Adaptive Resonance Theory-Based Topological Clustering With a Divisive Hierarchical Structure Capable of Continual Learning

NAOKI MASUYAMA[ID][1], (Member, IEEE), **NARITO AMAKO**[2], **YUNA YAMADA**[2],
YUSUKE NOJIMA[ID][1], (Member, IEEE), **AND HISAO ISHIBUCHI**[ID][3], (Fellow, IEEE)

[1]Graduate School of Informatics, Osaka Metropolitan University, Sakai-shi, Osaka 599-8531, Japan
[2]Graduate School of Engineering, Osaka Prefecture University, Sakai-shi, Osaka 599-8531, Japan
[3]Guangdong Provincial Key Laboratory of Brain-Inspired Intelligent Computation, Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China

Corresponding author: Hisao Ishibuchi (hisao@sustech.edu.cn)

ABSTRACT Adaptive Resonance Theory (ART) is considered as an effective approach for realizing continual learning thanks to its ability to handle the plasticity-stability dilemma. In general, however, the clustering performance of ART-based algorithms strongly depends on the specification of a similarity threshold, i.e., a vigilance parameter, which is data-dependent and specified by hand. This paper proposes an ART-based topological clustering algorithm with a mechanism that automatically estimates a similarity threshold from the distribution of data points. In addition, for improving information extraction performance, a divisive hierarchical clustering algorithm capable of continual learning is proposed by introducing a hierarchical structure to the proposed algorithm. Experimental results demonstrate that the proposed algorithm has high clustering performance comparable with recently-proposed state-of-the-art hierarchical clustering algorithms.

INDEX TERMS Adaptive resonance theory, topological clustering, hierarchical clustering, continual learning.

I. INTRODUCTION

Clustering is an essential technique to extract information from data. With the recent development of IoT technology, the importance of clustering increases as the availability of big data increases. Generally speaking, big data includes a wide range of useful information, e.g., hidden characteristics of data and explicit/implicit relationships among data points and/or attributes. To utilize useful information in big data with a pluralistic and appropriate information granularity, information extraction approaches such as hierarchical clustering

The associate editor coordinating the review of this manuscript and approving it for publication was Easter Selvan Suviseshamuthu[ID].

are widely studied. In particular, since a divisive hierarchical clustering algorithm can adaptively generate partitions in response to the changes in the data distribution, the algorithm has the potential to realize continual learning, which is an essential ability to efficiently utilize big data.

Typical divisive hierarchical clustering algorithms with an adaptive cluster structure are Growing Hierarchical Self-Organizing Map (GHSOM) [1] and Growing Hierarchical Neural Gas (GHNG) [2]. GHSOM uses Self-Organizing Map (SOM) [3] as a base clustering approach while GHNG uses Growing Neural Gas (GNG) [4]. Both GHSOM and GHNG can grow in the vertical and horizontal directions by constructing a tree-like structure based on the

distribution of data points. However, in general, SOM-based and GNG-based clustering algorithms cannot avoid the plasticity-stability dilemma [5], i.e., the trade-off between catastrophic forgetting and continual learning of new information.

In the research field of clustering, Adaptive Resonance Theory (ART) [6] is a successful approach for handling the plasticity-stability dilemma. Among existing ART-based clustering algorithms [7]–[12], Fast Topological Correntropy-Induced Metric-based ART (FTCA) [12] has superior clustering performance and functionality than the others. FTCA utilizes the Correntropy-Induced Metric (CIM) [13] as a similarity measure for realizing a fast and stable self-organizing ability while maintaining an appropriate number of nodes. During a learning process, FTCA adaptively generates topological structures by using nodes and edges to efficiently extract information from a dataset.

In our previous study [14] inspired by GHNG [2], we have introduced Hierarchical FTCA (HFTCA) to improve the clustering performance of FTCA [12]. Although HFTCA shows superior clustering performance than GHSOM and GHNG, HFTCA has a parameter (i.e., a similarity threshold) that significantly affects its clustering performance. In general, the optimal value of the similarity threshold depends on the dataset, which is a common difficulty in the use of ART-based clustering algorithms. Moreover, if an ART-based clustering algorithm has a hierarchical structure, it is necessary to specify a similarity threshold at each layer in advance. In this paper, to solve the above difficulty, we propose CIM-based ART with Edge and Age (CAEA), which has a mechanism to automatically estimate the similarity threshold from the distribution of data points. In addition, we also propose Hierarchical CAEA (HCAEA) for improving the clustering performance of CAEA. Thanks to the continual learning ability of CAEA, HCAEA is also capable of continual learning.

The contributions of this paper are summarized as follows:

(i) A new ART-based clustering algorithm, called CAEA, is proposed by introducing a mechanism to automatically estimate a similarity threshold from the distribution of data points.

(ii) A new divisive hierarchical clustering algorithm, called HCAEA, is proposed by integrating CAEA into a hierarchical structure inspired by HFTCA.

(iii) Empirical studies show that HCAEA has comparable clustering performance than recently-proposed state-of-the-art hierarchical algorithms: GHNG, HFTCA, and GH-EXIN.

The paper is organized as follows. Section II presents the literature review for clustering algorithms and divisive hierarchical clustering algorithms. Section IV describes details of the proposed algorithm and its hierarchical approach, namely CAEA and HCAEA. Section V presents simulation experiments to evaluate the information extraction performance of the proposed algorithms and the above-mentioned state-of-the-art algorithms by using real-world datasets. Section VI

summarizes characteristics of each algorithm based on the experimental results. Section VII concludes this paper.

II. LITERATURE REVIEW
A. CLUSTERING ALGORITHMS CAPABLE OF CONTINUAL LEARNING

Clustering is one of the most widely used approaches for extracting information from data. Gaussian Mixture Model (GMM) [15] and k-means [16] are typical examples of clustering algorithms. Although GMM and k-means are quite simple and highly applicable, they have a common limitation that the number of classes needs to be specified in advance.

One effective approach for solving the drawback of GMM and k-means is growing self-organizing clustering such as GNG [4] and Self-Organizing Incremental Neural Network (SOINN) [17]. GNG and SOINN can adaptively generate topological networks based on a distribution of data points. However, since these algorithms permanently insert new nodes into their networks for memorizing new information, they have the potential to forget learned information (i.e., catastrophic forgetting). This trade-off is called the plasticity-stability dilemma [5].

