Paper:

An Incremental Neural Network for Online Supervised Learning and Topology Learning

Youki Kamiya*, Shen Furao, and Osamu Hasegawa****

*Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology
R2-52, 4259 Nagatsuta, Midori-ku, Yokohama 226-8503, Japan
**Imaging Science and Engineering Lab., Tokyo Institute of Technology
***PRESTO, Japan Science and Technology Agency (JST)
E-mail: {youki, furaoshen, hasegawa}@isl.titech.ac.jp
[Received January 30, 2006; accepted May 19, 2006]

A new self-organizing incremental network is designed for online supervised learning. During learning of the network, an adaptive similarity threshold is used to judge if new nodes are needed when online training data are introduced into the system. Nodes caused by noise are deleted to decrease the misclassification. The proposed network, which is robust to noisy training data, suits the following tasks: (1) online or even life-long supervised learning; (2) incremental learning, i.e., learning new information without destroying old learned information; (3) learning without any predefined optimal condition; (4) representing the topology structure of inputting online data; and (5) learning the number of nodes needed to represent every class. Experiments of artificial data and high-dimension real-world data show that the proposed method can achieve classification with a high recognition ratio, high speed, and low memory.

Keywords: self-organizing, incremental, online supervised learning, topology learning

1. Introduction

Supervised learning is also referred to as discriminant analysis or supervised grouping. In supervised learning, a classifier is used to discriminate between members and non-members of a given class.

Many methods are suitable for supervised learning. Support Vector Machines (SVMs) produce stable and high-quality output [1], but they are strongly dependent on the selection of a kernel function [2]; ADA Boosting [3] requires advance knowledge of the base classifier. For classification tree methods such as CART [4], splits are increasingly created as the tree expands, and interpretation would necessarily be more complex. Linear Discriminant Analysis (LDA) [5] maximizes the ratio of between-class variance to the within-class variance in any particular dataset, thereby guaranteeing maximal separability, but it is only suitable to linear discriminant tasks. Learning vector quantization (LVQ) [6] is adopted in a supervised learning technique for the self-organizing map (SOM) [7], but it requires advance definition of the number of prototype vectors for classes; furthermore, the number is the same for every class. Artificial neural networks (ANN) [8] are efficient for some supervised learning tasks. However, if we add one new class to the system, the whole network must be retrained. For that reason, ANN is unsuitable for incremental learning tasks, which are an important aspect of many applications.

The nearest neighbor [9] algorithm differs from other classification methods in that it does not build a classifier from the training data, but uses the original training vector in the discriminant process. Salient advantages of nearest neighbor are that: it can learn from a small set of examples; it can add new information incrementally at runtime; it requires no optimization, it is capable of modeling very complex target functions using a collection of less complex approximations, and it is robust to noisy training data. On the other hand, its major disadvantage is the high complexity of recognition for large datasets. Shibata et al. [10] proposed a K-D decision tree method to speed up the nearest neighbor algorithm. However, that method and other traditional methods such as condensing [11] and editing [12] techniques were unsuitable for incremental learning.

In [13], an incremental network for unsupervised learning that extended Growing Neural Gas (GNG) [14] was proposed and used to process some face recognition and vector quantization tasks. This method can judge whether new nodes are needed when online training data come into the system. The method can delete nodes caused by noise to thereby decrease misclassification [15]. In this paper, we adjust the network in [13] and design a self-organizing incremental network for online incremental supervised learning tasks. First, we use the online training samples to train the network corresponding to each class and output the nodes of the networks. Then, such nodes are used as prototype vectors of the different classes to assign test data to different classes. This method is suitable for incremental learning: it can learn new nodes as prototypes for new classes easily without destroying old learned information. In light of the nearest neighbor advantages described above, we adopt the nearest neighbor method as

the main benchmark for comparison with the proposed method.

2. Proposed Method

The targets of the proposed algorithm are: (1) Without prior conditions such as number of nodes or a good network structure or a kernel function to conduct online supervised learning, and represent the topological structure of input probability density. (2) To learn new information without destroying old learned information. (3) To learn the number of nodes needed to represent every class to avoid catastrophic allocation of new nodes. (4) To delete nodes caused by noise and thereby decrease misclassification.

2.1. Overview of the Proposed Method

In this study, we adopt a self-organizing incremental neural network to realize our targets. Online training data are used to train the network for each class and generate a topological structure of the input pattern. Then the learned nodes of the networks are used as the prototype vectors to classify the test data.

