Neural Comput & Applic (2011) 20:1061–1074
DOI 10.1007/s00521-010-0428-y

ORIGINAL ARTICLE

An incremental online semi-supervised active learning algorithm based on self-organizing incremental neural network

Furao Shen · Hui Yu · Keisuke Sakurai · Osamu Hasegawa

Received: 20 November 2009 / Accepted: 21 July 2010 / Published online: 6 August 2010

Abstract An incremental online semi-supervised active learning algorithm, which is based on a self-organizing incremental neural network (SOINN), is proposed. This paper describes improvement of the two-layer SOINN to a single-layer SOINN to represent the topological structure of input data and to separate the generated nodes into different groups and subclusters. We then actively label some teacher nodes and use such teacher nodes to label all unlabeled nodes. The proposed method can learn from both labeled and unlabeled samples. It can query the labels of some important samples rather than selecting the labeled samples randomly. It requires neither prior knowledge, such as the number of nodes, nor the number of classes. It can automatically learn the number of nodes and teacher vectors required for a current task. Moreover, it can realize online incremental learning. Experiments using artificial data and real-world data show that the proposed method performs effectively and efficiently.

Keywords Semi-supervised learning · Active learning · Online incremental learning · Self-organizing incremental neural network

F. Shen (✉) · H. Yu
The State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, People's Republic of China
e-mail: frshen@nju.edu.cn

F. Shen · H. Yu
Jiangyin Information Technology Research Institute, Nanjing University, Nanjing, People's Republic of China

K. Sakurai · O. Hasegawa
Imaging Science and Engineering Laboratory, Tokyo Institute of Technology, Tokyo, Japan

1 Introduction

In general, to obtain good learning results, it is necessary to use many labeled training objects. However, the acquisition of labeled training data is costly and time-consuming because it requires the efforts of human annotators. Consequently, it is difficult to prepare sufficiently numerous labeled objects, although unlabeled samples are easily obtainable. Since semi-supervised learning can learn using both labeled and unlabeled samples, thereby reducing the cost of labeling the input training data, semi-supervised learning has attracted much research effort recently [1–4].

In practice, if we have to label a few instances for learning processes, it might be attractive to let the learning algorithm inform us which instances to label, rather than selecting them randomly. This learning method is called active learning, which means that, the system queries the labels of some important samples in accordance with the learning results; it then uses such labeled samples to obtain efficient learning for unlabeled data. Some active learning methods such as query-by-committee [5], co-testing [6], and other recently published methods [7, 8] have been proposed.

Both semi-supervised learning and active learning are designed to use fewer labeled data to obtain good learning results. Therefore, it might make sense to use active learning in conjunction with semi-supervised learning. Recently, some semi-supervised active learning methods have been proposed, such as co-EMT [9], generative model [10], co-training [11], transductive support vector machine (TSVM) [12], and GRF [13]. Some expanded semi-supervised active learning methods are described in [9, 14–17].

The methods listed above use batch learning: during the learning process, they must use all the input data. However, in a real-world environment, there might be plenty of data: it is probably impossible for us to store all the input data.

Therefore, it would be interesting if we were able to realize semi-supervised active learning in an online environment, i.e., update the learning machine step-by-step following the online input data and the latter update must be independent of the learned data.

Incremental learning addresses the ability of repeatedly training a network by using new data without destroying old prototype patterns. Incremental learning is useful in many applications. For example, if we intend to bridge the gap between the learning capabilities of humans and machines, we must consider the circumstances that allow the sequential acquisition of knowledge. The fundamental issue for incremental learning is how a learning system can adapt to new information without getting corrupted or forgetting previously learned information: the so-called stability-plasticity dilemma [18].

As an efficient online incremental learning method, Shen and Hasegawa [19] proposed a self-organizing incremental neural network (SOINN). The SOINN separates the data distribution into different clusters by measuring the similarity (distance) among the data. In fact, SOINN is an unsupervised learning method that requires no prior condition (such as the distribution of input data or number of classes). It can process non-stationary data, separate clusters with very complex shape, report a suitable number of clusters, and represent topological structures of the input data.

In this paper, we will explain an extension of SOINN to suit an online incremental semi-supervised active learning. During extraction of the input data's topology, we actively label some suitable nodes (queried by the system) and use such nodes to label all other nodes in SOINN automatically. The proposed method is able to learn from both labeled and unlabeled samples. It can query the label of some important samples rather than selecting the labeled samples randomly. It requires no prior knowledge such as the number of nodes or number of classes. It can automatically learn the number of nodes and teacher vectors needed for a current task; moreover, it can achieve online incremental learning.

This paper is organized as follows. In Sect. 2, we overview SOINN, which is the basis of the proposed method. In Sect. 3, we improve SOINN and describe a new clustering method to solve some shortcomings of SOINN. We extend the improved SOINN to online semi-supervised active learning in Sect. 4. In Sect. 5, we use some experiments to test the efficiency of the proposed method. In Sect. 6, we summarize the results of this research and discuss the future research.

2 Overview of SOINN

A SOINN adopts a two-layer network to realize the learning process. The first layer learns the probability density distribution of the input data and uses nodes to represent the distribution. The second layer separates the clusters by detecting the low-density area of the input data. The SOINN adopts the same learning algorithm for the first and second layers. Figure 1 presents a flowchart of the SOINN learning process.

An input vector finds the nearest node (winner) and the second nearest node (second winner) of the input vector when it is given to SOINN. It then judges if the input vector belongs to the same cluster of the winner or second winner by using the similarity threshold criterion. In the first layer of SOINN, it adaptively updates the similar threshold of every node because the input data distribution is unknown. The similar threshold T_i is calculated using the maximum distance between node i and its neighboring nodes if node i has neighboring nodes:

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

In this equation, N_i is the set of neighboring nodes of node i, and $\boldsymbol{W}_i$ is the weight vector of node i. The similar threshold T_i is defined as the minimum distances between node i and other nodes in the network if node i has no neighboring nodes.

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

Here, N is the set of all nodes.

The input vector belongs to a new cluster different from that of the winner and second winner if the distance

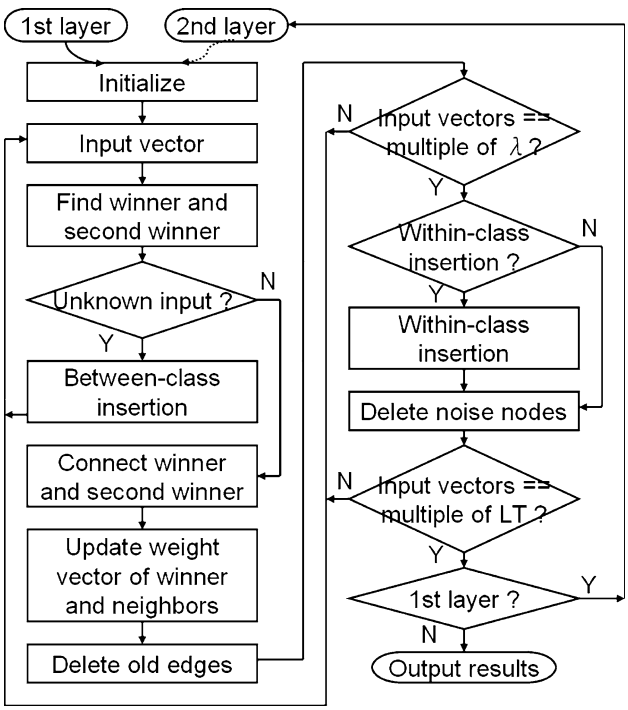

Fig. 1 Flowchart of SOINN

between the input vector and the winner or second winner is greater than the similarity threshold of a winner or second winner. In this situation, the input vector will be inserted into the network as a new node to represent the first node of a new class. This insertion is called a between-class insertion.

The weight vector of the winner and its neighboring nodes are updated if the input vector is judged as belonging to the same cluster of a winner or second winner.

