Perception Evolution Network Based on Cognition Deepening Model—Adapting to the Emergence of New Sensory Receptor

Youlu Xing, Furao Shen, and Jinxi Zhao

***Abstract*— The proposed perception evolution network (PEN) is a biologically inspired neural network model for unsupervised learning and online incremental learning. It is able to automatically learn suitable prototypes from learning data in an incremental way, and it does not require the predefined prototype number or the predefined similarity threshold. Meanwhile, being more advanced than the existing unsupervised neural network model, PEN permits the emergence of a new dimension of perception in the perception field of the network. When a new dimension of perception is introduced, PEN is able to integrate the new dimensional sensory inputs with the learned prototypes, i.e., the prototypes are mapped to a high-dimensional space, which consists of both the original dimension and the new dimension of the sensory inputs. In the experiment, artificial data and real-world data are used to test the proposed PEN, and the results show that PEN can work effectively.**

***Index Terms*— New dimension of perception, online incremental learning, perception evolution network (PEN), stability-plasticity dilemma, unsupervised learning.**

I. Introduction

BIOLOGICALLY inspired computing has spawned many classical and powerful algorithms, including perceptron [1], genetic algorithm [2], self-organizing map (SOM) [3], and deep learning [4].

In 2007, using genetic engineering, Jacobs *et al.* [5] inserted human L-pigment genes into female mice one X-chromosome and found that the heterozygous female mice, whose retinas contain both native mouse pigments (S and M pigments) and human L pigment, show enhanced long-wavelength sensitivity, and acquire a new capacity for chromatic discrimination. It means that the knock-in mice express the human gene in their cone cells and that the human L pigment transmits light signals with efficiency comparable with that of the mouse's native pigments. As a consequence, the knock-in mice are able to discriminate among green, yellow, orange, and red panels that, to ordinary mice, look exactly the same. From this paper, we can see that the brain of the organisms has an amazing adaptability and plasticity; it can use the newly introduced sensory receptor immediately. Such adaptability and plasticity of the brain will make the knock-in mice understand the world deeper than other mice.

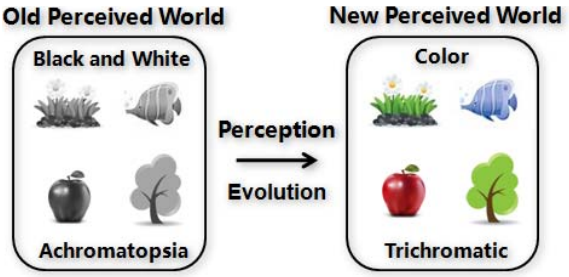

Fig. 1. Perception evolution permits the organisms to understand the real world more deeply. World in the eyes of the achromatopsia organisms and another world in the eyes of the trichromatic organisms.

The above experiment in [5] inspires us a very interesting and challenging problem: *can we exploit a computational model that is able to expand its cognitive dimension online freely?* If this is achieved, the agent with such computational model will be able to expand its sensing capability during its lifetime. This is different from the natural way that the offspring get a new cognitive dimension through genetic variation while their parents sacrifice. Here, the parental generation (which means the agent itself) is given a new cognitive dimension online *directly*. In addition, it will have broad applications such as in robot system [6], information fusion [7], and data stream mining [8]. For example, if we install new sensors to an agent to expand its sensing capability, with this model, we do not need to retrain the agent offline from scratch, the information gathered from the newly installed sensors is fused with the existed knowledge online automatically, in other words, the model is autonomous for the sensory dimension. However, to the best of our knowledge, there is no such computational model.

In the process of evolution from lower organisms to higher organisms, increasingly complex perceptual system is generated by the evolution of the sense organs; therefore, more information arrives at the brain of the organisms; as a consequence, the organisms are able to understand the real world more deeply, as shown in Fig. 1. In this paper, we propose a perception evolution network (PEN) to simulate

Manuscript received July 15, 2014; revised February 11, 2015; accepted March 17, 2015. Date of publication April 28, 2015; date of current version February 15, 2016. This work was supported in part by the National Science Foundation of China under Grant 61375064, Grant 61373001, and Grant 61375061, and in part by the Jiangsu Province National Science Foundation under Grant BK20131279. *(Corresponding author: Furao Shen.)*

The authors are with the National Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210046, China (e-mail: youluxing@sina.com; frshen@nju.edu.cn; jxzhao@nju.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2015.2416353

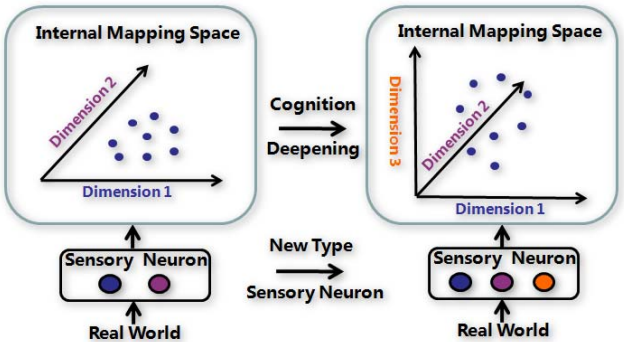

Fig. 2. Mathematical modeling of the PEN. With the coming of the new type of sensory neuron, the organisms have a new information channel to perceive the real world. As a consequence, the internal mapping space has a corresponding new dimension. Thus, the points in the low-dimensional feature space will be mapped to a higher dimensional feature space. We call it a cognition deepening process.

the evolution of the perceptual system. PEN permits the emergence of a new dimension of perception in the perception field of the network.

Fig. 2 gives the mathematical modeling of the PEN. In the beginning, the agent has two types of sensory neurons to perceive the real world; correspondingly, the external world is mapped to the agent's internal world, which is a 2-D space, while each dimension is attached to its corresponding type of sensory neurons. After the agent gets a new type of sensory neurons, i.e., the agent has a new information channel to perceive the real world, the patterns (or memories) stored in the agent should be integrated with the new dimensional sensory inputs, i.e., the patterns are mapped to a high-dimensional space, which consists of both the original dimension and the new dimension of the sensory inputs; we call it a cognition deepening process.

Meanwhile, the data in the real world are usually unlabeled; thus, PEN is designed as an unsupervised learning model to build the representations of the external data. On the other hand, the agent usually perceives the external world in an online way, and the agent is usually an open system, which is able to adapt to the changing or open-ended [9], [10] environment. In such environments, data with new knowledge is coming continuously. Under this condition, keeping learning new knowledge quickly without catastrophic forgetting of already learned, and still successful, memories (i.e., incremental learning [11]) is a very important ability for the agent [9]–[11], just as humans are able to learn new objects or words without forgetting the previously learned ones throughout their lifetimes. Thus, online incremental learning is a distinct property of the brain [12]. Therefore, online incremental learning is introduced to PEN.

As a summary, we give the neural network modeling of the PEN in Fig. 3. The prototypes and connections between prototypes in the prototype field of the network are created incrementally, i.e., new prototypes are created for some new patterns, which are dissimilar with the existing patterns recorded in the network. The perception field permits the emergence of new sensory neurons. After getting some new sensory neurons in the perception field, PEN automatically

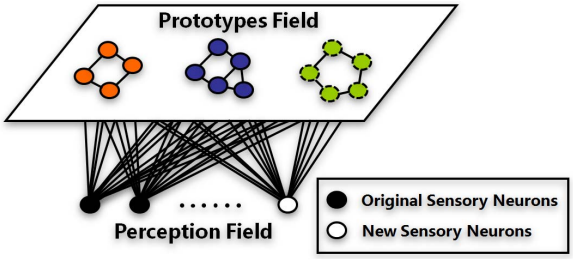

Fig. 3. Neural network modeling of the PEN. It is a kind of perception-prototype field open-ended neural network. The perception field permits the emergence of some new sensory neurons. The nodes with dashed edge in the prototype field are added for some new patterns during online learning.

integrates the new input signals (or information) received from these new sensory neurons with the existing knowledge (or prototypes) learned from the original sensory neurons. The perception field and the prototype field of PEN are open-ended; thus, we take PEN as a kind of perception-prototype field open-ended neural network model.

Briefly, the main targets of the PEN are as follows.

1) To permit the emergence of new sensory neurons in the perception field of the network, and integrate the new input signals with the prior learned knowledge.
2) To learn suitable prototypes from the data in an unsupervised and online incremental way, and not require predefined prototype number and similarity threshold.

II. Related Work

During the past 20 years, many scholars have studied unsupervised learning and incremental learning.

The SOM is one of the most famous unsupervised learning models; it provides a topology-preserving mapping from the high-dimensional space to map neurons. The neurons usually form a 2-D lattice; thus, the mapping is usually from a high-dimensional space onto a grid. There are many improved versions of the SOM. The parameterless SOM [13] eliminates the need for a learning rate and annealing schemes for learning rate and neighborhood size. The self-organizing potential field network [14] introduces competitive and cooperative behaviors to provide an ability to escape from the local optimum.

The topology representing networks (TRNs) [15] develop the SOM; it does not need any predefined topological structure; the topological structure is established gradually by the competitive Hebbian learning [16] and neural gas (NG) [17].

We call these methods perception-prototype field limited models; the abstract structure of such networks is shown in Fig. 4. The number of prototypes in the prototype field is fixed; it leads to the model facing a dilemma: a fixed number of prototypes can only represent limited knowledge. When the prototype field of the network reaches saturation, the model will not be able to learn new knowledge while keeping the prior learned knowledge. Unfortunately, in the real-world applications, data with new knowledge is coming gradually during the learning process [18]. In such a dynamic environment, we cannot know how many prototypes will be suitable for these applications. It is very difficult to predefine an appropriate number of prototypes for the perception-prototype field limited model.

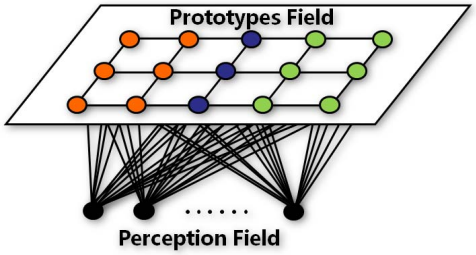

Fig. 4. Abstract structure of the perception-prototype field limited neural networks. The number of prototypes in the prototype field is fixed. Lines in the prototype field represent the topology relationship between prototypes. In some algorithms, topology relationship is not predefined but learned by the algorithm itself such as the TRN.

