Neural Networks 84 (2016) 143–160

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

A Self-Organizing Incremental Neural Network based on local distribution learning

Youlu Xing [a,b], Xiaofeng Shi [a], Furao Shen [a,*], Ke Zhou [c], Jinxi Zhao [a]

[a] The National Key Laboratory for Novel Software Technology, Nanjing University, China
[b] School of Computer Science and Technology, Anhui University, Hefei, 230601, China
[c] School of Statistics at University of International Business and Economics, Beijing, China

ARTICLE INFO

Article history:
Received 6 January 2016
Received in revised form 25 August 2016
Accepted 26 August 2016
Available online 13 September 2016

Keywords:
Incremental learning
Matrix learning
Self-Organizing Incremental Neural Network (SOINN)
Relaxation data representation

ABSTRACT

In this paper, we propose an unsupervised incremental learning neural network based on local distribution learning, which is called Local Distribution Self-Organizing Incremental Neural Network (LD-SOINN). The LD-SOINN combines the advantages of incremental learning and matrix learning. It can automatically discover suitable nodes to fit the learning data in an incremental way without a priori knowledge such as the structure of the network. The nodes of the network store rich local information regarding the learning data. The adaptive vigilance parameter guarantees that LD-SOINN is able to add new nodes for new knowledge automatically and the number of nodes will not grow unlimitedly. While the learning process continues, nodes that are close to each other and have similar principal components are merged to obtain a concise local representation, which we call a relaxation data representation. A denoising process based on density is designed to reduce the influence of noise. Experiments show that the LD-SOINN performs well on both artificial and real-word data.

1. Introduction

Unsupervised learning refers to the problem of finding hidden knowledge in unlabeled data. The *k*-means (Lloyd, 1982) and Neural Gas (NG) (Martinetz, Berkovich, & Schulten, 1993) aim to use some given nodes to represent the distribution of the learning data with low quantization error and get different clusters in the learning data. Self-Organizing Map (SOM) (Kohonen, 1982) and Topology Representing Networks (TRN) (Martinetz & Schulten, 1994) represent the distribution of learning data with some given nodes; moreover, they organize these nodes to map the topological structure of the learning data. In these methods, each node stores the mean feature vector of patterns belonging to the node and the metric method is the Euclidean distance. Correspondingly, the node tends to have isotropic form with spherical class boundary. However, spherical class boundary usually does not conform to the actual situation. In order to represent the original learning data faithfully, it is necessary to record the local data distribution information around each node.

The matrix learning methods (Arnonkijpanich, Hasenfuss, & Hammer, 2011; Huang, Yi, & Pu, 2008; Kohonen, 1995; López-Rubio, Muñoz-Pérez, & Gómez-Ruiz, 2004) record rich local distribution information and consider the anisotropy on different basis vectors. Adaptive-Subspace Self-Organizing Map (ASSOM) (Kohonen, 1995) assigns each node a linear-subspace represented by some basis vectors, and the distance metric of ASSOM is the orthogonal projection distance. The Principal Components Analysis Self-Organizing Map (PCASOM) (López-Rubio et al., 2004) stores the covariance matrix, replacing the orthogonal vector basis, to represent the subspace in each node and use Mahalanobis distance as the distance metric, which considers the anisotropy on different basis vectors. The local PCA-SOM (Huang et al., 2008) records the mean vector and covariance matrix in each node and uses the normalized Rayleigh ratios as the competition measure, which implicitly incorporates the reconstruction error and distance between the input data and the node center. Matrix NG and Matrix SOM (Arnonkijpanich et al., 2011) use a generalized Mahalanobis distance metric and extend the standard cost functions of NG and SOM (by Heskes, 2001) based on the metric. López-Rubio (2010) gives a detailed review about these Mixture-Model based Self-Organizing Maps, which they call the Probabilistic Self-Organizing Map.

* Corresponding author.
E-mail addresses: youluxing@sina.com (Y. Xing), cell000feng@gmail.com (X. Shi), frshen@nju.edu.cn (F. Shen), 02417@uibe.edu.cn (K. Zhou), jxzhao@nju.edu.cn (J. Zhao).

http://dx.doi.org/10.1016/j.neunet.2016.08.011
0893-6080/

The above mentioned matrix learning methods have a common shortcoming, i.e., they cannot deal with the Stability–Plasticity Dilemma (Carpenter & Grossberg, 1988). Many of these methods (Huang et al., 2008; Kohonen, 1995; López-Rubio et al., 2004) cannot learn new classes or distributions data after they are trained on the current data set. Some other methods (Arnonkijpanich et al., 2011; López-Rubio, 2010) are able to learn new classes or distributions data, but the previous learned knowledge will be forgotten. Unfortunately, in many real-world applications, patterns from new classes may arise at any time; thus, these matrix learning methods cannot be adopted for these applications.

In order to solve the Stability–Plasticity Dilemma, many "growing networks" or "incremental networks" are designed. Growing Self-Organizing Map (GSOM) (Alahakoon, Halgamuge, & Srinivasan, 1998, 2000) will grow a boundary node when the total error of the node is larger than a predefined growth threshold. Growing Cell Structure (GCS) (Fritzke, 1994) and Growing Neural Gas (GNG) (Fritzke, 1995) insert new node(s) after every λ patterns learned, where λ is a constant parameter. However, in these methods, during each λ period, the input pattern is forced to merge with a node no matter how big the gap between them. Considering the physical meaning, it is unreasonable to merge two patterns with significant difference.

Some neural network models that establish new nodes for input pattern under some effective judgment mechanism in real-time can handle this problem. Grow When Required (GWR) (Marsland, Shapiro, & Nehmzow, 2002) network can add nodes whenever the network in its current state does not sufficiently match the input. The Self-Organizing Adaptive Map (SOAM) (Piastra, 2009) develops the GWR; it adds a new node when the Euclidean distance between the input pattern and the nearest node to the input pattern is larger than a threshold. The Adaptive Resonance Theory network (ART) (Carpenter & Grossberg, 1988), Fuzzy ART (Carpenter, Grossberg, & Rosen, 1991), Ellipsoid-ART (Anagnostopoulos & Georgiopoulos, 2001) and TopoART (Tscherepanow, Kortkamp, & Kammer, 2011) (ART family) create a new node for the input pattern when there is no match between the current input pattern and the current set of categories. The Self-Organizing Incremental Neural Network (SOINN) (Shen & Hasegawa, 2006), Enhanced-SOINN (Shen, Ogurab, & Hasegawa, 2007), Adjusted-SOINN (Shen & Hasegawa, 2008) and LB-SOINN (Zhang, Xiao, & Hasegawa, 2014) decide whether to create a new node for the input pattern according to the node distribution around the local region of the input pattern. Araujo and Rego (2013) give a detailed review about these incremental Self-Organizing Maps.

Comparing with the matrix learning methods (Arnonkijpanich et al., 2011; Huang et al., 2008; Kohonen, 1995; López-Rubio et al., 2004), these "growing neural networks" and "incremental networks" (Alahakoon et al., 1998, 2000; Anagnostopoulos & Georgiopoulos, 2001; Carpenter et al., 1991; Fritzke, 1994, 1995; Marsland et al., 2002; Piastra, 2009; Shen & Hasegawa, 2006, 2008; Shen et al., 2007; Tscherepanow et al., 2011; Zhang et al., 2014) lose much of the useful information of the original learning data during learning. Recently, an online Kernel Density Estimator (oKDE) (Kristan, Leonardis, & Skočaj, 2011) was proposed to introduce matrix learning to online learning. However, the learning environment (stationary or non-stationary environments) must be known in advance to set different parameters. Moreover, a combination of SOINN and matrix learning, namely SOINN-AML (Okada & Nishida, 2011), was proposed to enhance the performance of SOINN by optimizing the distance matrix of the nodes with Adaptive Metric Learning (Ye, Zhao, & Liu, 2007) algorithm (AML). In SOINN-AML, the global covariance matrix learning is done in some of the intervals. However, in the incremental learning tasks, updating the global covariance matrix is likely to undermine the stability of the network, as learning a novel pattern would consequently change the distance metric of entire prior knowledge. Thus, updating the local covariance matrix of each node rather than the global covariance matrix will keep the network more stable.

In our previous work, we have proposed Local-SOINN (Ouyang, Shen, & Zhao, 2012) and ILDN (Xing, Cao, Shen, & Zhao, 2015) to combine the advantages of local matrix learning and SOINN together. This paper is an extended version of ILDN, in which we introduce a topology in the data space based on Hebbian Learning (Martinetz & Schulten, 1994) and a clustering strategy that harnesses such topology. Besides, in this paper, we make more analyses and comparative experiments to prove the validness of the network.

In this paper, we propose a Local Distribution Self-Organizing Incremental Neural Network (LD-SOINN) to combine the advantages of matrix learning and incremental learning by the following processes:

(1) By storing the covariance matrix and merging nodes that are close to each other and have similar principal components, LD-SOINN gets a concise representation of the learning data, which is called a *relaxation* data representation.
(2) Through giving each node an adaptive vigilance parameter with statistics theoretical support, LD-SOINN is able to add new nodes for new knowledge automatically, and the number of nodes will not grow unlimitedly.
(3) By calculating the mean density of the nodes and deleting low density nodes, LD-SOINN is robust to noise.
(4) Through finding all different connected node domains, LD-SOINN can automatically cluster without a predefined number of clusters.

The rest of this paper is organized as follows. Section 2 describes the LD-SOINN; theoretical analysis is presented in Section 3; Section 4 gives experimental results on some artificial and real-world data sets; and Section 5 concludes the paper.

2. LD-SOINN

2.1. Overview of LD-SOINN

As Fig. 1 shows, LD-SOINN inherits and develops the Self-Organizing Map. The patterns that are neighboring on the feature space, i.e., "similar patterns", are mapped to the same node or to nodes that are adjacent in the network. There are two layers of LD-SOINN: input layer and competitive layer. The input layer is used to receive external data and the competitive layer is used to process the received data and record the learning result. The structure of the competitive layer is not predefined (such as the node number and the topology structure); it is automatically obtained with the learning of external data in an incremental way. Different from SOM, LD-SOINN uses the winner-take-all method to update the winning node. It does not make use of a neighborhood function to adapt to the neighbors of the winner node. The nodes in the LD-SOINN not only record the weight vector but also record the data distribution information around its local region, i.e., the covariance matrix, which is represented by the ellipsoid in the figure. The nodes that are close to each other in the feature space are connected. The connected nodes will be merged during learning if a concise data representation can be obtained, which we call a *relaxation* data representation.

Each node i in the competitive layer is associated with a 4-tuple $\langle c_i, M_i, n_i, H_i\rangle$: c_i, M_i and n_i are the mean vector, covariance matrix and number of input patterns belonging to i. H_i is a vigilance parameter to decide whether an input pattern belongs to i; it dynamically changes with the learning process. Assuming that LD-SOINN receives d-dimensional data $x \in \mathbb{R}^d$, the node in LD-SOINN can be described as a hyper-ellipsoid region using c_i, M_i and H_i:

$$i : \sqrt{(x - c_i)^T M_i^{-1} (x - c_i)} < H_i \quad x \in \mathbb{R}^d. \tag{1}$$

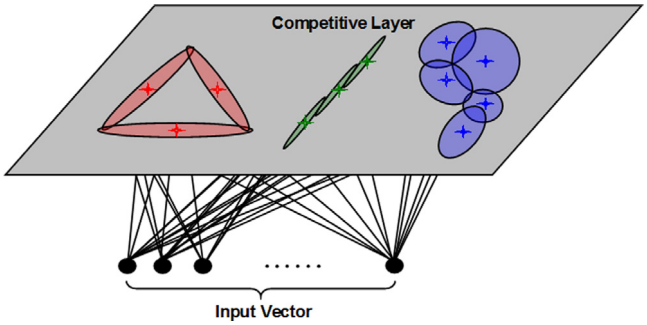


Fig. 1. Structure of LD-SOINN.

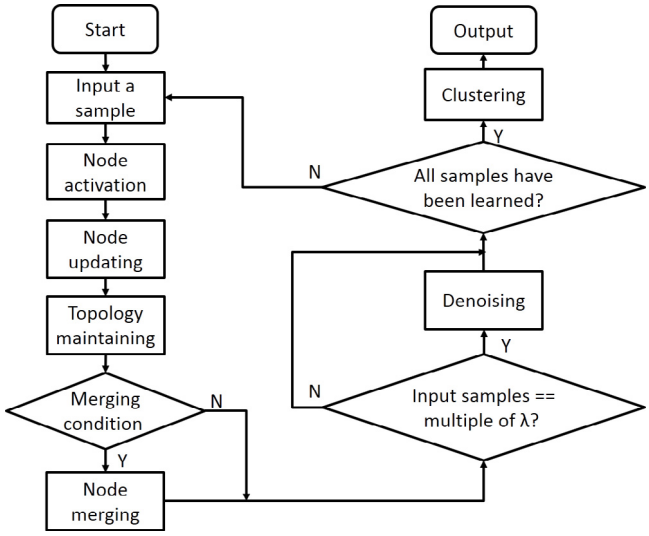


Fig. 2. The flowchart of the LD-SOINN.

In order to learn the topology structure, LD-SOINN organizes the nodes in the competitive layer to represent different topological structures by a connection list C. If two nodes i and j are activated by one pattern, a connection between i and j is created and added to C (Hebbian learning rule). When the learning process is finished, LD-SOINN will find all different connected node domains according to C to get different clusters.

The entire work flow of LD-SOINN is shown in Fig. 2. When an input pattern is received, LD-SOINN first conducts node activation to find some activated nodes that are recorded in an activating node set S. Then node updating is conducted according to set S. If there is no activated node in S, a new node will be established for this input pattern; else LD-SOINN will find a winner among the activated nodes and update this winner node. Topology maintaining module will create connections between the nodes in S and record these connections in the connection list set C. After that, LD-SOINN will check the merging condition between the winner node and its neighbor nodes. If the merging condition is satisfied, node merging between the winner node and its neighbors will be executed to get a concise local representation. Denoising is implemented for every λ patterns learned. When the learning process is finished, LD-SOINN will cluster the learned nodes and output the learning result.

