2017 International Symposium on Nonlinear Theory and Its Applications,
NOLTA2017, Cancun, Mexico, December 4-7, 2017

Continuous Learning of the SOM with an Adaptive Neighborhood Function

Hikari Yoshimi†, Hidetaka Ito†, and Hiroomi Hikawa†

†Faculty of Engineering Science, Kansai University
3-3-35 Yamate-cho, Suita, Osaka 564-8680, Japan
Email: k183811@kansai-u.ac.jp, h.ito@kansai-u.ac.jp, hikawa@kansai-u.ac.jp

Abstract—This paper proposes a new neighborhood function for the self-organizing map (SOM). As the learning of the SOM progresses, the conventional neighborhood function shrinks its magnitude and neighborhood radius, and the learning stops after pre-defined training iterations. On the other hand, the proposed neighborhood function uses only the distance between the weight vector of the winner neuron and the input vector, then the magnitude and radius are computed according to this distance. Since the proposed neighborhood function is not a function of the learning iterations, it allows the SOM to continue its learning without time constraints. This feature is especially effective under the changing input vector space that arises in, e.g., online learning.

1. Introduction

The self-organizing map (SOM) [1] rooted in artificial neural networks is a type of nonlinear principal component analysis that forms a topologically ordered mapping from the high-dimensional data space to a low-dimensional representation space. The SOM has been used in wide variety of applications, such as visualization, analysis, interpretation, and classification of large high-dimensional data set [2]-[6].

In the SOM algorithm, a neighborhood function plays a very important role in its learning, and many researchers proposed modified neighborhood functions.

One of the major problems of the SOM is a kind of topological defect such as a twist created in the map during training, and the elimination of the defect requires much more learning-steps. Aoki et.al [7] proposed an asymmetric neighborhood function for the SOM algorithm to overcome the problem, and it was found that the proposed function accelerates the ordering process of the SOM.

Conventional neighborhood function reduces its magnitude and neighborhood size as the SOM's learning progresses. A certain fixed number of learning iterations is defined beforehand as the stopping criterion for conventional neighborhood function, and its magnitude and neighborhood size are reduced so that they take minimum values at the end of training step. Consequently, after the predefined training step, no training is carried out, and the input vector space is not allowed to change during the learning. During the learning process, since the function characteristics (magnitude, neighborhood size) changes, the SOM with the conventional neighborhood function cannot appropriately cope with the changing input vector space. Such situation can happen in on-line training.

Hikawa et.al [8] proposed a new neighborhood function that additionally used the distance between the weight vector of the winner neuron and the input vector. Using the proposed function, a hardware SOM was implemented on field programmable gate array (FPGAA), and it was revealed that the new function improved the learning performance of the hardware SOM. However, this neighborhood function was still controlled by the learning steps, and therefore it stopped its learning after the predefined training step.

This paper proposes a new neighborhood function that uses only the distance between the weight vector of the winner neuron and the input vector, so that the function value is computed adaptively according to the distance. Since the iteration count is not used to compute the neighborhood function value, the proposed SOM performs learning continuously without stopping.

2. SOM with Proposed Adaptive Neighborhood function

The SOM employs an unsupervised learning algorithm to form a nonlinear mapping from a given high-dimensional input space to a lower-dimensional map of neurons.

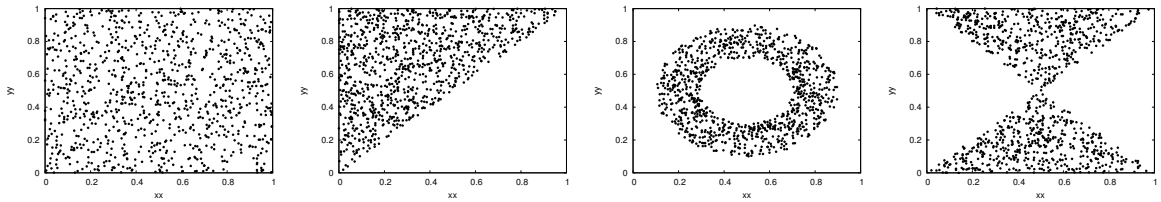
The neurons in the SOM are placed in a lattice on 2D, and an n-dimensional vector, $\vec{m}_i$ called the weight vector, is assigned to every neuron.

$$\vec{m}_i = \{\mu_{i1}, \mu_{i2}, \cdots, \mu_{in}\} \in \Re^n \tag{1}$$

The operation of the SOM can be divided into two modes, i.e. learning and recall modes. In the initial learning mode, the map is trained with a set of input vectors. After the learning mode, the weights of the map are kept unchanged and the map is used in the recall mode. The learning phase starts with an appropriate initialization of the weight vectors. Subsequently, input vectors, $\vec{x} \in \Re^n$, are presented to the map in multiple iterations.

