An Incremental Local Distribution Network for Unsupervised Learning

Youlu Xing[1], Tongyi Cao[2], Ke Zhou[3], Furao Shen[1(✉)], and Jinxi Zhao[1]

[1] National Key Laboratory for Novel Software Technology, Department of Computer Science and Technology at Nanjing University, Nanjing, China
youluxing@sina.com, {frshen,jxzhao}@nju.edu.cn
[2] School of Physics at Nanjing University, Nanjing, China
caotongyi.is.tc@gmail.com
[3] School of Statistics at University of International Business and Economics, Bejing, China
02417@uibe.edu.cn

Abstract. We present an **I**ncremental **L**ocal **D**istribution **N**etwork (**ILDN**) for unsupervised learning, which combines the merits of matrix learning and incremental learning. It stores local distribution information in each node with covariant matrix and uses a vigilance parameter with statistical support to decide whether to extend the network. It has a statistics based merging mechanism and thus can obtain a precise and concise representation of the learning data called relaxation representation. Moreover, the denoising process based on data density makes ILDN robust to noise and practically useful. Experiments on artificial and real-world data in both "closed" and "open-ended" environment show the better accuracy, conciseness, and efficiency of ILDN over other methods.

Keywords: Incremental learning · Matrix learning · Relaxation representation

1 Introduction

In the field of unsupervised learning, many algorithms are designed to extract information from the distribution of data. Classic methods include k-means [1] and Neural Gas [2], which use fixed number of nodes to get different clusters. Self-Organizing Map [3] and Topology Representing Networks [4] represent the distribution and topological structure of the data with some given nodes.

Two drawbacks are obvious for these early methods. First, each node stores the mean feature vector of patterns belonged to the node and the metric is Euclidean. Correspondingly, each node is a simple unit with isotropic form and spherical class boundary, and thus has a poor description ability. Second, these methods need a predefined structure or number of nodes that requires additional knowledge of the data which is often hard to know. Also, the fixed structures render them unable to perform incremental learning or to handle the "*open-ended*" environment, i.e. data from new distributions may occur during learning.

T. Cao et al. (Eds.): PAKDD 2015, Part I, LNAI 9077, pp. 646–658, 2015.
DOI: 10.1007/978-3-319-18038-0_50

Two kinds of improvements have been made corresponding to these two problems, namely matrix learning and incremental networks. For the first problem, in order to obtain a precise yet concise representation of the data, a more expressive node is preferred, often based on mixture model, including PCASOM [5], Self-Organizing Mixture Network [6], localPCASOM [7], and MatrixNG and MatrixSOM [8]. López-Rubio [9] gave a detailed review about these Mixture Model based Self-Organizing Maps which they called the Probabilistic Self-Organizing Map. For the second problem, many "growing networks" or "incremental networks" are proposed. Some grow after a fixed number of inputs learned such as GNG [11] and GSOM [12]; some use an adaptive threshold including GWR [13], Adjusted-SOINN [14], and TopoART [15]. Araujo and Rego [16] gave a detailed review about these incremental Self-Organizing Maps.

On one hand, though matrix learning methods record rich local distribution information and consider the anisotropy on different basis vectors, they have a common shortcoming - they cannot deal with the Stability-Plasticity Dilemma [10], i.e. many of these methods cannot learn data from new distributions after they are trained on the current data set; the other methods is able to learn data from new distributions but the previous learned knowledge will be forgotten, known as the "Catastrophic Forgetting". Thus, they can only work in a "*closed*" environment, with no new distributions occurring during learning. On the other hand, incremental networks process flexible structures that can adapt well for various data and environments, but they lose much useful information of the original learning data. Recently, an online Kernel Density Estimator (oKDE) [17] is proposed to introduce the Kernel Density Estimator to online learning. However, the learning environment ("*closed*" or "*open-ended*") must be known in advance to set different parameters.

