Proceedings of International Joint Conference on Neural Networks, Orlando, Florida, USA, August 12-17, 2007

An Online Semi-Supervised Clustering Algorithm Based on a Self-organizing Incremental Neural Network

Youki Kamiya, Toshiaki Ishii, Shen Furao, and Osamu Hasegawa

Abstract— **This paper presents an online semi-supervised clustering algorithm based on a self-organizing incremental neural network (SOINN). Using labeled data and a large amount of unlabeled data, the proposed semi-supervised SOINN (ssSOINN) can automatically learn the topology of input data distribution without any prior knowledge such as the number of nodes or a good network structure; it can subsequently divide the structure into sub-structures as the need arises. Experimental results we obtained for artificial data and real-world data show that the ssSOINN has superior performance for separating data distributions with high-density overlap and that ssSOINN Classifier (S3C) is an efficient classifier.**

I. Introduction

Numerous incremental learning algorithms based on neural networks, which are called incremental or competitive neural networks, have been proposed and applied in many application domains as classification models. A salient advantage of incremental neural networks is their capability of operating with information of new data incrementally. Thereby, they can improve their performance by gradually increasing their structural complexity. Well-known examples of incremental or competitive neural networks include Kohonen's self-organizing map (SOM) [1] and Growing Cell Structures (GCS) [2]. Neural network architecture consisting of SOM or GCS layer followed by a Radial Basis Function output layer [3] can realize supervised learning. In addition, the Growing Neural Gas (GNG) architecture is a modification to the GCS, in which the dimensionality of topological structures is not predefined, but is instead discovered during training. In recent years, further modifications have been proposed: Life-long Learning Cell Structures (LLCS) [4] for supervised classification and Self-Organizing Incremental Neural Network (SOINN) [5] for unsupervised clustering. In these methods, the insertion of nodes is stopped automatically. Thereby, these methods avoid the permanent increase of nodes. They are intended to balance stability and plasticity.

A main challenge in the design of efficient and robust learning algorithms is that new datasets are continually added to an already huge database. The use of the huge dataset introduces two issues. The first is how to learn new knowledge without forgetting previous knowledge. This problem is considered to be a main difficulty of the incremental learning algorithms described above. The second is how to learn in a case where labeled instances are difficult, expensive, or time consuming to obtain, because they require the efforts of experienced human annotators. This problem is discussed mainly in semi-supervised learning, and is addressed using a large amount of unlabeled data, together with labeled data, to build better classifiers. However, the former problem has been not considered as sufficient among many types of traditional semi-supervised learning methods.

Youki Kamiya (kamiya.y.ab@m.titech.ac.jp) and Toshiaki Ishii (ishii.t.aa@m.titech.ac.jp) are with the Department of Computer Intelligence and System Science, Tokyo Institute of Technology, Yokohama, 226-8503 JAPAN. Shen Furao (frshen@nju.edu.cn) and Osamu Hasegawa (hasegawa.o.aa@m.titech.ac.jp) are with the Imaging Science and Engineering Laboratory, Tokyo Institute of Technology . Shen Furao is also with the Department of Computer Science, Nanjin University, CHINA, and Japan Society for the Promotion of Science.

Incremental Growing Neural Gas (IGNG) [6] was proposed for incremental and semi-supervised clustering. It can realize semi-supervised clustering, alternating between clustering of a part of the huge dataset and correction of the network by the user. However, human operation is still complex: it includes finding clustering mistakes, correcting the network by insertion and deletion of edges, and assigning a label to each disjoint subgraph. Both Semi-Supervised Fuzzy ARTMAP (ssFAM) and Semi-Supervised Ellipsoidal ARTMAP (ssEAM) have their roots in Adaptive Resonance Theory [7] and are recent extensions of the basic Fuzzy ARTMAP (FAM) architecture [8]. Both ssFAM and ssEAM make use of semi-supervised learning [9]. Although these networks can prevent overfitting to training data to achieve zero post-training error, they require all labeled data for the training.