Several clustering algorithms have been proposed to cope with the plasticity-stability dilemma. As a GNG-based algorithm, Grow When Required (GWR) [18] and gamma-GWR [19] are successful algorithms which appropriately calculate a similarity threshold to prevent an excessive node creation. As a SOINN-based algorithm, SOINN+ [20] and SOINN+ with ghost nodes (GSOINN+) [21] can detect clusters of arbitrary shapes in noisy data streams while avoiding catastrophic forgetting. Another successful approach is ART-based clustering algorithms such as Fuzzy ART [7], Bayes ART [8], and their variants [9], [10]. In general, ART-based clustering algorithms show superior clustering performance than GNG-based and SOINN-based algorithms [11], [12]. Moreover, because ART-based clustering algorithms can theoretically realize sequential and class-incremental learning without catastrophic forgetting, a number of ART-based clustering algorithms and their improvements have been proposed in both supervised learning [22]–[24] and unsupervised learning [7], [8], [25], [26]. One common drawback of ART-based clustering algorithms that they need to specify a similarity threshold (i.e., a vigilance parameter). In most cases, the similarity threshold has a significant impact on their clustering performance while its optimal value depends on the dataset.

B. DIVISIVE HIERARCHICAL CLUSTERING ALGORITHMS

There are two types of hierarchical approaches, i.e., an agglomerative approach and a divisive approach [27]. An agglomerative hierarchical clustering algorithm takes a bottom-up approach where data points are first considered as separate clusters and then merged into larger clusters based on the similarity between clusters. A divisive hierarchical clustering algorithm takes a top-down approach where all

data points are considered as a cluster and then divided into smaller clusters based on the dissimilarity between clusters. In general, agglomerative hierarchical clustering algorithms require the entire dataset because of its learning procedure. Some well-known divisive hierarchical clustering algorithms such as the bisecting k-means algorithm [28] do not have this ability. That is, they need all data points in advance. Thus, they are not suitable for continual learning whereas they show high clustering performance. In contrast, divisive hierarchical clustering algorithms can be used for continual learning by adapting a cluster structure along with the change in the distribution of data points. Thanks to the above characteristics, divisive hierarchical clustering algorithms have the potential to realize continual learning.

In order to perform continual learning in a divisive hierarchical clustering algorithm, it is necessary to have the ability to adaptively extract information and generate a hierarchical structure in response to sequentially given data points. GHSOM [1] utilizes a growing SOM which can grows in the vertical and horizontal directions based on the distribution of data points. One drawback of GHSOM is raised by a SOM architecture, that is, it is difficult for GHSOM to represent multiple data distributions in a single SOM network. GHNG [2] has successfully resolved the drawback of GHSOM by using GNG instead of SOM. One difficulty of GHNG is a large number of parameters. Growing Hierarchical EXcitatory+INhibitory (GH-EXIN) [29] requires only a few parameters while maintaining comparable performance to GHNG. In general, however, GNG-based clustering algorithms suffer from excessive node generation and high sensitivity to the presentation order of input data points. Moreover, the plasticity-stability dilemma [5], i.e., the trade-off between catastrophic forgetting and continual learning of new information, is another unavoidable problem. Applying an ART-based clustering algorithm as a basis of divisive hierarchical clustering is a promising approach because it can theoretically avoid the plasticity-stability dilemma. HFTCA [14] showed the superior clustering performance compared to GHSOM, GHNG, and GH-EXIN while maintaining a high information compression ratio. Although HFTCA is the state-of-the-art of ART-based divisive hierarchical clustering algorithm, HFTCA requires careful parameterization at each hierarchy in order to maintain good clustering performance.

III. PRELIMINARY KNOWLEDGE
This section presents preliminary knowledge for a similarity measurement and a kernel density estimator that are used in the proposed algorithm.

A. CORRENTROPY AND CORRENTROPY-INDUCED METRIC
Correntropy [13] provides a generalized similarity measure between two arbitrary data points $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ and

$\mathbf{y} = (y_1, y_2, \ldots, y_d)$ as follows:

$$C(\mathbf{x}, \mathbf{y}) = E\left[\kappa_\sigma(\mathbf{x}, \mathbf{y})\right], \tag{1}$$

where $E[\cdot]$ is the expectation operation, and $\kappa_\sigma(\cdot)$ denotes a positive definite kernel with a bandwidth σ. The correntropy can be estimated as follows:

$$\hat{C}(\mathbf{x}, \mathbf{y}) = \frac{1}{d} \sum_{i=1}^{d} \kappa_\sigma(x_i, y_i). \tag{2}$$

In this paper, we use the following Gaussian kernel in the correntropy:

$$\kappa_\sigma(x_i, y_i) = \exp\left[-\frac{(x_i - y_i)^2}{2\sigma^2}\right]. \tag{3}$$

A nonlinear metric called CIM is derived from the correntropy [13]. The CIM quantifies the similarity between two data points $\mathbf{x}$ and $\mathbf{y}$ as follows:

$$\text{CIM}(\mathbf{x}, \mathbf{y}, \sigma) = \left[1 - \hat{C}(\mathbf{x}, \mathbf{y})\right]^{\frac{1}{2}}. \tag{4}$$

Here, thanks to the Gaussian kernel without a coefficient $\frac{1}{\sqrt{2\pi}\sigma}$ as defined in (3), a range of the CIM is limited to [0, 1].

In general, the Euclidean distance suffers from the curse of dimensionality. However, the CIM reduces this drawback thanks to the correntropy which calculates the similarity between two arbitrary data points by using a kernel function. Moreover, it has also been shown that the CIM with the Gaussian kernel has a high outlier rejection capability [13].

B. KERNEL DENSITY ESTIMATOR
In general, a bandwidth of a kernel function σ can be estimated based on a kernel density estimator [30], which is defined as follows:

$$\Sigma = U(F_\nu)\Gamma N^{-\frac{1}{2\nu+d}}, \tag{5}$$

$$U(F_\nu) = \left(\frac{\pi^{d/2}2^{d+\nu-1}(\nu!)^2 R(F)^d}{\nu\kappa_\nu^2(F)\left[(2\nu)!! + (d-1)(\nu!!)^2\right]}\right)^{\frac{1}{2\nu+d}}, \tag{6}$$

where Γ denotes a rescale operator (d-dimensional vector) which is defined by a standard deviation of attribute values of each attribute among N training data points. ν is the order of the kernel. $R(F)$ is a roughness function. $\kappa_\nu(F)$ is the moment of the kernel. The details of the derivation of (5) can be found in [30].

IV. PROPOSED ALGORITHM
In this section, first CAEA is explained in detail. Next, a hierarchical approach of CAEA, namely HCAEA, is introduced. Table 1 summarizes the main notations used in this paper.