Below, we give the details of the LD-SOINN.

2.2. Node activation

When an input pattern x is received, LD-SOINN first calculates the Mahalanobis distance between x and all nodes i in the competitive layer:

$$D_i(x) = \sqrt{(x - c_i)^T M_i^{-1}(x - c_i)}, \quad i = 1, 2, \ldots, |N| \tag{2}$$

where N is the set of nodes in the competitive layer, $|N|$ represents the total number of nodes. If $D_i(x) < H_i$, we say node i is activated. Then we put i in an activating set S and we get:

$$S = \{i | D_i(x) < H_i\}. \tag{3}$$

Then set S records all the activated nodes by the input pattern x.

Note that when we calculate the Mahalanobis distance between x and node i, the covariance matrix M_i is used. Although the estimation of covariance matrix from a small amount of local data samples around the node i sometimes yields over fitting to the samples, LD-SOINN does not get the over fitting problem in global data representation. As the learning continues, LD-SOINN merges the nodes that are close to each other and have similar principal components to obtain a concise and smooth representation, which avoids overfitting to a few local samples. Besides, in some intervals LD-SOINN executes the Denoising step to remove the nodes containing fewer patterns (we regard these nodes as noisy nodes). Hence, if a node is insufficiently trained and over fitted to the samples, LD-SOINN is likely to remove such node in the Denoising step. The details of Merging and Denoising steps are described in Sections 2.5 and 2.6.

2.3. Node updating

If $S = \emptyset$, i.e., no node is activated by x, it means x is a new knowledge. A new node a is created for x as:

$$a : \langle c_a = x, M_a = \sigma I, n_a = 1, H_a = \varepsilon_{n_a} * \chi^2_{d,q} \rangle. \tag{4}$$

To make M_a nonsingular, we initialize it as σI, where I is the identity matrix and σ is a small positive parameter. This initialization ensures that the covariance matrix M_a is positive definite during learning. A small positive σ guarantees that the initial hyper-ellipsoid is in compact convergence with the input pattern x; it decides the initial hyper-ellipsoid size of the new node. ε_{n_a} is a function of n_a to control the expansion trend of the ellipsoid. $\chi^2_{d,q}$ is a value of χ^2 distribution with d degrees of freedom and q confidence; usually q is equal to 0.90 or 0.95. The details of such parameters will be discussed in Section 3.

If $S \neq \emptyset$, i.e., some nodes are activated by x, it means x is not new knowledge. LD-SOINN will find a winner node i^* from set S:

$$i^* = \underset{i \in S}{\operatorname{argmin}} D_i(x). \tag{5}$$

Then node $i^* : \langle c, M, n, H \rangle$ is updated in a recursive way (the formula is proved in Appendix A):

$$\begin{aligned} n_{new} &= n + 1 \\ c_{new} &= c + (x - c)/(n + 1) \\ M_{new} &= M + [n(x - c)(x - c)^T - (n + 1)M]/(n + 1)^2 \\ H_{new} &= \varepsilon_{n_{new}} \chi^2_{d,q}. \end{aligned} \tag{6}$$

2.4. Topology maintaining

A topology preserving feature map is determined by a mapping Φ from a manifold $\mathcal{M}$ onto the vertices (neural nodes) $i, i = 1, 2, \ldots, N$ of a graph (network) $\mathcal{G}$. The mapping Φ is neighborhood preserving if similar feature vectors are mapped to vertices that are close within the graph. This requires that feature vectors v_i and v_j that are neighboring on the feature manifold $\mathcal{M}$ are assigned to vertices (nodes) i and j that are adjacent (connected) in the graph (network) $\mathcal{G}$ (Martinetz & Schulten, 1994).

Some methods, such as SOM (Kohonen, 1982) (structure of the network is usually defined as a two-dimensional grid), achieve the topology preserving feature map through a predefined structure $\mathcal{G}$. In this paper, the connections between nodes of $\mathcal{G}$ (network) are built in an adaptive way according to Hebbian learning rule: If two nodes are activated by one pattern, a connection between the two nodes is created.

According to the definition of set S, we know that all nodes in S are activated by the current pattern x. Thus, if no connection exists between nodes i and j in S, LD-SOINN will add a new connection $\{\langle i, j\rangle | i \in S \wedge j \in S \wedge i \neq j\}$ into the connection list C.

After a period of learning, these connections are able to organize the nodes into groups to represent different topologies of the learning data.

In topology preserving feature map (Martinetz & Schulten, 1994; Villmann, Der, Herrmann, & Martinetz, 1997), the definition of neighborhood of vertices is described as: two synaptic weight vectors w_i and w_j are adjacent on the manifold $\mathcal{M}$ if and only if their receptive fields R_i and R_j on $\mathcal{M}$ are adjacent. Where R_i and R_j are determined by masked Voronoi polygons $R_i = \tilde{V}_i$ and $R_j = \tilde{V}_j$, namely

$$\tilde{V}_i = \left\{v \in M \mid \|v - w_i\| \leq \left\|v - w_j\right\|, \forall j \in \mathcal{G}\right\} \tag{7}$$

in Euclidean space.

To define the same neighborhood relationship in LD-SOINN, where distance between vertices is calculated by Mahalanobis distance, we use the following equation to represent the Voronoi cell of node i:

$$\tilde{V}_i = \left\{v \in M \mid D_i(v) \leq D_j(v), \forall j \in \mathcal{G}\right\} \tag{8}$$

where $D_i(v)$ is defined in Eq. (2). We can prove that this definition is valid because the Voronoi cells are coherent in $\mathcal{G}$, where the membership to the set S of nodes to be connected is calculated by Mahalanobis distance.

According to Eq. (8), vector v is in the receptive fields of node i, if and only if for any node j that is connected to node i in $\mathcal{G}$, satisfies:

$$D_i(v) \leq D_j(v) \tag{9}$$

namely,

$$(v - c_i)^T M_i^{-1}(v - c_i) \leq (v - c_j)^T M_j^{-1}(v - c_j). \tag{10}$$

Then the inequality constraint (10) can be transformed into:

$$v^T(M_i^{-1} - M_j^{-1})v + 2(c_j^T M_j^{-1} - c_i^T M_i^{-1})v + (c_i^T M_i^{-1} c_i - c_j^T M_j^{-1} c_j) \leq 0. \tag{11}$$

Thus, we can use the hypersurface

$$v^T(M_i^{-1} - M_j^{-1})v + 2(c_j^T M_j^{-1} - c_i^T M_i^{-1})v + (c_i^T M_i^{-1} c_i - c_j^T M_j^{-1} c_j) = 0 \tag{12}$$

to define the divisional surface between the Voronoi cells of node i and node j in $\mathcal{G}$.

That shows the divisional surface exists between any two of the nodes that are connected in $\mathcal{G}$. Therefore, the Voronoi cells that are constrained by these divisional surfaces are coherent in $\mathcal{G}$, and the nodes that are connected in $\mathcal{G}$ are also adjacent on the resulting topology $\mathcal{M}$. Then we conclude that the definition of neighborhood relationship and the topology in LD-SOINN is valid.

2.5. Node merging

While the learning continues, there may be some nodes close to each other and having similar principal components. We need to merge such nodes to obtain a concise local representation. Two nodes will merge if the following two conditions are satisfied:

(i) the two nodes i and j are connected by an edge; and
(ii) the volume of the combined node m is less than the sum volume of the two nodes i and j, i.e.,

$$Volume(m) < Volume(i) + Volume(j). \tag{13}$$

With the hyper-ellipsoid defined in formula (1), the volume of node n can be calculated as (proved in Appendix C):

$$Volume(n) = 2^{\left\lceil \frac{d+1}{2} \right\rceil} \pi^{\left\lceil \frac{d}{2} \right\rceil} \left(\prod_{i=0}^{\lfloor d/2 \rfloor - 1} \frac{1}{d - 2i} \right) \sqrt{|M_n|} H_n^d \tag{14}$$

where M_n and H_n are the covariance matrix and the vigilance parameter of node n respectively. $|M_n|$ represents the determinant of M_n, $[\cdot]$ represents the rounding operation and d is the dimension of the sample space.

If the above conditions are satisfied for nodes i and j, we merge i and j (the formula is proved in Appendix B):

$$\begin{aligned} n_m &= n_i + n_j \\ c_m &= (n_i c_i + n_j c_j)/n_m \\ M_m &= \frac{n_i}{n_m}(M_i + (c_m - c_i)(c_m - c_i)^T) \\ &\quad + \frac{n_j}{n_m}(M_j + (c_m - c_j)(c_m - c_j)^T) \\ H_m &= \varepsilon_{n_m} \chi^2_{d,q}. \end{aligned} \tag{15}$$

In practice, we only merge the winner node and its neighbors when a pattern is fed into LD-SOINN.

Fig. 3 gives an example of the node merging step. In the figures, the node with red boundary-line represents the winner node when an input pattern is received. The node with blue boundary-line represents the neighbor node of the winner node. Under merging conditions (2) and (ii), LD-SOINN merges nodes that are close to each other and have similar principal components. LD-SOINN is able to obtain a concise local representation with the node merging step.

2.6. Denoising

The data from the learning environment may contain noise. Some nodes may be created by these noisy data. Since LD-SOINN records the local distribution of the learning data in each node, we can use this information to judge whether a node is a noise node or not.

After every λ patterns learned, LD-SOINN first calculates the mean value of the number of input patterns belonging to each node as:

$$Mean = \sum_{i=1}^{|N|} n_i / |N| \tag{16}$$

where $|N|$ represents the total number of nodes, n_i is the number of input patterns belonging to i. We assume that the probability density of the noise is lower than the useful data. Based on this assumption, if n_i is smaller than a threshold $k * Mean$, we mark node i as a noise node and remove it, where $0 \leq k \leq 1$. Large k means there is much noise in the learning environment and small k means there is a small amount of noise.

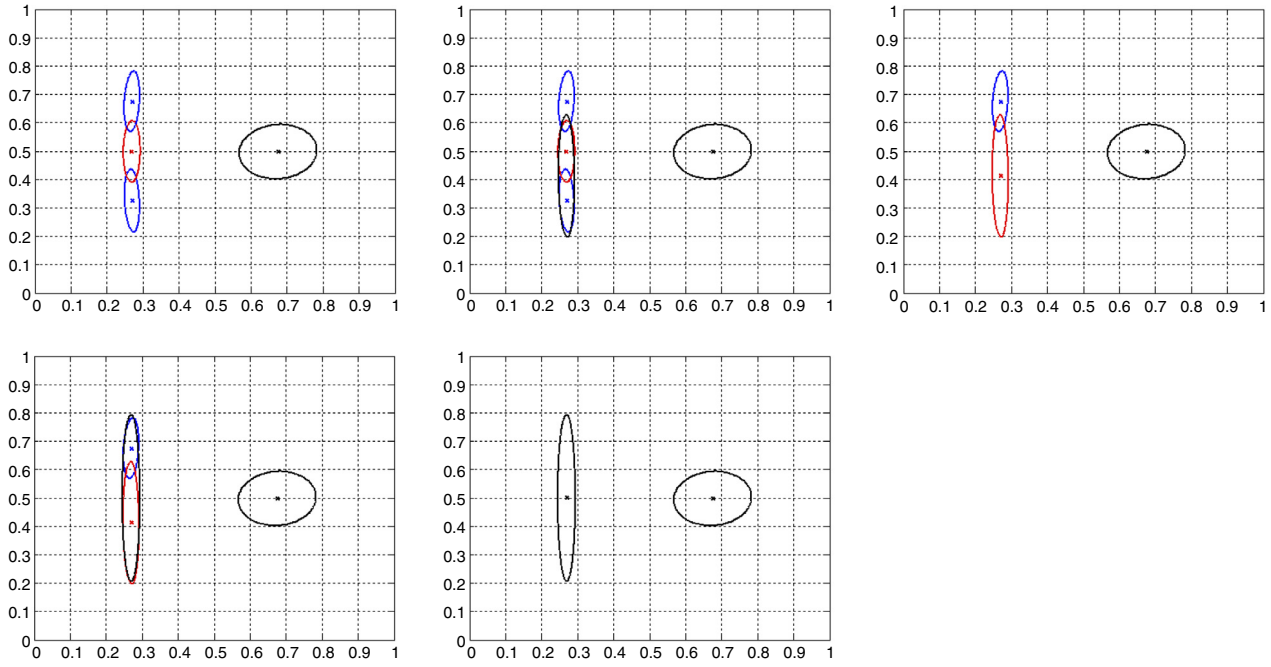


Fig. 3. An example of the node merging step. In the figures, the node with red boundary-line represents the winner node when an input pattern comes. The node with blue boundary-line represents the neighbor nodes of the winner node. By merging nodes which are close to each other and have similar principal components, LD-SOINN is able to obtain a concise local representation, i.e., a *relaxation* data representation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.7. Cluster

In Section 2.4, all the nodes that are activated by the same input pattern are connected in the network according to competitive Hebbian rule (Martinetz & Schulten, 1994), which forms a perfect topology structure under the assumption that the number of nodes is sufficient for obtaining a dense distribution. Then we take each connected subgraph in the topology as a cluster. Therefore, we can find different connected node domains as different clusters when the learning process is finished. Algorithm 1 shows the details of the cluster method.

Algorithm 1 Cluster nodes

1: Initialize all nodes as unclassified.
2: Choose one unclassified node i from node set N. Mark i as classified and label it as class C_i.
3: Search node set N to find all unclassified nodes that are connected to node i with a "path". Mark these nodes as classified and label them as the same class as node i.
4: Go to Step 2 to continue the classification process until all nodes are classified.

