A Perception Evolution Network for Unsupervised Fast Incremental Learning

Youlu Xing, Furao Shen* and Jinxi Zhao

***Abstract*— A perception evolution network (PEN) is proposed for unsupervised fast incremental (or on-line) learning in this paper. The network has two layers: the perception layer receives the external data from the reality environment and the knowledge layer learns and records the knowledge contained in the data from the perception layer in incremental and self-organizing way. In PEN model, new input channels can be added to the perception layer freely during learning. When the perception layer gets some new input channels, the network will create corresponding data transmission channels to the knowledge layer. The prior learned knowledge stored in the knowledge layer will be expanded to a higher-dimensional space which contains the attributes of the new input channels. For incremental (or on-line) learning, PEN can automatically obtain suitable prototypes and find the topology structure of the learning data without any priori knowledge. The noise processing mechanism guarantees PEN can work in the complex real-world (noisy) environment. The experiments for both artificial data set and real-world data set show that the proposed method is effective.**

I. Introduction

In machine learning, unsupervised learning refers to the problem of finding hidden knowledge in unlabeled data.

Many scholars have studied in this area and proposed a number of algorithms. The k-means [1] and Neural Gas [5] aim to use some given prototype nodes represent the distribution of the learning data with low quantization error and get different clusters in the learning data. Self-Organizing Map [2] and Topology Representing Networks [6] can learn the topological structure of the learning data. These methods are equivalent to the perception-knowledge layer limited neural network model as Fig. 1 shows. The number of the nodes in the knowledge layer is fixed which leads to the model facing a dilemma: fixed number of nodes can only represent limited knowledge, when knowledge layer reaches saturation, the model will not be able to learn new knowledge while keeping the prior learned knowledge. And that is what happens in real world application: data usually become available gradually which may contain new knowledge. This fact requires learning system have the capability to learn knowledge incrementally.

In order to solve this dilemma, many "growing neural networks" or "incremental networks" are designed. Growing Cell Structure (GCS) [7] and Growing Neural Gas (GNG) [8] insert new node(s) every λ learning iterations, where λ is a constant. This strategy also has shortcoming: during each iteration, input pattern is forced to merge with a node no matter how big the gap between them. However, considering the physical meaning, it is unreasonable to merge two patterns with significant difference.

Youlu Xing(youluxing@sina.com), Furao Shen(frshen@nju.edu.cn) and Jinxi Zhao(jxzhao@nju.edu.cn) are with the National Key Laboratory for Novel Software Technology, and Department of Computer Science and Technology at Nanjing University, Nanjing, 210046, P.R.China.

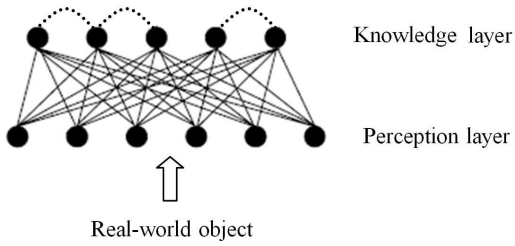


Fig. 1. Abstract architecture of Perception-Knowledge Layer Limited Neural Networks. The perception layer receives the external data. The knowledge layer learns and records the knowledge contained in the data. • represents the neural node. Dotted line in the knowledge layer represents the topology relationship between neural nodes. Topology relationship is not defined in some algorithms such as k-means.

Some neural networks which process input pattern in real-time can handle this problem. The Adaptive Resonance Theory network (ART) [3], Fuzzy ART [4] and TopoART [9] will 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 degree of matching is controlled by a parameter known as the vigilance parameter. The Self-Organizing Incremental Neural Network (SOINN) [10] and Enhanced SOINN (ESOINN) [11] decide whether to create a new node for the input pattern according to the node distribution around the local region of the input pattern.

This kind of methods are equivalent to the knowledge layer open-ended neural network model in Fig. 2. The number of the nodes in the knowledge layer dynamicly changes during learning. Compared to the model in Fig. 1, this model is much more suitable for real-world environment: it can add new node(s) for storing new knowledge. The knowledge layer is open-ended which makes learning model can accept new knowledge constantly.

The two models in Fig.1 and Fig. 2 are applied to various domains in past years and make a lot of great achievements, such as pattern recognition [13], computer vision [14], reasoning system [12] and so on. However, these two models also have limitation: the perception layer of the network is fixed which makes the model unable to receive new perception channels. We want to establish a learning model of which the perception layer is open-ended for new perception

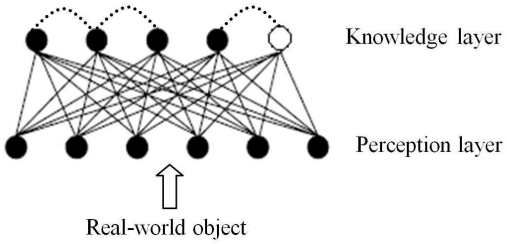


Fig. 2. Abstract architecture of Knowledge Layer Open-Ended Neural Networks. ○ represents the neural nodes added during learning.

