SICE Annual Conference in Sapporo, August 4-6, 2004
Hokkaido Institute of Tecnology, Japan

A Self-structurizing Neural Network for Online Incremental Learning

Osamu Hasegawa[1,2], Furao Shen[2]

1 Tokyo Institute of Technology, 4259, Nagatsuta, Midoriku, 226-8503, Japan
2 PRESTO, Japan Science and Technology Agency, Japan
{hasegawa, furaoshen}@isl.titech.ac.jp

Abstract: An on-line unsupervised learning mechanism is proposed for unlabeled data which is polluted by noises. By using a similarity threshold and local error based insertion criterion, the system is able to grow incrementally and to accommodate input patterns of on-line non-stationary data distribution. The definition of a utility parameter – "error-radius" enables this system to learn the number of nodes needed to solve a current task. The usage of a new technique for removing nodes in low probability density regions can separate the clusters with low-density overlaps and dynamically eliminate the noise in the input data. The design of two-layer neural network makes it possible for this system to represent the topological structure of unsupervised on-line data, report the reasonable number of clusters and give typical prototype patterns of every cluster without any priori conditions such as suitable number of nodes or a good initial codebook.

Keywords: machine learning, neural network, unsupervised learning

1. Introduction

One of the objectives of unsupervised learning is data clustering. Clustering algorithms can be categorized into hierarchical clustering, partitional clustering, artificial neural networks for clustering, and so on [1]. Hierarchical clustering algorithms such as single-link, complete-link and CURE [2] algorithms usually suffer from overload of computation and requirement of much memory space. Thus, they are unsuitable for large data set or on-line data. BIRCH [3] is properly applicable to data sets only consisting of isotropic clusters (i.e., a circle in 2-D, or a sphericity in 3-D, etc.). Specifically, chain-like and concentric clusters are difficult to be identified by BIRCH [2].

The k-means algorithm is one of the most famous partitional clustering algorithms. It suffers from dependency on the initial starting conditions and tendency to result in local minimum. Some new researches [4][5] improves the traditional k-means method in some aspects but still cannot solve the main difficulties: how to determine the number of clusters k in advance, and the limited applicability to only the data set consisting of isotropic clusters.

Another objective of unsupervised learning can be described as topology learning: given some high-dimensional data distribution, find a topological structure that closely reflects the topology of the data distribution. Self-organizing map (SOM) [6] generates mappings from high-dimensional signal spaces to lower-dimensional topological structures. The predetermined structure and size of Kohonen ' s model imply limitations on the resulting mappings [7]. A posterior choice of class labels for the prototypes of the SOM causes furthur problems [8]. As an alternative, a combination of "competitive Hebbian learning" (CHL) [9] and "neural gas" (NG) [10] is an effective method to construct topological structures. But it needs a priori decision about the network size and it has to perform a ranking of all nodes in each adaptation step. "Growing Neural Gas" (GNG) [11] combines the idea of vector quantization with a continuous neighborhood adaptation to solve such problems. The major drawbacks are their permanent increase in the number of nodes and drift of the centers to capture the input probability density [12].

For the much harder problems of non-stationary data distributions, on-line learning tasks, the above-mentioned methods are not suitable. The fundamental issue for such problems is: how can a learning system adapt to new information without corrupting or forgetting previously learned information – the so called Stability-Plasticity Dilemma [13]. By using a utility-based removal criterion, GNG-U [14] deletes nodes which are located in regions of a low input probability density, but the learned old prototype patterns will be destroyed.

Lim and Harrison [15] propose a hybrid network for incremental on-line learning. Hamker [12] proposes an extension to GNG for life-long learning tasks. Both methods can work for some supervised learning, but how to process the unsupervised learning is still a research topic.

In this work, our goal is to design an learning system for on-line unsupervised classification and topology representation tasks. The objective is to develop a network that operates autonomously, on-line, and in non-stationary environment.

PR0002/04/0000-2063 ¥400

2. Proposed method

In summary, the targets of the proposed algorithm are:

- To process the on-line non-stationary data.
- Without any priori condition such as suitable number of nodes or a good initial codebook or how many classes there are, to do the unsupervised learning: report a suitable number of classes; represent the topological structure of the input probability density.
- To separate the classes with some low-density overlaps and detect the main structure of clusters that are polluted by noises.

