2008 International Workshop on Nonlinear Circuits and Signal Processing
NCSP'08, Gold Coast, Australia, March 6-8, 2008

Neural Gas Containing Two Kinds of Neurons and its Behaviors

Keiko Kanda†, Haruna Matsushita†and Yoshifumi Nishio†

†Tokushima University, JAPAN
Email: {kanda, haruna, nishio}@ee.tokushima-u.ac.jp

Abstract - In this study, we propose two new Neural Gas algorithms which contains two kinds of neurons (called TN-NG) and contains two kinds of update functions (called TF-NG). In TN-NG, all neurons learn according to own character at each learning. On the other hand, in TF-NG, all neurons learn according to winner's character at each learning. We confirm that two proposed methods can obtain more effective learning results than the conventional Neural Gas.

1. Introduction

In the real world, living things with different various features are coexistent. For example, an elephant, which has a slow heart rate, and a mouse, which has a rapid heart rate, have different senses of time. However, in the learning of Neural Gas [1], the network contains only one type of neurons.

In this study, we propose two new Neural Gas algorithms. One method is a Neural Gas Containing Two Kinds of Neurons (called TN-NG). The other method is a Neural Gas Containing Two Kinds of Update Functions (called TF-NG). Both TN-NG and TF-NG have two kinds of neurons. In TN-NG, all neurons learn according to own character at each updating. On the other hand, in TF-NG, all neurons learn according to winner's character at each updating.

The learning behaviors of TN-NG and TF-NG are investigated with the computer simulation. In addition, TN-NG is applied to noise reduction and TF-NG is applied to feature extraction, by using each feature of the learning results. We confirm that TN-NG and TF-NG can obtain the more effective learning results than the conventional Neural Gas.

2. Neural Gas

We explain the learning algorithm of the conventional Neural Gas. The set A containing M neurons; $A = \{c_1, c_2, \cdots, c_M\}$. Each neuron c_i $(i = 1, 2, \cdots, M)$ has a d-dimensional weight vector $\boldsymbol{w}_i = (w_{i1}, w_{i2}, \cdots, w_{id})$. The initial values of the weight vectors are given 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.

(NG1) An input vector $\boldsymbol{x}_j$ is inputted to all the neurons at the same time in parallel.
(NG2) Euclid distances d_{ji} between the input vector $\boldsymbol{x}_j$ and all the weight vectors $\boldsymbol{w}_i$ are calculated;

$$d_{ji} = \|\boldsymbol{x}_j - \boldsymbol{w}_i\|. \tag{1}$$

(NG3) Each neuron c_i is assigned a rank $k = 0, \cdots, (M-1)$ depending on their d_{ji}. k_i denotes the rank of neuron c_i. The rank of 0 indicates the neuron closest to $\boldsymbol{x}_j$ and the rank of $(M-1)$ indicates the neuron farthest from $\boldsymbol{x}_j$.
(NG4) The weight vectors of the neurons are updated according to

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

where $E(t)$ is the learning rate;

$$E(t) = E_0 \left(E_0 / E_f\right)^{t/T}, \tag{3}$$

where t is the time step and T is the total number of training steps. E_0 and E_f are the initial and the final values of $E(t)$, respectively. $h_\lambda(t, k)$ is the neighborhood ranking function described as

$$h_\lambda(t, k) = \exp\left(-k_i / \lambda(t)\right), \tag{4}$$

where the parameter $\lambda(t)$ controls the convergence rate of the neighborhood ranking function;

$$\lambda(t) = \lambda_0 \left(\lambda_0 / \lambda_f\right)^{t/T}, \tag{5}$$

where λ_0 and λ_f are the initial and the final values of $\lambda(t)$, respectively.
(NG5) If $t < T$, return to (NG1).