channels, just like in the process of evolution from lower organisms to higher organisms, new ways of perception are generated by the evolution of the sense organs which leads to more and more complex perceptual system, therefore, more and more information arrive the brain of the biological which allows the biological to a better understanding of the real-world. Due to this inspiration, we design a new neural network model which is called Perception Evolution Networks (PEN) as Fig. 3 shows: The perception layer can add new input channels during learning and the number of nodes in the knowledge layer dynamically changes for new knowledge. In this model, new added input channels can receive new attribute data from the real-world environment. For example, a robot with a camera as the input device can only "see" objects, if we install a microphone on this robot, it can "hear" voice while "see" objects. The robot has one more way to perceive the real-world.

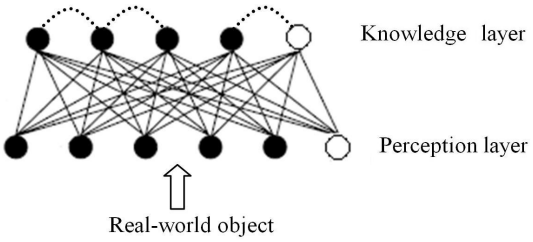


Fig. 3. Architecture of Perception Evolution Networks (PEN): Perception-Knowledge Layer Open-Ended Neural Networks Model.

We develop the following 3 design goals for our model:

Design goal 1: The input channels of the neural network can be extended freely.

Design goal 2: Fast incremental (or on-line) learning in complex real-world (noisy) environment in unsupervised way.

Design goal 3: Automatically obtain suitable prototypes for the learning data and find the topology structure of the learning data.

II. Perception Evolution Networks

Fig. 4 presents the flowchart of the proposed method which consists of the following 4 parts:

Node Competition: find the similarity between the input pattern and the nodes in the system.

System Updating: adjust system according to the result of the Node Competition step.

Noise Processing: discover and remove the noise nodes.

Classify: classify nodes to different clusters.

We assume that the input channels I of the network is established and added as the way of formula (1) shows:

$$\mathcal{I} = \overbrace{[\underbrace{I_{(C_1,1)},...,I_{(C_1,m_1)}}_{C_1}..... \underbrace{I_{(C_l,1)},...,I_{(C_l,m_l)}}_{C_l}]}^{l\ channels} \quad (1)$$

where channel C_k receives m_k-dimensional input vector $[I_{(C_k,1)},...,I_{(C_k,m_k)}]$. We define $S_k = [C_1, C_2, ..., C_k]$ as the k^{th} space, then S_k is $m_1 + m_2 + ... + m_k$ dimensional space. W_i as the feature vector of node i in the knowledge layer. If W_i is generated from channels C_1, C_2,...,C_k, then node i belongs to the S_k space. $W_i^{C_k}$ as the part of W_i which attached to channel C_k. T_i^k and V_i^k as the similarity threshold and the counter of winning-times of node i in the k^{th} space. All the spaces which each node in the knowledge layer belongs to are recorded in set R in ascending order.

The entire work flow of PEN is as follows: Knowledge layer is empty at the beginning. Learning patterns are fed into the network sequentially, PEN will create two nodes for the first two input patterns. For the input pattern latter, PEN first conduct node competition and find the winners and runner-ups in all spaces in R, then system updating is executed: pattern learning and self-organization between prototypes are conducted according to the result of the prototype competition step, meanwhile, the similarity threshold of the winners and runner-ups is updated. When all steps above are done, PEN will process next input pattern. Noise processing is conducted every λ patterns are learned. When users want to know the learning result, PEN will report the learned prototypes and the topology structure between them.

A. Node Competition

For unsupervised learning task, when an input pattern ξ comes, we first find the similarity between ξ and the nodes in the system. In PEN, we choose Euclidean distance as the similarity measurement. Because the dimension of the input pattern and the nodes may be different after adding some new input channels, we define $\|x - y\|_S$ to measure the distance between different dimensional vectors x and y in the S space, where assume S is d-dimensional space:

$$\|x - y\|_S = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2} \quad (2)$$

where assume x is a a-dimensional vector $[x_1, x_2, ..., x_a]$, y is a b-dimensional vector $[y_1, y_2, ..., y_b]$ and $d \leq min\{a, b\}$.