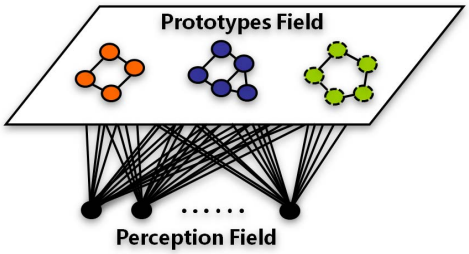

Fig. 5. Abstract structure of the prototype field open-ended neural networks. The prototypes with dashed edge in the prototype field are added or pruned during learning. Some algorithms do not define the topology relationship between the prototypes such as the ART network.

To solve this problem, many growing neural networks or incremental networks are designed. Growing SOM (GSOM) [19], growing cell structure [20], and growing NG (GNG) [21] insert new prototype(s) for every λ samples learned, where λ is a constant parameter. Life-long learning cell structure [22] introduces an insertion criterion to decide whether to insert a new prototype for every λ patterns learned; meanwhile, it also deletes prototypes to avoid overfitting. However, in these methods, during each λ period, the input sample is forced to merge with a prototype no matter how big the gap between them. Considering the physical meaning, it is unreasonable to merge two patterns with significant difference.

There is another problem, which is well known as the stability-plasticity dilemma [9], i.e., making the system quickly learn about new knowledge (e.g., new objects) without just as quickly being forced to forget previously learned, but still useful, memories. The adaptive resonance theory (ART) network [9], fuzzy ART [23], evolving SOMs [24], and TopoART [25] (ART family) will create a new prototype when no match occurs between the current input sample and the current category set. The degree of matching is controlled by a parameter known as the vigilance parameter. This strategy makes the network add new prototype, when the input sample is not similar to the existing prototypes that the network has learned. However, the vigilance parameter should be predefined. It is a difficult job when we have little prior knowledge about the learning task, especially for the unsupervised learning task. Evolving vector quantization [26] introduces an online split-and-merge strategy to overcome the poor setting of the vigilance parameter.

Self-organizing incremental neural network (SOINN) [27], enhanced-SOINN [28], adjusted-SOINN [29], and load-balancing-SOINN [30] decide whether to create a new prototype for the input sample according to the prototype-distribution around the local region of the input sample. These methods overcome the disadvantages of GSOM, GNG, and ART family algorithm. Incremental learning vector quantization [31], [32] introduces the idea of the adjusted-SOINN to the learning vector quantization and achieves very good results.

We summarize these unsupervised incremental learning methods as prototype field open-ended models, as shown in Fig. 5. This type of network focuses on the prototype field; it makes the prototype field open-ended for new categories by adding new prototypes. In recent years, these methods are applied to various domains, including reasoning system [33], pattern recognition [34], and computer vision [35].

However, the perception-prototype field limited and prototype field open-ended models are not able to expand the perception field, which we think is a very important ability for unsupervised learning as mentioned in Section I. For example, if we install new sensors to a robot as a new information channel, and we want the robot to use the new sensors effectively. The perception-prototype field limited and prototype field open-ended models cannot deal with such a task, and the proposed PEN is designed to solve such problems.

III. Perception Evolution Network

A. Problem Formulation

Assume that the original neural network $\mathcal{N}$ has n neurons in the perception field, which receive n-dimensional external data $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$. After a period of learning, some prototypes are created in the prototype field of PEN. Then, m new sensory neurons emerge in the perception field of PEN. In addition, the received data becomes $\mathbf{x} = (x_1, x_2, \ldots, x_n, x_{n+1}, \ldots, x_{n+m}) \in \mathbb{R}^{n+m}$. The learned prototypes will be mapped to a high-dimensional space, which contains the dimension of these m new sensory neurons. If the new sense brings some new distinguishable categories, PEN will create prototypes for such new categories.

The entire workflow of the PEN is as follows. The prototype field of PEN is empty in the beginning, and learning samples are fed into the network sequentially, i.e., in an online way. The PEN will create two prototypes using the first two input samples. For the latter input sample, PEN first conducts prototype competition, then prototype learning and prototype self-adaptive associating are conducted according to the result of the competition step; meanwhile, the similarity threshold of the activated prototypes will be updated. When all the steps are done, PEN will process the next input sample. Prototype pruning is conducted after every λ samples learned.

When some new sensory neurons are introduced, PEN will find some low-dimensional prototype to map to high-dimensional space for each input sample. Prototype self-adaptive associating and similarity threshold updating are conducted similarly as the procedure before

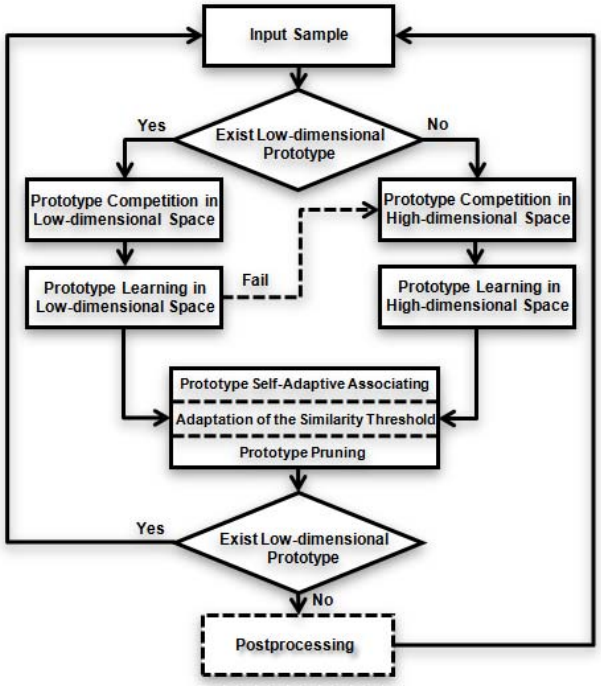

Fig. 6. Workflow of PEN.

new sensory neurons appear. We remove the prototypes that cannot be mapped to the high space after a long period of learning because they are potential distortion prototypes. Postprocessing is executed when all prototypes are mapped to the high-dimensional space. Fig. 6 gives the workflow of the PEN.

Next, we assume that the original neural network has n sensory neurons; after a period of learning, m new sensory neurons emerge. We denote the original low-dimensional space as $S_l = \mathbb{R}^n$, the new high-dimensional space as $S_h = \mathbb{R}^{n+m}$, the learned prototypes in the prototype field are stored in set P.

B. Prototype Competition

When an input sample $\mathbf{x} = (x_1, x_2, \ldots, x_n, \ldots, x_{n+m}) \in S_h$ comes, we first calculate the Euclidean distance between $\mathbf{x}$ and all prototypes in the set P. The dimension of each P_i may be different with $\mathbf{x}$ (because during learning, some P_i may be already mapped to space S_h and some may still stay in space S_l). For the prototype $P_i \in S_l$, we calculate the Euclidean distance between $\mathbf{x}$ and P_i using the first n-dimensional attributes, i.e., the attributes of space S_l. For the prototype $P_i \in S_h$, we use all $(n+m)$-dimensional attributes to calculate the Euclidean distance. Then, we find the winner prototype P_w^l in space S_l as

$$P_w^l = \underset{P_i \in P}{\operatorname{argmin}} \|\mathbf{x} - W_{P_i}\|_{S_l} \tag{1}$$

where W_{P_i} is the weight vector of P_i and $\|\cdot\|_{S_l}$ represents the Euclidean distance in space S_l. For the prototypes belonging to S_h, we use their first n-dimensional weights (i.e., the part of their weight vector in space S_l) to compete with the prototypes belonging to S_l. Therefore, P_w^l may belong to space S_l or S_h.

Then, we find the winner and runner-up prototypes P_w^h, P_r^h in space S_h as

$$P_w^h = \underset{P_i \in P^h}{\operatorname{argmin}} \|\mathbf{x} - W_{P_i}\|_{S_h} \tag{2}$$

$$P_r^h = \underset{P_i \in P^h \setminus P_w^h}{\operatorname{argmin}} \|\mathbf{x} - W_{P_i}\|_{S_h} \tag{3}$$

where P^h represents the set of prototypes belonging to space S_h, and $\|\cdot\|_{S_h}$ represents the Euclidean distance in space S_h. The prototypes belonging to S_l cannot be applied to $\|\cdot\|_{S_h}$. Therefore, the prototypes that belong to S_l do not take part in this competition. For convenience in notation, we rewrite P_w^l as P_c, P_w^h as P_a, and P_r^h as P_b.

Note: If all prototypes in P have been mapped to space S_h, we do not need to conduct the competition in space S_l, i.e., (1).

C. Prototype Learning

1) Learning in Low-Dimensional Space: The winner prototype P_c is the candidate prototype to be mapped to the high-dimensional space S_h. If

$$\mathbf{dim}(P_c) = n \tag{4}$$

i.e., the dimension of P_c is equal to n, $P_c \in S_l$. It is indeed a low-dimensional prototype. Then, we check the following condition:

$$\|\mathbf{x} - W_{P_c}\|_{S_l} \leq T_{P_c}^l \tag{5}$$

where $T_{P_c}^l$ represents the similarity threshold of P_c in space S_l. Equation (5) means that the distance between P_c and $\mathbf{x}$ is less than the similarity threshold $T_{P_c}^l$, i.e., sample $\mathbf{x}$ is very similar to P_c in space S_l. Then, P_c is activated, we add the accumulated times of activation of P_c in space S_l by 1, that is

$$M_{P_c}^l = M_{P_c}^l + 1. \tag{6}$$

Then, we map P_c to space S_h as

$$W^l = W_{P_c} + (1/M_{P_c}^l)(\mathbf{x}^l - W_{P_c}) \tag{7}$$

$$W^{h-l} = \mathbf{x}^{h-l} \tag{8}$$

$$W_{P_c} = (W^l, W^{h-l}) \tag{9}$$

where $\mathbf{x}^l$ represents the attributes of space S_l, i.e., $\mathbf{x}^l = (x_1, x_2, \ldots, x_n)$. $\mathbf{x}^{h-l}$ represents the attributes of the new perception neurons, i.e., $\mathbf{x}^{h-l} = (x_{n+1}, x_{n+2}, \ldots, x_{n+m})$. According to (7)–(9), the weights of P_c learned from original perception neurons are moved toward $\mathbf{x}$, and the weights learned from new perception neurons are added in the tail of W_{P_c}. Meanwhile, the threshold $T_{P_c}^h$ (the similarity threshold of P_c in space S_h) is initialized to $+\infty$, and $M_{P_c}^h$ (the accumulated times of activation of P_c in space S_h) is initialized to 1.

