IWVCC'08
@ Xian
Nov. 1-3, 2008

Growing Grid with False-Neighbor Degree

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

Abstract

We apply the self-organization considering false-neighbor degree to the fine-tuning of Growing Grid after the growth process is finished. This study proposes a Growing Grid with False-Neighbor degree (FN-GG). We confirm that FN-GG can self-organize the input data, which is in a rectangular input space, most effectively.

1. Introduction

It is important to investigate various clustering methods [1], since we can accumulate a huge amount of data in recent years. The Self-Organizing Map (SOM) is an unsupervised neural network [2] and has attracted attention for its clustering properties. SOM can obtains statistical features of input data, so, it is simplified model of the self-organization process of the brain.

However, the topology of the conventional SOM has to be fixed in advance. It is difficult to understand statistical feature of the high-dimensional input data and choose an appropriate topology in advance. Therefore, Growing Grid network [3], which one kinds of SOM, was proposed. The network structure is a rectangular grid which increases its size during learning. By inserting complete rows or columns of neurons, the network can adapt its height/width ratio to the distribution of the input data automatically.

In our past study, we have proposed a new SOM algorithm, SOM with False-Neighbor degree between neurons (called FN-SOM) [5]. False-Neighbor degrees are allocated between adjacent rows and adjacent columns of FN-SOM. They are increased with learning and act as a burden of the distance between map nodes when the weight vectors of neurons are updated.

In this study, we combine the concept of Growing Grid and FN-SOM and propose a Growing Grid with False-Neighbor degree (called FN-GG). We apply the self-organization considering false-neighbor degree to the fine-tuning of Growing Grid after the growth process is finished.

We explain the learning algorithm of FN-GG in detail in Sect. 2. The learning behaviors of FN-GG for 2-dimensional input data are investigated with applying to the clustering in Sect. 3.1. In addition, we apply FN-GG to the feature extraction problem in Sect. 3.2. Learning performance is evaluated both visually and quantitatively using three measurements in comparison with the conventional SOM, Growing Grid and FN-SOM. From these results, we confirm that the results of FN-GG have the fewest inactive neurons and its map topology is the most appropriate for the input data which is in a rectangular input space.

2. Growing Grid with False-Neighbor Degree

We explain the learning algorithm of the Growing Grid with False-Neighbor Degree (FN-GG). The network of FN-GG consists of nm neurons located at a rectangular $n \times m$ grid. Each neuron has a d-dimensional weight vector $\boldsymbol{w}_i$ $\boldsymbol{w}_i = (w_{i1}, w_{i2}, \cdots, w_{id})$ $(i = 1, 2, \cdots, nm)$ as the conventional SOM. The initial values of all the weight vectors are given over the input space at random. A winning frequency γ_i is associated with each neuron and is set to zero initially. 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.

2.1. Learning

(FN-GG1) An input vector $\boldsymbol{x}_j$ is inputted to all the neurons at the same time in parallel.
(FN-GG2) Distances between $\boldsymbol{x}_j$ and all the weight vectors are calculated. A 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, in this study, the Euclidean distance is used.
(FN-GG3) Increment of the winning frequency of winner c by ${\gamma_c}^{\mathrm{new}} = {\gamma_c}^{\mathrm{old}} + 1$.
(FN-GG4) The weight vectors of the neurons are updated according to

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

where $h_{G_{c,i}}(t)$ is the neighborhood function of FN-GG;

$$h_{G_{c,i}}(t) = \alpha_0 \exp\left(-\frac{{d_g}^2(c,i)}{2{\sigma_0}^2}\right), \tag{3}$$

where α_0 is a constant learning rate, and σ_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) as

$$d_g(s_1, s_2) = |r_1 - r_2| + |k_1 - k_2|, \tag{4}$$

where s_1 is located at r_1-th row and k_1-th column, and s_2 is located at r_2-th row and k_2-th column.
(FN-GG5) If $\sum_{i=1}^{nm} \gamma_i \geq \lambda_g$ is satisfied, we insert new rows and columns according to steps from (FN-GG6) to (FN-GG9). If not, we return to (FN-GG1).