A. CIM-BASED ART WITH EDGE AND AGE
We use the following notations: A set of training data points is $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \ldots\}$ where $\mathbf{x}_n = (x_{n1}, x_{n2}, \ldots, x_{nd})$ is a d-dimensional feature vector. A set of prototype nodes in CAEA at the time of the presentation of a data point $\mathbf{x}_n$

TABLE 1. Summary of notations.

Notation	Description
$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \ldots\}$	A set of training data points
$\mathbf{x}_n = (x_{n1}, x_{n2}, \ldots, x_{nd})$	d-dimensional training data point (the nth data point)
$\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K\}$	A set of prototype nodes
$\mathbf{y}_k = (y_{k1}, y_{k2}, \ldots, y_{kd})$	d-dimensional prototype node (the kth node)
$S = \{\sigma_1, \sigma_2, \ldots, \sigma_K\}$	A set of bandwidths for a kernel function
κ_σ	Kernel function with a bandwidth σ
CIM	Correntropy-Induced Metric
k_1, k_2	Indexes of the 1st and 2nd winner nodes
$\mathbf{y}_{k_1}, \mathbf{y}_{k_2}$	The 1st and 2nd winner nodes
V_{k_1}, V_{k_2}	Similarities between a data point $\mathbf{x}_n$ and winner nodes ($\mathbf{y}_{k_1}$ and $\mathbf{y}_{k_2}$)
$V_{\text{threshold}}$	Similarity threshold (a vigilance parameter)
$\mathcal{N}_{k_1}$	A set of neighbor nodes of node $\mathbf{y}_{k_1}$
M_{k_1}	The number of data points that have accumulated by the node $\mathbf{y}_{k_1}$
λ	Predefined interval for computing σ and deleting an isolated node
$a_{(k_1, k_2)}$	Age of edge between nodes $\mathbf{y}_{k_1}$ and $\mathbf{y}_{k_2}$
$a_{\max}$	Predefined threshold of an age of edge
$\mathbf{C} = (\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K)$	Models of CAEA for constructing a hierarchical structure.

is $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K\}$ ($K \in \mathbb{Z}^+$) where a node $\mathbf{y}_k = (y_{k1}, y_{k2}, \ldots, y_{kd})$ has the same dimension as $\mathbf{x}_n$. Furthermore, each node $\mathbf{y}_k$ has an individual bandwidth σ for the CIM, i.e., $S = \{\sigma_1, \sigma_2, \ldots, \sigma_K\}$.

The learning procedure of CAEA is divided into four parts: 1) initialization process for nodes and a bandwidth of a kernel function in the CIM, 2) winner node selection, 3) vigilance test, and 4) node learning and edge construction. Each of them is explained in the following subsections.

1) INITIALIZATION PROCESS FOR NODES AND A BANDWIDTH OF A KERNEL FUNCTION IN THE CIM

In the case that CAEA does not have any nodes, i.e., a set of prototype node $\mathcal{Y} = \varnothing$, the 1st to $\lambda/2$th training data points $X_{\text{init}} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_{\lambda/2}\}$ directly become prototype nodes, i.e., $Y_{\text{init}} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{\lambda/2}\}$, where $\lambda \in \mathbb{Z}^+$ is a predefined parameter of CAEA. Note that λ must be greater than or equal to 2, and if $\lambda/2$ is not an integer, it is rounded to the nearest integer. This parameter is also used for a node deletion process.

In an ART-based clustering algorithm, a vigilance parameter (i.e., a similarity threshold) plays an important role in a self-organizing process. Typically, the similarity threshold is data-dependent and specified by hand. On the other hand, CAEA uses the minimum pairwise CIM value between each of nodes in $Y_{\text{init}} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{\lambda/2}\}$, and the average of pairwise CIM values is used as the similarity threshold $V_{\text{threshold}}$, i.e.,

$$V_{\text{threshold}} = \frac{1}{\lambda/2} \sum_{i=1}^{\lambda/2} \min_{j \neq i} \left[\text{CIM} \left(\mathbf{y}_i, \mathbf{y}_j, \sigma \right) \right], \quad (7)$$

where σ is a bandwidth for a kernel function in the CIM.

Here, σ is calculated by using (5) and (6). In this paper, we set $N = \lambda/2$, and we utilize the Gaussian kernel for the CIM. Therefore in (6), $\nu = 2$, $R(F) = (2\sqrt{\pi})^{-1}$, and

$\kappa_\nu^2(F) = 1$ are derived. As a result, we obtain:

$$\Sigma = \left(\frac{4}{2+d} \right)^{\frac{1}{4+d}} \Gamma \left(\lambda/2 \right)^{-\frac{1}{4+d}}. \quad (8)$$

In (8), Σ contains the bandwidth of each attribute. In this paper, the median of Σ is selected as a representative bandwidth of the Gaussian kernel in the CIM, i.e.,

$$\sigma = \text{median} \left(\Sigma \right). \quad (9)$$

In CAEA, the initial prototype nodes $Y_{\text{init}} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{\lambda/2}\}$ have a common bandwidth of the Gaussian kernel in the CIM, i.e., $S_{\text{init}} = \{\sigma_1, \sigma_2, \ldots, \sigma_{\lambda/2}\}$. When a new node $\mathbf{y}_{K+1}$ is generated from $\mathbf{x}_n$, a bandwidth σ_{K+1} is estimated from the past $\lambda/2$ data points, i.e., $\{\mathbf{x}_{n-\lambda/2}, \ldots, \mathbf{x}_{n-2}, \mathbf{x}_{n-1}\}$ by using (5) and (6). As a result, each new node has a different bandwidth σ depending on the distribution of training data points. Although the similarity threshold $V_{\text{threshold}}$ depends on the distribution of the initial $\lambda/2$ training data points, we regard that an adaptive $V_{\text{threshold}}$ estimation is realized by assigning a different bandwidth σ, which affects the CIM value, for each node in response to the changes in the data distribution.

