Neurocomputing 421 (2021) 66–83

Contents lists available at ScienceDirect

Neurocomputing

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

Autonomous cognition development with lifelong learning: A self-organizing and reflecting cognitive network

Ke Huang, Xin Ma *, Rui Song, Xuewen Rong, Yibin Li

Center for Robotics, School of Control Science and Engineering, Shandong University, Jinan 250061, China

ARTICLE INFO

Article history:
Received 14 December 2019
Revised 26 August 2020
Accepted 17 September 2020
Available online 28 September 2020
Communicated by Zidong Wang

Keywords:
Cognitive development
Object concepts
Self-organizing incremental neural network
CFS clustering
Lifelong learning

ABSTRACT

Lifelong learning is still a great challenge for cognitive robots since the continuous streaming data they encounter is usually enormous and no-stationary. Traditional cognitive methods suffer from large storage and computation consumption in this situation. Therefore, we propose a self-organizing and reflecting cognitive network (SORCN) to realize robotic lifelong cognitive development through incremental learning and regular reflecting. The network integrates a self-organizing incremental neural network (SOINN) with a modified CFS clustering algorithm. SOINN develops concise object concepts to alleviate storage consumption. Moreover, we modify SOINN by an efficient competitive method based on reflection results to reduce the learning computation. The modified CFS clustering algorithm is designed for reflecting knowledge learned by SOINN periodically. It improves the traditional CFS as a three-step clustering method including clustering, merging and splitting. Specifically, an autonomous center selection strategy is employed for CFS to cater to online learning. Moreover, a series of cluster merging and splitting strategies are proposed to enable CFS to cluster data incrementally and improve its clustering effect. Additionally, the reflection results are utilized to adjust the topological structure of SOINN and guide the future learning. Experimental results demonstrate that SORCN can achieve better learning effectiveness and efficiency over several state-of-art algorithms.

1. Introduction

Cognitive development plays a significant role in the realization of robots to display intelligent behaviors like humans, such as perception, attention, reasoning, adaptivity and action [1,2]. As cognitive robots often work in more complex and dynamic environment, they must learn continuous streaming data online, transform perceptive features into concepts, remember what they have learned, and retrieve them properly according to the emerging situations [3]. In addition, these skills can serve as a scaffolding to support higher level cognitive abilities rather than just be performed in a single task [4]. Hence, it is necessary for robots to have the ability to develop cognition progressively over their entire lifetime, which is referred to as lifelong learning [5,6].

However, developing cognition regarding lifelong learning is still a huge challenge for robots due to the continuous streaming data they may encounter. On the one hand, the sequential and uncertain inputs require robots to detect novel instances and classify familiar objects in real time [7–10]. On the other hand, the large-scale data can give rise to enormous storage and computation consumption [9,11]. Most cognitive models still suffer from the second problems when confronted with steaming data [12–14]. Although some methods are efficient due to their convenient online updating [15,16], they cannot solve the storage problem. Some methods even turn out to be training an incremental classifier for object recognition rather than developing original object representations [17]. Fortunately, various self-organizing incremental neural networks (SOINN) [18–23] have been proved to not only process streaming data appropriately according to Hebbian learning, but also generate concise representations to reduce storage due to its generalization ability. Therefore, SOINN might be a promising method to realize lifelong learning.

However, SOINN also has some shortcomings. Its self-organizing results rely heavily on activation conditions or similarity thresholds [21]. Although some improved methods can adjust similarity thresholds during the learning process, they may be affected by the input sequence of data [24,25]. We have proposed an audio-visual integrative cognitive architecture [22] and an interactive cognitive model [26] to adjust the similarity threshold of nodes dynamically, but these two methods utilized other modal information as auxiliary direction for the adjustment. In addition,

* Corresponding author.
E-mail address: maxin@sdu.edu.cn (X. Ma).

https://doi.org/10.1016/j.neucom.2020.09.027
0925-2312/

the competitive learning principle of SOINN requires the network to traverse all nodes to search the best matching node for a new input. Thus, it would still confront huge computational loads and efficiency declines as nodes increase.

Kolbs experiential learning theory (ELT) [27] describes the human learning process as an ongoing cycle. As shown in Fig. 1 [28,29], humans firstly obtain concrete experience from perception. Then they reflect learned knowledge and generate abstract concepts, which can be used for future experiments or applications. Particularly, reflection can integrate new experiences into existing knowledge structures, which contributes to promoting cognitive development and lifelong learning [30]. Inspired by this theory, we propose a new viewpoint of cognitive development for robots to realize lifelong learning. Robots can reflect the learned knowledge in the intervals of learning like humans to understand relationships across data. Therefore, the research problem becomes how to design a cognitive developmental model that introduces an appropriate reflection algorithm into SOINN to reduce storage and computation consumption as well as address the effect of similarity threshold for SOINN to realize robots' lifelong learning. Given that object concepts are often represented as categories [31], an effective clustering algorithm can be used to simulate human reflection.

In order to solve this problem, this paper presents a Self-Organizing and Reflecting Cognitive Network (SORCN), which integrates a SOINN with a CFS clustering algorithm [32]. It develops object concepts through incremental learning in two stages and regular reflection. The CFS algorithm for reflection is modified by an autonomous center selection strategy and several merging as well as splitting strategies to improve its clustering effect and realize its incremental clustering. A new competitive method for SOINN is designed based on the reflecting result to reduce computation and enhance its learning speed. Although similar learning methods combining SOINN with various cluster algorithms, known as DenSOINN [11] and LD-SOINN [33], have been developed, they all adopt an online learning and offline clustering way. In contrast, our method executes a 'learning-reflecting' loop over the development process. Reflecting results can be used to adjust the network's topological connection and nodes' similarity threshold, which contributes to the future learning of the network.

The contributions proposed in this work are the following: (1) An autonomous cognitive network is established by integrating an incremental self-organizing neural network and a CFS clustering algorithm, which simulates human learning-reflection process to realize lifelong learning; (2) An efficient competitive learning strategy for SOINN is designed based on reflection results, which can quickly find the winner for each input without traversing all nodes and reduce computation consumption; (3) A three-step CFS clustering algorithm is proposed as reflection, which contains a novel merging strategy based on intra-class topological reconstruction and a new splitting strategy based on intra-class center reselection. The major advantages are obvious. It can realize incremental clustering for CFS and conduct the learning of SOINN to promote the cognitive network's development.

The rest of this paper is organized as follows: Section 2 reviews related works; Section 3 illustrates the details of SORCN; Section 4 presents the experimental results; Section 5 concludes the paper.

2. Related work

2.1. Cognitive architecture of lifelong learning

Lifelong learning for robots, which was originally proposed by Thrun et.al [5,6], intends to learn and accumulate knowledge continuously. Compared with traditional cognitive models [34–39], it provides a long-term view and a knowledge transfer perspective that prompt robots to become more and more knowledgeable and effective at learning [40].

There has been great interest in the study of developing robots' cognition with lifelong learning. One common approach is designing an incremental classifier to process continuous streaming data. Most researches adopt a probabilistic model to recognize familiar objects and learn new classes. For example, Camoriano et al. [17] proposed the Regularized Least Squares for Classification (RLSC) algorithm, but it focused on object recognition rather than developing object concepts to improve robots' cognition. Two other models based on Bayes classifier [15] or multimodal hierarchical Dirichlet process (MHDP) [31] both relied on a human trainer to tell the objects' name. Liu et al. [16] developed an efficient online dictionary learning algorithm based on second-order Taylor expansion and sparse coding for heterogeneous multi-modal tasks. However, it needs samples, labels and modalities to train the classifier, whereas label information may not be available in lifelong learning. Eriksen et.al [41] proposed a deep neural network for lifelong

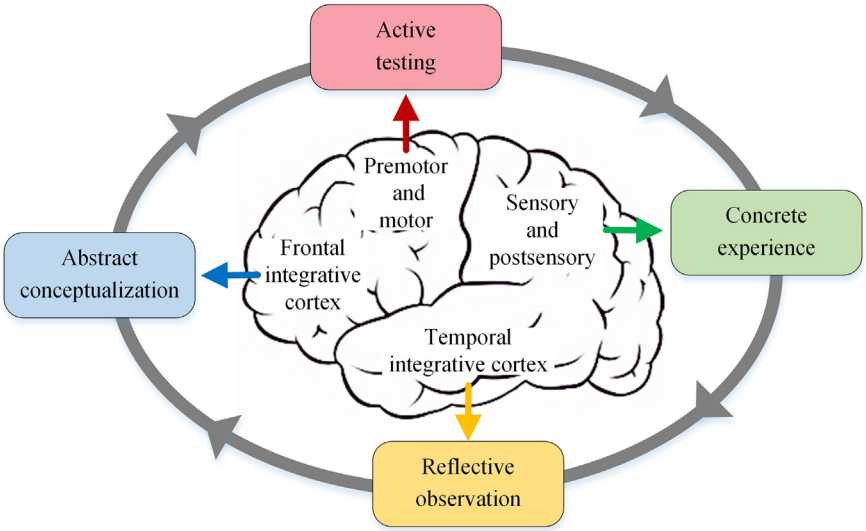


Fig. 1. Kolb's learning cycle [28] and regions of the cerebral cortex [29].

learning. Nevertheless, it should be retrained when capturing new object images.

Another approach used intrinsic motivation to drive lifelong learning. Ammer et al. [12] proposed a lifelong RL framework to develop robots multi-task or multi-view learning ability. Glover et al. [13] leveraged a distributed semi-Markov decision model to realize lifelong affordance learning. However, the state and action spaces in this kind of model would consume large amounts of memory. Multilevel Darwinist Brain (MDB) cognitive architecture [4,14,42,43] mainly focused on fulfilling the robots' goals in a developmental view. As the incrementally acquired knowledge is stored in STM and LTM, MDB is still confronted with storage and computation consumption problems.

Instead of storing original representations, some other methods based on various self-organizing incremental neural networks developed prototypes for several similar objects by neural nodes to reduce storage demand. The incremental learning way of SOINN is suitable for processing streaming data [44,45]. However, some networks use a fixed similarity threshold, such as Gamma-GWR [21,46], or the fixed ratio adopted by PCN [23]. They may achieve a stable learning result, but not work for all features. GAM [24] and AKDESOINN [25] can adjust the similarity threshold of each node. Nevertheless, the adjustment of similarity thresholds completely relies on the spatial positions of nodes. Thus, different learning sequences of the same dataset may generate different similarity thresholds. A more serious problem for all SOINNs is that they should traverse all nodes for each input to get the winner. This would cause large computation consumption. Our previous work [22] designed a hierarchical cognitive developmental architecture based on SOINN and proposed a dynamically adjustable similarity threshold strategy and a top-down response strategy. Another research of ours [26] integrated SOINN with interactive reinforcement learning to adjust learned knowledge under human guidance. They both utilized additional information to conduct the learning, but could not autonomously generate guidance signals under a single model.

Compared with above methods, the proposed cognitive network can not only alleviate storage due to its generalization ability, but also reduce computation consumption using its efficient competitive learning strategy. Moreover, our method can perform an intra-class topological construction during each reflecting process, which can provide a reliable guidance to the adjustment of the similarity threshold for each node.

2.2. Clustering algorithm based on CFS

CFS proposed by Rodriguez et.al [32] has been proved an effective and efficient clustering algorithm for data with different shape and distribution [8]. The core idea of this algorithm is selecting cluster centers according to two conditions: higher density than their neighbors and large distance from points with higher densities.

The local density of point i can be calculated in two ways. One is using cut-off kernel which counts the number of points whose distance from point i is less than the cutoff distance d_c as shown in (1).

$$\rho_i = \sum_{i \neq j} \chi(d_{ij} - d_c), \tag{1}$$

where d_{ij} represents the Euclidean distance between points i and j. d_c is calculated by 2% of the distances between nodes of buffer in ascending order. $\chi(x) = 1$, if $x > 0$. Otherwise, $\chi(x) = 0$. Another is utilizing Gaussian kernel where d_c acts as its density estimation parameter as shown in (2). The density of point i is estimated by its neighbors within radius d_c.

$$\rho_i = \sum_{i \neq j} e^{-(d_{ij}/d_c)^2}. \tag{2}$$

The distance of point i is defined as the minimum distance between i and any other points with higher density as shown in (3). If the local density of i is the highest, it is calculated by the maximum distance from all other points.

$$\delta_i = \begin{cases} \min\limits_{j:\rho_j > \rho}(d_{ij}) & if\ \exists j s.t.\ \rho_j > \rho_i \\ \max\limits_{j}(d_{ij}) & otherwise \end{cases}. \tag{3}$$

In order to find a proper number of clusters, CFS sorts the gamma $\gamma_i = \rho_i \cdot \delta_i$ of each point in decreasing order and selects the points with big gamma as cluster centers. Finally, the remaining points are assigned to the same cluster as their nearest neighbor with higher density.

In this paper, we introduce CFS into SOINN to simulate human reflection. Therefore, we investigate many researches of CFS to find the joint point of this integration. As CFS still has some drawbacks, researches mainly focus on how to improve this algorithm. Firstly, the cluster result is sensitive to the value of cutoff distance. One solution is finding the optimal cutoff distance by minimizing the estimation entropy of density [47] or Gaussian distance [48]. However, this can lead to enormous computation. Another approach tries to avoid the calculation of traditional cutoff distance, such as redefining the parameters of CFS [49,51] or selecting cluster validity indices as objective functions for intrinsic clustering measurement [52]. Secondly, the cluster centers are manually selected from a decision graph, which is unsuitable for the online learning way of SOINN. To track this problem, some researches proposed two constraint conditions of centers based on distance and density to realize autonomous selection [47,53–55]. Other methods are in view of gamma, such as selecting a bump point of decreasing order gammas [50] or a weighted average [56] as the threshold. Thirdly, CFS may separate one cluster into several parts because it contains multiple density peaks. One common solution is designing a proper merging strategy to integrate these adjacent parts into a global cluster [57,58]. Additionally, Zhao et al. [59] identified multiple representatives for each cluster to replace the one-to-one mapping between center and category.