Suppose input pattern ξ is generated from channels $C_1, C_2, ..., C_l$. Then we get the winners $\boldsymbol{g}$ and runner-ups $\boldsymbol{s}$ in all spaces in R as:

$$g_j = \text{argmin}_{i \in N}\|\xi - W_i\|_{S_j} \quad (3)$$

$$s_j = \text{argmin}_{i \in N \setminus g_j}\|\xi - W_i\|_{S_j} \quad (4)$$

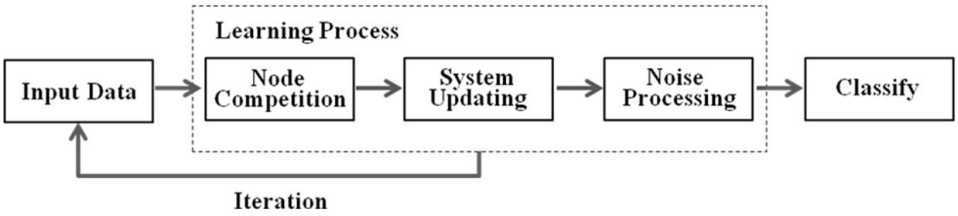


Fig. 4. Flow chart of the Perception Evolution Network

where N is the node set in the knowledge layer. $S_j \in R$. Notice that winner g_j and runner-up s_j may not belong to space S_j, they may belong to some space higher than S_j.

B. System updating

1) Node updating: To guarantee the continuity of the learning behavior after adding some new input channels, PEN will update the knowledge layer based on the prior learned knowledge: the existed knowledge will be expanded to a higher space which contains the attributes of the new input channels. The main idea is very simple: decide whether to expand a winner node in $\boldsymbol{g}$ according to the similarity between ξ and this winner node.

The node updating process begins at the lowest space in R and ends with some winner node is expanded or a new node is created. For the $S_j (j < l)$ space, if node g_j satisfied $\|\xi - W_{g_j}\|_{S_j} \leq T^j_{g_j}$ (i.e. winner node g_j is very "similar" to ξ in the S_j space) and g_j belongs to space S_j which is lower than ξ's space, the feature vector W_{g_j} will be expanded as:

$$W^{C_k}_{g_j} = \begin{cases} W^{C_k}_{g_j} + (1/V^k_{g_j})(\xi^{C_k} - W^{C_k}_{g_j}) & 1 \leq k \leq j \\ \xi^{C_k} & j < k \leq l \end{cases} \quad (5)$$

before formula (5), $V^k_{g_j}$ will be added by 1 when $k \leq j$ and initialized to 1 when $j < k \leq l$. $T^k_{g_j}$ will be initialized to positive infinity where $j < k \leq l$. In formula (5), we can see the learning rate is inversely proportional to the counter V.

If $\|\xi - W_{g_j}\|_{S_j} > T^j_{g_j}$, a new node will be created for ξ as formula (6):

$$W_{new} = \xi, T^k_{new} = +\infty, V^k_{new} = 1 \qquad 1 \leq k \leq l \quad (6)$$

If no expanding or creating operation occurs in the S_j space, the same process will be executed in the next space in R.

If no node in lower-dimensional space than space of ξ satisfies expanding or creating conditions, PEN will handle this situation in the space of ξ. If $\|\xi - W_{g_l}\|_{S_l} > T^l_{g_l}$ or $\|\xi - W_{s_l}\|_{S_l} > T^l_{s_l}$ is satisfied, a new node will be created using (6), else node g_l will be updated as formula (7):

$$W^{C_k}_{g_l} = W^{C_k}_{g_l} + (1/V^k_{g_l})(\xi^{C_k} - W^{C_k}_{g_l}) \qquad 1 \leq k \leq l \quad (7)$$

before formula (7), $V_{g_l} = V_{g_l} + \mathbf{1}$.

2) Self-organization: Besides node updating, we need do another important task: maintain the topology structure of the nodes. In PEN, we take each cluster as a manifold and use Voronoi polygons to represent its structure [6]. Self-organization process among nodes is conducted through the connection establishment, strengthen, weaken and elimination between nodes.

Connection establishment and strengthen only occur in the space of ξ. The connection establishment and strengthen condition is $\|\xi - W_{g_l}\|_{S_l} \leq T^l_{g_l} \&\& \|\xi - W_{s_l}\|_{S_l} \leq T^l_{s_l}$, the formula implies that there is overlap between the areas under the jurisdiction of g_l and s_l, the pattern falling in the overlap region will excite the relationship between g_l and s_l. If there is no connection between g_l and s_l, establish a connection and set the outdate degree $Age_{(g_l,s_l)}$ of this connection to 0 to represent the connection is most recent. If there is already a connection between g_l and s_l, set $Age_{(g_l,s_l)}$ to 0 to strengthen this connection.

For the reason of the dynamically changing environment in incremental (or on-line) learning, the nodes change their locations slowly during learning and this situation may lead nodes that are connected to each other at an early stage might not be near at a more advanced stage, we need to remove such connections. According to the situation above, the movement (feature vector expanding or updating) of a node i will lead to the outdate degree of connections from this node increase as formula (8):

$$Age_{(i,j)} = Age_{(i,j)} + 1 \qquad (j \in B_i) \quad (8)$$

where neighbor node set B_i is the set of nodes connected to node i. If connections with outdate degree Age greater than a predefine threshold Age_{max}, they will be removed.

