An Incremental Self-organizing Neural Network Based on Enhanced Competitive Hebbian Learning

Hao Liu, Masahito Kurihara, Satoshi Oyama, Haruhiko Sato

Abstract— **Self-organizing neural networks are important tools for realizing unsupervised learning. Recently, a difficult task has involved the incremental, efficient and robust learning in noisy environments. Most of the existing techniques are poor in this regard. In this paper, we first propose a new topology generating method called enhanced competitive Hebbian learning (enhanced CHL), and then propose a novel incremental self-organizing neural network based on the enhanced CHL method, called enhanced incremental growing neural gas (Hi-GNG). The experiments presented in this paper show that the Hi-GNG algorithm can automatically and efficiently generate a topological structure with a suitable number of neurons and that the proposed algorithm is robust to noisy data.**

I. Introduction

SELF-ORGANIZING neural networks have played an important role in the field of unsupervised learning over the past few decades. Unsupervised learning has two main objectives: clustering and data topology learning. Clustering aims to divide a given data set into several clusters, where each pair of data in the same cluster has greater similarity than that in two different clusters [1]. On the other hand, data topology learning can be described as follows: given a high-dimensional data distribution, project input data into a topological structure in which similar data in the input space are projected into topological adjacent units [2]. Recently, this technique has been widely applied to data mining, vector quantization, pattern recognition, computer vision and many other related fields [3].

Recently, with the increasing number and dimensions of data, the learning algorithms are required to efficiently deal with large number of signals. Moreover, a much more difficult task for on-line learning is to efficiently and robustly learn data from the distributions in which noisy data exist. The main difficulty is that learning algorithms have no prior knowledge of the whole data distribution. Thus, upon the arrival of the first several data of a certain distribution, the amount of data is not sufficient to represent the whole distribution. At this time, learning algorithms cannot judge whether these data are noisy or normal. Consequently, for each iteration, the existing techniques (such as self-organizing map (SOM) [4], neural gas (NG) [5], etc.) have to respond to the new data and update the weight vectors of the corresponding neurons, which usually cause a critical deviation of the topology reflection in the result.

Hao Liu, Masahito Kurihara, Satoshi Oyama and Haruhiko Sato are with Division of Synergetic Information Science in Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan.
E-mail: liuhao@complex.ist.hokudai.ac.jp, kurihara@ist.hokudai.ac.jp, oyama@ist.hokudai.ac.jp, haru@complex.ist.hokudai.ac.jp

Another problem is that in most 'growing-type' self-organizing neural networks, such as growing neural gas (GNG) [6] (See more discussions in Section II), the number of neurons will continuously increase owing to the growth strategy. The large number of neurons increases the computational cost of searching for the winner neurons in each iteration, which makes the training procedure inefficient. In fact, such a number cannot increase indefinitely because of the limited computer resources, such as CPU speed and memory size. To partially solve this problem, the maximum number of neurons in the network is usually predefined by an additional parameter. However, the consequent problem is in determining this parameter. If the parameter is set with a low value, some small clusters will be merged indicating that the clustering result will be wrong. If the parameter is set too high, many neurons will be generated and some large cluster may be divided into several smaller clusters, which may also results in a low clustering quality.

To solve the problems mentioned above, in this paper, we will first extend the traditional competitive Hebbian learning (CHL) [7] method (See more discussions in Section III-A) as a new topology generating technique called enhanced competitive Hebbian learning (enhanced CHL), and then we will propose a novel incremental self-organizing neural network based on the enhanced CHL method, called enhanced incremental growing neural gas (Hi-GNG). The main contributions of this algorithm are summarized as follows:

1) The algorithm can efficiently learn a given data distribution.
2) The algorithm can automatically adapt a suitable number of neurons to generate the topological structure.
3) The generated topological structure can be constantly and robustly maintained without being influenced by noisy data.

The rest of this paper is organized as follows. In Section II, we briefly describes the overview of the self-organizing neural networks. In Section III, we first review the CHL method, and then propose the enhanced CHL method. In Section IV, we propose the Hi-GNG algorithm. In Section V, experimental results are presented. Finally, we summarize the features of Hi-GNG and give conclusions in Section VI.

II. Related Work

One of the most well-known techniques is SOM, also known as the Kohonen network. The original SOM has two layers: the input layer and the competitive layer. The competitive layer consists of a set of units (also called neurons or nodes) with lateral connections, which is usually constructed

as a two-dimensional topological structure. A weight vector is assigned to each unit and is updated during training procedures according to the simple competitive learning (SCL) [8] strategy. When the training procedure is finished, SOM divides the input space into several regions, which can be considered that the input space is described by SOM as a Voronoi diagram, and each best matching unit (BMU) is the site of the corresponding Voronoi cell, indicating that SOM is suitable for clustering tasks. Another powerful feature of SOM is that high-dimensional data can be projected to a low-dimensional topological structure, which means that SOM can be applied to data visualization. Unfortunately, SOM has some drawbacks, one of which is its fixed structure in its competitive layer. The size of the network must be predefined and is unchangeable during training procedures, which is similar to providing an integer number k as a parameter in the k-means clustering algorithm. In general, it is difficult to determine k without any prior knowledge of the given data set [9]. In addition, the fixed structure limits SOM to the detection of clusters which are presented as complex shapes.

In order to overcome the drawback caused by the fixed structure, a series of 'growing-type' self-organizing neural networks have been proposed. These techniques usually have dynamic network structures.