In this paper, we improve SOINN to address the online semi-supervised clustering. The targets of our proposed algorithm are: (1) to conduct online semi-supervised learning without prior conditions such as the number of nodes or a good network structure; (2) to learn new information without destroying old learned information; and (3) to adjust old learned clusters automatically and continuously based on scarce labeled data.

II. Overview of SOINN

In this section, we introduce the self-organizing incremental neural network (SOINN) [5], which is the foundation of the proposed method. Fundamentally, SOINN adopts a two-layer network. The training results of the first layer are used as the training set for the second layer. The goals of SOINN use are realizing unsupervised learning and representing the topology of the input distribution.

For those purposes, SOINN adopts two schemes: between-class insertion and within-class insertion, to insert new nodes and thereby realize incremental learning and topology representation. In between-class insertion, SOINN uses a threshold distance T to judge whether two samples belong to the same cluster. The sample will be inserted to the network as a new node if a sample belongs to a new cluster. In within-class insertion, SOINN uses a local-error-based insertion criterion by which, on a time average, nodes with high error serve as criteria to insert a new node. To realize within-class insertion, SOINN uses five user-determined parameters and another parameter, the 'error-radius', to judge whether the insertion is successful. The latter feature complicates the system and renders it difficult to understand.

In fact, during training of the first layer of SOINN, between-class insertion is the main task: within-class insertion has little contribution for inserting new nodes. During training of the second layer of SOINN, both between-class insertion and within-class insertion are necessary to give numerically sufficient nodes for representing the topological structure.

The similarity threshold T_i is defined as the Euclidean distance from the boundary to the center of the Voronoi region V_i of node i. During the learning process, node i will change its position to meet the inputting pattern distribution. Consequently, the Voronoi region V_i of the node i will also change; for that reason, the similarity threshold T_i will also change.

Algorithm 2.1: Calculation of 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 *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 neighbor, T_i is updated as the minimum distance of node i and all other nodes in A, $T_i = \min_{c \in A \setminus \{i\}} ||W_i - W_c||$, and A is the node set.

Here, *winner* designates the nearest node to the input pattern, and *second winner* denotes the second-nearest node to the input pattern. In addition, W_i is the weight vector of node i.

To build connections among neural nodes, SOINN adopts the competitive Hebbian rule [10]: for each input signal, connect the two closest nodes (*winner* and *second winner*, which are measured using Euclidean distance) by an edge. During online learning, the nodes alter their locations slowly but permanently; nodes that are neighboring at an early stage might not be neighboring at a more advanced stage. The competitive Hebbian rule removes connections that have not been recently refreshed.

To delete the nodes created by noise, SOINN uses the following strategy: if the number of input signals generated so far is an integer multiple of a parameter λ, remove those nodes that have no neighbor or only one topological neighbor. That idea is based on the consideration that, if the node has no neighbor or only one neighbor, the probability that the node has been created by noise is very high. With the help of the second-layer network, SOINN can delete a slightly higher density of noise and even separate the classes with low-density overlap.

Here we described a brief outline of SOINN. Details of SOINN are described elsewhere [5]. In addition, *Algorithm 3.1* in the next section will aid understanding of SOINN.

III. Semi-Supervised SOINN (ssSOINN)

The proposed semi-supervised SOINN (ssSOINN) inherited some properties of SOINN such as incremental learning and robustness against noise. A shortcoming of SOINN is that it has too many parameters to realize within-class insertion. Another is that it cannot separate the classes with high-density overlap. Moreover, the target of SOINN is to realize unsupervised learning and topology representation. In this study, we seek to perform semi-supervised learning and divide clusters with high-density overlap using labeled and unlabeled data. We improve SOINN in the following respects: (1) adjust SOINN with fewer parameters to represent the topological structure of input data; (2) improve SOINN to obtain the ability to divide clusters with high-density overlap. The proposed method is based on SOINN. Therefore, we designated the proposed algorithm as ssSOINN. A flowchart of ssSOINN is shown in Fig. 1. Figure 1 illustrates that ssSOINN performs adjusted SOINN and subsequent cluster separation processes, which are indicated by bold-typeface elements. In fact, ssSOINN deals with individual patterns in the order that they become available for learning.

