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

An Online Semi-supervised Active Learning Algorithm with Self-organizing Incremental Neural Network

Shen Furao, Keisuke Sakurai, Youki Kamiya, Osamu Hasegawa

Abstract — **An online semi-supervised active learning algorithm is proposed, which is based on self-organizing incremental neural network (SOINN). The proposed method do not need any priori knowledge such as number of nodes or number of classes; it can automatically learn number of nodes and teacher vectors needed by the current task; It can realize online incremental learning even life-long learning. The experiments for artificial data and real world data show that the proposed method works efficiently.**

I. Introduction

In general, to obtain good learning results, we need lots of labeled training objects for supervised learning. However, the acquisition of labeled training data is costly and time consuming, whereas unlabeled samples can be obtained easily. For the reason that semi-supervised learning is able to learn from both labeled and unlabeled samples, to reduce the cost of labeling the input training data, semi-supervised learning has been the focus of much research in the last few years.

In practice, if we have to label a few instances for learning process, it may be attractive to let the learning algorithm tell us which instances to label, rather than selecting them randomly. This learning method is called active learning. It may make sense to utilize active learning in conjunction with semi-supervised learning. Recently, some semi-supervised active learning methods such as Generative Model[9], Co-Training[2], Transductive SVM (TSVM)[1], and GRF[11], are proposed. Some expanded semi-supervised active learning methods for the above methods are shown in [7][12].

The above-mentioned methods belong to batch learning, during the learning process, they need to use all the input data. It means that, to realize the batch learning, we need to store all the input data. But in the real-world environment, there may be plenty of data, and it is probably impossible for us to store all the input data. Therefore, it will be very interesting if we can realize semi-supervised active learning under the online environment.

As one of the efficient online learning methods, Shen and Hasegawa [10] proposed a self-organizing incremental neural network (SOINN). SOINN separates the data distribution to different clusters by measuring the similarity (distance) between the data. SOINN is one of the unsupervised learning method, without any priori condition (such as the distribution of input data, number of classes, etc.), it can process non-stationary data, separate clusters with very complex shape, report suitable number of clusters, and represent topology structure of the input data.

In this paper, we will expand SOINN to suit the online semi-supervised active learning. During the extraction of topology structure of the input data, we actively label some suitable nodes (queried by the system), and use such nodes to label all other nodes in SOINN automatically.

Shen Furao is with the State Key Laboratory for Novel Software Technology at Nanjing University, Nanjing, 210093, China (email: frshen@nju.edu.cn) and Japan Society for the Promotion of Science; Keisuke Sakurai (email: sakura@isl.titech.ac.jp) and Youki Kamiya (email: kamiya.y.ab@m.titech.ac.jp) are with Department of Computer Intelligence and System Science, Tokyo Institute of Technology; Osamu Hasegawa is with the Imaging Science and Engineering Laboratory, Tokyo Institute of Technology.

II. Overview of SOINN

SOINN is a combination of self-organizing map [4] and competitive Hebbian learning [6] that can be used to learn the topology structure of the input data. It generates nodes to represent the typical data point of input data, and uses edges to connect those nodes if the nodes satisfying the Hebbian rule. SOINN can learn the number of nodes needed for topology representation of input data, and it suits the online incremental learning.

SOINN adopts two-layer network. The first-layer learns the density distribution of the input data and uses nodes and edges to represent the distribution. The second-layer separates the clusters by detecting the low-density area of input data. SOINN adopts the same learning algorithm for first-layer and second-layer.

Figure 1 shows the flowchart of the learning process of SOINN.

When an input vector is given to SOINN, SOINN finds the nearest node (winner) and second nearest node (second-winner) of the input vector, then judges if the input vector belongs to the same cluster of the winner or second winner by using the criterion of similar threshold. In the first-layer of SOINN, SOINN adaptively update the similar threshold of every node for the reason that the distribution of input data is unknown. If node i has neighbor nodes, the similar threshold T_i is calculated by the maximum distance between node i and its neighbor nodes:

$$T_i = \max_{j \in N_i} ||\boldsymbol{W}_i - \boldsymbol{W}_j|| \quad (1)$$

Here, N_i is the set of neighbor nodes of node i, $\boldsymbol{W}_i$ is the weight vector of node i. If node i has no neighbor nodes, the similar threshold T_i is defined by the minimum distance between node i and other nodes in the network.

