degrees of columns C_k $(1 \leqq k \leqq m-1)$ are allocated between adjacent columns of BL-FNSOM. 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 the false-neighbor degrees are set to zero. and the initial values of all the weight vectors are given over the input space at random. Moreover, a winning frequency γ_i is associated with each neuron and is set to zero initially. The initial weight vectors of BL-FNSOM are given as BL-SOM method. Learning algorithm of BL-FNSOM contains two steps: *Learning steps* and *Considering False-Neighbors*.

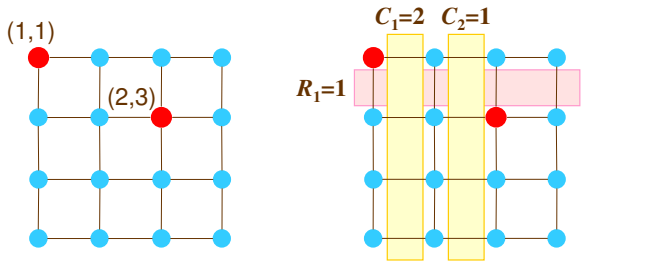
Learning Steps
(BL-FNSOM1) An input vector $\boldsymbol{x}_j$ is inputted to all the neurons at the same time in parallel.
(BL-FNSOM2) Distances between $\boldsymbol{x}_j$ and all the weight vectors are calculated, and winner c_j is chosen according to Eq. (1).
(BL-FNSOM3) The winning frequency of winner c_j is increased by

$$\gamma_{c_j}{}^{\text{new}} = \gamma_{c_j}{}^{\text{old}} + 1. \tag{5}$$

(BL-FNSOM4) After repeating the steps (BL-FNSOM1) and (BL-FNSOM3) for all the input data set, all the weight vectors are updated as

$$\boldsymbol{w}_i^{\text{new}} = \frac{\sum_{j=1}^{N} h_{F c_j,i} \boldsymbol{x}_j}{\sum_{j=1}^{N} h_{F c_j,i}}, \tag{6}$$

where $h_{F c_j,i}$ is the neighborhood function of FN-SOM:



(a) (b)

Fig. 2 Neighborhood distance between neuron 1 and neuron 7. (a) Conventional SOM method according to Eq. (3). (b) FN-SOM method according to Eq. (8).

$$h_{Fc_j,i} = \exp\left(-\frac{d_F(i,c_j)}{2\sigma^2(t)}\right). \tag{7}$$

$d_F(i,c_j)$ is the neighboring distances between the winner c_j and the other neurons calculated with considering the false-neighbor degrees. 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;

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

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 $\sum_{k=k_1}^{k_2-1} C_k$ means the sum of the false-neighbor degrees between the column k_1 and k_2 (as Fig. 2).

(BL-FNSOM5) 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-SOM6) to (FN-SOM9). If not, we perform step (FN-SOM10). In other words, we consider the false-neighbors every time when the learning steps are performed for λ input data.

Considering False-Neighbors

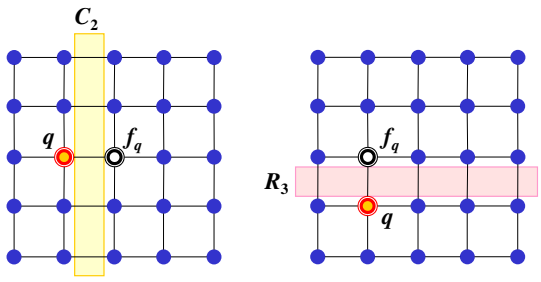
(BL-FNSOM6) We find a set of neurons S which have never become the winner:

$$S = \{i \mid \gamma_i = 0\}. \tag{9}$$

If the neuron, which have never become the winner, does not exist, namely $S = \emptyset$, we return to (FN-SOM1) without considering the false-neighbors.

(BL-FNSOM7) A false-neighbor f_q of each neuron q in S is chosen from the set of direct topological neighbors of q denoted as N_{q_1}. f_q is the neuron whose weight vector is most distant from q;

$$f_q = \arg\max_i \{\|\boldsymbol{w}_i - \boldsymbol{w}_q\|\}, \quad q \in S,\ i \in N_{q_1}. \tag{10}$$



(a) (b)

Fig. 3 Increment of the false-neighbor degree. (a) q and its false-neighbor f_q 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. (11). (b) q and f_q are in the 2nd column and in the 4th and 3rd row, respectively. The false-neighbor degree R_3 between rows 3 and 4 is increased by Eq. (12).