2.2. Insertion of new rows and columns

(FN-GG6) 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;

$$q = \arg\max_i \{\gamma_i\}. \tag{5}$$

We find the neuron f which is with the most different weight vector in direct topological neighbors of q denoted as N_{q_1}.
(FN-GG7) We insert a new row (or column) between q and f. Without loss of generality, we assume that q and f are in the r-th row and k-th and $(k+1)$-th columns. We insert a new column k' with n neurons between columns k and $k+1$. The weight vectors of the new neurons are interpolated from their neighbors which does increase the density of weight vectors in the vicinity of $\boldsymbol{w}_q$.

$$\boldsymbol{w}_{rk'} = 0.5\left(\boldsymbol{w}_{rk} + \boldsymbol{w}_{r(k+1)}\right), \quad 1 \leq r \leq n. \tag{6}$$

(FN-GG8) The number n of rows (or m of columns) are increased; $n = n + 1$, then all the winning frequencies are reset: $\gamma_i = 0$.
(FN-GG9) If $nm \geq nm_{\max}$ is fulfilled, we stop growth process and perform Fine-tuning of the weight vectors according to steps from (FN-GG10) to (FN-GG15). If not, we continue with the next round of learning, i.e., return to (FN-GG1).

2.3. Fine-tuning considering false-neighbors

After the growth process is finished, the network has false-neighbor degrees. The false-neighbor degrees of rows R_r $(1 \leq r \leq n-1)$ and of columns C_k $(1 \leq k \leq m-1)$ are allocated between adjacent rows and columns of FN-GG, respectively. The initial values of R_r and C_r are set to zero.

(FN-GG10) We fine-tune the weight vectors using a decreasing learning rate with considering the false-neighbors. We perform $t'_{\max} = n \times m \times \lambda_f$ steps according to steps from (FN-GG1) to (FN-GG4). λ_f denotes how many fine-tuning steps per neurons are performed.

In Eq. (2), we use following neighborhood function $h_{Fc,i}(t')$ in place of $h_{Gc,i}(t)$

$$h_{Fc,i}(t') = \alpha(t') \exp\left(-\frac{{d_f}^2(c,i)}{2\sigma_0^2}\right). \tag{7}$$

where $\alpha(t')$ is time-dependent learning rate

$$\alpha(t') = \alpha_0(\alpha_1/\alpha_0)^{t'/t'_{\max}}, \tag{8}$$

where t' denotes the learning step in the fine-tuning phase which starts after the growth phase is finished.

$d_g(c,i)$ is calculated considering 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;

$$\begin{aligned} d_f(s_1, s_2) = &\left(|r_1 - r_2| + \sum_{r=r_1}^{r_2-1} R_r\right) \\ &+ \left(|k_1 - k_2| + \sum_{k=k_1}^{k_2-1} C_k\right), \end{aligned} \tag{9}$$

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.
(FN-GG11) 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-GG12) to (FN-GG15). If not, we return to (FN-GG10).

(FN-GG12) We find a set of neurons S which have never become the winner:

$$S = \{i \mid \gamma_i = 0\}. \tag{10}$$

If the neurons, which have never become the winner, do not exist, namely $S = \emptyset$, we return to (FN-GG10) without considering the false-neighbors.
(FN-GG13) A false-neighbor f_q of each neuron q in S is chosen from the set of N_{q_1}. f_q is the neuron whose weight vector is most distant from q;

$$f_q = \arg\max_i \{\|\boldsymbol{w}_i - \boldsymbol{w}_q\|\}, \;\; q \in S, \; i \in N_{q_1}. \tag{11}$$