To realize these targets, we emphasize that an insertion of new nodes, a similarity threshold and a deletion of low probability density nodes are key aspects.

2.1 Overview of proposed method

we adopt a two-level neural network structure to realize our targets. The first-level will be used to generate the topological structure of input pattern. In the second-level, the nodes identified by first-level will be used as new input data set. And second-level will report the number of clusters and give typical prototype nodes of every cluster.

For the unsupervised classification task, we need to determine if an input sample belongs to a previously learned clusters or belongs to a new cluster never seen before. We set the input signal as a new node (the first node of a new cluster) when the distance between the signal and the nearest node (or the second nearest node) is larger than a threshold T. In first-level, T is permanently adapted to the present situation. In second-level, we calculate the typical within-cluser distance and typical between-cluser distance based on those nodes generated in first-level, and then give a constant threshold distance T_c according to the within-cluster distance and between-cluster distance.

To represent topological structure, in on-line learning tasks, insertion of new nodes is an important feature to decrease the error of the task and to adapt to changing environments while preserving old prototype patterns. But insertion must be stopped to prohibit a permanent increase in the number of nodes and to avoid overfitting. For within-cluster insertion, we adopt the scheme as some increment networks (such as GNG) used, to insert a node between the node q with maximum accumulated error and node f which is among the neighbors of q with maximum accumulated error. Current increment networks do not have the ability to learn whether a further insertion of node is useful or not. The insertion of nodes will lead to catastrophic allocation of new nodes. In this paper, we evaluate the insertion by a utility parameter "error-radius" to judge if the insertion is successful. This strategy assures that the insertion of new node will lead to decreasing of error and it automatically controls the increment of nodes.

In general, there are overlaps between clusters. To detect the number of clusters more precisely, we suppose the input data is separable, it means the overlaps between clusters will have lower probability density than centric part of clusters. Fritzke [7] designed a coarse estimation method to find the low probability density region, if the density is below some threshold η, the nodes in that region will be removed. This method suffers from difficulties such as computational load of estimating the probability density and how to determine the threshold η. The GNG-U [14] uses a utility parameter to delete low density nodes but it will completely destroy the old learned prototype patterns. In this paper, we propose a novel strategy: if the number of input signals generated so far is an integer multiple of a parameter, remove those nodes with no or only one topological neighbor. The idea is based on this consideration, if the node has no or only one neighbor, it means after a period of learning, the accumulated error of this node has gotten very low chance to be the maximum and insertion of new nodes near this node is difficult, i.e., the probability density of the region that the node lies in is very low. This strategy works well for removing nodes in low density regions without additive computation load and avoids the usage of special parameters. Also, the usage of this technique is able to periodically remove nodes caused by noises for the reason that the probability density of noises will be low.

2.2 Complete Algorithm

This algorithm will be used for both the first-level and second-level neural networks. The difference between two levels is the input data set of second-level will be the nodes generated by first-level, and a constant threshold T_c will be used in second-level instead of the adaptive threshold T used in first-level.

Notations to be used in the algorithm

A : node set, which will be used to store nodes.
N_A : number of nodes in A.
C : edge set, which will be used to store connections between nodes.
N_C : number of edges in C.
W_i : n-dimension weight vector of the node i.
N_i : set of direct topological neighbors of node i.
L_i : number of topological neighbors of node i.
$age_{(i,j)}$: age of edge that connects node i and node j.
E_i : local accumulated error of node i.
M_i : local accumulated number of node i.
T_i : similarity threshold of node i.
Q : number of clusters.
C_i : cluster label of node i.
R_i : inherited error-radius of node i. "error-radius" is defined by the mean of accumulated error E_i/M_i. R_i serves as memory for the error-radius of node i at the moment of insertion.
path : If there are a series of nodes $x_i \in A$,

$i = 1,2,\ldots,n$, makes (i,x_1), (x_1,x_2),..., (x_{n-1},x_n), $(x_n,j) \in C$. We say there is a "path" between node i and node j.

Basic Algorithm

1. Initialize the node set A to contain two nodes c_1, c_2 with weight vectors chosen randomly from the input pattern. Initialize the edge set C to empty set.