Growing cell structures (GCS) [10] have a flexible network consisting of k-dimensional simplices. Each node is assigned a local variable, called a signal counter, which presents the local error of a neuron in the training procedure. GCS starts with a random k-dimensional simplex, e.g. the network is a triangular network if $k = 2$, and then, at timed intervals, new nodes will be inserted by splitting the $longest\ edge$ emanating from the node with the maximum accumulated error. However, the topology dimensions in GCS must be kept strictly: when inserting a node, additional edges are also required to be added in order to maintain the topological structures. Further, another problem is that once a node is created in GCS, it cannot be removed.

Unlike SOM and GCS, the GNG method has a more flexible network structure in which nodes can be added or removed, and there are no constrains for the topological structure. Actually, it can be considered as a combination of CHL, NG and the growing strategy of GCS. In the training procedure, GNG starts with two neurons. An edge will be created between two nodes with the highest activity if there is no edge between them. New nodes are inserted in the same way as that of GCS. The algorithm will stop if some predefined condition is met. Because of the presence of these important features, the GNG algorithm is considered to be suitable for the task of on-line learning.

With the increasing number and dimensions of data, the learning algorithms are required to efficiently deal with large number of signals. In general, there are two main approaches to speed up the GNG algorithm. The first approach is to reduce the computational complexity of GNG, such as density-based growing neural gas (DB-GNG) [11] and the literature [12]. They usually establish a supporting structure, such as an R-tree, slim-tree or KD-tree, in order to reduce the cost of finding the nearest neuron. Another approach tries to modify the growing mechanism, such as fast autonomous growing neural gas (fAGNG) [13] and incremental growing neural gas (IGNG) [14]. The fAGNG algorithm performs the insertion of k neurons every λ signals. IGNG uses a distance-check strategy to insert neurons, which means that there is no necessary to calculate the local accumulated errors. Furthermore, it is not required to find the neurons with the maximum accumulated error when inserts neurons. These features make IGNG more efficient than the original GNG algorithm.

Considering that the given data distribution contains some noisy data, it will become a very difficult task for the methods mentioned above to learn such data distributions robustly. Unfortunately, few self-organizing neural networks were developed to address this problem. The self-organizing incremental neural network (SOINN) [15] was designed for the learning of non-stationary data distributions. SOINN has two layers in its architecture. The first layer performs like the GNG algorithm and the second layer is used to detect the potential low-density areas using the output of the first layer as its input. However, this two-layer architecture is sometimes too complicated for an on-line learning algorithm as users do not know when to stop the first layer and when to start the second layer. In addition, two layers means that there are more parameters to be determined. In order to deal with noisy data, Robust Growing Neural Gas (RGNG) [16] extended the GNG algorithm with a noisy data resistant scheme. However, like the original GNG algorithm, RGNG also needs to calculate the local accumulated errors, which means that RGNG has a higher computational complexity than the IGNG like techniques.

III. Enhanced Competitive Hebbian Learning

A. Competitive Hebbian Learning

The classical Hebbian theory (also called Hebbian rule) [17] describes a basic mechanism for how neurons can connect with each other, more precisely, it shows how to create the connections between neurons. According to Hebbian theory, any two neurons sufficiently near to each other that are repeatedly activated at the same time will tend to become 'associated' indicating that a connection can be created between them.

CHL can be considered as an extension of the classical Hebbian theory in competitive learning systems. The key idea of CHL is that given an input signal, we find the nearest neuron and the second-nearest neuron in the network, after which we create a connection (edge) between these two neurons if such a connection does not already exist.

CHL is a good method for the creation of connections between neurons in competitive learning systems. However, CHL only provides a way of creating connections, but does not address the elimination of the existing connections. Some learning algorithms, such as GNG, use a local variable to record the 'age' of a connection. The 'age' of a connection

will increase according to some updating strategy. If a connection's 'age' has exceeded a particular threshold, it will accordingly be removed. In general, such a threshold is a positive integer, e.g. the threshold is given by the parameter α_{max} in the GNG algorithm. Here, α_{max} is usually set with a big value because the network will not grow if α_{max} is too small. Unfortunately, considering that the given data set contains some noisy data, the final graph will be significantly influenced by the noisy data if α_{max} is big, because the local variable 'age' may be easily reset to zero and the corresponding connections will be kept for a long time. This means that the learning algorithm will not robustly reflect the data relationships from noisy data distributions.

B. Enhanced Competitive Hebbian Learning

In order to solve this problem, we propose a new technique to generate topological structures, called enhanced competitive Hebbian learning (enhanced CHL). The enhanced CHL method considers both the creation of connections between neurons and the elimination of the existing connections between neurons.

The aim of enhanced CHL is to generate an undirected weighted graph, $\langle G, w\rangle$, where $G = \langle V, E, \psi_G\rangle$ consists of a vertices set, an edge set and an incidence function. The weighting $w : E \to \mathbb{R}$ can be considered as a vector whose coordinates are indexed by the edge set E. For example, let $\{v_1, v_2\}$ be an unordered pair of two vertices of G and e be an edge of G. Then $\psi_G(e) = \{v_1, v_2\}$ indicates that $\{v_1, v_2\}$ is associated with e, and $w(e)$ represents the weight of the edge e.

In this paper, the vertex and the weighted graph are called a $neuron$ and a $network$, respectively, and are denoted by n and net, respectively. On the basis of these standard concepts of graph theory, we give the formal definition of a connection in the enhanced CHL method.