To represent the topological structure in online or life-long learning tasks, growth is an important feature for decreasing task error and adapting to changing environments while preserving old prototype patterns. Insertion of new nodes is extremely useful to grow the network without interfering with previous learned knowledge. However, insertion must be stopped to prohibit a permanent increase in the number of nodes and to avoid overfitting.

We use a similarity threshold for every old node to judge if a new input pattern originates from a learned region or from a region that never happened. The similarity threshold T_i is defined as the distance (Euclidean distance) from the boundary to the center of Voronoi region V_i of node i. The Voronoi diagram is the nearest-neighbor map for a set of points. In this method, the set of points means the network. Each Voronoi region contains those points that are nearer one node than any other nodes. During the learning process, node i must change its position to meet the input pattern distribution. Therefore, the Voronoi region V_i of the node i will change and the similarity threshold T_i will also change. The similarity threshold is changed adaptively according to the change of position of node i. We set the input signal as a new node (the input signal comes from a never-happened area) when the distance between the signal and the nearest node (node k) is greater than the similarity threshold T_k of node k. After some learning steps, we reject the insertion of the new node in the well learned area because the distance between the input and the nearest node is less than the similarity threshold T_j of node j (the input signal comes from the area where nodes are sufficiently numerous).

To build connections between neural nodes, we adopt the competitive Hebbian rule [16]. The competitive Hebbian rule can be described as follows: for each input signal, connect the two closest nodes (measured using Euclidean distance) by an edge. In online or life-long learning, the nodes change their locations slowly but permanently; nodes that are neighboring at an early stage might not be neighboring at a more advanced stage. Therefore, the competitive Hebbian rule removes connections that have not been refreshed for a while.

In general, noise exists in real world data. To delete the nodes caused by noise and thereby reduce misidentification, we propose the following strategy: if the number of generated input signals up to that time is an integer multiple of a parameter, remove those nodes with no topological neighbor or which have only one. The idea is based on the behavior of adding and removing edges from the network. We adopt the competitive Hebbian rule that connects the two closest nodes for each input signal. Thereby, nodes having many edges produce a network in the area where many input signals come. After numerous input signals, if the node has no neighbor or only one, then during a period, this node has a very low chance to be selected as one of the two closest nodes and the insertion of new nodes near this node is difficult. As a result, we infer that the probability that the node is caused by noise is very high.

2.2. Complete Algorithm

Along with the analysis of Section 2.1, we present the complete algorithm here.

Notations to be used in the algorithm

A	Node set, used to store nodes.
C	Connection set (or edge set), used to store connections (edges) between nodes.
W_i	n-dimension weight vector of node i.
T_i	Similarity threshold. If the distance between an input pattern and node i is greater than T_i, the input pattern is a new node.
N_i	Set of direct topological neighbors of node i. If a node links with node i by an edge, we say the node is the neighbor of node i.
M_i	Local accumulated number of signals of node i.
$age_{(i,j)}$	Age of the edge that connects node i and node j.

Algorithm 2.1: The proposed algorithm for generating a self-organizing incremental network

1. Initialize node set A to contain two nodes, c_1 and c_2 with weight vectors selected 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

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

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

If the distance between ξ and s_1 or s_2 is greater than similarity threshold T_{s_1} or T_{s_2}, the input signal is a new node; add it to A and go to Step2 to process the next signal. The similarity threshold T is calculated using *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 1 to the local accumulated number of signals M_{s_1}.

7. Adapt the weight vectors of the winner and its direct topological neighbors by fraction $\varepsilon_1(t)$ and $\varepsilon_2(t)$ of the total distance to the input signal,

$$\Delta W_{s_1} = \varepsilon_1(t)(\xi - W_{s_1}) \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (3)$$
$$\Delta W_i = \varepsilon_2(t)(\xi - W_i\ ) \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad . \quad (4)$$

for all direct neighbors i of s_1.

We adopt a scheme to adapt the learning rate over time by $\varepsilon_1(t) = 1/t$ and $\varepsilon_2(t) = 1/100t$, and t is the local accumulated number M_{s_1}.

8. Remove edges with an age greater than a predefined threshold age_{dead}. If this creates nodes having no more emanating edges, remove them as well.

9. If the number of input signals generated up to that time is an integer multiple of parameter λ, delete nodes caused by noise as follows: for all nodes in A, search for nodes having no neighbor or only one neighbor, then remove them.

10. Go to Step2 to continue the online supervised learning.

The similarity threshold T (in *Algorithm 2.1*, step3) is defined by the distance from the boundary to the center of the Voronoi region of the node.

