An Incremental Neural Network for Non-stationary Unsupervised Learning

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

[1] Tokyo Institute of Technology, Yokohama, Japan
furaoshen@isl.titech.ac.jp
[2] PRESTO, Japan Science and Technology Agency (JST)

Abstract. A new online learning mechanism is proposed for unsupervised classification and topology representation. It can represent the topological structure of unsupervised online non-stationary data, report the reasonable number of clusters, eliminate noise, and give typical prototype patterns of every cluster without priori conditions such as a suitable number of nodes or a good initial codebook.

1 Introduction

Traditional clustering methods such as k-means [1] and LBG [2] suffer from the dependence on initial starting conditions and predetermining of number of clusters. Learning of topology structure for nonstationary data distribution is difficult for some self-organizing neural networks such as self-organizing map [3] and neural gas [4] for the use of decaying adaptation parameters will lead to "frozen network". Growing neural gas (GNG) [5] suffers from the permanent increase in number of nodes and GNG-U [6] will destroy the learned old prototype patterns. Hamker's life-long learning model [7] only suits for some supervised learning tasks.

In this work, our goals are: (1) to process online and nonstationary data. (2) without priori conditions such as a suitable number of nodes or a good initial codebook or knowing how many classes there are, to conduct unsupervised learning, report a suitable number of classes and represent the topological structure of input probability density. (3) to separate classes with low-density overlap and detect the main structure of clusters that are polluted by noise.

2 Proposed Algorithm

Suppose we say that two samples belong to the same cluster if the Euclidean distance between them is less than threshold distance T. To obtain "natural" clusters, T must be greater than typical within-cluster distance and less than typical between-cluster distance [8].

For within-cluster insertion, we insert a node between node q with maximum accumulated error and node f, which is among the neighbors of q with maximum accumulated error. To ensure insertion of a new node leads to a decrease in error,

N.R. Pal et al. (Eds.): ICONIP 2004, LNCS 3316, pp. 641–646, 2004.

and to avoid catastrophic allocation of new nodes, when a new node is inserted, we evaluate insertion by a utility parameter to judge if insertion is successful.

We separate clusters by removing nodes located in a region with low probability density. 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 the consideration that if a node has no or only one neighbor, it means during a period, the accumulated error of that node has very low chance to become maximum and the insertion of new nodes near that node is difficult, i.e., the probability density of the region the node lies in is very low. Unlike some other techniques in [9] and [6], this proposed strategy works well for removing nodes in low density regions without additive computation load and avoids the use of special parameters. In addition, this technique periodically removes nodes caused by noise.

Algorithm 2.1: Proposed algorithm

1. Initialize node set A to contain two nodes, c_1 and c_2 with weight vectors chosen randomly from the input pattern. Initialize connection set C, $C \subset A \times A$, to the empty set.
2. Input new pattern $\xi \in R^n$.
3. Search the nearest node (winner) s_1, and the second-nearest node (second winner) s_2 by $s_1 = \arg\min_{c \in A} \|\xi - W_c\|$, $s_2 = \arg\min_{c \in A \setminus \{s_1\}} \|\xi - W_c\|$. If the distance between ξ and s_1 or s_2 is greater than similarity threshold T_{s_1} or T_{s_2}, the input signal is a new node, add it to A and go to Step2 to process the next signal. The threshold T is calculated by *Algorithm 2.2*.
4. If a connection between s_1 and s_2 does not exist, create it. Set the age of the connection between s_1 and s_2 to zero.
5. Increment the age of all edges emanating from s_1 by 1.
6. Add Euclidian distance between input signal ξ and the winner to a local accumulated error E_{s_1}, $E_{s_1} = E_{s_1} + \|\xi - W_{s_1}\|$.
7. Add 1 to a local accumulated number of signals M_{s_1}, $M_{s_1} = M_{s_1} + 1$.
8. Adapt the weight vectors of the winner and its direct topological neighbors by fraction $\epsilon_1(t)$ and $\epsilon_2(t)$ of the total distance to the input signal,
   $\Delta W_{s_1} = \epsilon_1(t)(\xi - W_{s_1})$
   $\Delta W_i = \epsilon_2(t)(\xi - W_i)$ for all direct neighbors i of s_1
   We adopt a scheme like k-means to adapt the learning rate over time by $\epsilon_1(t) = 1/t$ and $\epsilon_2(t) = 1/100t$.
9. Remove edges with an age greater than a predefined threshold age_{dead}. If this results in nodes having no more emanating edges, remove them as well.
10. If the number of input signals generated so far is an integer multiple of parameter λ, insert a new node as follows:
    (a) Determine node q with maximum accumulated error E.
    (b) Determine among neighbors of q node f with maximum accumulated error.
    (c) Add new node r to the network and interpolate its weight vector from q and f, i.e., $W_r = (W_q + W_f)/2.0$.