The winner and second winner are connected with an edge if no edge connects them. The "age" of the edge is set as "0"; subsequently, the age of all edges linked to the winner is increased by '1'. The system removes that edge if the age of one edge is greater than a predefined parameter age_{max}.

After λ learning iterations, the SOINN inserts new nodes in the position where the accumulating error is extremely large. It cancels the insertion if the insertion cannot decrease the total error. This type of insertion is called within-class insertion. SOINN then finds the nodes whose neighbors are less than or equal to 1 and deletes such nodes on the basis of the presumption that such nodes lie in the low-density area.

SOINN is able to process online non-stationary data, conduct unsupervised learning without prior conditions, report a suitable number of classes and represent the topological structure of input probability density. It can also separate classes with low-density overlap and detect the main structure of clusters that are polluted by noise. It can incrementally learn new input information without destroying the learned knowledge. With the above-mentioned features, SOINN has been used in applications such as pattern classification [20, 21], associative memory [22], pattern-based reasoning [23], word grounding [25], and grammar learning [26].

However, the two-layer SOINN has some shortcomings. Chief among them is that it is difficult to choose when to halt first-layer learning and begin second-layer learning. Also, for the second layer, if the learning results of the first layer were changed, all learned results of the second layer would be destroyed, thereby necessitating retraining of the second layer. It implies that the SOINN second layer is unsuitable for online incremental learning.

3 Improvement of SOINN

For the first layer of SOINN, the output results are the weights of nodes that are typical points of the input data. Such nodes and edges between the nodes represent the topological structure of the input data. After *LT* learning iterations of the first layer, the learning results will be used as the input for the second layer.

In fact, because the similarity threshold of the first layer of SOINN is updated adaptively, the accumulated error will not be high; hence, the within-class insertion is slightly successful. The within-class insertion for the first layer is unnecessary. The second layer of SOINN uses the same learning algorithm as the first layer. The similarity threshold of the second layer is calculated using the results of the first layer and it is constant. Therefore, for the second layer of SOINN, both between-class and within-class insertions are important.

However, for the second layer, if the learning results of the first layer are changed, all learned results of the second layer will be destroyed. Moreover, the second layer must be retrained, which means that the second layer of SOINN is unsuitable for an online incremental learning process. Also, we must confront the problem of how to separate the overlapped clusters appropriately if we only adopt the first layer.

In [24], it is shown that the first layer of SOINN is able to represent the topological structure of the input data. In this paper, we propose a new cluster method that adopts only the first layer of SOINN to realize online incremental learning. It can achieve the same classification performance as the two-layer SOINN. We present a flowchart depicting the proposed single-layer SOINN in Fig. 2.

One SOINN feature is that it can represent the density distribution of input data and generate nodes to represent the topological structure of the input data. In general, many nodes will be generated in the high-density area. Therefore, in a high-density area, the distance between neighboring nodes, i.e., the length of the edges linking such nodes, will be short. Here, we define the density $D(i)$ of node i using the mean of the distance between node i and its neighbors, as

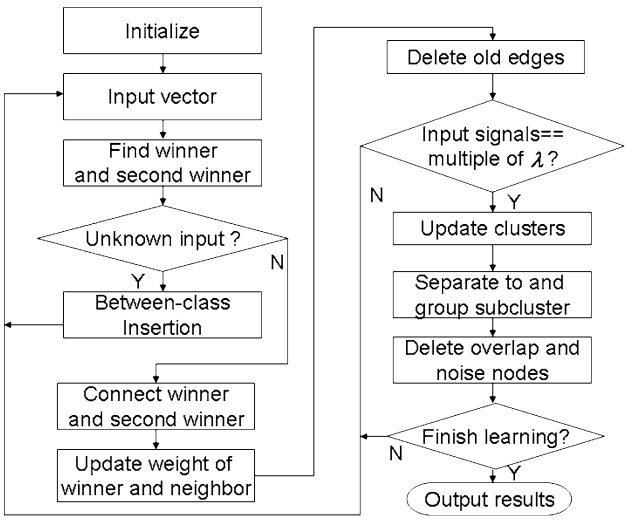

Fig. 2 Flowchart of single-layer SOINN

$$D(i) = \frac{1}{(1+d_i)^2}, \tag{3}$$

where

$$d_i = \frac{\sum_{j \in N_i} \|\boldsymbol{W}_i - \boldsymbol{W}_j\|}{|N_i|} \tag{4}$$

is the mean of the distance between node i and its neighbors. In addition, $|N_i|$ represents the number of elements in the neighboring set N_i.

According to the definition of density of a node, if the density of the input data is high, the density of the associated node will also be high. We presume that, in the central part of a cluster, the input data density is high; in the overlapped area, the input data density is low (Fig. 3). Consequently, high-density nodes lie in the central part, and low-density nodes lie in the overlapped area.

We designated node i as the central node if one node i satisfies $D(i) > D(j)$ for all $j \in N_i$. With the central node, we can separate the clusters obtained from the single-layer SOINN into small subclusters. Algorithm 1 describes how to separate the clusters into subclusters.

In real tasks, the density distribution is not smooth but fluctuated (Fig. 4); it might be caused by noise or a dearth of samples. Under this situation, the original data will be unnecessarily separated into several small subclusters (Fig. 5). Here, we try to smoothen the density distribution by judging whether we need to separate the fluctuated distribution into subclusters, and group the resultant separated subclasses.

We use the density difference of the central node to the boundary of subclusters to achieve grouping. Presuming that node a belongs to subcluster A and that node b belongs to subcluster B, if an edge (a, b) links node a and b, we call edge (a, b) a "boundary edge" between subclusters A and B. For example, in Fig. 4, some boundary edges can be seen to lie in the overlapped valley area, and Fig. 6 shows examples of such boundary edges. We call the density in that valley the "boundary edge density":

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

In general, numerous boundary edges lie between subclusters and we call the boundary edge with the highest density the "maximum boundary edge":

Algorithm 1 Separate SOINN nodes into subclusters

1: Find all central nodes $\{c_i\}_{i=1}^n$ with local maximum density and assign such nodes different class labels. Initialize the set of labeled nodes as $J_S = \{c_i\}_{i=1}^n$.
2: For all unlabeled nodes, find node i with the highest density, as

$$i = \underset{j \in N \setminus J_S}{\operatorname{argmax}} D(j) \tag{5}$$

3: Find node j with the highest density among the neighbors of node i, as

$$j = \underset{k \in N_i}{\operatorname{argmax}} D(k) \tag{6}$$

4: Label node i with the label of node j. Update the labeled dataset J_S with $J_S = J_S \cup \{i\}$.
5: If $J_S = N$, stop. Otherwise, go to Step 2.

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

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

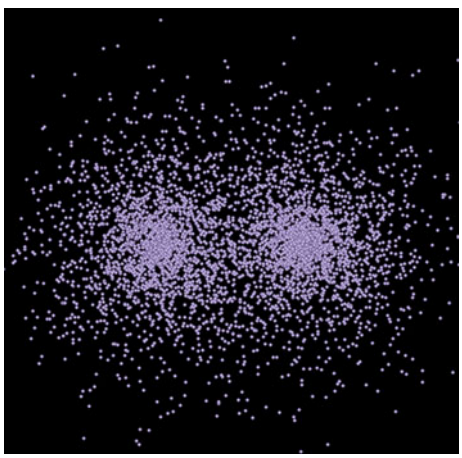

Fig. 3 Two overlapped classes: two-dimensional Gaussian distribution

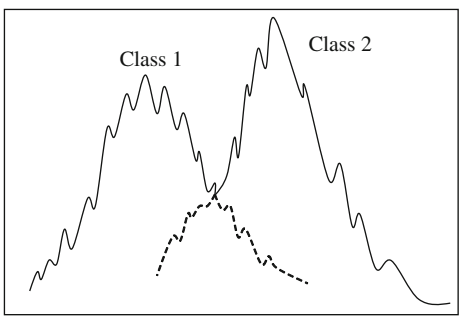

Fig. 4 Clusters with overlapped area (with fluctuation)