The last but most important drawback, which hampers the integration between SOINN and CFS, is that CFS is designed for static data and cannot cluster streaming data in an incremental way. Only a few researches are offered for this problem. Zhang et al. [57] firstly clustered new inputs by traditional CFS, and then treated the cluster results as new clusters or merged them with known clusters according to their centers' distance. Zhao et al. [59] proposed another method named Enhanced ICFSMR to assign each new arrival to a proper cluster with multiple representations. However, new clusters were just generated by its one-time cluster splitting and merging strategy.

Our modified CFS algorithm contains three clustering steps inspired by [50,59]. However, there are many differences with these two approaches. We adopt the autonomous center selection strategy proposed by [54] to fit the online learning way of SOINN. A series of new merging and splitting strategies are designed to adjust one-time cluster results dynamically and realize incremental clustering. In contrast to [57], we put forward a mean neighbor distance for each cluster as the merging condition rather than a predefined cutoff distance.

3. Proposed method

3.1. Problem formulation

This paper aims to establish a cognitive developmental model based on SOINN to realize robot's lifelong learning. Suppose that a robot would confront with a streaming data set $X = \{x_T\},\ T = 1, 2, 3, \ldots$, where x_T is a d-dimensional vector input at timestamp T during the lifelong learning. The cognitive model SORCN should traverse the whole network V for competitive learning when meeting each x_T. The problem is that SOINN has no ability to handle large-scale data as its computation consumption would explode with the growth of nodes. Moreover, the cognitive results heavily rely on nodes' similarity thresholds, but they may be not reliable as only developed by local information without global nodes distribution.

Therefore, a modified CFS clustering algorithm is introduced to reduce the computational consumption and improve the development of SORCN. The rationality of integrating SOINN with CFS for lifelong learning is that SOINN has great generalization ability to reduce storage of streaming data and CFS is good at finding appropriative clusters for learned knowledge. They can compensate for each other. The clustering results of CFS can not only help to adapt the similarity threshold for each node of SOINN, but also contribute to reducing the computation of competitive learning by offering reliable category information. In turn, SOINN provides instances incrementally to support CFS cluster in lifelong learning.

The cognitive development of SORCN in lifelong learning is divided into two parts: online learning and regular reflecting. The former learns each x_T by updating its best matching node or creating a new node that is also recorded in a buffer B. SORCN can obtain new knowledge and generate category-based object concepts with subgraphs G according to its self-organizing ability. The latter adjusts the subgraphs G by a modified CFS algorithm to promote the development of SORCN. Assume that SORCN has formed a subgraph set G^{t-1} after $t-1$ turns of reflection. At t time reflection, CFS is used to cluster the unknown class nodes in B for the result B^t. The reflection aims to integrate the result B^t into the subgraph set G^{t-1} so that SORCN can develop a new subgraph set $G^t = G^{t-1} \cup B^t$. According to the above analysis, there are three major challenges for SORCN to realize lifelong learning. The first one is how to integrate the result B^t into the previous subgraph set G^{t-1} for the current subgraph set G^t. The second one is how to reduce the computational consumption of SOINN's competitive learning according to the results of each reflection so that SORCN can process enormous data in lifelong learning. The final one is how to adjust the similarity threshold of nodes in SOINN in the reflecting process to promote the development of SORCN.

3.2. Overview of SORCN

As shown in Fig. 2, SORCN is a single-layer self-organizing incremental neural network combined with a modified CFS clustering algorithm. It starts from an empty network and then gradually develops nodes according to Hebbian learning as streaming data arrives. SORCN can recognize familiar objects as known classes and update their matching nodes. When detecting a novel input, it creates a new node to learn the knowledge. Simultaneously, the new node is also recorded in a buffer. When the buffer is full, SORCN executes an effective CFS clustering algorithm as a reflection to cluster these novel nodes according to their density distribution. Then, the cluster results are used to update SORCN, such as merging with known clusters or creating new classes. After reflection, the buffer is cleared and the network begins a new learning iteration. Algorithm 1 summarizes the learning process of SORCN.

Algorithm 1 SORCN

```
1: Input an object to SORCN.
2: Execute completive learning to judge the class of the
   object.
3: if the object belongs to a known class then
4:       Output the class as the recognized object concept.
5: else
6:       Create a new node to learn the knowledge.
7:       Record the new node into a buffer.
8:       if the buffer is full then
9:             Cluster the nodes in the buffer.
10:            Update the network of SORCN by the cluster
   result in the buffer.
11:            Clear the buffer.
12:      end if
13: end if
14: Go to Step 1 to process the next input.
```

SORCN continuously repeats "learning-reflecting" turns over the lifelong learning process. SORCN firstly undergoes the initial learning phase due to its little knowledge. We modified the similarity threshold strategy to balance its adjustability and the robustness for the data's input sequence. SORCN adjusts its learning result into appropriate clusters by the modified CFS clustering algorithm in the reflection phase. In order to apply CFS into the online learning task, an autonomous center selection strategy is utilized to avoid manual setting. Moreover, a series of cluster merging and splitting strategies are designed to integrate new clusters in buffer with known classes of SORCN, which can not only realize the incremental clustering, but also adjust the similarity threshold of nodes to promote the development of SORCN. In the

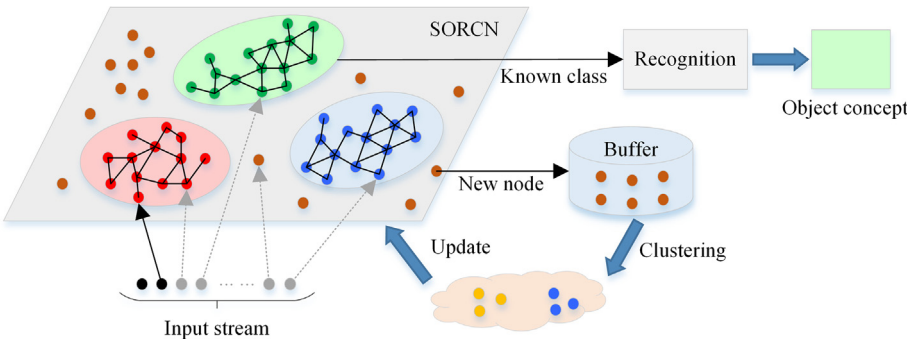


Fig. 2. Overview of SORCN.

developmental learning phase, a new competitive learning method is proposed based on the reflection result to reduce the computation consumption of SORCN, which allows SORCN to realize lifelong learning. Table 1 summarizes the definitions of all variables, parameters and symbols that appear in the following sections.

3.3. Online learning

The framework of SORCN is based on the adjusted SOINN [20]. In contrast to its traditional single learning process, SORCN contains two phases: initial learning phase and developmental learning phase. They are assigned different competitive ways due to different levels of knowledge.

3.3.1. Initial learning phase

In the initial learning phase, the SORCN with little experience adopts the same competitive way of the adjusted SOINN: the network should traverse all nodes to find the best matching node for input. However, we modified the adjusted SOINN in two aspects. Firstly, its adaptive similarity threshold combines fixed ratio and space position methods as shown in (4). Each new node has an initial similarity threshold represented by a discrepancy rate $\varepsilon$of its weight w_i. This can protect of being affected by data's input sequence. When node i has neighbors, its similarity threshold is updated by its longest connection so that TH_i can dynamically adjust according to data distribution. Although the fixed ratio may cause more nodes, it can provide more precise data distribution for the following reflection algorithm in Section 3.4. Moreover, this disadvantage would be overcome in reflection phase. Secondly, in order to learn all knowledge, we do not consider noise and delete the steps of removing isolated nodes like PCN [23]. The reason is that removing nodes may cause the network forget some useful representations.

$$TH_i = \begin{cases} \varepsilon \cdot \|w_i\|, & N_i = \varnothing \\ \max\limits_{j \in N_i}(w_i, w_j), & N_i \neq \varnothing \end{cases} \tag{4}$$

The flowchart of the initial learning phase is shown in Fig. 3. When an input x_T arrives, the network creates a new node i by $w_i = x_T$ directly if the number of nodes is less than 2. Otherwise, SORCN starts competitive learning among all nodes by measuring their Euclidean distances to the input to find the winner b and the second winner s. When $\|x_T - w_b\| < TH_b$ and $\|x_T - w_s\| < TH_s$, b is activated. This means the input can be represented by the winner and s is also similar to b. Thus, SORCN

Table 1
Definitions of variables, parameters and symbols.

Element	Meaning
x_T	Input at T timestamp
w_i	Weight of node i
TH_i	Similarity of node i
N_i	Neighbors of node i
ξ_i	Learning rate of node i
n_i	Instance number of node i
V	Set of nodes in SORCN
B	Buffer
G^{t-1}	Set of subgraphs in SORCN after $t-1$ time reflection
g_k^{t-1}	The kth subgraph
C_k^{t-1}	Set of centers in g_k^{t-1}
c_{kl}^{t-1}	the lth center in C_k^{t-1}
B_p^t	New cluster in the buffer at t time reflection
$Merge_p$	Set of subgraphs merged with B_p^t
$\rho_i, \delta_i, \gamma_i$	Local density, distance and gamma of node i

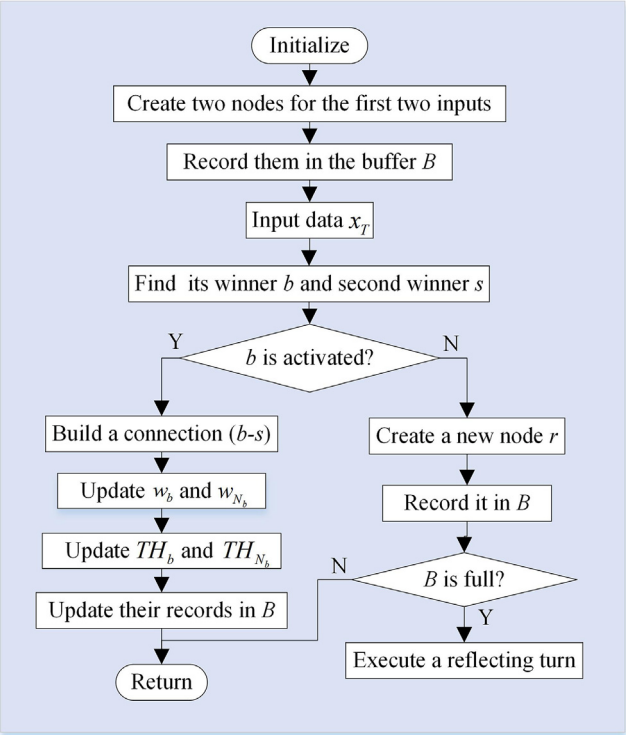


Fig. 3. Flowchart of the initial learning process.

increases the instance number n_b of the winner by 1 and builds a topological connection $(b-s)$ between two nodes. Then, the weights of b and its neighbors N_b are updated by (5) and (6).

$$w_b' = w_b + \xi_b \cdot (w_b - x_T), \tag{5}$$

$$w_{N_b}' = w_{N_b} + \xi_{N_b} \cdot (w_{N_b} - x_T), \tag{6}$$

where $\xi_b = \frac{1}{n_b}$ and $\xi_{N_b} = \frac{1}{100 n_b}$ are the learning rate of node b and its neighbors as used in the original SOINN [18]. The decreased learning rates allow the weight of nodes to be stable with the increase of activation times. ξ_{N_b} is less than ξ_b to prevent the knowledge of neighbors from being seriously affected and avoid catastrophic forgetting. Therefore, the similarity thresholds of b and its neighbors N_b are also updated by (4). Otherwise, a new node is created to learn the novel input.

A buffer B is designed to record new created nodes for future reflection. If one node is updated, its record in the buffer is also changed accordingly. When the buffer is full, SORCN begins to reflect the learned knowledge by the modified CFS clustering algorithm to establish initial clusters, and the details are illustrated in Section 3.4. Then, it turns into the developmental learning phase. Algorithm 2 elaborates the details of the initial learning phase.