3) Similarity threshold updating: Similarity threshold T is used to make the decision whether to add a new node or to move a node when an input pattern comes. Fixed threshold (such as [3]) is obvious not suitable for the learning with less or without prior knowledge. In PEN, the threshold T is an adaptive process with the learning process. Threshold T_i (where $i \in \mathbf{g}, \mathbf{s}$) is decided by the node distribution around node i: If node i has neighbor nodes, the similarity threshold T_i is calculated using the maximum distance between node i and its neighbor nodes, assume node i belongs to the S_j space:

$$T^k_i = \max_{c \in B_i} \|W_i - W_c\|_{S_k} \quad 1 \leq k \leq j \quad (9)$$

If node i has no neighbor nodes, the similarity threshold T_i is calculated using the minimum distance between node i and all the other nodes:

$$T_i^k = \min_{c \in N/i} \|W_i - W_c\|_{S_k} \quad 1 \leq k \leq j \tag{10}$$

(9) and (10) are calculated when they satisfy the definition of $\| * \|_{S_k}$. Threshold of new added node is initialized to $+\infty$.

C. Noise processing

Denoising process will be conducted every λ patterns are learned. We define noise as the random distribution low-density data over the whole learning set. Due to the noise from new added input channels, PEN may mistakenly expand a node to higher space. We calculate the mean value of the winning-times in the k^{th} space which is recorded as $DensityMean_k$, and the number of the neighbor of each node i in the k^{th} space (notice that nodes which are neighbors in high-dimensional space are also neighbors in low-dimensional space) which is recorded as $Neighbor_i^k$ to help make judgment.

For node i (assume that node i belongs to space S_k at present), if $Neighbor_i^k = 0$ and V_i^k is less than $DensityMean_k$ (i.e. node i is isolated and low-density in the S_k space), or if $Neighbor_i^k = 1$ and V_i^k is less than $c * DensityMean_k$, we consider node i dose not belong to space S_k. ($c \in [0, 1]$ is a predefine threshold to control the remove behavior). Then we check node i in S_{k-1} space until the conditions are not met, we find the space which node i belongs to (assume it is the $S_{k'}$ space) and update node i as:

$$W_i = W_i^{D_1,...,D_{k'}}, V_i = [V_{i1}, ..., V_{ik'}], T_i = [T_{i1}, ..., T_{ik'}] \tag{11}$$

formula (11) is actually a vector truncation operation. If node i does not belong to any space, it will be removed directly.

D. Classify

[5] prove a theorem that the competitive Hebbian rule forms perfectly topology preserving map of a manifold under the assumption that the number of nodes is sufficient for obtaining a dense distribution. Based on their opinion, we take each cluster as a manifold, therefore we can find different connected domains as different clusters [10] when the learning process is finished.

III. Analysis

A. move a node or add a new node?

As shown in Fig. 5, learning process of prototype-based clustering can be understood as a process of nodes fitting the data distribution. Assume there are N nodes in the system and the feature vector of node i denoted by W_i, and the fitting error $Error$ can be represented as follow:

$$Error = \int P(u) \min_{1 \leq i \leq N} \|u - W_i\| du \tag{12}$$

Learning process is to make (12) as small as possible or minimize it. The mainstream of current method to reduce the fitting error is based on Hebbian competitive learning:

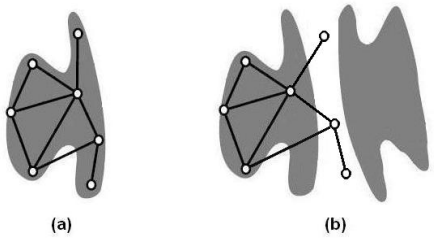


Fig. 5. (a) Distribution of nodes after a period of learning on $P(u)$. (b) A new data distribution $Q(v)$ is introduced to the learning system, and some nodes are moving toward to the area of $Q(v)$ during learning.

make the winner node (and nodes around the winner) move to the input pattern.

Now, we assume that a new data distribution $Q(v)$ arrives as Fig. 5(b). The winner node (and nodes around the winner) will move toward to the area of $Q(v)$. The learning process present is actually under the following fitting error formula:

$$Error = \int Q(v) \min_{1 \leq i \leq N} \|v - W_i\| dv \tag{13}$$

In (13), we can see that the fitting error formula ignores the original data distribution $P(u)$. After a period of learning, a large fitting error may occur on the area of $P(u)$. Another problem during learning is: the previously learned knowledge is destroyed and the new knowledge is not represented correctly, some information stored in the system is meaningless just like the 3 nodes in the intermediate position of distribution $P(u)$ and $Q(v)$.