Algorithm 2.2: Similarity threshold T

1. Initialize 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 runner-up, update the 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, as
     $T_i = \max_{c \in N_i} \|W_i - W_c\|$, N_i is the neighbor set of node i.
   - If node i has no neighbor, T_i is updated as the minimum distance of node i and all other nodes in A, $T_i = \min_{c \in A \setminus \{i\}} \|W_i - W_c\|$.

In *Algorithm 2.1*, we need to determine two parameters: age_{dead} and λ. These two parameters will influence the frequency of deleting connections among nodes.

The value of age_{dead} indicates the brittleness of edges. The edges are difficult to remove if the value is large, and they are removed easily if the value is small. It is better to set this value small, thereby predisposing the nodes to be isolated and to eradicate the influence of noise if noise occurs frequently. On the other hand, it is better to set the value large to preserve connections and to compel nodes to form a cluster if the distribution of input signals is large.

The value of λ shows the frequency in judgment of deleting nodes. If the value is large, then the frequency is low; a small value indicates high frequency. It is better to set this value small to delete isolated nodes and to eradicate the influence of noise if noise occurs frequently. On the other hand, it is better to set the value large to preserve many nodes and to produce edges between the nodes easily if the distribution of input signals is large.

Thus, if we want to make a network soon, we can choose a large value for these two parameters; on the other hand, we can set the value of these two parameters as small to eradicate the influence of noise if noise occurs frequently. The two parameters depend on the real condition of the task. In the test of the real world dataset in Sec. 3.2, we present a discussion of the two parameters.

To deal with the Stability-Plasticity Dilemma, the proposed algorithm controls the stability and the plasticity of the network on a rate at which the weight vectors of the nodes are adapted to the input signal and a judgment of the insertion of new nodes. Regarding the adaptation of nodes, we set the distance to an inverse value of M_i in step7. That brings stability of the node in the area where the input signal comes frequently, along with plasticity of the new node. In addition, we use the similarity threshold T_i to judge the insertion of a new node. Using this threshold, an input signal which comes from an area surrounded by nodes and edges never becomes a new node (stability). The input signal coming from other areas has a high probability of becoming a new node (plasticity).

3. Experiment

3.1. Artificial Dataset

We conducted our experiment on the dataset shown in **Fig. 1** An artificial 2-D dataset is used to take advantage of its intuitive manipulation, visualization, and resulting insight into system behavior. The dataset is separated into five parts: A, B, C, D, and E. Part A is sinusoidal and separated into A1, A2, and A3 to show incremental properties. The B and C datasets are a concentric circle example. D is a circular area. E is a dataset that satisfies 2-D Gaussian distribution. We add random noise to the dataset to simulate real-world data. For the two parameters age_{dead} and λ, we set $age_{dead} = 100$ and $\lambda = 100$ in this experiment.

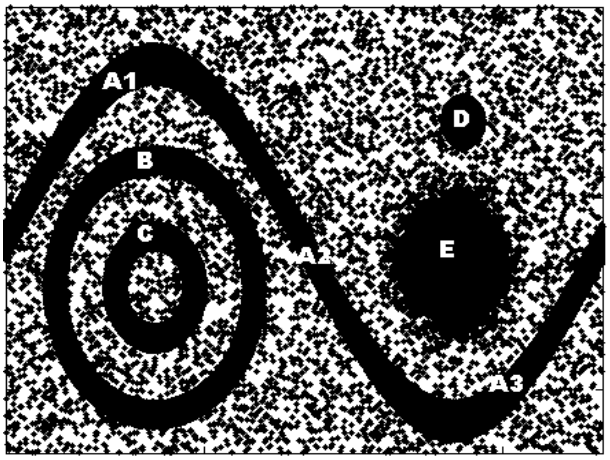


Fig. 1. Original data.

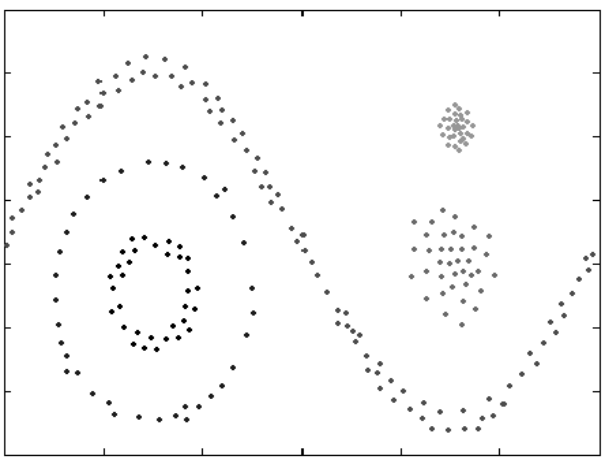


Fig. 2. Learned results of the proposed method, normal environment.