If (5) is not satisfied, i.e., sample $\mathbf{x}$ is not similar to P_c in space S_l, we will create a new prototype for $\mathbf{x}$ as

$$\begin{aligned} W_{\text{new}} &= \mathbf{x} \\ M_{\text{new}}^l &= 1, \quad M_{\text{new}}^h = 1 \\ T_{\text{new}}^l &= +\infty, \quad T_{\text{new}}^h = +\infty. \end{aligned} \tag{10}$$

2) Learning in High-Dimensional Space: If (4) is not satisfied, i.e.,

$$\mathbf{dim}(P_c) = n + m \tag{11}$$

P_c already belongs to S_h. The learning of PEN in the low space S_l is disabled (failed). PEN will deal with $\mathbf{x}$ in the high space S_h using P_a and P_b, which are calculated by (2) and (3). If

$$\|\mathbf{x} - W_{P_a}\|_{S_h} = 0 \tag{12}$$

i.e., sample $\mathbf{x}$ is equal to P_a, then P_a is activated and the accumulated times of activation $M^l_{P_a}$ and $M^h_{P_a}$ will be added by 1. Else, we will check the following condition:

$$\|\mathbf{x} - W_{P_a}\|_{S_h} \le T^h_{P_a} \quad \textbf{and} \quad \|\mathbf{x} - W_{P_b}\|_{S_h} \le T^h_{P_b}. \tag{13}$$

If it is satisfied, i.e., the distances between P_a and $\mathbf{x}$, P_b and $\mathbf{x}$ are less than the similarity threshold $T^h_{P_a}$ and $T^h_{P_b}$, P_a and P_b are activated simultaneously, we do

$$M^l_{P_a} = M^l_{P_a} + 1, \quad M^h_{P_a} = M^h_{P_a} + 1. \tag{14}$$

Then, we move prototype P_a toward $\mathbf{x}$ as

$$\begin{aligned} W^l_{P_a} &= W^l_{P_a} + \left(1/M^l_{P_a}\right)\left(\mathbf{x}^l - W^l_{P_a}\right) \\ W^{h-l}_{P_a} &= W^{h-l}_{P_a} + \left(1/M^h_{P_a}\right)\left(\mathbf{x}^{h-l} - W^{h-l}_{P_a}\right) \end{aligned} \tag{15}$$

where W^l_* represents the attributes of space S_l, i.e., $W^l_* = (w_1, w_2, \ldots, w_n)$. W^{h-l}_* represents the attributes of new perception neurons, i.e., $W^{h-l}_* = (w_{n+1}, w_{n+2}, \ldots, w_{n+m})$. If (13) is not satisfied, i.e.,

$$\|\mathbf{x} - W_{P_a}\|_{S_h} > T^h_{P_a} \quad \textbf{or} \quad \|\mathbf{x} - W_{P_b}\|_{S_h} > T^h_{P_b} \tag{16}$$

we will create a new prototype for $\mathbf{x}$ using (10).

D. Prototype Self-Adaptive Associating

Prototype self-adaptive associating process among prototypes is conducted through connection establishing, strengthening, weakening, and removing.

Connection establishing and strengthening are conducted according to the Hebbian learning rule [36]. If P_a and P_b are activated simultaneously, i.e., (13) is satisfied, we will establish a connection between P_a and P_b; if there is no connection between them, then we set the outdating degree $\mathrm{Age}_{(a,b)}$ to 0 to represent that the connection is the most recent. If there is already a connection between P_a and P_b, we set $\mathrm{Age}_{(a,b)}$ to 0 to strengthen the connection.

For the dynamically changing environment, the prototypes change their locations slowly during learning. Prototypes that are connected to each other at an early stage may not be near at an advanced stage, and we remove such connections. If prototype P_i is mapped to space S_h or moved toward the input sample $\mathbf{x}$, we weaken the connections emanating from P_i by increasing the outdating degree of these connections as

$$\mathrm{Age}_{(i,j)} = \mathrm{Age}_{(i,j)} + 1, \quad P_j \in N_{P_i} \tag{17}$$

where $\mathrm{Age}_{(i,j)}$ is the outdating degree of the connection between prototype P_i and P_j. N_{P_i} is the neighbor prototype set of P_i, i.e., the set of the prototypes connected to P_i. Connections whose Age is larger than a predefined threshold $\mathrm{Age}_{\max}$ will be removed.

E. Adaptation of the Similarity Threshold

Similarity threshold T is used to decide whether to add a new prototype or move an existing prototype when an input sample comes. In PEN, we do not use global predefined threshold; we give each prototype a similarity threshold, which is adaptively changing with the environment. The threshold of P_i is decided by the prototype-distribution around P_i. If P_i does not have a neighbor prototype, the similarity threshold of P_i is calculated using the minimum distance between P_i and all the other prototypes

$$T^k_{P_i} = \min_{P_j \in P \setminus P_i} \|W_{P_i} - W_{P_j}\|_{S_k}, \quad k = l, h. \tag{18}$$

If P_i has neighbor prototypes, the similarity threshold is calculated using the maximum distance between P_i and its neighbor prototypes

$$T^k_{P_i} = \max_{P_j \in N_{P_i}} \|W_{P_i} - W_{P_j}\|_{S_k}, \quad k = l, h. \tag{19}$$

For the prototype P_i belonging to space S_l, we calculate only threshold $T^l_{P_i}$. For the prototype P_i belonging to space S_h, we calculate the thresholds $T^l_{P_i}$ and $T^h_{P_i}$. To improve the computational efficiency, in practice, we update only the thresholds of P_a, P_b, and P_c when an input sample comes.

F. Prototype Pruning

Prototype pruning is conducted after every λ samples learned. The prototypes that existed before the new sensory neurons appear are not the targets to be pruned, because these prototypes are gained by a period of learning and they are already pruned by PEN during that period; they are a knowledge base of PEN. The prototypes that are created after new sensory neurons appear will be pruned, and we use set Q to represent these prototypes. The isolated prototype with small M value in Q will be removed. First, we calculate the mean value of $M^h_{P_i}$ as

$$M^h_{\mathrm{mean}} = \sum_{P_i \in P^h} M^h_{P_i} / |P^h| \tag{20}$$

where P^h represents the prototypes belonging to space S_h, $|P^h|$ represents the element number of set P^h, i.e., the number of prototypes belonging to space S_h. If

$$|N_{Q_i}| = 0 \quad \textbf{and} \quad M^h_{Q_i} < M^h_{\mathrm{mean}} \tag{21}$$

or if

$$|N_{Q_i}| = 1 \quad \textbf{and} \quad M^h_{Q_i} < c \times M^h_{\mathrm{mean}} \tag{22}$$

is satisfied, where $Q_i \in Q$, $0 \le c \le 1$, large c means much noise, vice versa, $|N_{Q_i}|$ represents the neighbor number of prototype Q_i, we will remove prototype Q_i, connections from Q_i are also removed simultaneously.

G. Postprocessing

Postprocessing is enabled after all prototypes are mapped to space S_h. This procedure will disable some operations in the prototype competition, prototype learning phases, and simplify the prototype pruning process. Meanwhile, some parameters will be simplified.

Because all prototypes are mapped to space S_h, there is no prototype that needs to be mapped to space S_h. We do not find winner prototype P_c when an input sample $\mathbf{x} \in S_h$ comes, we only find prototypes P_a and P_b. Then, procedure from (4)–(10) (i.e., the procedure of mapping a prototype to space S_h) does not need to be executed anymore; we directly deal with $\mathbf{x}$ using P_a and P_b: if (12) is satisfied, $M^h_{P_a}$ is added by 1, else, we decide whether to move the winner prototype or create a new prototype according to (13) and (16); we replace parameters $M^l_{P_i}$ and $M^h_{P_i}$ with one parameter M_{P_i}

$$M_{P_i} = \left(M^h_{P_i} + M^l_{P_i}\right)/2, \quad P_i \in P. \tag{23}$$

Then, weight vector updating (15) can be simplified into

$$W_{P_a} = W_{P_a} + (1/M_{P_a})(\mathbf{x} - W_{P_a}). \tag{24}$$

For (10), which is used to create a new prototype, the parameters M^l_{new} and T^l_{new} are no longer needed, that is

$$W_{\text{new}} = \mathbf{x}; \quad M^h_{\text{new}} = 1; \quad T^h_{\text{new}} = +\infty. \tag{25}$$

Prototype self-adaptive associating process remains unchanged. The parameter $T^l_{P_i}$ of each prototype P_i can be removed, then we only update threshold $T^h_{P_i}$ of each prototype P_i. The prototype pruning is conducted on all prototypes, i.e., set P, instead of set Q.

Note: We remove the prototypes that cannot be mapped to space S_h after a long period of learning (in the experiment, after all samples $\mathbf{x} \in S_h$ are learned) because they are potential distortion prototypes (Fig. 20).

If some new sensory neurons emerge later, we take the current space S_h as space S_l, the space with dimension of these new sensory neurons as new S_h, then we get a recursive process.