Note: if two nodes can be linked with a series of connections, we say that a "path" exists between the two nodes, i.e., if there are a series of nodes $x_k \in N, k = 1, 2, \ldots, n$, making $(i, x_1), (x_1, x_2), \ldots, (x_{n-1}, x_n), (x_n, j) \in C$, we say that a "path" exists between node i and node j.

2.8. Complete algorithm of LD-SOINN

As a summary, we give the complete algorithm of LD-SOINN in Algorithm 2.

Algorithm 2 Complete algorithm of LD-SOINN

1: Initialize the network with $N = \emptyset, C = \emptyset$.
2: Input new sample $x \in \mathbb{R}^d$.
3: Determine set S using formula (3), where the elements of S are the nodes activated by sample x.
4: If $S = \emptyset$, initialize a new node as formula (4) then goto Step 2.
5: If $S \neq \emptyset$, choose the winner node i^* from set S by formula (5) and update i^* using formula (6).
6: Establish connections between the activated nodes in set S.
7: If the winner node i^* and its neighbors satisfy the merging conditions (i) and (ii), implement merging procedure as formula (15).
8: If the number of input patterns presented so far is an integer multiple of parameter λ, denoising as in Section 2.6.
9: If all samples have been inputted to LD-SOINN, go to step 10. Else, go to Step 2 to learn a new sample.
10: Cluster using Algorithm 1 and output the learning result.

3. Analysis

3.1. The expansivity of the nodes

The parameter H of each node is used to judge whether the corresponding node is activated when an input pattern $x \in \mathbb{R}^d$ is received. As each node in LD-SOINN only records the local regional distribution, we can easily assume that the patterns belonging to one node are generated by a Gaussian distribution:

$$X \sim \mathcal{N}(c_X, \Sigma_X) \tag{17}$$

where c_X is the center of the node and Σ_X is the covariance matrix. The hyper-ellipsoid boundary equation of every node is:

$$(x - c_X)^T \Sigma_X^{-1} (x - c_X) = H^2. \tag{18}$$

Let $K = H^2$, then K is a χ^2 distribution with d degree of freedom. The probability density function of K is:

$$f(k) = \frac{1}{2^{d/2}\Gamma(d/2)} k^{(d-2)/2} e^{-k/2}. \tag{19}$$

Giving a confidence q, the patterns from random variables X that lie in the hyper-ellipsoid drawn by K can be described as:

$$P\{(x - c_X)^T \Sigma_X^{-1} (x - c_X) < K\} = q. \tag{20}$$

Then K can be solved by $K = \chi^2_{d,q}$. For $K = H^2$, we can get the value of H. In practice, for node i, we set $H_i = \varepsilon_{n_i}\sqrt{\chi^2_{n,q}}$, where $\varepsilon_{n_i} \geq 1$ and ε_{n_i} decreases when n_i increases. This strategy lets the hyper-ellipsoid have a tendency of expansivity at the preliminary stage when $\varepsilon_{n_i} > 1$. With more patterns included in node i, H_i^2 approaches to $\chi^2_{d,q}$; then the hyper-ellipsoid arrives at a stable state. In this paper, we set $\varepsilon_{n_i} = (1 + 2 * 1.05^{1-n_i})$.

In LD-SOINN, σ decides the initial hyper-ellipsoid size of new nodes. It can be understood as the initial size of the window we observe in the learning data. Very big σ may lead to a new node cover for several different clusters; therefore, LD-SOINN prefers small σ. Though small σ may make LD-SOINN initially generate many nodes with small hyper-ellipsoid size, LD-SOINN can merge these nodes following the learning process.

Owing to the tendency of expansiveness and the ability of node merging, the size of each node grows during learning. After a period of learning, some nodes generated by LD-SOINN cover the distribution area of the learning data. Then LD-SOINN does not create new nodes for this area. This avoids the situation where the number of nodes grows unlimitedly.

3.2. The relaxation data representation

In the merging condition (ii) of two neighboring nodes, the volume of each node can be calculated as formula (14). However, $|M|$ in formula (14) may be very close to 0 in some high dimensional task. Thus it is not suitable to use (14) to calculate the volume directly.

To avoid calculating the volume directly with formula (14), for the covariance matrix M_i, M_j and M_m of node i, j and merging node m in node merging condition (ii), we do Singular Value Decomposition (SVD) as:

$$M = E^T \begin{pmatrix} \lambda_1 & \cdots & \cdots & 0 \\ \vdots & \lambda_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \lambda_d \end{pmatrix} E \tag{21}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$, and we get $\lambda_1^i, \lambda_2^i, \ldots, \lambda_d^i$ for node i, $\lambda_1^j, \lambda_2^j, \ldots, \lambda_d^j$ for node j and $\lambda_1^m, \lambda_2^m, \ldots, \lambda_d^m$ for node m.

Then we find a truncated position t of all singular value with a predefined scaling factor ρ: $p = \text{argmin}_{1\leq p\leq d} \sum_{i=1}^{t} \lambda_i \geq \rho \sum_{i=1}^{d} \lambda_i$. For node i, j and merging node m, we get t_i, t_j and t_m respectively. In this paper, we set ρ=0.95.

Finally, we get a common truncated position $t = \max(p_i, p_j, p_{merge})$ and calculate $|M_k|$ by using $\lambda_1^k \times \lambda_2^k \times \cdots \times \lambda_t^k$, $k = i, j, m$.

Substituting $|M_k|$ (where $k = i, j, m$) into formula (14) and replacing d with t, we get $Volume(i)$, $Volume(j)$ and $Volume(m)$. Substituting these three volumes into formula (13) and after a series of simplifications, we get an equivalence merging condition to (13) as:

$$\sqrt{\frac{\lambda_1^i H_i}{\lambda_1^m H_m} \cdot \frac{\lambda_2^i H_i}{\lambda_2^m H_m} \cdots \frac{\lambda_t^i H_i}{\lambda_t^m H_m}} + \sqrt{\frac{\lambda_1^j H_j}{\lambda_1^m H_m} \cdot \frac{\lambda_2^j H_j}{\lambda_2^m H_m} \cdots \frac{\lambda_t^j H_j}{\lambda_t^m H_m}} \geq 1. \tag{22}$$

In practice, we use formula (22) to judge the merging condition (ii). If formula (22) is not satisfied, merging between two nodes is unsuccessful, and node i and j remain unchanged.

At the node merging step, we have two candidate data representations: (1) representation before merging and (2) representation after merging. Assume the domain of distribution $Q_i(x)$ of node i is $x \in R_i$, the domain of distribution $Q_j(x)$ of node j is $x \in R_j$. Then the data representation f_1 before node merging in domain $R_i \cup R_j$ is:

$$f_1 : \begin{cases} \sqrt{(x - c_i)^T M_i^{-1} (x - c_i)} < H_i & x \in R_i \\ \sqrt{(x - c_j)^T M_j^{-1} (x - c_j)} < H_j & x \in R_j. \end{cases} \tag{23}$$

According to node merging condition (i), we know that $R_i \cap R_j \neq \emptyset$.

The data representation f_2 after node merging in domain $R_i \cup R_j$ is:

$$f_2 : \sqrt{(x - c_m)^T M_m^{-1} (x - c_m)} < H_m \quad x \in R_i \cup R_j. \tag{24}$$

Assuming learning data in domain $R_i \cup R_j$ are generated by a Gaussian distribution, setting $H_m = \chi^2_{d,q}$ will guarantee the probability that the learning data in domain $R_i \cup R_j$ falls in f_2 equals q (usually $q \geq 90\%$). Meanwhile, according to node merging condition (ii), the volume of node m is less than the total volume of nodes i and j; thus, we get a much concise data representation on domain $R_i \cup R_j$.

Comparing the two representations, we can see that f_1 uses more parameters than f_2, and the two regions in f_1 overlap each other. Thus, the representation f_1 is *tight*, and we call f_1 as a *tight* data representation. On the other hand, the representation f_2 expresses the data distribution more concisely than f_1; it relaxes the requirement of the parameters. Hence, we call representation f_2 as a *relaxation* data representation.

3.3. The relationship with local-PCA

PCA is a popular method of projecting learning data to its subspace and gaining an approximate representation for dimension reduction. It is an important result of spectrum analysis and is widely applied in data compression. But it is inefficient in dealing with nonlinear situations (Kambhatla & Leen, 1997). The global nonlinear approach and local linear description are two main extensions to solve that problem. We focus on the latter approach in which the data distribution is represented by a collection of local PCA units (Weingessel & Hornik, 2000).

The principal components are obtained by computing the eigenvectors of the covariance matrix of each node in LD-SOINN. The dimension reduction is implemented with the projection on the subspace spanned by these components, and the nodes in LD-SOINN can be regarded as local PCA units.

On the other hand, the training of LD-SOINN is a process of finding representative samples to stand for the original data. It is a reduction based on the distribution of the learning data, which greatly reduce the storage space of the samples. With the reduction in dimensions and samples, LD-SOINN holds great promise in data compression.

3.4. The relationship with IPCA

Incremental principal component analysis (Hall, Marshall, & Martin, 1998) (IPCA) is proposed as a feature extraction and classifier learning method, in which observations are added to an eigenspace model in an online and one pass manner. In IPCA, training samples must be learned one by one, which causes inefficiency in computations because the eigenvalue decomposition in IPCA must be applied to each training sample in a chunk of data. Thus, chunk IPCA (Ozawa, Pang, & Kasabov, 2008) is proposed to solve this problem. Chunk IPCA updates the eigenspace by performing single eigenvalue decomposition when learning a chunk of samples. Another extension of IPCA, namely IKPCA (Chin & Suter, 2007), introduces kernel methods to IPCA and obtains accurate nonlinear principal components to describe a complex data distribution incrementally.

For the purpose of feature extraction, IPCA learns an eigenspace model by training all the input samples in an online manner. In difference with IPCA, LD-SOINN learns a number of local eigenspaces, each of which are trained by using partial samples that lie in a local dense region and joint to each other. At this point, LD-SOINN aims at representing complex data distributions and unsupervised learning rather than feature extraction and classifier learning.

3.5. The relationship with GMM

Gaussian distribution has a density function:

$$f(x) = (2\pi)^{-d/2}|M|^{-1/2}\exp^{-\frac{1}{2}(x-c)^T M^{-1}(x-c)}. \quad (25)$$

It is one of the most important and useful distributions. In recent decades, it has been widely applied to data mining to simulate the input distribution. Gaussian Mixture Model (GMM) is one implementation of these applications. Mixture models are able to approximate arbitrarily complex probability density functions. This fact makes them an excellent choice for representing distribution of complex input data (Figueiredo & Jain, 2002).

Taking note of this, the formula of Mahalanobis distance and Gaussian density function both contain the term $(x-c)^T M^{-1}(x-c)$, where M is the covariance matrix and c is the mean vector, which respectively refer to the unbiased estimations of variance and expectation in mathematic statistics. In LD-SOINN, each node stores the mean vector and the covariance matrix to represent the distribution of a local region of the learning data. The learning process of LD-SOINN is a process of updating the mean vector and the covariance matrix of each node; it is just the thing that GMM does. Because LD-SOINN is an incremental learning model that can add new nodes for new knowledge in on-line manner, LD-SOINN can be interpreted as an incremental GMM.

4. Experiments

In this section, we test LD-SOINN on some artificial and some real-world data sets. As LD-SOINN aims to combine the advantages of incremental learning and matrix learning, for comparing, we find some classical and state-of-the-art methods for the same. The matrix learning methods include localPCASOM (Huang et al., 2008), BatchMatrixNG (Arnonkijpanich et al., 2011) and oKDE (oKDE is also an incremental learning method) (Kristan et al., 2011). The incremental learning methods include TopoART (Tscherepanow et al., 2011) and Adjusted-SOINN (Shen & Hasegawa, 2008).

We introduce the experiments on the artificial data sets first, and then we introduce the experiments on the real-world data sets.

4.1. Artificial data

4.1.1. Observe the periodical learning results

In order to observe the periodical result during learning, we use the artificial data that is distributed in two belt areas; this data set is also used in PCASOM (López-Rubio et al., 2004). The two belt areas orient in the y-axis, but have different centers in the x-axis. Each belt area represents a cluster and generates samples uniformly. We set parameters of LD-SOINN as $\chi^2_{d,q} = \chi^2_{(2,0.90)}$, $\sigma = 1e-5$, $k = 0.01$, $\lambda = 1000$. Fig. 4 gives the periodical learning results of the whole learning behavior. Fig. 4(a) shows that at the early stage, many ellipsoids are generated to cover the learning data set. While the learning process continues, some ellipsoids are merged together as at the hundred patterns stage, which is shown in Fig. 4(b). Fig. 4(c) shows that after two hundred patterns, all ellipsoids are merged into two ellipsoids. Finally, LD-SOINN gets two ellipsoids, which fit the original data set very well. Contrary to PCASOM, LD-SOINN does not need to predetermine the number of nodes; it automatically generates two nodes in this task.

Fig. 5 gives the learning results for different σ. LD-SOINN works very well with a small value of σ. With a big value of σ, the learning results of early stage include regions that do not belong to the learning set, and with the learning going on, LD-SOINN shrinks to fit the learning set well. Fig. 5 indicates that though σ takes two values separated by a big gap, the learning results are both good. Thus, LD-SOINN is not so sensitive to σ.