We propose an **I**ncremental **L**ocal **D**istribution **N**etwork (**ILDN**), which, by combining the advantages of matrix learning and incremental learning, is able to obtain a precise and concise representation of the data as well as learning incrementally without forgetting previous knowledge. In summary, the characteristics of ILDN are:

(1) By storing in each node the covariant matrix to record rich local distribution information of the learning data, and adopting statistically supported node merging and denoising criterions, ILDN is able to obtain a precise and concise representation of the learning data, called a relaxation data representation.

(2) Through giving each node an adaptive vigilance parameter with statistics theoretical supporting, ILDN is able to learn new distributions data effectively without forgetting the previous learnt but still useful knowledge. That is, ILDN can handle the Stability-Plasticity Dilemma effectively.

2 Incremental Local Distribution Network

ILDN is an online incremental learning model which combines the advantages of matrix learning and incremental networks. The nodes in the network record not only the weight vector but also the data distribution information around its

local region, i.e. the covariance matrix. Nodes which 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 is called a relaxation data representation.

In ILDN, each node i 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 belonged to node i. H_i is a vigilance parameter to decide whether an input pattern belongs to node i, it dynamically changes with the learning process. Assume that ILDN receives d-dimensional data $x \in \mathbb{R}^d$, the node can be described as a hyper-ellipsoid region using c_i, M_i and H_i:

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

The entire workflow of ILDN is as follows: when an input pattern comes, ILDN first conducts the Node Activation to find some activated nodes which 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 ILDN 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, ILDN 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. When all the steps above are done, ILDN will process the next input pattern. Denoising is implemented every λ patterns are learned. When the learning process is finished or users want the learning result, ILDN will Cluster the learned nodes and output the learning result.

2.1 Node Activation

When an input pattern x comes, ILDN first calculates the Mahalanobis distance between x and all node $i \in N$:

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

where N is the set of nodes, $|N|$ represents the total number of the 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.

2.2 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 be 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 compact convergence to 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 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. ILDN 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 as:

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

2.3 Topology Maintaining

A topology preserving feature map is determined by a mapping Φ from a manifold $\mathcal{M}$ onto the vertices (or nodes) $i \in N$ of a graph (or 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 (or nodes) i and j that are adjacent (or connected) in the graph or network $\mathcal{G}$ [4].

Some methods achieve the topology preserving feature map through a predefined structure $\mathcal{G}$ such as SOM [3]. In this paper, the connections between nodes of $\mathcal{G}$ are built 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 input pattern x. Thus, if no connection exists between node i and j in S, ILDN 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 is able to organize the nodes into groups to represent different topology of the learning data.

2.4 Node Merging

As learning continues, there may be some nodes closing to each other and having similar principal components. Such nodes will be merged to obtain a concise local representation. At merging stage, two nodes will merge if the following two conditions are satisfied:

(i) 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.

$$Volume(m) < Volume(i) + Volume(j) \tag{7}$$

If the above conditions are satisfied for node i and j, we merge i and j and let m represent the data in i and j collectively:

$$\begin{aligned} c_m &= (n_i c_i + n_j c_j)/n_m; \quad n_m = n_i + n_j; \quad H_m = \varepsilon_{n_m} \chi^2_{d,q} \\ M_m &= \tfrac{n_i}{n_m}(M_i + (c_m - c_i)(c_m - c_i)^T) + \tfrac{n_j}{n_m}(M_j + (c_m - c_j)(c_m - c_j)^T) \end{aligned} \tag{8}$$

In practice, we only merge the winner node and its neighbors when a pattern is fed into ILDN. After merging, all the connections with original nodes in C are attached to the new node.

2.5 Denoising

The data from the learning environment may contain noise. Some nodes may be created by these noise data. Since ILDN records the distribution density of each node, we can use this information to judge whether a node is a noise node.

After every λ patterns learned, ILDN first calculates the mean value of the number of the input patterns belonged to each node as:

$$Mean = \sum_{i=1}^{|N|} n_i / |N| \tag{9}$$

where $|N|$ represents the total number of the nodes, n_i is the number of input patterns belonged to node 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 $0 \leq k \leq 1$ control the intensity of denoising. After denoising, all the connections with deleted nodes in C are also deleted.