3.1.1. Experiment in Normal Environment

- Training
  In a normal environment, we selected 100,000 patterns randomly from the original dataset (**Fig. 1**) to train the proposed network. All classes are trained simultaneously online . During the training process, no new class occurs. This training condition is necessary for many methods such as ANN. Therefore, we say it is a normal environment.

  Figure 2 shows the training results; the proposed method can represent the topological structure of an input dataset very well. We list the number of nodes needed for every class in **Table 1**. **Table 1** shows that, unlike some self-organizing map (SOM)-based methods, for different classes, different number of nodes are needed for the proposed network. The total number of nodes is 232.

- Testing
  We randomly select 100,000 patterns from areas A, B, C, D and E as the test data, and then use two methods to classify the test data to different classes.

Table 1. Number of nodes for every class in a normal environment.

Class	A	B	C	D	E	Total
Number	99	36	33	25	39	232

1. Nearest Neighbor method
   We use all 100,000 training patterns of the training dataset as the prototype vectors of every class. Then we classify the test data using the nearest neighbor rule: find the nearest prototype vector of a test sample; if the nearest prototype vector belongs to class *i*, the testing sample also belongs to class *i*. Here, the distance between vectors is measured by Euclidean Distance.

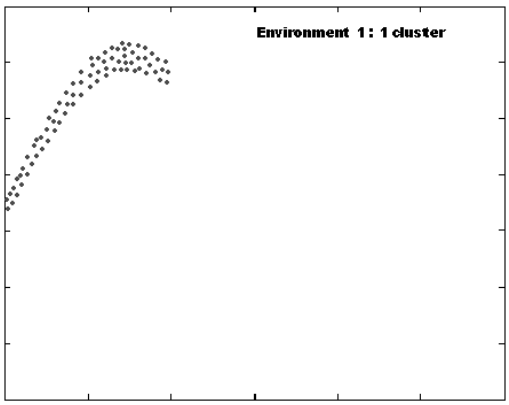
   This method is the traditional Nearest Neighbor method, i.e., all training data are used as the prototype vector set (in this experiment, 100,000 training samples are used as the prototype vectors). The correct recognition ratio is 100%.
2. The proposed method
   We use the network nodes learned using the proposed method as the prototype vectors of every class. Then, the nearest neighbor rule is used to classify the test data to different classes. In this experiment, the total number of nodes is 232. Therefore, we use only 232 prototype vectors to classify the testing dataset. The correct recognition ratio is 100%.

- Comparing
  Comparison of the results of the proposed method with the traditional Nearest Neighbor method shows that the proposed method can obtain an equivalent correct recognition ratio (100%), but requires only 0.232% (calculated by 232/100,000) of the computational load and 0.232% (calculated by 232/100,000) of the memory space of the traditional Nearest Neighbor method. The proposed method speeds up the traditional Nearest Neighbor method by 430 times (calculated by 100,000/232) with the same recognition ratio and very small memory space (only 0.232% of Nearest Neighbor).

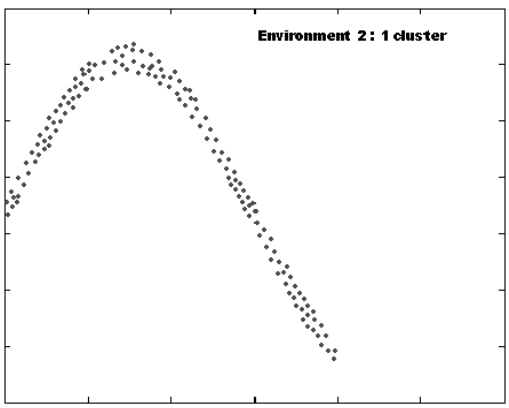
3.1.2. Experiment in Incremental Environment

In this experiment, new classes will come to the system during the training process. Therefore, we name the environment an incremental environment.