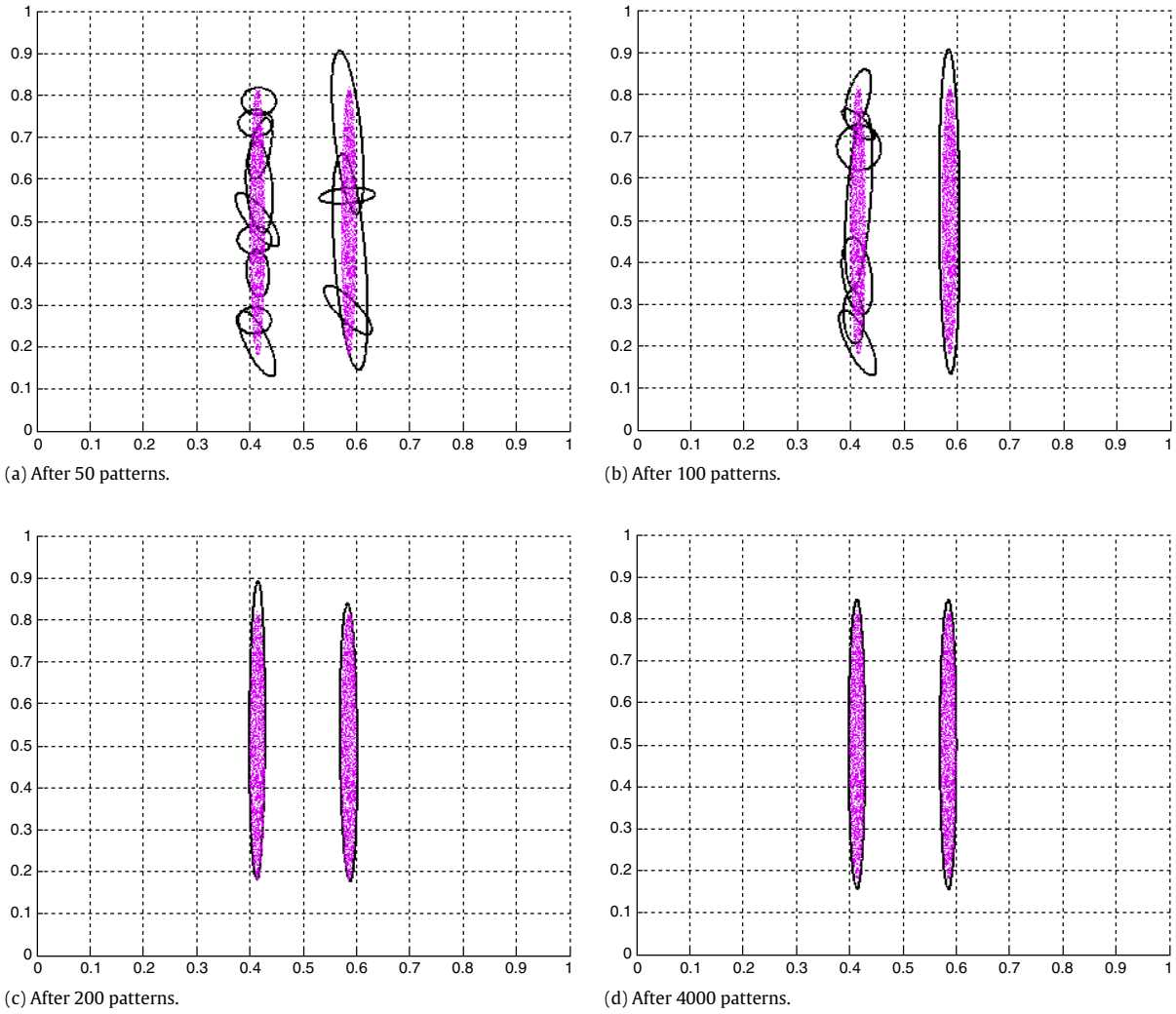
4.1.2. Compare with matrix learning methods

The spirals data set is shown in Fig. 6. The learning data set consists of 20 000 patterns. We compare LD-SOINN with some matrix learning methods including localPCASOM (Huang et al., 2008), BatchMatrixNG (Arnonkijpanich et al., 2011) and oKDE (Kristan et al., 2011). We randomly select thirty patterns from the original data set as the initial nodes for BatchMatrixNG and localPCASOM. Ten patterns are randomly selected from the original data set as the initial nodes for oKDE. We set the parameters of localPCASOM as $N = 30$, $\varepsilon = 0.01$, $H = 1.5$; of BatchMatrixNG as $N = 30$, $epoch = 1000$; of oKDE as $N = 10$, $D_{th} = 0.01$, $f = 1$; and of LD-SOINN as $\chi^2_{d,q} = \chi^2_{(2,0.90)}$, $\sigma = 1e-5$, $k = 0.01$, $\lambda = 1000$. Learning patterns are randomly selected from the whole data set. The learning results are drawn by black boundary-line ellipses. Fig. 6(a) shows the learning result of localPCASOM. Some ellipses overlap in the center of the spirals and also do not fit the original data set well. Fig. 6(b) shows the learning result of BatchMatrixNG. Here some ellipses are across the blank area of the spirals. In Fig. 6(c), oKDE automatically generates 38 nodes to fit the learning set; however, the direction of the principal component of some ellipses is deviated. Comparing Fig. 6(d) with Fig. 6(a)–(c), the proposed LD-SOINN fits the original data much better than other methods. Moreover, LD-SOINN does not need to predetermine the number of the nodes; it automatically generates 29 nodes in this task.

4.1.3. Working in complex environment

In this section, we conduct our experiment on the data set shown in Fig. 7. The data set is separated into five parts: A, B, C, D, and E. Data sets A and B satisfy 2-D Gaussian distribution. C and D are concentric-rings distribution. E is sinusoidal distribution. We also add 10% noise to the data set. As shown in Fig. 7, noise is distributed over the entire data set.

We train six different types of networks: localPCASOM, BatchMatrixNG, Adjusted-SOINN, TopoART, oKDE and LD-SOINN. We set the parameters of localPACSOM as $N = 20$, $\varepsilon = 0.01$, $H = 1.5$; of BatchMatrixNG as $N = 20$, $epoch = 1000$; of Adjusted-SOINN as $\lambda = 200$, $age_{\max} = 25$, $c = 0.5$; of TopoART as $\beta_{sbm} = 0.32$, $\rho = 0.96$, $\varphi = 5$, $\tau = 100$; of oKDE as $D_{th} = 0.01$, $N = 10$,



(a) After 50 patterns.

(b) After 100 patterns.

(c) After 200 patterns.

(d) After 4000 patterns.

Fig. 4. Periodical learning results of LD-SOINN: during learning, LD-SOINN automatically creates new nodes, merges the "similar" nodes and finally gets two clusters. Different from PCASOM, it does not need to predetermine the node number. The parameters of LD-SOINN are $\chi^2_{(d,q)} = \chi^2_{(2,0.90)}, \sigma = 1e-5, k = 0.5$. The pink area represents the distribution of the learning data. Ellipses with black boundary-lines are the learning result. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$f = 1$ for stationary environment and $f = 0.99$ for non-stationary environment; and of LD-SOINN as $\chi^2_{(d,q)} = \chi^2_{(2,0.90)}, \sigma = 1e-5$, $k = 0.5, \lambda = 1000$.

We conduct the experiment in two environments for these methods. In the stationary environment, patterns are randomly selected from the whole learning set. In the non-stationary environment, learning period is divided into five different learning environments. In each environment one data set is activated such as in environment I, from step 1 to 4000, data set A is activated, in environment II, from step 4001 to 8000, data set B is activated, etc. The experiment in the non-stationary environments is aimed to test whether the methods can deal with the Stability–Plasticity Dilemma or not, i.e., whether or not the learning method will be able to learn new knowledge effectively, while preserving the priori learned knowledge.

Fig. 8, Fig. 9 and Table 1 show the learning results. Different clusters are colored differently in Fig. 9(b)–(d).

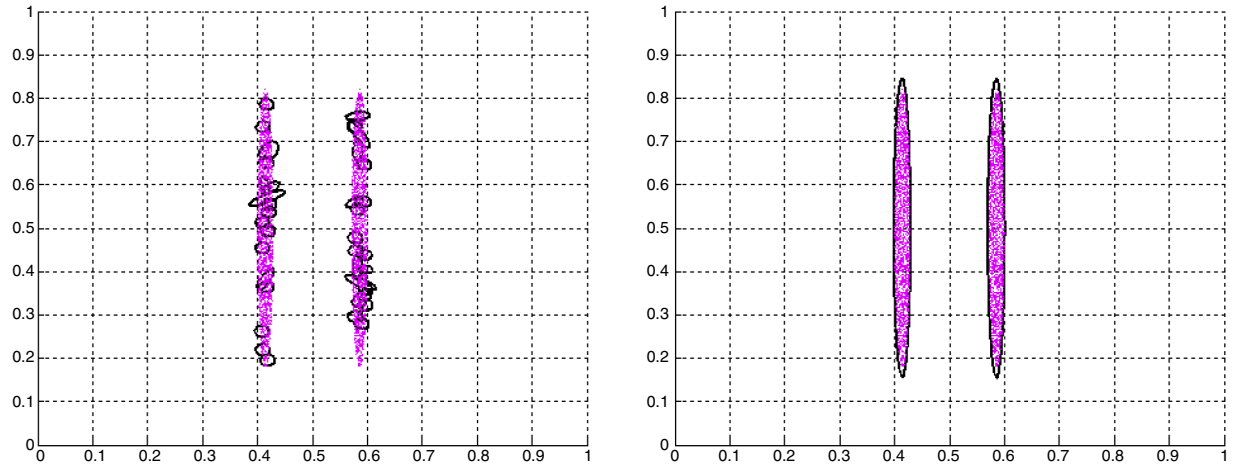
Fig. 8 shows the learning results of localPCASOM and BatchMatrixNG; the nodes learned by localPCASOM and BatchMatrixNG do not fit the learning data set well in both environments. In the non-stationary environment, after the learning period of environment 1, localPCASOM cannot learn new class distribution data at all, i.e., it has bad plasticity. On the other hand, BatchMatrixNG can learn new class distribution data in the non-stationary environment; however, the previous class (class A and B) distribution cannot be retained, i.e., it has bad stability. As these two methods face the Stability–Plasticity Dilemma, they are not suitable for this task.

Table 1
Learned node number in stationary and non-stationary environment for data set Fig. 7.

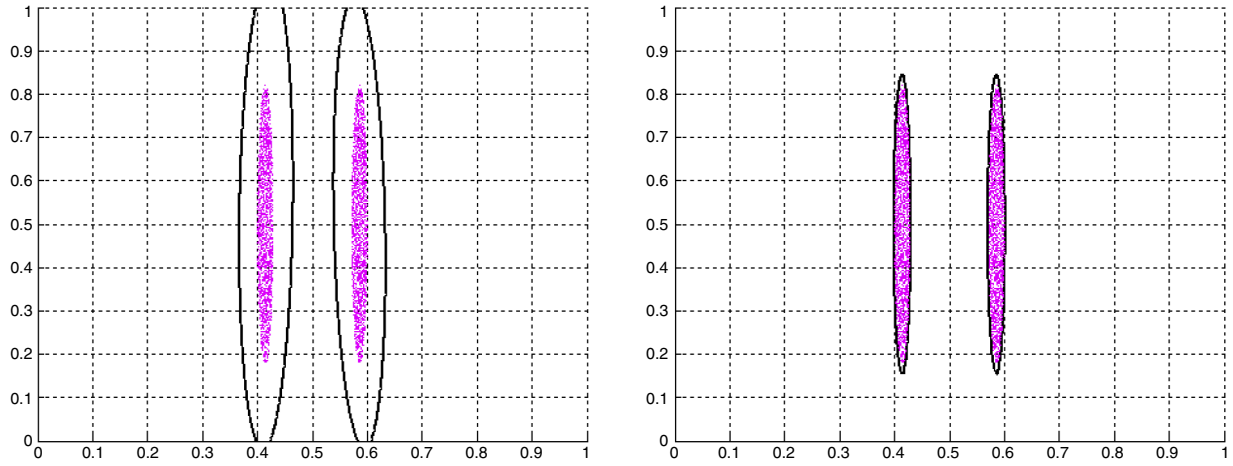
Environment	oKDE	Adjusted-SOINN	TopoART	LD-SOINN
Stationary	45	83	205	18
Non-stationary	69	194	246	20

Fig. 9 shows the learning results of the incremental learning methods. Comparing LD-SOINN with the other three incremental learning methods, namely, oKDE, TopoART and Adjusted-SOINN, LD-SOINN obtains the best fitting results using the least nodes. LD-SOINN can merge some local small ellipsoids into a big one, which leads to a more concise data representation that oKDE cannot match. On the other hand, LD-SOINN gets the local distribution information about the learning data that Adjusted-SOINN and TopoART cannot get.

Table 1 gives the number of nodes in the learning result. It indicates that LD-SOINN is the most stable method. Comparing the



(a) $\sigma = 4e - 6$.



(b) $\sigma = 4e - 3$.

Fig. 5. Learning behaviors with different initial sizes of ellipsoid after 50 and 4000 patterns. The pink area represents the distribution of the learning data. Ellipses with black boundary-line are the learning result. Though σ takes two values with a big gap, the learning results are both well; we can conclude that LD-SOINN is not so sensitive to σ. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

two environments, oKDE gets 45 nodes in the stationary environment and 69 nodes in the non-stationary environment (45/69). Adjusted-SOINN gets 83/194 nodes for the two different environments. TopoART gets 205/246 nodes for both environments. The three methods have a big gap in the number of nodes between the different environments in the learning results. LD-SOINN gets a much better stability: 18 nodes in the stationary environment and 20 nodes in the non-stationary environment. It means that LD-SOINN is not sensitive to external learning environment.