Algorithm 2 Initial learning phase

Input: New arriving object x_T.
1: Initialize the SORCN by an empty node dictionary $V = \{\}$.
2: Input an object x_T to SORCN.
3: **if** the number of node in SORCN is less than 2 **then**
4: Add a new node i: $V = V \cup \{i\}$, $w_i = x_T$, $n_i = 1$, $TH_i = \varepsilon \cdot \|w_i\|$.
5: Record node i in the buffer: $B = B \cup \{i\}$.
6: **else**
7: Find the first two nearest nodes b and s:

$b = \arg\min_{i \in V} \|x_T - w_i\|$, $s = \arg\min_{i \in V/\{b\}} \|x_T - w_i\|$.
8: **if** $\|x_T - w_b\| < TH_b$ and $\|x_T - w_s\| < TH_s$ **then**
9: $n_b = n_b + 1$ and build a connection $(b - s)$.
10: Update TH_b and TH_{N_b} by (4).
11: Update w_b and w_{N_b} by (5) and (6).
12: Update the records of b and its neighbors in the buffer B.
13: **else**
14: Add a new node r: $V = V \cup \{r\}$, $w_r = x_T$, $n_r = 1$, $TH_r = \varepsilon \cdot \|w_r\|$.
15: Record node r in the buffer: $B = B \cup \{r\}$.
16: **end if**
17: **end if**
18: **if** B is full **then**
19: Execute a reflection process as shown in **Algorithm 4–6**.
20: Turn into a developmental learning phase.
21: **else**
22: Go to step 2 to process the next object.
23: **end if**

3.3.2. Developmental learning phase

In the developmental learning phase, SORCN already has a certain degree of knowledge and category concepts due to previous initial learning and reflection. Therefore, it employs a new competitive way based on reflection results to avoid traversing all nodes and reduce computation compared with other self-organizing networks [18,24]. The flowchart of the developmental learning phase is illustrated in Fig. 4. Suppose SORCN has experienced $t-1$ turns of reflection. In the current learning time, SORCN contains m known subgraphs $G^{t-1} = \{g_1^{t-1}, g_2^{t-1}, \ldots, g_m^{t-1}\}$ and g_k^{t-1} possesses l centers $C_k^{t-1} = \{c_{k1}^{t-1}, c_{k2}^{t-1}, \ldots, c_{kl}^{t-1}\}$, where $k = \{1, 2, \ldots, m\}$. The remaining unknown class nodes in the network are also recorded in buffer B.

Firstly, SORCN computes the distances between the new arrival and all cluster centers $\{C_k^{t-1}\}_{k=1}^{m}$ rather than all nodes. The closer the input is to the center of one subgraph, the more likely it is to belong to that class. Each subgraph is sorted from near to far with the input as G_{sorted}^{t-1}. Secondly, SORCN begins to traverse each subgraph $g_{k'}^{t-1}$ in order of G_{sorted}^{t-1}. It selects the nearest subgraph and finds two nearest nodes b and s for the input in this subgraph. If the activation condition mentioned in the initial learning phase is satisfied, the input is recognized as the concept of $g_{k'}^{t-1}$. Then, SORCN updates node b and its neighbors as well as their topological connections. Otherwise, SORCN picks the next nearest subgraph in G_{sorted}^{t-1} to continue the activation test until finding the final winner. If none of subgraphs is close enough to the input, SOCRN selects another two nearest nodes b' and s' in the buffer. If b' is also not activated, the input is treated as a novel class and a new node is created to obtain this knowledge. Meanwhile, the new node is also recorded in the buffer. Then, SORCN continues to learn the next data. When the buffer is full, it turns into a new reflecting turn. Algorithm 3 illustrates the developmental learning phase.

Algorithm 3 Developmental learning phase

Input: New arriving object x_T.
1: Input an object x_T to SORCN.
2: Calculate the distances between x_T and all cluster centers $\{C_k^{t-1}\}_{k=1}^{m}$.
3: Sort all subgraphs according to the distances as G_{sorted}^{t-1}.
4: **for** $k' = 1, 2, \ldots, m$ **do**
5: Find two nearest nodes b and s for the input in $g_{k'}^{t-1}$.
6: **if** $\|x_T - w_b\| < TH_b$ and $\|x_T - w_s\| < TH_s$ **then**
7: Update node b and its neighbors as step 8 in **Algorithm 2**.
8: Break.
9: **end if**
10: **end for**
11: **if** no node in all subgraphs satisfies the activation condition **then**
12: Find other two nearest nodes b' and s' for the input in B .
13: **if** $\|x_T - w_{b'}\| < TH_{b'}$ and $\|x_T - w_{s'}\| < TH_{s'}$ **then**
14: Update node b' and its neighbors as step 8 in **Algorithm 2**.
15: **else**
16: Add a new node as step 13 in **Algorithm 2**.
17: **end if**
18: **end if**
19: **if** B is full **then**
20: Execute a reflection process as shown in **Algorithm 4–6**.
21: **end if**
22: Go to step 1 to process the next object.

With the proposed competitive learning way, SORCN just needs to compute distances between the input and all cluster centers as well as the nodes of the nearest subgraph in the fastest situation. Even in the most pessimistic situation where the network traverses all subgraphs and the buffer, SORCN just takes extra but not too much time in the distance calculation between inputs and centers compared with traditional competitive method. Therefore, our competitive learning way can significantly reduce computation effort and accelerate the learning speed. It enables SORCN the ability to process large-scale data and contributes to realizing lifelong learning.

3.4. Reflecting process

During the reflecting process, SORCN is occupied in analyzing the internal relations of learned knowledge and generating appropriate category concepts by the modified CFS clustering algorithm. Fig. 5 illustrates the flow of the modified CFS clustering algorithm. The algorithm consists of three steps: clustering, merging and splitting. They can help to improve the clustering effectiveness of CFS and allow CFS to cluster data incrementally. Moreover, the topological connections of the network are adjusted according to cluster results, which contributes to developing more reliable similarity thresholds of nodes. The flowchart of the reflecting process is shown in Fig. 6.

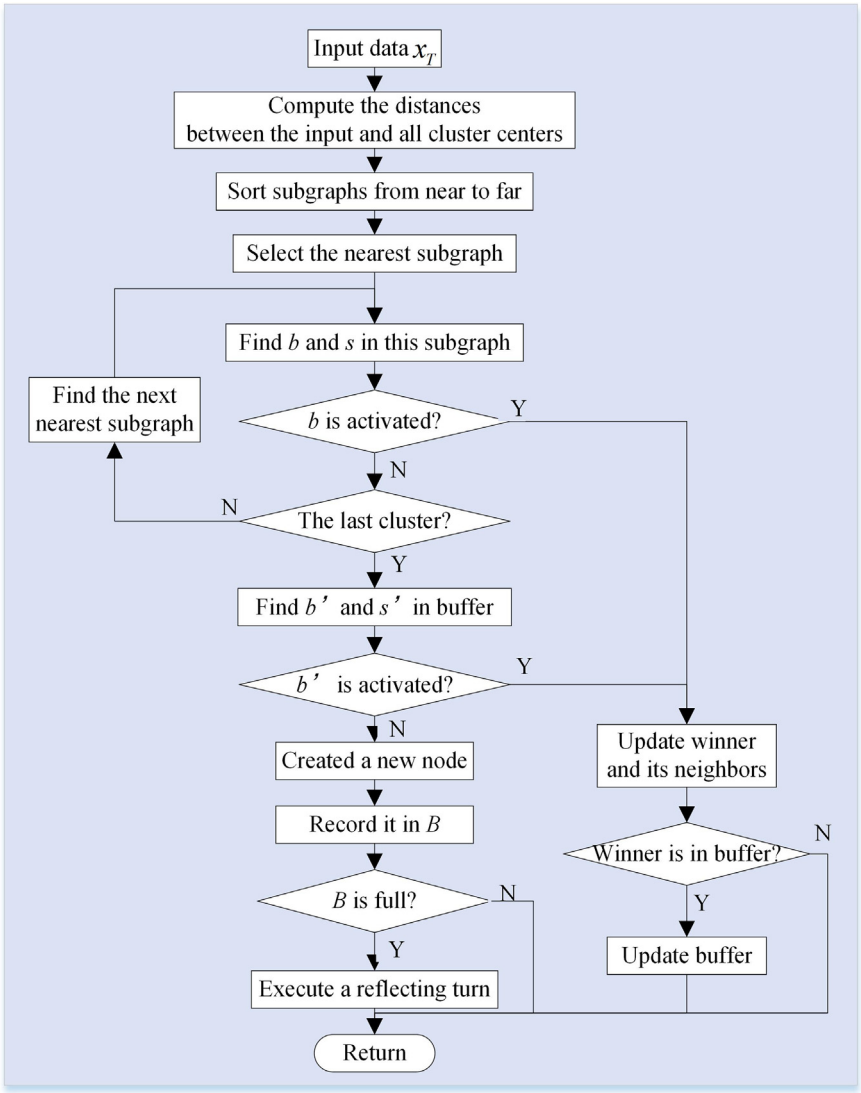


Fig. 4. The flowchart of the developmental learning phase.

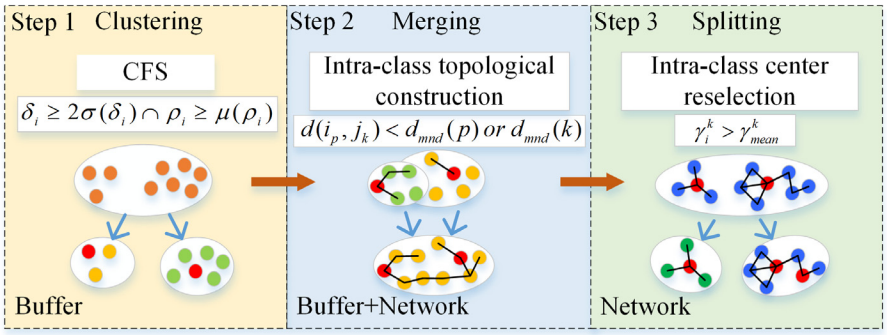


Fig. 5. Flow of the modified CFS clustering algorithm. Red points represent cluster centers.

3.4.1. CFS clustering

In the clustering step, a CFS algorithm [32] is utilized to cluster the new nodes in the buffer. We calculate the local density for node in the buffer by a Gaussian kernel as shown in (2) rather than a cut-off kernel [60]. The reason is that it can prevent different nodes from having the same local density value. Although CFS is mainly applied for original instances, this density formula is also suitable for neural nodes that are equal to prototypes of several objects. We do not consider how many instances the node has as utilized in [11], because this method would lead to CFS missing the true center with few instances but selecting a border node activated many times as a center. This mistake may assign the true center and its neighboring nodes to a wrong cluster.

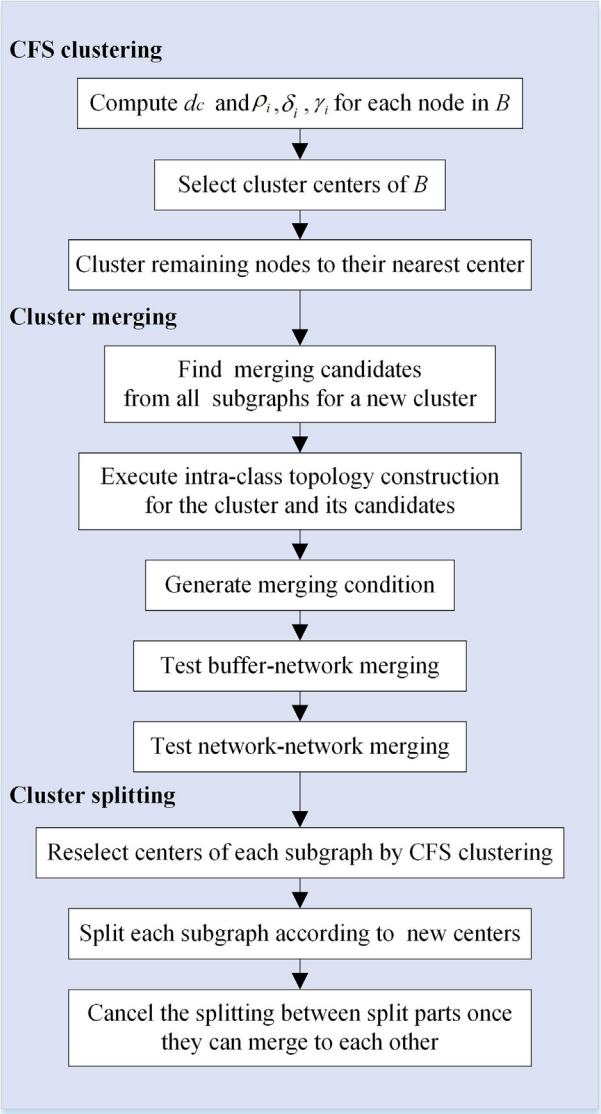


Fig. 6. The flowchart of the reflection process.

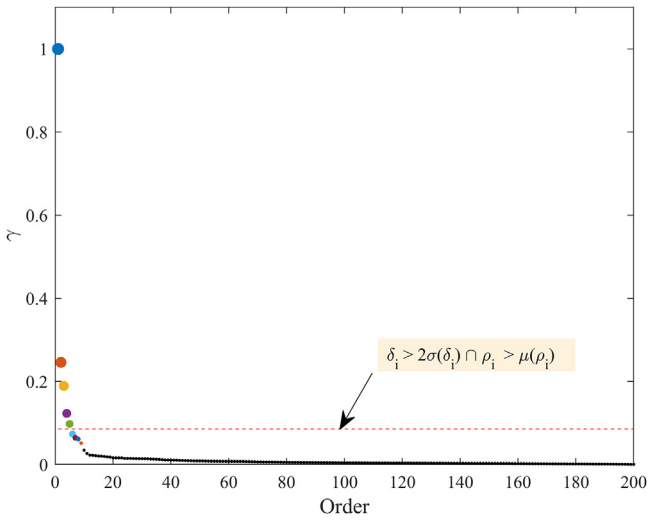


Fig. 7. Decision graph of the autonomous center selection strategy.