Definition 1: ($connection$) Let n_1 and n_2 be two neurons in the network ($n_1 \neq n_2$) and let $\{n_1, n_2\}$ be the edge between n_1 and n_2. A $connection$, denoted as $c(n_1, n_2)$, is a pair:

$$c(n_1, n_2) = \langle \{n_1, n_2\}, s\rangle \tag{1}$$

where s is a non-negative integer, called the $connection\text{-}strength$ of $c(n_1, n_2)$. For convenience, we also call it the $connection\text{-}strength$ between n_1 and n_2, denoted by $s(n_1, n_2)$.

Given an input signal ξ and a network, net, the nearest neuron, $n^\star$, and the second-nearest neuron, $n^{\star\star}$, to the signal ξ can be obtained by Equation 2.

$$\begin{aligned} n^\star &= \arg\min_{n_i \in N(net)} \|\xi - W_{n_i}\| \\ n^{\star\star} &= \arg\min_{n_i \in N(net)\setminus\{n^\star\}} \|\xi - W_{n_i}\| \end{aligned} \tag{2}$$

where W_{n_i} is the weight vector of n_i and $N(net)$ is the neuron set of the network.

Then let net be a network and $n^\star$, $n^{\star\star}$ be the nearest neuron and the second-nearest neuron to an input signal ξ, respectively ($n^\star, n^{\star\star} \in N(net)$). The enhanced CHL method is described in the following four parts:

1) CREATION: Create a new $connection$ $c(n^\star, n^{\star\star})$ if $c(n^\star, n^{\star\star})$ does not exist. When the new $connection$ is established, the $connection\text{-}strength$ is initialized by: $s(n^\star, n^{\star\star}) \leftarrow 1$.
2) UPDATING: If the $connection$, $c(n^\star, n^{\star\star})$, already exists, then its $connection\text{-}strength$ is enhanced by: $s(n^\star, n^{\star\star}) \leftarrow s(n^\star, n^{\star\star}) + \Delta s$, where Δs is a positive integer. If $\exists c(n^\star, n_i)$ where $n_i \in N(net)$ such that $n_i \neq n^{\star\star}$, then the $connection\text{-}strength$ of these $connections$ are reduced by: $s(n^\star, n_i) \leftarrow s(n^\star, n_i) - 1$.
3) DECAYING: when the elapsed time is an integer multiple of some desired number, the $connection\text{-}strength$ of every $connection$ in the network will be reduced by: $s(n_i, n_j) \leftarrow s(n_i, n_j) - 1, n_i, n_j \in N(net)$.
4) ELIMINATION: if $\exists c(n_i, n_j)$ where $n_i, n_j \in N(net), n_i \neq n_j$ such that $s(n_i, n_j) = 0$, then remove the $connection$, $c(n_i, n_j)$.

In the creation part, the enhanced CHL method follows the idea of Hebbian theory, i.e. a $connection$ is created between two neurons if they are near enough and are activated at the same time. However, in our model, we do not care only about how to create a $connection$, but also about the $connection\text{-}strength$ of the created $connection$. The $connection\text{-}strength$ of the newly created $connections$ are initialized by one, indicating that these $connections$ are temporal ones. If they are not enhanced in the following few steps, they can be quickly removed.

In the second part, the updating is also related to Hebbian theory. If a $connection$ between two $neurons$ has already been created in some earlier steps and subsequently the two $neurons$ are again activated at the same time, then the $connection\text{-}strength$ will be enhanced. Moreover, we also consider the opposite situation: if the two neurons are not simultaneously activated, the $connection\text{-}strength$ between them will be weakened.

In the decaying part, the $connection\text{-}strength$ between neurons will be gradually weakened with time. In order to make an easy calculation, the $connection\text{-}strength$ will be reduced by one if the total elapsed time is an integer multiple of some desired number.

In the eliminating part, the $connections$ in the $network$ will be removed if their $connection\text{-}strength$ values are zero.

Like CHL, the enhanced CHL method is also a pure topology generating method. To incrementally learn the given data distribution, $p(\xi)$, the learning algorithm also requires a way of inserting neurons and a vector quantization technique in order to generate and place the neurons in regions in which the probability density $P(\xi) > 0$. These methods are presented in Section IV.

IV. The Algorithm of Enhanced Incremental Growing Neural Gas

In this section, we present the enhanced incremental growing neural gas (Hi-GNG) algorithm which is based on the enhanced CHL method.

In Hi-GNG, the classical SOM neuron model is modified by assigning an additional local attribute called a $maturity\text{-}level$. Now we give the definition of a $neuron$ in the Hi-GNG algorithm.

Definition 2: ($neuron$) A $neuron$, denoted by n, is a pair:

$$n = \langle W, m \rangle \tag{3}$$

where W is the weight-vector of n in the input space and m is a non-negative integer called the $maturity\text{-}level$. We denote the weight of n and the $maturity\text{-}level$ of n by W_n and $m(n)$, respectively.

In this modified neuron mode, the new attribute m functions as a local counter to count the number of times that a $neuron$ becomes the nearest neuron or the second-nearest neuron. Given an integer number, $\hat{m}$, as the threshold, if the $maturity\text{-}level$ of $neuron$ n is greater than $\hat{m}$, i.e. $m(n) > \hat{m}$, we say n becomes 'mature'.