2) WINNER NODE SELECTION

Once a data point $\mathbf{x}_n$ is presented to CAEA with the prototype node set $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K\}$, two nodes which have a similar state to the data point $\mathbf{x}_n$ are selected, namely, winner nodes $\mathbf{y}_{k_1}$ and $\mathbf{y}_{k_2}$. The winner nodes are determined based on the state of the CIM as follows:

$$k_1 = \underset{\mathbf{y}_i \in \mathcal{Y}}{\arg \min} \left[\text{CIM} \left(\mathbf{x}_n, \mathbf{y}_i, \text{mean}(S) \right) \right], \quad (10)$$

$$k_2 = \underset{\mathbf{y}_i \in \mathcal{Y} \setminus \{\mathbf{y}_{k_1}\}}{\arg \min} \left[\text{CIM} \left(\mathbf{x}_n, \mathbf{y}_i, \text{mean}(S) \right) \right], \quad (11)$$

where k_1 and k_2 denote the indexes of the 1st and 2nd winner nodes, i.e., $\mathbf{y}_{k_1}$ and $\mathbf{y}_{k_2}$, respectively. S is a bandwidth of the Gaussian kernel in the CIM for each node.

3) VIGILANCE TEST

Similarities between the data point $\mathbf{x}_n$ and the 1st and 2nd winner nodes are defined as follows:

$$V_{k_1} = \text{CIM}\left(\mathbf{x}_n, \mathbf{y}_{k_1}, \text{mean}(S)\right), \qquad (12)$$

$$V_{k_2} = \text{CIM}\left(\mathbf{x}_n, \mathbf{y}_{k_2}, \text{mean}(S)\right). \qquad (13)$$

The vigilance test classifies the relationship between a data point and a node into three cases by using a predefined similarity threshold $V_{\text{threshold}}$, i.e.,

- Case I
  The similarity between the data point $\mathbf{x}_n$ and the 1st winner node $\mathbf{y}_{k_1}$ is larger (i.e., less similar) than $V_{\text{threshold}}$, namely:

$$V_{\text{threshold}} < V_{k_1} < V_{k_2}. \qquad (14)$$

- Case II
  The similarity between the data point $\mathbf{x}_n$ and the 1st winner node $\mathbf{y}_{k_1}$ is smaller (i.e., more similar) than $V_{\text{threshold}}$, and the similarity between the data point $\mathbf{x}_n$ and the 2nd winner node $\mathbf{y}_{k_2}$ is larger (i.e., less similar) than $V_{\text{threshold}}$, namely:

$$V_{k_1} \leq V_{\text{threshold}} < V_{k_2}. \qquad (15)$$

- Case III
  The similarities between the data point $\mathbf{x}_n$ and the 1st and 2nd winner nodes are both smaller (i.e., more similar) than $V_{\text{threshold}}$, namely:

$$V_{k_1} \leq V_{k_2} \leq V_{\text{threshold}}. \qquad (16)$$

4) NODE LEARNING AND EDGE CONSTRUCTION

Depending on the result of the vigilance test, a different operation is performed.

If $\mathbf{x}_n$ is classified as Case I by the vigilance test (i.e., (14) is satisfied), a new node $\mathbf{y}_{K+1} = \mathbf{x}_n$ is added to the prototype node set $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K\}$. A bandwidth σ_{K+1} for node $\mathbf{y}_{K+1}$ is calculated by (9). In addition, the number of data points that have been accumulated by the node $\mathbf{y}_{K+1}$ is initialized as $M_{K+1} = 1$.

If $\mathbf{x}_n$ is classified as Case II by the vigilance test (i.e., (15) is satisfied), first, the age of each edge connected to the first winner node $\mathbf{y}_{k_1}$ is updated as follows:

$$a_{(k_1, j)} \leftarrow a_{(k_1, j)} + 1 \quad \left(\forall j \in \mathcal{N}_{k_1}\right), \qquad (17)$$

where $\mathcal{N}_{k_1}$ is a set of nodes that are connected to $\mathbf{y}_{k_1}$ by an edge. After updating the age of each of those edges, an edge whose age is greater than a predefined threshold $a_{\max}$ is deleted. Then, $\mathbf{y}_{k_1}$ is updated as follows:

$$\mathbf{y}_{k_1} \leftarrow \mathbf{y}_{k_1} + \frac{1}{M_{k_1}}\left(\mathbf{x}_n - \mathbf{y}_{k_1}\right). \qquad (18)$$

When updating the node, the difference between $\mathbf{x}_n$ and $\mathbf{y}_n$ is divided by M_{k_1}. Thus, the larger M_{k_1} becomes, the smaller the node position changes. This is based on the idea that the information around a node, where data points are frequently given, is important and should be held by the node.

In Case II, a counter M for the number of data points that have been accumulated by $\mathbf{y}_{k_1}$ is also updated as follows:

$$M_{k_1} \leftarrow M_{k_1} + 1. \qquad (19)$$

If $\mathbf{x}_n$ is classified as Case III by the vigilance test (i.e., (16) is satisfied), the same operations as Case II (i.e., (17), (18), and (19)) are performed. In addition, the neighbor nodes of $\mathbf{y}_{k_1}$ are updated as follows:

$$\mathbf{y}_j \leftarrow \mathbf{y}_j + \frac{1}{10 M_j}\left(\mathbf{x}_n - \mathbf{y}_j\right) \quad \left(\forall j \in \mathcal{N}_{k_1}\right). \qquad (20)$$

Equation (20) has the same concept as (18), but it should be less affected by the data point than $\mathbf{y}_{k_1}$ because it is the neighbor node of $\mathbf{y}_{k_1}$. Thus, the value is multiplied by $1/10$.

In Case III, moreover, if there is no edge between $\mathbf{y}_{k_1}$ and $\mathbf{y}_{k_2}$, a new edge is generated and its age is initialized as follows:

$$a_{(k_1, k_2)} \leftarrow 0. \qquad (21)$$

In the case that there is an edge between nodes $\mathbf{y}_{k_1}$ and $\mathbf{y}_{k_2}$, its age is also reset by (21).

Apart from the above operations in Cases I-III, as a noise reduction purpose, the nodes without edges are deleted every time λ training data points are given.

The learning procedure of CAEA is summarized in Algorithm 1. Note that the prediction procedure of CAEA is that an unknown data point is assigned to the class of its nearest neighbor node.

B. HIERARCHICAL APPROACH FOR CAEA

The procedure for creating the hierarchical structure of HCAEA is as follows. First, CAEA is trained with a set of training data points $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ to generate a topological network (nodes and edges) in the first layer. Supposing that $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K\}$ is generated. Here, we preserve training data points that affect each node $\mathbf{y}_k$ during training process. As a result, we can define a new set of K training data point sets $\mathcal{X}' = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_K\}$ for the second layer, where $\mathcal{X}'$ satisfies $\mathcal{X}_k \neq \varnothing$, $\bigcup_{\mathcal{X}_k \in \mathcal{X}'} \mathcal{X}_k = \mathcal{X}$, and $\forall A, B \in \mathcal{X}' A \neq B \implies A \cap B = \varnothing$. In the second layer, CAEA is independently trained by using each subset $\mathcal{X}_k$ ($k = 1, 2, \ldots, K$) of training data points. This mechanism is used in a hierarchical manner for the training of CAEA. That is, CAEA in the $(h+1)$th layer is independently trained by using subsets of training data points, each of which is defined by the corresponding node in the hth layer. The above procedure is repeated until an additional layer is no longer created by CAEA. That is, when the number of nodes (i.e., K) becomes two in the hth layer, it is too small to represent the distribution of data points. As a result, the training for creating the $(h+1)$th layer is not performed.