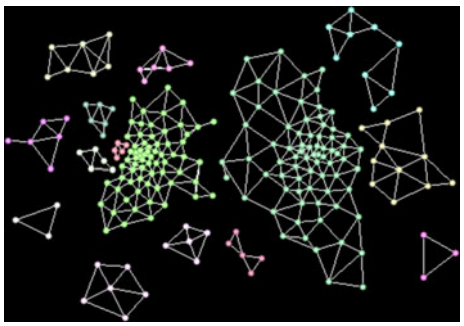

Fig. 5 With the fluctuated distribution of Fig. 4, Fig. 3 is separated into many subclusters

We use the density of the "maximum boundary edge" to judge whether we need to group the subclusters. If

$$D(c_A) - D_{AB} < G_C \tag{9}$$

or

$$D(c_B) - D_{AB} < G_C \tag{10}$$

is satisfied, we say that subclusters A and B belong to the same group. Here, c_A and c_B denote the central nodes of subcluster A and B, respectively; G_C is the group threshold of the cluster C, to which both A and B belong.

$$G_C = \frac{\alpha}{|E_C|} \sum_{(i,j)\in E_C} |D(i) - D(j)| \tag{11}$$

In the above equation, E_C is the edge set of cluster C, $|E_C|$ is the number of elements of E_C, and α is the smoothing parameter, which means that, if the difference between the density of the central nodes and the density of the maximum boundary edges is less than a group threshold G_C, the two subclusters A and B will be grouped in the same group.

The group threshold G_C is determined by the mean density difference between boundary nodes and the smoothing parameter α. The grouping of subclusters will be difficult if a small value is adopted for α. All subclusters will be grouped together if a very large α is adopted.

According to the maximum boundary edge, with the formula shown in (9) and (10), we judge whether we need to group the subclusters. Algorithm 2 summarizes the grouping process. Figure 7 shows the grouping results of Fig. 5.

Using Algorithm 1 and 2, we merely adjust the clustering technique of SOINN without changing the topological structure. It becomes possible for us to adopt only the improved single-layer SOINN to detect the overlapped area and present the topology of the input data.

Algorithm 2 Grouping the subclusters

1: Find the set of maximum boundary edges $J_G = \{(a_i, b_i)\}_{i=1}^{n}$. For every node, assign a different group label to the subcluster to which the node belongs.
2: Randomly choose one $(a_i, b_i) \in J_G$; then presume that node a_i belongs to subcluster A, and that node b_i belongs to subcluster B. If $D(c_A) - D_{AB} < G_C$ or $D(c_B) - D_{AB} < G_C$ is satisfied, assign all nodes belonging to A and B the same group label, and update $J_G = J_G \backslash \{(a_i, b_i)\}$.
3: If $J_G = \emptyset$, stop. Otherwise, go to Step 2.

We summarize the improved one-layer SOINN in Algorithm 3. For the reason described above, we do not include a within-class insertion part in the algorithm.

In Algorithm 3, some parameters are determined by the user. Actually, λ will be used to decide when to cluster nodes, and when to separate and group the subclusters. As described in [19], λ is not sensitive. In addition, α influences the grouping of subclusters. These parameters are difficult to determine automatically, they depend on the real-world environment, distribution of input data, number of samples, and the amount of noise. For example, if few

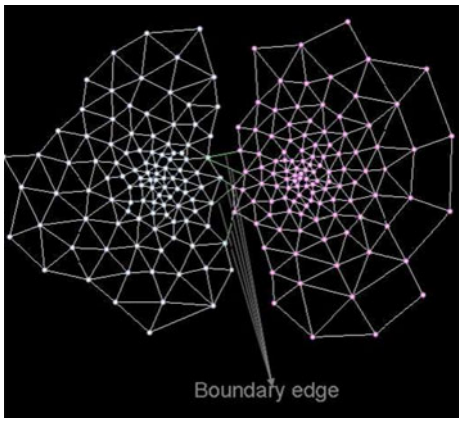

Fig. 6 Boundary edges between two subclusters

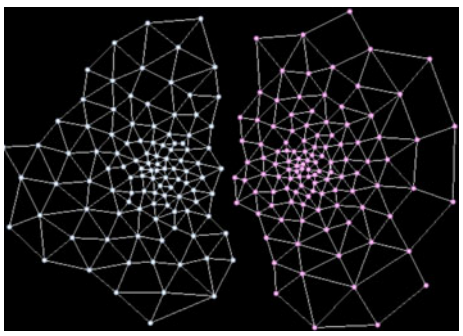

Fig. 7 Grouping results of Fig. 5

samples are in the training set, the density distribution will fluctuate and a large value of α is needed to simplify the grouping process. If the training set contains lots of samples, we can set α small value to make the grouping process difficult. We will explain the sensitivity analysis of α in Sect. 5.

4 Extension to semi-supervised active learning

4.1 Semi-supervised learning SOINN (S-SOINN)

In fact, SOINN is an unsupervised learning method. Hence, even if we assign a label to the input training data, SOINN will process the data as unlabeled data. Here, we extend SOINN to process the labeled data. If a labeled vector with weight $\boldsymbol{W}_s$ is input to SOINN, we find the nearest node w_1 to the vector by formula (12).

Algorithm 3 generates nodes and the topological structure on the basis of the input data density; data belonging to the same class will be allocated to the same cluster. For low-density nodes, it is possible that such nodes lie in the overlapped area between classes. For high-density nodes, it is possible that such nodes lie in the central part of a class. Algorithm 3 detects the overlapped area, separates the nodes into different subclusters, and combines some over-separated subclusters into groups. The probability of such nodes having an identical label is high if the nodes belong to the same group. If nodes belong to the same subcluster, the probability of such nodes having the same label is much higher. We give these nodes an identical label if the generated nodes belong to the same subcluster. Therefore, given teacher nodes, we are able to label the unlabeled nodes with the same label as the teacher node within the same subcluster.

Algorithm 3 Learning algorithm of the improved single-layer SOINN

1: Initialize node set $N = \{n_1, n_2\}$, weight vector $\boldsymbol{W}_{n_1}, \boldsymbol{W}_{n_2}$ are randomly chosen from input dataset I_s.
2: Randomly choose one input vector $\boldsymbol{W}_s \in I_s$.
3: Find winner w_1 and second-winner w_2 from the node set N, as

$$w_1 = \underset{i \in N \setminus \{i\}}{\operatorname{argmin}} \|\boldsymbol{W}_i - \boldsymbol{W}_s\| \tag{12}$$

$$w_2 = \underset{i \in N \setminus \{i, w1\}}{\operatorname{argmin}} \|\boldsymbol{W}_i - \boldsymbol{W}_s\| \tag{13}$$

4: Calculate similarity threshold T_{w_1} and T_{w_2} by formula (1) or (2).
5: **if** $\boldsymbol{W}_s - \boldsymbol{W}_{w_1} > T_{w_1}$ or $\boldsymbol{W}_s - \boldsymbol{W}_{w_2} > T_{w_2}$ **then**
6: Add a new node n_s with weight vector $\boldsymbol{W}_s$ to node set N, as $N = N \cup \{n_s\}$;
7: Set the initial accumulated number of signals $M_{n_s} = 0$, go to Step 19.
8: **else**
9: $M_{w_1} = M_{w_1} + 1$.
10: **end if**
11: **if** No connection exists between w_1 and w_2 **then**
12: Connect w_1 and w_2 with an edge (w_1, w_2);
13: $\text{age}_{(w_1,w_2)} = 0$.
14: **end if**
15: Update weight vector of the winner with

$$\boldsymbol{W}_{w_1} \leftarrow \boldsymbol{W}_{w_1} + \Delta \boldsymbol{W}_{w_1} \leftarrow \boldsymbol{W}_{w_1} + \frac{1}{M_{w_1}}(\boldsymbol{W}_s - \boldsymbol{W}_{w_1}) \tag{14}$$

16: Update the weight vector of the neighboring nodes of winner $i \in N_{w_1}$ with

$$\boldsymbol{W}_i \leftarrow \boldsymbol{W}_i + \Delta \boldsymbol{W}_i \leftarrow \boldsymbol{W}_i + \frac{1}{100 M_{w_1}}(\boldsymbol{W}_s - \boldsymbol{W}_i) \tag{15}$$

17: Increase the age of edges linked with winner by 1, as $\text{age}_{(w_1,i)} = \text{age}_{(w_1,i)} + 1$.
18: Find the edges whose age is greater than $\text{age}_{\max}$; delete such edges.
19: **if** The number of input vectors is a multiple of λ **then**
20: Remove the nodes whose neighbor nodes are less than or equal to 1;
21: Call Algorithm 1 to cluster all generated nodes;
22: Call Algorithm 2 to group subclusters.
23: **end if**
24: If the learning is finished, stop. If not, go to Step 2.