2.6 Cluster

In [2], it is proved that the competitive Hebbian rule forms perfect topology preserving map of a manifold if the data is dense enough. Based on this opinion, we take each cluster as a manifold, therefore we can find different connected node domains as different clusters. Algorithm 1 shows the details of the clustering method.

2.7 Complete Algorithm of ILDN

As a summary for this section, we give the complete algorithm of ILDN (Algorithm 2).

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.

Algorithm 2. Incremental Local Distribution Network

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 is the nodes which 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^* 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 (8).
8: If the number of input patterns presented so far is an integer multiple of parameter λ. Denoising as in **Section 2.5**.
9: If the learning process is not finished, go to Step 2 to continue unsupervised online learning.
10: Cluster using Algorithm 1 and output the learning result.

3 Analysis

3.1 The Expansivity of the Nodes

As each node records the local regional distribution, we can assume that the patterns belonged to one node are generated by a Gaussian distribution $\mathcal{N}(c_X, \Sigma_X)$ where c_X is the center and Σ_X the covariance matrix. The hyper-ellipsoid boundary equation of each node is:

$$(x - c_X)^T \Sigma_X^{-1} (x - c_X) = H^2 \tag{10}$$

Let $K = H^2$, then K is a χ^2 distribution with d degrees of freedom. Giving a confidence q, the patterns from random variables X lie in the hyper-ellipsoid drawn by K can be described as:

$$P\{(x - c_X)^T \Sigma_X^{-1} (x - c_X) < K\} = q \tag{11}$$

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 let the hyper-ellipsoid has 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 a stable state. We set $\varepsilon_{n_i} = (1 + 2 * 1.05^{1-n_i})$.

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

3.2 The Relaxation Data Representation

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

$$Volume(n) = 2^{[\frac{d+1}{2}]}\pi^{[\frac{d}{2}]}\left(\prod_{i=0}^{[d/2]-1}\frac{1}{d-2i}\right)\sqrt{|M_n|}H_n^d \tag{12}$$

where M_n and H_n are the covariance matrix and vigilance parameter of node n. $|M_n|$ represents the determinant of M_n, $[\cdot]$ represents the rounding operation. d is the dimension of the sample space. However, $|M|$ in formula (12) may be very close to 0 in some high dimensional task and thus not suitable to calculate the volume directly with it. To avoid directly calculate the volume with formula (12), 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:

$$M = E^T diag(\lambda_1, \lambda_2, ..., \lambda_d)E \tag{13}$$

where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$, and we get λ_1^i, λ_2^i,..., λ_d^i for node i, λ_1^j, λ_2^j,..., λ_d^j for node j and λ_1^m, λ_2^m,..., λ_d^m for node m.

Then we find a truncated position t of all singular value with a predefined scaling factor ρ: p=$\underset{1\leq p\leq d}{\operatorname{argmin}}\sum_{i=1}^{t}\lambda_i \geq \rho\sum_{i=1}^{d}\lambda_i$. 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 $\lambda_1^k \times \lambda_2^k \times \ldots \times \lambda_t^k$, $k = i, j, m$. Substituting $|M_k|$ (where $k = i, j, m$) into formula (12) and replace d with t, we get $Volume(i)$, $Volume(j)$ and $Volume(m)$. Substituting these three volumes and after a series of simplification, we get an equivalence merging condition:

$$\sqrt{\frac{\lambda_1^i H_i}{\lambda_1^m H_m}\cdot\frac{\lambda_2^i H_i}{\lambda_2^m H_m}\cdots\frac{\lambda_t^i H_i}{\lambda_t^m H_m}} + \sqrt{\frac{\lambda_1^j H_j}{\lambda_1^m H_m}\cdot\frac{\lambda_2^j H_j}{\lambda_2^m H_m}\cdots\frac{\lambda_t^j H_j}{\lambda_t^m H_m}} \geq 1 \tag{14}$$

In practice, we use formula (14) to judge the merging condition (ii). If formula (14) is not satisfied, node i and j will remain unchanged.