A. Adjusted SOINN

As indicated in Section II, during training of the first layer of SOINN, between-class insertion is the main part; within-class insertion contributes little for inserting new nodes. The target of the second layer is to delete redundant nodes and delete nodes resulting from noise.

For the topology-representation target, the first layer can achieve better topology representation than the second layer. Here we adopt only the first layer of SOINN as part of the proposed ssSOINN, then delete the within-class insertion part to ease its comprehension and to retain five user-determined parameters. The deletion of within-class insertion will not affect the learning results because, if we adopt only a single-layer network, the between-class insertion ensures that the node density will be sufficient to represent the topology structure. With the definition of the similarity threshold, newly inserted nodes might not come from the appropriate area, and the subsequent processes (e.g. competitive Hebbian rule [10]) will link such new nodes with old nodes. The adaptively updated similarity threshold will not be too large to make the nodes sparse. With between-class insertion, when the nodes reach the class boundary, the insertion will be stopped automatically for that class; thereby, we avoid the permanent increase of nodes. *Algorithm 3.1* is the detailed algorithm of adjusted SOINN.

Algorithm 3.1: Adjusted SOINN

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 (edge) set C, $C \subset A \times A$, to the empty set.

2) Input new pattern $\xi \in R^n$.
3) Search for 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 the similarity threshold T_{s_1} or T_{s_2}, the input signal is a new node: add it to A and go to Step 2) to process the next signal. The similarity threshold T is calculated using *Algorithm 2.1*.
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) Adapt the weight vectors of the *winner* and its direct topological neighbors by fraction ϵ_1 and ϵ_2 of the total distance to the input signal,
   $\Delta W_{s_1} = \epsilon_1(\xi - W_{s_1})$
   $\Delta W_i = \epsilon_2(\xi - W_i)$ for all nodes i in N_{s1}, which is a direct neighbor set of s_1.
   We adopt a scheme to adapt the learning rate over time using $\epsilon_1 = 1/M_{s_1}$ and $\epsilon_2 = 1/100M_i$: M_a represents the frequency from which node a is selected as the *winner*.
7) Remove edges with age greater than a predefined threshold a_d. If this removal creates nodes that have no more emanating edges, remove them as well.
8) If the number of input signals generated so far is an integer multiple of parameter λ, delete some nodes as follows: for all nodes in A, search for nodes having no neighbor or only one neighbor, then remove them.
9) Go to Step 2) to continue learning until the learning time is satisfied.

For *Algorithm 3.1*, we must determine two parameters a_d and λ. These two parameters will influence the frequency of deletion of the connections between nodes and nodes located in sparse areas. If we wish to retain previously learned knowledge, we choose a large value for these two parameters, and obtain many nodes to realize a detailed topology representation. We set the respective values of these two parameters as small to remove nodes and edges frequently if we desire fewer nodes to save memory space. The two parameters are therefore dependent upon the real condition of the task; we can use these parameters to control the ssSOINN performance.