3. Proposed Methods

3.1. NG Containing Two Kinds of Neurons (TN-NG)

We explain the algorithm of the proposed NG which has two kinds of neurons whose features are different (called TN-NG). The network consists of a set A containing M neurons; $A = \{c_1, c_2, \cdots, c_M\}$. The all neurons are classified into two kinds of sets; A_1 and A_2 at random. A_1 contains m_1 neurons, and A_2 contains m_2 neurons, namely $M = m_1 + m_2$. The neurons belonging to A_1 converges rapidly and the neurons

belonging to A_2 converges slowly. To be specific, the convergence rate of the neighborhood ranking function is different between A_1 and A_2. The initial value of the weight vectors are given at the same method of the conventional NG.

(TN-NG1) An input vector $\boldsymbol{x}_j$ is inputted to all the neurons at the same time in parallel.
(TN-NG2) Euclid distances d_{ji} between an input vector $\boldsymbol{x}_j$ and all the weight vectors $\boldsymbol{w}_i$ are calculated according to Eq. (1).
(TN-NG3) Each neuron c_i is assigned a rank $k = 0, \cdots, (M-1)$ depending on their d_{ji}.
(TN-NG4) The weight vectors of the neurons are updated according to

$$\boldsymbol{w}_i(t+1) = \boldsymbol{w}_i(t) + E(t)h_{\lambda_N}(t,k)(\boldsymbol{x}_j - \boldsymbol{w}_i(t)). \tag{6}$$

$h_{\lambda_N}(t,k)$ is the neighborhood ranking function of TN-NG described as;

$$h_{\lambda_N}(t,k) = \exp\left(-k_i/\lambda_N(t)\right), \tag{7}$$

where $\lambda_N(t)$ is decided by neuron's character;

$$\lambda_N(t) = \begin{cases} \lambda_0\left(\dfrac{\lambda_f}{\lambda_0}\right)^{\frac{t}{T}}, & c_i \in A_1, \\ (p+\lambda_0-\lambda_f)\left(\dfrac{p}{p+\lambda_0-\lambda_f}\right)^{\frac{t}{T}} & \\ \quad +\lambda_f - p, & c_i \in A_2, \end{cases} \tag{8}$$

where the fixed parameter p controls the convergence rate of the $\lambda_N(t)$. As a result of Eq. (8), λ_N of A_1 decreases rapidly, and λ_N of A_2 decreases slowly. In other words, the neurons of A_1 converge rapidly, and the neurons of A_2 converge slowly.
(TN-NG5) If $t < T$, return to (TN-NG1).

3.2. NG Containing Two Kinds of Update Functions (TF-NG)

We explain the learning algorithm of the proposed NG which has two kinds of update functions whose features are different (called TF-NG). The network consists of a set A containing M neurons; $A = \{c_1, c_2, \cdots, c_M\}$. The all neurons are classified into two kinds of sets; A_1 and A_2 at random. A_1 contains m_1 neurons, and A_2 contains m_2 neurons, namely $M = m_1 + m_2$. The neurons belonging to A_1 converges rapidly and the neurons belonging to A_2 converges slowly. To be specific, the convergence rate of the learning rate is different between A_1 and A_2. The initial value of the weight vectors are given at the same method of the conventional NG.

(TF-NG1) An input vector $\boldsymbol{x}_j$ is inputted to all the neurons at the same time in parallel.
(TF-NG2) Euclid distances d_{ji} between an input vector $\boldsymbol{x}_j$ and all the weight vectors $\boldsymbol{w}_i$ are calculated according to Eq. (1).
(TF-NG3) Each neuron c_i is assigned a rank $k = 0, \cdots, (M-1)$ depending on their d_{ji}. The closest neuron to $\boldsymbol{x}_j$, namely $k_i = 0$, is defined as a winner.
(TF-NG4) The weight vectors of the neurons are updated according to

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

where $E_F(t)$ is decided by winner's character;

$$E_F(t) = \begin{cases} E_0\left(\dfrac{E_f}{E_0}\right)^{\frac{t}{T}}, & \text{winner} \in A_1, \\ (q+E_0-E_f)\left(\dfrac{q}{q+E_0-E_f}\right)^{\frac{t}{T}} & \\ \quad +E_f - q, & \text{winner} \in A_2, \end{cases} \tag{10}$$

where the fixed parameter q controls the convergence rate of the learning rate $E_F(t)$ and $q \geq E_f$. In this way, all neurons learn according to winner's character. As a result of Eq. (10), the learning rate $E_F(t)$ of A_1 converges rapidly, and $E_F(t)$ of A_2 converges slowly. In other words, if winner belongs to A_1, all neurons learn based on the character of A_1, otherwise, all neurons learn based on the character of A_2.
(TF-NG5) If $t < T$, return to (TF-NG1).