We also assign node w_1 the same label as that of the input vector $\boldsymbol{W}_s$. In this manner, if one node is labeled directly with the label of an input vector, the node is called a "teacher node".

We can label the unlabeled node with teacher nodes within the same group if no teacher node exists in the subcluster. Algorithm 4 shows details of the labeling process.

Algorithm 4 Label the nodes with teacher nodes

1: Initialize teacher node set N_L and labeled node set with $J_L = N_L$.
2: Choose one node $i \in N \backslash J_L$. Assume that N_S is the node set within the same subcluster as node i, and N_G is a node set within the same group as node i.
3: **if** $N_S \cap J_L \neq \emptyset$ **then**
4: Find node l which satisfies

$$l = \underset{j \in N_S \cap J_L}{\operatorname{argmax}} D(j). \tag{16}$$

5: Go to Step 12.
6: **end if**
7: **if** $N_G \cap J_L \neq \emptyset$ **then**
8: Find node l which satisfies

$$l = \underset{j \in N_G \cap J_L}{\operatorname{argmax}} D(j). \tag{17}$$

9: Go to Step 12.
10: **end if**
11: For all nodes in N_S, assign new class labels to the nodes. Go to Step 13.
12: For all nodes in N_S, assign the nodes the same label as that of the teacher node l.
13: Update the labeled node set $J_L = J_L \cup N_S$.
14: If $J_L = N$, then stop. Otherwise, go to Step 2.

According to this algorithm, if we use the teacher vector to label some teacher nodes generated by Algorithm 3, we can label all unlabeled nodes that represent the distribution of the input data: Algorithm 3 is thereby expanded to suit semi-supervised learning tasks. Algorithm 5 portrays the semi-supervised learning process.

Algorithm 5 Semi-supervised learning with improved SOINN (S-SOINN)

1: Using Algorithm 3 to train all labeled and unlabeled data (both online and batch training are possible). Generated nodes are separated into different groups, and subclusters.
2: After a period of learning, do the following Steps 3 and 4 (for batch learning, Steps 3 and 4 can be followed after the learning of Algorithm 3 is completed).
3: Using labeled data (teacher vectors) to find teacher nodes from the generated node set with formula (12).
4: Using Algorithm 4 to label all nodes with teacher nodes found in Step 3.
5: If online learning is not finished, go to Step 1.

Using Algorithm 5, we know that the method is not only suited to batch learning, but that it is also suitable for online incremental learning because Algorithm 3 is an online incremental learning method. When expanding Algorithm 3 to semi-supervised learning, we need only store teacher vectors in a memory buffer for finding teacher nodes in Step 3 of Algorithm 5. All learned unlabeled data can be discarded; after a period of learning, even the buffered teacher vectors can be released.

4.2 Active learning SOINN (A-SOINN)

As described in the preceding section, for high-density nodes, it is possible that such nodes lie in the central part of a class. Using Algorithm 4 we know that, if the central node of a subcluster is a teacher node, no other teacher node within the same subcluster influences the labeling of unlabeled nodes. Consequently, we merely need to set the central node as a teacher node, which is sufficient for the labeling task. Using Algorithm 1, we can use a subcluster or a group of subclusters to represent the input data. For that reason, we can actively set central nodes as teacher nodes in the order of group and subcluster. In other words, it is possible to use much fewer teacher nodes and yet realize semi-supervised active learning.

For low-density nodes, it is possible that such nodes lie in the overlapped area between classes, i.e., the neighboring nodes might have different labels. In this situation, we say that this area is the boundary between classes, and call such nodes "boundary nodes" (for example, in Fig. 6, the nodes linked by boundary edges are boundary nodes). We set the boundary nodes as teacher nodes and ask for their label if we want to realize high-precision learning.

Using the analysis presented above, Algorithm 6 summarizes the active learning process (query for teacher vectors).

Algorithm 6 Actively query for class label

1: Assume that the group set without a teacher node is $A_G = \{G_i\}_{i=1}^{m}$.
2: **if** $A_G \neq \emptyset$ **then**
3: Find group G with $G = \text{argmax}_{a \in A_G} |N_a|$. (% find biggest group)
4: Find node q with the highest density in group G: $q = \text{argmax}_{i \in N_G} D(i)$.
5: Go to Step 19.
6: **end if**
7: Assume that the subcluster set without a teacher node is $A_S = \{S_i\}_{i=1}^{l}$.
8: **if** $A_S \neq \emptyset$ **then**
9: Find subcluster S with $S = \text{argmax}_{a \in A_S} |N_a|$. (% find biggest subcluster)
10: Find the node with the highest density in subcluster S: $q = \text{argmax}_{i \in N_S} D(i)$.
11: Go to Step 19.
12: **end if**
13: Assume the boundary node set without a teacher node is N_B.
14: **if** $N_B = \emptyset$ **then**
15: Stop learning without the query.
16: **else**
17: Randomly choose one $q \in N_B$.
18: **end if**
19: Ask for the output label of the weight vector $\boldsymbol{W}_q$ of node q. Then stop.

With the active query process Algorithm 6, we expand Algorithm 5 to the active learning version in Algorithm 7.

Algorithm 7 Active learning with improved SOINN (A-SOINN)

1: Using Algorithm 3 to train all labeled and unlabeled data (both online and batch training are possible), generated nodes are separated into different groups and subclusters.
2: After a learning period (for batch learning, the next steps can be followed after the learning of Algorithm 3 is completed), use Algorithm 6 to query for the label of the central nodes of groups and subclusters. If the boundary edges exist, query for the label of the boundary nodes.
3: Using Algorithm 4 to label all nodes with teacher nodes found in Step 2.
4: If online learning is not finished, go to Step 1.

Compared to Algorithm 5, Step 2 of Algorithm 7 is different. This actively querying process ensures that we need fewer labeled samples for Algorithm 7 than when using Algorithm 5. The choosing of central nodes and the boundary nodes as the teacher nodes allows the system to achieve higher learning efficiency than by randomly choosing teacher nodes from the SOINN node set. Note that the system finds the teacher nodes in an online manner and the labels of teacher nodes are also queried in an online manner.

5 Experiment

5.1 Experiment using artificial data

In this experiment, we use artificial data as the input vector to perform the experiment. The dataset, which is shown in Fig. 8, comprises 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 the same class (class A); other distributions belong to different classes (B, C, and D). We also add 10% noise to the dataset. Here, noise means that the training dataset contains an unknown amount of noise in the features and class labels. The noise is distributed randomly throughout the whole feature space. We randomly label such noise samples by using class names.