(d) Interpolate 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 by $E_r = (E_q + E_f)/6.0$, $M_r = (M_q + M_f)/4.0$, and $R_r = (R_q + R_f)/4.0$. Here, error-radius of node i is defined by 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. It is updated at each insertion, but only for affected nodes.
(e) Decrease accumulated error of q and f by $E_q = 2E_q/3.0$, $E_f = 2E_f/3.0$.
(f) Decrease accumulated number of signal of q and f by $M_q = 3M_q/4.0$, $M_f = 3M_f/4.0$.
(g) Judge whether insertion is successful or not. If the error-radius is larger than inherited error-radius R_i ($\forall i \in \{q, r, f\}$), i.e., insertion is not able to decrease the mean error of this local area, insertion is not successful; else, update the inherited error-radius, i.e., if $E_i/M_i > R_i$ ($\forall i \in \{q, r, f\}$), insertion is not successful, new node r is removed; else, $R_q = E_q/M_q$, $R_f = E_f/M_f$, and $R_r = E_r/M_r$.
(h) If insertion is successful, insert edges connecting new node r with nodes q and f, and remove the original edge between q and f.
(i) For all nodes in A, search for nodes having no neighbor or only one neighbor, then remove them.

11. After long constant time period LT, use *Algorithm 2.3* to classify nodes into different clusters. Report the number of clusters, output all nodes belonging to different clusters.
12. Go to Step2 to continue unsupervised online learning.

The similarity threshold T (in *Algorithm 2.1*, step3) must be greater than within-cluster distance and less than between-cluster distance. Based on this idea, we propose *Algorithm 2.2* to calculate similarity threshold T_i.

Algorithm 2.2: Calculation of similarity threshold T

1. Initial the similarity threshold of node i to $+\infty$ when node i is generated as a new node.
2. When node i is a winner or second winner, update similarity threshold T_i:
   - If the node has direct topological neighbors, T_i is updated as the maximum distance between node i and all of its neighbors, $T_i = \max_{c \in N_i} \|W_i - W_c\|$, N_i is the neighbor set of node i.
   - If node i has no neighbors, T_i is updated as the minimum distance of node i and all other nodes in A, $T_i = \min_{c \in A \setminus \{i\}} \|W_i - W_c\|$.

If there is 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.

Algorithm 2.3: Classify nodes to different clusters

1. Initialize all nodes as unclassified.
2. Randomly choose one unclassified node i from node set A. Mark node i as classified and label it as class C_i.
3. Search A to find all unclassified nodes connected to node i with a "path." Mark these nodes as classified and label them as the same class as node i.
4. If there are unclassified nodes, go to Step2 to continue classification until all nodes are classified.