The learning procedure of HCAEA is summarized in Algorithm 2. Note that the prediction procedure of HCAEA is similar to CAEA. One difference from CAEA is that an unknown data point is assigned to the class of its nearest

Algorithm 1: Learning Algorithm of CAEA

Input:
a set of training data points: $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \ldots\}$
where $\mathbf{x}_n = (x_{n1}, x_{n2}, \ldots, x_{nd})$ $(\boldsymbol{x}_l \in \Re^d)$,
the interval for computing σ and deleting an isolated node: λ,
and the threshold of an age of edge: $a_{\max}$.
Output:
the CAEA model.
   model contents
   a set of generated nodes: $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K\}$
   $(K \in \mathbb{Z}^+)$,
   a set of bandwidths for a kernel function:
   $S = \{\sigma_1, \sigma_2, \ldots, \sigma_K\}$
   a set of counters: $M = \{M_1, M_2, \ldots, M_K\}$,
   the matrix of edge connections: $\mathbf{e}$,
   and the matrix of edge age: $\mathbf{a}$.

1 **function** LearningCAEA($\mathcal{X}, \lambda, a_{\max}$)
2   **forall** $l \in 1, 2, \ldots, L$ **do**
3   **if** $K < \lambda/2$ **then**
4   Create the new node as $\mathbf{y}_{K+1} = \mathbf{x}_l$.
5   Calculate the bandwidth for a kernel function σ_{K+1} by (8) and (9).
6   **if** $K = \lambda/2$ **then**
7   Calculate the vigilance parameter $V_{\text{threshold}}$ by (7).
8   **else**
9   Search the indexes of winner nodes k_1 and k_2 by (10) and (11), respectively.
10   Update the edge age $a_{(k_1, j)}$ by (17).
11   **if** $a_{(k_1, j)} > a_{max}$ **then**
12   Delete the edge.
13   **if** $V_{k_1} > V_{\text{threshold}}$ **then**
14   Create the new node as $\mathbf{y}_{K+1} = \boldsymbol{x}_l$.
15   Calculate the bandwidth for a kernel function σ_{k+1} by (8) and (9).
16   **else**
17   Update the state of $\mathbf{y}_{k_1}$ by (18).
18   **if** $V_{k_2} \leq V_{\text{threshold}}$ **then**
19   Update the state of neighbor nodes $\mathbf{y}_j$ by (20).
20   Create a new edge $e_{(k_1, k_2)}$ between $\mathbf{y}_{k_1}$ and $\mathbf{y}_{k_2}$.
21   **if** the number of data point inputs l is multiple of a topology adjustment cycle λ **then**
22   **forall** $k \in 1, 2, \ldots, K$ **do**
23   **if** $\mathbf{y}_k$ does not have any edge **then**
24   Remove $\mathbf{y}_k$ from $\mathcal{Y}$.
25   **return** the CAEA model.

Algorithm 2: Learning Algorithm of HCAEA

Input:
a set of training data points: $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \ldots\}$
where $\mathbf{x}_n = (x_{n1}, x_{n2}, \ldots, x_{nd})$ $(\boldsymbol{x}_l \in \Re^d)$,
the interval for computing σ and deleting an isolated node: λ,
and the threshold of an age of edge: $a_{\max}$.
Output:
the HCAEA model.
   model contents
   a set of training data points for the next layer:
   $\mathcal{X}' = \{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_K\}$ $(\mathcal{X}_k \neq \varnothing,$
   $\bigcup_{\mathcal{X}_k \in \mathcal{X}'} \mathcal{X}_k = \mathcal{X}$, and
   $\forall A, B \in \mathcal{X}' A \neq B \implies A \cap B = \varnothing)$,
   a set of generated nodes:
   $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K\}$ $(K \in \mathbb{Z}^+)$,
   a set of bandwidths for a kernel function:
   $S = \{\sigma_1, \sigma_2, \ldots, \sigma_K\}$
   a set of counters: $M = \{M_1, M_2, \ldots, M_K\}$,
   the matrix of edge connections: $\mathbf{e}$,
   the matrix of age of edge: $\mathbf{a}$,
   and a set of child models: $C = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_K\}$.

1 **function** LearningHCAEA($\mathcal{X}, \lambda, a_{\max}$)
2   HCAEA model = LearningCAEA($\mathcal{X}, \lambda, a_{\max}$).
3   **if** $K \geq 2$ **then**
4   **forall** $k \in 1, 2, \ldots, K$ **do**
5   Extract a set of data points $\mathcal{X}_k$ that have accumulated by the node y_k as the training data points for the next layer.
6        $\mathbf{c}_k$ = LearningHCAEA($\mathcal{X}_k, \lambda, a_{\max}$).
7   Update the HCAEA model.
8   **else**
9   **return**.
10   **return** the HCAEA model.

V. SIMULATION EXPERIMENTS

This section presents quantitative comparisons focusing on the information extraction performance of CAEA, HCAEA, GHNG [2], GH-EXIN [29], and HFTCA [14] based on the classification performance. In general, the evaluation of the clustering performance is subjective if a dataset does not have label information (i.e., each pattern in the dataset has no class label). In this paper, therefore, we use datasets with label information and perform classification tasks by using a clustering result as a classifier. This approach allows us to indirectly evaluate the clustering performance, that is, the performance of approximating the data distribution.

The source code of GHNG,[1] GH-EXIN,[2] and HFTCA [3] are provided by the authors of the related papers.

neighbor node which does not have a child node in the generated tree-like structure.

---

[1] http://www.lcc.uma.es/ ejpalomo/software.html
[2] https://github.com/pietrobarbiero/ghexin
[3] https://github.com/Masuyama-lab/HFTCA

A. DATASETS

We utilize five synthetic datasets and nine real-world datasets selected from the commonly used clustering benchmarks [31] and public repositories [32], [33]. Table 2 summarizes statistics of the datasets.