In the traditional method, the number of centers is selected manually. However, this method is not suitable for online learning. We adopt an autonomous center selection strategy as proposed by [54] to accommodate the incremental learning. It contains two constraint conditions derived from the definition of center in CFS [32] that the expected cluster center must have large distance from other cluster centers and high density as shown in (7):

$$\delta_i \geqslant 2\sigma(\delta_i) \cap \rho_i \geqslant \mu(\rho_i), \tag{7}$$

where $\sigma(\delta_i)$ represents the standard deviation of all distances between each node, and $\mu(\rho_i)$ is the mean of all local densities. One novel point is that we also consider the gamma value of nodes. The gamma list for all nodes is sorted as $\gamma' = \{\gamma'_i\}_{i=1}^{B_{size}}$ in decreasing order for clustering decision. B_{size} is the size of buffer, and stands for the node order in the sorted list. We apply the conditions to nodes in the order of $\{\gamma_o\}_{o=1}^{B_{size}}$ until one node does not meet them rather than traverses all nodes in the buffer to reduce the computations, as shown in Fig. 7. Thus, at t turn reflecting, the nodes satisfying two conditions simultaneously are selected as centers $C_B^t = \{c_{B_1}^t, c_{B_2}^t, \ldots, c_{B_p}^t\}$. The remaining nodes are assigned to corresponding clusters as the way in the traditional CFS. Finally, the CFS clustering algorithm clusters all nodes in the buffer into p clusters $B^t = \{B_1^t, B_2^t, \ldots, B_p^t\}$. Moreover, the distance matrix D_p between nodes in each cluster can be obtained according to d_{ij}. They are reserved for further operations to avoid repetitive computation. Algorithm 4 in Appendix A illustrates the details of the CFS clustering step.

3.4.2. Cluster merging

The cluster merging step aims to merge the new clusters in the buffer with the known subgraphs in SORCN. Therefore, it is the crucial step to realize the integration of SOINN and CFS. A new cluster merging strategy based on intra-class topology construction is proposed in this section (see Fig. 5). This strategy contains two parts: buffer-network merging and network-network merging. In the buffer-network merging, each new cluster B_p^t in the buffer B^t is tested to determine whether to integrate with known subgraphs $\{g_k^{t-1}\}_{k=1}^m$ in the network G^{t-1} or not. Then, the network-network merging integrates the subgraphs close to each other in SORCN. As the merging actions of them are same, we mainly introduce the buffer-network merging phase and Algorithm 5 in Appendix B illustrates the details.

Firstly, we find the merging candidates of B_p^t in SORCN by computing the center distances between $c_{B_p}^t$ and $\{C_k^{t-1}\}_{k=1}^m$. Considering that one cluster may overlap many subgraphs, we select the first three nearest subgraphs $\{g_{bu}^{t-1}\}_{u=1}^3$ ($bu \in \{1, 2, \ldots, m\}$) for merging. Secondly, an intra-class topology construction is designed for cluster $h \in \{B_p^t, g_{b1}^{t-1}, g_{b2}^{t-1}, g_{b3}^{t-1}\}$based on a k-nearest neighbors (k-NN) graph to generate a reliable merging condition. We select the four nearest nodes $\{n_v^{i_h}\}_{v=1}^4$ in the cluster for each node i_h as neighbors. Then, the mean neighbor distance of the whole cluster can be obtained as (8).

$$d_{mnd}(h) = \mu\left(\sum_{i=1}^{len(h)}\left(\sum_{v=1}^{4} d(i_h, n_v^{i_h})\right)\right), \tag{8}$$

where $len(\cdot)$ represents the operation of getting the total number of nodes and $d(i_h, n_v^{i_h})$ means the neighbor distance of i_h. These neighbor distances of B_p^t can be directly extracted from the distance matrix D_p. Thus, we just need to compute the intra-class distance matrix D_{bu} of g_{bu}^{t-1}. We select the neighbors of i_h which satisfy $d(i_h, n_v^{i_h}) < d_{mnd}(h)$ and build a connection $(i_h - n_v^{i_h})$ to adjust the

topological structure of cluster h. Their similarity thresholds are also updated by (4). Therefore, the intra-class topology construction can accelerate the development of SORCN and provide reliable guidance for the adjustment of nodes' similarity threshold. Steps 2–11 of Algorithm 5 illustrate the details of intra-class topology construction.

Thirdly, we find the nearest nodes i_p and j_{bu} between B_p^t and g_{bu}^{t-1}. Once the minimum distance $d(i_p, j_{bu})$ is less than their mean neighbor distance as shown in (9), B_p^t is close enough to merge with g_{bu}^{t-1}. Thus, their nearest nodes are connected and g_{bu}^{t-1} is recorded in a merging set $Merge_p$ of B_p^t.

$$d(i_p, j_{bu}) < d_{mnd}(p) \cup d(i_p, j_{bu}) < d_{mnd}(bu). \tag{9}$$

If $Merge_p \neq \varnothing$ after tested all candidates, B_p^t and other subgraphs are merged into the first merging subgraph $g_{b1}^t = g_{b1}^{t-1} \cup Merge_p \cup \{B_p^t\}, C_{b1}^t = C_{b1}^{t-1} \cup \{c_{B_p}^t, C_{bu}^{t-1}\}$, where $g_{bu}^{t-1} \in Merge_p$. Here is the reason: If g_{b1}^{t-1} cannot merge with B_p^t, the other two subgraphs would also not satisfy the merging condition. Meanwhile, its intra-class distance matrix D_{b1} is also updated. If no merging occurs, B_p^t is treated as a new subgraph $G_{merge}^t = G^{t-1} \cup \{B_p^t\}$. Then, SORCN processes another cluster in the buffer. In order to avoid repeatedly merging, the merged subgraphs do not participate in the next buffer-network merging. After all clusters in the buffer are assigned into the network, the buffer is cleared as $B^t = \varnothing$ and SORCN turns into the network-network merging.

A novel point of the proposed merging strategy is that it advocates merging classes but not merging centers. Specifically, it preserves all centers after merging rather than finds an optimal center for the merged cluster by K-mediods as presented in [57]. Each center is treated as a representation for the merged cluster because a single center cannot characterize nonspherical and unbalance data adequately. Hence, it can overcome the shortcoming of CFS: one cluster with multiple density peaks may not be correctly clustered, and improve the accuracy when SORCN searches the best matching cluster for the input.

3.4.3. Cluster splitting

As the streaming inputs might be non-spherical and non-stationary, one-time clustering may not divide buffer nodes accurately. On top of that, different subgraphs may be merged improperly, because a scattered cluster in the buffer overlaps them. Consequently, a new splitting strategy as demonstrated in Fig. 5 is proposed to solve these problems.

This strategy analyzes the inner distribution of each subgraph and re-selects its centers by CFS clustering in Section 3.4.1 to judge whether the subgraph should be split or not. The difference is that the cutoff distance of each subgraph g_k^t is compute based on its intra-class distance matrix D_k which has been obtained in merging step. Moreover, we adopt a new center selection strategy rather than that in the clustering step. We select the first o' values $\{\gamma_o^k\}_{o=1}^{o'}$ in sorted gamma list and calculate their mean values γ_{mean}^k as center threshold. A node is picked out as a center candidate if $\gamma_i^k > \gamma_{mean}^k$ and the selection stops once the splitting condition is not satisfied. This center selection condition is looser than that in Section 3.4.1 and can generate more centers, which contributes to improve accuracy of proposed competitive learning.

If more than one center is obtained, g_k^t should be split and the remaining nodes are assigned to the nearest center. In order to prevent the cluster with multiple density peaks from being split, we add a merging test between these split parts to confirm the rationality of this segmentation. Suppose that g_k^t was split into z parts $\{g_{k\_1}^t, g_{k\_2}^t, \ldots, g_{k\_z}^t\}$. For $g_{k\_i}^t$ and $g_{k\_j}^t$, we can extract their between-class distance matrix $D_k(k\_i, k\_j)$ from D_k. Then, their nearest nodes i_{k_i}, j_{k_j} and the minimum distance $d(i_{k\_i}, j_{k\_j}) = \min(D_k(k\_i, k\_j))$ between $g_{k\_i}^t$ and $g_{k\_j}^t$ can be obtained. Once the minimum distance is less than the similarity threshold of i_{k_i} or j_{k_j} as shown in (10), the splitting between $g_{k\_i}^t$ and $g_{k\_j}^t$ is canceled.

$$d(i_{k\_i}, j_{k\_j}) < T_{i_{k\_i}} \cup d(i_{k\_i}, j_{k\_j}) < T_{j_{k\_j}}. \tag{10}$$

Otherwise, SORCN confirms the splitting and all connections between these two parts are broken. If g_k^t still has only one center, it reserves the original result. Algorithm 6 in Appendix C shows the details of the cluster splitting.

3.5. Computational complexity

The computational costs [59,61–63] of our proposed cognitive method consist of two major parts: online learning and regular reflecting. Suppose the input streaming data has $|X|$ samples, where each sample is d-dimensional vector. SORCN has $|V|$ nodes, which may change from time to time during the online learning. The initial learning phase just executes finite iterations due to the fixed size of the buffer. Therefore, the computation complexity can be treated as $O(1)$. During the developmental learning phase, SORCN should compute $(d|C| \cdot |X| + d|g_k| \cdot |X|)$ times at a minimum, where it just traverses few centers and one subgraph's nodes. The lower asymptotic bound of the complexity is $\Omega(d|X|)$ in this situation. Even at the maximum, SORCN computes $(d|C| \cdot |X| + d|V| \cdot |X|)$ times, which is near to $\Omega(d|V| \cdot |X|)$.

In the reflecting phase, the clustering step just computes B_{size} nodes. Therefore, its computational complexity can be ignored. In the merging step, the most time consuming computation is getting the intra-class distance matrix D_k of each subgraph, which is executed less than an upper asymptotic bound $O(d\sum|g_k|^2) \leqslant O(d|V|^2)$. In the splitting step, all distances can be obtained from each D_k directly. Thus, SORCN needs not take too much computational consumption and costs $O(|V|)$. Normally, $|V|$ is far smaller than $|X|$. The total computational complexity of SORCN in the lifelong learning is less than $O(d|V| \cdot |X|) + O(d|V|^2) + O(|V|)$, which is near to $\Omega(d|V| \cdot |X|)$. This indicates the computational complexity of our method is in the same level as that of traditional self-organizing incremental neural network. When all inputs are different, SORCN creates the same number of node as the inputs, namely, $|V| = |X|$. The complexity becomes $\Omega(d|X|^2)$.

4. Experimental results

4.1. Evaluation methods and criteria

Since a lifelong learning algorithm must be able to process high volume streaming data, we conduct our experiments on a large dataset to evaluate the learning effectiveness of our cognitive network. In our experiments, SORCN is compared with the other four methods. The first is SORCN without reflection, which equals to a SOINN, to demonstrate the effect of the modified CFS clustering algorithm. The second is DT-SOINN, which is the learning network of the visual sample layer in our previous work [22]. The third is the visual Primary Sensory Area (PSA) of PCN [23]. The last is the kernel density estimation and self-organizing incremental neural network (KDESOINN) [64]. All comparative algorithms are implemented with Python code.

We evaluate the experimental results from two aspects. External evaluation criteria consist of Accuracy, Normalized Mutual Information (NMI) [65] and V-measure [66]. Internal evaluation criteria include the numbers of nodes and clusters, Root Mean Squared Error (RMSE), execution time as well as Silhouette

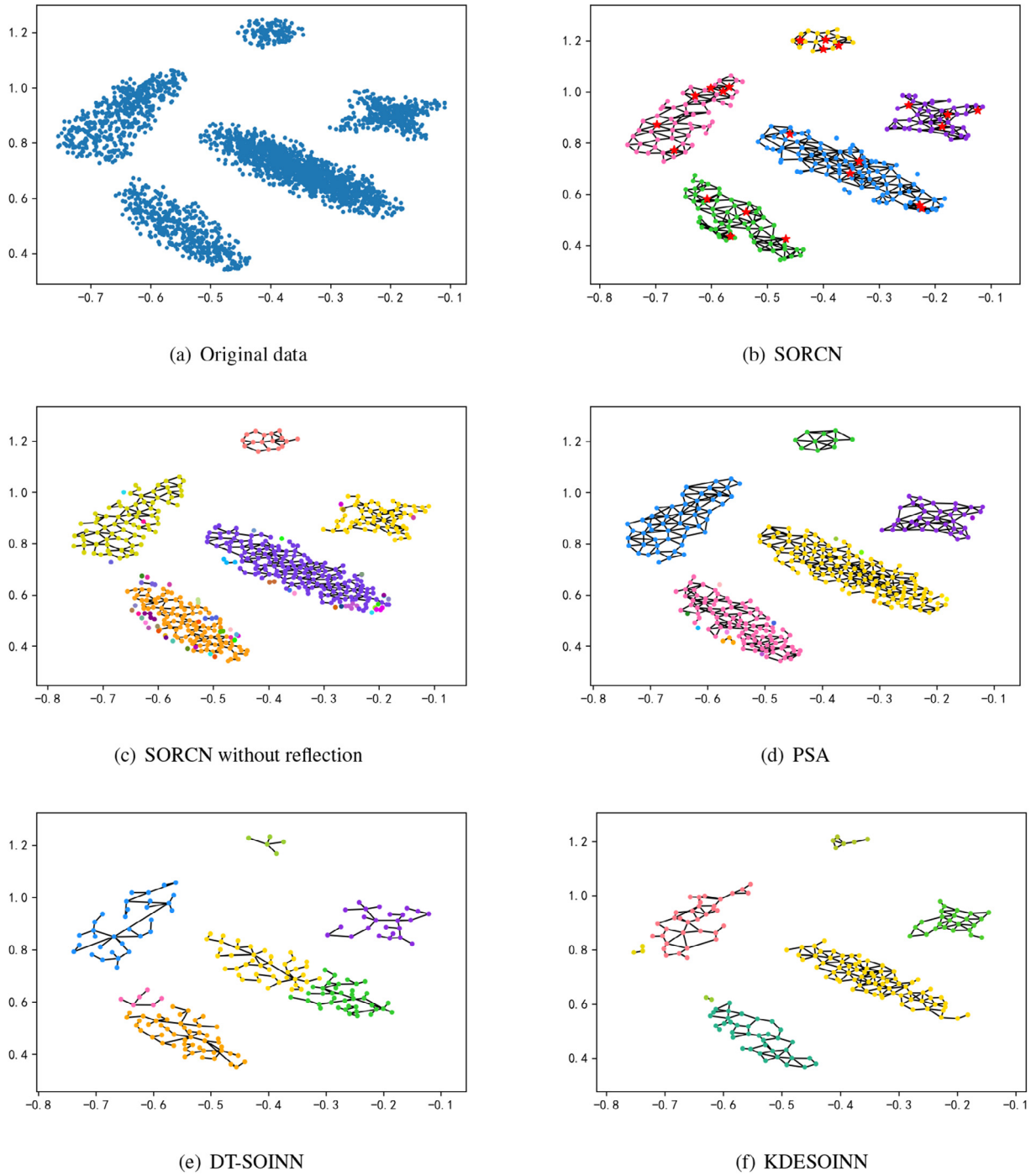
Coefficient [67]. Higher values of accuracy, NMI, V-measure as well as Silhouette Coefficient mean better learning performance. RMSE is utilized to measure quantization error, which is calculated by the square root of the distance between each input and its best matching node [11] as show in (11).