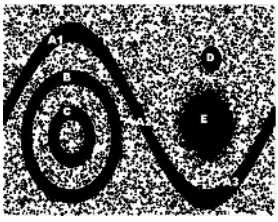

Fig. 1. Original data set

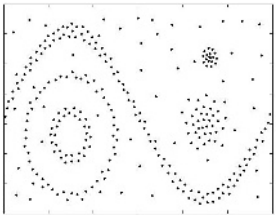

Fig. 2. GNG results

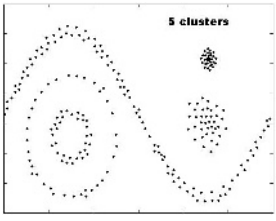

Fig. 3. Proposed results

3 Experiment

We conducted our experiment on the data set shown in Fig. 1. An artificial 2-D data set is used to take advantage of its intuitive manipulation, visualization, and resulting insight into system behavior. The data set is separated into five parts, i.e., A, B, C, D, and E. A is shaped like a sinusoid and separated into A1, A2, and A3 to show incremention property. The B and C data set is a famous single-link example. D is a circular area. E is a data set satisfied 2-D Gaussian distribution. We add random noise (5% of useful data) to the data set to simulate real-world data. In Fig. 1, there are overlaps between clusters, and noise is distributed over the whole data set.

In this experiment, we compare our proposed method with GNG to show the advantage of our method. In all experiments, we set parameter $\lambda = 100$, and $age_{dead} = 100$. For GNG, the maximum number of nodes is predefined as 300.

3.1 Experiment in Stationary Environment

First, we use Fig. 1 as a stationary data set, 100,000 patterns are randomly chosen from areas A, B, C, D, and E. Topological results of GNG and proposed method are shown in Fig. 2 and Fig. 3. For stationary data, GNG can represent the topological structure, but it is affected by noise and all nodes are linked to form one cluster. The proposed method efficiently represents the topology structure and gives the number of clusters and typical prototype nodes of every cluster.

3.2 Experiment in Nonstationary Environment

We simulate online learning by using a paradigm as follows: from step 1 until 20,000, patterns are randomly chosen from area A1. At step 20,001, the environment changes and patterns from area A2 are chosen. At step 40,001, the environment changes again, etc. Table 1 details specifications of the test environment. The environment changes from I to VII. In each environment, areas used to generate patterns are marked with "1," and other areas are marked with "0." For every environment, we add 5% noise to test data and noise is distributed over the whole data space.

The results for GNG and GNG-U in Fig. 4 and Fig. 5 show that GNG cannot represent the topological structure of online nonstationary data well, GNG-U

deletes all old learned patterns and only represents the structure of new input patterns. Neither method eliminates noise. In GNG-U results, for example, nodes beyond area E are all caused by noise distributed over the whole space (Fig. 5).

Table 1. Nonstationary

Area	Environment						
	I	II	III	IV	V	VI	VII
A1	1	0	1	0	0	0	0
A2	0	1	0	1	0	0	0
A3	0	0	1	0	0	1	0
B	0	0	0	1	1	0	0
C	0	0	0	0	1	0	0
D	0	0	0	0	0	1	0
E	0	0	0	0	0	0	1