On the basis of this modified neuron model and the enhanced CHL method presented in Section III-B, we present the Hi-GNG algorithm as follows:

1) Initialized the network with an empty graph.
2) Randomly generate a signal ξ from $P(\xi)$.
3) If the network is empty or the Euclidean distance between the input signal and the nearest (or the second-nearest) neuron satisfies Equation 4, then insert two $neurons$.

$$\exists n \in \{n^\star, n^{\star\star}\},\ \|\xi - W_n\| > \sigma \tag{4}$$

where ξ is the current input signal and W_n is the weight vector of the nearest neuron ($n^\star$) or the second-nearest neuron ($n^{\star\star}$).
Let n_1 and n_2 be the two newly inserted neurons. The weight of n_1 is set with $W_{n_1} \leftarrow \xi$ and n_2 is placed very close to n_1 in the input space. Here, W_{n_2} is randomly generated, such that $\|W_{n_1} - W_{n_2}\| < \frac{\sigma}{10}$. Then, create a $connection$ between n_1 and n_2 with the $connection\text{-}strength$: $s(n_1, n_2) \leftarrow 1$. Finally, the adaption of the current input signal, ξ, is complete. Continue to adapt the next signal.
4) Update the weight vectors of the neurons using Equation 5.

$$W_n = \begin{cases} W_n + \epsilon_b (\xi - W_n), & \text{if } n = n^\star \\ W_n + \epsilon_n (\xi - W_n), & \text{if } \exists c(n, n^\star) \end{cases} \tag{5}$$

where $n^\star$ is the nearest neuron to the input signal ξ, and ϵ_b and ϵ_n are two real numbers in the range of $(0, 1)$, which represent the constant learning rate for the nearest neuron and its topological neighbors. In general cases, ϵ_b is greater than ϵ_n.
5) Increment the $maturity\text{-}level$ of the nearest neuron, $n^\star$, and the second-nearest neuron, $n^{\star\star}$.
6) Create a $connection$ between $n^\star$ and $n^{\star\star}$ if there is no such a $connection$. Initialize the $connection\text{-}strength$ by one.
7) Increment the $connection\text{-}strength$ using Equation 6, if a connection exists between $n^\star$ and $n^{\star\star}$.

$$s(n^\star, n^{\star\star}) \leftarrow s(n^\star, n^{\star\star}) + \Delta s \tag{6}$$

where Δ s is a positive integer presenting the increment of the $connection\text{-}strength$.
8) Reduce the $connection\text{-}strength$ between the nearest neuron, $n^\star$, and its topological neighbors using Equation 7.

$$s(n^\star, n_i) \leftarrow s(n^\star, n_i) - 1, \text{ if } \exists c(n^\star, n_i) \tag{7}$$

9) Remove the $connections$ whose $connection\text{-}strength$ is zero and if this isolates some $neurons$, also remove the isolated $neurons$ as well.
10) Reduce the $connection\text{-}strength$ of all the $connections$ in the network using Equation 8, if the number of iterations so far is an integer multiple of the parameter α.

$$s(n^\star, n_i) \leftarrow s(n^\star, n_i) - 1, n_i \in N(net) \tag{8}$$

11) If the stopping condition is not met, then go to step 2). Otherwise, output the network which only consists of the $neurons$ whose $maturity\text{-}level$ is larger than the parameter $\hat{m}$

In step 3), as with the IGNG algorithm, the Hi-GNG algorithm also applied a distance-check strategy when neurons are inserted. However, the IGNG algorithm inserts only one neuron, whereas the Hi-GNG algorithm inserts two neurons. The main reason for inserting two neurons is that the distance between the input signal and the nearest neuron is larger than a threshold, which means that the current input signal is far away from the signals learned so far. Thus, there is a high probability that the current signal belongs to a new cluster or is noisy signal. In this case, the network should create a 'new land' (a new topological structure disjointed from the existing structure) to represent a new cluster or a noisy signal (this can also be regarded as a small cluster). If we consider that each cluster is represented by a component in the network graph, two connected neurons can be inserted to create a minimal component for representing a new cluster. A component which contains only one isolated neuron will be directly removed in the next iteration because of step 9).

In step 5), the $maturity\text{-}level$ is incremented, but note that only the nearest neuron and the second-nearest neuron will be manipulated, which is different from other techniques, such as IGNG. The IGNG algorithm uses a local variable, called 'age', to identify whether or not a neuron is 'mature'. The nearest neuron and all its topological neighbors will be manipulated in the IGNG algorithm.

From step 6) to step 10), the enhanced CHL method is applied. When a $connection$ is created, the $connection\text{-}strength$ is very weak, which means that the new $connection$ must be enhanced within a small numbers of

Algorithm 1 Hi-GNG

Input: X, σ, ϵ_b, ϵ_n, Δs, α, $\hat{m}$

Output: an undirected weighted graph only consists of the mature neurons ($m(n) > \hat{m}$) and their corresponding connections.