TABLE 2. Statistics of datasets for classification tasks.

Type	Dataset	Number of Attributes	Number of Classes	Number of Instances
Synthetic	Aggregation	2	7	788
	Compound	2	6	299
	Hard Distribution	2	3	1,500
	Jain	2	2	373
	Pathbased	2	3	300
Real-world	Breast Cancer	30	2	569
	COIL20	1,024	20	1,440
	Iris	4	3	150
	Isolet	617	26	1,560
	OptDigits	64	10	5,620
	Seeds	7	3	210
	Sonar	60	2	208
	TOX171	5,748	4	171
	Wine	13	3	178

B. PARAMETER SPECIFICATIONS

CAEA, HCAEA, GHNG [2], GH-EXIN [29], and HFTCA [14] have their own parameters, which influence their clustering performance. This section presents parameter specifications of each algorithm in detail.

Table 3 summarizes parameter settings of each algorithm. In each algorithm, one parameter which has a large impact on the clustering performance is specified by grid search while the other parameters are specified as follows: In GHNG, the parameter τ is unique and rest of them are the same as GNG. Therefore, we use the commonly used GNG parameter settings in GHNG. In GH-EXIN, some parameters are the same as GNG while $\min_{\text{card}}$ and H_{perc} are unique ones. $\min_{\text{card}}$ and H_{perc} are specified in [29]. In HFTCA, a topology construction cycle λ is the same as in [14].

During grid search in our experiments, the training data points in each dataset are presented to each algorithm only once without pre-processing. In addition, each training data point is input under a stationary environment, i.e., the training data points are randomly selected from the entire data. For each parameter specification, we repeat the evaluation 20 times (i.e., 2×10-fold cross validation) with a different random seed for obtaining consistent averaging results.

Table 4 summarizes parameters which are specified by grid search. Using the parameter specifications in Tables 3 and 4, each algorithm shows the highest Normalized Mutual Information (NMI) [34] score for each dataset under a stationary environment. Note that a symbol "−" in HFTCA means that the layer is not generated.

C. CLASSIFICATION TASKS

1) CONDITIONS

In order to demonstrate the information extraction performance of CAEA and HCAEA, we conduct classification tasks not only in a stationary environment but also in a non-stationary environment. In the stationary environment, the training data points are randomly selected from the entire dataset. In the non-stationary environment, the training data points are randomly selected from a specific class in the dataset, and the class is shifted sequentially.

Similar to grid search, we repeat the evaluation 20 times with no pre-processed training data points and a different random seed. The same parameter specifications are used in the stationary and non-stationary environments for each algorithm as explained in Tables 3 and 4. The classification performance is evaluated by Accuracy, NMI [34], Adjusted Rand Index (ARI) [35], and macro-F_1.

As a statistical analysis, the Friedman test and Nemenyi post-hoc analysis [36] are utilized. The Friedman test is used to test the null hypothesis that all algorithms perform equally. If the null hypothesis is rejected, the Nemenyi post-hoc analysis is then conducted. The Nemenyi post-hoc analysis is used for all pairwise comparisons based on the ranks of results over all the evaluation metrics for all datasets. Here, the null hypothesis is rejected at the significance level of 0.05 both in the Friedman test and the Nemenyi post-hoc analysis. All computations are carried out on Matlab 2020a with 2.2GHz Xeon Gold 6238R processor and 768GB RAM.

2) STATIONARY ENVIRONMENT

Table 5 shows the results of the classification performance in the stationary environment. The best value in each metric is indicated by bold. The standard deviation is indicated in parentheses. N/A indicates that an algorithm could not build a predictive model. As an overall trend, GH-EXIN and HCAEA show better performance than CAEA, GHNG and HFTCA. Here, the null hypothesis is rejected on the Friedman test over all the evaluation metrics and datasets. Thus, we apply the Nemenyi post-hoc analysis. Fig. 1 shows a critical difference diagram based on the classification performance including all the evaluation metrics and datasets. Better performance is shown by lower average ranks, i.e., on the right side of a critical distance diagram. In theory, different methods within a critical distance (i.e., a red line) do not have a statistically significance difference [36]. In Fig. 1, GH-EXIN shows the best performance but there is no statistically significant difference from HCAEA. Comparing HCAEA and CAEA, although there is no statistically significant difference, HCAEA shows a lower rank. This suggests that a divisive hierarchical structure of HCAEA has a positive impact on the classification/clustering performance.

Table 6 shows the number of generated nodes by each algorithm in the stationary environment. As a general tendency, GH-EXIN generates a large number of nodes and GHNG generates a small number of nodes. The number of

TABLE 3. Parameter settings of each algorithm for classification tasks.

Algorithm	Parameter	Value	Grid Range	Description
GHNG	τ	grid search	{0.001, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5}	a learning coefficient
	ϵ_W	0.2	—	a learning coefficient
	ϵ_N	0.006	—	a learning coefficient
	α	0.5	—	a learning coefficient
	β	0.995	—	a learning coefficient
	age_{max}	50	—	a maximum age of edge
	λ	100	—	a node insertion cycle
GH-EXIN	H_{max}	grid search	{0.00001, 0.0001, 0.001, 0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}	a task-dependent index
	min_{card}	100	—	a data cardinality
	H_{perc}	0.5	—	a stopping criterion
	ϵ_W	0.2	—	a learning coefficient
	ϵ_N	0.006	—	a learning coefficient
	age_{max}	50	—	a maximum age of edge.
HFTCA	$V_1, V_2, \ldots, V_L$	grid search	{0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.95}	a vigilance parameter of each layer
	λ	100	—	a topology construction cycle
CAEA	λ	grid search	{10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30}	an interval for adapting σ
	a_{max}	10	—	a maximum age of edge
HCAEA	λ	grid search	{10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30}	an interval for adapting σ
	a_{max}	10	—	a maximum age of edge

TABLE 4. Parameters specified by grid search for classification tasks.