$$RMSE = \sqrt{\frac{1}{|X|}\sum_{i=1}^{|x|}(X_i - w_{b_i})^2}. \tag{11}$$

where b_i is the best matching node of input X_i. RMSE can evaluate the representation effect of nodes on input samples, and smaller value means that the network fits data batter. Fewer nodes and less execution time indicate lower network complexity. Meanwhile, a reasonable number of clusters is also an important criterion. The evaluation should consider the above two aspects simultaneously.

4.2. Artificial dataset

We test our cognitive network on a 2D artificial dataset utilized in the traditional CFS [32] to show visual results. The dataset consists of 4000 instances which belong to five categories separately. As the last 1000 instances are mainly noise points, we just utilize



(a) Original data (b) SORCN

(c) SORCN without reflection (d) PSA

(e) DT-SOINN (f) KDESOINN

Fig. 8. The development results of five cognitive networks on the artificial dataset. (a) Original data of the artificial dataset. (b) SORCN correctly develops five subgraphs and each subgraph contains multiple centers marked by red stars. (c) SORCN without reflection does not integrate these border nodes into the nearest cluster. (d) PSA can self-organize majority nodes into correct clusters, but some isolated nodes still exist. (e) DT-SOINN develops a more concise network but generates more clusters than ground truth. (f) KDESOINN generates a sparse network by removing all isolated nodes.

the first 3000 instances for the experiments as shown in Fig. 8(a). The first experiment is aimed at validating the learning effect of SORCN.

We conduct a comparative trial between SORCN and other four networks. Parameters of SORCN are set as $\xi = 0.02, B_{size} = 100, o' = len(\gamma)/6$ according to experimental experience. Most parameters of DT-SOINN and PSA adopt the default values suggested in [22,23] respectively. In order to guarantee the fairness of comparison, two discrepancy rates of DT-SOINN are set as $\varepsilon_H = 0.2, \varepsilon_L = 0.02$ which determines two similarity thresholds as $TH = TL = 0.02 \cdot \|x_i\|$. The visual similarity threshold of PSA is also set as 0.02 and $\theta = 0.5$. The parameters of KDESOINN are set as $\lambda = 100, age_{\max} = 100, \rho = 0.02, k = 4$.

Fig. 8 shows each network after development. Each cluster is assigned a different color. We can see that SORCN can achieve accurate number of clusters. SORCN without reflection and PSA can cluster majority nodes into correct clusters, but cannot classify isolated nodes into their nearest categories. Although DT-SOINN has better generalization capacity with sparser network, the cluster results are worse than SORCN. Compared with PSA with fixed ratio similarity threshold and DT-SOINN with dynamic adaptive similarity threshold, the proposed combined similarity threshold in Section 3.3.1 may not improve network's generalization but will provide reliable nodes distribution to avoid wrong clustering between two clusters. At the same time, the proposed reflection algorithm can make up its disadvantage and help to develop more concise networks and improve its self-organizing result. KDESOINN develops the fewest nodes in this experiment, because it removes all isolated nodes during learning process. However, this method may lose the true data distribution and generate many small clusters.

We repeat the experiment 10 times and objects are shuffled each time. Table 2 displays the average learning results of 10 times experiments. It can be seen that SORCN develops the most accurate clusters. The average number of nodes of SORCN is more than that of DT-SOINN and KDESOINN but less than that of PSA and SORCN without reflection. Furthermore, SORCN achieves the highest Silhouette Coefficient with 0.5836. This demonstrates that SORCN can also self-organize nodes into the most appropriate clusters. Whereas PAS and SORCN without reflection get negative scores of Silhouette Coefficient because they generate too many clusters. Therefore, our method has good learning performance in this dataset. The RMSE results are inversely proportional to the number of nodes. It indicates that SORCN can balance the good learning effect and the low complexity of the model as well as reduce storage consumption. More important is that the computation time of SORCN is higher than DT-SOINN but lower than other three networks. This means our proposed competitive method indeed reduces the computation consumption during the lifelong learning process.

In order to analyze the computation time of each network in detail as the input sample increases, Fig. 9 illustrates a case of online learning results of five networks on the same streaming sequence. We can see that SORCN can accelerate the learning speed after each reflection, while the other four networks take more and more time due to the number of nodes increases. The reason is that SORCN only needs to compute the distances between input and few centers as well as the nodes in the nearest clusters. It indicates that our proposed competitive learning method can reduce computation consumption obviously. PSA, DT-SOINN and SORCN without reflection have no way to alleviate the continuous incensement of computation time as the number of nodes increases. KDESOINN can alleviate the problem by removing isolated nodes. Nevertheless, it takes extra time for k-NN graph constriction after fixed inputs. Although SORCN also has a reflection process, the reflection turn depends on the number of nodes. In addition, it only computes the intra-class distance matrixes and a portion of the between-class distance matrixes rather than the distances between all nodes. These distance matrixes of SORCN are only calculated once and can be utilized in the following clustering steps. Thus, the modified CFS clustering algorithm is computationally efficient.

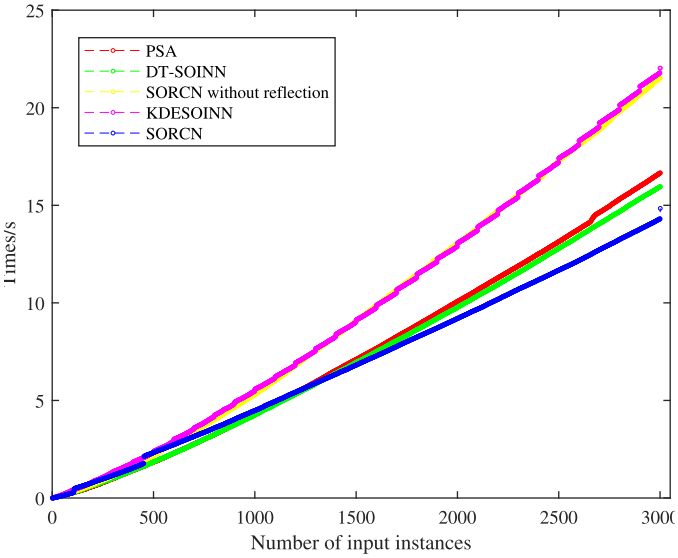
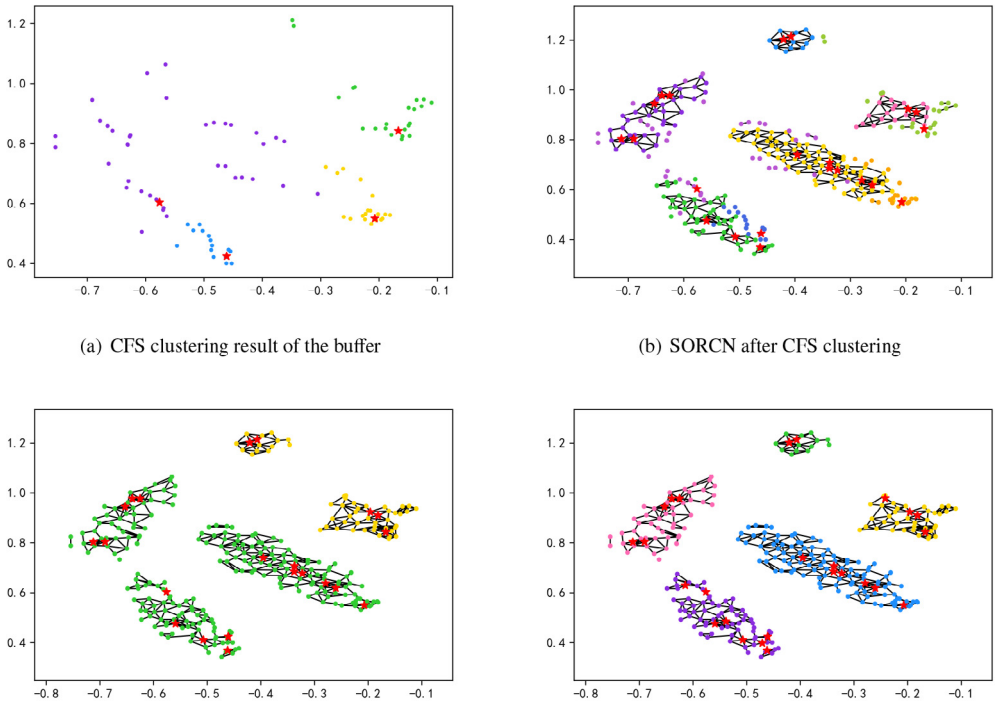


Fig. 9. Computation time versus number of input instances on artificial dataset. The final learning time and each indicator value are marked at the endpoint of each curve. The jump points in SORCN represent its reflecting process and those in KDESOINN indicate its k-NN graph construction.

The second experiment is conducted to validate the modified CFS clustering algorithm. Fig. 10 depicts a case of reflection in the intervals of the developmental learning phase. CFS clustering step may not select appropriate cluster numbers as shown in Fig. 10(a). The reason should be ascribed to the traditional CFS's inability to appropriately process clusters with multiple density peaks as well as nonspherical and unbalanced data. Fig. 10(b) and (c) manifest that merging step can properly integrate new clusters with known subgraphs and realize incrementally clustering of CFS. Simultaneously, the intra-class topological construction in the merging step builds many connections between near nodes. This promotes the development of SORCN and provides a reliable guidance for the adjustment of nodes' similarity thresholds. Fig. 10(d) indicates that the proposed splitting strategy has a good effect on improving the clustering result of CFS.

Table 2
Average results after development on the 2D artificial dataset.

Cognitive architecture	Nodes	Clusters	Silhouette Coefficient	RMSE	Time (s)
PSA [23]	317.7	15.2	−0.0643	8.4853	16.8560
DT-SOINN [22]	197.1	6.6	0.4837	12.2482	15.3462
KDESOINN [61]	184.0	6.3	0.5331	13.4193	21.3793
SORCN	304.3	5.1	0.5836	9.3196	16.1847
SORCN without reflection	498.4	67.2	−0.2651	6.6046	20.7060



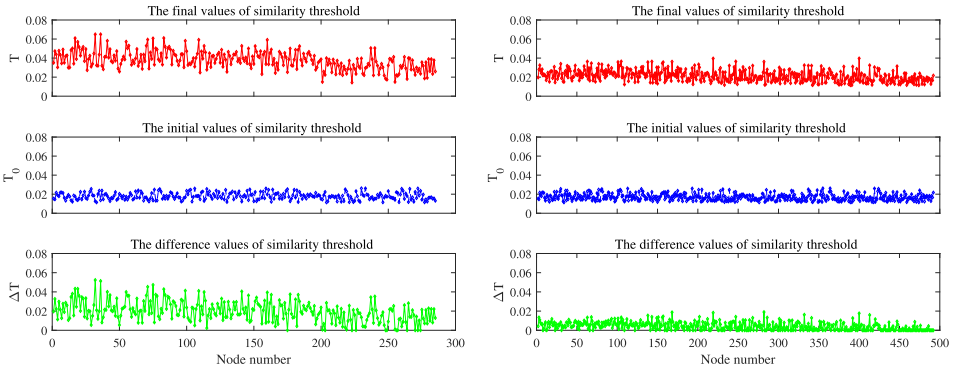
(a) CFS clustering result of the buffer

(b) SORCN after CFS clustering

(c) Merging result of SORCN

(d) Splitting result of SORCN

Fig. 10. A reflection process of SORCN in the intervals of the developmental learning phase. (a) CFS clustering step cannot cluster scattered data correctly. (b) New clusters and known subgraphs in SORCN before merging. (c) Merging step integrates new clusters with the known subgraphs they overlap. (d) Splitting step separates into correct clusters.



(a) SORCN

(b) SORCN without reflection

Fig. 11. The comparison of similarity threshold of each node between SORCN with and without reflection. The top subfigure shows the similarity threshold of each node after development. The middle subfigure shows the initial values of each node. The bottom subfigure shows the difference between final and initial similarity threshold.