1: Initialize the network with an empty graph;
2: $g \leftarrow 0$;
3: **repeat**
4: $\xi \leftarrow$ generate a signal ξ from $P(\xi)$ randomly;
5: Search the network for the nearest neuron, $n^\star$ and the second-nearest neuron, $n^{\star\star}$ by:
$n^\star \leftarrow \arg\min_{n_i \in N(net)} \|\xi - W_{n_i}\|$
$n^{\star\star} \leftarrow \arg\min_{n_i \in N(net)\setminus\{n^\star\}} \|\xi - W_{n_i}\|$
6: **if** $N(net) = \phi$ or $\|\xi - W_{n^\star}\| > \sigma$ or $\|\xi - W_{n^{\star\star}}\| > \sigma$ **then**
7: Insert two neurons, n_1 and n_2, to the network and set them by:
$W_{n_1} \leftarrow \xi$ and randomly generate W_{n_2}, such that $\|W_{n_1} - W_{n_2}\| < \frac{\sigma}{10}$. Then create a connection between n_1 and n_2 and set the connection-strength by: $s(n_1, n_2) \leftarrow 1$;
8: continue;
9: **end if**
10: Update the neurons by:
$W_{n^\star} \leftarrow W_{n^\star} + \epsilon_b (\xi - W_{n^\star})$
$W_{n_i} \leftarrow W_{n_i} + \epsilon_n (\xi - W_{n_i})$ // $\exists c(n_i, n^\star)$
11: Incremental the maturity level of $n^\star$ and $n^{\star\star}$ by:
$m(n^\star) \leftarrow m(n^\star) + 1$
$m(n^{\star\star}) \leftarrow m(n^{\star\star}) + 1$
12: **if** $c(n^\star, n^{\star\star})$ has already exist **then**
13: Update the connection-strength of $c(n^\star, n^{\star\star})$ by:
$s(n^\star, n^{\star\star}) \leftarrow s(n^\star, n^{\star\star}) + \Delta s$;
14: **else**
15: Create a connection $c(n^\star, n^{\star\star})$ and set the connection strength as:
$s(n^\star, n^{\star\star}) \leftarrow 1$;
16: **end if**
17: Reduce the connection-strength of the connections between $n^\star$ and its topological neighbors by:
$s(n^\star, n_i) \leftarrow s(n^\star, n_i) - 1$;
18: Remove the connections whose connection-strength is zero and if this make some neurons isolated, also remove the isolated neurons.
19: **if** $g \mod \alpha = 0$ **then**
20: Reduce the connection-strength of all existing connections by:
$s(n_i, n_j) \leftarrow s(n_i, n_j) - 1$;
21: **end if**
22: $g \leftarrow g + 1$;
23: **until** the stopping condition is reached

iterations, or it will be removed quickly owing to step 9). This strategy makes the Hi-GNG algorithm robust to noisy signals. When a noisy signal is input, there is a high possibility that Equation 4 is satisfied. Then, the algorithm inserts two connected neurons to respond to this noisy signal. Because the noisy signal can be expected to have a low probability of being regenerated, it is also expected that the *connection* between the just created two neurons is not easily enhanced. Therefore, the neurons presenting the noisy data can be quickly removed. This is similar to the short-term plasticity in biological neural behaviors. On the other hand, the *connections* between those neurons which present the normal data are easily enhanced time and again. Such neurons can therefore be kept in the network for a long time. This appears like the long-term plasticity in biological neural behaviors.

Details regarding the implementation of Hi-GNG are presented in Algorithm 1.

V. Experiments

In the following experiments, two artificial data sets were used to test the performance of Hi-GNG compared with that of the GNG algorithm. We summarized the information regarding these artificial data sets as follows:

1) Data Set 1 contained 1386 two-dimensional data in four clusters. The cluster at the top of Fig. 1(a) was represented by a twisty shape and another three smaller clusters were located under the former bigger one. No noisy data existed in this data set.
2) Data Set 2 had two banana shaped clusters, as illustrated in Fig. 1(b). In total, there were 16532 data including some noisy data.

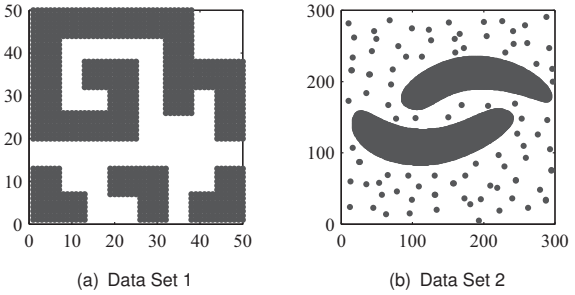

Fig. 1. Artificial data sets used in the experiments. (a) Data Set 1. (b) Data Set 2.

In the experiment involving Data Set 1, we captured five snapshots during the learning procedure (Fig. 2). We observed that the Hi-GNG algorithm generated neurons faster than the GNG algorithm. At the beginning, Hi-GNG started with an empty graph and inserted many neurons very quickly, and the newly inserted neurons started as immature neurons (represented by the 'diamond' shaped markers in Fig. 2(f)), whereas the GNG algorithm started with two neurons and inserted neurons slowly (Fig. 2(a) - Fig. 2(c)). We also noticed that in the Hi-GNG learning procedure, immature neurons