$$\vec{x} = \{\xi_{i1}, \xi_{i2}, \cdots, \xi_{in}\} \in \Re^n \tag{2}$$

For each input vector, distances to all weight vectors are calculated and the neuron-c with the smallest distance is



(A) (B) (C) (D)

Figure 1: Training vector sets, (A) Random, (B) Triangle, (C) Donut, (D) Hourglass.

determined.

$$c = \arg\min_i \{\|\vec{x} - \vec{m}_i\|\} \tag{3}$$

Usually the Euclidean distance is used to find the winner neuron.

$$\|\vec{x} - \vec{m}\| = \sqrt{(\xi_1 - \mu_1)^2 + (\xi_2 - \mu_2)^2 + \cdots (\xi_n - \mu_n)^2} \tag{4}$$

After the winner neuron is determined, this neuron and weight vectors in its neighborhood are updated so that they are closer to the input vector.

$$\vec{m}_i(t+1) = \vec{m}_i(t) + h_{ci}\{\vec{x}(t) - \vec{m}_i(t)\} \tag{5}$$

where, t is a sample number.

The neighborhood function h_{ci} is defined as

$$h_{ci} = \alpha \exp\left(-\frac{\|\vec{r_c} - \vec{r_i}\|}{2\sigma^2}\right) \tag{6}$$

where, $\vec{r_c} \in \Re^2$ and $\vec{r_i} \in \Re^2$ are the position vectors of the winner neuron-c and neuron-i. α and σ are learning coefficient and radius of the neighborhood.

A very important feature of the mapping is its topology-preserving nature, i.e., two vectors that are neighbors in the input space will be represented close to each other on the map, too. This feature is realized by the weight update by using the neighborhood function.

In order for the SOM to do the learning successfully, the parameters α and σ should be controlled appropriately. In the conventional neighborhood function, they are controlled by the learning iteration count t. Starting with some initial values they are decreased toward their minimum values until t reaches a predefined iteration number T. This monotonic decrease strategy is employed assuming that the input vector space does not change during the learning. Obviously after the predefined training iterations, no learning is carried out. Any change in the input vector space during the learning degrades the learning performance because α and σ have been made already smaller, with which sufficient weight update can not be done. In the case of online training, the change of the input vector space can happen.

To cope with such situation, the parameters α and σ are modified as

$$\alpha = \eta_a \cdot d_c / \overline{d_c} \tag{7}$$

$$\sigma = \eta_s \cdot d_c / \overline{d_c} \tag{8}$$

where, η_a and η_s are adjustment coefficients, d_c is the vector distance of winner neuron, and $\overline{d_c}$ is the exponential average of d_c obtained by the recursive formula

$$\overline{d_c}(t+1) = k \cdot \overline{d_c}(t) + (1-k) \cdot d_c(t) \tag{9}$$

$$d_c = \|\vec{x} - \vec{m_c}\| \tag{10}$$

where, k is an averaging coefficient, and c is the winner neuron's index. Strong weight update is necessary for proper learning when the vector distance is large. With this modification α and σ are made larger in proportion to the vector distance of the winner neuron to reinforce the learning.

3. Experiments

To evaluate the proposed method, SOM with the proposed neighborhood function was trained with artificial data. Figure 1 shows four training data-sets, each of which consists of 1500 vectors.

The first experiment examines the response of the proposed SOM against the change of input vector space. In this test, four data sets are sequentially fed to the SOM. Each data set is applied to the SOM for 100 epochs. Since each data set consists of 1500 vectors, one *epoch* consists of 1500 iteration steps. Four data sets were applied to the SOM, and each data set was applied for 100 epochs. Therefore, during the training, $T = 6000$ iterations were carried out in total.

For the proposed neighborhood function, α and σ are computed with $\eta_a = 0.3$ and $\eta_s = 0.5$. The exponential average $\overline{d_c}$ was computed with $k = 0.7$. These parameters were determined by preliminary experiments.

Fig. 2 shows the transition of weight vectors during the learning. Initial weight vectors of the SOM are random small vectors, and the SOM successfully re-organized its weight vectors to represent the different input vector space.