In the first experiment, we perform the experiment under a stationary environment, i.e., in one iteration, choose an input vector randomly from the dataset. After 100,000 training iterations, according to Algorithm 6, the system asks for labels of 10 teacher nodes. The parameters of the

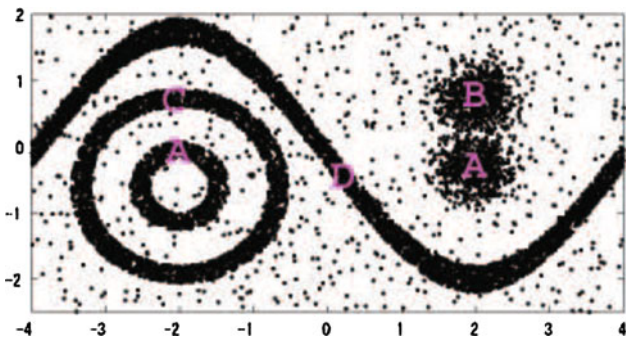

Fig. 8 Artificial data used for experiments

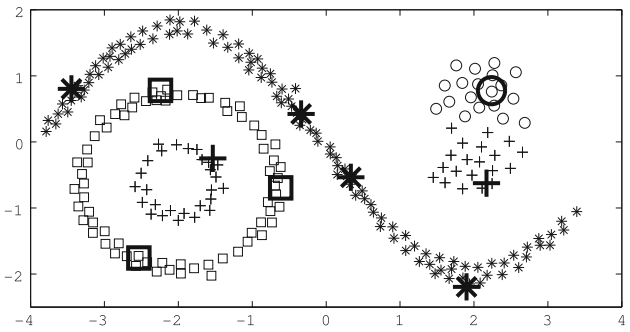

Fig. 9 Experimental results obtained using artificial data under a stationary environment. Unlabeled nodes are labeled with *different marks*; teacher nodes are labeled with *large marks*

proposed method are $\lambda = 500$, $\text{age}_{\max} = 30$, and $\alpha = 2.0$. The experimental results are shown in Fig. 9.

In Fig. 9, we use different marks to illustrate the nodes given with different labels. The teacher nodes that are queried by the system are denoted with large marks. Actually, Fig. 9 shows that the proposed method can remove the noise, represent the distribution of input data, and give suitable cluster labels to all nodes with only 10 labeled teacher nodes.

We also use different values of α to test the degree to which α is sensitive to the learning results. The results are shown in Table 1.

Table 1 shows that if a small α is adopted, the grouping of subclusters will be difficult, and there will be numerous subclusters. More subclusters are grouped following the increase in α. The grouping process becomes stable in the range $\alpha \in [2.0, 10.0]$. All subclusters will be grouped together if we adopt a very large value for α.

In the second experiment, we performed online incremental learning under a non-stationary environment, that is, the input data of classes are input incrementally to the system. Table 2 details specifications of the test environments. The environment changes from I to IV (i.e., there is concept drift in this task). In each environment, classes used to generate samples are marked as "1"; other classes are marked as "0." As presented in Table 2, we simulate online incremental learning by using the following paradigm: from iteration 1 to 50,000, samples are generated randomly from class A. From iteration 50,001, the environment changes and samples from class B are generated.

Table 1 Sensitivity analysis result of parameter α: stationary environment

α	0.01	0.5	1.0	1.5	2.0	4.0	10.0	40.0	100.0
No. of groups	30	21	12	6	4	4	4	2	1
No. of teachers	57	46	28	19	10	10	10	5	1

Table 2 Experimental environments for online incremental learning

	I	II	III	IV
A	1	0	1	0
B	0	1	0	1
C	0	0	1	0
D	0	0	0	1

From iteration 100,001, the environment changes again, and so on. The parameters are set as $\lambda = 250$, $\text{age}_{\max} = 40$, and $\alpha = 3.2$.

After every 50,000 training iterations, Algorithm 6 asks for the class label of some teacher nodes. Figure 10 presents results for every 50,000 training iterations. The system will generate nodes to represent the new distribution and ask for the label of teacher nodes of such new input classes if data from new classes are added to the system. From the top to the bottom of Fig. 10, the displayed results were obtained after 50,000, 100,000, 150,000, and 200,000 training iterations.

In Environment I, only class A is used to generate training samples. For a Gaussian distribution, the system only queried the label of one teacher node; for the ring, the system asks for labels of three teacher nodes. In Environment II, the probability of getting samples from class A changes to zero. Remaining nodes of class A play a major role in online learning. They preserve the knowledge of the previous situation for future decisions. In Environment III, the reappearance of class A does not change the learned results because knowledge is preserved completely. Consequently, no more teacher nodes are needed for labeling the reappeared class A. Environments II, III, and IV add new information to the system (classes B, C, and D); the system asks for labels of the new added information for some teacher nodes. The system realizes incremental learning without destroying the learned knowledge.

Compared to Fig. 9, the system depicted in Fig. 10 needs more teacher nodes and more nodes. This means that incremental learning with non-stationary environments is more difficult than with the stationary environment, and the system must have more nodes to preserve learned knowledge and learn new information. It also requires more teacher nodes to label those generated nodes.

For the incremental non-stationary environment, we also tested the sensitivity of parameter α. The results are presented in Table 3.

Table 3 shows that, as we know from the experiment with the stationary environment, small α leads to many groups, and very large α leads to only one group.

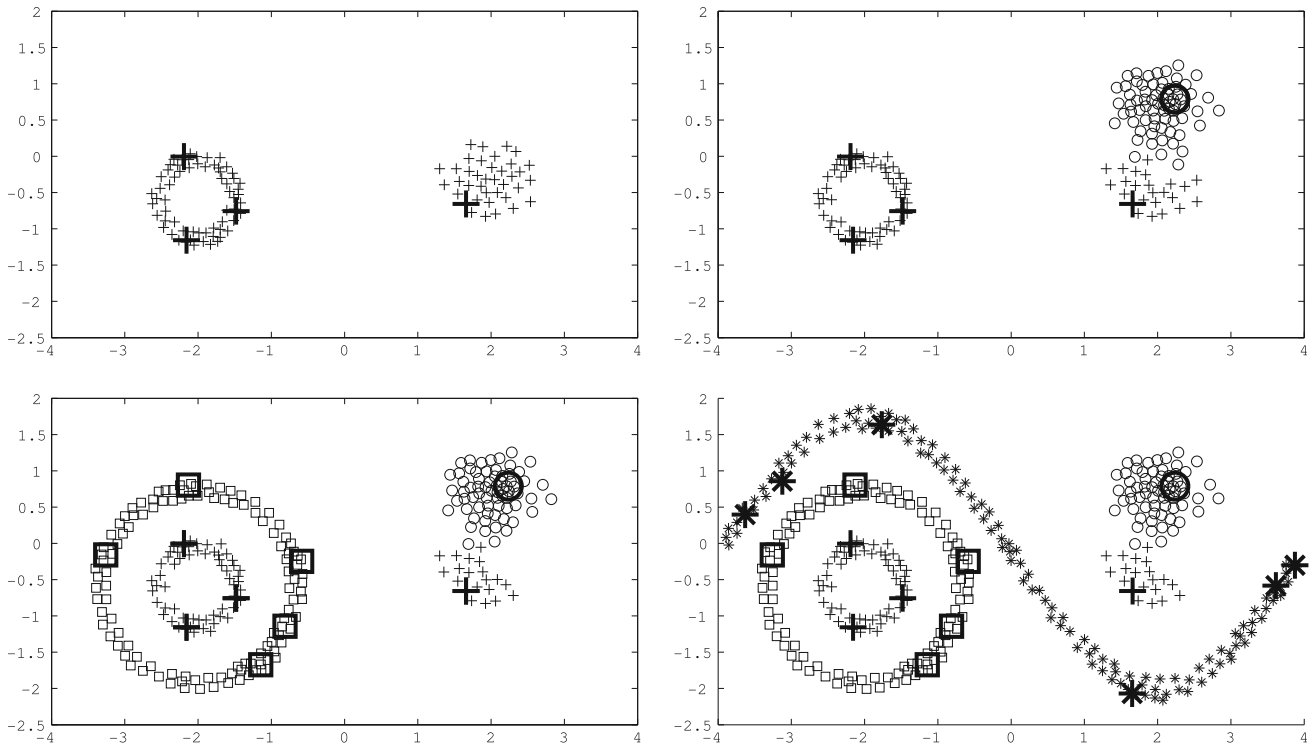

Fig. 10 Incremental learning results of artificial data. Unlabeled nodes are labeled with *different marks*; teacher nodes are labeled with *large marks*. From the *top left* to *down right*, the figures represent the learning results for 50,000, 100,000, 150,000, and 200,000 training iterations