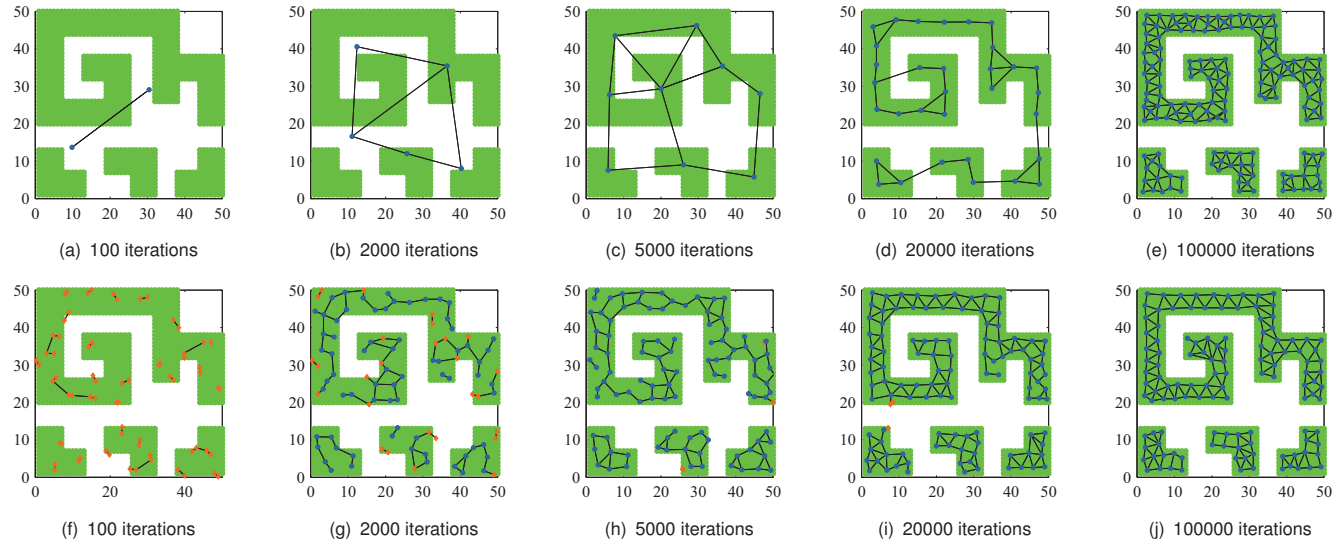

Fig. 2. The snapshots in the learning procedure for Data Set 1. The first row shows the result for GNG and the second row shows the result for Hi-GNG. The parameters of GNG were as follows: $\lambda = 650$, $\epsilon_b = 0.05$, $\epsilon_n = 0.006$, $\alpha_{max} = 80$, $\alpha = 0.5$, $d = 0.0005$. The parameters of Hi-GNG were: $\sigma = 5$, $\epsilon_b = 0.05$, $\epsilon_n = 0.006$, $\Delta s = 20$, $\hat{m} = 30$, $\alpha = 80$.

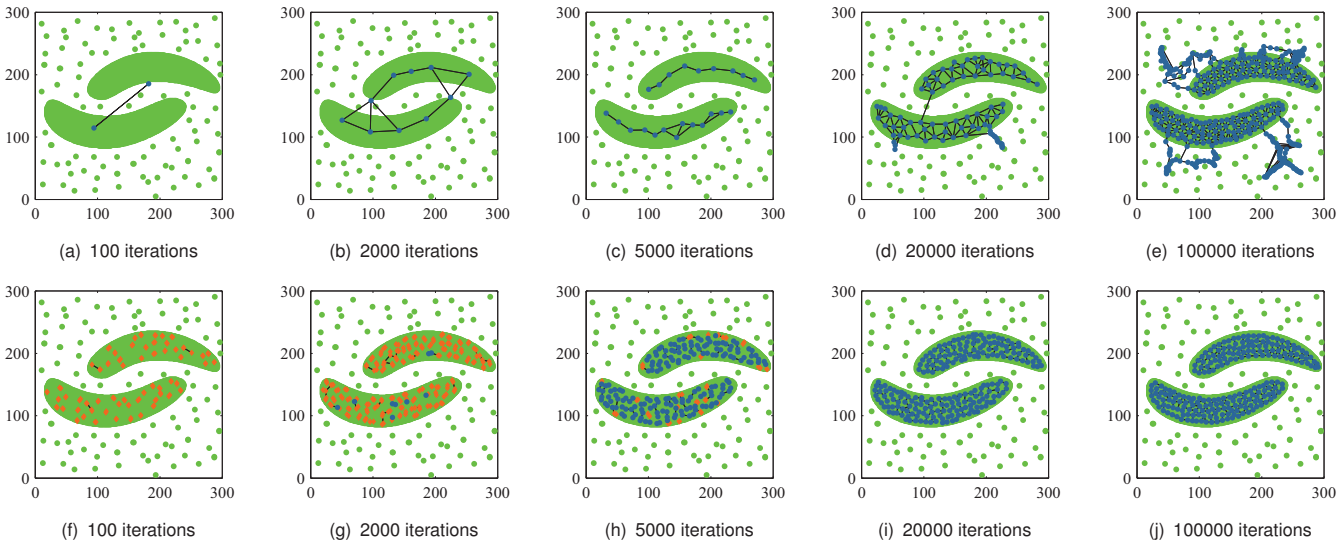

Fig. 3. The snapshots in the learning procedure for Data Set 2. The first row shows the result for GNG and the second row shows the result for Hi-GNG. The parameters of GNG were: $\lambda = 250$, $\epsilon_b = 0.05$, $\epsilon_n = 0.006$, $\alpha_{max} = 80$, $\alpha = 0.5$, $d = 0.0005$. The parameters of Hi-GNG were as follows: $\sigma = 10$, $\epsilon_b = 0.05$, $\epsilon_n = 0.006$, $\Delta s = 20$, $\hat{m} = 30$, $\alpha = 50$.

were widely placed according to the input distribution. In each iteration, new neurons were directly inserted near to the input data if the distance-check (by Equation 4) was satisfied. About 1500-2000 iterations later, some immature neurons became mature (represented by 'solid circle' shaped markers in Fig. 2(g)), and when the number of iteration was 5000, most immature neurons become mature and more connections were established (Fig. 2(h)). When the number of iterations was more than 20000, the data distribution was well adapted by Hi-GNG (Fig. 2(i)) comparing with that of the GNG algorithm, which still needed more iterations to finish the adaption (Fig. 2(d)). Although we could choose a smaller value for the parameter λ in GNG, a smaller value may sometimes result in a bad adaption result.