B. ssSOINN Algorithm

In this subsection, we describe the proposed semi-supervised SOINN in detail. The ssSOINN algorithm includes: (1) the process of adjusted SOINN, which is indicated as unstressed elements in Fig. 1; and (2) the process of judgment to divide a cluster as bold-typeface elements in Fig. 1. As indicated in subsection III-A, adjusted SOINN can

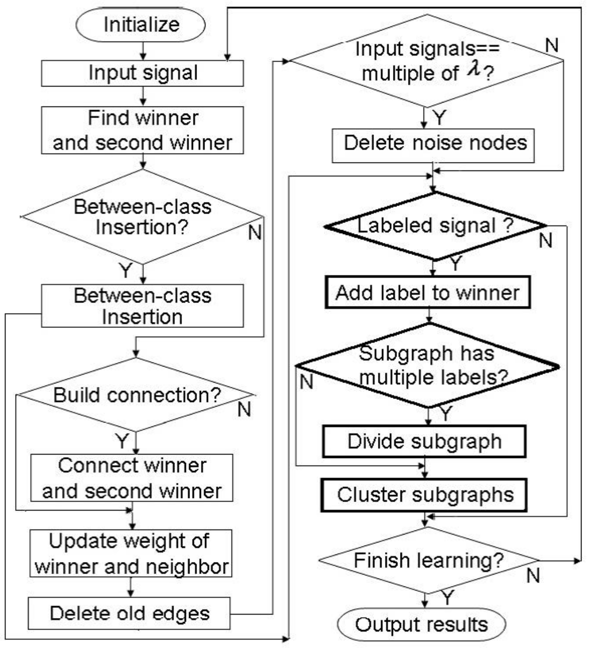

Fig. 1. Learning process of the Semi-Supervised SOINN (ssSOINN)

represent the topology of the input distribution and eliminate noise. Therefore, whether an input pattern is labeled or not, the process of adjusted SOINN learns the topology representation using positions of each input pattern. Then, only when an input pattern is labeled is the subsequent process executed for judgment of cluster division.

In the judgment process, a cluster is divided as circumstances demand. The judgment operates only on the cluster where the nearest node (*winner*) of the input pattern belongs. Here, we present the definition that, if nodes exist where we can follow along some edges, these connected nodes belong to the same cluster. First, in this process, a label of input pattern is added to the *winner*. Then, the *winner* and its connected nodes are only included in a same cluster if they have no label or the same label of *winner*. On the other hand, if the cluster has nodes whose labels differ from each other, that cluster is divided to multiple sub-clusters.

The division of clusters is undertaken based on two assumptions: (1) the nodes connected by an edge tend to be more mutually similar than other disconnected nodes, and (2) clusters tend to shape unimodal distributions. Initially, we collect the labeled nodes in a divided cluster, then we convey the information of its label to direct topological neighbors of each labeled node, and repeat the conveyance to neighbors of these neighbor nodes. Although this process is mainly based on assumption (1), it pauses for some interval based on assumption (2). In other words, the conveyance is interrupted at the nodes whose *winning* frequency, the selected frequency as the nearest node, is less than that of all of its unclassified neighbors. That interruption occurs because the node located in a low-density distribution would be in the boundary area between clusters and would rarely be selected as the nearest node of an input pattern. Moreover, the interrupted conveyances resume, except for assumption (2), if unclassified nodes exist when all conveyances are interrupted. *Algorithm 3.2* is the detailed algorithm of ssSOINN.

Algorithm 3.2: Semi-Supervised SOINN (ssSOINN)