Fig. 11 provides clearer evidence to prove that the proposed reflection process contributes to the adjustment of similarity threshold for each node. The variation amplitude of the similarity threshold in our method is larger than that without reflection. A large similarity threshold means a high generalization ability. This is why our method generates fewer nodes. This suggests that the proposed reflection process indeed improves the adjustment of similarity threshold for SOINN through intra-class topology construction.

4.3. Real dataset

We present experimental results on MNIST dataset [68] to evaluate the lifelong learning effect. We let five networks learn its train dataset with 60000 instances in random order and test their learning effect on the test dataset with 10000 instances. Each network is repeatedly run 5 times on this dataset. In order to measure the accuracy and NMI, we label each new node by the "ground-truth" of its first instance. These label information has no effect on networks' learning. Parameters of SORCN are set as $\xi = 0.5, B_{size} = 1000$. Two discrepancy rates of DT-SOINN are set as $\varepsilon_H = 1, \varepsilon_L = 0.5$. The visual similarity threshold of PSA is set as 0.5 and $\theta = 1$. The parameters of KDESOINN are set as $\lambda = 1000, age_{\max} = 100, \rho = 5, k = 4$. The average experimental results are reported in Table 3.

In the external evaluation, the average accuracy and NMI of SORCN are both higher than KDESOINN, but lower than other three networks. The reason is that more nodes contribute to better

Table 3
Average results after development on MNIST dataset.

Cognitive architecture	Accuracy	NMI	Nodes	Clusters	RMSE	Time (s)
PSA [23]	0.9645	0.9139	26286.1	17338.3	4.4204	20169.220
DT-SOINN [22]	0.9602	0.9084	18320.4	181.1	4.6637	11406.786
KDESOINN [61]	0.8647	0.7423	967.2	44.9	5.4513	10169.962
SORCN	0.9152	0.8271	3636.2	231.3	4.5825	3922.461
SORCN without reflection	0.9657	0.9133	35274.3	28634.4	4.3940	20423.960

learning effectiveness, whereas lower complexity of network may cause lower accuracy. These four contrastive networks all fall into the quantity-quality dilemma. However, our method can balance the complexity of network and the accuracy of recognition. In the internal evaluation, SORCN achieves the second fewest number of nodes and middle number of clusters. Compared with SORCN without reflection, the proposed reflection algorithm indeed improves the learning effect and reduces storage as well as computation consumption of SOINN. The reason is that the intra-class topology construction in the merging step can provide a reliable guidance for the adjustment of similarity threshold. Thus, SORCN can improve nodes' generalization to reduce storage consumption and network's self-organizing ability to develop more appropriate object category concepts.

PSA and DT-SOINN heavily rely on the similarity threshold for learning. Therefore, these networks only use local information to self-organize nodes. In contrast, our method not only utilizes nodes' similarity threshold for competitive learning, but also reflects the self-organizing results in the intervals of learning to adjust clusters more precisely according to global data distribution, which contributes to guiding the future learning. Thus, the proposed reflecting algorithm can solve SOINN's dependency of similarity threshold.

The RMSE of SORCN is less than that of KDESOINN and comparable to the results of other three network with more nodes. This indicates SORCN can develop better prototypes for original data, whereas KDESOINN with high RMSE might not represent instances accurately due to over-generalization. More important is that the computation time of SORCN is the least. This demonstrates that SORCN can solve the computation time of traditional completive learning way. Thus, SORCN is a very efficient cognitive network.

Fig. 12 demonstrates the real time learning processes of five networks. From Fig. 10(a), we can see that SORCN's nodes becomes stable quickly. The nodes of PSA, DT-SOINN and SORCN without reflection still increase with input instances. It indicates that SORCN has developed good prototypes to represent learned new instances in early learning phase. Therefore, it can recognize a majority of instances in subsequent learning and the node growth rate is very slow. Although KDESOINN finally develops the least nodes, it still grows fast over the learning process. The reason is that some isolated nodes may be not noise but useful information. These representations are deleted after fixed interval. Therefore, once meeting similar instances, KDESOINN needs to create new nodes to fill the gap of these representations. This is why we omit the step of removing node in SOINN.

In Fig. 12(b), DT-SOINN can develop stable categories, but does not prevent new intra-class nodes from increasing. PSA generates more and more categories during the learning process. One reason is that the predefined similarity threshold may be not suitable for this dataset. Hence, the fixed ratio strategy of similarity threshold cannot apply to lifelong learning tasks with complex streaming data. In contrast, SORCN has the ability of summarizing learned knowledge and adjusting the categories at each reflection turn. This allows the network to adapt dynamic situations autonomously. Consequently, SORCN is an appropriate cognitive network for lifelong learning task.

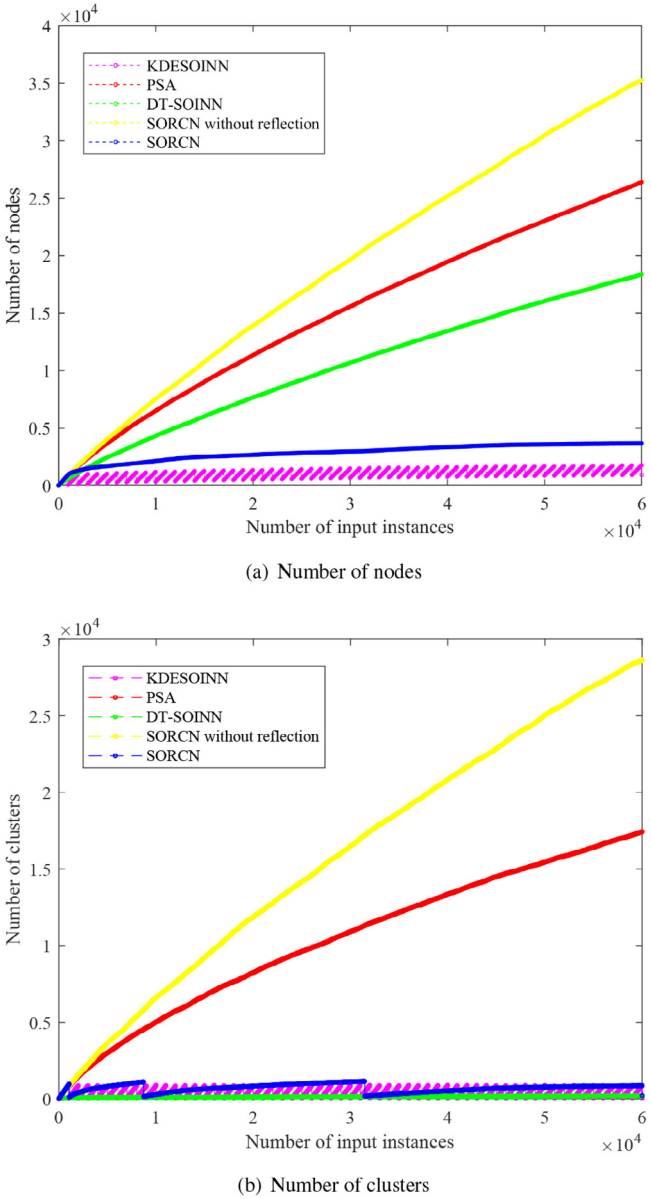


Fig. 12. Number of nodes and Number of clusters versus number of input instances under a same instance sequence.

In Fig. 13, the computation time of SORCN is always low over the learning process. Compared with PSA, DT-SOINN and SORCN without reflection, SORCN generates fewer nodes and its nodes increase very slowly. Therefore, one reason why SORCN can reduce computation time is that it can improve the generalization of the network to generate fewer nodes. Although the number of nodes in SORCN is more than that of KDESOINN, our method still learn faster due to its computationally efficient competitive way and reflection algorithm. In conclusion, SORCN

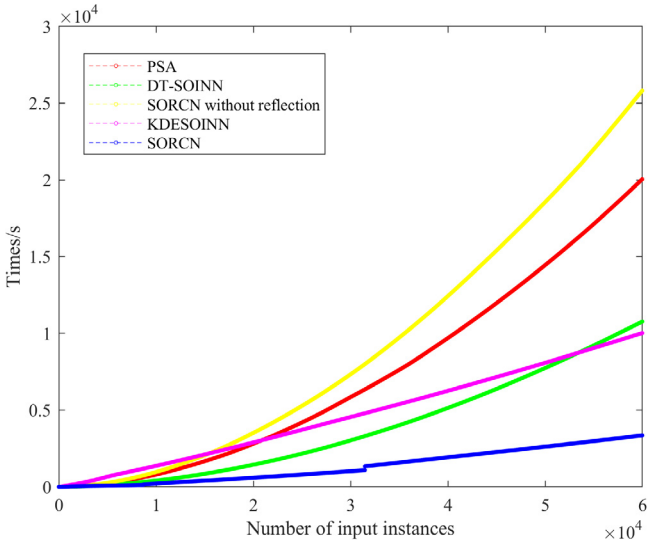


Fig. 13. Computation time versus number of input instances on MNIST dataset.

can solve the storage and computation consumption problems for lifelong learning.

In order to evaluate the learning effect of SORCN for no-stationary data, we also conduct an open-ended experiment as evaluated in [15]. We randomly select 6 categories including 6000 instances from the train dataset of MNIST. These categories are introduced into SORCN one by one in a random sequence. In this experiment, we record the online Average Protocol Accuracy (APA) during learning process as a major evaluating indicator. APA is computed in the way by [15] and the threshold, which indicates the new object category is ready learned, is set as 0.67. 10 experiments are carried out and the learning results are reported in Table 4.

From Table 4, SORCN can get high scores for external evaluation in most situation. For example, experiment 7 achieves the best performance with 0.9970 APA, 0.9943 NMI and 0.8207 V-measure. It indicates that SORCN can not only represent its learned instances, but also allocate most nodes into an appropriate cluster. In the worst case (experiment 8), the introduced categories may have similar instance, which leads to overlap between clusters and low value of V-measure. However, the APA value is still higher than the threshold. This means that SORCN can still master most category concepts introduced during the learning process. Thus, our proposed cognitive model is effective for no-stationary data. In the internal evaluation, SORCN generates more nodes compared with the learning results of SORCN for unordered input in Table 3. One reason is that the selected instances in a same category may be various or dispersive. Therefore, the network needs more nodes to learn the characteristics of these no-stationary data. In addition, the value of RMSE is relatively stable while the number of clusters and the computational time fluctuate between the 10 experiments. The reason is that different categories and their orders introduced into SORCN cause different learning performance. This is inevitable in the learning of no-stationary data.

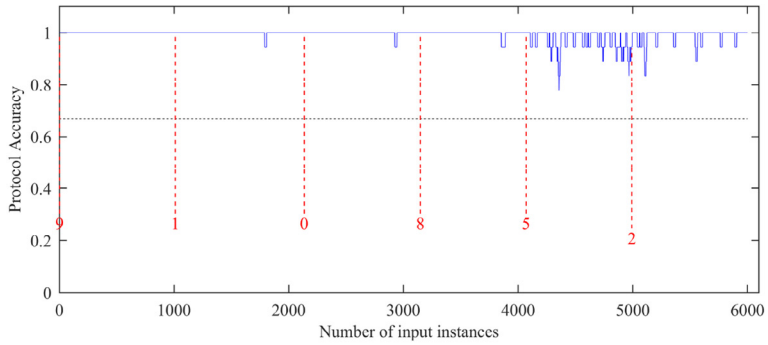
Fig. 14 shows the protocol accuracy of SORCN in experiments 1, 3, 5, 7. The black horizon line describes the protocol accuracy threshold and each vertical red line represents the point that a new category is introduced. It is obvious that experiment 1 and 7 achieves excellent learning performance. Their protocol accuracies are always higher than the threshold and only a few instances are identified incorrectly. In experiment 3, when digit "7" is introduced, the protocol accuracy starts with 1 but drops under the threshold afterwards. The reason is that some handwritten instances of digit "7" are similar to digit "1" learned before. Since SORCN does not involve human feedback and corrections as used in [15], the protocol accuracy is not recovered in the subsequent study of digit "7". However, it rises up to 1 again when meeting new digit "8". This situation also appeared in Experiment 5. These indicate the performance of old categories does not affect the learning of new categories. Thus, SORCN is suitable for learning no-stationary data.

5. Conclusion

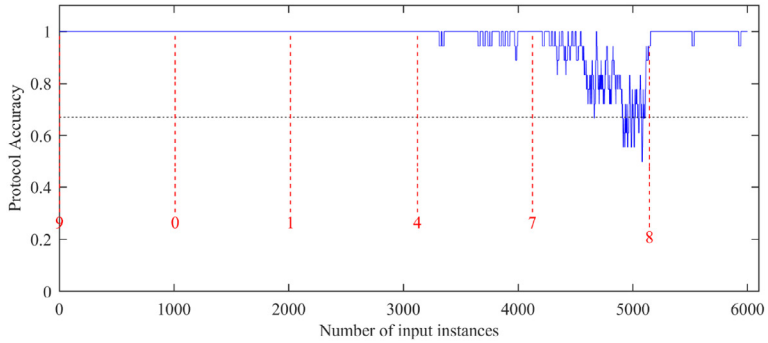
In this paper, a self-organizing and reflecting cognitive network is proposed to develop cognition of robots in lifelong learning. The network introduces a modified CFS clustering algorithm into SOINN according to a novel opinion of cognition development that robots can reflect the learned knowledge during the intervals of learning. The modified CFS clustering algorithm contains clustering, merging and splitting three steps for reflection. CFS clustering adopts an autonomous center selection strategy to realize the integration of SOINN and CFS. The designed merging and splitting strategies not only improve its cluster effectiveness, but also enable CFS to process streaming data in an incremental way. Moreover, a new competitive method based on reflection results is proposed for SOINN to reduce computation. A series of contrast experiments on online and final learning effects have demonstrated that the proposed reflecting algorithm can improve the learning effectiveness and reduce the computation of SOINN. Therefore, SORCN is an efficient cognitive network and can achieve good learning performance in lifelong learning.