At the node merging step, we have two candidate data representation that are: (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:

$$f_1 : \begin{cases} \sqrt{(x-c_i)^T M_i^{-1}(x-c_i)} < H_i & x \in R_i \\ \sqrt{(x-c_j)^T M_j^{-1}(x-c_j)} < H_j & x \in R_j \end{cases} \tag{15}$$

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

$$f_2 : \sqrt{(x-c_m)^T M_m^{-1}(x-c_m)} < H_m \qquad x \in R_i \cup R_j \tag{16}$$

When assume 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 to q (usually $q \geq 90\%$). Meanwhile, according to node merging condition (ii), the volume of node m is less than the total volume of node i and j. Thus, we get a much concise data representation on domain $R_i \cup R_j$.

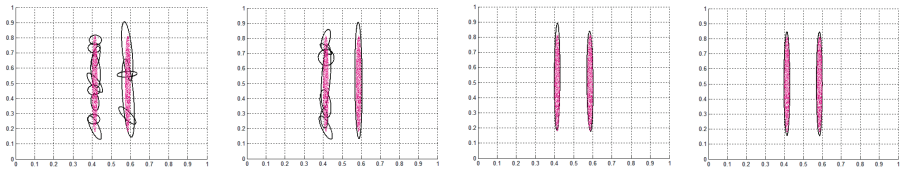
Comparing of the two representations, f_1 uses more parameters than f_2, and the two regions in f_1 overlap each other. Thus, the representation f_1 is *tight* and we call f_1 as a *tight* data representation. On the other hand, the representation f_2 expresses the data distribution more concisely than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

4 Experiments

As ILDN aims to combine the advantages of incremental learning and matrix learning, we compare it with some classical and state-of-the-art methods of matrix learning and incremental learning. The matrix learning methods include localPCASOM [7], BatchMatrixNG [8] and oKDE (oKDE is also an incremental learning method) [17]. The incremental learning methods include TopoART [15] and Adjusted-SOINN (ASOINN) [14].

4.1 Artificial Data

Observe the Periodical Learning Results. We use the artificial data which is distributed in two belt areas (also used in [5]). Each belt area represents a



(a) After 50 patterns (b) After 100 patterns (c) After 200 patterns (d) After 4000 patterns

Fig. 1. Periodical learning results of ILDN. The pink area represents the distribution of the learning data. Ellipses with black boundary-line are the learning result.

cluster and generates samples uniformly. Parameters of ILDN are $\chi^2_{d,q} = \chi^2_{(2,0.90)}$, $\sigma = 1e\text{-}5$, $k = 0.01$, $\lambda = 1000$. Fig. 1 illustrates the learning process of ILDN. At the early stage, many ellipsoids are generated to cover the learning data set (Fig. 1(a)). With the learning process continues, some ellipsoids are merged together, as at the 100 patterns stage (Fig. 1(b)). After 200 patterns, all ellipsoids are merged into 2 ellipsoids (Fig. 1(c)). Finally, ILDN gets 2 ellipsoids which fits the original data set very well. ILDN does not need to predetermine the number of the nodes, it automatically generates 2 nodes in this task.

Work in Complex Environment. In this section, we conduct our experiment on the data set shown in Fig. 2. The dataset is separated into five parts containing 20000 samples in total. 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% Gaussian noise and random noise to the dataset. Noise is distributed over the entire data set.

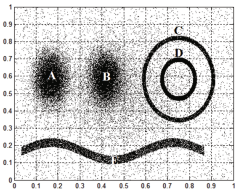


Fig. 2. Artificial data set used for the experiment. 10% noise is distributed over the entire data set.

The experiments are conducted in two environments. In the "closed" environment, patterns are randomly selected from the whole learning set. In the "open-ended" environments, five parts of the data are presented successively. In stage I, i.e. step 1 to 4000, patterns are chosen randomly from data set A. In stage II, i.e. step 4001 to 8000, the environment changes and patterns from B are chosen, etc.

We set the parameters of localPACSOM as $N = 20$, ε=0.01, H=1.5; BatchMatrix as $N = 20$, $epoch = 1000$; ASOINN as λ=200, age_{max}=25, c=0.5; TopoART as β_{sbm}=0.32, ρ=0.96, φ=5, τ=100; oKDE as $D_{th} = 0.01$, $N = 10$, $f = 1$ for the "closed" environment and $f = 0.99$ for the "open-ended" environment. ILDN as $\chi^2_{(d,q)}$=$\chi^2_{(2,0.90)}$, σ=1e-5, k=0.5, $\lambda = 1000$.