1) Initialize node set A and connection set C, $C \subset A \times A$, as in the case of *Algorithm 3.1*.
2) Input new pattern $\xi \in R^n$.
3) Execute Steps 3) through 8) of the adjusted SOINN (*Algorithm 3.1*). Note that if the input pattern is judged as a new node on Step 3) of the adjusted SOINN, go to Step 4) in this algorithm.
4) If input pattern ξ is unlabeled, go to Step 8).
5) Add label L of input pattern ξ to *winner* s_1; $L_{s_1} = L_\xi$.
6) Cluster these connected nodes if every node that we can follow from s_1 along some edge has no label except L_{s_1}. Then go to Step 8).
7) If some nodes with different labels exist in cluster D, divide cluster D as the following procedure.
   (a) Initialize all nodes in the cluster D to be unclassified and $k = 0$. Initialize terminal node set B_{end} to the empty set and search node set B_k as
   $$B_k = B_0 = \{a | \forall a (a \in D, L_a \neq \emptyset)\}. \qquad (1)$$
   For all nodes $u \in B_0$, change cluster numbers $cl(u_i)$ of each node into new numbers which are mutually different,
   (b) Move all nodes $a \in B_k$ to B_{end}, whose *winning* frequency M_a is less than that of all its unclassified neighbors.
   (c) For all nodes $u \in B_k$, search unclassified neighbors of node u. Set new cluster numbers of all the neighbors to that of node u. In this regard, however, if a neighbor node exists, of which multiple nodes whose labels are different from each other, we do not set a new cluster number of the neighbor, but set it to be classified. Finally, set all searched neighbors to B_{k+1}.
   $$B_{k+1} = \{a | \forall a \in D, a \in N_u \ \ (\forall u \in B_k)\} \qquad (2)$$
   (d) Increment k by 1. If B_k is not the empty set, go to (b). If not, $B_k = B_{end}$.
   (e) Search the unclassified nodes that are neighbors of the nodes in B_k. Set a new cluster number of them to that of the neighbor node in B_k. Add them to B_{k+1}.
   (f) If some unclassified nodes remain, increment k by 1 and go to (e). If not, go to Step 8).
8) Go to Step 2) to continue the online semi-supervised learning until the learning time is satisfied.

After learning, we can construct a prototype-based classifier using the output nodes of ssSOINN as the prototypes. We adopt the One-Nearest-Neighbor (1-NN) rule [11] to classify unknown patterns. We call the classifier the Semi-Supervised SOINN Classifier (S3C). According to Step 7)-(c), we add the cluster number to the nodes except the boundary nodes. Using only the numbered nodes as prototypes, the classifier seems to perform based on the k-Nearest Neighbor rule [11]. The classification performance of S3C is discussed in the next section.

IV. Experimental Results

We conducted a series of experiments using three artificially-generated datasets as well as two real datasets. In experiments using the artificial datasets, we test ssSOINN and illustrate the details of ssSOINN. Additionally, on both artificial and real datasets, we compare the ssSOINN classifier with some semi-supervised classifiers and neural-based classifiers.

A. Description of datasets

Le et al. compared four neural-based classifiers including the GCS, GNG, ssEAM, and ssFAM [12]. Our proposed ssSOINN classifier is also based on neural networks. Therefore, we compare it to these state-of-the-art classifiers. Experimental conditions of datasets resemble those of the experiment described in [12]. Below we provide a brief description of each dataset used in our experiments.

1) G4 datasets: Three datasets were generated by sampling from a bi-variate mixture of isotropic Gaussian distributions with equal priors consisting of four components; each component corresponded to a separate class distribution. The class means were positioned in such a way that featured four-fold symmetry with respect to the horizontal and vertical axes. We generated three datasets named G4LO, G4ME and G4HI with respective Bayes errors of 0.05, 0.15 and 0.4. For the aforementioned three cases, we generated a training set, a cross-validation set, and a test set of 500, 5,000 and 5,000 patterns, respectively. Two labeling selection rates are used in our investigations: 10% and 20%. Fig. 2(a) depicts scatter plots of training patterns from each dataset. In those plots, labeled signals are indicated in assorted colors and unlabeled ones are indicated by black dots.

2) Glass, Pima Indian Diabetes datasets: We drew model comparisons on two out of the four datasets used in [12]. Because of lack of space, we refer the reader to [13] for a detailed description of the sets. We mention only briefly that the sets are small, respectively containing only 214 and 768 total examples. Furthermore, we used identical training, cross-validation, and test subset sizes as in [14].

We must mention that these datasets were not identical to those in [12]. Instead, we generated the datasets 10 times according to the identical distribution scheme and the data subset sizes; we then tested for each dataset. Finally, we calculated the average classification performance of the ssSOINN classifier to compare with other classifiers.