$$T_i = \min_{j \in N \setminus \{i\}} ||\boldsymbol{W}_i - \boldsymbol{W}_j|| \quad (2)$$

1-4244-1380-X/07/$25.00 ©2007 IEEE

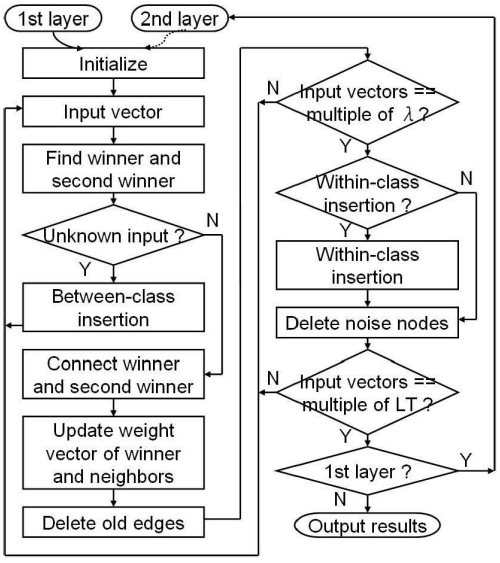

Fig. 1. Flowchart of SOINN

Here, N is the set of all nodes.

If the distance between the input vector and winner or second-winner is greater than similar threshold of winner or second-winner, the input vector will be inserted to the network as a new node to represent the first node of a new class. This insertion is called between-class insertion.

If the input vector is judged as belonging to the same cluster of winner or second winner, update the weight vector of winner and its neighbor nodes. We use i to mark the node which is winner, and use M_i to show the times for node i to be winner. The change to the weight of winner $\Delta \boldsymbol{W}_i$ and change to the weight $\Delta \boldsymbol{W}_j$ of the neighbor node $j(\in N_i)$ of i are defined by:

$$\Delta \boldsymbol{W}_i = \frac{1}{M_i}(\boldsymbol{W}_s - \boldsymbol{W}_i) \tag{3}$$

$$\Delta \boldsymbol{W}_j = \frac{1}{100M_i}(\boldsymbol{W}_s - \boldsymbol{W}_j) \tag{4}$$

Here, $\boldsymbol{W}_s$ is the weight of the input vector.

If there are no edge existing between the winner and second-winner, connect the winner and second-winner with an edge, and set the 'age' of the edge be '0', and increase the age of all edges linked with winner by '1'. If the age of one edge is greater than a predefined parameter $\text{age}_{\max}$, remove that edge.

After λ times learning, SOINN inserts new nodes in the position where the accumulating error is very large. If the insertion cannot lead to decreasing of error, cancel the insertion. The insertion here is called within-class insertion. Then SOINN finds the nodes whose neighbor is less than or equal to 1, and delete such nodes for the supposition that such nodes lie in low-density area.

In fact, because the similar threshold of first-layer of SOINN is adaptively updated, the accumulation error will not be high, thus the within-class insertion is hardly to be

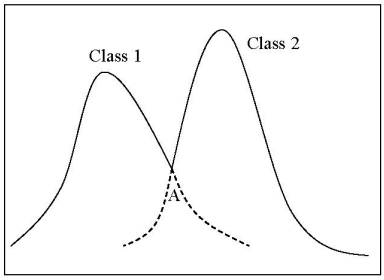

Fig. 2. Density distribution of overlapped classes

success. It means that the within-class insertion for first-layer is not needed.

After LT times learning of first-layer, the learning results will be used as the input for the second-layer. The second-layer of SOINN uses the same learning algorithm as first-layer.

III. The Proposed Method

In this section, at first, we propose a new clustering method that only adopts the first-layer of SOINN to realize the same performance as the two-layer SOINN. After that, we expand the unsupervised learning single-layer SOINN to semi-supervised learning.

A. Single-layer SOINN for clustering

In the traditional SOINN, the first-layer learns the density distribution of input data and generates the topology structure, and second-layer separates the low-density area to realize clustering. However, for second-layer, if the learning results of first-layer are changed, all learned results of second-layer would be destroyed, and we need to re-train the second-layer. It means that the second-layer of SOINN is not suitable for online incremental learning process. If we only adopts the first-layer, we also have the problem that how to separate the overlapped clusters appropriately.