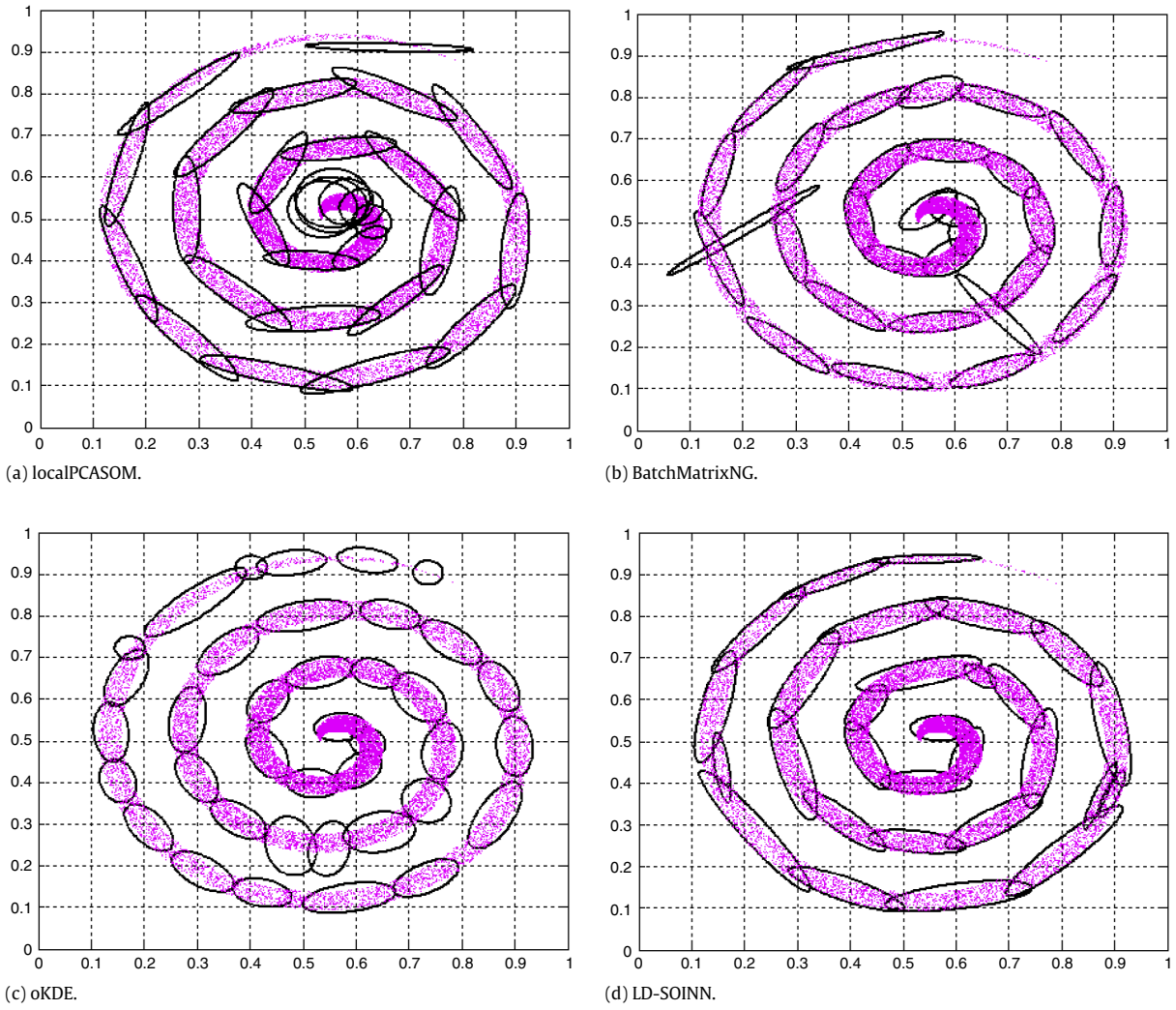
Fig. 10 gives a periodical learning result in a non-stationary environment. The networks were successively trained with samples from the sub-distributions A, B, C, D, and E, which are depicted in the upper row. The lower row shows the learning results of LD-SOINN after finishing the respective learning period. When the learning environment changes, while the prior learned nodes remain stable, new nodes for the new distribution are created. Fig. 10 indicates that LD-SOINN solves the Stability–Plasticity Dilemma very well. Meanwhile, through this experiment, LD-SOINN preserves the order property of the learning data.

In order to observe the performance of *relaxation* data representation and *tight* data representation, we do an experiment on the data set in Fig. 7 without noise on oKDE and LD-SOINN. The learning result is shown in Fig. 11. We can see that *relaxation* data representation (b) can generate a more concise representation than *tight* data representation (a) on the two Gaussian distributions. For the distribution that cannot be further simplified, *relaxation* data representation is able to get results comparable with *tight* data representation, such as the sinusoidal distribution. This experiment demonstrates that *relaxation* data representation has greater ability in data representation than *tight* data representation. Fig. 12 gives the trend of the number of the node learned by LD-SOINN in stationary and non-stationary environment. The figure shows that, in the stationary environment, LD-SOINN creates nodes rapidly in the early stage and converges soon later. In the non-stationary environment, LD-SOINN creates nodes when new distributions come. Thus, the line shows a step style. Overall, the learning gradually converges to a stable state in both environments.

4.2. Real-world data

First, we do an experiment on the ATT_FACE database.[1] There are 40 persons in the database and each person has 10 images that include different lighting conditions, facial expressions and facial detail images. Feature vectors of such images are taken as

[1] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.



(a) localPCASOM. (b) BatchMatrixNG.

(c) oKDE. (d) LD-SOINN.

Fig. 6. A result of some matrix learning methods on the spiral data set. The pink area represents the distribution of the learning data. Ellipses with black boundary-line are the learning results. Comparing Fig. 6(d) with Fig. 6(a)–(c), the proposed LD-SOINN fits the original data much better than other methods; moreover, LD-SOINN does not need to predefine the number of nodes, it automatically generates 29 nodes in this task. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

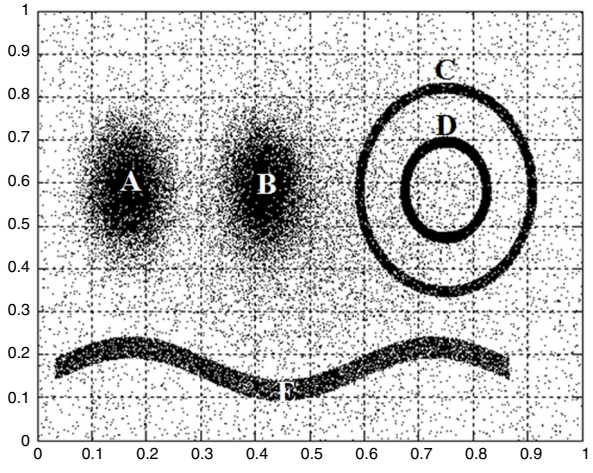


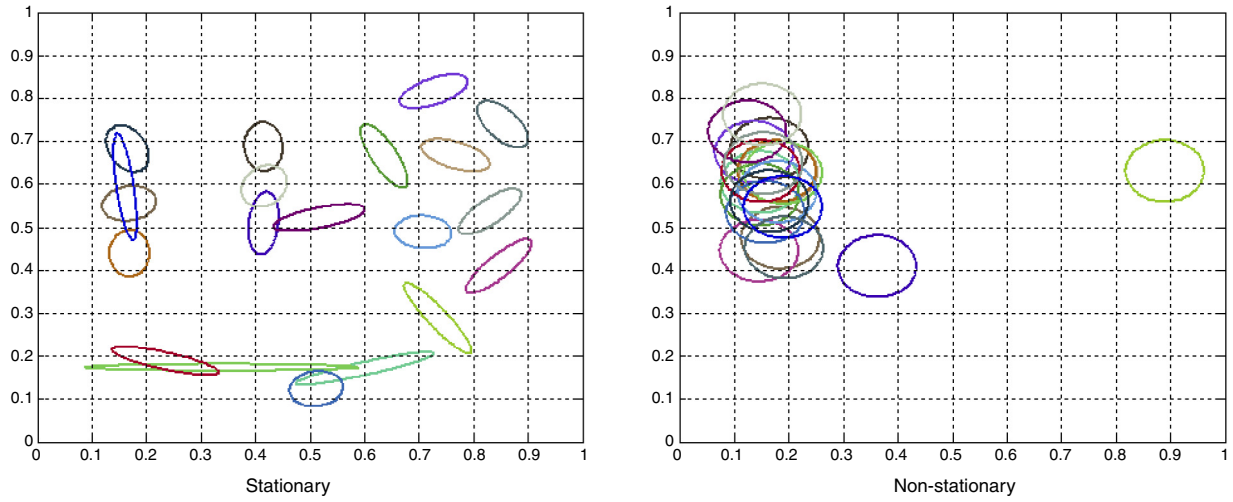
Fig. 7. Artificial data set used for the experiment in the complex environment. 10% noise is distributed over the entire data set.

follows: first, the original image with size 92×112 is re-sampled to 23×28 image using the nearest neighbor interpolation method. Then Gaussian smoothing is used to smooth the 23×28 image with $HSIZE = [4; 4]$, $\sigma = 2$ to obtain the input feature vectors.

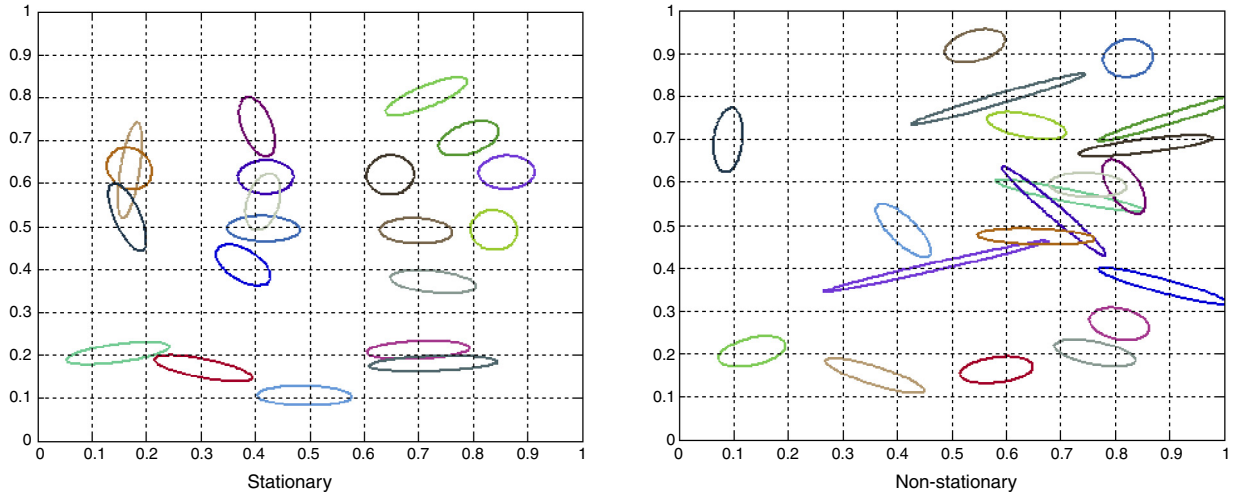
We conduct the experiment in stationary and non-stationary environments for oKDE, Adjusted-SOINN, TopoART and LD-SOINN. In stationary environment, patterns are randomly selected from the whole learning set. In non-stationary environment, from step 1 to 200, patterns are chosen randomly from person 1. From step 201 to 400, the environment changes and patterns from person 2 are chosen, and so on. We repeat the learning process 100 times in different random order to compare the stability of oKDE, Adjusted-SOINN, TopoART and LD-SOINN. BatchMatrixNG and localPCASOM are not incremental method; we conduct these two methods only in the stationary environment. Such methods need a predefined node number to guarantee a good learning result; we set it as 200 for these two methods and give a good initial condition: the initial nodes contain images of all 40 persons.

We set the parameters of localPCASOM as $N = 200$, $\varepsilon = 0.01$; of BatchMatrixNG as $N = 200$, $epoch = 200$; of oKDE as $D_{th} = 0.01$, $N = 10$, $f = 1$ for stationary environment and $f = 0.99$ for non-stationary environment; of Adjusted-SOINN as $\lambda = 100$, $age_{\max} = 50$, $c = 0.25$; TopoART as $\beta_{sbm} = 0.6$, $\rho = 0.96$, $\varphi = 3$, $\tau = 100$; and of LD-SOINN as $\chi^2_{(d,q)} = \chi^2_{(644,0.90)}$, $\sigma = 1e-3$, $k = 0.01$, $\lambda = 1000$.

We select four factors to evaluate the learning results: node number, missing person number, node purity and recognition ratio. These four factors imply that the ideal cluster result should



(a) localPCASOM.



(b) BatchMatrixNG.

Fig. 8. Learning results of localPCASOM and BatchMatrixNG in the stationary and non-stationary environments. The learning results of localPCASOM and BatchMatrixNG do not fit the learning data set well. In the non-stationary environment, localPCASOM cannot learn new class distribution data at all, i.e., it has bad plasticity. On the other hand, BatchMatrixNG can learn new class distribution data in the non-stationary environment; however, the previous class distribution cannot be retained, i.e., it has bad stability. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

have the following: the learning result should be as concise as possible; the lost classes should be as few as possible; the data merged in a node should be from the same class; the learning result should have high generalization ability. However, in practice, we often need to strike a balance between these factors. As localPCASOM and BatchMatrixNG need predefined node numbers, we do not introduce the evaluation factor node number to them. Table 2 gives the mean node number of 100 times learning results of the four incremental learning methods: oKDE, Adjusted-SOINN, TopoART and LD-SOINN in both stationary and non-stationary environments. In Table 2, oKDE gets only 3 nodes in the learning result in both environments. Most persons' images are averaged into a single node; thus, oKDE is not suitable for this task. It loses lots of persons (38 persons per 40 persons as shown in Table 3). We do not calculate the remaining two factors (node purity and recognition ratio) for oKDE because the learning result is bad. According to Tables 2 and 3, though LD-SOINN gets larger node number in stationary environment than TopoART, it does not lose any person; however, TopoART loses 26.05 persons per 40 persons. In non-stationary environment, LD-SOINN gets the smallest node number (excluding oKDE) and does not lose any person in the learning results. On the other hand, the Adjusted-SOINN "forget" 5.01 persons.