4. Simulation Results

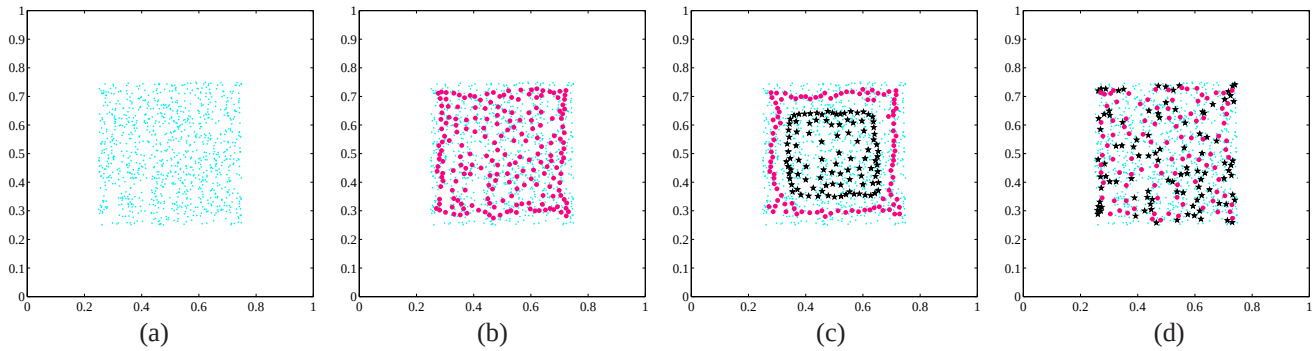
We carry out learning simulation for the 2-dimensional input data, shown in Fig. 1(a). The total number of the input data N is 1000, and the input data has a square cluster. All the input data are sorted at random.

The leaning conditions are as follows. The conventional NG, TN-NG and TF-NG have 200 neurons. In TN-NG and TF-NG, the numbers of neurons included in A_1 and A_2 are equal, namely $m_1 = m_2 = 100$. We repeat the learning 5 times for all input data, namely $T = 5000$. The parameters of the learning are chosen as follows;

(NG) $E_0 = 0.7$, $E_f = 0.01$, $\lambda_0 = 70$, $\lambda_f = 0.2$
(TN-NG) $E_0 = 0.7$, $E_f = 0.01$, $\lambda_0 = 70$, $\lambda_f = 0.2$, $p = 50$
(TF-NG) $E_0 = 0.7$, $E_f = 0.01$, $\lambda_0 = 70$, $\lambda_f = 0.2$, $q = 50$

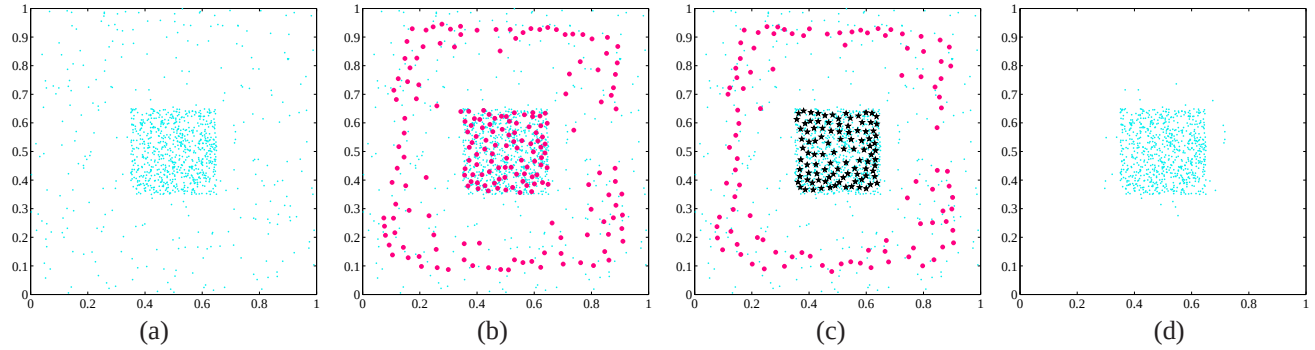
where we use the same E_0, E_f, λ_0, and λ_f to the conventional NG, TN-NG and TF-NG for the comparison.