(FN-GG14) 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, the false-neighbor degree C_k between columns k and $k+1$ is increased according to

$${C_k}^{\text{new}} = {C_k}^{\text{old}} + \frac{1}{n}. \tag{12}$$

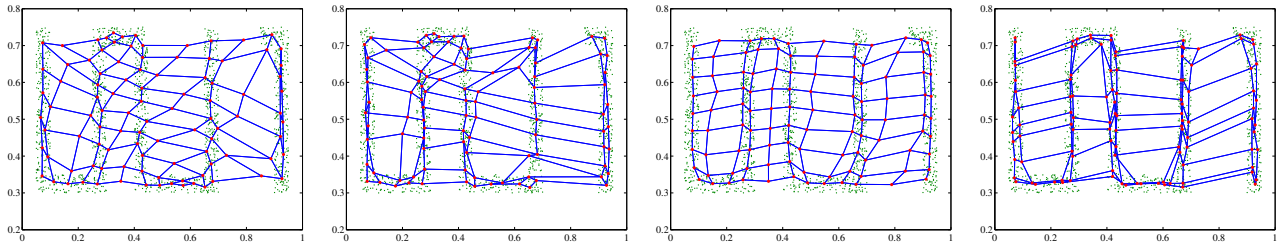
In the same way, if q and f_q are in the k-th column and in the $(r+1)$-th and r-th row, the false-neighbor degree R_r between rows r and $r+1$ is also increased according to

$${R_r}^{\text{new}} = {R_r}^{\text{old}} + \frac{1}{m}. \tag{13}$$

(FN-GG15) The winning frequency of all the neurons are reset to zero: $\gamma_i = 0$.

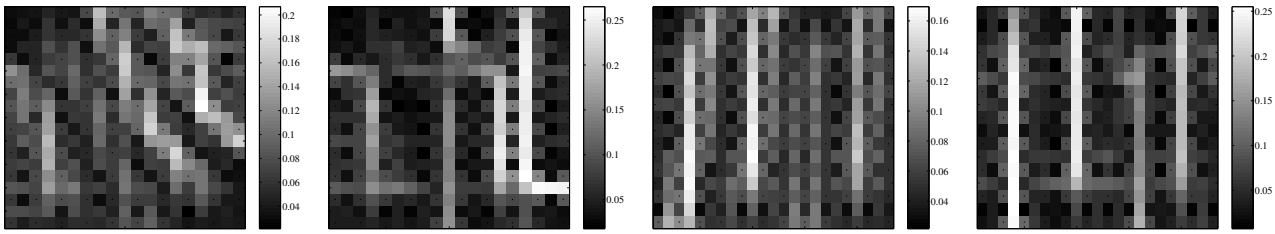
3. Experimental Results

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



(a) (b) (c) (d)

Figure 1: Learning results for 2-dimensional data. (a) Conventional SOM. (b) FN-SOM. (c) Growing Grid. (d) FN-GG. Map size of (c) and (d) after learning are 9×12.



(a) (b) (c) (d)

Figure 2: U-matrix of each learning result. (a) Conventional SOM. (b) FN-SOM. (c) Growing Grid. (d) FN-GG.

3.1. Application to Clustering

First, we consider 2-dimensional landscape input data. Total number of the input data N is 1500 and all the input data are sorted at random.

Both the conventional SOM and FN-SOM has $nm = 100$ neurons (10×10). Growing Grid and FN-GG starts learning with 2×2 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 21 times for all input data. The parameters of the learning are chosen as follows; $\alpha_0 = 0.5$, $\sigma_0 = 0.9$, $\alpha_1 = 0.006$, $\lambda_g = 30$, $\lambda_f = 100$, $\lambda = 15$.

Respective learning results of four algorithms are shown in Fig. 1, and Fig. 2 shows its U-matrix [4] which represent the learned maps in output space. Each gray value corresponds to the distance between the neurons. We consider the conventional SOM and FN-SOM which have predefined structure. The map of SOM is not matching the network topology and has a lot of inactive neurons between clusters as Fig. 1(a). Therefore, Fig. 2(a) can not reflect the distribution state of the input data correctly. The result of FN-SOM has few inactive neurons as Fig. 1(b) by effects of the false-neighbor degree, however, it is a distorted map for the input data as the conventional SOM. On the other hand, the result of Growing Grid is able to automatically choose an appropriate row/column ratio during the growth process shown in Fig. 1(c). However, it is hard to understand boundaries between clusters from Fig. 2(c) because there are a lot of inactive neurons in this result. The result of FN-GG (see Fig. 1(d)) can obtain the statistical features of the input data and fits the given data. Consequently, we can understand the structure of the input data from Fig. 2(d).

Furthermore, in order to evaluate the learning performance of FN-GG in comparison with other three algorithms, we use the following three measurements.