Table 3 Sensitivity analysis result of parameter α: incremental non-stationary environment

α	0.5	1.0	1.5	2.0	2.5	3.2	6.0	40.0	100.0
No. of groups	26	17	9	7	6	4	4	2	1
No. of teachers	66	49	35	29	23	16	15	7	1

5.2 Experiment using real-world data

5.2.1 *Efficiency of active learning*

In this experiment, we use Optdigits [27] to test the efficiency of Algorithm 6. We compare the results of actively querying the teacher nodes and querying the teacher nodes randomly. Actually, Optdigits is a dataset comprising 3,823 training vectors and 1,797 testing vectors: every datum is a 64-dimensional handwritten digit vector. We use all training vectors as input data. After 200,000 training iterations, we actively query teacher nodes according to Algorithm 6 and report the results. Then we randomly choose teacher nodes and report the results. We use the same parameters for both experiments: $\lambda = 10{,}000$, $\text{age}_{\max} = 100$, $\alpha = 1.0$. We conduct 100 tests and take the average as the final result, as shown in Fig. 11. In Fig. 11, the solid line represents the results of active queries, and the dotted line shows the results of random queries. The vertical axis shows the recognition ratio, and the horizontal axis shows the number of teacher vectors. As depicted in Fig. 11, with the same number of teacher vectors, active queries obtain a much higher recognition ratio than random queries. Following the increase in teacher vectors, the recognition ratio of active queries increases much faster than random queries. These results show that, by using Algorithm 6, active learning can achieve highly efficient learning.

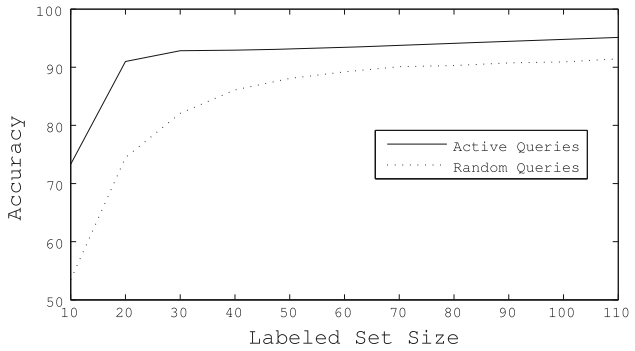

Fig. 11 For Optdigits, actively querying teacher vectors, and randomly querying teacher vectors: a comparison. The *solid line* represents the results of active queries; the *dotted line* shows the results of random queries. The *vertical axis* shows the recognition ratio. The *horizontal axis* shows the number of queried teacher vectors

5.2.2 Experiment for incremental learning

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

The input data change in the order of odd digits, even digits, and all digits for every 50,000 iterations. After every 10,000 learning iterations, the system queries for labels of teacher nodes by using Algorithm 6. The parameters are set as $\lambda = 10{,}000$, $\text{age}_{\max} = 100$, and $\alpha = 1.0$. Figures 12 and 13 depict those results. Here, we adopt an average of the obtained results through 100 times of learning. After the first 50,000 training iterations (only odd digits are trained), we used only odd digits in the testing data to calculate the recognition ratio. Figure 12 shows that when new classes are introduced into the system (e.g., from the 50,001st training iteration, even digits are input into the system for training), the necessary teacher vectors increase drastically. Figure 13 shows that when new classes are input to the system, the recognition ratio decreases (we now use all digits in the testing dataset to calculate the recognition ratio). The system judges that unknown classes are in the input data. Following the learning process, the system automatically determines how many teacher vectors are necessary to learn the new classes. During the incremented 50,000 learning iterations, the recognition ratio is increased because, following the learning process, the proposed method can increasingly approximate the input distribution better. The learned knowledge of odd digits is preserved; such knowledge is useful to classify testing samples from odd digits. Results show that, under an incremental learning environment (there exists concept drift), the proposed method can maintain a good recognition ratio, and other semi-supervised active learning methods are not suitable for this task.

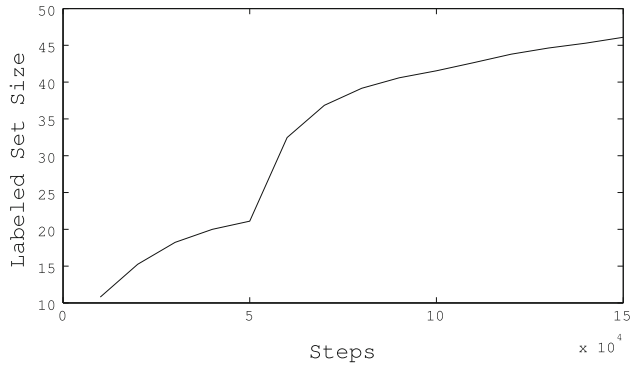

Fig. 12 For Optdigits, new classes are added incrementally to the system. This figure shows the relationship between training iterations and the queried amount of teacher vectors

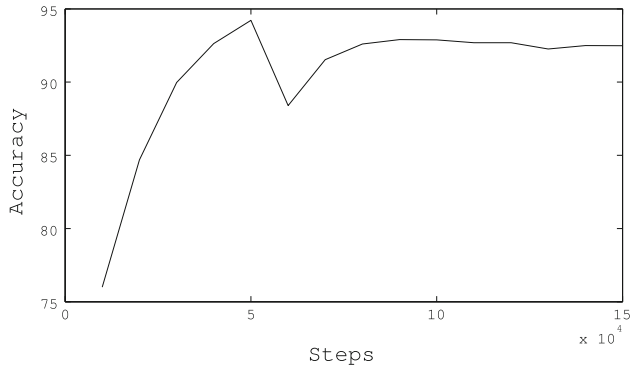

Fig. 13 For Optdigits, new classes are added incrementally to the system. This figure shows the relationship between training iterations and the recognition ratio

5.2.3 Online incremental learning with facial images

In this experiment, we incrementally train images of 10 people one by one by using online facial images taken from a fixed video camera. Facial images are taken at 30 frames per second. As a pre-process for the raw video images, we transform the color images to 256 gray-scale images and then resample the images to 20×26 pixel images (Fig. 14). Such images are used as input vectors (520 dimensional vectors).

During the experiment, the candidates progressively change the direction they are facing (Fig. 15). For every person, 3,000 frames are used as training data; the other 700 frames are used as testing data. The parameters used in this experiment are $\lambda = 2{,}000$, $\text{age}_{\max} = 200$, and $\alpha = 1.0$. After every 2,000 training iterations, the system queries for a label of teacher nodes according to Algorithm 6. The results (average of 100 times learning) are presented in Fig. 16: the upper figure depicts the required number of teacher vectors; the lower figure portrays the recognition accuracy. The lower figure also shows that when unknown faces appear, the recognition accuracy is very low. The system cannot recognize such unknown faces. After learning the unknown faces, the recognition accuracy becomes normal: The system learns unknown faces very well and retains knowledge of the learned faces, i.e., it can realize the incremental online learning very well. After 30,000 training iterations, the required number of nodes is 290; the required number of teacher vectors is 89. Use of the complete storage method requires storage of all 30,000 input data, and the proposed method uses only 1/100 of the memory and calculation of the complete storage method.

Results of this experiment show that, by using complicated high-dimensional real-world data, the proposed method is able to achieve incremental online semi-supervised active learning with good recognition accuracy and a good compression ratio.