H. Initial Learning Stage of PEN

In this section, we give the details of the initial learning stage before the n sensory neurons evolute to $m + n$ sensory neurons. The prototype field of PEN is empty in the beginning, and learning samples are fed into the network sequentially. PEN will create two prototypes using the first two input samples. For the latter input sample, the learning process is conducted similar to the postprocessing process: when an input sample $\mathbf{x} \in \mathbb{R}^n$ (i.e., $x \in S_l$) comes, we find the winner and runner-up prototypes P_a and P_b in space S_l as

$$P_a = \underset{P_i \in P}{\operatorname{argmin}} \|\mathbf{x} - W_{P_i}\|_{S_l} \tag{26}$$

$$P_b = \underset{P_i \in P \setminus P_a}{\operatorname{argmin}} \|\mathbf{x} - W_{P_i}\|_{S_l}. \tag{27}$$

Then, we check the following condition formula:

$$\|\mathbf{x} - W_{P_a}\|_{S_l} = 0 \tag{28}$$

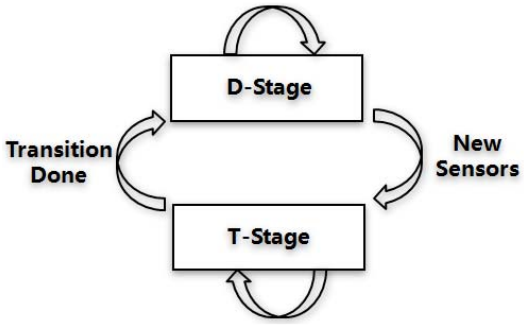

Fig. 7. Learning circle of PEN.

if it is satisfied, sample $\mathbf{x}$ is equal to P_a, then P_a is activated and the activation time $M^l_{P_a}$ is added by 1; else, we decide whether to create a new prototype or move the winner prototype according to the following condition:

$$\|\mathbf{x} - W_{P_a}\|_{S_l} \le T^l_{P_a} \quad \textbf{and} \quad \|\mathbf{x} - W_{P_b}\|_{S_l} \le T^l_{P_b}. \tag{29}$$

If it is satisfied, i.e., P_a and P_b are activated simultaneously, we add the activation time $M^l_{P_a}$ as

$$M^l_{P_a} = M^l_{P_a} + 1 \tag{30}$$

then, we move prototype P_a toward $\mathbf{x}$ as

$$W_{P_a} = W_{P_a} + \left(1/M^l_{P_a}\right)(\mathbf{x} - W_{P_a}). \tag{31}$$

If the condition is not satisfied, we will create a new prototype for $\mathbf{x}$ as

$$W_{\text{new}} = \mathbf{x}; \quad M^l_{\text{new}} = 1; \quad T^l_{\text{new}} = +\infty. \tag{32}$$

Prototype self-adaptive associating process remains unchanged. The threshold is updated by (18) and (19), where $k = l$. The prototype pruning process will be conducted on all prototypes instead of set Q; similarly, h is replaced by l in (20)–(22).

I. Complete Algorithm of PEN

As a summary, we give the complete algorithm of PEN in Algorithm 1. The algorithm is divided into two stages, as shown in Fig. 7: 1) the development-stage (D-Stage) and 2) the transition-stage (T-Stage). T-Stage is the learning stage when new sensory neurons emerge, i.e., in this stage, low-dimensional prototypes will be mapped to high-dimensional space. After all prototypes are mapped to high-dimensional space, PEN enters the D-Stage, i.e., input samples and prototypes stored in the network are in the same space in D-Stage. Actually, both the postprocessing and the initial learning stage of PEN belong to D-Stage.

IV. Analysis

A. Move a Prototype or Add a New One?

As shown in Fig. 8, the learning process of prototype-based methods can be understood as a process of prototypes fitting the data distribution. Assume there are N prototypes in the network and the weight vector of prototype P_i $(0 \le i \le N)$ is

Algorithm 1 Perception Evolution Network

Initialize: Prototype set P with the first two input sample $\mathbf{x_1}$ and $\mathbf{x_2}$, i.e., $P = \{P_1, P_2\}$, where $P_i : \{W_i = \mathbf{x}_i, T_i^l = \|\mathbf{x}_1 - \mathbf{x}_2\|, M_i^l = 1\}, i = 1, 2$. Connection set $C = \emptyset$. Give the initial value of λ, c, and $\text{Age}_{\max}$.

D-Stage:

Receive input sample $\mathbf{x} \in S_l$.
Find prototype P_a and P_b using formula (26) and (27).
if $\|\mathbf{x} - W_{P_a}\|_{S_l} = 0$ **then**
  Add $M_{P_a}^l$ by 1.
else
  if $\|\mathbf{x} - W_{P_a}\|_{S_l} \le T_{P_a}^l$ **and** $\|\mathbf{x} - W_{P_b}\|_{S_l} \le T_{P_b}^l$ **then**
    Update P_a using formula (30) and (31).
  else
    Create a new prototype using formula (32).
  end if
end if
Execute Prototype Self-Adaptive Associating procedure in space S_l.
Update the Similarity Threshold of the prototypes in space S_l.
if λ samples have been learned **then**
  Execute Prototype Pruning procedure.
end if
if New sensory neurons emerge **then**
  Go to T-Stage.
end if
Go to Step 1.

T-Stage:

Receive input sample $\mathbf{x} \in S_h$.
Find prototype P_c using formula (1).
Find prototype P_a and P_b using formula (2) and (3).
if $\dim(P_c) = \dim(S_l)$ **then**
  if $\|\mathbf{x} - W_{P_c}\|_{S_l} \le T_{P_c}^l$ **then**
    Update prototype P_c using formula (6) and (9).
  else
    Create a new prototype using formula (10).
  end if
else
  if $\|\mathbf{x} - W_{P_a}\|_{S_h} = 0$ **then**
    Add $M_{P_a}^l$ and $M_{P_a}^h$ by 1.
  else
    if $\|\mathbf{x} - W_{P_a}\|_{S_h} \le T_{P_a}^h$ **and** $\|\mathbf{x} - W_{P_b}\|_{S_h} \le T_{P_b}^h$ **then**
      Update P_a using formula (14) and (15).
    else
      Create a new prototype using formula (10).
    end if
  end if
end if
Execute Prototype Self-Adaptive Associating procedure.
Update the Similarity Threshold of the prototypes.
if λ samples have been learned **then**
  Execute Prototype Pruning procedure.
end if
if All prototypes are mapped to space S_h **then**
  Simplify paraments as in Postprocessing procedure.
  Let $S_l = S_h$. Go to D-stage.
end if
Go to Step 21.

denoted by W_i, and the fitting error $Error$ can be represented as follows:

$$\text{Error} = \int P(u)\min_{1\le i\le N}\|u - W_{P_i}\|du. \tag{33}$$

Learning process is to make (33) as small as possible or minimize it. The mainstream of the current method to reduce the fitting error is based on competitive learning: make the winner prototype (and prototypes around the winner) move toward the input pattern.

Now, we assume that a new data distribution $Q(v)$ arrives, as shown in Fig. 8(b). The winner prototype (and prototypes around the winner) will move toward the area of $Q(v)$.

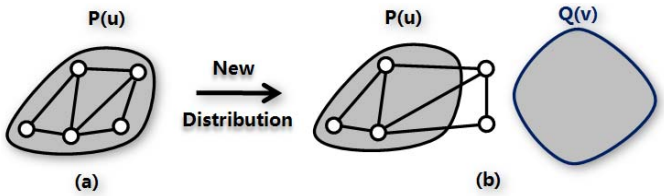

Fig. 8. (a) Distribution of prototypes after a period of learning on $P(u)$. (b) New data distribution $Q(v)$ is introduced to the learning system, and some prototypes are moving toward the area of $Q(v)$.

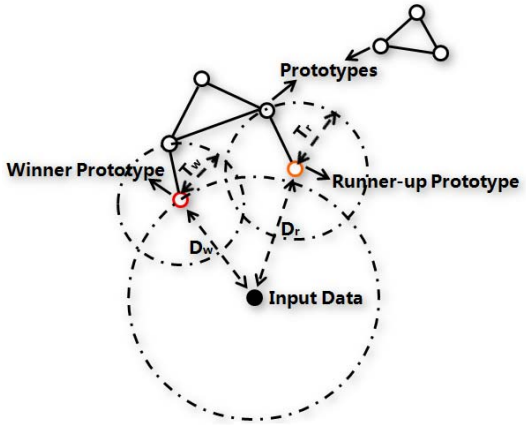

Fig. 9. Example of feature map $\mathcal{M}$ of the PEN in the feature space. T_w and T_r represent the similarity threshold of the winner prototype and the runner-up prototype. The input data is D_w away from the winner and D_r away from the runner-up. Assume $D_w > T_w$ and $D_r > T_r$.

The learning process present is actually under the following fitting error formula:

$$\text{Error} = \int Q(v)\min_{1\le i\le N}\|v - W_{P_i}\|dv. \tag{34}$$

In (34), 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 that 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 two prototypes in the intermediate position of distribution $P(u)$ and $Q(v)$.

For the above problem, a 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 an existing 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 III-E) to help make judgment.

In our method, we give each prototype a similarity threshold, which is decided by the nearby prototype-distribution and adaptively adjusted according to the changing environment (Section III-E). To illustrate our idea, we introduce an example as in Fig. 9. Assume that we already have a feature map $\mathcal{M}$ as shown in the figure, when an input

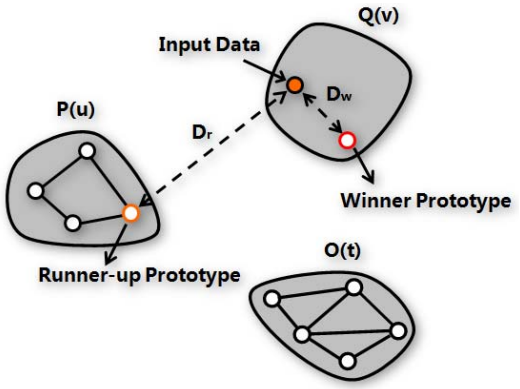

Fig. 10. Relaxation condition for the situation of nonstationary data. After a period learning of $O(t)$ and $P(u)$, a new data distribution $Q(v)$ is introduced, and a new prototype is created for the first input data of $Q(v)$.

data comes, we first find the winner and runner-up prototypes, which is defined as the local region of the input sample, and we get the following useful information about the prototype-distribution around this local region in the example.

1) *Information 1:* Some prototypes exist T_w away from the winner and T_r away from the runner-up.
2) *Information 2:* The input data is D_w away from the winner and D_r away from the runner-up. Meanwhile, $D_w > T_w$ and $D_r > T_r$.
3) *Information 3:* No prototype exists inside the round centered at the input data with radius D_w.