For the problem above, more effective approach is to create new nodes for distribution $Q(v)$. However, in unsupervised learning we have no label information of the data, so that we have no idea about when a new distribution comes. This brings a problem: when to add a new node or move a existed node? We make the decision according to the node distribution around the local region of the input pattern, and give each node a threshold (section II) to help make judgment.

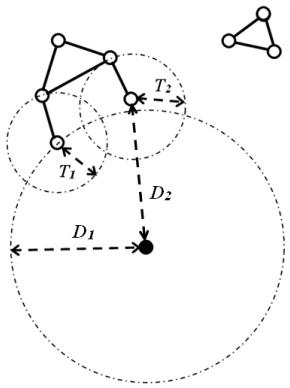


Fig. 6. T_1 represents the threshold of the winner. T_2 represents the threshold of the runner-up. The input pattern is D_1 away from the winner and D_2 away from the runner-up. Assume $D_1 > T_1$ and $D_2 > T_2$.

To illustrate our idea, we introduce an example as Fig. 6

shows. In this example we get 3 useful information about the node distribution:

Information 1. Some nodes exist T_1 away from the winner and T_2 away from the runner-up.

Information 2. The input pattern is D_1 away from the winner and D_2 away from the runner-up. $D_1 > T_1$ and $D_2 > T_2$.

Information 3. No node exists inside the round centered at the input pattern with radius D_1.

According to the 3 information above, we can get following *conclusion*: there is *no sufficient reason to reject* to create a new node for this input pattern.

This learning process adding new nodes to fit the new data distribution while retaining the original fitting result will get a better fitting result than the learning process under (13).

B. why use $||$ not $\&\&$?

In section III A, we give an example to illustrate when to add a new node. In that example we use condition

$$T_1 < D_1 \ \&\& \ T_2 < D_2 \tag{14}$$

However, in PEN, we relax this constraint to

$$T_1 < D_1 \ || \ T_2 < D_2 \tag{15}$$

The reason can be explained by the example as Fig. 7 shows. In this example, we assume that:

Assumption 1. The distance between $P(u)$ and $Q(v)$ is far enough to guarantee that the furthest distance inside $P(u)$ and $Q(v)$ is less than the nearest distance between $P(u)$ and $Q(v)$.

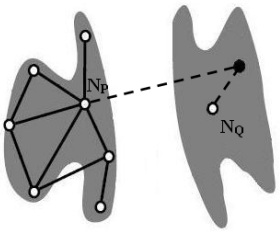


Fig. 7. After period of $P(u)$, a new data distribution $Q(v)$ is introduced to the learning system, and a new node is created for the first input pattern of $Q(v)$.

When the first pattern of distribution $Q(v)$ arrives, the learning system creates a node N_Q for this pattern. For the further input pattern ξ from $Q(v)$ (solid black point in Fig. 7), we find the winner node N_Q and runner-up node N_P. According to *Assumption 1*, we can get:

$$T_{N_Q} > \|\xi - N_Q\| \tag{16}$$

is true for all patterns from $Q(v)$. This will lead to condition (14) never be satisfied, therefore no more node will be created for $Q(v)$. Then the topological structure of some distribution such as ring and sinusoidal distribution can not be learned. However,

$$T_{N_P} < \|\xi - N_P\| \tag{17}$$

may be satisfied then judging condition (15) is satisfied. So when use (15) , new nodes for the following input pattern can be created and organized to represent the topological structure of $Q(v)$.

IV. Experiment

A. Artificial data set

We conducted this experiment on the artificial data set shown in Fig. 8. The learning set is separated into 5 parts: Set A satisfies 3-D sinusoidal distribution. Set B and C satisfies 3-D ring distribution. Set D and E satisfy the same 2-D Gaussian distribution in the X-Y space but different 3-D Gaussian distribution in the X-Y-Z space. D and E explain an actual phenomenon: in the early period, we have few perception channels that the information we gained is only enough to make a rough classification. However, with the deepening of cognition, we get more information through some new perception channels so that we can make a more elaborate classification. D and E are designed to test whether PEN can be adapted to this situation. In each round of learning (λ patterns), we add 10% random noise. We set parameters of PEN as $\lambda = 50$, $Age_{max} = 25, c = 1$ in this experiment.

During experiment, we first give PEN two input channels X and Y to accept the X-Y space data, after 50000 patterns from A, B, C, D and E are learned (10000 patterns each set), we add a new input channel Z to accept the Z space data, then PEN can simultaneously accept data from channel X, Y and Z, we fed another 50000 patterns (10000 patterns each set) to PEN. We conduct the experiment in two environments: in the **stationary environment**, patterns are randomly selected from the whole learning set. In the **non-stationary environment** (for the Stability-Plasticity Dilemma [3]), from step 1 to 10000, patterns are chosen randomly from A. From step 10001 to 20000, the environment changes and patterns from B are chosen, etc. When channel Z is added, repeat the process above from step 60000 to 100000.