2. Input a new pattern $\xi \in R^n$.

3. Search the node set A to determine the winner s_1, and second winner s_2 by

$$s_1 = \arg\min_{c\in A} ||\xi - W_c|| \quad (1)$$

$$s_2 = \arg\min_{c\in A\backslash\{s_1\}} ||\xi - W_c|| \quad (2)$$

If the distance between ξ and s_1 or s_2 is greater than similarity threshold T_{s_1} or T_{s_2} respectively, the input signal will be a new node, add it to A and go to step 2) to process the next signal. I.e., if $||\xi - W_{s_1}|| > T_{s_1}$ or $||\xi - W_{s_2}|| > T_{s_2}$, then $A = A \cup r$ and $W_r = \xi$.

4. If a connection between s_1 and s_2 does not exist already, create it and add it to the edge set C. Set the age of the edge between s_1 and s_2 to zero.

5. Increment the age of all edges emanating from s_1

$$age_{(s_1,i)} = age_{(s_1,i)} + 1 \quad (\forall i \in N_{s_1}) \quad (3)$$

6. Add the Euclidian distance between the input signal and the winner to local accumulated error E_{s_1}

$$E_{s_1} = E_{s_1} + ||\xi - W_{s_1}|| \quad (4)$$

7. Add 1 to local accumulated number of signals M_{s_1}

$$M_{s_1} = M_{s_1} + 1 \quad (5)$$

8. Adapt the weight vectors of the winner and its direct topological neighbors by fraction $\epsilon_1(t)$ and $\epsilon_2(t)$ respectively

$$\Delta W_{s_1} = \epsilon_1(t)(\xi - W_{s_1}) \quad (6)$$

$$\Delta W_i = \epsilon_2(t)(\xi - W_i) \quad (\forall i \in N_{s_1}) \quad (7)$$

Here, we call $\epsilon_1(t)$ the learning rate of winner, and $\epsilon_2(t)$ the learning rate of neighbor.

9. Remove edges with an age larger than a predefined threshold age_d. If this results in nodes having no more emanating edges, remove those nodes as well. I.e., if $(i,j) \in C$, and $age_{(i,j)} > age_d$ $(\forall i,j \in A)$, then $C = C\backslash\{(i,j)\}$; if $L_i = 0$ $(\forall i \in A)$, then $A = A\backslash\{i\}$.

10. If the number of input signals generated so far is an integer multiple of a parameter λ, insert a new node and remove low probability density nodes as follows:

- Determine the node q with the maximum accumulated error E. I.e., $q = \arg\max_{c\in A} E_c$.
- Determine among the neighbors of q the node f with the maximum accumulated error. I.e., $f = \arg\max_{c\in N_q} E_c$.
- Add a new node r to the network and interpolate its weight vector from q and f

$$A = A \cup \{r\},\ W_r = (W_q + W_f)/2.0 \quad (8)$$

- Interpolate the accumulated error E_r, accumulated number of signal M_r and inherited error-radius R_r from E_q, E_f , M_q, M_f and R_q, R_f

$$E_r = \alpha_1(E_q + E_f) \quad (9)$$

$$M_r = \alpha_2(M_q + M_f) \quad (10)$$

$$R_r = \alpha_3(R_q + R_f) \quad (11)$$

- Decrease the accumulated error variables of q and f by

$$E_q = \beta E_q,\quad E_f = \beta E_f \quad (12)$$

- Decrease the accumulated number of signal variables of q and f by

$$M_q = \gamma M_q,\quad M_f = \gamma M_f \quad (13)$$

- Judge whether the insertion is successful or not. If the error-radius is larger than the inherited error-radius R_i $(\forall i \in \{q,r,f\})$, i.e., the insertion is not able to decrease the mean error of this local area, the insertion is not successful; else, update the inherited error-radius. I.e., if $E_i/M_i > R_i$ $(\forall i \in \{q,r,f\})$, the insertion is not successful, the new node r will be removed from set A, and all parameters will be restored; else, $R_q = E_q/M_q$, $R_f = E_f/M_f$, and $R_r = E_r/M_r$.
- If the insertion is successful, insert edges connecting the new node r with nodes q and f, and remove the original edge between q and f. I.e., $C = C \cup \{(r,q),(r,f)\}$, $C = C\backslash\{(q,f)\}$.
- For all nodes in A, search for the nodes only have one neighbor, then remove them. I.e., if $L_i = 1$ $(\forall i \in A)$, then $A = A\backslash\{i\}$.
- For all nodes in A, search for the isolated nodes, then delete them. I.e., if $L_i = 0$ $(\forall i \in A)$, then $A = A\backslash\{i\}$.