According to these, we can get the following conclusion. There is no sufficient reason to reject to create a new prototype for this input data. Simply stated, the PEN actually makes decision based on the state of the current feature map $\mathcal{M}$. After that, the PEN updates the current feature map using the present sample according to the decision. It is just like the Bayesian procedure. Then, all we need is giving an initial state of the feature map $\mathcal{M}$ of the PEN: $P = \{P_1, P_2\}$, where $P_i : \{W_i = \mathbf{x}_i, T_i = \|\mathbf{x}_1 - \mathbf{x}_2\|, M_i = 1\}, i = 1, 2$.

B. Why Use *or* Not *and*?

In Section IV-A, we give an example to illustrate when to add a new prototype. In that example, we use condition

$$D_w > T_w \textbf{ and } D_r > T_r. \tag{35}$$

However, in practice, we relax this constraint to

$$D_w > T_w \textbf{ or } D_r > T_r \tag{36}$$

as (16). This relaxation condition is for the situation of non-stationary data. The reason can be explained by the example shown in Fig. 10. In this example, we assume the following.

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 $Q(v)$.

When the first data of distribution $Q(v)$ arrives, the PEN is able to create a new prototype for this data according to (35). For the further input data $\mathbf{x}$ from $Q(v)$ (as shown in Fig. 10), the PEN finds the winner and runner-up prototypes. According to Assumption 1, we obtain

$$D_w < T_w \tag{37}$$

is true for all data from $Q(v)$. This will lead to (35) never being satisfied; therefore, no more prototypes will be created for $Q(v)$, which will lead to a large fitting error. However

$$D_r > T_r \tag{38}$$

may be satisfied, and then (36) is satisfied. Therefore, when using (36), new prototypes for the following input data can be created and organized to fit $Q(v)$ well.

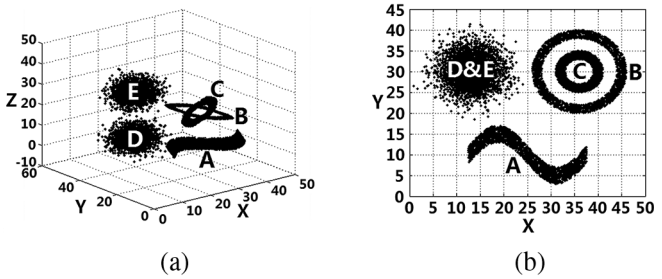

Fig. 11. (a) Artificial data set used in the experiment X-Y-Z space. (b) Representation of the learning data in the X-Y space.

V. Experiments

A. Artificial Data

We conduct this experiment on the artificial data set, as shown in Fig. 11(a). The data set is separated into five parts, and each data set contains 2000 samples. Set A satisfies 3-D sinusoidal distribution. Sets B and C satisfy 3-D ring distribution. They are perceived as two concentric rings in the X-Y space; however, in the X-Y-Z space, there is an angle between them. Sets 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. Data sets D and E show an actual phenomenon: in the early period, we have few perception neurons, the information we gained is only enough to make a rough cognition. However, with the evolution of perception, we get more information through some new perception neurons; then we can make a further cognition. It is a cognition deepening process. The parameters of the PEN are set as: 1) $\lambda = 200$; 2) $\text{Age}_{\max} = 200$; and 3) $c = 0.5$.

In the experiment, we first give the PEN two sensory neurons, i.e., one sensory neuron receives the X-dimension data and the other receives the Y-dimension data. After all samples have been learned, we give the PEN another sensory neuron to receive the Z-dimension data. We conduct the experiment in two environments.

1) *Closed Environment:* In this environment, samples are randomly selected from the whole learning set and fed to the network.
2) *Open-Ended Environment:* Much research focus on the concept-drifting problem [37]–[40], i.e., the statistical properties of one concept change over time in unforeseen ways. However, in this paper, we mainly focus on solving the stability-plasticity dilemma [9], i.e., keeping

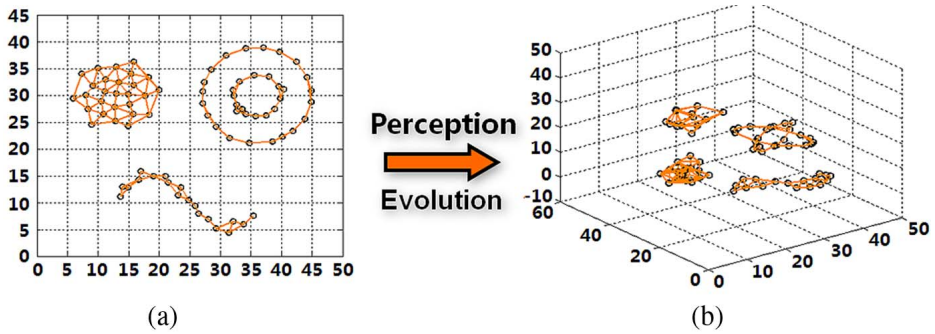

Fig. 12. Learning results of the PEN on the artificial data set in the closed environment. (a) Originally perception—learning result of the PEN in the X-Y space. (b) After perception evolution—learning result of the PEN after new sensory neurons appear. Black circles: learned prototypes and yellow lines are the connections between prototypes. As shown in (a), the PEN gets a satisfactory learning result, the learned prototypes fit the learning data very well. As shown in (b), learning result after PEN gets a new sensory neuron, which receives the input signal from Z-dimension. All prototypes are mapped to the X-Y-Z space, and the prototypes fit the learning data very well. We see that the PEN introduces the transition between the old cognition and the new higher cognition, i.e., the agent comes to a new world through the new sensory neuron Z using the PEN.

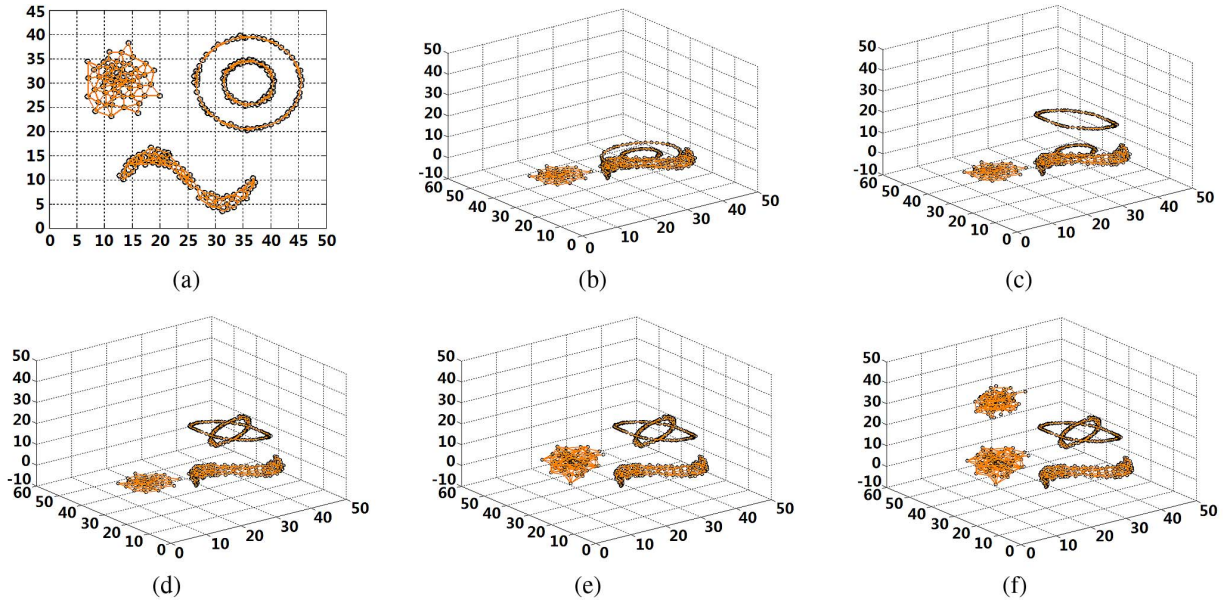

Fig. 13. Learning results of PEN on the artificial data set in the open-ended environment. (a) Learning result in X-Y space. (b) Perceiving A again. (c) Perceiving B again. (d) Perceiving C again. (e) Perceiving D again. (f) Perceiving E again. Black circles: learned prototypes. Yellow lines: connections between prototypes. (a) Learning result of PEN in the X-Y space, we see that the learned prototypes fit the learning data very well. (b)–(f) Periodical learning results of PEN after new sensory neurons appear. PEN is able to map the corresponding prototypes in the X-Y space to the X-Y-Z space accurately when data sets A to D are perceived again. After data set D, all prototypes are mapped to the X-Y-Z space. In (f), when we let PEN perceive data set E again, PEN is able to create new prototypes to represent data set E in the new high space. The transition between the old cognition (X-Y space) and the new higher cognition (X-Y-Z space) is done successfully.

learning new knowledge quickly without catastrophic forgetting of already learned, and still successful, memories. Both stability and plasticity are very important abilities for artificial intelligence [41], [42]. Learning period is divided into five different learning environments, in environment I, i.e., from steps 1 to 2000, samples are chosen randomly from data set A and fed to the network. After learning is finished in environment I, environment II is enabled, i.e., from steps 2001 to 4000, the environment changes and samples from B are chosen randomly and fed to the network, and so on.

The learning results of the PEN in the closed environment are shown in Fig. 12. As shown in Fig. 12(a), the PEN gets a satisfactory learning result, and the learned prototypes (black circles) fit the learning data very well. Fig. 12(b) shows the learning result after PEN gets a new sensory neuron, which receives the input signal from Z-dimension. All prototypes are mapped to the X-Y-Z space, and the prototypes fit the learning data very well. We can see that sets B and C are no longer concentric rings in the X-Y-Z space, sets D and E, which cannot be distinguished in the X-Y space, are separated from each other in the X-Y-Z space. The PEN comes to a new world through the new sensory neuron Z.