B. Experimental Setup

In this subsection, we explain our experimental setup and the particular methodology we used to conduct the comparisons on classification accuracy. As described above, we referred to the experimental setup in [12] and set our experiments to compare the ssSOINN classifier to four neural-based classifiers.

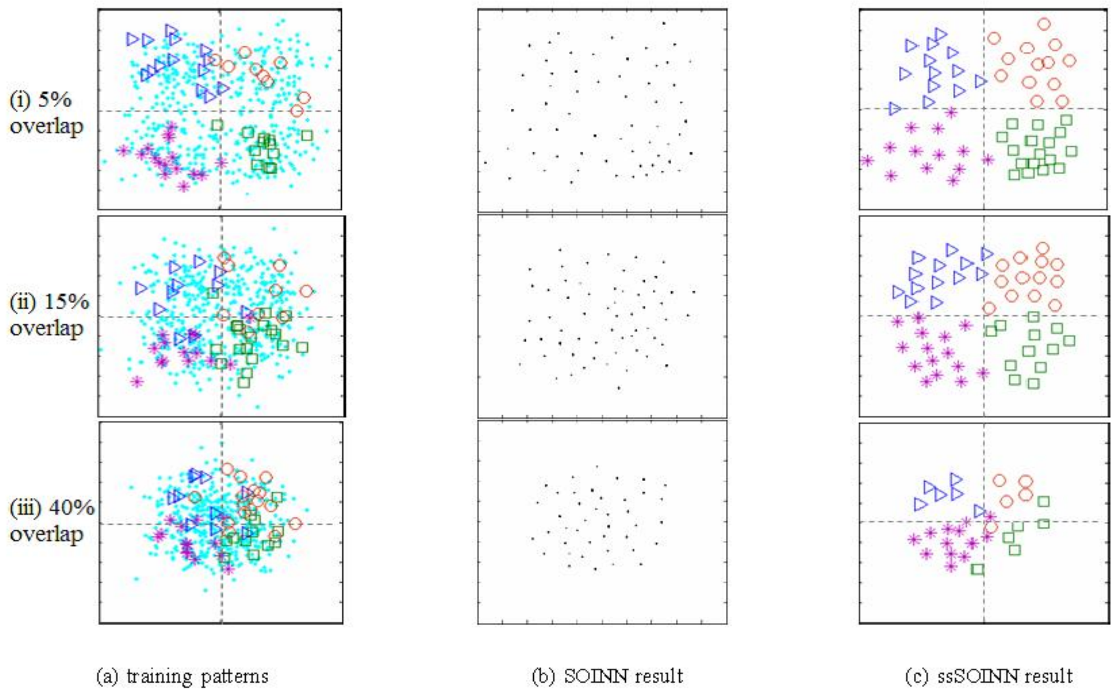

Fig. 2. An example of clustering results on the G4LO (i), G4ME (ii), and G4HI (iii): (a) Scatter plots of training patterns for a 4-Gaussian problem with 10% labeled data (circle, triangle, square, and asterisk marks) and 90% unlabeled data indicated by small dots; (b) Clustering results of SOINN; (c) Clustering results of ssSOINN.

For each dataset pair, we used numerous combinations of training parameter values. The same combinations of parameter values were used for all datasets; each combination gave rise to an individual trained model. From the set of all trained models, we selected the 100 most accurate models in terms of classification performance on the cross-validation set. Then we assessed the accuracy of those 100 best networks on a separate test set. We repeated this particular scheme 10 times to avoid both data and parameter selection bias.

We let two parameters of ssSOINN (a_{dt} and λ) take values in [100:50:300] and [100:300:1000], respectively. Moreover, the training set patterns were presented in 100 different orders during training; the selection of each particular order can be thought of as an additional training parameter. The aforementioned parameter values and value ranges gave 2,000 training parameter value combinations per dataset.