(BL-FNSOM8) A false-neighbor degree between each q and its false-neighbor f_q, R_r or C_k, is increased. If q and f_q 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

$${C_k}^{\text{new}} = {C_k}^{\text{old}} + \frac{n+m}{2nm}. \tag{11}$$

In the same way, if q and f_q 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

$${R_r}^{\text{new}} = {R_r}^{\text{old}} + \frac{n+m}{2nm}. \tag{12}$$

These amounts of increasing the false-neighbor degree are derived by the number of neurons numerically and are fixed.

(BL-FNSOM9) The winning frequency of all the neurons are reset to zero: $\gamma_i = 0$.

(BL-FNSOM10) The steps from (FN-SOM1) to (FN-SOM9) are repeated for all the input data.

4. Experimental Results

We apply BL-FNSOM to various input data and compare BL-FNSOM with the conventional BL-SOM.

4.1 For 2-dimensional data

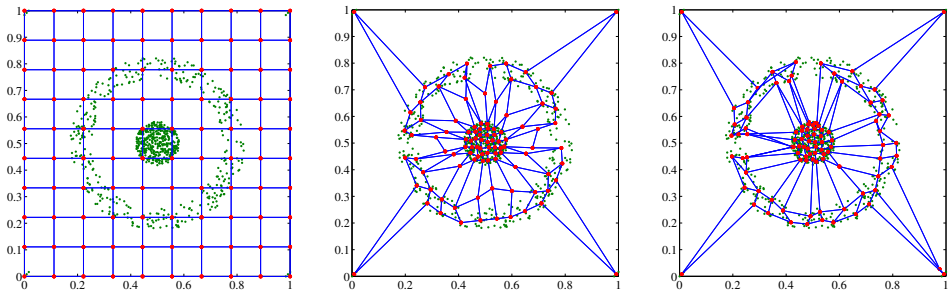
First, we consider 2-dimensional input data as shown in Fig. 4(a). The input data is Target data set, which has a clustering problem of outliers [7]. The total number of the input data N is 770, and the input data has six clusters which include 4 outliers.

Both BL-SOM and BL-FNSOM have $nm = 100$ neurons (10×10). We repeat the learning 20 times for all input data, namely $t_{\max} = 15400$. The parameters of the learning are chosen as follows;

(For SOM) $\sigma_0 = 4$

(For FN-SOM) $\sigma_0 = 4,\ \lambda = 3000,$

where we use the same σ_0 to BL-SOM and BL-FNSOM for



(a) (b) (c)

Fig. 4 Learning results for Target data. (a) Input data and initial state. (b) Conventional BL-SOM. (c) BL-FNSOM.

the comparison and the confirmation of the false-neighbor degree effect.

The learning result of the conventional BL-SOM is shown in Fig. 4(b). We can see that there are some inactive neurons between clusters. The other side, the result of BL-FNSOM is shown in Fig. 4(c). We can see from this figure that there are just a few inactive neurons between clusters, and BL-FNSOM can obtain the more effective map reflecting the distribution state of input data than BL-SOM.

Furthermore, we use the following three measurements to evaluate the training performance of the two algorithms.

Quantization Error Qe**:** This measures the average distance between each input vector and its winner [2];

$$Qe = \frac{1}{N}\sum_{j=1}^{N} \|\boldsymbol{x}_j - \bar{\boldsymbol{w}}_j\|, \tag{13}$$

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

Topographic Error Te**:** This describes how well the SOM preserves the topology of the studied data set [8];

$$Te = \frac{1}{N}\sum_{j=1}^{N} u(\boldsymbol{x}_j), \tag{14}$$

where N is the total number of input data, $u(\boldsymbol{x}_j)$ is 1 if the winner and 2nd winner of $\boldsymbol{x}_j$ are NOT 1-neighbors each other, otherwise $u(\boldsymbol{x}_j)$ is 0. The small value Te is more desirable. Unlike the quantization error, it considers the structure of the map. For a strangely twisted map, the topographic error is big even if the quantization error is small.

Neuron Utilization U**:** This measures the percentage of neurons that are the winner of one or more input vector in the map [9];

$$U = \frac{1}{nm}\sum_{i=1}^{nm} u_i, \tag{15}$$

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.

Table 1 Quantization error Qe, Topographic error Te and Neuron utilization U for Target data.

	Qe	Te	U
BL-SOM	0.0144	0.2156	0.82
BL-FNSOM	0.0131	0.1922	0.98
Improvement rate	9.02%	10.85%	19.51%

The calculated three measurements are shown in Table 1. The quantization error Qe of BL-FNSOM is smaller value than BL-SOM, and by using BL-FNSOM, the quantization error Qe has improved 9.02% from using the conventional BL-SOM. This is because the result of BL-FNSOM has few 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 BL-FNSOM is larger value than BL-SOM. It means that 98% neurons of BL-FNSOM are the winner of one or more input data, namely, there are few inactive neurons. On the other hand, the topographic error Te of BL-FNSOM is smaller value although Qe and U are better values. It means that BL-FNSOM self-organizes most effectively with maintenance of top quality topology.