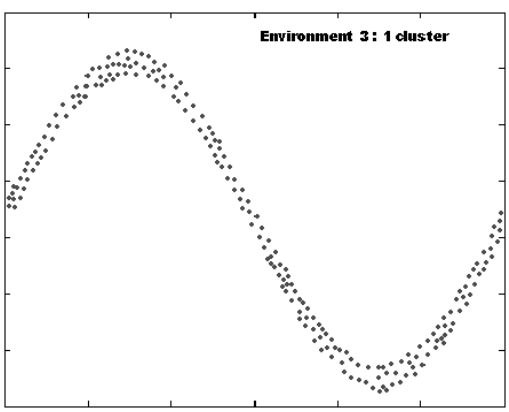
We simulate online learning using the following paradigm: from step 1 to 20,000, patterns are chosen randomly from area A1. At step 20,001, the environment changes and patterns from area A2 are chosen. At step 40,001, the environment changes again, etc. **Table 2** details specifications of the test 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.”



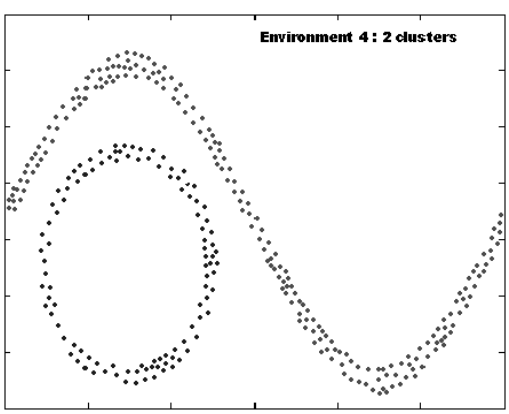
(a): Environment I



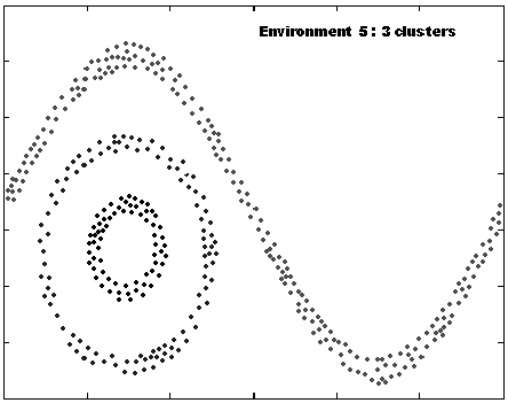
(b): Environment II



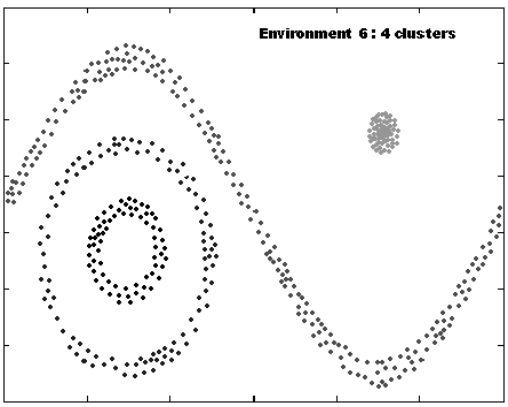
(c): Environment III



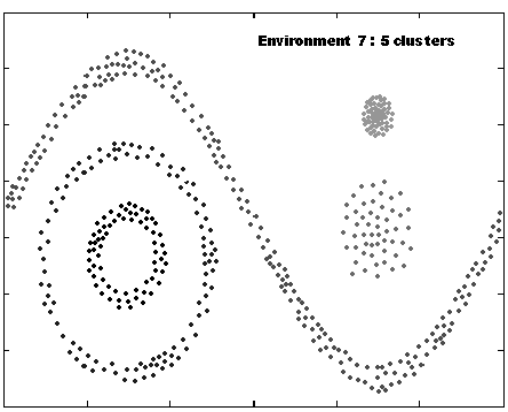
(d): Environment VI



(e): Environment V



(f): Environment VI



(g): Environment VII

Fig. 3. Results of the incremental environment.

Table 2. Incremental environment.

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

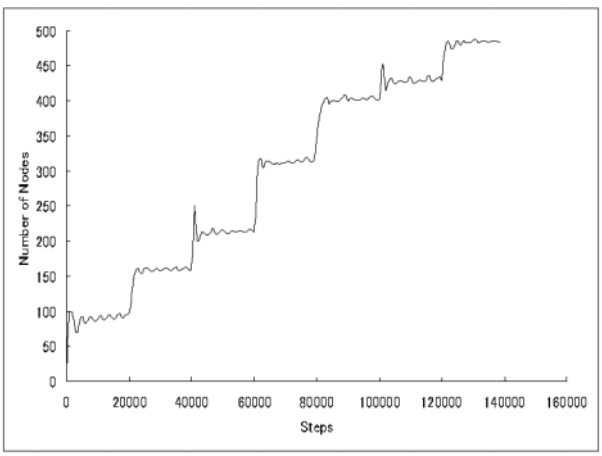


Fig. 4. Number of nodes during incremental learning (Environment I – Environment VII).