11. After a long constant time period LT, report the number of clusters, output all the nodes belong to different clusters. The following method will be used to classify the nodes to different classes.

- Initialize all nodes as unclassified.
- LOOP: Randomly choose one unclassified node i from the node set A. Mark node i as classified and label it as class C_i.
- Search A to find all unclassified nodes that are connected to node i by a "path". Mark these nodes as classified and label them as the same class as node i (C_i).
- If there are nodes unclassified, go to LOOP to go on the classification process until all nodes are classified.

12. Go to step 2) to go on the on-line unsupervised learning process.

2.3 Parameter discussion

There are some parameters in the algorithm. Here we will discuss how to determine such parameters.

1. Similarity Threshold T_i of node i
   In step 3) of basic algorithm, threshold T_i will be used to judge if the input signal belongs to the previously learned clusters or not. For first-level, we have no priori knowledge of input data and adopt an adaptive threshold scheme. At first, we suppose all data come from one same cluster, thus the initial threshold of every node will be $+\infty$. After a period of learning, the input pattern will be separated to different small groups and each group is composed of one node and its direct topological neighbors. The similarity threshold must be greater than within-cluster distances and less than the between-cluster distances. With this idea, we define the similarity threshold T_i of node i by the following method.

   (a) Initialize the similarity threshold T_i of node i to $+\infty$.

   (b) When node i is winner or second winner, update the similarity threshold T_i by

   - If the node has some direct topological neighbors ($L_i > 0$), the T_i will be updated as the maximum distance between the node i and all its neighbors.

$$T_i = \max_{c \in N_i} ||W_i - W_c|| \qquad (14)$$

   - If the node i has no neighbors ($L_i = 0$), the T_i will be updated as the minimum distance of the node i and all other nodes in A.

$$T_i = \min_{c \in A \setminus \{i\}} ||W_i - W_c|| \qquad (15)$$

   For second-level, the input data set is the results of first-level. After the learning of first-level, we get a coarse clustering results and topology structure. With this knowledge, we can calculate the within-cluster distance d_w by

$$d_w = \frac{1}{N_C} \sum_{(i,j) \in C} ||W_i - W_j|| \qquad (16)$$

and between-cluster distance $d_b(C_i, C_j)$ of cluster C_i and C_j by

$$d_b(C_i, C_j) = \min_{i \in C_i, j \in C_j} ||W_i - W_j|| \qquad (17)$$

I.e., the within-cluster distance will be the mean distance of all edges, and the between-cluster distance will be the minimum distance between two clusters. With d_w and d_b, we can give a constant threshold T_c for all nodes. The threshold must be greater than within-cluster distance and less than between-cluster distance. Influenced by the overlap or noise, some between-cluster distances will be less than within-cluster distance. We use the following method to calculate the threshold distance T_c for second-level.

   (a) Set T_c as minimum between-cluster distance.

$$T_c = d_b(C_{i_1}, C_{j_1}) \qquad (18)$$
$$= \min_{k,l=1,\ldots,Q,k \neq l} d_b(C_k, C_l) \qquad (19)$$

   (b) If T_c is less than within-cluster distance, set T_c as the next minimum between-cluster distance.

$$T_c = d_b(C_{i_2}, C_{j_2}) \qquad (20)$$
$$= \min_{k,l=1,\ldots,Q,k \neq l,k \neq i_1,l \neq j_1} d_b(C_k, C_l) \qquad (21)$$

   (c) Go to step b) until T_c is greater than d_w.

2. Adaptive learning rate
   We adopt the scheme like k-means method to adapt the learning rate over time by

$$\epsilon_1(t) = \frac{1}{t}, \quad \epsilon_2(t) = \frac{1}{100t} \qquad (22)$$

   Here, the time parameter t stands for the number of input signals for which this particular node has been winner so far, i.e. $t = M_i$. This algorithm is known as k-means, the node is always the exact arithmetic mean of the input signals it has been winner for. The reason for adoption of this scheme is that we hope to make the position of the node more stable by decreasing the learning rate when the node becomes winner for more and more input patterns.