Table 2
Mean node number of 100 times learning results in stationary and non-stationary environment for ATT_FACE.

Environment	oKDE	Adjusted-SOINN	TopoART	LD-SOINN
Stationary	3	276.24	16.65	247.29
Non-stationary	3	317.33	260.82	247.31

Table 3
Mean missing person number of 100 times learning results in stationary and non-stationary environment for ATT_FACE.

Environment	oKDE	Adjusted-SOINN	TopoART	LD-SOINN
Stationary	38	0	26.05	0
Non-stationary	38	5.01	0	0

Comparing the node number of the two environments in Table 2, we find that there is a big gap between stationary and non-stationary environments in Adjusted-SOINN and TopoART: 276.24 in stationary environment and 317.33 in non-stationary environment of Adjusted-SOINN; 16.65 in stationary environment and

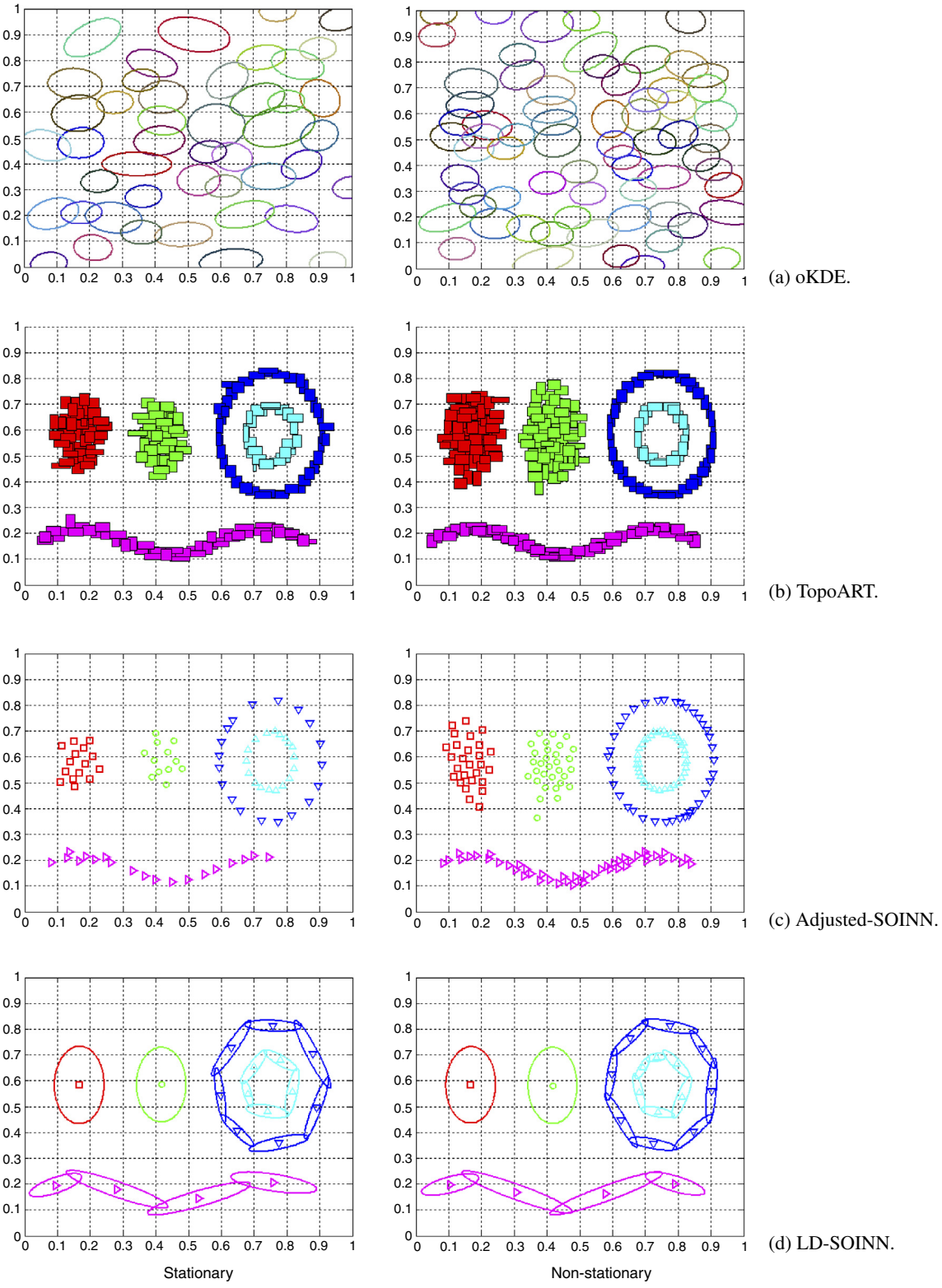
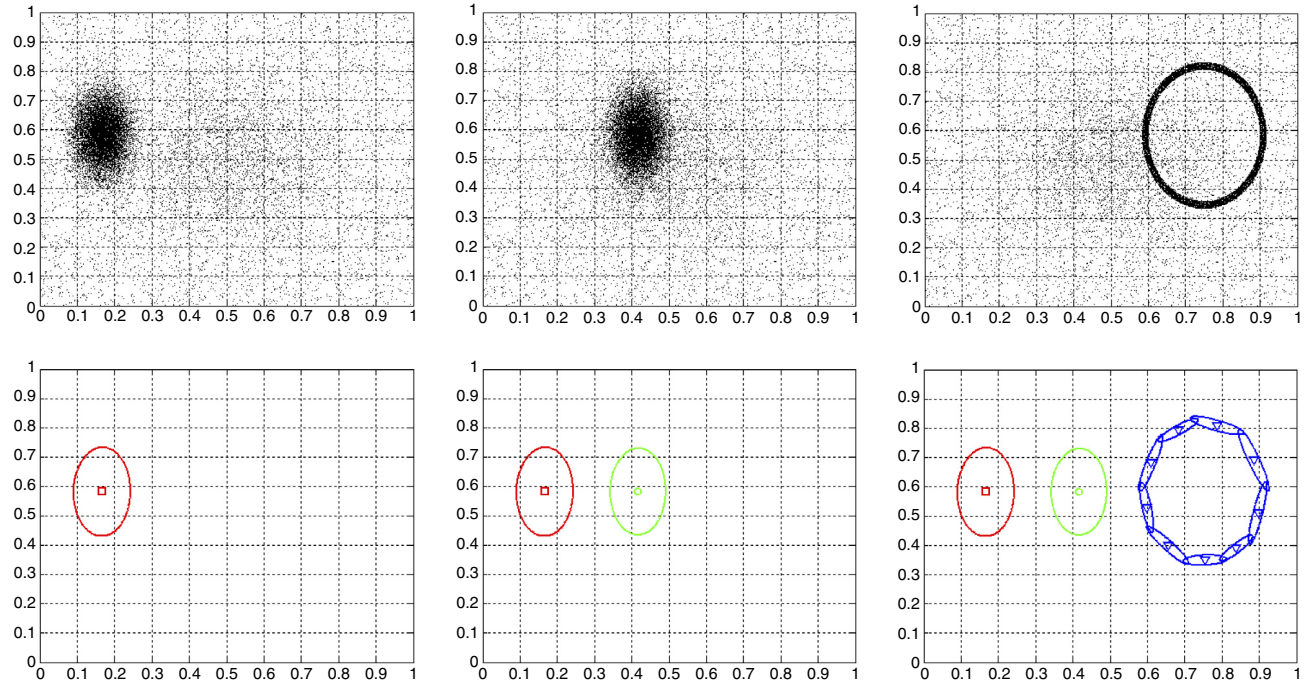


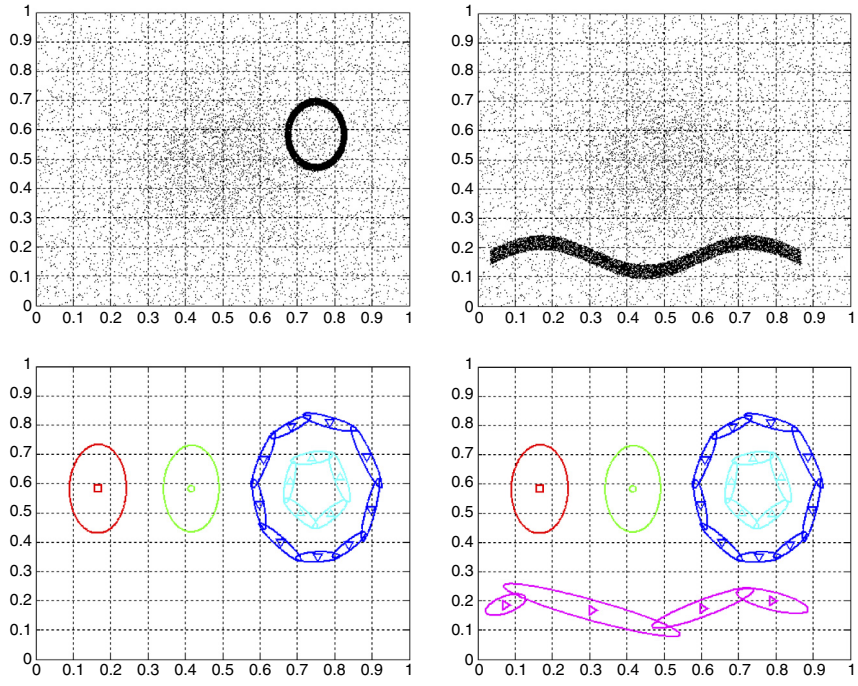
Fig. 9. Comparing results for data set Fig. 4 in the stationary and non-stationary environment. We can see that (a) oKDE is not suitable for this task due to the noise. (b) TopoART and (c) Adjusted-SOINN are sensitive to the external learning environment; the nodes they get fit the learning data set very differently in the two learning environments. LD-SOINN obtains the best fitting results using the least nodes. Moreover, LD-SOINN is not sensitive to the external learning environment. The noise is effectively removed by LD-SOINN. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

260.82 in non-stationary environment of TopoART. The proposed LD-SOINN gets a much better stability: 247.29 in stationary environment and 247.31 in non-stationary environment. Comparing the two environments in Table 3, only LD-SOINN does not lose any person in both environments. We can conclude that LD-SOINN is a much more stable incremental method than Adjusted-SOINN and TopoART.

For unsupervised clustering, we need some nodes as pure as possible as the weight vector of a node is calculated by the input patterns assigned to the node during learning. It is preferable that



(a) 4000 patterns. (b) 8000 patterns. (c) 12 000 patterns.



(d) 16 000 patterns. (e) 20 000 patterns.

Fig. 10. Periodical learning results for a changing data distribution of LD-SOINN. In order to simulate non-stationary environment, the networks are successively trained with samples from the subdistributions A, B, C, D, and E, which are depicted in the upper row. The lower row shows the learning results of LD-SOINN after finishing the respective learning period.

all the input patterns during learning assigned to a prototype are from one object. We give each node an entropy to measure the purity. Assuming there are n objects in the learning set, and node i has m patterns coming from n objects $\{o_1, o_2, \ldots, o_n\}$, the entropy of node i can be calculated as:

$$Entropy(i) = -\sum_{i=1}^{n} p(o_i) \log p(o_i) \tag{26}$$

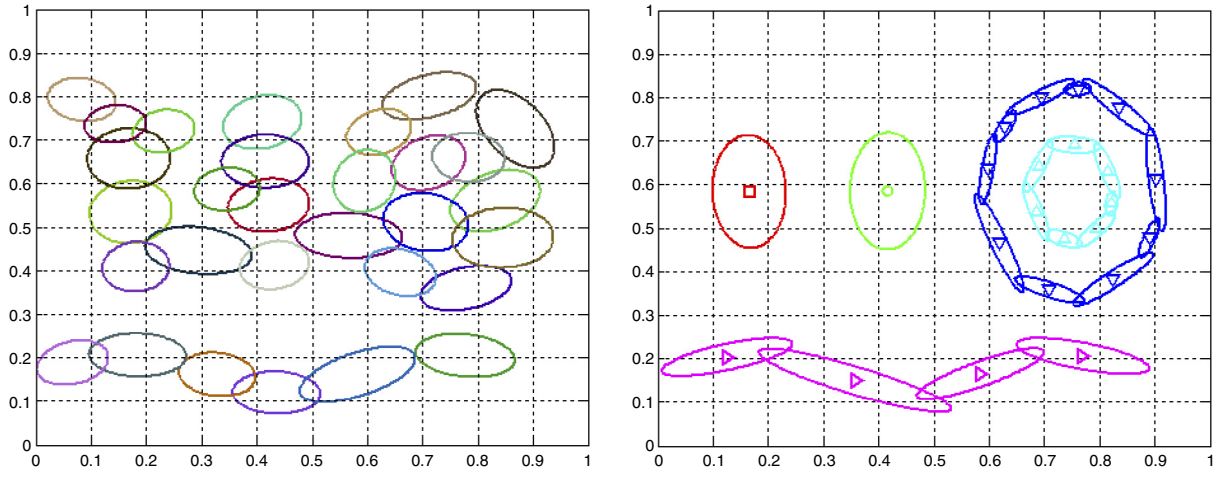
where $p(o_i)$ represents the probability of objects o_i in these m patterns. We calculate the entropy of each node in the learning result and get the mean result. Table 4 shows that LD-SOINN gets a smaller entropy (more pure) than BatchMatrixNG, localPCASOM and Adjusted-SOINN in stationary environment. TopoART gets a smaller entropy value than LD-SOINN, because TopoART loses much useful information: 26.05 persons (as shown in Table 3). Overall, the absolute value 0.012 of LD-SOINN is an acceptable

Table 4
Mean entropy of the nodes in 100 times learning results for ATT_FACE.