4. 2 For 3-dimensional data

Next, we consider 3-dimensional input data.

4. 2. 1 Atom data

We consider Atom data set shown in Fig. 5(a), which has clustering problems of linear not separable, different densities and variances. The total number of the input data N is 800, and the input data has two clusters.

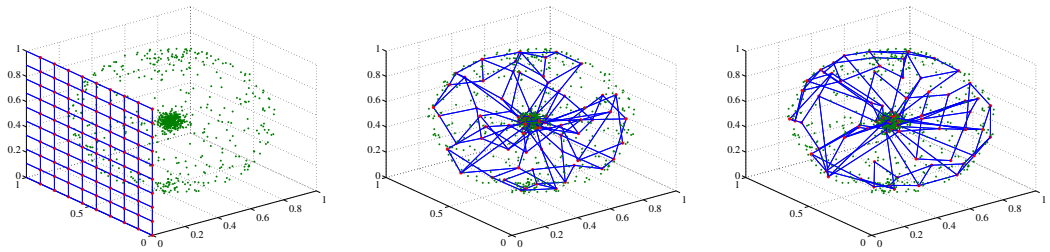
We repeat the learning 19 times for all input data, namely $t_{\max} = 15200$. The learning conditions are the same used in Fig. 4 except $\sigma_0 = 5$.

The learning results of the conventional BL-SOM and BL-FNSOM are shown in Figs. 5(b) and (c). We can see that BL-FNSOM can self-organize edge data more effective than BL-SOM.

The calculated three measurements are shown in Table. 2. In all measurements, FN-BLSOM can obtain better value than BL-SOM.

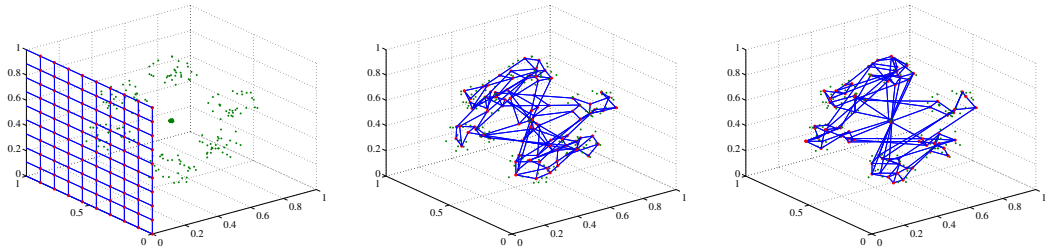
4. 2. 2 Hepta data

We consider Hepta data set shown in Fig. 6(a), which has a clustering problem of different densities in clusters. The total number of the input data N is 212, and the input data



(a) (b) (c)

Fig. 5 Learning results for Atom data. (a) Input data and initial state. (b) Conventional BL-SOM. (c) BL-FNSOM.



(a) (b) (c)

Fig. 6 Learning results for Hepta data. (a) Input data and initial state. (b) Conventional BL-SOM. (c) BL-FNSOM.

Table 2 Quantization error Qe, Topographic error Te and Neuron utilization U for Atom data.

	Qe	Te	U
BL-SOM	0.0507	0.2975	0.84
BL-FNSOM	0.0461	0.2713	0.93
Improvement rate	9.07%	8.81%	10.71%

Table 3 Quantization error Qe, Topographic error Te and Neuron utilization U for Hepta data.

	Qe	Te	U
BL-SOM	0.0264	0.3208	0.72
BL-FNSOM	0.0211	0.2689	0.97
Improvement rate	20.08%	16.18%	34.72%

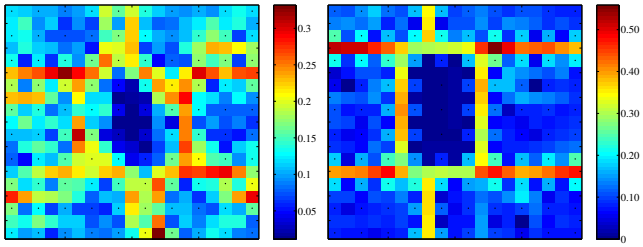
has seven clusters.

We repeat the learning 70 times for all the input data, namely $t_{\max} = 14840$. The learning conditions are the same used in Fig. 5.