3. Decreasing rate of the accumulated variables E (error) and M (number of signals)

   In Fig.1, the Voronoi regions of node q and f before insertion are shown in left part, and right part shows the Voronoi regions belong to q, f and new

node r after insertion. We suppose the signals in these Voronoi regions are uniformly distributed. Comparing left part with right part, we find there are 1/4 accumulated number of signals of q and f reallocated to new node r and there are 3/4 accumulated number of signals remained for q and f. For the accumulated error, it is reasonable to suppose the error caused by $V1$ is double of the error caused by $V2$ for node q. Thus, after the insertion, the accumulated error of q and f will be 2/3 of the accumulated error before insertion. We also suppose the error to r caused by $V1$ is the same of error to q caused by $V2$, so the reallocated accumulated error for r will be 1/6 of the E_q and E_f . With the above analysis, the decreasing rate will be: $\alpha_1 = 1/6$, $\alpha_2 = 1/4$, $\alpha_3 = 1/4$, $\beta = 2/3$, and $\gamma = 3/4$.

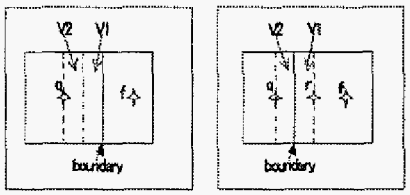

Fig.1: Illustration of decreasing rate

3. Experiment

We perform our experiment on the data set shown in Fig.2. An artificial 2D data set is utilized to take advantage of its intuitive manipulation, visualization and its resulting insight into the behavior of the system. The data set is separated to five parts named A, B, C, D and E. A is a data set satisfied two-dimensional Gaussian distribution. B is a circular area. C and D data set is a famous single-link example. E is a shape like sinusoid and we separate the E to E1, E2 and E3 to illustrate the incremental process clearly. We also add some random noises (10% of the useful data) to the data set to simulate real world data. In Fig.2, there are overlaps between clusters, and noises are distributed on the whole data set. As stated in [2], both algorithm BIRCH and single-link clustering cannot partition such data set correctly.

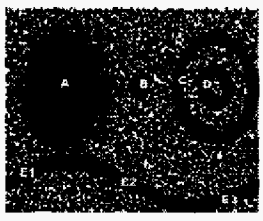

Fig.2: Data Set

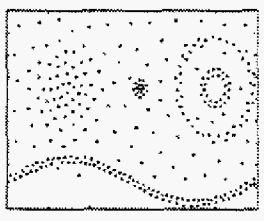

Fig.3: GNG: 1 cluster

In this experiment, we will compare our proposed method with a typical increment network–GNG to show the advantage of our method. In all experiments, we set the parameter $\lambda = 100$, and $age_d = 100$. For GNG, the maximum number of nodes is predefined as 300.

3.1 Experiment in stationary environment

At first, we test Fig.2 as a stationary data set. 100,000 patterns are randomly chosen from all areas A, B, C, D and E. The topological results of GNG and proposed method are shown in Fig.3, Fig.4 and Fig.5. For stationary environment, GNG can represent the topological structure, but it is affected by noises and all nodes linked together to form one big cluster. The first-level of proposed method partially eliminates the influence of noises and separates the data set to some different clusters (Fig.4). The second-level of proposed method efficiently represents the topology structure and gives the number of clusters and the typical prototype nodes of every cluster (Fig.5).

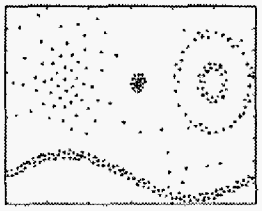

Fig.4: Proposed: First-Level Stationary Environment

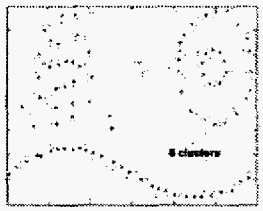

Fig.5: Proposed: Second-Level

3.2 Experiment in NON-Stationary environment

Then we simulate the on-line learning process by employing a paradigm as follows: from step 1 until 20,000, patterns are randomly chosen from area A. At step 20,001 the environment changes and patterns from area B are chosen. At step 40,001 the environment changes again, etc. Table1 gives the detail specification of test environment. Environment changes from I to VII. In each environment, the areas to be used to generate patterns are marked with '1', other areas are marked with '0'. For every environment, we add 10% noises to the test data and the noises are distributed on whole data space.