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

$$Qe = \frac{1}{N} \sum_{j=1}^{N} \|\boldsymbol{x}_j - \bar{\boldsymbol{w}}_j\|, \tag{14}$$

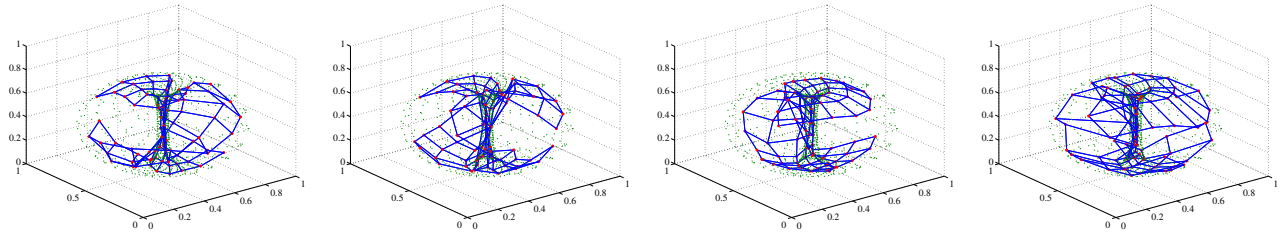
where $\bar{\boldsymbol{w}}_j$ is the weight vector of the corresponding winner of the input vector $\boldsymbol{x}_j$. Therefore, the small value Qe is more desirable.

Topographic Error *Te***:** This describes how well the SOM preserves the topology of the studied data set [7];

$$Te = \frac{1}{N} \sum_{j=1}^{N} u(\boldsymbol{x}_j), \tag{15}$$

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



(a) (b) (c) (d)

Figure 3: Learning results for Langford data. (a) Conventional SOM. (b) FN-SOM. (c) Growing Grid. (d) FN-GG. Map size of (c) and (d) after learning are 8×13.

Table 1: Quantization error Qe, Topographic error Te and Neuron utilization U for 2-dimensional data.

	SOM	Growing Grid	FN-SOM	FN-GG
Qe	0.0177	0.0188	**0.0158**	**0.0158**
Te	0.1653	**0.0647**	0.1987	0.1787
U	0.7800	0.7407	0.9200	**0.9907**

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

	SOM	Growing Grid	FN-SOM	FN-GG
Qe	0.0523	0.0581	0.0507	**0.0495**
Te	0.227	0.157	0.298	**0.142**
U	0.9100	0.9327	0.9600	**0.9711**

neurons that are the winner of one or more input vector in the map [6];

$$U = \frac{1}{nm} \sum_{i=1}^{nm} u_i, \tag{16}$$

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 and larger value of U means fewer inactive neurons.

Table 1 shows the calculated three measurements. Qe and U of FN-GG are the best values. Te of Growing Grid is the smallest, but we can confirm that FN-GG has improved from FN-SOM.

3.2. Application to Feature Extraction

Next, we apply FN-GG to the feature extraction for the data generated by Langford equation [8]. The data is onto 2-torus and has the cavity. The total number of data N is 1000.

The learning results are shown in Fig. 3. We can confirm that FN-GG can self-organize up to the edge of the input data in all the methods. Furthermore, the calculated measurements are shown in Table 2. For all measurements, FN-GG obtains the best values. These results mean that in the learned map of FN-GG, there are few errors between the input data and the neurons as FN-SOM, and FN-GG self-organizes most effectively with maintenance of top quality topology.

4. Conclusions

This study has proposed the Growing Grid with False-Neighbor degree (FN-GG). This method applied the false-neighbor degree to the Growing Grid network. We have applied FN-GG to clustering and feature extraction and have confirmed its efficiency.

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

[5] H. Matsushita and Y. Nishio, "Self-Organizing Map with False-Neighbor Degree between Neurons for Effective Self-Organization," *IEICE Transactions on Fundamentals*, vol. E91-A, no. 6, pp. 1463–1469, Jun. 2008.

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

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

[8] W. F. Langford, "Numerical Studies of torus bifurcations," *International Series of Numerical Mathematics*, vol.70, pp.285-295, 1984.