One feature of SOINN is it can represent the density distribution of input data and generate nodes to represent the topology structure of input data. Generally, in the high-density area, plenty of nodes will be generated. Therefore, in the high-density area, the distance between neighbor nodes will be short. Here, we define the density $D(i)$ of node i with

$$D(i) = \frac{1}{(1+d_i)^2} \tag{5}$$

where

$$d_i = \frac{\sum_{j \in N_i} ||\boldsymbol{W}_i - \boldsymbol{W}_j||}{|N_i|} \tag{6}$$

is the mean of the distance between node i and its neighbors. $|N_i|$ means the number of elements in the neighbor set N_i.

According to the definition of density, if the density of input data is large, the density of the associated node will also be large. We suppose that in the central part of a cluster,

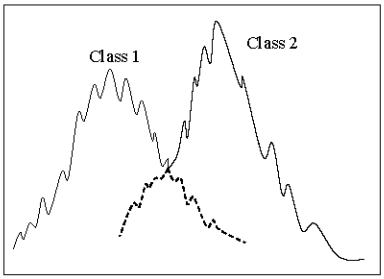

Fig. 3. Clusters with overlpped area (with fluctuation)

the density of the input data is high, and in the overlapped area, the density of input data is low (Figure 2). Thus those nodes with high density lie in the central part, and those nodes with low density lie in overlapped area.

If one node i satisfy $D(i) > D(j)$ for all $j \in N_i$, we call node i the central node. With the central node, we separate the clusters obtained from single-layer SOINN to some small subclusters.

Algorithm 3.1: Separate nodes to subclclusters

1. Find all central nodes $\{c_i\}_{i=1}^n$ with local maximum density, give such nodes different class labels. Initialze the set of labeled nodes $J_S = \{c_i\}_{i=1}^n$.
2. For all unlabeled nodes, find the node i with highest density, $i = \text{argmax}_{k \in N \setminus J_S} D(k)$.
3. Find the node j with highest density in the neighbors of node i, $j = \text{argmax}_{k \in N_i} D(k)$
label node i with the label of node j. Update the labeled data set J_S with $J_S = J_S \cup \{i\}$
4. If $J_S = N$, stop. Else, go to Step2.

The density of nodes will be influenced by noise, and the density distribution is not smooth but fluctuated (Figure 3). Under this situation, Algorithm 3.1 will separate original data to lots of small subclusters unnecessarily. Here, we try to smooth the density distribution by grouping the separated subclasses.

We use the density difference of central node to the boundary of subclusters to realize grouping. Suppose node a belongs to subcluster A, node b belongs to subcluster B, if there is an edge (a, b) links node a and b, we call the edge (a, b) 'boundary edge' between subcluster A and B. For example, in Figure 2, there is boundary edge lies in the valley area A. We call the density in the valley 'density of boundary edge':

$$D(a, b) = \min(D(a), D(b)) \tag{7}$$

Generally, there are many boundary edges lie between subclusters, we call the boundary edge with highest density 'maximum boundary edge':

$$D_{AB} = \max_{(a,b) \in E_{AB}} D(a, b) \tag{8}$$

Here, E_{AB} is the set of boundary edges between subclusters A and B.

We use the density of 'maximum boundary edge' to judge if we need to group the subclusters. If $D(c_A) - D_{AB} < G_C$ or $D(c_B) - D_{AB} < G_C$ is satisfied, we say subcluster A and B belong to a same group. Here, c_A, c_B is the central nodes of subcluster A and B respectively, G_C is the group threshold of the cluster C which A and B belong to:

$$G_C = \frac{\alpha}{|E_C|} \sum_{(i,j) \in E_C} |D(i) - D(j)| \tag{9}$$

Here, E_C is the edge set of cluster CC$|E_C|$ is the number of elements of E_C, and α is the smoothing parameter.

Algorithm 3.2: Grouping the subclusters

1. Find the set of maximum boundary edges $J_G = \{(a_i, b_i)\}_{i=1}^n$.
2. Randomly choose one $(a_i, b_i) \in J_G$, suppose that node a_i belongs to subcluster A, and node b_i belongs to subcluster B. If III-A or III-A is satisfied, give all nodes belong to A and B the same group label, and update
$$J_G = J_G \setminus \{(a_i, b_i)\} \tag{10}$$
3. If $J_G = \Phi$, stop. Else, go to Step2.