Figures 3(a)-(g) show the results of the 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 come to the system sequentially. We find nodes of area A increasing following the change of environment from I to II and III. In environment I, the area A1 dataset is tested. If probability changes to zero in some regions, such as in environment II, the remaining nodes of area A1, often called "dead nodes," play a major role in online or life-long learning. They preserve the knowledge of previous situations for future decisions. In the future, the reappearance of area A1 (environment III) does not increase error and knowledge is completely preserved, so almost no insertion occurs and most nodes remain at their positions. Environments IV and V test a difficult situation (datasets such as areas B and C). The system also works well. Environments VI and VII test an isolated dataset: area D (circular) and area E (Gaussian distribution).

Figure 4 shows how the number of nodes changes during incremental learning. The number of nodes increases when the input signal comes from a new area (see **Table 2** and the 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 the number of nodes converges to a constant because a many nodes and edges exist in the current learning area, i.e., further insertion cannot engender a decrease in error. **Table 3** lists the number of nodes that are needed for different classes. For different classes with different shape and size, the number of nodes is also different; the total number of nodes is 644.

Table 3. Number of nodes for every class in the incremental environment.

Class	A	B	C	D	E	Total
Number	289	99	86	67	103	644

Table 4. Number of samples in the optdigits database.

Class	Training samples	Test samples
0	376	178
1	389	182
2	380	177
3	389	183
4	387	181
5	376	182
6	377	181
7	387	179
8	380	174
9	382	180
Total	3823	1797

For testing, we randomly selected 100,000 samples from areas A, B, C, D and E of **Fig. 1** as the test dataset. For traditional Nearest Neighbor method, the 140,000 training samples are used as the prototype vectors and the correct recognition ratio is 100%. We use the learned 644 nodes as the prototype vectors, and use the nearest neighbor rule to classify the test data. The correct recognition ratio is 100%. The proposed method requires only a 0.46% (calculated by 644/140,000) computation load and 0.46% (calculated by 644/140,000) memory space of the traditional Nearest Neighbor method. The proposed method speeds up the traditional Nearest Neighbor 217 times (calculated by 140,000/644) while providing the same recognition ratio and using very little memory space (only 0.46% of Nearest Neighbor).

3.2. Real World Data: Character Recognition

In this experiment, we use the Optical Recognition of Handwritten Digits database (optdigits) [17] to test our proposed method. In these databases, 10 classes (handwritten digits) are obtained from 43 people, 30 of whom contributed to the training set and a different 13 to the test set. The number of samples in the training set and testing set of every class are listed in **Table 4**. The training set has 3823 samples, and 1797 samples in the test set. The dimensions of the samples are 64. For details of the database, please see the appropriate reference [13].

With traditional Nearest Neighbor method, using the 3823 training samples as the prototype vectors, we classify the test samples to different classes using Euclidean distance as the metric. The correct recognition ratio is 98%. Using some common loss-based decoding schemes for different types of Error-Correcting Output Codes (ECOC) schemes, SVMs as the base binary classifier were trained on a Gaussian kernel with the same value of the variance for all experiments [18]. The maximum recognition ratio is 97.4%.

We use the training set to train the proposed network. We test the proposed method under an incremental environment. At the first stage, we randomly select samples from a training set of class 0 to train the system. At this stage, no samples from other classes are chosen; after 10,000 training iterations, at the second stage, the samples input to the system are selected randomly from a training set of class 1, and training is done 10,000 times. Then for classes $2, 3, \ldots, 9$, we use the same method as the first and second stage to train the proposed network incrementally. After training, we obtain the nodes of the generated self-organizing network. Then we use such nodes as the prototype vectors of classes to classify the test samples of the test dataset and report the correct recognition ratio.

During training, we tested four different parameter sets for $\{age_{dead}, \lambda\}$: (1) $\{2520, 90\}$, (2) $\{100, 100\}$, (3) $\{20, 90\}$, and (4) $\{20, 50\}$.

We adopt GNG as a comparative incremental neural network. As in the experiment with the proposed method, we train the networks and obtain nodes for use as the prototype vectors of classes to discriminate test samples. We set common parameters age_{dead} and λ to the same parameter set as that of the proposed method. We define in advance the maximum number of nodes for each class to halt the permanent insertion of nodes using GNG. We set them to the same number of nodes as a result of the proposed method. We also set some parameters of GNG: $\alpha = 0.2$, $\beta = 0.006$, $\varepsilon_b = 0.5$, and $\varepsilon_n = 0.0005$.