In addition, we also observed that the number of neurons generated by these two algorithms was significantly different during the whole learning procedure (shown in Fig. 4(a)). If we had not provided an upper limit for the number of neurons, the insertion of neurons would have been a never-ending operation in the GNG algorithm. On the other hand, the number of neurons in Hi-GNG stopped increasing at around 120 after 45000 iterations. Moreover, we separately

recorded the number of mature neurons and immature neurons. The number of mature neurons started from zero and slowly increased at the beginning of several hundreds of iterations, but started to increase very quickly afterwards. Two or three thousands of iterations later, the rate of increase slowed again. Finally, the number became stable when it was at around 120. The number of immature neurons also started at zero, but on the contrary, it first increased quickly and then also decreased quickly. Finally, the number became stable around zero or two and there were very few fluctuations.

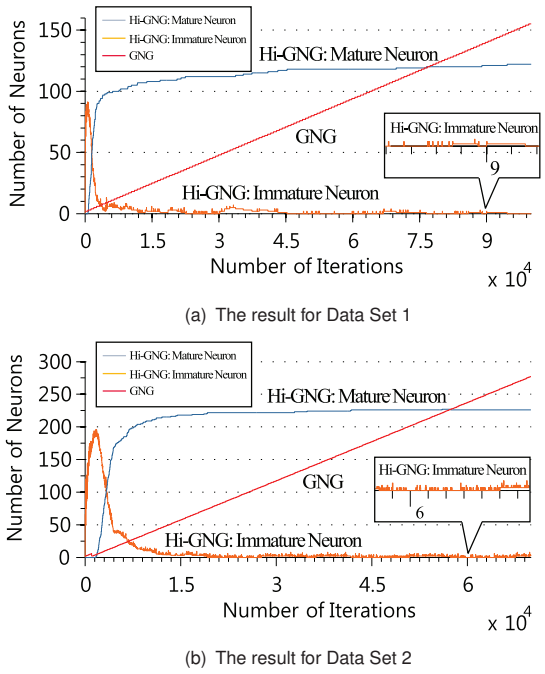

Fig. 4. The number of neurons during the learning procedures.

We also captured five snapshots of the experiment regarding Data Set 2 (Fig. 3). First, we observed that Hi-GNG could generate neurons faster than GNG, which was similar to the previous experiment with Data Set 1. However, because Data Set 2 contains noisy data, the topological structure generated by these two algorithms were quite different. The topological structure generated by GNG became twisty and distorted after 20,000 iterations (Fig. 2(d)), and continuously deteriorated with the increasing number of iterations (Fig. 2(e)). The reason was that the noisy data were usually far away from the normal data, which resulted in a large distance value being calculated using $\|\xi - W\|^2$. Then, this large distance significantly affected the updating of the weights. On the other hand, the topological structure generated by Hi-GNG was stable and always reflected the distribution of the normal data. We recorded the number of neurons to help explain as to why these results were obtained (Fig. 4). As previous result for Data Set 1, the number of mature neurons in Hi-GNG became stable when the number of iterations was more than 20,000. However, at the same time, the number of immature neurons was not so stable and frequently changed. This unstable number appeared when Hi-GNG responded to the noisy data; new immature neurons were inserted into the network as the large distance between the noisy data and the nearest neuron, satisfying the distance-check (Equation 4). Because the connection-strength between the two newly inserted immature neurons was very weak and not easily enhanced, it easily decreased to zero after a few numbers of iterations, which led to the removal of the newly inserted neurons. As a result, the mature neurons which represented the normal data were kept and those immature neurons which represented the noisy data were generated and removed quickly. To better understand this experiment, we have uploaded a video demonstration of this learning procedure on the YouTube website, which can be found at http://youtu.be/s7NhrPqaVXg.

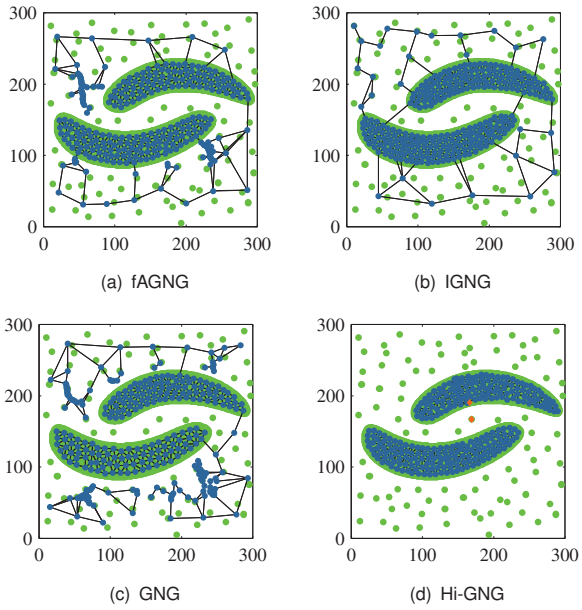

Fig. 5. The comparable results for Data Set 2 when the number of iterations was 500,000. The maximum number of neurons was limited at 250 for GNG, fAGNG and IGNG. The parameters of GNG were: $\lambda = 250$, $\epsilon_b = 0.05$, $\epsilon_n = 0.006$, $\alpha_{max} = 80$, $\alpha = 0.5$, $d = 0.0005$. The parameters of fAGNG were as follows: $k = 3$, $\lambda = 250$, $\epsilon_b = 0.05$, $\epsilon_n = 0.006$, $\alpha_{max} = 80$, $\alpha = 0.5$, $d = 0.0005$. The parameters of IGNG were: $\alpha_{mature} = 20$, $\epsilon_b = 0.05$, $\epsilon_n = 0.006$, $\alpha_{max} = 80$. The parameters of Hi-GNG were as follows: $\sigma = 10$, $\epsilon_b = 0.05$, $\epsilon_n = 0.006$, $\Delta s = 20$, $\hat{m} = 30$, $\alpha = 50$.