Environment	BatchMatrixNG	localPCASOM	oKDE	Adjusted-SOINN	TopoART	LD-SOINN
Stationary	0.29	0.33	–	0.098	0	0.012
Non-stationary	–	–	–	0	0	0

Table 5
Mean recognition ratio of the methods in 100 times learning results for ATT_FACE.

Environment	BatchMatrixNG	localPCASOM	oKDE	Adjusted-SOINN	TopoART	LD-SOINN
Stationary	24.1%	23.4%	–	96.7%	32.0%	98.5%
Non-stationary	–	–	–	89.6%	96.3%	98.5%



(a) *tight* representation. (b) *relaxation* representation.

Fig. 11. The *tight* data representation (a) oKDE vs. the *relaxation* data representation (b) LD-SOINN. The relaxation data representation (b) can generate a more concise representation than the tight data representation (a) on the two Gaussian distributions. For the distribution that cannot be further simplified, relaxation data representation is able to get results comparable with tight data representation, such as the sinusoidal distribution.

result. In non-stationary environment, these three incremental methods get pure nodes in the learning results. One may be concerned that the learning method can get the optimal entropy 0 by setting one prototype per input sample; nevertheless, in Table 2, we can see that the numbers of nodes of the three methods (excluding oKDE) are in the same order of magnitude. Under this condition, using entropy to measure the performance is relatively fair.

Finally, nearest neighbor is used to classify vectors in the original image vector set. The correct recognition ratio is reported as a performance index of the learning methods. As a node may merge patterns from more than one object, the node label is consistent with the most frequently occurring object. The label of the nearest neighbor to the input data should be equal to the label of the input data. As shown in Table 5, the correct recognition ratio of LD-SOINN is higher than that of other methods in both stationary and non-stationary environments.

Next, we do the experiments on some real-world data sets including Segment, Letter, Shuttle, Webspam and KDD99, which differed in data dimensionality, number of classes and instances. The instances in Segment data set are drawn randomly from a database of seven outdoor images; it contains 2310 instances. The Letter data set contains 20,000 instances from twenty-six classes. The Shuttle data set is generated originally to extract comprehensible rules for determining the conditions under which an autolanding would be preferable to manual control of a spacecraft. It contains 58,000 instances from seven classes. The Webspam is the subset used in the Pascal Large Scale Learning Challenge, which contains 350,000 instances from two classes. The KDD99 contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment. KDD99 is a large scale data set that includes 4,898,431 instances for training and 311,029 instances for testing from two classes. The details of the data sets are shown in Table 6.

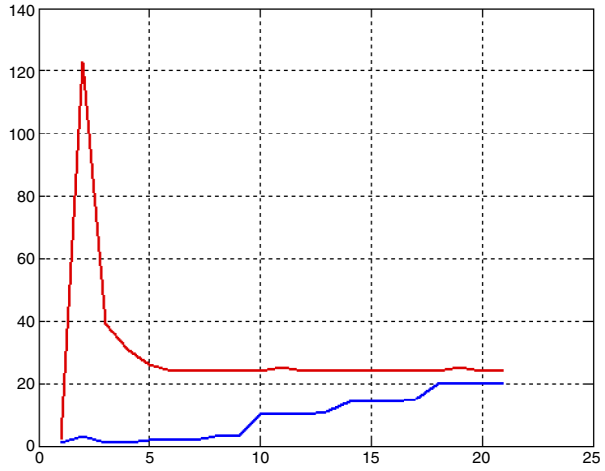


Fig. 12. The trend of the number of the node learned by LD-SOINN in stationary (red line) and non-stationary (blue line) environment. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The parameters of the comparison methods are adjusted following their suggestions to find the best parameters among groups of some candidate parameters. The parameters of LD-SOINN are set as $\sigma = 1e-3$ for Segment, Shuttle, and KDD99, $\sigma = 1e-4$ for Webspam, $\sigma = 1.5e-2$ for Letter, $k = 0.01$, $\lambda = 1000$ for all.

Table 6
Real-world data sets used in the experiments.

Data set	Class	Dimension	# Train	# Test
Segment	7	19	2,000	310
Letter	26	16	16,000	4,000
Shuttle	7	16	43,500	14,500
Webspam	2	254	300,000	50,000
KDD99	2	127	4,898,431	311,029

We define the node number of localPCASOM and BatchMatrixNG the same as LD-SOINN, marked with $*$ in the learning result. Since oKDE is a kind of mixture model, the entropy is not a suitable metric to it. Thus, we do not calculate the entropy for oKDE.

The learning results are shown in Table 7. The N/A in the table shows that the methods do not give the learning result within reasonable time. In the experiment, we set the time as ten days. Comparing LD-SOINN with the three matrix learning methods, namely localPCASOM, BatchMatrixNG and oKDE, we find that LD-SOINN gets much better testing accuracy. For the Segment data set, though oKDE gets less nodes than LD-SOINN, its testing accuracy is much lower than LD-SOINN. Moreover, LD-SOINN can handle the very large scale data set KDD99 very well, while localPCASOM, BatchMatrixNG and oKDE cannot give a learning result within ten days. We can also see that the incremental matrix learning method oKDE cannot even give a learning result on the relatively small data set Webspam within ten days. Thus, LD-SOINN is a more practical incremental matrix learning method than oKDE.

Comparing LD-SOINN with the other two incremental methods, TopoART and Adjusted-SOINN, LD-SOINN gets better testing accuracy. The learned nodes of LD-SOINN are purer than the two methods. Meanwhile, considering all the data sets, LD-SOINN gets more stable learning results.

We also compare the clustering performance with other SOINN family algorithms, namely one-layer SOINN (Shen & Hasegawa, 2006), Enhanced SOINN (Shen et al., 2007), SOINN-PCA (Okada & Nishida, 2011) and SOINN-AML (Okada & Nishida, 2011) on ten real-world data sets. SOINN-PCA integrated SOINN with PCA by using the same approach with SOINN-AML, which is proposed as a comparison method in Okada et al.'s work (Okada & Nishida, 2011). The data sets include 7 UCI Machine Learning Repository data sets (Isolet, Iris, Glass, Letter, Segment, Soybean, Wine) and 3 image data sets (USPS Hull, 1994, COIL-100 Nene, Nayar, & Murase, 1996, Yale Face B Georghiades, Belhumeur, & Kriegman, 2001). For Soybean data set, we removed the instances with unknown values. For YaleFaceB data set, we reduced the image size from 648*480 to 40*30. The details of the data sets are shown in Table 8.

Table 8
Real-world data sets used for clustering.

Data set	Class	Dimension	Instance
ISOLET	26	617	7 797
Iris	3	4	150
Glass	6	10	214
Letter	25	16	20 000
Segment	7	19	2 310
Soybean	15	35	562
Wine	3	13	178
COIL	100	1024	7 200
USPS	10	256	9 298
Yale Face B	10	1200	5 850

We use the Normalized Mutual Information (NMI) (Manning, Raghavan, & Schütze, 2008) as a criteria of the clustering performance. The results of the methods for comparison on these data sets have been reported in Okada et al.'s work (Okada & Nishida, 2011). In this paper, we conduct LD-SOINN on these data sets. In addition, to analyze the contribution of the denoising step in LD-SOINN, we also make a comparison between LD-SOINN with denoising and that without denoising. We first normalized all the data sets to make each feature has mean 0 and standard deviation 1. The parameters of LD-SOINN and that without denoising are set as $\sigma = 4e-2$ for ISOLET, Glass and Iris, $\sigma = 2e-2$ for Letter, $\sigma = 1e-1$ for Segment, Soybean and Wine, $\sigma = 1.5e-1$ for COIL and USPS, $\sigma = 3e-1$ for YaleFaceB. In LD-SOINN with denoising, we set $k = 0.001$, $\lambda = 1000$ for all the data sets. For the data sets that have less samples than the denoising parameter λ, we conduct the denoising process in LD-SOINN with denoising when all the samples have been learned. All the experiments are performed for 20 times with random input order and the average NMI are reported. The clustering results are shown in Table 9. In Table 9, LD-SOINN without the denoising process is denoted as LD-SOINN #, and that with denoising process is denoted as LD-SOINN.

In Table 9, LD-SOINN outperforms other methods in most cases. Especially in Iris, Letter, Segment, Wine and COIL data sets, LD-SOINN improves the clustering performance greatly. However, we also find that LD-SOINN does not perform as well on some other sparse data sets, for example the YaleFaceB data set which contains only 5850 samples in the 1200-dimensional feature space. Since LD-SOINN is a density-based clustering method which relies on the dense data distributions in local regions, it is hard for LD-SOINN to reduce the number of the clusters by merging or connecting the nodes on such data sets, which greatly undermines its performance under the criteria of NMI.

Meanwhile, from Table 9, we find LD-SOINN with denoising outperforms that without denoising in most of these data sets. As

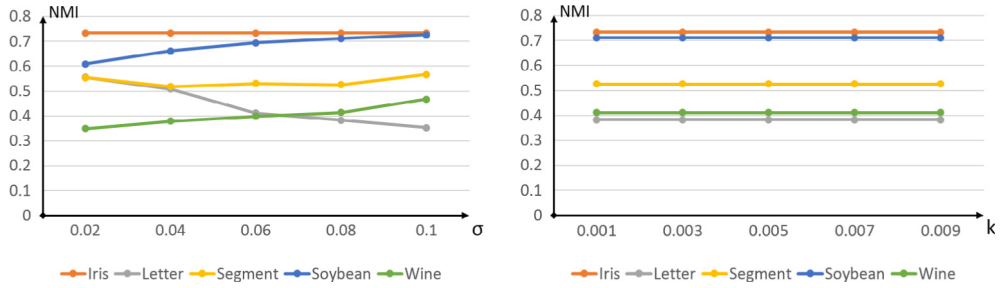
Table 7
Learning results on the real-world data sets. N/A represents that the methods do not give the learning result within 10 days. The best performance on each data set is in bold.

Data set		localPCASOM	BatchMatrixNG	oKDE	Adjusted-SOINN	TopoART	LD-SOINN
Segment	Node number	465*	465*	**24**	376	242	465
	Entropy	1.95	0.97	–	0.15	0.10	**0.029**
	Accuracy	31.33%	35.16%	73.55%	85.01%	88.39%	**90.32%**
Letter	Node number	6562*	6562*	1935	1546	**500**	6562
	Entropy	3.26	2.93	–	2.46	0.27	**0.025**
	Accuracy	28.9%	31.3%	84.18%	74.80%	61.50%	**89.85%**
Shuttle	Node number	9*	9*	30	69	43	**9**
	Entropy	0.67	0.69	–	0.23	0.27	**0.17**
	Accuracy	79.16%	81.37%	90.66%	90.39%	70.54%	**92.96%**
Webspam	Node number	745*	745*		3531	4643	**745**
	Entropy	0.53	0.49	N/A	0.73	0.46	**0.21**
	Accuracy	65.78%	67.25%		85.45%	83.05%	**86.50%**
KDD99	Node number				127	149	**32**
	Entropy	N/A	N/A	N/A	0.030	0.0052	**0.0016**
	Accuracy				92.15%	92.10%	**92.81%**

We define the node number of localPCASOM and BatchMatrixNG the same as LD-SOINN, marked with * in the learning result.

Table 9
Average NMI on nine real-world data sets. The best performance on each data set is in bold.

Data set	ISOLET	Iris	Glass	Letter	Segment	Soybean	Wine	COIL	USPS	YaleB
SOINN	0.635	0.554	0.584	0.229	0.382	0.672	0.276	0.607	0.507	0.792
ESOINN	0.662	0.633	0.515	0.376	0.406	0.571	0.260	0.513	0.607	0.782
SOINN-PCA	0.284	0.504	0.534	0.212	0.353	0.571	0.240	0.436	0.153	0.730
SOINN-AML	**0.692**	0.618	**0.621**	0.372	0.406	0.693	0.320	0.665	**0.637**	**0.806**
LD-SOINN #	0.579	0.627	0.487	0.553	0.543	0.715	**0.467**	**0.751**	0.498	0.592
LD-SOINN	0.622	**0.734**	0.487	**0.556**	**0.567**	**0.726**	**0.467**	0.739	0.516	0.629



(a) NMI results with changing σ.

(b) NMI results with changing k.

(c) NMI results with changing λ.

(d) NMI results with changing q.

Fig. 13. The NMI results by changing each parameter (σ, k, λ, q) of LD-SOINN. In Fig. 13(a), σ ranges from 0.02 to 0.1, $k = 0.001$, $\lambda = 1000$, $q = 0.95$. In Fig. 13(b), k ranges from 0.001 to 0.009, $\sigma = 0.08$, $\lambda = 200$, $q = 0.95$. In Fig. 13(c), λ ranges from 100 to 1300, $\sigma = 0.08$, $k = 0.001$, $q = 0.95$. In Fig. 13(d), q ranges from 0.90 to 0.98, $\sigma = 0.08$, $k = 0.005$, $\lambda = 200$.

the denoising step removes the node with smaller number of input patterns belonging to it, less nodes are produced in LD-SOINN with denoising. These nodes construct a more concise representation of the data distribution. Whereas in LD-SOINN without denoising, the existence of the inferior nodes that should be removed in the denoising step would aggrandize the number of clusters, some of which contains only a few instances. Thus, these nodes are likely to undermine the clustering performance. Moreover, in some large data sets like COIL and USPS, removing the noisy nodes after a period of training can control the growth of the nodes and make the training much faster.

To examine the parameters' influence on the performance of LD-SOINN, we also do the experiments with changing each parameter, namely σ, k, λ, q, on the 5 UCI data sets (Iris, Letter, Segment, Soybean, Wine). The NMI results on such data sets are shown in Fig. 13. Fig. 13(b) and (c) shows that the denoising parameters (k, λ) have little influence on the performance of LD-SOINN. And Fig. 13(a) and (d) shows that the parameter σ and q indeed influence the performance, but in a slight degree. From observation, we find that in clustering tasks, LD-SOINN prefers a small σ for the dense data sets (like Letter data set), while prefers a large σ for the sparse data sets (like Soybean and Wine data sets). Then we can conclude that LD-SOINN is not very sensitive to the parameters.