Figures 6(b) and (c) show the learning results of BL-SOM and BL-FNSOM, respectively. We can see that BL-FNSOM has fewer inactive neurons than BL-SOM between clusters.

Table 3 shows the calculated three measurements. All measurements are improved significantly from using BL-SOM. Because the inactive neurons have been reduced by 34.72%, more neurons are attracted to clusters and Qe has been decreased by about 20%. It means that BL-FNSOM can extract the feature of the input data more effectively.

In order to investigate extracted cluster structure using each algorithm, we calculate U-matrix [10] of the learning results. U-matrix can visualize distance relationships in a high dimensional data space. Figure 7 shows the U-matrix from Figs. 6(b) and (c). From these figures, we can see that cluster boundaries of BL-FNSOM are clearer than BL-SOM. It means that more neurons of BL-FNSOM self-organize the input data of cluster part In other words, we can obtain more detail on the cluster input data in the high dimensional data.



(a) (b)

Fig. 7 U-matrix of learning results for Hepta data. (a) Conventional BL-SOM. (b) BL-FNSOM.

4.3 Real world data

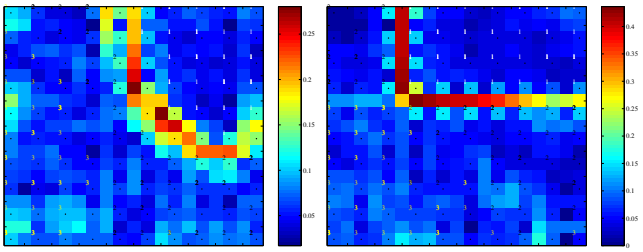
Furthermore, we apply BL-FNSOM to the real world clustering problem. We use the Iris plant data [11] as real data. This data is one of the best known databases to be found in the pattern recognition literature [12]. The data set contains three clusters of 50 instances respectively, where each class refers 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. 5.

Table 4 shows the calculated three measurements. We can confirm that all measurement are improved significantly from

Table 4 Quantization error Qe, Topographic error Te and Neuron utilization U for Iris data.

	Qe	Te	U
BL-SOM	0.0150	0.3733	0.82
BL-FNSOM	0.0137	0.2867	0.91
Improvement rate	8.67%	23.20%	10.98%



(a) (b)

Fig. 8 U-matrix of learning results for Iris data. Label 1, 2 and 3 correspond to *Iris setosa, Iris versicolor* and *Iris virginica*, respectively. (a) Conventional BL-SOM. (b) BL-FNSOM.

using BL-SOM. We can say that BL-FNSOM has fewer inactive neurons than BL-SOM and can self-organize more effective with keeping good map topographic.

The U-matrix calculated from each learning result are shown in Fig. 8. The boundary line of BL-FNSOM between *Iris setosa* and the other two is clearer than the conventional BL-SOM. Meanwhile, it is hard to recognize the boundary line between *Iris versicolor* and *Iris virginica* on both results. However, on the BL-FNSOM result, we can faintly observe the boundary line. Because BL-FNSOM has few inactive neurons, more neurons can self-organize the cluster input data, and BL-FNSOM can extract slight different between *Iris versicolor* and *Iris virginica*.

5. Conclusions

In this study, we have proposed the Batch-Learning Self-Organizing Map with False-Neighbor degree between neurons (BL-FNSOM). We have applied BL-FNSOM to 2-dimensional data, 3-dimensional data and the real world data. Learning performance has evaluated both visually and quantitatively and has compared with BL-SOM. We have confirmed that BL-FNSOM has fewer inactive neurons and can obtain the more effective map reflecting the distribution state of input data than BL-SOM.

Acknowledgment

This work was partly supported by Yamaha Music Foundation.

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*, 2nd ed., 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] H. Matsushita and Y. Nishio, "Self-Organizing Map with False Neighbor Degree between Neurons for Effective Self-Organization," *Proc. of Workshop on Self-Organizing Maps*, pp. WeP-P-13, 2007.

[6] I. T. Jolliffe, *Principal Component Analysis*, New York, Springer, 1986.

[7] A. Ultsch, "Clustering with SOM: U*C", *Proc. Workshop on Self-Organizing Maps*, pp. 75–82, 2005.

[8] K. Kiviluoto, "Topology Preservation in Self-Organizing Maps", *Proc. of International Conference on Neural Networks*, pp. 294–299, 1996.

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

[10] A. Ultsch and H.P.Siemon, "Kohonen's Self Organizing Feature Maps for Exploratory Data Analysis," *Proc. International Neural Network Conference*, pp. 305–308, 1990.

[11] 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].

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