Type	Dataset	GHNG τ	GH-EXIN H_{max}	HFTCA $(V_1, V_2, V_3, V_4, V_5)$	CAEA λ	HCAEA λ
Synthetic	Aggregation	0.200	0.30000	(0.95, 0.95, 0.60, −, −)	30	30
	Compound	0.100	0.00010	(0.95, 0.95, 0.10, −, −)	26	30
	Hard Distribution	0.010	0.00001	(0.70, 0.40, 0.30, 0.20, 0.10)	26	28
	Jain	0.300	0.10000	(0.90, 0.90, 0.80, 0.30, −)	26	26
	Pathbased	0.001	0.01000	(0.90, 0.80, 0.70, −, −)	28	28
Real-world	Breast Cancer	0.300	0.00001	(0.70, 0.60, 0.20, −, −)	26	26
	COIL20	0.010	0.00001	(0.40, −, −, −, −)	26	26
	Iris	0.500	0.00001	(0.80, 0.70, 0.50, −, −)	28	28
	Ioslet	0.150	0.00100	(0.70, 0.60, −, −, −)	30	30
	OptDigits	0.001	0.40000	(0.70, −, −, −, −)	20	20
	Seeds	0.100	0.00010	(0.80, 0.60, 0.40, −, −)	28	16
	Sonar	0.250	0.00100	(0.40, −, −, −, −)	24	24
	TOX171	0.300	0.00100	N/A	26	26
	Wine	0.150	0.20000	(0.90, 0.80, 0.50, −, −)	24	28

N/A indicates that an algorithm could not build a predictive model.
A symbol "−" in HFTCA means that a layer is not generated.
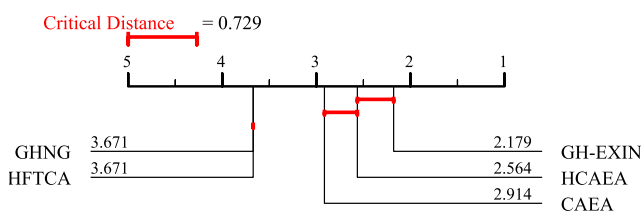
FIGURE 1. Critical difference diagram of classification tasks in the stationary environment.

nodes in HCAEA is larger than that of CAEA because of its hierarchical structure. Note that the number of layers of HCAEA is automatically specified by the algorithm. In this experiment, HCAEA generates a single layer for COIL20, Isolet, Sonar, TOX171, and Wine datasets. These datasets have a small number of training data points compared to the number of attributes. From the above observations, it can be concluded that HCAEA can adaptively and automatically specify the sufficient number of nodes and layers for extracting information.

Table 7 summarizes the results of training time in the stationary environment. In general, GHNG is faster than all the other algorithms because the number of generated nodes is smaller than the others. Regarding GH-EXIN, the computation speed depends on the number of attributes in the dataset. Namely, GH-EXIN is fast when the number of attributes is small while it is slow when the number of attributes is large. The computational efficiency of HCAEA and CAEA depends on the number of generated nodes. In short, the computation speed is slow when the number of generated nodes is large, and vice versa.

TABLE 5. Results of the classification performance in the stationary environment.