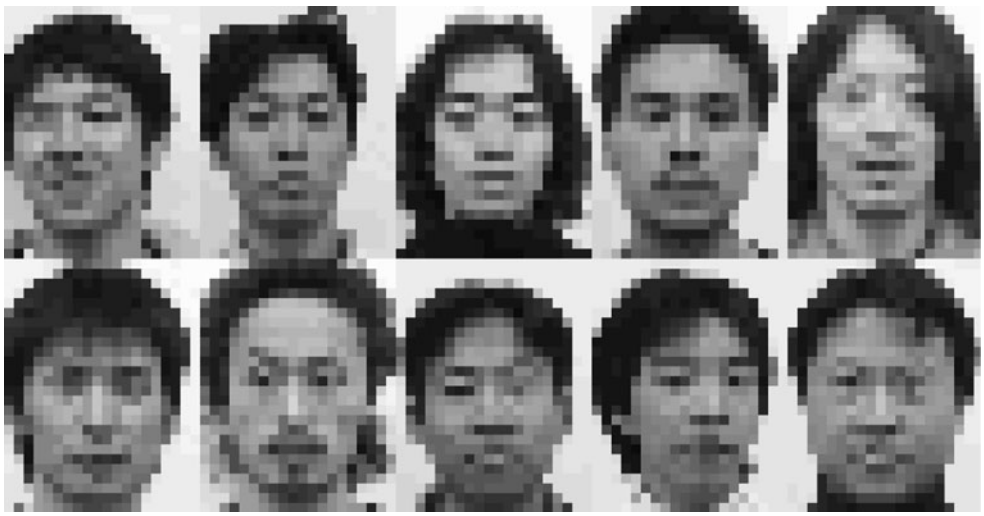

Fig. 14 Facial images of 10 people

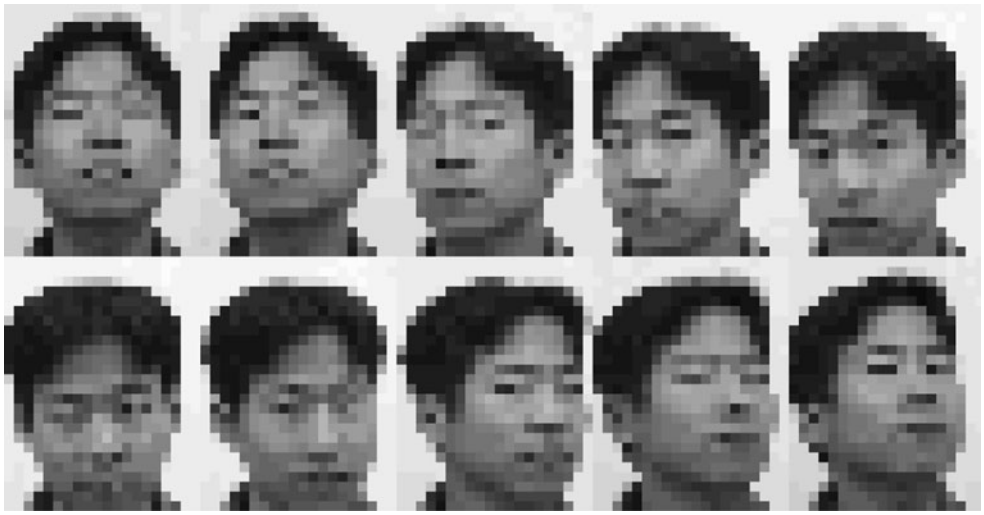

Fig. 15 Facial images used for experiments. The facial orientation changes gradually

5.3 Comparison to other methods

In this experiment, we compare the proposed method with support vector machine (SVM) [28], original SOINN [19], growing neural gas (GNG) [29], and semi-supervised GNG (SSGNG) [4]. For the input dataset, we perform the same pre-process for the ORL face database as in [4]: The original 92 × 112 images are reduced to 46 × 56 images, and then Principal Component Analysis (PCA) was applied to all the images to reduce the number of dimensions from 2,576 to 60 (corresponding to 86.65% of the total variance). There are 40 subjects and 10 different frontal images for each subject in the ORL face database. In this experiment, the ORL dataset is randomly partitioned to have six training images and four testing images for each subject. The training dataset is then randomly partitioned to have a varying number of labeled images, from three, two to one for each subject to see the effect of different amount of labeled and unlabeled data on SVM, SOINN, GNG, SSGNG, and S-SOINN algorithms. For A-SOINN algorithm, the system will actively query the labeled images.

As in [4], the classification accuracy is measured by how many data points or images from the test set are correctly assigned to their true classes. To determine which class would be assigned to the presented testing sample, the Euclidean distance between the data and each weight in the network is measured. The node with the weight vector having the shortest distance is the winning node and the testing sample is assigned to the class of that particular node. The assigned classes of the test set are then compared

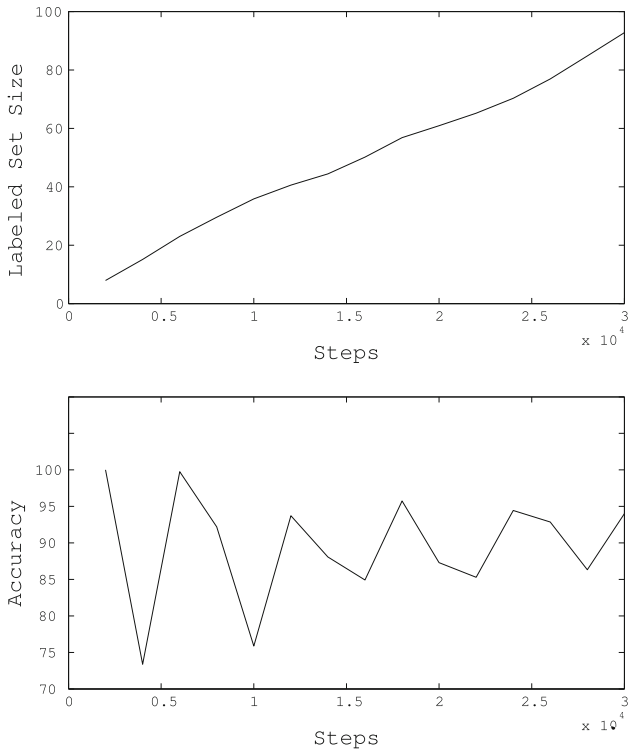

Fig. 16 Learning results of facial images. The *vertical axis* of the *upper figure* denotes the queried amount of teacher vectors. The *vertical axis* of the *lower figure* shows the recognition accuracy. *Horizontal axes* of *both figures* show the training iterations. For every 2,000 training iterations, the system queries for the label of teacher nodes

to their original classes to determine the classification accuracy. The parameters for GNG and SSGNG algorithms are chosen on a trial-and-error basis as described in [4]: $\lambda = 300$, $a_{\max} = 100$, $e_b = 0.2$, $e_n = 0.006$, $\alpha = 0.5$, $\beta = 0.995$. The growth of the GNG network was stopped when the network size reached 120 nodes. For SOINN and the proposed method, the parameters are set as $\lambda = 25$, $\text{age}_{\max} = 25$, $\alpha = 2.0$. For SVM, Matlab library LIBSVM [28] is used to train the SVM classifier and Gaussian Kernel is adopted. Even if there are six training samples for every subject, only labeled samples are used to train SVM, then four testing images are used to test the classification accuracy of SVM. Table 4 represents the experimental results.