Fig. 3 shows the results. localPCASOM and BatchMatrix suffer from the Stability-Plasticity Dilemma. oKDE is vulnerable to noise. ILDN obtains best fitting with least nodes. On one hand, ILDN can merge some local small ellipsoids into a big one, leading to a more concise data representation than other matrix learning methods and can learn incrementally. On the other, ILDN learns the local distribution to describe original data while other incremental networks have to use a large number of nodes.

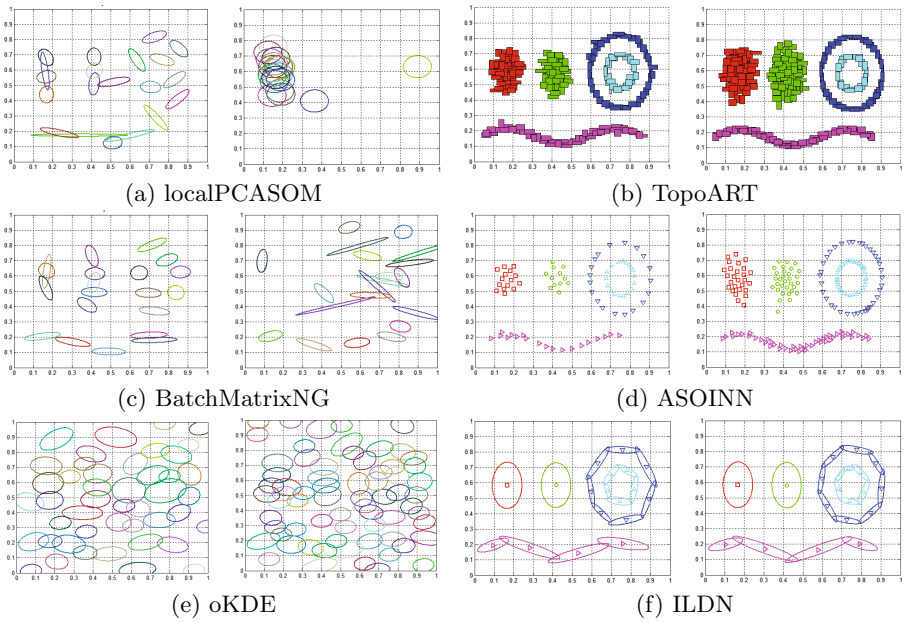


Fig. 3. Comparing results in the closed and open-ended environment. The left column of a method is the result in the closed environment, the right is the open-ended environment.

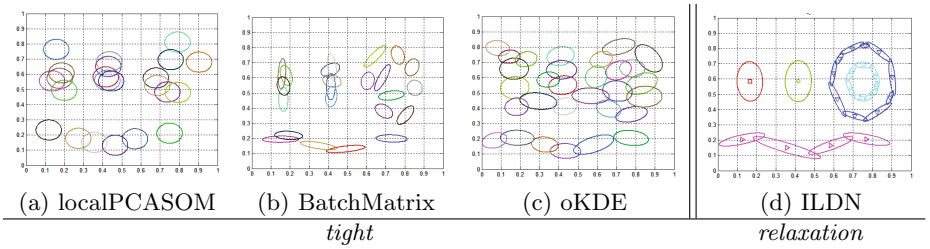


Fig. 4. The *relaxation* data representation vs. the *tight* data representation

In order to clearly observe the performance of the *relaxation* and the *tight* data representations, we use the dataset in Fig. 2 without noise to test localPCASOM, BatchMatrix, oKDE, and ILDN in the "closed" environment. The result is shown in Fig. 4. The *relaxation* representation can generate a more concise representation than the *tight* representation on the two Gaussian distributions. For the distribution which cannot be further simplified, the *relaxation* representation is able to get comparable result with the *tight* representation, such as the sinusoidal distribution. This experiment demonstrates that the *relaxation* data representation gets a much concise representation than the *tight* data representation while maintaining the same level of details and preciseness.