The learning results of the PEN in the open-ended environment are shown in Fig. 13. Fig. 13(a) shows the learning result in the X-Y space, i.e., the original perception space. We see that the learned prototypes fit the learning data very well. After PEN gets a new sensory neuron, we let the learning system perceive data set A again, and the original learned prototypes, which are used to represent set A in the X-Y space, are mapped to the X-Y-Z space as shown in Fig. 13(b); meanwhile, we can see that other prototypes

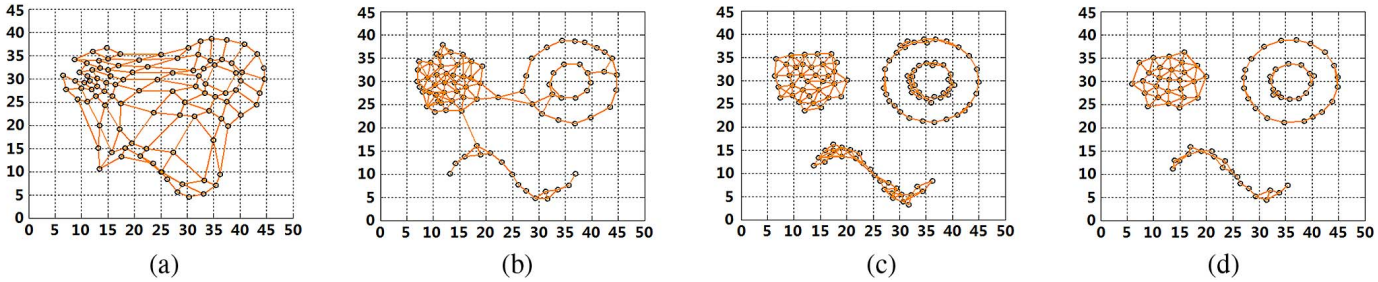

Fig. 14. Learning results of (a) SOM, (b) GNG, (c) ASOINN, and (d) PEN on the artificial data set in the X-Y space in the closed environment. We can see that there are some prototypes located in the areas where the learning data does not distribute in the learning results of SOM and GNG; they are distortion prototypes. The learning result of the PEN is much better than that of SOM and GNG, and comparable with that of ASOINN.

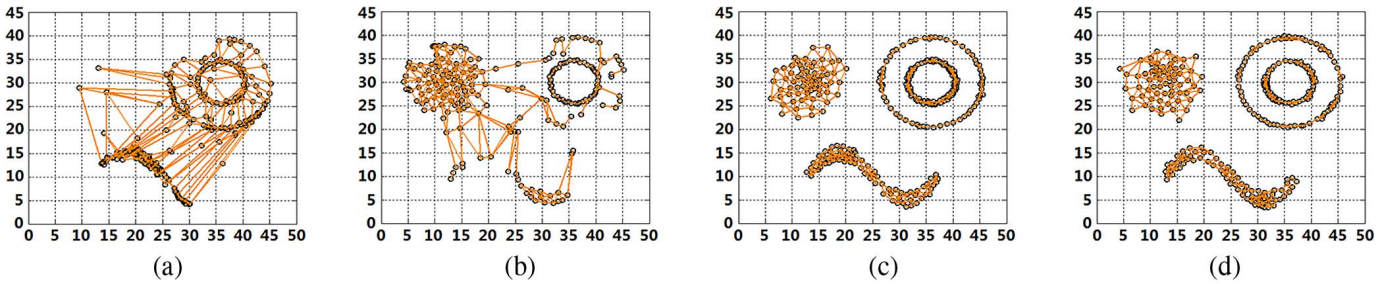

Fig. 15. Learning results of (a) SOM, (b) GNG, (c) ASOINN, and (d) PEN on the artificial data set in the X-Y space in the open-ended environment. The SOM is not able to learn new data distribution after a period of learning; it reaches the saturation point for learning new knowledge. The GNG is able to learn new data distribution; however, the previous learned structure of the data distribution is destroyed such as distributions A and B (catastrophic forgetting). The learning result of the PEN is much better than that of the SOM and GNG, and comparable with that of ASOINN.

remain unchanged. After learning is finished in environment I, we let the learning system perceive data set B again; similarly, the original learned prototypes, which are used to represent set B in the X-Y space, are mapped to the X-Y-Z space and other prototypes remain unchanged, as shown in Fig. 13(c). Fig. 13(d) and (e) shows a similar situation as Fig. 13(a) and (b). Then, the transition between the old cognition (X-Y space) and the new higher cognition (X-Y-Z space) is done by the PEN. When we let the learning system perceive data set E again, the PEN creates some new prototypes to represent the new knowledge data set E.

Meanwhile, we compare the PEN with some typical methods including SOM, GNG, and ASOINN. We also conduct the experiment in two environments.

1) In the closed environment, samples are randomly selected from the whole learning set and fed to the network.
2) In the open-ended environment (for the stability-plasticity dilemma), the networks are successively trained with samples from the subdistributions A, B, C, D, and E.

To facilitate the observation, we train the four methods using the X-Y space data. To make SOM and GNG generate similar scale of prototypes with the PEN, we set the grid size of SOM as 9×9 in the stationary environment and 15×15 in the nonstationary environment. λ of GNG is set to 125 in the stationary environment and 45 in the nonstationary environment; other parameters are set as [21]. The parameters and ASOINN are set as [29].

The learning results of SOM, GNG, ASIONN, and PEN in the closed environment are shown in Fig. 14. We can see that in the learning results of SOM and GNG, there are some prototypes located in the areas where the learning data does not distribute; they are distortion prototypes. The learning result of the PEN is much better than that of SOM and GNG, and comparable with that of ASOINN.

The learning results in the open-ended environment are shown in Fig. 15. In Fig. 15(a), we can see that the SOM is not able to learn new data distribution after a period of learning; it reaches the saturation point for learning new knowledge. On the other hand, the GNG is able to learn new data distribution; however, the previous learned structure of the data distribution is destroyed, as shown in Fig. 15(b). The learning result of the PEN is much better than that of SOM and GNG, and comparable with that of the ASOINN. Moreover, the PEN has the ability of permitting the emergence of new sensory neurons, which SOM, GNG, and ASOINN do not have.

B. Real-World Data

The RGB-D object data set [43] is used in this experiment. The data set is collected using a Kinect-style 3-D camera that records synchronized and aligned RGB and depth images. The objects have been segmented from the background by the authors. We use 20 objects, and the first 25 images (the object in each image is rotated by an angle) of each object (500 images in total) are used in the training phase of the experiment. In the testing phase, we use 200 more images, the testing images are taken in different angles from the training images, and each object includes ten testing images. We crop the images to 80×80. Figs. 16 and 17 show some examples of the RGB images and depth images used in the experiment.

As mentioned in [5], the dichromatic color vision comes from just two kinds of visual pigments: 1) one absorbs short-wavelength light and 2) the other is more sensitive to longer wavelengths. These two kinds of wavelengths

Fig. 16. Twenty objects (RGB image) used in the experiment. Due to space constraints, we only show one image per object.

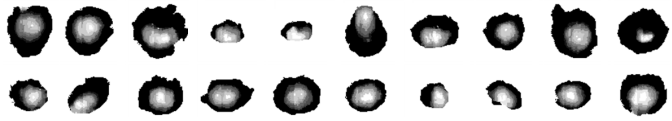

Fig. 17. Twenty objects (depth image) used in the experiment. Due to space constraints, we only show one image per object.

correspond to the colors of blue and green. The trichromatic color vision has one more visual pigment, which absorbs light corresponding to the red color. Therefore, in the experiment, we first give the PEN two perception channels (or two types of perception neurons) to receive the G and B parts of the RGB images (Period I); after all samples have been learned, we give another perception channel (or perception neurons) to the PEN to receive the R part of the RGB images, i.e., the PEN is used to perceive green, blue, and red color synchronously (Period II).

In recent years, depth cameras have been widely used. We assume a scene: there is a robot using an RGB camera as its eyes; now, we install a depth camera to the robot's eyes, then, the robot has one more way to perceive the real world. Thus, after the learning has finished in the RGB space (Period II), we give another perception channel (or perception neurons) to the PEN to receive the depth information, i.e., the PEN is used to perceive the RGB color and spatial information synchronously (Period III).

Overall, the evolution of the perception is as follows:

$$\textbf{GB} \rightsquigarrow \textbf{RGB} \rightsquigarrow \textbf{RGB-D}. \tag{39}$$

The parameters of the PEN are set as: 1) $\lambda = 200$; 2) $\text{Age}_{\max} = 200$; and 3) $c = 0.5$.

We also conduct this experiment in stationary and open-ended environments. We do 10 000 learning iterations for each period, i.e., Periods I, II, and III. In the stationary environment, training images are randomly chosen from the training data set in the 10 000 learning iterations. In the open-ended environment, we design a more complex situation: we first give the images of the first ten objects in the first 5000 learning iterations, training images are randomly chosen from the training data set during learning. After the learning has finished, we give the images of the remaining ten objects in the second 5000 learning iterations; similarly, training images are randomly chosen. We conduct the experiment 100 times in different random sequences of the samples for both closed- and open-ended environments.

The number of the prototypes learned by the PEN in the closed environment ranges from 143 to 171 (for 100 times experiments), the mean value of the prototype number is 155.6. In the open-ended environment, the number of the prototypes learned by PEN ranges from 225 to 236, the mean value of the prototype number is 228.4. Fig. 18 shows the learning result

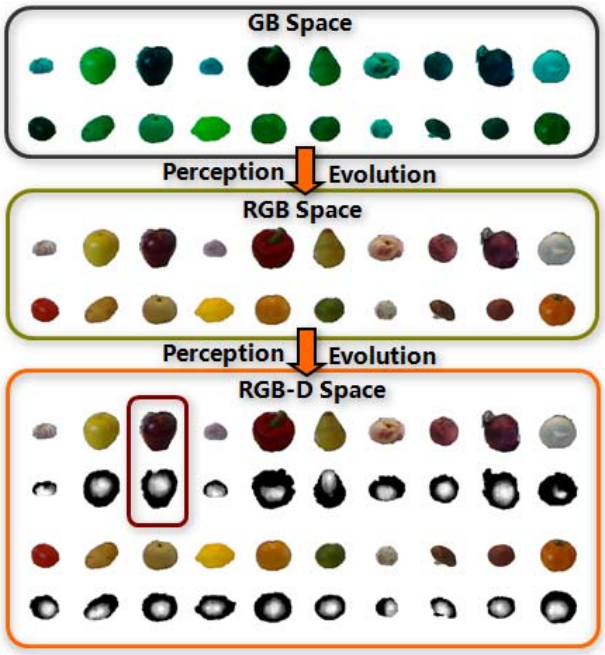

Fig. 18. Learning result of the PEN from GB space to RGB-D space. The prototypes are mapped to high-dimensional space perfectly when PEN gets a new dimension of perception. In the GB space, there are few kinds of colors. After the dimension R comes, much more distinguishable colors appear. At last, after dimension D comes, the PEN acquires both the image information and the spatial information of the objects. Then, the agent comes to a new world with the concept of space.