Using the four parameter sets, we obtain different results. **Table 5** lists the number of nodes of every class for different parameter sets. Using the nodes as the prototype vectors, we classify the test samples to different class and give the recognition results for different parameter sets in **Table 6**. From **Table 5** we know that, for different classes, the number of nodes that are necessary to represent the class is also different. **Table 6** shows that: (1) the proposed method speeds up the traditional Nearest Neighbor method 4.53 times using only 22.1% of the memory space while improving the correct recognition ratio from 98% to 98.5% (Column II of **Table 6**); (2) if we want to speed up the classification process greatly with much lower memory space, the correct recognition ratio will be decreased (Column II – Column V of **Table 6**), and the parameter sets $\{age_{dead}, \lambda\}$ will be useful to control the classification property; (3) the performance of the proposed method matches or exceeds that of GNG, in which the number of parameters that must be defined in advance is directly proportional to the number of classes.

Table 5. Number of nodes for different classes of the optdigits database with different parameter sets $\{age_{dead}, \lambda\}$.

Class	No. of nodes for different sets of $\{age_{dead}, \lambda\}$			
	(1)	(2)	(3)	(4)
0	68	42	43	7
1	88	62	52	40
2	82	59	34	33
3	80	56	35	43
4	97	58	38	40
5	84	63	63	28
6	87	46	43	31
7	95	50	41	34
8	81	51	30	48
9	83	57	36	30
Total number	845	544	415	334

Table 6. Comparing classification results of the proposed method with Nearest Neighbor method and Growing Neural Gas for the optdigits database with different parameter sets $\{age_{dead}, \lambda\}$. For the recognition ratio, the upper row describes results of the proposed method. The lower row shows those of GNG.

	set of $\{age_{dead}, \lambda\}$			
	(1)	(2)	(3)	(4)
Recognition ratio (%)	98.5%	97.1%	96.5%	96.0%
	97.5%	97.1%	96.4%	96.0%
No. of prototypes	845	544	415	334
Speed up (times)	4.53	7.02	9.21	11.45
Memory (%)	22.1%	14.2%	10.8%	8.7%

4. Conclusion

This paper presents a new online incremental learning method for supervised classification and topology representation. Using an adaptive similarity threshold, the system can grow incrementally and can accommodate input patterns of online incremental data distribution. In summary, the algorithm can realize online or even life-long incremental supervised learning. It is independent of predefined optimal conditions. It represents the topology structure of inputting online data and learns the number of nodes needed to represent every class. Moreover, it is robust to noisy training data. Experiments using artificial data and high-dimension real-world data show that the proposed method can realize classification with a high recognition ratio, high speed, and low memory.

We must mention that some other problems remain unsolved. For example, two parameters must be determined by the user: age_{dead} and λ. The difficulty of automatically

determining such parameters arises from the fact that, for different tasks, the optimal choice of such parameters will be different. It is difficult to give a standard of such parameters for all tasks. We hope that some methods are useful to induce optimal choice of such parameters according to the different tasks. This problem remains as a subject for our future research.

References:

[1] B. Santosa, T. B. Trafalis, and T. Conway, "Knowledge Base-Clustering and Application of Multi-Class SVM for Genes Expression Analysis," ASME Press, 2002.

[2] L. P. Li, C. R. Weinberg, T. A. Darden, and L. G. Pedersen, "Gene selection for sample classification based on gene expression data: study of sensitivity to choice of parameters of the GA/KNN method," Bioinformatics, Vol.17, No.12, pp. 1131-1142, 2001.

[3] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of online learning and an application to boosting," Journal of Computer and System Sciences, Vol.55, No.1, pp. 119-139, 1997.

[4] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and Regression Trees," Belmont, CA: Wadsworth, 1984.

[5] M. A. Mendez, C. Hodar, C. Vulpe, M. Gonzalez, and V. Cambiazo, "Discriminant analysis to evaluate clustering of gene expression data," Federation of European Biochemical Societies Letters, Vol.522, Nos.1-3, pp. 24-28, 2002.

[6] T. Kohonen, "Self-organization and associative memory," Springer, Berlin, 3rd edition, 1989.

[7] T. Kohonen, "Self-organized formation of topologically correct feature maps," Biological Cybernetics, Vol.43, pp. 59-69, 1982.

[8] R. Feraud and R. Clerot, "A methodology to explain neural network classification," Neural Networks, Vol.15, No.2, pp. 237-246, 2002.