Under this on-line Non-stationary environment, GNG cannot represent the topological structure well (Fig.6). GNG-U will delete all old learned patterns and only represent the structure of new input patters (Fig.7). Both methods cannot get rid of noises. For example, in the GNG-U results, the nodes beyond the area E3 are all caused by noises that are distributed on the whole space.

Table 1: Experiment environment for on-line learning

Area	Environment						
	I	II	III	IV	V	VI	VII
A	1	0	1	0	0	0	0
B	0	1	0	1	0	0	0
C	0	0	1	0	0	1	0
D	0	0	0	1	1	0	0
E1	0	0	0	0	1	0	0
E2	0	0	0	0	0	1	0
E3	0	0	0	0	0	0	1

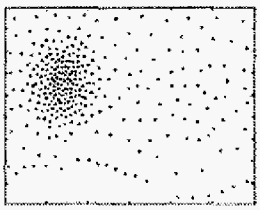

Fig.6: GNG

Fig.7: GNG-U

Fig.8 is the first-level results of our proposed method. After the learning under one environment, we report the intermediate topological structures. Environment I tests 2D Gaussian distribution. If the probability changes to zero in A region (Environment II from 20,001 to 40,000 steps), those remaining nodes of area A, often called "dead nodes", play a major role in on-line learning. They preserve the knowledge of previous situations for future decision. In the future, the reappearance of area A (Environment III, 40,001 to 60,000 steps) does not raise the error and the knowledge is completely preserved, thus nearly no insertion happened and most of the nodes remain at their positions. Environment III and Environment IV test for the concentric circular data sets (area C and area D), the system removes noises between the areas and separate C and D from each other. Environment V, VI, and VII test a complicated artificial shape (area E). E is separated to three parts and the data come to system sequentially. The nodes of area E increasing following the change of environment from V to VI and VII, but all nodes linked together to form one same class.

Then Fig.8 is used as the input data set for second-level, and a constant threshold is calculated. Fig.9 is the second-level results. It reports the number of clusters and gives the prototype nodes for every cluster.

Fig.4 and Fig.8 show that the first-level can detect main structure from original data which is polluted by noises. It can control the number of nodes needed for the current task. Fig.5 and Fig.9 show that second-level can not only report the number of clusters precisely but can remove reduntant nodes and re-arrange position of nodes to represent the topology more efficiently. We say after the second-level, the system can "understand" the original raw input data very well.

Fig.10 is the illustration of how the number of nodes changes during the on-line learning in first-level. When the input signal coming from a new area happens (see Table1, environment changes from I to VII), the number of nodes will increase. In the same environment, after some learning steps, the increase of nodes will stop and converge to constant. It is for the reason that a further insertion cannot lead to error decreasing. The noise will lead to the system frequently insert and delete nodes, thus, there are small fluctuate of number of nodes in Fig.10.