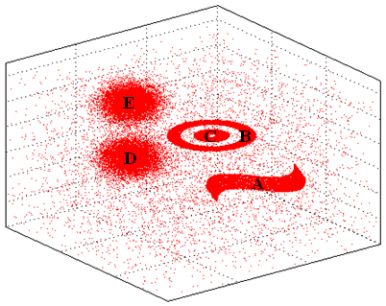
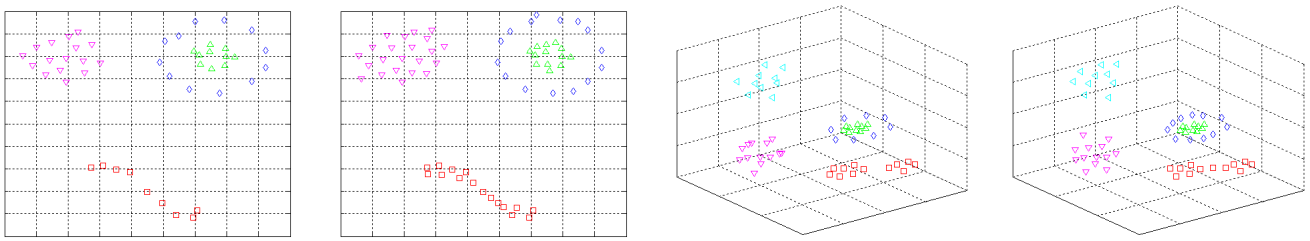


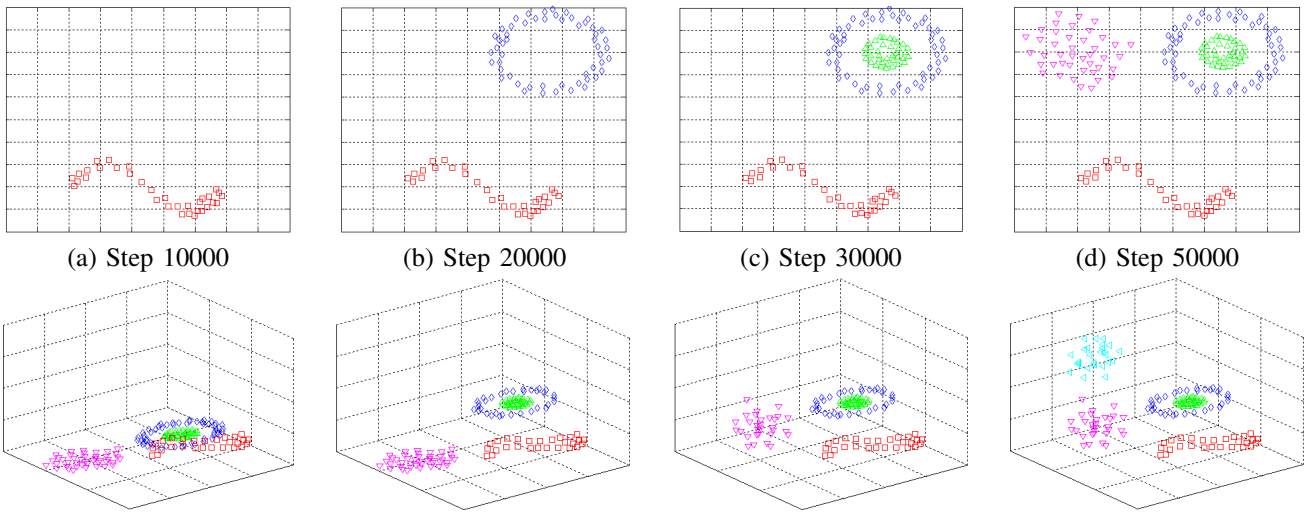
Fig. 8. Artificial data set used for the experiment.

1) Experiment in a stationary environment : The result is shown in Fig. 9. We can see that PEN fits the learning data well quickly at step 5000 (4 clusters in this space) and improves the fitting until the end of the X-Y space data. When channel Z is added, the nodes are expanded to the X-Y-Z space fast at step 55000, then PEN improve the fitting



(a) Step 5000 (b) Step 50000 (c) Step 55000 (b) Step 100000

Fig. 9. A result in a stationary environment. 4 clusters in X-Y space and 5 clusters in X-Y-Z space. The noise is effectively removed.



(a) Step 10000 (b) Step 20000 (c) Step 30000 (d) Step 50000

(e) Step 60000 (f) Step 80000 (g) Step 90000 (h) Step 100000

Fig. 10. A result in a non-stationary environment. The noise is effectively removed.

until the end of the whole learning process. Data set D and E are separated to each other which is shown in Fig. 9 (c) and (d). The noise during learning is effectively removed.