Overall, the experimental section indicates that the proposed LD-SOINN has a good data expression ability, generalization ability and stability.

5. Conclusion

This paper presents an unsupervised incremental learning neural network based on local distribution learning called Local Distribution Self-Organizing Incremental Neural Network (LD-SOINN). It aims to automatically obtain suitable expression of learning data in an incremental (or on-line) self-organizing way without a priori knowledge such as the structure of the network. Each node in the network records the local data distribution of the original learning data. Through giving each node an adaptive vigilance parameter with statistics as theoretical support, LD-SOINN is able to add new nodes for new knowledge automatically, and the number of nodes will not grow unlimitedly. While the learning process continues, nodes that are close to each other and have similar principal components will be merged to get a relaxation data representation. Denoising processing based on density is designed to reduce the influence of noise. The experiments for both artificial data set and real-world data set show that the proposed method is effective.

Acknowledgments

This work is supported in part by the 973 Program of China (2015CB351705) and the National Science Foundation of China under Grant Nos. (61375064, 61373001) and Jiangsu NSF grant (BK20131279).

Appendix A

When a pattern x is assigned to a node i^*: $\langle n, c, M, H\rangle$, the new value of the 4-tuple can be calculated in the form of recursive relation (6). $n_{new} = n+1$ and $H_{new} = \varepsilon_{n_{new}}\chi^2_{d,q}$ are obviously valid. For c_{new}, we calculate it as:

$$\begin{aligned} c_{new} &= (nc + x)/(n+1) \\ &= c + (x-c)/(n+1). \end{aligned} \tag{A.1}$$

For M_{new}, we first assume that:

$$M = \sum_{i=1}^{n}(x_i - c)(x_i - c)^T/n \tag{A.2}$$

where $\{x_i, 1 \le i \le n\}$ are the original patterns belonging to node i^*. When the pattern x is assigned to node i^*, we get $x_{n+1} = x$. Then the new covariance matrix M_{new} is calculated as follows:

$$\begin{aligned} M_{new} &= \sum_{i=1}^{n+1}(x_i - c_{new})(x_i - c_{new})^T/(n+1) \\ &= \sum_{i=1}^{n+1}[(x_i - c)(x_i - c)^T/(n+1) \\ &\quad - ((x-c)(x_i-c)^T + (x_i-c)(x-c)^T)/(n+1)^2 \\ &\quad + (x-c)(x-c)^T/(n+1)^3] \\ &= n/(n+1)^2(x-c)(x-c)^T + n/(n+1)M \\ &= M + [n(x-c)(x-c)^T - (n+1)M]/(n+1)^2. \end{aligned} \tag{A.3}$$

Thus Eq. (6) is proved.

Appendix B

For Eq. (15), $n_m = n_i + n_j$ and $H_m = \varepsilon_{n_m}\chi^2_{d,q}$ are obviously valid. For M_m, assume we know M_i and M_j as:

$$M_i = \sum_{k=1}^{n_i}(x_k - c_i)(x_k - c_i)^T/n_i \tag{B.1}$$

$$M_j = \sum_{k=1}^{n_j}(x_k - c_j)(x_k - c_j)^T/n_j. \tag{B.2}$$

Now, we assume that the merge node m contains the patterns from i and j; for simplicity note that $\{x_k, 1 \le k \le n_i\}$ are from i and $\{x_k, n_i + 1 \le k \le n_i + n_j\}$ are from j. And the covariance matrix M_m of the merge node m can be calculated as:

$$\begin{aligned} M_m &= \sum_{k=1}^{n_i+n_j}(x_k - c_m)(x_k - c_m)^T/(n_i + n_j) \\ &= \sum_{k=1}^{n_i}(x_k - c_m)(x_k - c_m)^T/(n_i + n_j) \\ &\quad + \sum_{k=n_i+1}^{n_i+n_j}(x_k - c_m)(x_k - c_m)^T/(n_i + n_j) \\ &= \frac{n_i M_i + n_i(c_i - c_m)(c_i - c_m)^T}{n_i + n_j} \\ &\quad + \frac{n_j M_j + n_j(c_j - c_m)(c_j - c_m)^T}{n_i + n_j} \\ &= \frac{n_i}{n_m}(M_i + (c_m - c_i)(c_m - c_i)^T) \\ &\quad + \frac{n_j}{n_m}(M_j + (c_m - c_j)(c_m - c_j)^T). \end{aligned} \tag{B.3}$$

Thus Eq. (15) is proved.

Appendix C

Considering the hyper-ellipsoid boundary equation of each node in formula (1):

$$\sqrt{(x-c)^T M^{-1}(x-c)} = H \tag{C.1}$$

where $x \in \mathbb{R}^d$, mark $x = [x_1, x_2, \ldots, x_d]$ and $c = [c_1, c_2, \ldots, c_d]$. In order to get the standard form of hyper-ellipsoid equations, we get the Singular Value Decomposition of symmetric positive definite matrix M as:

$$M = E^T \begin{pmatrix} \lambda_1 & \cdots & \cdots & 0 \\ \vdots & \lambda_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \lambda_d \end{pmatrix} E \tag{C.2}$$

where E is orthogonal matrix, assume $e_1, e_2, \ldots, e_d$ are the column vector of E. Substituting Eq. (C.1) into Eq. (B.3), we get the standard form Ω of hyper-ellipsoid Eq. (B.3) as:

$$\Omega : \sum_{i=1}^{d}\frac{(e_i^T(x_i - c_i))^2}{\lambda_i} = H^2. \tag{C.3}$$

Then we use mathematical induction to prove that:

$$\begin{aligned} Volume(\Omega) &= \int_{\Omega} d\Omega \\ &= 2^{\left[\frac{d+1}{2}\right]}\pi^{\left[\frac{d}{2}\right]}\left(\prod_{i=0}^{[d/2]-1}\frac{1}{d-2i}\right)\sqrt{\prod_{i=1}^{d}\lambda_i H^d} \end{aligned} \tag{C.4}$$

for $d = 1, 2, 3$, it is easy to verify formula (C.4). Assuming formula (C.4) is correct when $d = k$, we check the situation when $d = k+1$:

$$\begin{aligned} Volume(\Omega_{k+1}) &= \int_{-\sqrt{\lambda_1}}^{\sqrt{\lambda_1}} dx_1 \int_{\Omega_k(1 - x_1^2/\lambda_1)} d\Omega \\ &= \sqrt{\lambda}_1 * Volume(\Omega_k)\int_{-\pi/2}^{\pi/2}\cos\theta^{k+1}\, d\theta. \end{aligned} \tag{C.5}$$

At the same time, we have the following recursion formula:

$$\int_{-\pi/2}^{\pi/2}\cos\theta^k\, d\theta = \frac{k-1}{k}\int_{-\pi/2}^{\pi/2}\cos\theta^{k-2}\, d\theta. \tag{C.6}$$

Substituting formula (C.6) into formula (C.5), we can get Eq. (C.3). It is obvious that $\prod_{i=1}^{d}\lambda_i = |M|$; changing $\prod_{i=1}^{d}\lambda_i$ in formula (C.3) to $|M|$, we get formula (14).

References

Alahakoon, D., Halgamuge, S.K., & Srinivasan, B. (1998). A self growing cluster development approach to data mining. In *Proceedings of IEEE international conference on systems, man, and cybernetics* (pp. 2901–2906).

Alahakoon, D., Halgamuge, S. K., & Srinivasan, B. (2000). Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Transactions on Neural Networks*, *11*(3), 601–614.

Anagnostopoulos, G.C., & Georgiopoulos, M. (2001). Ellipsoid art and artmap for incremental clustering and classification. In *Proceedings of the international joint conference on neural networks* (pp. 1221–1226).

Araujo, A. F. R., & Rego, R. L. M. E. (2013). Self-organizing maps with a time-varying structure. *ACM Computing Surveys*, *46*(1), 1–38.

Arnonkijpanich, B., Hasenfuss, A., & Hammer, B. (2011). Local matrix adaptation in topographic neural maps. *Neurocomputing*, *74*(4), 522–539.

Carpenter, G. A., & Grossberg, S. (1988). The art of adaptive pattern recognition by self-organising neural network. *IEEE Computer*, *21*(3), 77–88.

Carpenter, G. A., Grossberg, S., & Rosen, D. B. (1991). Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, *4*(6), 759–771.

Chin, T., & Suter, D. (2007). Incremental kernel principal component analysis. *IEEE Transactions on Image Processing*, *16*(6), 1662–1674.
Figueiredo, M. A. T., & Jain, A. K. (2002). Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *24*(3), 381–396.
Fritzke, B. (1994). Growing cell structures–a self-organizing network for unsupervised and supervised learning. *Neural Networks*, *7*(9), 1449–1460.
Fritzke, B. (1995). A growing neural gas network learns topologies. In *Proceedings of the 1995 advances in neural information processing systems* (pp. 625–632).
Georghiades, A. S., Belhumeur, P. N., & Kriegman, D. J. (2001). From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *23*(6), 643–660.
Hall, P., Marshall, D., & Martin, R. (1998). Incremental eigenanalysis for classification. In *British machine vision conference* (pp. 286–295).
Heskes, T. (2001). Self-organizing maps, vector quantization, and mixture modeling. *IEEE Transactions on Neural Networks*, *12*(6), 1299–1305.
Huang, D., Yi, Z., & Pu, X. (2008). A new local pca-som algorithm. *Neurocomputing*, *71*(16–18), 3544–3552.
Hull, J. (1994). A database for handwritten text recognition research. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *16*(5), 550–554.
Kambhatla, N., & Leen, T. (1997). Dimension reduction by local principal component analysis. *Neural Computation*, *9*(7), 1493–1516.
Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, *43*(1), 59–69.
Kohonen, T. (1995). The adaptive-subspace som (assom) and its use for the implementation of invariant feature detection. In *Proceedings of the 1995 international conference on artificial neural networks* (pp. 3–10).
Kristan, M., Leonardis, A., & Skočaj, D. (2011). Multivariate online kernel density estimation with Gaussian kernels. *Pattern Recognition*, *44*(10–11), 2630–2642.
Lloyd, S. P. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory*, *28*(2), 129–137.
López-Rubio, E. (2010). Probabilistic self-organizing maps for continuous data. *IEEE Transactions on Neural Networks*, *21*(10), 1543–1554.
López-Rubio, E., Muñoz-Pérez, J., & Gómez-Ruiz, A. (2004). A principal components analysis self-organizing map. *Neural Networks*, *17*(2), 261–270.
Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.
Marsland, S., Shapiro, J., & Nehmzow, U. (2002). A self-organising network that grows when required. *Neural Networks*, *15*(8–9), 1041–1058.
Martinetz, T. M., Berkovich, S. G., & Schulten, K. J. (1993). Neural gas network for vector quantization and its application to timeseries prediction. *IEEE Transactions on Neural Networks*, *4*(4), 556–558.
Martinetz, T., & Schulten, K. (1994). Topology representing networks. *Neural Networks*, *7*(3), 507–552.
Nene, S. A., Nayar, S. K., & Murase, H. (1996). *Columbia object image library (coil-100), Tech. rep., CUCS-006-96*.
Okada, S., & Nishida, T. (2011). Online incremental clustering with distance metric learning for high dimensional data. In *Proceedings of the international joint conference on neural networks* (pp. 2047–2054).
Ouyang, Q., Shen, F., & Zhao, J. (2012). A local distribution net for data clustering. In *PRICAI 2012: Trends in artificial intelligence* (pp. 411–422). Springer.
Ozawa, S., Pang, S., & Kasabov, N. K. (2008). Incremental learning of chunk data for online pattern classification systems. *IEEE Transactions on Neural Networks*, *19*(6), 1061–1074.
Piastra, M. (2009). A growing self-organizing network for reconstructing curves and surfaces. In *Proceedings of the international joint conference on neural networks* (pp. 2533–2540).
Shen, F., & Hasegawa, O. (2006). An incremental network for on-line unsupervised classification and topology learning. *Neural Networks*, *19*(1), 90–106.
Shen, F., & Hasegawa, O. (2008). A fast nearest neighbor classifier based on self-organizing incremental neural network. *Neural Networks*, *21*(10), 1537–1547.
Shen, F., Ogurab, T., & Hasegawa, O. (2007). An enhanced self-organizing incremental neural network for online unsupervised learning. *Neural Networks*, *20*(8), 893–903.
Tscherepanow, M., Kortkamp, M., & Kammer, M. (2011). A hierarchical art network for the stable incremental learning of topological structures and associations from noisy data. *Neural Networks*, *24*(8), 906–916.
Villmann, T., Der, R., Herrmann, M., & Martinetz, T. M. (1997). Topology preservation in self-organizing feature maps: exact definition and measurement. *IEEE Transactions on Neural Networks*, *8*(2), 256–266.
Weingessel, A., & Hornik, K. (2000). Local pca algorithms. *IEEE Transactions on Neural Networks*, *11*(6), 1242–1250.
Xing, Y., Cao, T., Shen, F., & Zhao, J. (2015). An incremental local distribution network for unsupervised learning. In *Pacific-Asia conference on knowledge discovery and data mining* (pp. 646–658).
Ye, J., Zhao, Z., & Liu, H. (2007). Adaptive distance metric learning for clustering. In *IEEE conference on computer vision and pattern recognition* (pp. 1–7).
Zhang, H., Xiao, X., & Hasegawa, O. (2014). A load-balancing self-organizing incremental neural network. *IEEE Transactions on Neural Networks and Learning Systems*, *25*(6), 1096–1105.