With the Algorithm 3.1 and Algorithm 3.2, we only adjust the clustering technique of SOINN without change of the topology structure, and we can detect the overlapped area and expand the improved SOINN to semi-supervised active learning.

B. Expand SOINN to semi-supervised active learning

SOINN is an unsupervised learning method, even if we give label to input training data, SOINN will process such data as unlabeled data. Now, we expand SOINN to process labeled data. If a labeled vector with weight $\boldsymbol{W}_s$ is input to SOINN, we find the nearest node i to the vector: $i = \text{argmin}_{i \in N} \|\boldsymbol{W}_i - \boldsymbol{W}_s\|$, and give node i the same label of the input vector. Like this way, if one node is directly labeled with the label of an input vector, this node is called 'teacher node.'

SOINN generates nodes and topology structure based on the density of input data, and the data belong to the same class will be allocated to a same cluster. With Algorithm 3.1, we use the distribution of nodes within a cluster to detect the overlapped area, and separate the cluster to different subclusters. If the generated nodes belong to the same subcluster, we give these nodes the same label. Therefore, given teacher nodes, we label the unlabeled nodes with the same label as the teacher node within the same subcluster. If there are no teacher nodes in the subcluster, we can label the unlabeled node with teacher nodes within the same group or the same cluster.

Algorithm 3.3: Label the nodes with teacher nodes

1. Initialize the teacher node set N_L, and initialize the labeled node set with $J_L = N_L$.
2. Choose one node $i \in N \backslash J_L$.
3. Suppose N_S is the node set within same subcluster as node i, if $N_S \cap J_L \neq \Phi$, find node l satisfies $l = \text{argmax}_{j \in N_S \cap J_L} D(j)$, go to Step 7.
4. Suppose N_G is the node set within same group as node i, if $N_G \cap J_L \neq \Phi$, find node l satisfies $l = \text{argmax}_{j \in N_G \cap J_L} D(j)$, go to Step 7.
5. Suppose N_C is the node set within same cluster as node i, if $N_C \cap J_L \neq \Phi$, find node l satisfies $l = \text{argmax}_{j \in N_C \cap J_L} D(j)$, go to Step 7.
6. For all nodes in N_S, give the nodes new class labels. Update the labeled node set $J_L = J_L \cup N_S$, go to Step 8.
7. For all nodes in N_S, give the nodes the same label as the teacher node l. Update the labeled node set $J_L = J_L \cup N_S$.
8. If $J_L = N$, stop. Else, go to Step2.

According to this algorithm, if we use teacher vector to label some teacher nodes generated by SOINN, we can label all unlabeled nodes of SOINN, i.e., SOINN is expanded to suit semi-supervised learning task.

Also, if the central node is a teacher node, all other teacher nodes within the same subcluster have no influence for the labeling for unlabeled nodes. Thus we only need to set the central node as teacher node and it is enough for labeling task. From the single-layer SOINN we also know that, the neighbor nodes may have different labels, and we call such nodes 'boundary node.' If we want to realize high precision learning, we set the boundary nodes as teacher nodes and ask for their label.

Algorithm 3.4: Actively query for class label

1. Suppose the cluster set without teacher nodes is $A_C = \{C_i\}_{i=1}^n$, if $A_C \neq \Phi$, find cluster C with $C = \text{argmax}_{a \in A_C} |N_a|$, and find node q with highest density in cluster C:$q = \text{argmax}_{i \in N_C} D(i)$, go to Step5.
2. Suppose the group set without teacher node is $A_G = \{G_i\}_{i=1}^m$, if $A_G \neq \Phi$, find group G with $G = \text{argmax}_{a \in A_G} |N_a|$, and find node q with highest density in group G: $q = \text{argmax}_{i \in N_G} D(i)$, go to Step5.
3. Suppose the subcluster set without teacher node is $A_S = \{S_i\}_{i=1}^l$, if $A_S \neq \Phi$, find subcluster S with $S = \text{argmax}_{a \in A_S} |N_a|$, and find the node with highest density in subcluster S: $q = \text{argmax}_{i \in N_S} D(i)$, go to Step5.
4. Suppose the boundary node set without teacher node is N_B, if $N_B = \Phi$, stop the learning without query. If $N_B \neq \Phi$, randomly choose one $q \in N_B$.
5. Ask for the output label of the weight vector $\boldsymbol{W}_q$ of node q, stop.