For comparison, the learning performance of the conventional SOM is examined using the same training data. Here, α and σ are set according to the following equation.

$$\alpha = 0.6 \cdot (1.0 - t/T) \tag{11}$$

$$\sigma = 0.6 \cdot (1.0 - t/T) \tag{12}$$

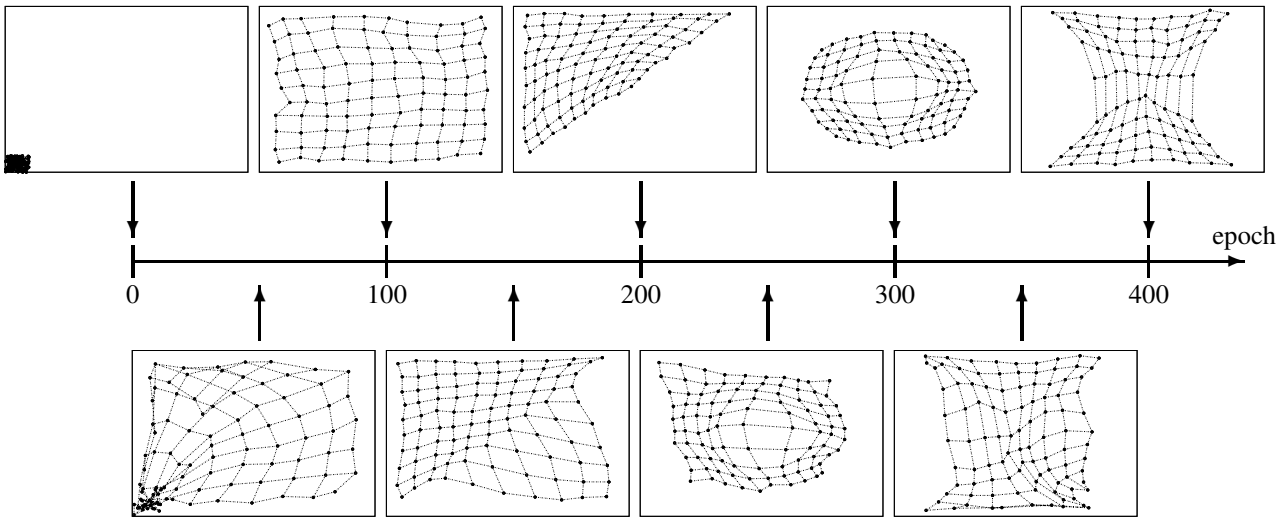


Figure 2: Transition of the weight vectors of the proposed SOM during the learning.

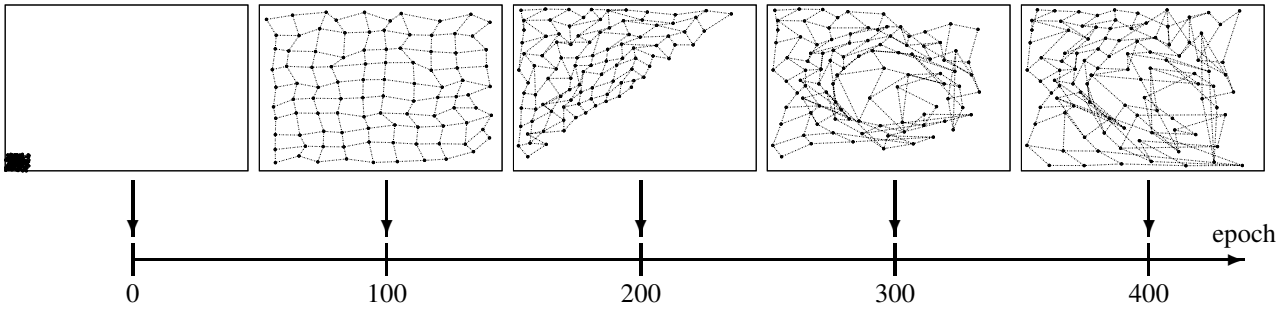


Figure 3: Transition of the weight vectors of the conventional SOM during the learning.

where, $T = 60000$ and $t = 0 \sim 5999$. The results for the conventional SOM are shown in Fig. 3. For the first training data (Random), the weight vectors were properly arranged and the SOM succeeded in the training, but it failed to learn the remaining three input data spaces. When the input switched to new data sets ($epoch = 100, 200$, and 300), α and σ had become smaller as expressed by equations (11) and (12). The small α and σ made it difficult for the SOM to re-organize its weight vectors to represent the input vector space.

For the training, weight vectors of the SOM are usually very small vectors. In the next test, large random vectors were assigned to the weight vectors as the initial values, and the SOM's learning behaviors were examined. In this experiment, each of the four data sets was applied to the SOM individually. Therefore, the conventional SOM was trained by using $T = 15000$ and $t = 0 \sim 14999$ in equations (11) and (12).

Fig. 4 shows the initial vectors consisting of large random vectors. Not only they were large, but also they were twisted heavily. Using these vectors as the initial weight vectors, one of the input data sets shown in Fig. 1 was fed to the proposed SOM and the conventional SOM.