The learning results of the conventional NG, TN-NG and TF-NG are shown in Figs. 1(b), (c) and (d), respectively. In Fig. 1(c) and (d), we can see that TN-NG and TF-NG has two kinds of neurons. In Fig. 1(c), we can see that the input data on the interior of the cluster are learned by the neurons of A_2, and the exteriors are learned by the neurons of A_1. In other words, in TN-NG, the neurons of A_1 and A_2 do not mingle and respective neurons have own area by each group. This is because the convergence rate to the input data is different in two



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

Figure 1: Computer simulation for 2-dimensional data. (a) Input data. (b) Learning result of conventional NG. (c) Learning result of TN-NG. ● and ★ denote neurons belonging to A_1 and A_2, respectively. (d) Learning result of TF-NG. ● and ★ denote neurons belonging to A_1 and A_2, respectively.



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

Figure 2: Simulation for data including noises. (a) Input data. (b) Learning result by conventional NG. (c) Learning result by TN-NG. (d) Extracted cluster by TN-NG.

kinds of neurons. Meanwhile, in Fig. 1(b) and (d), it is difficult to see differences of the simulation results between the conventional NG and TF-NG. Therefore, we use a following measurement to evaluate the learning performances of three algorithms.

Quantization Error *Qe*: This measures the average distance between each input vector and its winner [3]. Therefore, the small value *Qe* is more desirable.

In Table 1, the quantization error *Qe* of TF-NG is the smallest in three algorithms and has improved 5.47% from the conventional NG. However, *Qe* of TN-NG is larger than the conventional NG and has not improved.

Table 1: Quantization error *Qe* for 2-dimensional data

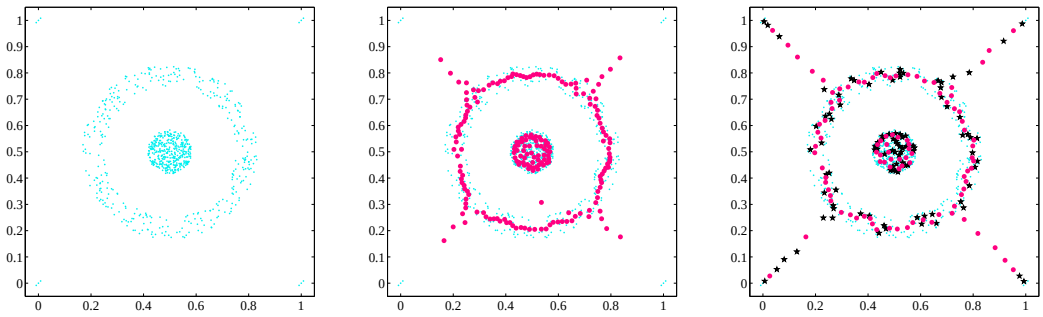
	Conventional NG	TN-NG	TF-NG
Qe	0.0146	0.0155	0.0138

5. Application of TN-NG to Noise Reduction

In the simulation for Fig. 1, the quantization error *Qe* has not improved from the conventional NG. However, because we can confirm that the each neuron group of A_1 and A_2 has own area, we apply TN-NG to noise reduction.