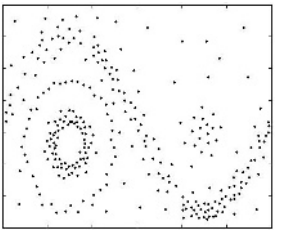

Fig. 4. GNG results

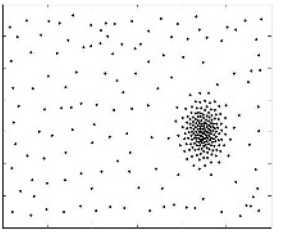

Fig. 5. GNG-U results

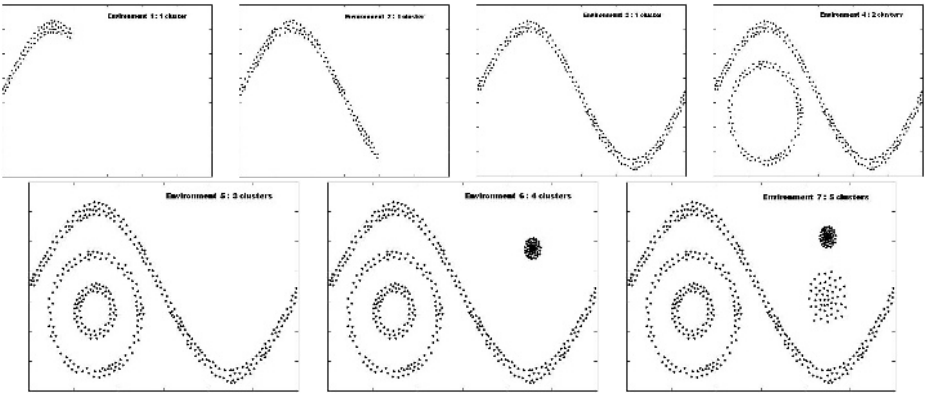

Fig. 6. Nonstationary environments, results of proposed method

Fig. 6 shows the results of proposed method. After learning in one environment, we report intermediate topological structures. Environments I, II, and III test a complicated artificial shape (area A). A is separated into three parts and data comes to the system sequentially. We find nodes of area A increasing following the change of environment from I to II and III, but all nodes linked to form the same class. In environment I, the data set of area A1 is tested. If probability changes to zero in some regions, such as in environment II, remaining nodes of area A1 preserve the knowledge of previous situations for future decisions. In the future, the reappearance of area A1 (environment III) does not raise error and knowledge is completely preserved, so nearly no insertion happens and most nodes remain at their positions. Environments IV and V test for a difficult situation (data sets such as areas B and C). The system also works well, removing noise between areas and separating the B and C. Environments VI and VII test an isolated data set: area D (circular) and area E (Gaussian distribution).

Fig. 3 and Fig. 6 show that the proposed method processes stationary and nonstationary environments well. It detects the main structure from original data, which is polluted by noise. It controls the number of nodes needed for representation of topology and reports the number of clusters.

Fig. 7 shows how the number of nodes changes during online learning. When an input signal comes from a new area (see Table 1, environment changes from I to VII), the number of nodes increases. In the same environment, after some learning steps, the increase in nodes stops and converges to a constant because further insertion cannot lead to decreasing of error. Noise leads to the system frequently inserting and deleting nodes; thus, there is a small fluctuation in the number of nodes in Fig. 7.

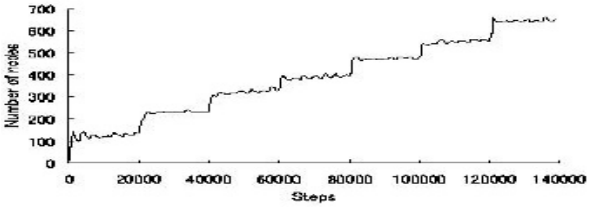

Fig. 7. Number of nodes during online learning

In addition, we tested up to 5000-dimensional non-linear artificial data set and the proposed method worked very well. For real world data, we tested digits data set from Japan ETL6 handwriting character database, it is composed of digits from '0' to '9', and there are 1,383 samples for every digit. The system reported that there were 10 classes, and then the prototype patterns of every class were also reported. Due to lack of space, we can not report the detail of the experiment for real data in this paper.

References

1. Likas, A., Vlassis, N., & Verbeek, J.J. (2003). "The global k-means clustering algorithm," Pattern Recognition, Vol. 36, 451-461.
2. Patane, G., & Russo, M. (2001). "The enhanced LBG algorithm," Neural Networks, Vol.14, 1219-1237.
3. Kohonen, T. (1982). "Self-organized formation of topologically correct feature maps," Biological Cybernetics, Vol 43, 59-69.
4. Martinetz, T.M., Berkovich, S.G., & Schulten, K.J. (1993). ""Neural-gas" network for vector quantization and its application to time-series prediction," IEEE Transactions on Neural Networks, Vol. 4, No. 4, 558-569.
5. Fritzke, B. (1995). "A growing neural gas network learns topologies," In Advances in neural information processing systems, 625-632.
6. Fritzke, B. (1997). "A self-organizing network that can follow non-stationary distributions," In Proceedings of ICANN-97, 613-618.
7. Hamker, F.H. (2001). "Life-long learning cell structures – continuously learning without catastrophic interference," Neural Networks, Vol. 14, 551-573.
8. Duda, R.O., Hart, P.E., & Stork, D.G. (2001). "Pattern classification, 2nd ed.," A Wiley-Interscience Publication.
9. Fritzke, B. (1994). "Growing cell structures – a self-organizing network for unsupervised and supervised learning," Neural Networks, Vol. 7, 1441-1460.