4.2 Real-World Data

In this section, we first do an experiment on the ATT_FACE database. There are 40 persons in the database and each person has 10 images differing in lighting conditions, facial expressions and details. The original image (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 $HSIZE = [4; 4]$, $\sigma = 2$.

The experiments are conducted in the "closed" and "open-ended" environment for oKDE, ASOINN, TopoART and ILDN. In the closed environment, patterns are randomly selected from the whole learning set. In the open-ended 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, etc. For BatchMatrix and localPCASOM are not incremental method, we only conduct these two methods in the closed environment. Such methods need a predefined node number, to guarantee a good learning result, we set it as 200 which gives it a good initial condition: the initial nodes contain images of all 40 persons. We set the parameters of localPCASOM as $N = 200$, $\epsilon = 0.01$; BatchMatrix as $N = 200$, $epoch = 200$; oKDE as $D_{th} = 0.01$, $N = 10$, $f = 1$ for the closed environment and $f = 0.99$ for the open-ended environment; the parameters of ASOINN is set as λ=100, age_{max}=50, c=0.25; TopoART as β_{sbm}=0.6, ρ=0.96, φ=3, τ=100; ILDN as $\chi^2_{(d,q)}=\chi^2_{(644,0.90)}$, σ=1e-3, k=0.01, λ=1000.

Table 1. Mean node number, missing person number and accuracy of 100 times learning results in closed and open-ended environment for ATT_FACE. The best performance is bolded.

	Environment	localPCASOM	BatchMatrix	oKDE	ASOINN	TopoART	ILDN
Node number	closed	200*	200*	**3**	276.24	16.65	247.29
	open-ended	—	—	**3**	317.33	260.82	247.31
Missing persons	closed	—	—	38	**0**	26.05	**0**
	open-ended	—	—	38	5.01	**0**	**0**
Accuracy	closed	23.4%	24.1%	—	96.7%	48.25%	**98.5%**
	open-ended	—	—	—	89.6%	96.3%	**98.5%**

We adopt 3 factors to evaluate the learning results: node number, missing person number and recognition ratio. Table 1 gives the learning results of 100 times learning results in both environments. oKDE only gets 3 nodes in the learning result in both environments, losing 38 persons. Though ILDN gets larger node number in the closed environment than TopoART, it does not lose any person. In the open-ended environment, ILDN gets the smallest node number (excluding oKDE) and does not lose any person. On the other hand, the ASOINN "forget" 5.01 persons. Moreover, the proposed ILDN has the least difference of number of node and do not "forget" any person in the two environments, which means ILDN is a more stable incremental method than others. At last, nearest neighbor is used to classify vectors in the original image vector. We do not

Table 2. Learning results on some UCI datasets. N/A represents that the methods do not give the learning result within 10 days. The best performance on each data set is bolded.

Dataset	localPCASOM	BatchMatrix	oKDE	ASOINN	TopoART	ILDN
Segment	465*	465*	**24**	376	110	465
	10.00%	15.16%	73.55%	85.01%	37.10%	**90.32%**
Shuttle	9*	9*	30	69	63	**9**
	79.16%	81.37%	90.66%	90.39%	86.48%	**92.96%**
Webspam	745*	745*	N/A	3531	4643	**745**
	65.78%	67.25%		85.45%	83.05%	**86.50%**
KDD99	N/A	N/A	N/A	127	149	**32**
				92.15%	92.10%	**92.81%**

test the oKDE because it is unsuitable for this task. Table 1 shows that the recognition accuracy of ILDN is higher than the other methods.

Next, we do the experiments on some UCI datasets including Segment, Shuttle, Webspam and KDD99 which differ in the length, dimensionality as well as the number of classes. The parameters of the comparison methods are set as they suggest. The parameters of ILDN are set as σ=1e-3, k=0.01, λ=1000. We define the node number of localPCASOM and BatchMatrix as same as ILDN, marked with $*$ in the learning result.