We consider 2-dimensional input data as shown in Fig. 2(a). The input data has 1 cluster and noises. There are 700 points in the cluster and 300 points for noises. All the input data are sorted at random. We repeat the learning 5 times for all input data, namely $T = 5000$. The leaning conditions are the same used in Fig. 1.

The simulation results of the conventional NG and TN-NG are shown in Figs. 2(b) and (c), respectively. We can see that some neurons of the conventional NG learn the input data of not only the cluster but also the noises. The other side, in the result of the TN-NG, the A_2 neurons learn the cluster and the A_1 neurons learn the noises. In other words, TN-NG learns by using different kind of neurons in the cluster and noises while the conventional NG uses only one kinds of neurons. Furthermore, we classify each input data into the group corresponding to its winner's character. Fig. 2(d) shows the extracted input data corresponding to the group of



(a) (b) (c)

Figure 3: Simulation for Target data. (a) Input data. (b) Learning result by conventional NG. (c) Learning result by TF-NG.

A_2. As a result, we can see that noises are removed from the input data and only cluster is extracted. Therefore, we can say that TN-NG can be used for noise reduction and it is effective.

6. Application of TF-NG to Feature Extraction

Because TF-NG has more effective learning result with smaller the quantization error *Qe* than the conventional NG in the simulation for Fig. 1, we apply TF-NG to feature extraction.

We consider the 2-dimensional input data; Target data set [2] shown in Fig. 3(a). The total number of the input data N is 770, and the input data has six clusters which include 4 outliers. All the input data are sorted at random. We repeat the learning 5 times for all input data, namely $T = 3850$. The leaning conditions are the same used in Fig. 1.

The learning results of the conventional NG and TF-NG are shown in Figs. 3(b) and (c), respectively. We can see that some neurons of TF-NG are attracted to the corner input data more strongly than the conventional NG. Thus, TF-NG can obtain more effective result reflecting the features of the input data than the conventional NG.

The calculated quantization error *Qe* are summarised in Table 2. *Qe* of TF-NG is smaller than the conventional NG and has improved 25.7% from the conventional NG.

Table 2: Quantization error *Qe* for Target data

	Conventional NG	TF-NG
Qe	0.0171	0.0127

Fig. 4 shows the quantization error *Qe* of the learning result when changing the maximum number of the repeating times $T_{\max}$ of all the input data ($T = N \times T_{\max}$). In this figure, all *Qe* of TF-NG are smaller than conventional NG. This implies that TF-NG can converge much faster than the conventional NG. From these results, we can say that TF-NG can obtain more effective result than the conventional NG and can be used for feature extraction.

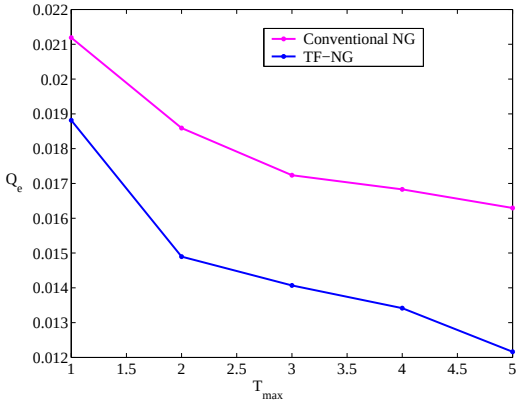


Figure 4: Quantization error *Qe* versus maximum repeating times $T_{\max}$.

7. Conclusions

In this study, we have proposed two new Neural Gas algorithm; Neural Gas containing two kinds of update Neurons (TN-NG) and Neural Gas Containing Two Kinds of Update Functions (TF-NG). From each learning behavior, we have confirmed that TN-NG is effective for application to noise reduction and TF-NG is effective for application to feature extraction.

Acknowledgment

This work was partly supported by Yamaha Music Foundation.

References

[1] T. M. Martinetz and K. J. Schulten, "A "neural-gas" network learns topologies. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors," *Artificial Neural Networks*, pp. 397–402, 1991.

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

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