As the benchmark classifier, we used a 1-NN classifier [11]. To construct the 1-NN classifier, labeled signals in the training dataset are used as prototypes. We assessed the accuracy of the 1-NN classifier on a separate test set. We constructed individual classifiers for 100 different datasets and calculated the average performance to avoid data selection bias.

C. Observations

In the following presentation and discussion of the results obtained through this study, PIC represents the percent incorrect classification, that is, the error rate of a classifier. Ideally, in online semi-supervised learning tasks, a classifier should have the lowest possible PIC using the fewest labeled nodes.

Figure 2 shows the distributions of each training dataset (Fig. 2(a)), the clustering results using only adjusted SOINN (Fig. 2(b)), and the clustering results of ssSOINN (Fig. 2(c)). The results of adjusted SOINN prove that, in the case of these data distributions, adjusted SOINN can not realize an individual distribution of classes, but it can learn the topology structure of all training data in online unsupervised learning. In contrast, ssSOINN can separate the individual distributions of classes, as shown in assorted colored nodes in Fig. 2(c).

Table I depicts the classification accuracy and standard deviation for the PIC on the test set for each classifier type and each dataset considered. In this table, the best values for each row are depicted in bold typeface. We use the results in [12] to compare the proposed S3C with GSC, GNG, ssEAM, and ssFAM. Furthermore, as baseline performance, the results of 1-NN classifier are also shown.

Table I shows that, for the G4 datasets, S3C exhibits the smallest PIC point estimates. From the results in Fig. 2 and Table I, we infer that ssSOINN can achieve sufficient performance using few labeled data, when the assumptions in the process to judgment at division of cluster are satisfied ideally. For the Glass and Pima Indian Diabetes datasets, the PIC of S3C also indicates a smaller point than that of 1-NN classifier, which uses only labeled data. Although it is a larger point than the PIC of other neural-based classifiers, training data for these classifiers is labeled completely.

In comparison to state-of-the-art semi-supervised learning

TABLE I

COMPARISON OF GENERALIZATION PERFORMANCES FOR FIVE DATASETS

		PIC Test for 100 classifiers		PIC Test for 100 best classifiers					
		1-NN		S3C		GCS	GNG	ssEAM	ssFAM
		10%	20%	10%	20%	with 100% labeled data [12]			
G4LO	avg.	8.82	8.12	7.09	**6.76**	10.48	10.4	10.62	10.38
	std.	1.9167	1.1488	0.4958	0.3993	0.09211	0.07888	**0.06457**	0.10203
G4ME	avg.	22.44	21.87	19.09	**17.92**	25.36	24.74	26	25.2
	std.	3.1218	2.0340	0.9127	0.7471	0.14061	**0.12173**	0.26508	0.25875
G4HI	avg.	50.07	49.03	42.33	**40.42**	42.16	41.58	42.4	42.06
	std.	4.0464	2.6683	1.1799	0.9599	0.21283	**0.1378**	0.34975	0.25216
GLASS	avg.	49.76	42.45	47.33	42.11	35.8491	39.6226	39.6226	**33.9623**
	std.	10.6437	8.7528	6.3257	5.7478	**1.1024**	2.4243	1.8443	1.3564
DIABETES	avg.	34.76	33.00	30.84	30.00	25.8333	26.3542	**25**	**25**
	std.	4.6356	4.0069	2.8882	2.8713	**1.0586**	1.0824	1.0232	1.6368

methods, Zhang et al. calculated the classification accuracy of their performance-driven incremental learning algorithm and other methods [15]: for these methods, the average accuracies of the Glass dataset are 40.09–44.52% (PIC: 55.48–59.91) using 10% labeled data, 40.82–46.49% (PIC: 53.51–59.18) using 20% labeled data. Dimitriadou et al. evaluated their mixed ensemble approach for the Pima Indian Diabetes dataset. They reported PIC: 30.8 ± 2.6 using 10% labeled data [16]. Table I shows that the S3C can attain equivalent or better classification accuracy than those of the aforementioned algorithms.