[9] T.M. Cover and P.E. Hart, "Nearest neighbor pattern classification," IEEE Trans. on Information Theory, Vol.IT-13, No.1, pp. 21-27, 1967.

[10] T. Shibata, T. Kato, and T. Wada, "K-D decision tree: An accelerated and memory efficient nearest neighbor classifier," MIRU2004.

[11] B. V. Dasarathy, "Minimal consistent set (MCS) identification for optimal nearest neighbor decision systems design," IEEE Trans. Syst. Man Cybern., Vol.24, No.3, pp. 511-517, 1994.

[12] G. W. Gates, "The reduced nearest neighbor rule," IEEE Trans. Inf. Theory, Vol.IT-18, No.3, pp. 431-433, 1972.

[13] S. Furao and O. Hasegawa, "An incremental neural network for non-stationary unsupervised learning," International Conference on Neural Information Processing (ICONIP2004), Calcutta, India, 2004.

[14] B. Fritzke, "A Growing Neural Gas Network Learns Topologies," In Advances in Neural Information Processing Systems, Vol.7, pp. 625-632, 1995.

[15] S. Furao and O. Hasegawa, "An On-line Learning Mechanism for Unsupervised Classification and Topology Representation," IEEE International Conference on Computer Vision and Pattern Recognition (CVPR05), San Diego, 2005 (accepted).

[16] T. M. Martinetz, "Competitive Hebbian learning rule forms perfectly topology preserving maps," ICANN, pp. 427-434, 1993.

[17] C. Merz and M. Murphy, "UCI repository of machine learning databases, Irvine, CA," University of California Department of Information, 1996.
http://www.ics.uci.edu/˜mlearn/MLRepository.html

[18] A. Passerini, M. Pontil, and P. Frasconi, "From Margins to Probabilities in Multiclass Learning Problems," in F. van Harmelen (ed.), Proc. 15th European Conf. on Artificial Intelligence, 2002.

Name:
Youki Kamiya

Affiliation:
Department of Computational Intelligence and Systems Science, Tokyo Institute of Technology

Address:
R2-52, Hasegawa Research Group, Imaging Science and Engineering Lab., 4259 Nagatsuta, Midori-ku, Yokohama 226-8503, Japan

Brief Biographical History:
2003- Master course student at Tokyo Institute of Technology
2005- Ph.D. student at Tokyo Institute of Technology

Membership in Learned Societies:
• The Institute of Electronics, Information and Communication Engineering (IEICE)

Name:
Shen Furao

Affiliation:
Imaging Science and Engineering Laboratory, Tokyo Institute of Technology

Address:
R2-52, Hasegawa Research Group, Imaging Science and Engineering Lab., 4259 Nagatsuta, Midori-ku, Yokohama 226-8503, Japan

Brief Biographical History:
1998- Lecturer at Nanjing University, China
2000- Senior researcher in Foursis Inc., Japan

Name:
Osamu Hasegawa

Affiliation:
Associate Professor, Imaging Science and Engineering Laboratory, Tokyo Institute of Technology

Address:
R2-52, 4259 Nagatsuta, Midori-ku, Yokohama 226-8503, Japan

Brief Biographical History:
1993 Joined Electrotechnical Laboratory (ETL)
1999- Visiting Scientist, Carnegie Mellon University (CMU)
2000- Advanced Industrial Science and Technology (AIST)
2002 Joined Tokyo Institute of Technology (TITech)
2002- PRESTO, Japan Science and Technology Agency (JST)

Main Works:
• N. Enomoto, T. Kanade, H. Fujiyoshi, and O. Hasegawa, "A method for monitoring activities of multiple objects by using stochastic model," IEICE Trans. on Information and Systems, Vol.84-D No.12, pp. 1705-1712, 2001.
• B. Raytchev, O. Hasegawa, and N. Otsu, "User-independent online gesture recognition by relative motion extraction," Pattern Recognition Letters, 21 (1), pp. 69-82, 2000.
• S. Nobe, S. Hayamizu, O. Hasegawa, and H. Takahashi, "On listener's Fixation and Comprehension of Hand Gestures," Gestures. – Meaning and Use, Fernando Pessoa University Press, pp. 359-363, 2003.

Membership in Learned Societies:
• The Institute of Electrical and Electronics Engineers (IEEE)
• The Institute of Electronics, Information, and Communication engineers (IEICE)
• The Information Processing Society of Japan (IPSJ)
• The Japan Society of Artificial Intelligence (JSAI)
• The Japanese Cognitive Science Society (JCSS)