Table 4
Learning results of SORCN for no-stationary data in MNIST dataset.

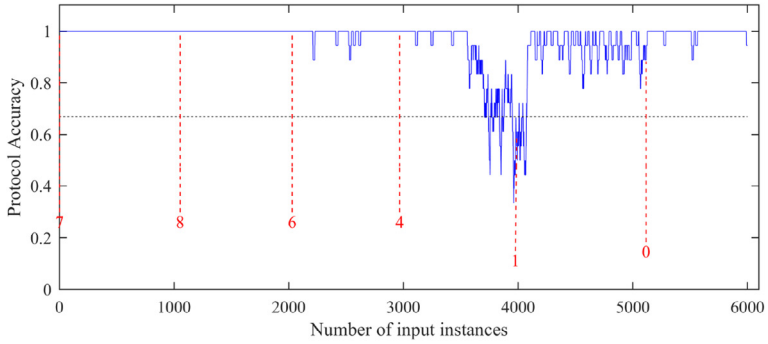
Experiment number	APA	NMI	V-measure	Nodes	Clusters	RMSE	Time (s)
1	0.9925	0.9890	0.5683	4112	239	1.3881	731.378
2	0.8116	0.816	0.5701	3802	162	1.6626	828.039
3	0.9726	0.9655	0.6828	3822	192	1.4452	606.847
4	0.7728	0.8042	0.4301	3458	204	1.9395	802.642
5	0.9634	0.9524	0.5555	3655	163	1.6127	634.427
6	0.8165	0.8056	0.3445	3976	201	1.555	709.238
7	0.9970	0.9943	0.8207	3883	146	1.571	324.633
8	0.7901	0.7801	0.3092	3912	206	1.679	688.0477
9	0.9643	0.9090	0.5828	4326	259	1.708	681.423
10	0.7902	0.7872	0.3081	3910	204	1.5792	583.729



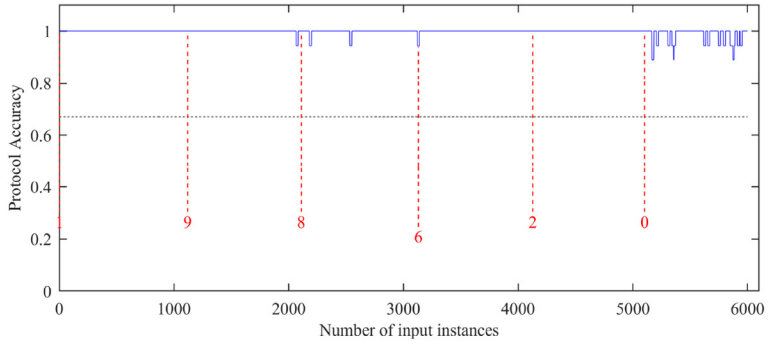
(a) Experiment 1



(b) Experiment 3



(c) Experiment 5



(d) Experiment 7

Fig. 14. Evolution of protocol accuracy versus number of input instances in experiments 1, 3, 5 and 7.

CRediT authorship contribution statement

Ke Huang: Data curation, Formal analysis, Methodology, Software, Validation, Visualization, Writing - original draft. **Xin Ma:** Conceptualization, Funding acquisition, Project administration, Supervision, Writing - review & editing. **Rui Song:** Investigation. **Xuewen Rong:** Resources. **Yibin Li:** Funding acquisition, Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the National Key Research and Development Program, No. 2018YFB1305803, the Joint Fund of National Natural Science Foundation of China and Shandong Province under Grant No. U1706228 and the Fund of National Nature Science Foundation of China under Grant No. 61673245, the Key Development Program for Basic Research of Shandong Province under Grant No. ZR2019ZD07.

Appendix A

Algorithm 4 CFS clustering

Input: Nodes in the buffer B.

Output: Clusters in the buffer B^t, cluster centers C_B^t and distance matrix D.

Calculate the Euclidean distance between nodes in B: $d_{ij} = \|w_i - w_j\|$, $i,j \in \{1,2,\ldots,B_{size}\}$.
Build a distance matrix D using each d_{ij}.
Sort all d_{ij} in ascending order, and set the distance at position $2\% \cdot B_{size}$ as d_c.
for i = $1,2,\ldots,B_{size}$ **do**
    Calculate ρ_i by (2).
end for
for i = $1,2,\ldots,B_{size}$ **do**
    Calculate δ_i by (3).
    $\gamma_i = \rho_i \cdot \delta_i$.
end for
Sort the list $\gamma = \{\gamma_1,\gamma_2,\ldots,\gamma_{B_{size}}\}$in decreasing order as $\gamma' = \{\gamma_o\}_{o=1}^{B_{size}}$.
for o = $1,2,\ldots,B_{size}$ **do**
    Pick out the node i mapping to order o.
    if $\delta_i \geqslant 2\sigma(\delta_i)$ and $\rho_i \geqslant \mu(\rho_i)$ **then**
        Select the node i as a center candidate $c_{B_p}^t$ and $C_B^t = C_B^t \cup \{c_{B_p}^t\}$.
    else
        Break.
    end if
end for
for i = $1,2,\ldots,B_{size}$ **do**
    Assign node i to its nearest center $c_{B_p}^t$ which belong to cluster B_p^t.
    $B^t = B^t \cup \{B_p^t\}$.
end for

Appendix B

Algorithm 5 Buffer-network cluster merging

Input: Clusters in the buffer B^t, cluster centers C_B^t, distance matrix D, network G^{t-1} and centers of subgraphs $\{C_k^{t-1}\}_{k=1}^m$.

Output: Merged network G_{merge}^t, merged centers C_{merge}^t and new distance matrix.

for $p = 1,2,\ldots,len(B^t)$ **do**
    Calculate the distance between $c_{B_p}^t$ of cluster B_p^t with all centers $\{C_k^{t-1}\}_{k=1}^m$ of G^{t-1}.
    Find the first three nearest subgraphs $\{g_{bu}^{t-1}\}_{u=1}^3$ as the merging candidates of B_p^t.
    for $h = \{B_p^t, g_{b1}^{t-1}, g_{b2}^{t-1}, g_{b3}^{t-1}\}$ **do**
        for $i_h = 1,2,\ldots,len(h)$ **do**
            Find four nearest nodes $\{n_v^{i_h}\}_{v=1}^4$ in h for node i_h as its neighbors.
        end for
        Calculate the mean neighbor distance $d_{mnd}(h)$ of h by (8).
        Build a connection $(i_h - n_v^{i_h})$ between each i_h and its neighbors once $d(i_h, n_v^{i_h}) < d_{mnd}(h)$.
        Update TH_{i_h} and $TH_{n_v^{i_h}}$ by (4).
    end for
    for $u = 1,2,3$ **do**
        Get the nearest nodes i_p, j_{bu} between B_p^t and g_{bu}^{t-1}.
        if $d(i_p,j_{bu}) < d_{mnd}(p)$ and $d(i_p,j_{bu}) < d_{mnd}(bu)$ **then**
            $Merge_p = Merge_p \cup \{g_{bu}^{t-1}\}$.
        end if
    end for
    if $Merge_p \neq \varnothing$ **then**
        Merge clusters $g_{b1}^t = g_{b1}^{t-1} \cup Merge_p \cup \{B_p^t\}$.
        Merge centers $C_{b1}^t = C_{b1}^{t-1} \cup \{c_{B_p}^t, C_{bu}^{t-1}\}$, $g_{bu}^{t-1} \in Merge_p$.
        Update G_{merge}^t by g_{b1}^t and C_{merge}^t by C_{b1}^t, and remove merged subgraphs.
        Update the intra-class distance matrix D_{b1}.
    else
        Assign B_p^t as a new subgraph: $G_{merge}^t = G^{t-1} \cup \{B_p^t\}$, $C_{merge}^t = C^{t-1} \cup \{c_{B_p}^t\}$.
    end if
end for
Clear the buffer: $B^t = \varnothing$.

Appendix C

Algorithm 6 Cluster splitting

Input: Merged network G_{merge}^t, merged centers C_{merge}^t and distance matrix D.

Output: Network G^t and centers $\{C_k^t\}_{k=1}^m$ after splitting.

for $k = 1,2,\ldots len(G_{merge}^t)$ **do**
    Calculate the cutoff distance d_c^k for g_k^t based on its intra-class distance matrix D_k.
    for $i = 1,2,\ldots len(g_k^t)$ **do**
        Calculate ρ_i^k, δ_i^k and γ_i^k of node i.
    end for
    Sort γ^k in decreasing order.
    Select the first o' values $\{\gamma_o^k\}_{o=1}^{o'}$ and calculate their mean values γ_{mean}^k.

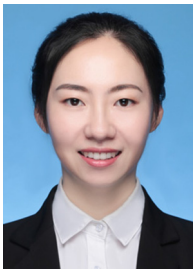
(*continued on next page*)

for $i = 1, 2, \ldots, o'$ **do**
  if $\gamma_i^k > \gamma_{mean}^k$ **then**
    Appoint node i as a center.
  else
    Break.
  end if
end for
if g_k^t gets z centers **then**
  Suppose g_k^t was into split z parts $\{g_{k\_1}^t, g_{k\_2}^t, \ldots, g_{k\_z}^t\}$.
  for $i = 1, 2, \ldots, z-1$ **do**
    for $j = i+1, i+2, \ldots, z$ **do**
      Extract the between-class distance matrix $D_k(k\_i, k\_j)$ of $g_{k\_i}^t$ and $g_{k\_j}^t$.
      Find their nearest nodes i_{k_i}, j_{k_j}.
      Find their minimum distance $d(i_{k\_i}, j_{k\_j}) = \min(D_k(k\_i, k\_j))$.
      if $d(i_{k\_i}, j_{k\_j}) < TH_{i_{k\_i}}$ and $d(i_{k\_i}, j_{k\_j}) < TH_{i_{k\_j}}$ **then**
        Cancel the splitting between $g_{k\_i}^t$ and $g_{k\_j}^t$.
      else
        Split $g_{k\_i}^t$ and $g_{k\_j}^t$, and break all connections between two subgraphs.
      end if
    end for
  end for
end if
end for