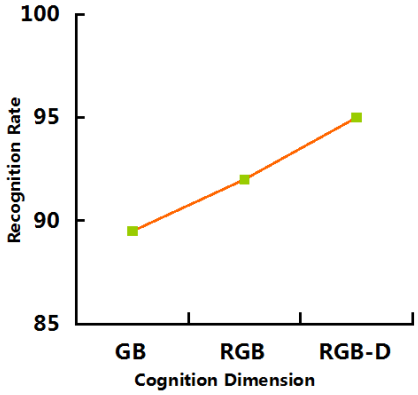

Fig. 19. Trend of the recognition rate according to the cognition dimension.

of PEN from GB space to RGB-D space of 20 prototypes (one prototype for each object). Due to space constraint, we do not show all the prototypes; learning results of the other prototypes are similar to these 20 prototypes. In the RGB-D space of Fig. 18, the RGB image and the depth image of a prototype are shown separately in two adjacent rows; actually, the two images are integrated in one prototype by PEN. In Fig. 18, we can see that the prototypes are mapped to high-dimensional space perfectly when PEN gets a new dimension of perception. In the GB space, there are few kinds of colors. After dimension R comes, much more distinguishable colors appear. At last, after dimension D comes, the PEN acquires both the image information and the spatial information of the objects. Then, the agent comes to a new world with the concept of space.

Fig. 19 shows the trend of the recognition rate according to the cognition dimension. In the GB space, the PEN can

TABLE I

STATISTICAL RESULTS OF 100 EXPERIMENTS IN THE CLOSED ENVIRONMENT (MEAN + STD). ME: MEAN ENTROPY. MQE: MEAN QUANTIZATION ERROR. CR: CORRECT RECOGNITION RATIO. THE BEST RESULTS ARE IN BOLD

	SOM	GNG	ASOINN	PEN
ME	2.51±0.03	2.49±0.03	0.92±0.01	**0.021±0.004**
MQE	7.47±0.31	5.89±0.28	3.05±0.21	**1.43±0.18**
CR	73.5±0.80%	76.6±0.80%	88.2±0.71%	**94.2±0.58%**

TABLE II

STATISTICAL RESULTS OF 100 EXPERIMENTS IN THE OPEN-ENDED ENVIRONMENT (MEAN + STD). ME: MEAN ENTROPY. MQE: MEAN QUANTIZATION ERROR. CR: CORRECT RECOGNITION RATIO. THE BEST RESULTS ARE IN BOLD

	SOM	GNG	ASOINN	PEN
ME	2.53±0.06	2.51±0.06	1.13±0.04	**0.021±0.003**
MQE	8.50±0.40	6.17±0.38	3.28±0.27	**1.41±0.15**
CR	55.5±0.86%	57.9±0.79%	83.4±0.64%	**94.5±0.51%**

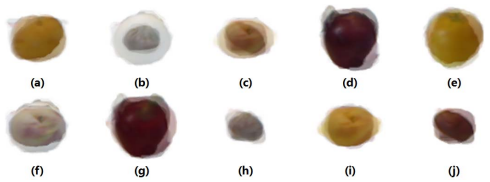

Fig. 20. (a)–(j) Distortion prototypes generated by SOM, GNG, and ASOINN. Depth images are not displayed.

only see green and blue color, and it classifies the GB part of the testing images by nearest neighbor method. The RGB and RGB-D spaces are treated similarly. In Fig. 19, we see that the recognition rate is increasing with the deepening of the cognition dimension.

We also compare the PEN with SOM, GNG, and ASOINN. The RGB-D images are used to train the methods. We conduct the experiment for the above methods in both closed and open-ended environments. To make SOM and GNG generate similar scale of prototypes with the PEN, we set the grid size of SOM as 12×13 in the closed environment and 15×15 in the open-ended environment. λ of GNG is set to 65 in the closed environment and 45 in the open-ended environment; other parameters of GNG are set as in [21]. The parameters of ASOINN are set as in [29].

In the learning results, as shown in Fig. 20, we find that there are many distortion prototypes generated by SOM, GNG, and ASOINN; these prototypes merge many different types of objects. They are meaningless and useless. To quantify the comparison, we give each prototype an entropy to measure the purity of the prototype. Assume there are k objects in the learning set, prototype P_i merges m samples coming from k objects $\{o_1, o_2, \ldots, o_k\}$, the entropy of P_i can be calculated as

$$\text{Entr}(P_i) = -\sum_{i=1}^{k} p(o_i) \log p(o_i). \tag{40}$$

Because the weight vector of P_i, i.e., W_{P_i}, is the weighted sum of the weight vectors of the m samples, $p(o_i)$ represents the proportion of the weight vector of o_i in W_{P_i}. If $p(o_i) = 0$, then $p(o_i) \log p(o_i) = 0$. $Entr(P_i) = 0$ means that the m samples come from one object; obviously, this is the best case. We calculate the entropy of each prototype and get the mean value mean entropy (ME). We find that almost all prototypes gained by the PEN get 0 entropy value, and the MEvalue of the PEN is much smaller than the other methods in both stationary and open-ended environments as shown in Table I (for closed environment) and Table II (for open-ended environment); it means that PEN seldom produces distortion prototypes.

We also calculate the mean quantization errors (MQEs) of these four methods; assume that there are k samples in the learning set, i.e., $L = \{\mathbf{x}_i | 1 \le i \le k\}$, the learning method gets n prototypes, i.e., $P = \{P_i | 1 \le i \le n\}$. The total quantization error (TQE) is calculated as

$$\text{TQE} = \sum_{i=1}^{k} \min_{1 \le j \le n} \|\mathbf{x}_i - W_{P_j}\|. \tag{41}$$

Then, we calculate the MQE, i.e., TQE divided by the sample number k. Tables I and II show that PEN gets a much smaller MQE than the other methods in both closed and open-ended environments.

Finally, we use 200 more images to test the model learned by these four methods. The testing images are taken in different angles from the 500 training images, and each object includes ten different testing images. Nearest neighbor is used to classify the testing images.

The correct recognition (CR) ratio of the four methods in the closed environment and the open-ended environment are shown in Tables I and II, and the CR ratio of the PEN is much higher than the other methods. Comparing the learning results of the two environments, we can see that the CR ratio of SOM and GNG drops rapidly in the open-ended environment: 1) 73.5% and 76.6% in the closed environment and 2) 55.5% and 57.9% in the open-ended environment. The reason is that the SOM is not able to learn new data distribution after a period of learning; most of the last ten objects are not remembered by SOM. On the other hand, the GNG is able to learn the last ten objects; however, the previous learned prototypes from the first ten objects are destroyed when GNG learns the new ten objects. The CR ratio of ASOINN also drops in the open-ended environment: 1) 88.2% in the closed environment and 2) 83.4% in the

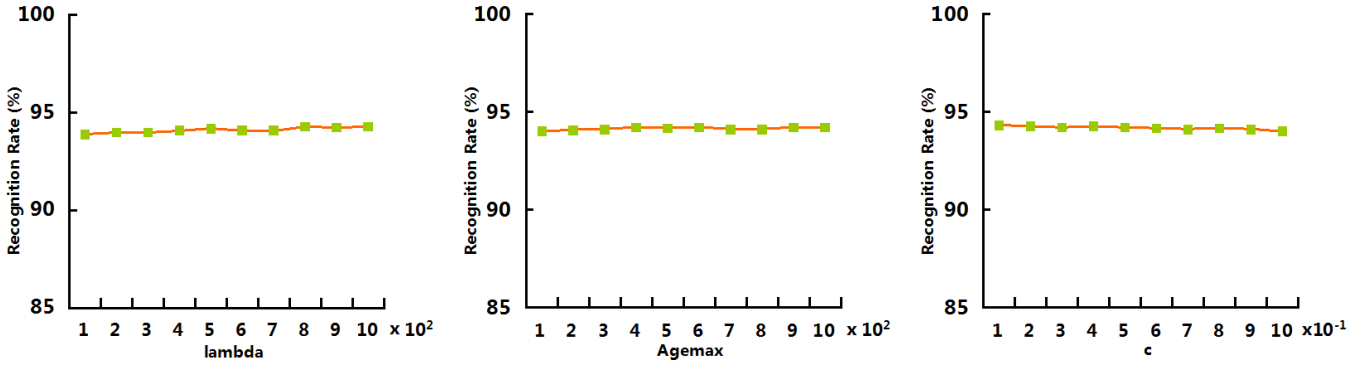

Fig. 21. Parameter influence on PEN. (a) Influence of λ on the accuracy by varying λ from 100 to 1000. (b) Influence of $\text{Age}_{\max}$ on the accuracy by varying $\text{Age}_{\max}$ from 100 to 1000. (c) Influence of c on the accuracy by varying c from 0.1 to 1.

open-ended environment. The CR ratio of the PEN in both stationary and open-ended environments is very stable; all values are larger than 90%: 1) 94.2% in the closed environment and 2) 94.5% in the open-ended environment, which are much better than that of SOM, GNG, and ASOINN.

C. Parameter Influence

Here, we discuss the influence of parameters on PEN, i.e., λ, $\text{Age}_{\max}$, and c. When setting different values of the parameters, the prototypes are mapped to high-dimensional space perfectly as PEN gets a new dimension of perception, which is similar to Fig. 18. We also test the influence of parameters on the recognition ratio CR. Fig. 21 shows that the setting of λ, $\text{Age}_{\max}$, and c does not have a strong influence on the classification accuracy CR.