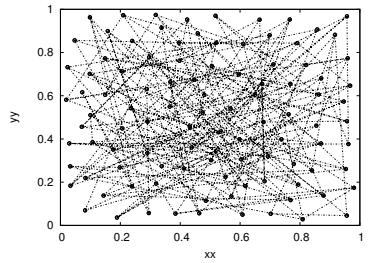
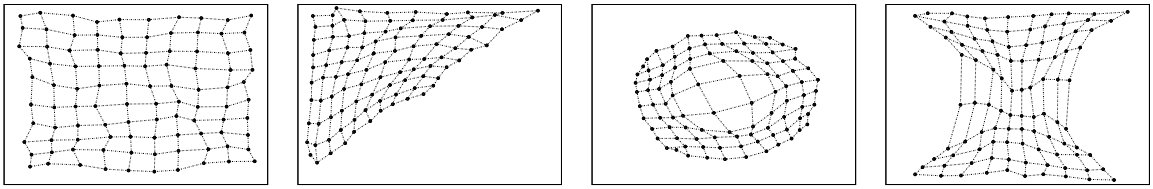


Figure 4: Large random initial state.

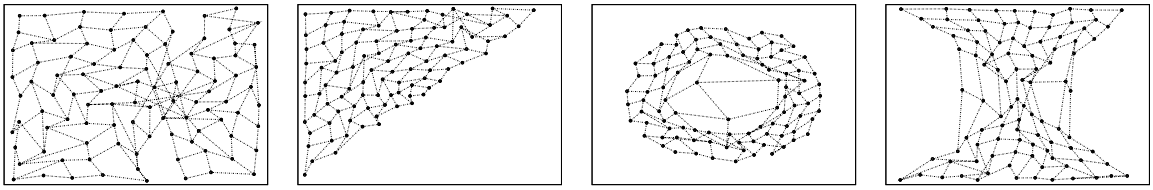
Figs. 5 and 6 show the weight vectors in the proposed SOMs after the learning. Apparently, arrangement of the vectors in Figs 5 is much better than those of the conventional SOM given in Fig. 6. The arrangement of the vectors in Fig. 5 looks smooth, while the vectors in Fig. 6 look ragged and small twists can be seen.

.



(A) (B) (C) (D)

Figure 5: Weight vectors of the proposed SOM after training, (A) Random, (B) Triangle, (C) Donut, (D) Hourglass.



(B) (C) (D) (E)

Figure 6: Weight vectors of the conventional SOM after training, (A) Random, (B) Triangle, (C) Donut, (D) Hourglass.

4. Conclusion

This paper proposed a new neighborhood function that is computed by only the winner neuron's vector distance to the input vector. In the proposed neighborhood function, computation of the learning coefficient α and the neighborhood radius σ are modified so that they only depend on the distance between the weight vector of the winner neuron and the input vector.

The experimental results revealed that the proposed neighborhood function allows the SOM to adaptively learn the training data space that changes in time. It was also found that the SOM with the proposed neighborhood function is robust to the arrangement of initial weight vectors.

References

[1] T. Kohonen, *Self-Organizing Maps*, Springer-Verlag, New York, 2001.

[2] P. Kossakowski, P. Bilski, "Application of self-organizing maps to the stock exchange data analysis," *Proc. of 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS),* 2015.

[3] G. S. Silva, T. C. Mattozo, J. A. F. Costa, A. P. Fernandes Neto, R. L. S. Franca, "Mobile Communications Market Segmentation: An Analysis of Data Combining Self-organizing Maps and Structural Equation Modeling," *IEEE Latin America Transactions,* pp.2390-2397, Vol. 13, Issue 7, 2015.

[4] Z. A. Marasigan, A. Dionisio, G. Solano, "Microarray data clustering and visualization tool using self-organizing maps," *Proc. of 2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA),* 2015.

[5] A. Mohamed, M. S. Hamdi, S. Tahar, "Self-Organizing Map-Based Feature Visualization and Selection for Defect Depth Estimation in Oil and Gas Pipelines," *Proc. of 2015 19th International Conference on Information Visualisation (iV),* 2015.

[6] T. Lamjiak, J. Polvichai, P. Varnakovida, "A geometrical data classification using Self-Organizing map with fixed possible matching units," *Proc. of 2016 International Computer Science and Engineering Conference (ICSEC),* 2016.

[7] T. Aoki, T. Aoyagi, "Self-Organizing Maps with Asymmetric Neighborhood Function," *Neural Computation*, Vol. 19, Issue 9, pp.2515-2535, 2007.

[8] H. Hikawa, Y. Maeda, "Improved Learning Performance of Hardware Self-Organizing Map Using a Novel Neighborhood Function," *IEEE Trans. on Neural Networks and Learning Systems",* Vol. 26, No. 11, pp.2861-2873, Nov. 2015.