2) Experiment in a non-stationary environment : The learning result in X-Y space is shown in Fig. 10 (a) to (d). We can see that PEN can learn new knowledge while keeping the prior learned knowledge well. After adding a new input channel which allows Z-axis data passing through, the perception layer can receive X-Y-Z axis data. The learning results are shown in Fig. 10 (e) to (h). During step 50001 to 60000, we can see that cluster □ is upgraded to the 3-D space, and other clusters keep unchanged. During step 60001 to 80000, cluster ♢ and △ are upgraded to the 3-D space, and other clusters keep unchanged including the 2-D space cluster and 3-D space cluster. Then cluster ▽ is upgraded to the 3-D space for the coming of data set D during step 80001 to 90000. During step 90001 to 10000, data set E arrives, PEN creates new nodes and organizes these nodes forming a new cluster. Finally, the proposed method gets 5 clusters in the X-Y-Z space. The noise during learning is effectively removed.

Fig. 11 shows how the number of nodes changes during learning. The number of nodes increases when input patterns come from a new distribution and the nodes fit the new data distribution very fast. After input channel Z is added, the node in the knowledge layer is expanded to the X-Y-Z space and the number of nodes keeps stable before data set E comes. When learning patterns of E arrive, PEN creates new nodes for E very quickly. Due to the noise, a small fluctuation exists in the number of nodes.

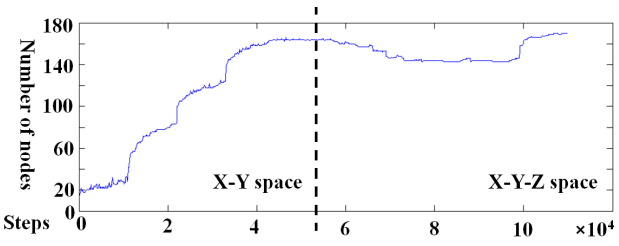


Fig. 11. Number of nodes during on-line learning in a non-stationary environment.

In this section, we can see that the experimental results is as good as SOINN [10], ESOINN [11] and TopoART [9] in the X-Y space, moreover, our method can add new input channels during learning which SOINN, ESOINN and TopoART can not do.

B. Real-world data

AR face image database [15] is used in this experiment. 10 persons' photos, 6 different photos per person, are selected

for learning as Fig. 12 shows. Feature vectors are taken as follows: first, the original image with size 120*165 is re-sampled to 24*33 image using the nearest neighbor interpolation method. Then Gaussian smoothing is used to smooth the 24*33 image with Gaussian width=4, s=2. Each image is divided into three parts of the R, G and B to simulate 3 input channels. We set parameters of PEN as $\lambda = 100$, $Age_{max} = 50, c = 0.5$. A smaller c is used to retain more prototypes.

Fig. 12. Original facial image and feature vectors.

During experiment, we first give PEN only one input channel R to accept the R part of the images, after 2000 photos of 10 persons (200 per person) are learned we add another input channel G to accept the G part data, then PEN can simultaneously accept R-G data and learn another 2000 photos of 10 persons. At last input channel B is added which makes PEN can accept R-G-B data, then learn yet another 2000 photos of 10 persons. We conduct the experiment in both **stationary environment** and **non-stationary environment** (same as artificial data set experiment). Fig. 13 gives a result in a stationary environment, we can see that PEN gets a correct clustering result with reasonable prototypes and expands the prior learned prototypes very well when channel G and B arrive.

To test the stability of PEN, we performed 100 times learning in both stationary and non-stationary environment. We recorded the frequency of the number of clusters. The result is shown in Fig. 14: with the adding of the input channels, the distribution of the clustering results become more compact and the number of the correct clustering become larger. For PEN is an unsupervised incremental (or on-line) learning model, we also record the prototypes and number of persons in learning result in Table 1. The term "Person" in "Prototype/Person" means the mean value of the number of different persons in the 100 experiment results. "Prototype" means the mean value of the number of the learned prototypes. Table. 1 shows that the person lost rate is low and the prototype number is reasonable comparison with Fig. 12. We can also see that the number of prototypes in non-stationary environment is larger than in stationary environment, the reason is: in non-stationary environment, patterns usually come from one person during a long period, this makes the subtle difference between patterns of one person is learned by PEN. In stationary environment patterns are chosen randomly, the subtle difference between patterns of one person is neglected because of the significant difference between patterns of different persons.