D. Experiment Summary

In the experiments, the PEN is able to create new prototypes for new categories of objects effectively; this implies that PEN has a degree of freedom on the breadth of cognition. It also permits the emergence of a new dimension of perception, the prototypes are mapped to high-dimensional space perfectly by PEN; it means that PEN has a degree of freedom on the depth of cognition. Meanwhile, the new dimension of perception will promote the breadth of cognition, i.e., the PEN will find some new categories of objects in the new perceived world.

Comparing with some typical methods including SOM, GNG, and ASOINN, PEN gets much better learning results in both the closed- and open-ended environments.

VI. Conclusion

In this paper, we propose a PEN for unsupervised learning and online learning. It is a biologically inspired computing model, which permits the emergence of a new dimension of perception in the perception field of the network. When some new dimension of perception appears, the PEN is able to integrate the new dimensional sensory inputs with the learned prototypes. Meanwhile, it also can learn suitable prototypes automatically from data in an incremental and unsupervised way, and it does not require any predefined prototype number or similarity threshold. The experiments for both artificial data set and real-world data set show that the PEN is effective.

From the experiment on the RGB-D data set, we see that the PEN can deal with many potential practical problems. For example, if we install new sensors to a robot to expand its sensing capability, with PEN, we do not need to retrain the robot offline from scratch; the information gathered from the new installed sensors is fused with the existing knowledge in the system by the PEN automatically. This means that the PEN will have broad applications such as robot system, information fusion, and data stream mining.

Acknowledgment

The authors would like to thank the editors and anonymous reviewers for providing their valuable and constructive suggestions.

References

[1] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, pp. 386–408, 1958.
[2] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Cambridge, MA, USA: MIT Press, 1992.
[3] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybern.*, vol. 43, no. 1, pp. 59–69, 1982.
[4] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, Jul. 2006.
[5] G. H. Jacobs, G. A. Williams, H. Cahill, and J. Nathans, "Emergence of novel color vision in mice engineered to express a human cone photopigment," *Science*, vol. 315, pp. 1723–1725, Mar. 2007.
[6] D. Viejo, J. Garcia-Rodriguez, and M. Cazorla, "Combining visual features and growing neural gas networks for robotic 3D SLAM," *Inf. Sci.*, vol. 276, pp. 174–185, Aug. 2014.
[7] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Inf. Fusion*, vol. 14, no. 1, pp. 28–44, 2013.
[8] J. Gama, *Knowledge Discovery From Data Streams*. Boca Raton, FL, USA: Chapman & Hall, 2010.
[9] G. A. Carpenter and S. Grossberg, "The ART of adaptive pattern recognition by a self-organizing neural network," *Computer*, vol. 21, no. 3, pp. 77–88, Mar. 1988.
[10] G. A. Carpenter and S. Grossberg, "Adaptive resonance theory," Dept. Cognitive Neural Syst. Center Adapt. Syst. Center Excellence Learn. Edu., Sci., Technol., Boston Univ., Boston, MA, USA, Tech. Rep. CAS/CNS-TR-2009-008, 2009.
[11] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 31, no. 4, pp. 497–508, Nov. 2001.

[12] S. Grossberg, "How does a brain build a cognitive code?" in *Studies of Mind and Brain* (Boston Studies in the Philosophy of Science), vol. 70. Amsterdam, The Netherlands: Springer-Verlag, 1982, pp. 1–52.
[13] E. Berglund and J. Sitte, "The parameterless self-organizing map algorithm," *IEEE Trans. Neural Netw.*, vol. 17, no. 2, pp. 305–316, Mar. 2006.
[14] L. Xu and T. W. S. Shing, "Self-organizing potential field network: A new optimization algorithm," *IEEE Trans. Neural Netw.*, vol. 21, no. 9, pp. 1482–1495, Sep. 2010.
[15] T. Martinetz and K. Schulten, "Topology representing networks," *Neural Netw.*, vol. 7, no. 3, pp. 507–552, 1994.
[16] T. Martinetz, "Competitive Hebbian learning rule forms perfectly topology preserving maps," in *Proc. Int. Conf. Artif. Neural Netw.*, Amsterdam, The Netherlands, Sep. 1993, pp. 427–434.
[17] T. M. Martinetz, S. G. Berkovich, and K. J. Schulten, "'Neural-gas' network for vector quantization and its application to time-series prediction," *IEEE Trans. Neural Netw.*, vol. 4, no. 4, pp. 558–569, Jul. 1993.
[18] M. Sayed-Mouchaweh and E. Lughofer, Eds., *Learning in Non-Stationary Environments: Methods and Applications*. New York, NY, USA: Springer-Verlag, 2012.
[19] H.-U. Bauer and K. R. Pawelzik, "Quantifying the neighborhood preservation of self-organizing feature maps," *IEEE Trans. Neural Netw.*, vol. 3, no. 4, pp. 570–579, Jul. 1992.
[20] B. Fritzke, "Growing cell structures—A self-organizing network for unsupervised and supervised learning," *Neural Netw.*, vol. 7, no. 9, pp. 1441–1460, 1994.
[21] B. Fritzke, "A growing neural gas network learns topologies," in *Proc. Adv. Neural Inf. Process. Syst.*, Nov. 1995, pp. 625–632.
[22] F. H. Hamker, "Life-long learning cell structures—Continuously learning without catastrophic interference," *Neural Netw.*, vol. 14, nos. 4–5, pp. 551–573, May 2001.
[23] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Netw.*, vol. 4, no. 6, pp. 759–771, Dec. 1991.
[24] D. Deng and N. Kasabov, "On-line pattern analysis by evolving self-organizing maps," *Neurocomputing*, vol. 51, no. 4, pp. 87–103, Apr. 2003.
[25] M. Tscherepanow, M. Kortkamp, and M. Kammer, "A hierarchical ART network for the stable incremental learning of topological structures and associations from noisy data," *Neural Netw.*, vol. 24, no. 8, pp. 906–916, Oct. 2011.
[26] E. Lughofer, "Extensions of vector quantization for incremental clustering," *Pattern Recognit.*, vol. 41, no. 3, pp. 995–1011, Mar. 2008.
[27] S. Furao and O. Hasegawa, "An incremental network for on-line unsupervised classification and topology learning," *Neural Netw.*, vol. 19, no. 1, pp. 90–106, Jan. 2006.
[28] S. Furao, T. Ogura, and O. Hasegawa, "An enhanced self-organizing incremental neural network for online unsupervised learning," *Neural Netw.*, vol. 20, no. 8, pp. 893–903, Oct. 2007.
[29] F. Shen and O. Hasegawa, "A fast nearest neighbor classifier based on self-organizing incremental neural network," *Neural Netw.*, vol. 21, no. 10, pp. 1537–1547, Dec. 2008.
[30] H. Zhang, X. Xiao, and O. Hasegawa, "A load-balancing self-organizing incremental neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 6, pp. 1096–1105, Jun. 2014.
[31] Y. Xu, S. Furao, O. Hasegawa, and J. Zhao, "An online incremental learning vector quantization," in *Proc. 13th Pacific-Asia Conf. Knowl. Discovery Data Mining*, Bangkok, Thailand, Apr. 2009, pp. 1046–1053.
[32] Y. Xu, F. Shen, and J. Zhao, "An incremental learning vector quantization algorithm for pattern classification," *Neural Comput. Appl.*, vol. 21, no. 6, pp. 1205–1215, Sep. 2012.
[33] S. Furao, A. Sudo, and O. Hasegawa, "An online incremental learning pattern-based reasoning system," *Neural Netw.*, vol. 23, no. 1, pp. 135–143, Jan. 2010.
[34] E. Sesa-Nogueras and M. Faundez-Zanuy, "Biometric recognition using online uppercase handwritten text," *Pattern Recognit.*, vol. 45, no. 1, pp. 128–144, Jan. 2012.
[35] J. Liu, Y. Yang, I. Saleemi, and M. Shah, "Learning semantic features for action recognition via diffusion maps," *Comput. Vis. Image Understand.*, vol. 116, no. 3, pp. 361–377, Mar. 2012.
[36] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. New York, NY, USA: Wiley, 1949.
[37] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: An ensemble method for drifting concepts," *J. Mach. Learn. Res.*, vol. 8, pp. 2755–2790, Dec. 2007.
[38] R. Elwell and R. Polikar, "Incremental learning in nonstationary environments with controlled forgetting," in *Proc. Int. Joint Conf. Neural Netw.*, Atlanta, GA, USA, Jun. 2009, pp. 771–778.
[39] H. He, S. Chen, K. Li, and X. Xu, "Incremental learning from stream data," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 1901–1914, Dec. 2011.
[40] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2283–2301, Oct. 2013.
[41] C. W. Abraham and A. Robins, "Memory retention—The synaptic stability versus plasticity dilemma," *Trends Neurosci.*, vol. 28, no. 2, pp. 73–78, Feb. 2005.
[42] M. Mermillod, A. Bugaiska, and P. Bonin, "The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects," *Frontiers Psychol.*, vol. 4, Aug. 2013, Art. ID 504.
[43] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view RGB-D object dataset," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 2011, pp. 1817–1824.

Youlu Xing received the B.S. degree in computer science and technology from the East China University of Science and Technology, Shanghai, China, in 2009, and the M.S. degree in software engineering from Nanjing University, Nanjing, China, in 2011, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology.

His current research interests include artificial intelligence, physiology, and philosophy.

Furao Shen received the B.Sc. and M.Sc. degrees in mathematics from Nanjing University, Nanjing, China, in 1995 and 1998, respectively, and the Ph.D. degree from the Tokyo Institute of Technology, Tokyo, Japan, in 2006.

He is currently a Full Professor of Computer Science and Technology with Nanjing University. His current research interests include neural computing and robotic intelligence.

Jinxi Zhao received the Ph.D. degree in mathematics from Nanjing University, Nanjing, China, in 1987.

He has been with Nanjing University since 1977, where he was a Lecturer and an Associate Professor, in 1986 and 1988, respectively, and a Full Professor of Computer Science and Technology since 1999. His current research interests include intelligent computing, numerical linear algebra, the theory and algorithm in computation of matrix, and ill-conditioned system.