References

[1] I. Kotseruba, J.K. Tsotsos, 40 years of cognitive architectures: Core cognitive abilities and practical applications, Artif. Intell. Rev. 53 (1) (2020) 17–94.
[2] A. Lieto, M. Bhatt, A. Oltramari, D. Vernon, The role of cognitive architectures in general artificial intelligence, Cogn. Syst. Res. 48 (2018) 1–3.
[3] G.I. Parisi, R. Kemker, J.L. Part, C. Kanan, S. Wermter, Continual lifelong learning with neural networks: A review, Neural Netw. 113 (2019) 54–71.
[4] A. Prieto, A. Romero, F. Bellas, R. Salgado, R.J. Duro, Introducing separable utility regions in a motivational engine for cognitive developmental robotics, Integr. Comput. Aided Eng. 26 (2018) 3–20.
[5] S. Thrun, A lifelong learning perspective for mobile robot control, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, (IROS), IEEE, 1994, pp. 23–30.
[6] S. Thrun, T.M. Mitchell, Lifelong robot learning, Robot. Auton. Syst. 15 (1995) 25–46.
[7] A. Rannen, R. Aljundi, M.B. Blaschko, T. Tuytelaars, Encoder based lifelong learning, in: Proceedings of the 16th IEEE International Conference on Computer Vision, (ICCV), IEEE, 2017, pp. 1329–1337.
[8] Z. Zhang, Y. Li, Z. Zhang, C. Jin, M. Gao, Adaptive matrix sketching and clustering for semisupervised incremental learning, IEEE Signal Process. Lett. 25 (2018) 1069–1073.
[9] X. Mu, F. Zhu, J. Du, E.-P. Lim, Z.-H. Zhou, Streaming classification with emerging new class by class matrix sketching, in: Proceedings of the 31st AAAI Conference on Artificial Intelligence, (AAAI), AAAI, 2017, pp. 2373-2379.
[10] X. Mu, K.M. Ting, Z. Zhou, Classification under streaming emerging new classes: A solution using completely-random trees, IEEE Trans. Knowl. Data Eng. 29 (2017) 1605–1618.
[11] B. Xu, F. Shen, J. Zhao, A density-based competitive data stream clustering network with self-adaptive distance metric, Neural Netw. 110 (2019) 141–158.
[12] H.B. Ammar, E. Eaton, J.M. Luna, P. Ruvolo, Autonomous cross-domain knowledge transfer in lifelong policy gradient reinforcement learning, in: Proceedings of the 24th International Joint Conference on Artificial Intelligence, (IJCAI), IJCAI, 2015, pp. 3345-3351.
[13] A.J. Glover, G.F. Wyeth, Toward lifelong affordance learning using a distributed Markov model, IEEE Trans. Cogn. Dev. Syst. 10 (2018) 44–55.
[14] F. Bellas, R.J. Duro, A. Faina, D. Souto, Multilevel Darwinist Brain (MDB): Artificial evolution in a cognitive architecture for real robots, IEEE Trans. Auton. Ment. Dev. 2 (2010) 340–354.
[15] S.H. Kasaei, M. Oliveira, G.H. Lim, L.S. Lopes, A.M. Tom, Towards lifelong assistive robotics: A tight coupling between object perception and manipulation, Neurocomputing 291 (2018) 151–166.
[16] H. Liu, F. Sun, B. Fang, Lifelong learning for heterogeneous multi-modal tasks, in: Proceedings of the 2019 International Conference on Robotics and Automation, (ICRA), IEEE, 2019, pp. 6158–6164.
[17] R. Camoriano, G. Pasquale, C. Ciliberto, L. Natale, L. Rosasco, G. Metta, Incremental robot learning of new objects with fixed update time, in: Proceedings of the 2017 IEEE International Conference on Robotics and Automation, (ICRA), IEEE, 2017, pp. 3207-3214.
[18] F. Shen, O. Hasegawa, An incremental network for on-line unsupervised classification and topology learning, Neural Netw. 19 (2006) 90–106.
[19] F. Shen, T. Ogura, O. Hasegawa, An enhanced self-organizing incremental neural network for online unsupervised learning, Neural Netw. 20 (2007) 893–903.
[20] F. Shen, O. Hasegawa, A fast nearest neighbor classifier based on self-organizing incremental neural network, Neural Netw. 21 (2008) 1537–1547.
[21] G.I. Parisi, J. Tani, C. Weber, S. Wermter, Lifelong learning of human actions with deep neural network self-organization, Neural Netw. 96 (2017) 137–149.
[22] K. Huang, X. Ma, R. Song, X. Rong, X. Tian, Y. Li, An autonomous developmental cognitive architecture based on incremental associative neural network with dynamic audiovisual fusion, IEEE Access 7 (2019) 8789–8807.
[23] Y. Xing, X. Shi, F. Shen, J. Zhao, J. Pan, A. Tan, Perception coordination network: A neuro framework for multimodal concept acquisition and binding, IEEE Trans. Neural Netw. Learn. Syst. 30 (2018) 1–15.
[24] F. Shen, Q. Ouyang, W. Kasai, O. Hasegawa, A general associative memory based on self-organizing incremental neural network, Neurocomputing 104 (2013) 57–71.
[25] W. Kim, O. Hasegawa, Improved kernel density estimation self-organizing incremental neural network to perform big data analysis, in: Proceedings of the 25th International Conference on Neural Information Processing, (ICONIP), Springer, 2018, pp. 3-13.
[26] K. Huang, X. Ma, R. Song, X. Rong, X. Tian, Y. Li, A self-organizing developmental cognitive architecture with interactive reinforcement learning, Neurocomputing 377 (2020) 269–285.
[27] D.A. Kolb, Experiential Learning: Experience in the Source of Learning and Development, Prentice Hall Press, Englewood Cliff, NJ, 1984.
[28] A.Y. Kolb, D.A. Kolb, Learning styles and learning spaces: Enhancing experiential learning in higher education, Acad. Manag. Learn. Educ. 4 (2005) 193–212.
[29] J.E. Zull, The art of Changing the Brain: Enriching Teaching by Exploring the Biology of Learning, Stylus Press, Sterling, Virginia, 2002.
[30] A.W. Bernard, D. Gorgas, S. Greenberger, A. Jacques, S. Khandelwal, The use of reflection in emergency medicine education, Acad. Emerg. Med. 19 (2012) 978–982.
[31] T. Nakamura, T. Nagai, Ensemble-of-concept models for unsupervised formation of multiple categories, IEEE Trans. Cogn. Dev. Syst. 10 (2018) 1043–1057.
[32] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (2014) 1492.
[33] Y. Xing, X. Shi, F. Shen, K. Zhou, J. Zhao, A self-organizing incremental neural network based on local distribution learning, Neural Netw. 84 (2016) 143–160.
[34] Y. Lim, S. Ramasamy, A. Gardi, T. Kistan, R. Sabatini, Cognitive human-machine interfaces and interactions for unmanned aircraft, J. Intell. Robot. Syst. 91 (2018) 755–774.
[35] T. Madl, S. Franklin, K. Chen, R. Trappl, A computational cognitive framework of spatial memory in brains and robots, Cogn. Syst. Res. 47 (2018) 147–172.
[36] C. Moulin-Frier, T. Fischer, M. Petit, G. Pointeau, J. Puigbo, U. Pattacini, S.C. Low, D. Camilleri, P. Nguyen, M. Hoffmann, H.J. Chang, M. Zambelli, A. Mealier, A. Damianou, G. Metta, T.J. Prescott, Y. Demiris, P.F. Dominey, P.F.M.J. Verschure, DAC-h3: A proactive robot cognitive architecture to acquire and express knowledge about the world and the self, IEEE Trans. Cogn. Dev. Syst. 10 (2018) 1005–1022.
[37] M. Zhang, Y. Zhang, X. Cheng, An enhanced coupling PD with sliding mode control method for underactuated double-pendulum overhead crane systems, Int. J. Control Autom. Syst. 17 (2019) 1579–1588.
[38] M. Zhang, Y. Zhang, H. Chen, X. Cheng, Model-independent PD-SMC method with payload swing suppression for 3D overhead crane systems, Mech. Syst. Signal Proc. 129 (2019) 381–393.
[39] J.-A. Cervantes, J.-H. Rosales, S. Lpez, F. Ramos, M. Ramos, Integrating a cognitive computational model of planning and decision-making considering affective information, Cogn. Syst. Res. 44 (2017) 10–39.
[40] Z. Chen, B. Liu, Lifelong machine learning, Synthesis Lectures on Artificial Intelligence and Machine Learning 10 (3) (2016) 1–145.
[41] C. Eriksen, A. Nicolai, W. Smart, Learning object classifiers with limited human supervision on a physical robot, in: Proceedings of the 2nd IEEE International Conference on Robotic Computing, (IRC), IEEE, 2018, pp. 282–287.

[42] R. Salgado, A. Prieto, F. Bellas, L. Calvo-Varela, R.J. Duro, Motivational engine with autonomous sub-goal identification for the Multilevel Darwinist Brain, Biol. Inspired Cogn. Archit. 17 (2016) 1–11.
[43] A. Romero, F. Bellas, A. Prieto, R.J. Duro, A re-description based developmental approach to the generation of value functions for cognitive robots, in: Proceedings of the 13th International Conference on Hybrid Artificial Intelligent Systems, (HAIS), Springer Verlag, 2018, pp. 671–683.
[44] W. Kim, O. Hasegawa, Time series prediction of tropical storm trajectory using self-organizing incremental neural networks and error evaluation, Journal of Advanced Computational Intelligence and Intelligent Informatics 22 (2018) 465–474.
[45] W. Kim, O. Hasegawa, Simultaneous forecasting of meteorological data based on a self-organizing incremental neural network, Journal of Advanced Computational Intelligence and Intelligent Informatics 22 (2018) 900–906.
[46] G.I. Parisi, J. Tani, C. Weber, S. Wermter, Lifelong learning of spatiotemporal representations with dual-memory recurrent self-organization, Front. Neurorobotics 12 (78) (2018) 1–15.
[47] L. Sun, R. Liu, J. Xu, S. Zhang, An adaptive density peaks clustering method with fisher linear discriminant, IEEE Access 7 (2019) 72936–72955.
[48] S. Wang, D. Wang, C. Li, Y. Li, G. Ding, Clustering by fast search and find of density peaks with data field, Chin. J. Electron. 25 (2016) 397–402.
[49] R. Liu, H. Wang, X. Yu, Shared-nearest-neighbor-based clustering by fast search and find of density peaks, Inf. Sci. 450 (2018) 200–226.
[50] J. Gao, L. Zhao, Z. Chen, P. Li, H. Xu, Y. Hu, ICFS: An improved fast search and find of density peaks clustering algorithm, in: Proceedings of the 2016 IEEE 14th International Conference on Dependable, Autonomic and Secure Computing, (DASC), IEEE, 2016, pp. 537-543.
[51] Z. Zhou, G. Si, Y. Zhang, K. Zheng, Robust clustering by identifying the veins of clusters based on kernel density estimation, Knowledge-Based Syst. 159 (2018) 309–320.
[52] X. Li, K. Wong, Evolutionary multiobjective clustering and its applications to patient stratification, IEEE T. Cybern. 49 (2019) 1680–1693.
[53] R. Mehmood, R. Bie, H. Dawood, H. Ahmad, Fuzzy clustering by fast search and find of density peaks, in: Proceedings of the 2015 International Conference on Identification, Information, and Knowledge in the Internet of Things, (IIKI), IEEE, 2015, pp. 258-261.
[54] R. Bie, R. Mehmood, S. Ruan, Y. Sun, H. Dawood, Adaptive fuzzy clustering by fast search and find of density peaks, Pers. Ubiquitous Comput. 20 (2016) 785–793.
[55] R. Mehmood, S. El-ashram, R. Bie, H. Dawood, A. Kos, Clustering by fast search and merge of local density peaks for gene expression microarray data, Sci. Rep. 7 (2017) 45602.
[56] Z. Lv, Z. Han, L. Yang, W. Gang, An improved CFSFDP algorithm with cluster center automatically selected based on weighted average method, in: Proceedings of the 7th IEEE Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems, (CYBER), IEEE, 2017, pp. 955-959.
[57] Q. Zhang, C. Zhu, L.T. Yang, Z. Chen, L. Zhao, P. Li, An incremental CFS algorithm for clustering large data in industrial internet of things, IEEE Trans. Ind. Inform. 13 (2017) 1193–1201.
[58] X. Xu, S. Ding, H. Xu, H. Liao, Y. Xue, A feasible density peaks clustering algorithm with a merging strategy, Soft Comput. 23 (2019) 5171–5183.
[59] L. Zhao, Z. Chen, Y. Yang, L. Zou, Z.J. Wang, ICFS clustering with multiple representatives for large data, IEEE Trans. Neural Netw. Learn. Syst. 30 (2019) 728–738.
[60] J. Xie, H. Gao, W. Xie, X. Liu, P.W. Grant, Robust clustering by detecting density peaks and assigning points based on fuzzy weighted K-nearest neighbors, Inf. Sci. 354 (2016) 19–40.
[61] J. Li, K. Lu, Z. Huang, L. Zhu, H.T. Shen, Transfer independently together: A generalized framework for domain adaptation, IEEE Trans. Cybern. 49 (2019) 2144–2155.
[62] J. Li, K. Lu, Z. Huang, L. Zhu, H.T. Shen, Heterogeneous domain adaptation through progressive alignment, IEEE Transactions on Neural Networks and Learning Systems 30 (2019) 1381–1391.
[63] Y. Han, L. Zhu, Z. Cheng, J. Li, X. Liu, Discrete optimal graph clustering, IEEE Transactions on Cybernetics 50 (2020) 1697–1710.
[64] Y. Nakamura, O. Hasegawa, Nonparametric density estimation based on self-organizing incremental neural network for large noisy data, IEEE Transactions on Neural Networks and Learning Systems 28 (2017) 8–17.
[65] A.L.N. Fred, A.K. Jain, Robust data clustering, in: Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, (CVPR), IEEE, 2003, pp. II/128-II/133.
[66] A. Rosenberg, J. Hirschberg, V-Measure: A conditional entropy-based external cluster evaluation measure, in: Proceedings of 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, (EMNLP-CoNLL), ACL, 2007, 410–420.
[67] P.J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, J. Comput. Appl. Math. 20 (1987) 53–65.
[68] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (1998) 2278–2324.



Ke Huang and received her B.S. degree from the school of Automation Engineering, Qingdao University, Qingdao, China, in 2015. She is currently working toward a Ph.D. with the School of Control Science and Engineering, Shandong University, Jinan, China. Her research interests include autonomous cognitive development of robots, self-organizing incremental neural network, multi-modalities integration and interactive reinforcement learning.

Xin Ma received the B.S. degree in industrial automation and the M.S. degree in automation from Shandong Polytech University (now Shandong University), Shandong, China, in 1991 and 1994, respectively. She received the Ph.D. degree in aircraft control, guidance, and simulation from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1998. She is currently a Professor at Shandong University. Her current research interests include artificial intelligence, machine vision, human-robot interaction, and mobile robots.

Rui Song received his B.E. degree in industrial automation in 1998, M.S. degree in control theory and control engineering in 2001 from Shandong University of Science and Technology, and Ph.D. in control theory and control engineering from Shandong University in 2011. He is engaged in research on intelligent sensor networks, intelligent robot technology, and intelligent control systems. His research interests include Medical robots, and the quadruped robots. He is currently an Associate Professor at the School of Control Science and Engineering of Shandong University in Jinan China, and one of the directors of the Center of Robotics of Shandong University.

Xuewen Rong received his B.S. and M.S. degrees from Shandong University of Science and Technology, China, in1996 and 1999, respectively. He received his Ph.D. from Shandong University, China, in 2013. He is currently a senior engineer at the School of Control Science and Engineering, Shandong University, China. His research interests include robotics, mechatronics, and hydraulic servo driving technology.

Yibin Li received his B.S. degree in automation from Tianjin University, Tianjin, China, in 1982, M.S. degree in electrical automation from Shandong University of Science and Technology, Shandong, China, in 1990, and Ph. D. in automation from Tianjin University, China, in 2008.From 1982 to 2003. He worked with Shandong University of Science and Technology, China. Since 2003, he has been the Director of the Center for Robotics, Shandong University. His research interests include robotics, intelligent control theories, and computer control system.