With the Algorithm 3.3 and Algorithm 3.4, we realized the semi-supervised active learning.

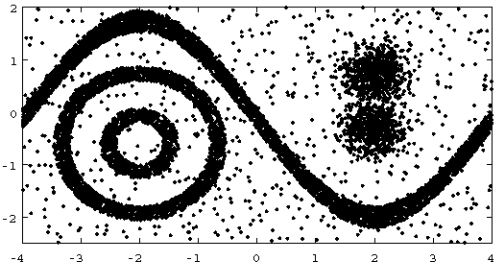

Fig. 4. Artificial data used for experiment

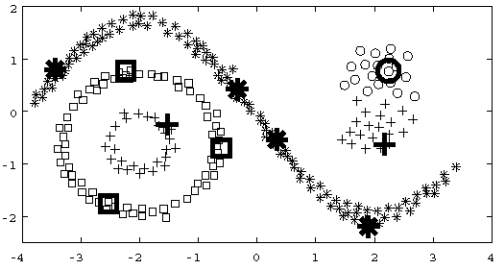

Fig. 5. Experiment results of artificial data under stationary environment. Unlabeled nodes are labeled with different marks, and teacher nodes are labeled with large marks.

IV. Experiment

A. Experiment with artificial data

In this experiment, we use artificial data as the input vector to do the experiment. The data set is shown in figure 4. The data set is composed of two overlapped Gaussian distributions, two concentric rings, and a sinusoidal curve. We set the inside of the concentric rings and the lower Gaussian distribution as a same class, other distributions belong to different classes, and there are totally 4 classes. To simulate the real world data, we add 10% noise to the data set.

In the first experiment, we do the experiment under stationary environment, i.e., choose input vector randomly from the data set. After 100,000 times training, according to Algorithm 3.4, the system asks for label of 10 teacher nodes. The parameters of the proposed method are $\lambda = 500$, $\text{age}_{\max} = 30$, and $\alpha = 2.0$. The experiment results are shown in figure 5.

From Figure 5 we know that, without influence of noise, the proposed method can represent the distribution of input data, and give suitable cluster labels to all nodes with only 10 labeled teacher nodes.

In the second experiment, we do the online learning under non-stationary environment, i.e., the input data of classes are incrementally input to the system. The input vectors are chosen from classes by every 50000 times with the order that lower Gaussian distribution and inside concentric ring at first, then higher Gaussian, then outside concentric ring, and sinusoidal curve at last. The parameters are set as $\lambda = 250$, $\text{age}_{\max} = 40$, and $\alpha = 3.2$.

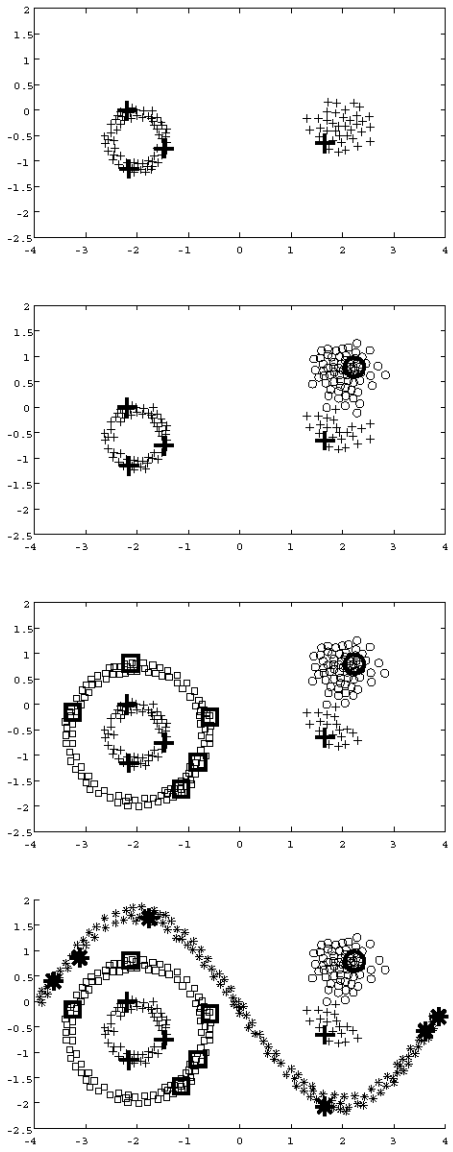

Fig. 6. Incremental learning results of artificial data. From top to down, show the results after 50000, 100000, 150000, and 200000 times training.