The learning results are shown in Table 2. ILDN obtains highest accuracy in all datasets. Compared with three matrix learning methods localPCASOM, BatchMatrix and oKDE, ILDN uses far less nodes in three out of four datasets. Moreover, ILDN can handle very large scale dataset like KDD99, while localPCASOM, BatchMatrix and oKDE cannot give a learning result within 10 days. We can also find that the incremental (or online) matrix learning method oKDE cannot give a learning result on the relatively small data set like Webspam within 10 days. Thus, ILDN is a more practical incremental (or online) matrix learning method than others.

5 Conclusion

This paper presents an incremental local distribution learning network (ILDN). It combines the advantages of matrix learning and incremental learning. The covariant matrix, statistical vigilance parameter and merging mechanism enable it to obtain precise and concise representation of the data and learn incrementally. Moreover, the denoising processing based on data density makes it robust to noise. The experiments on both artificial datasets and real-world datasets validate our claims and show that ILDN is more effective over other matrix learning methods and incremental networks, obtaining higher accuracy and being able to handle large-scale data.

Acknowledgments. We thank the anonymous reviewers for their time and effort. This work is supported in part by the National Science Foundation of China under Grant Nos. (61375064, 61373001) and Jiangsu NSF grant (BK2013-1279).

References

1. Lloyd, S.P.: Least squares quantization in PCM. IEEE Transactions on Information Theory **28**(2), 129–137 (1982)
2. Martinetz, T.M., Berkovich, S.G., Schulten, K.J.: Neural gas network for vector quantization and its application to timeseries prediction. IEEE Transactions on Neural Networks **4**(4), 556–558 (1993)
3. Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological Cybernetics **43**(1), 59–69 (1982)
4. Martinetz, T., Schulten, K.: Topology representing networks. Neural Networks **7**(3), 507–552 (1994)
5. López-Rubio, E., Muñoz-Pérez, J., Gómez-Ruiz, J.A.: A pricipal components analysis self-organizing map. Neural Networks **17**, 261–270 (2004)
6. Yin, H., Allinson, N.M.: Self-organizing mixture networks for probability density estimation. IEEE Transactions on Neural Networks **12**(2), 405–411 (2001)
7. Huang, D., Yi, Z., Pu, X.: A new local PCA-SOM algorithm. Neurocomputing **71**, 3544–3552 (2008)
8. Arnonkijpanich, B., Hasenfuss, A., Hammer, B.: Local matrix adaptation in topographic neural maps. Neurocomputing **74**, 522–539 (2011)
9. López-Rubio, E.: Probabilistic self-organizing maps for continuous data. IEEE Transactions on Neural Networks **21**(10), 1543–1554 (2010)
10. Carpenter, G.A., Grossberg, S.: The ART of adaptive pattern recognition by self-organising neural network. IEEE Computer **21**(3), 77–88 (1988)
11. Fritzke, B.: A growing neural gas network learns topologies. In: Proceedings of the 1995 Advances in Neural Information Processing Systems, pp. 625–632 (1995)
12. Alahakoon, D., Halgamuge, S.K., Srinivasan, B.: Dynamic self-organizing maps with controlled growth for knowledge discovery. IEEE Transactions on Neural Networks **11**(3), 601–614 (2000)
13. Marsland, S., Shapiro, J., Nehmzow, U.: A self-organising network that grows when required. Neural Networks **15**(8–9), 1041–1058 (2002)
14. Shen, F., Hasegawa, O.: A fast nearest neighbor classifier based on self-organizing incremental neural network. Neural Networks **21**, 1537–1547 (2008)
15. Tscherepanow, M., Kortkamp, M., Kammer, M.: A hierarchical ART network for the stable incremental learning of topological structures and associations from noisy data. Neural Networks **24**(8), 906–916 (2011)
16. Araujo, A.F.R., Rego, R.L.M.E.: Self-organizing maps with a time-varying structure. ACM Computing Surveys **46**(1) Article No. 7 (2013)
17. Kristan, M., Leonardis, A., Skočaj, D.: Multivariate online kernel density estimation with Gaussian kernels. Pattern Recognition **44**(10–11), 2630–2642 (2011)