TABLE I
PROTOTYPE/PERSON MEAN RESULT.60/10 IN LEARNING SET

environment	Prototype/Person (mean)		
	R-space	RG-space	RBG-space
stationary	25.14/9.80	38.57/9.85	40.86/9.89
non-stationary	45.34/9.67	49.68/9.74	51.01/9.83

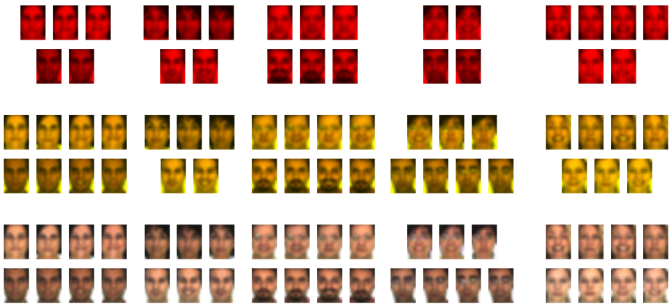
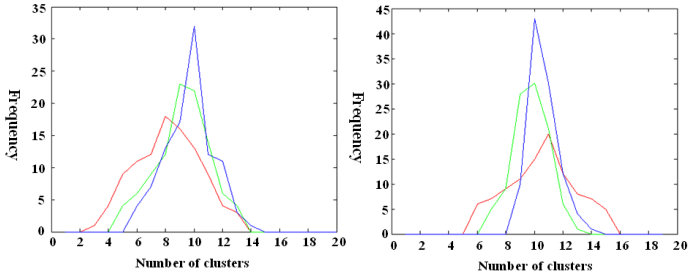


Fig. 13. A result in a stationary environment.



(a) Stationary environment (b) Non-stationary environment

Fig. 14. Distribution of the cluster result. The red, green and blue lines represent the learning result in R-space, RG-space and RGB-space.

In this section, our experimental results are as good as SOINN [10] and ESOINN [11], moreover, our method can add input channels during learning which SOINN and ESOINN can not do.

V. CONCLUSION

In this paper, we introduce the perception evolution network for unsupervised fast incremental learning. The input channel of the network can be extended freely. For the pattern which contains data of the new input channels, we compute the similarity between the input pattern and the nodes under the common attributes space, then decide whether to expand any node or establish a new node. The topology structure of the learning data is represented by Voronoi polygons. Nodes with low winning frequency and few topology neighbors will be trimmed or removed to reduce the influence of noise after

every λ learning iterations. The experiments for both artificial data set and real-world data show that the proposed method is effective.

ACKNOWLEDGEMENTS

This work was supported in part by the 973 Program 2010CB327903, the Fund of the National Natural Science Foundation of Jiangsu NSF grant BK2011567. And the Project of Graduate Education Innovation of Jiangsu Province CXLX12_0052.

REFERENCES

[1] S. P. Lloyd, "Least Squares Quantization in PCM," *IEEE TRANSACTIONS ON INFORMATIONT HEORY*, vol. 28(2), pp. 129-137, 1982.

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

[3] Gail A. Carpenter and Stephen Grossberg, "The ART of adaptive pattern recognition by self-organising neural network," *IEEE Computer*, vol. 21(3), pp. 77-88, 1988.

[4] Gail A. Carpenter and Stephen Grossberg and David B. Rosen, "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System," *Neural Networks*, vol. 4(6), pp. 759-771, 1991.

[5] Thomas M. Martinetz and Stanislav G. Berkovich and Klaus J. Schulten, "Neural gas network for vector quantization and its application to timeseries prediction," *IEEE Transactions on Neural Networks*, vol. 4(4), pp. 556-558, 1993.

[6] T. Martinetz and K. Schulten, "Topology representing networks," *Neural Networks*, vol. 7(3), pp. 507-552, 1994.

[7] B. Fritzke, "Growing cell structuresłA self-organizing network for unsupervised and supervised learning," *Neural Networks*, vol. 7(9), pp. 1449-1460, 1994.

[8] Betnd Fritzke, "A growing neural gas network learns topologies," *Proceedings of the 1995 Advances in Neural Information Processing Systems*, pp. 625-632, 1995.

[9] Marko Tscherepanow and Marco Kortkamp and Marc Kammer, "A hierarchical ART network for the stable incremental learning of topological structures and associations from noisy data," *Neural Networks*, vol. 24(8), pp. 906-916, 2011.

[10] Furao Shen and Osamu Hasegawa, "An incremental network for online unsupervised classification and topology learning," *Neural Networks*, vol. 19(1), pp. 90-106, 2006.

[11] Furao Shen and Tomotaka Ogurab and Osamu Hasegawa, "An enhanced self-organizing incremental neural network for online unsupervised learning," *Neural Networks*, vol. 20(8), pp. 893-903, 2007.

[12] Furao Shen and Akihito Sudo and Osamu Hasegawa, "An online incremental learning pattern-based reasoning system," *Neural Networks*, vol. 23(1), pp. 135-143, 2010.

[13] Enric Sesa Nogueras and Marcos Faundez Zanuy, "Biometric recognition using online uppercase handwritten text," *Pattern Recognition*, vol. 45(1), pp. 128-144, 2012.

[14] J. Liu and Y. Yang and I. Saleemi and M. Shah, "Learning semantic features for action recognition via diffusion maps," *Computer Vision and Image Understanding*, vol. 116(3), pp. 361-377, 2012.

[15] Aleix Martinez and Robert Benavente, "The ar face database," *CVC Technical Report*,1998.