After every 50000 times training, Algorithm 3.4 asks for the class label of some teacher nodes. Figure 6 gives the results for every 50000 times training. If the data from new classes coming to the system, the system will generate nodes to represent the new distribution, and ask for the label of teacher nodes of such new input classes.

B. Experiment with real world data

1) Compare to complete storage method: In this experiment, we set the nodes generated by the proposed method as prototypes for classification, and compare it with the nearest neighbor method that storage all the training data. We use recognition ratio and number of teacher vectors needed as the comparison criterion. The data sets used for test are chosen from UCI Repository database: Pima, WDBC, Iris, and Optdigits [8]. We test Pima, WDBC, Iris data sets by leave-one-out technique. For Optdigits, we do 100 times testing and take the average as the result. For such data sets, the parameters are set as in TableI. For Pima, WDBC, and Iris, after 100000 times training of single-layer SOINN, the system asks for label of teacher nodes with Algorithm 3.4. For Optdigits, after 200000 times training, the system asks for label of teacher nodes.

TABLE I

PARAMETERS FOR DIFFERENT DATA SETS

	Pima	WDBC	Iris	Optdigits
λ	1000	400	300	10000
$\text{ag}_{\max}$	30	150	150	100

TABLE II

RECOGNITION RATIO COMPARISON

	Pima	WDBC	Iris	Optdigits
1-NN	68.0	91.6	96.0	98.0
Proposed	69.4	92.1	96.7	96.5

TABLE III

NEEDED TEACHER VECTORS COMPARISON

	Pima	WDBC	Iris	Optdigits
1-NN	767	568	149	3823
Proposed	78.9	34.2	14.2	147.4

The classification results are shown in TableII and III. According to TableII, the proposed method can get nearly the same recognition ratio as complete storage method (nearest neighbor classifier). According to TableIII, the needed teacher vectors for the proposed method is much less than the complete storage method. It means that, compared with nearest neighbor method, the proposed method needs much less teacher vectors and obtain the same recognition ratio.

In [5], the comparison experiments are realized for TSVM, GRF, and Logistic GRF. The best results of such methods are: for Pima, with 70 teacher vectors to get nearly 72% recognition ratio; and for WDBC, with 30 teacher vectors to get 95.5% recognition ratio. Even the recognition ratio is a little less than the above-mentioned results, the proposed method can realize online learning and new classes can incrementally add to the system during the learning process (incremental learning), and this property is a much better predominance for other semi-supervised active learning methods.

2) Efficiency of active learning: In this experiment, we use Optdigits to test the efficiency of Algorithm 3.4. We compare the results of actively query the teacher nodes and randomly query the teacher nodes. After 200000 times training, we actively query teacher nodes according to Algorithm 3.4 and report the results. Then we randomly choose teacher nodes and report the results. We use same parameter for both experiments: $\lambda = 10000$,$\text{age}_{\max} = 100$,$\alpha = 1.0$. We do 100 times test and take the average as the last results, as shown in Figure 7. With less teacher vectors, active queries obtain much higher recognition ratio than random queries. Following the increasing of teacher vectors, the recognition

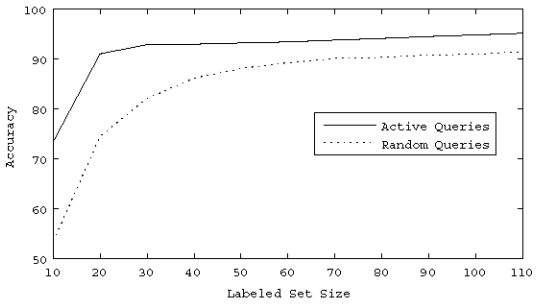

Fig. 7. Comparison: actively queries or random queries

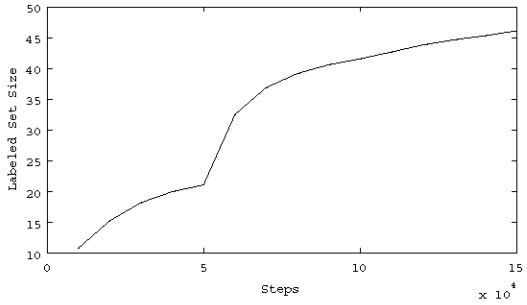

Fig. 8. Incremental learning: needed number of teacher vectors

ratio of active queries is increased much faster than random queries.