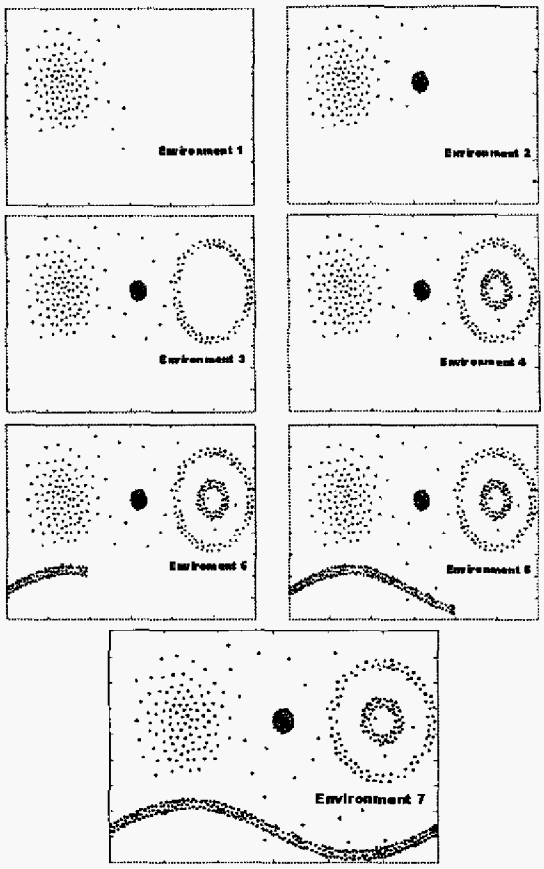

Fig.8: Proposed: first-level, NON-STATIONARY Environment

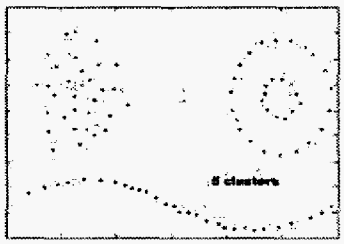

Fig.9: Proposed: second-level, NON-STATIONARY Environment

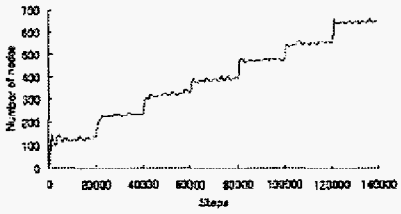

Fig.10: Number of nodes during on-line learning (Environment I – Environment VII)

4. Conclusion

In this paper, we proposed a new on-line learning method for unsupervised classification and topology representation. The combination of similarity threshold and a local accumulated error makes the algorithm

fit for non-stationary data distribution. A novel on-line criterion for removals of nodes is proposed to classify the data set well and eliminate noises periodically. The usage of a utility parameter "error-radius" is able to judge if the insertion is successful and control the increasing of nodes. The adoption of two-level neural network makes it possible for this system to "understand" the original data set very well. Summarizing, the algorithm is able to cope with some difficulties of online unsupervised learning, such as overlaps, never seen inputs, temporarily not appearing patterns, and noises.

References

[1] M.N. Murty, A.K. Jain, P.J. Flynn, "Data Clustering: a Review," *ACM Comput. Surv.*, Vol. 31, No.3 pp. 264-323, 1999.

[2] S. Guha, R. Rastogi, K. Shim, "CURE: An Efficient Clustering Algorithm for Large Databases," *Proceeding of ACM SIGMOD*, pp. 73-84, 1998.

[3] T. Zhang, R. Ramakrishnan, M. Livny, "BIRCH: An Efficient Data Clustering Method for Very Large Database," *Proc. ACM SIGMOD Conf. on Management of Data*, pp. 103-114, 1996.

[4] A. Likas, N. Vlassis, J.J. Verbeek, "The Global k-means Clustering Algorithm," *Pattern Recognition*, Vol. 36, pp. 451-461, 2003.

[5] G. Patane, M. Russo, "The Enhanced LBG Algorithm," *Neural Networks*, Vol. 14, pp. 1219-1237, 2001.

[6] T. Kohonen, "Self-organized Formation of Topologically Correct Feature Maps," *Biological Cybernetics*, Vol 43, pp. 59-69, 1982.

[7] B. Fritzke, "Growing Cell Structures – a Self-organizing Network for Unsupervised and Supervised Learning," *Neural Networks*, Vol. 7, pp. 1441-1460, 1994.

[8] T. Villmann, F.-M. Schleif, B. Hammer, "Supervised Neural Gas and Relevance Learning in Learning Vector Quantization," *Proc. WSOM*, Japan, 2003.

[9] T.M. Martinetz, "Competitive Hebbian Learning Rule Forms Perfectly Topology Preserving Maps," *ICANN*, pp. 427-434, 1993.

[10] T.M. Martinetz, S.G. Berkovich, K. J. Schulten, " 'Neural-gas' Network for Vector Quantization and Its Application to Time-series Prediction," *IEEE Transactions on Neural Networks*, Vol. 4, No. 4, pp. 558-569, 1993.

[11] B. Fritzke, "A Growing Neural Gas Network Learns Topologies", In *Advances in neural information processing systems*, pp. 625-632, 1995.

[12] F.H. Hamker, "Life-long Learning Cell Structures – Continuously Learning without Catastrophic Interference," *Neural Networks*, Vol. 14, pp. 551-573, 2001.

[13] G.A. Carpenter, S. Grossberg, "The ART of Adaptive Pattern Recognition by a Self-organizing Neural Network," *IEEE Computer*, Vol. 21, pp. 77-88, 1988.

[14] B. Fritzke, "A Self-organizing Network that can Follow Non-stationary Distributions," *In Proceedings of ICANN-97*, pp. 613-618, 1997.

[15] C.P. Lim, R.F. Harrison, "A Incremental Adaptive Network for On-line Supervised Learning and Probability Estimation," *Neural Networks*, Vol. 10, pp. 925-939, 1997.