In Table 4, L denotes training with labeled data only, while $L + U$ denotes training with labeled and unlabeled data. S-SOINN means semi-supervised SOINN; Algorithm 5 is adopted for S-SOINN. A-SOINN means active learning SOINN; Algorithm 7 is adopted for A-SOINN. The results for GNG, SSGNG, SOINN, S-SOINN, and A-SOINN are the average percentages of the classification performance over 10 independent experiments. For GNG and SSGNG, the total number of nodes is predetermined as 120. For SOINN, S-SOINN and A-SOINN, the total number of nodes is automatically generated. In this experiment, the generated nodes for SOINN, S-SOINN, and A-SOINN are 120 ± 6. For SVM, GNG, and SOINN, there are no unlabeled data as inputs. For SSGNG and S-SOINN, labeled and unlabeled data are used as inputs, and labeled data are predetermined. For A-SOINN, all training data are used as unlabeled data to train the A-SOINN system, and A-SOINN actively queries the label of some generated nodes. The number of queried nodes (average L per subject) is listed in column 8 of Table 4.

For SVM, GNG, and SOINN, only labeled data are adopted for training, the classification performance is less than SSGNG, S-SOINN, and A-SOINN. SSGNG is a semi-supervised version of GNG, it works better than GNG. S-SOINN (Algorithm 5) is a semi-supervised version of SOINN, with the help of unlabeled data, S-SOINN also works better than SOINN. For A-SOINN, it actively queries the labeled data. With nearly the same amount of labeled data, A-SOINN works better than S-SOINN, it shows the efficiency of active learning. Also, compared with the previous study on SSGNG, the proposed S-SOINN and A-SOINN perform better.

From the experiment results, we can conclude that the use of unlabeled data in SSGNG and S-SOINN markedly improves the performance of the classifier. Actively queried labeled data in A-SOINN are able to perform better than randomly queried labeled data.

6 Conclusion

This paper presented an online incremental semi-supervised active learning method based on SOINN. The

Table 4 Classification rates on the test set for the ORL face database

Data size		Classification accuracy						
L	U	L only			$L + U$		A-SOINN	
		SVM (%)	GNG (%)	SOINN (%)	SSGNG (%)	S-SOINN (%)	Queried L	Accuracy (%)
3	3	84.13	82.47	83.34	84.88	84.95	3.2	85.12
2	1	79.38	76.82	76.78	80.44	81.23	2.1	81.46
1	5	67.75	65.97	67.12	69.00	69.33	1.1	70.10

proposed method improves two-layer SOINN to single-layer SOINN to represent the topological structure of input data, separates the generated nodes into different groups and subclusters, actively labels some teacher nodes, and uses such teacher nodes to label all unlabeled nodes. The benefits of the proposed method are that it requires no prior knowledge such as the number of nodes or number of classes; consequently, it can automatically learn the number of nodes and teacher vectors needed to accomplish current tasks. Most important, it can realize online incremental learning, and even life-long learning, which is impossible for some other semi-supervised active learning methods. The proposed method can also alleviate the influence of noise. The experimental results verify that the proposed method works both effectively and efficiently.

Some problems remain unresolved. For example, λ determines when the online learning starts clustering, separating subclusters, and grouping over-separated subclusters. In addition, α influences the grouping of subclusters. Such parameters depend heavily on the real environment. For this study, we let the user determine the parameters and gave no theoretical standard for determining such parameters. Future studies will be designed to solve this problem.

Acknowledgements This work was supported in part by the Fund of the National Natural Science Foundation of China (Grant No. 60975047, 60723003, 60721002), 973 Program (2010CB327903), and Jiangsu NSF grant (\#BK2009080). This study was also supported in part by the New Energy and Industrial Technology Development Organization (NEDO) of Japan.

References

1. Cohen I, Cozman FG, Sebe N, Cirelo MC, Huang TS (2004) Semisupervised learning of classifiers: theory, algorithms, and their application to human-computer interaction. IEEE Trans Pattern Anal Mach Intell 26(12):1553–1566
2. Baraldi A, Bruzzone L, Blonda P (2006) A multiscale expectation-maximization semisupervised classifier suitable for badly posed image classification. IEEE Trans Image Process 15(8): 2208–2225
3. Yeung D-Y, Chang H (2007) A kernel approach for semisupervised metric learning. IEEE Trans Neural Netw 18(1):141–149
4. Zaki SM, Yin H (2008) A semi-supersized learning algorithm for growing neural gas in face recognition. J Math Model Algorithms 7:425–435
5. Freund Y, Seung H, Shamir E, Tishby N (1997) Selective sampling using the query by committee algorithm. Mach Learn 28:133–168
6. Muslea I, Minston S, Knoblock C (2000) Selective sampling with redundant views. In: Proceedings of the national conference on artificial intelligence, pp 621–626
7. Kothari R, Jain V (2003) Learning from labeled and unlabeled data using a minimal number of queries. IEEE Trans Neural Netw 14(6):1496–1505
8. Mingkun L, Sethi IK (2006) Confidence-based active learning. IEEE Trans Pattern Anal Mach Intell 28(8):1251–1261
9. Muslea I, Minston S, Knoblock C (2002) Active + semi-supervised learning = robust multi-view learning. In: Proceedings of ICML-02, 19th international conference on machine learning, pp 435–442
10. Nigam K, McCallum AK, Thrum S, Mitchell T (2000) Text classification from labeled and unlabeled documents using em. Mach Learn 39:103–134
11. Blum A, Mitchell T (1998) Combining labeled and unlabeled data with co-training. In: Proceedings of the conference on computational learning theory, pp 92–100
12. Bennett K, Demiriz A (1999) Semi-supervised support vector machines. Adv Neural Inf Process Syst 11:368–374
13. Zhu X, Ghahramani Z, Lafferty J (2003) Semi-supervised learning using gaussian fields and harmonic functions. In: ICML-03, 20th international conference on machine learning, pp 912–919
14. Zhu X, Lafferty J, Ghahramani Z (2003) Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In: ICML 2003 workshop on the continuum from labeled to unlabeled data in machine learning and data mining, pp 58–65
15. Wang Z, Song Y, Zhang C (2009) Efficient active learning with boosting. In: SDM 2009, pp 1230–1241
16. Huang R, Lam W (2009) An active learning framework for semi-supervised document clustering with language modeling. Data Knowl Eng 68(1):49–67
17. Hoi SCH, Jin R, Zhu J, Lyu MR (2009) Semi-supervised SVM batch mode active learning with applications to image retrieval. ACM Trans Inf Syst 27(3):16:1–16:29
18. Carpenter GA, Grossberg S (1988) The art of adaptive pattern recognition by a self-organizing neural network. IEEE Comput 21:77–88
19. Shen F, Hasegawa O (2006) An incremental network for on-line unsupervised classification and topology learning. Neural Netw 19:90–106
20. Kamiya Y, Shen F, Hasegawa O (2007) An incremental neural network for online supervised learning and topology learning. J Adv Comput Intell Intell Inform 11(1):87–95
21. Shen F, Hasegawa O (2008) A fast nearest neighbor classifier based on self-organizing incremental neural network. Neural Netw 21:1537–1547
22. Sudo A, Sato A, Hasegawa O (2007) Associative memory for online incremental learning ina noisy environment. In: The 2007 international joint conference on neural networks
23. Shen F, Sudo A, Hasegawa O (2010) An online incremental learning pattern-based reasoning system. Neural Netw 23(1):135–143
24. Shen F, Ogura T, Hasegawa O (2007) An enhanced self-organizing incremental neural network for online unsupervised learning. Neural Netw 20:893–903
25. He X, Kojima R, Hasegawa O (2007) Developmental word grounding through a growing neural network with a humanoid robot. IEEE Trans Syst Man Cybern B 37(2):451–462
26. He X, Ogura T, Satou A, Hasegawa O (2007) Developmental word acquisition and grammar learning by humanoid robots through a self-organizing incremental neural network. IEEE Trans Syst Man Cybern B 37(5):1357–1372
27. Merz C, Murphy M (1996) Uci repository of machine learning database. University of California Department of Information, Irvine
28. Chang C-C, Lin C-J (2001) IBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/cjlin/libsvm
29. Fritzke B (1995) A growing neural gas network learns topology. In: Advances in neural information processing systems 7. MIT, Cambridge