3) Experiment for incremental learning: In this experiment, we use Optdigits to test the incremental learning: new classes are added to the system during the online learning process.

The input data changes in order of odd digits, even digits, and all digits for every input 50000 times. After every 10000 times learning, system query for labels of teacher nodes. The parameters are set as $\lambda = 10000$, $\text{age}_{\max} = 100$, $\alpha = 1.0$. Figure 8 and Figure 9 show the results. Figure8 shows that, the needed teacher vectors increase drastically when new classes coming to the system. Figure 9 shows that, when new classes are input to the system, the recognition ratio will be decreased, but following the learning process, the recognition ratio will be satisfied.

V. Conclusion

In this paper, we proposed an online semi-supervised active learning method, which is expanded from SOINN. We do not need any priori knowledge such as number of nodes or number of classes for the proposed method. It can automatically learn number of nodes and teacher vectors needed by current tasks. It can realize online incremental learning even life-long learning which is impossible for some other semi-supervised learning methods. The proposed method also can get rid of the influence of noise. According to the experiments, the proposed method works efficiently.

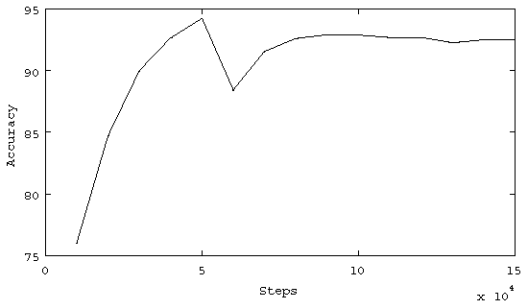

Fig. 9. Incremental learning: recognition ratio

Acknowledgment

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

This study was primarily supported by Industrial Technology Research Grant Programin '04 from New Energy and Industrial Technology Development Organization (NEDO) of Japan. This work was supported in part by the National Natural Science Foundation of China under grant no. 60573157.

References

[1] K. Bennett, and A. Demiriz, "Semi-supervised Support Vector Machines", *Advances in Neural Information Processing Systems 11*, pp. 368-374, 1999.

[2] A. Blum, and T. Mitchell, "Combining Labeled and unlabeled Data with Co-training", *Proc. of the Conference on Computational Learning Theory*, pp. 92-100, 1998.

[3] B. Fritzke, "A Growing Neural Gas Network Learns Topologies", *In Advances in Neural Information Processing System*, vol. 7, pp. 625-632, 1995.

[4] T. Kohonen, " Self-organized Formation of Topologically Correct Feature Maps", *Biol. Cybern*, vol. 43, no. 1, pp. 59-69, 1982.

[5] B. Krishnapuram, D. Williams, Y. Xue, A. Hartemink, L. Carin, and M. Figueiredo, "On Semi-Supervised Classification", *In Advances in Neural Information Processing Systems 17*, pp. 721-728, 2004.

[6] T.M. Martinetz, S.G. Berkovich, and K.J. Schulten, ""Neural-Gas" Network for Vector Quantization and its Application to Time-Series Rrediction", *IEEE Trans. Neural Networks*, vol. 4, no. 4, pp. 558-569, Jul 1993.

[7] I. Muslea, S. Minston and C. Knoblock, "Active + Semi-supervised Learning = robust Multi-view Learning", *Proceedings of ICML-02, 19th International Conference on Machene Learning* pp. 435-442, 2002.

[8] C. Merz and M. Murphy, UCI Repository of machine learning database, Irvine, CA: University of California Department of Information, 1996.

[9] K. Nigam, A.K. McCallum, S. Thrum, and T. Mitchell, "Text Classification from Labeled and Unlabeled Documents Using EM", *Machine Learning*, 39, pp. 103-134, 2000.

[10] 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.

[11] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised Learning Using Gaussian Fields and Harmonic Functions", *ICML-03, 20th Internatonal Conference on Machine Learning*, pp. 912-919, 2003.

[12] X. Zhu, J. Lafferty, and Z. Ghahramani, "Combining Active Learning and Semi-supervised Learning Using Gaussian Fields and Harmonic Functions", *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pp. 58-65, 2003.