In addition, we compared the result of Hi-GNG with those of fAGNG, IGNG and GNG. In this experiment, the maximum number of neurons in GNG, fAGNG and IGNG was set with 250. We captured the experimental snapshots when the number of iterations was 500,000, because all the algorithms became stable at that time (Fig. 5). We observed that fAGNG and IGNG placed some neurons in the low-density areas and all the neurons in the network were connected with each other, which means that there was only one component in the graph indicating that only one cluster could be reported (shown in Fig. 5(a) and Fig. 5(b), respectively). The GNG algorithm reported two components

in its topological graph indicating that there were two clusters in the given data set (Fig. 5(c)). However, the two clusters found by GNG were not the true clusters. On the other hand, the Hi-GNG algorithm automatically inserted a suitable number of neurons to generate the topological structure without being provided the maximum number of neurons in the network. The number of mature neurons became stable at around 270 and the number of immature neurons fluctuated around six. Furthermore, the generated topological structure could always clearly reported that there were two clusters which were exactly the true clusters in the given data set (Fig. 5(d)).

VI. Conclusion

In this paper, we first proposed a new topology generation method, called enhanced competitive Hebbian learning (enhanced CHL), which considers how to both create and eliminate connections between neurons. On the basis of the enhanced CHL method, we further proposed a novel incremental self-organizing neural network, called enhanced incremental growing neural gas (Hi-GNG). The experiments presented in this paper proved that the Hi-GNG algorithm could efficiently learn the given data distribution and the generated topological structure was not influenced by noisy data. In addition, Hi-GNG could automatically generate a topological structure with a suitable number of neurons.

Although Hi-GNG is usually faster than the original GNG algorithm, the efficiency can be further improved by indexing the neurons by some supporting structures, such as R-tree, slim-tree or KD-tree, in order to reduce the cost related with the search for the nearest (or the second-nearest) neuron in the training procedure.

Some promising applications of the Hi-GNG algorithm are incremental clustering with noisy existing data sets, image segmentation and 3-D surface reconstruction.

Acknowledgment

This work was supported by National Natural Science Foundation of China (No. 61272357) and Program for New Century Excellent Talents in University (NCET-10-0221).

References

[1] R. Xu and D. C. W. II, "Survey of clustering algorithms." *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, 2005.

[2] T. Martinetz and K. Schulten, "Topology representing networks," *Neural Networks*, vol. 7, no. 3, pp. 507–522, 1994.

[3] K.-L. Du, "Clustering: A neural network approach," *Neural Networks*, vol. 23, no. 1, pp. 89–107, 2010.

[4] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[5] T. M. Martinetz, S. G. Berkovitsch, and K. J. Schulten, ""Neural-gas" network for vector quantization and its application to time-series prediction," *IEEE Trans. Neural Netw.*, vol. 4, pp. 558–569, 1993.

[6] B. Fritzke, "A growing neural gas network learns topologies," in *Advances in Neural Information Processing Systems 7*. MIT Press, 1995, vol. 7, pp. 625–632.

[7] T. M. Martinetz, "Competitive hebbian learning rule forms perfectly topology preserving maps," in *The 1993 International Conference on Artificial Neural Networks (ICANN'93)*, 1993, pp. 427–434.

[8] T. Kohonen, *Self-Organizing Maps*. Berlin: Springer, 2001.

[9] P. Mangiameli, S. K. Chen, and D. West, "A comparison of som neural network and hierarchical clustering methods," *European Journal of Operational Research*, vol. 93, no. 3, pp. 402–417, Sep. 1996.

[10] B. Fritzke, "Growing cell structures—a self-organizing network for unsupervised and supervised learning," *Neural Networks*, vol. 7, pp. 1441–1460, 1994.

[11] A. Ocsa, C. Bedregal, and E. Cuadros-Vargas, "DB-GNG: A constructive self-organizing map based on densilty," in *The 2007 International Joint Conference on Neural Networks(IJCNN 2007)*, 2007, pp. 1953–1958.

[12] D. Fišer, J. Faigl, and M. Kulich, "Growing neural gas efficiently," *Neurocomputing*, vol. 104, pp. 72 – 82, 2013.

[13] J. Garcia-Rodriguez, A. Angelopoulou, J. García-Chamizo, A. Psarrou, S. Orts-Escolano, and V. Morell-Gimenez, "Fast autonomous growing neural gas," in *The 2011 International Joint Conference on Neural Networks(IJCNN 2011)*, 2011, pp. 725–732.

[14] Y. Prudent and A. Ennaji, "An incremental growing neural gas learns topologies," in *The 2005 International Joint Conference on Neural Networks(IJCNN 2005)*, vol. 2, 2005, pp. 1211–1216.

[15] S. Furao and O. Hasegawa, "An incremental network for on-line unsupervised classification and topology learning." *Neural Networks*, vol. 19, no. 1, pp. 90–106, 2006.

[16] A. Qin and P. Suganthan, "Robust growing neural gas algorithm with application in cluster analysis," *Neural Networks*, vol. 17, no. 89, pp. 1135 – 1148, 2004.

[17] D. O. Hebb, *The organization of behavior: a neuropsychological theory*. New York: John Wiley & Sons, 1949.