V. SUMMARY

In this paper, we proposed a new online semi-supervised clustering algorithm based on a self-organizing incremental neural network (SOINN). Through the adjusted SOINN process and the process of judgment to divide clusters, the proposed semi-supervised SOINN (ssSOINN) can learn the topology structure of input data and can divide a cluster as the need arises. Using the nodes of ssSOINN as prototypes, we can design the ssSOINN classifier (S3C) with 1-NN as the classification rule. In the experiment, clustering performance of ssSOINN is evaluated and S3C is compared to some other classifiers. The results show that ssSOINN achieves superior performance when the our assumptions is satisfied ideally, and shows itself as a equivalent or better method than the traditional semi-supervised learning methods.

We must also state that some other problems remain unsolved. For example, if the occurrence probability of labeled data of each class is different, the adjustment of previous learned information becomes difficult. This problem remains as a subject for our future research.

ACKNOWLEDGMENT

Funding for this study was primarily provided by New Energy and Industrial Technology Development Organization. The authors express their appreciation.

REFERENCES

[1] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, pp. 59–69, 1982.

[2] B. Fritzke, "Growing cell structures - a self-organizing network for unsupervised and supervised learning," *Neural Networks*, vol. 7, no. 9, pp. 1441–1460, 1994.

[3] J. Moody and C. Darken, "Learning with localized receptive fields," in Proc. 1988 Connectionist Models Summer School, 1988, pp. 133-143.

[4] F. Hamker, "Life-long learning cell structures — continuously learning without catastrophic interference," *Neural Networks*, vol. 14, no. 4, pp. 551–572, 2001.

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

[6] Y. Prudent and A. Ennaji, "An incremental growing neural gas learns topologies," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN '05)*, Montreal, Canada, 2005, pp. 1211–1216.

[7] S. Grossberg, "Adaptive pattern recognition and universal encoding ii: Feedback, expectation, olfaction, and illusions," *Biological Cybernetics*, vol. 23, pp. 187–202, 1976.

[8] G. Carpenter, S. Grossberg, N. Markuzon, J. Reynolds, and D. Rosen, "Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps," *IEEE Trans. on Neural Networks*, vol. 3, no. 5, pp. 698–713, 1992.

[9] G. Anagnostopoulos, M. Bharadwaj, M. Georgiopoulos, S. Verzi, and G. Heileman, "Exemplar-based pattern recognition via semi-supervised learning," in *Proc. of IJCNN '03*, vol. 3, 2003, pp. 1350–1356.

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

[11] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Trans. on Information Theory*, vol. IT-13, no. 1, pp. 21–27, 1967.

[12] Q. Le, G. C. Anagnostopoulos, M. Georgiopoulos, and K. Ports, "An experimental comparison of semi-supervised artmap architectures, gcs and gng classifiers," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN '05)*, Montreal, Canada, 2005, pp. 3121–3126.

[13] P. Murphy and D. Aha, "Uci repository of machine learning databases," 1998, http://www.ics.uci.edu/m̃learn/MLRepository.heml.

[14] F. Hamker and D. Heinke, "Implementation and comparison of growing neural gas, growing cell structures and fuzzy artmap," Schriftenreihe des FG Neuroinformatik der TU Ilmenau, Tech. Rep. Report 1/97, 1997.

[15] R. Zhang and A. Rudnicky, "A new data selection principle for semi-supervised incremental learning," in *Proc. ICPR'06*, vol. 02, 2006, pp. 780–783.

[16] E. Dimitriadou, A. Weingessel, and K. Hornik, "A mixed ensemble approach for the semi-supervised problem," in *Proc. ICANN'02*, 2002, pp. 571–576.