Type	Dataset	Metric	GHNG	GH-EXIN	HFTCA	CAEA	HCAEA
Synthetic	Aggregation	Accuracy	0.896 (0.048)	0.993 (0.009)	0.489 (0.059)	0.957 (0.051)	**0.996 (0.006)**
		NMI	0.883 (0.047)	0.987 (0.016)	0.337 (0.049)	0.948 (0.057)	**0.992 (0.012)**
		ARI	0.825 (0.074)	0.986 (0.018)	0.227 (0.063)	0.929 (0.077)	**0.992 (0.013)**
		macro-F_1	0.701 (0.107)	0.987 (0.018)	0.300 (0.045)	0.872 (0.116)	**0.992 (0.014)**
	Compound	Accuracy	0.783 (0.107)	0.930 (0.042)	0.574 (0.072)	0.871 (0.055)	**0.956 (0.030)**
		NMI	0.796 (0.080)	0.906 (0.044)	0.443 (0.081)	0.861 (0.058)	**0.936 (0.039)**
		ARI	0.704 (0.146)	0.888 (0.070)	0.361 (0.122)	0.797 (0.088)	**0.916 (0.062)**
		macro-F_1	0.589 (0.185)	0.911 (0.050)	0.329 (0.055)	0.794 (0.107)	**0.945 (0.049)**
	HardDistribution	Accuracy	0.989 (0.009)	**0.995 (0.006)**	0.992 (0.007)	0.987 (0.008)	0.984 (0.012)
		NMI	0.955 (0.031)	**0.976 (0.025)**	0.967 (0.030)	0.950 (0.029)	0.939 (0.039)
		ARI	0.968 (0.025)	**0.984 (0.017)**	0.976 (0.022)	0.962 (0.023)	0.953 (0.034)
		macro-F_1	0.989 (0.009)	**0.995 (0.005)**	0.992 (0.007)	0.987 (0.008)	0.984 (0.012)
	Jain	Accuracy	0.887 (0.068)	**1.000 (0.000)**	0.842 (0.055)	0.991 (0.016)	0.999 (0.006)
		NMI	0.503 (0.224)	**1.000 (0.000)**	0.304 (0.156)	0.937 (0.103)	0.991 (0.039)
		ARI	0.569 (0.230)	**1.000 (0.000)**	0.417 (0.174)	0.959 (0.069)	0.994 (0.025)
		macro-F_1	0.835 (0.107)	**1.000 (0.000)**	0.771 (0.087)	0.986 (0.023)	0.998 (0.007)
	Pathbased	Accuracy	0.653 (0.095)	0.955 (0.038)	0.548 (0.101)	0.905 (0.071)	**0.982 (0.025)**
		NMI	0.476 (0.109)	0.871 (0.105)	0.181 (0.094)	0.788 (0.114)	**0.947 (0.071)**
		ARI	0.363 (0.137)	0.862 (0.113)	0.124 (0.098)	0.753 (0.156)	**0.943 (0.080)**
		macro-F_1	0.594 (0.105)	0.956 (0.039)	0.527 (0.105)	0.897 (0.074)	**0.982 (0.024)**
Real-world	Breast Cancer	Accuracy	0.889 (0.057)	**0.914 (0.033)**	0.887 (0.044)	0.910 (0.039)	0.909 (0.041)
		NMI	0.541 (0.176)	**0.588 (0.118)**	0.503 (0.161)	0.588 (0.136)	0.586 (0.139)
		ARI	0.606 (0.172)	**0.679 (0.110)**	0.594 (0.141)	0.667 (0.132)	0.665 (0.135)
		macro-F_1	0.878 (0.058)	**0.904 (0.037)**	0.875 (0.050)	0.900 (0.043)	0.900 (0.043)
	COIL20	Accuracy	0.433 (0.056)	**0.830 (0.043)**	0.705 (0.057)	0.555 (0.062)	0.555 (0.062)
		NMI	0.716 (0.037)	**0.892 (0.027)**	0.832 (0.023)	0.735 (0.035)	0.735 (0.035)
		ARI	0.416 (0.060)	**0.765 (0.069)**	0.627 (0.060)	0.446 (0.072)	0.446 (0.072)
		macro-F_1	0.333 (0.049)	**0.815 (0.045)**	0.654 (0.056)	0.462 (0.068)	0.462 (0.068)
	Iris	Accuracy	0.883 (0.068)	0.960 (0.055)	0.947 (0.056)	**0.967 (0.046)**	0.960 (0.050)
		NMI	0.806 (0.070)	0.917 (0.111)	0.888 (0.099)	**0.927 (0.096)**	0.913 (0.106)
		ARI	0.747 (0.118)	0.896 (0.143)	0.873 (0.115)	**0.909 (0.128)**	0.895 (0.134)
		macro-F_1	0.846 (0.112)	0.950 (0.075)	0.924 (0.100)	**0.960 (0.061)**	0.950 (0.067)
	Isolet	Accuracy	0.505 (0.078)	**0.812 (0.014)**	0.482 (0.031)	0.332 (0.032)	0.332 (0.032)
		NMI	0.681 (0.029)	**0.828 (0.011)**	0.613 (0.023)	0.515 (0.020)	0.515 (0.020)
		ARI	0.423 (0.044)	**0.691 (0.021)**	0.351 (0.026)	0.199 (0.024)	0.199 (0.024)
		macro-F_1	0.465 (0.090)	**0.808 (0.013)**	0.444 (0.042)	0.274 (0.034)	0.274 (0.034)
	OptDigits	Accuracy	0.899 (0.016)	**0.953 (0.010)**	0.760 (0.021)	0.841 (0.085)	0.835 (0.112)
		NMI	0.832 (0.026)	**0.912 (0.018)**	0.637 (0.033)	0.766 (0.077)	0.757 (0.109)
		ARI	0.797 (0.031)	**0.901 (0.022)**	0.565 (0.042)	0.710 (0.114)	0.701 (0.162)
		macro-F_1	0.898 (0.017)	**0.953 (0.010)**	0.753 (0.018)	0.832 (0.099)	0.826 (0.125)
	Seeds	Accuracy	**0.881 (0.059)**	0.843 (0.064)	0.864 (0.060)	0.869 (0.065)	0.876 (0.059)
		NMI	**0.715 (0.123)**	0.639 (0.132)	0.691 (0.117)	0.703 (0.136)	0.702 (0.113)
		ARI	**0.669 (0.149)**	0.596 (0.146)	0.654 (0.141)	0.650 (0.163)	0.665 (0.142)
		macro-F_1	**0.872 (0.064)**	0.825 (0.074)	0.848 (0.070)	0.862 (0.067)	0.865 (0.060)
	Sonar	Accuracy	0.493 (0.115)	0.664 (0.099)	0.625 (0.089)	**0.688 (0.116)**	**0.688 (0.116)**
		NMI	0.038 (0.043)	0.149 (0.120)	0.083 (0.070)	**0.182 (0.162)**	**0.182 (0.162)**
		ARI	0.014 (0.034)	0.111 (0.117)	0.064 (0.076)	**0.160 (0.162)**	**0.160 (0.162)**
		macro-F_1	0.453 (0.123)	0.651 (0.100)	0.614 (0.087)	**0.674 (0.121)**	**0.674 (0.121)**
	TOX171	Accuracy	0.346 (0.119)	0.424 (0.086)	N/A	**0.471 (0.139)**	**0.471 (0.139)**
		NMI	0.239 (0.138)	0.357 (0.110)	N/A	**0.359 (0.126)**	**0.359 (0.126)**
		ARI	0.080 (0.069)	**0.143 (0.137)**	N/A	0.125 (0.113)	0.125 (0.113)
		macro-F_1	0.250 (0.098)	0.367 (0.092)	N/A	**0.387 (0.136)**	**0.387 (0.136)**
	Wine	Accuracy	0.657 (0.086)	0.657 (0.098)	0.874 (0.081)	**0.876 (0.089)**	0.846 (0.122)
		NMI	0.475 (0.118)	0.481 (0.125)	**0.729 (0.136)**	0.720 (0.179)	0.684 (0.191)
		ARI	0.334 (0.140)	0.348 (0.167)	0.644 (0.190)	**0.655 (0.218)**	0.610 (0.248)
		macro-F_1	0.586 (0.081)	0.629 (0.094)	**0.883 (0.076)**	0.869 (0.097)	0.835 (0.133)

The best value in each metric is indicated by bold. The standard deviation is indicated in parentheses.
N/A indicates that the corresponding algorithm could not build a predictive model.

TABLE 6. Results of the number of leaf nodes in the stationary environment.

Type	Dataset	GHNG	GH-EXIN	HFTCA	CAEA	HCAEA
Synthetic	Aggregation	23.2 (1.9)	141.2 (9.4)	108.8 (12.1)	35.5 (4.1)	445.7 (55.2)
	Compound	14.8 (1.8)	136.5 (9.7)	64.7 (5.7)	28.0 (2.8)	33.9 (15.5)
	HardDistribution	14.0 (0.0)	294.7 (40.5)	79.6 (10.4)	30.1 (1.6)	30.1 (1.6)
	Jain	11.9 (1.4)	95.1 (6.9)	50.1 (5.8)	25.4 (3.4)	245.1 (35.2)
	Pathbased	39.3 (3.2)	311.9 (9.4)	43.1 (5.1)	33.2 (4.9)	685.6 (73.9)
Real-world	Breast Cancer	3.0 (0.0)	33.8 (3.6)	54.0 (6.1)	25.7 (3.3)	65.8 (15.0)
	COIL20	65.5 (13.4)	2255.8 (144.8)	34.6 (3.5)	32.4 (2.5)	32.4 (2.5)
	Iris	9.8 (1.5)	86.1 (5.1)	25.9 (5.7)	25.6 (3.9)	218.7 (29.9)
	Isolet	154.7 (4.1)	1226.9 (134.8)	76.1 (14.3)	38.3 (7.2)	400.5 (127.1)
	OptDigits	8.5 (1.5)	72.4 (6.9)	723.9 (19.3)	25.6 (2.6)	168.7 (16.7)
	Seeds	3.6 (1.2)	49.2 (4.6)	45.2 (6.1)	25.4 (2.8)	69.4 (19.6)
	Sonar	3.8 (1.3)	44.7 (8.1)	15.7 (2.3)	23.9 (2.2)	23.9 (2.2)
	TOX171	3.0 (0.0)	31.0 (8.2)	N/A	27.3 (3.2)	27.3 (3.2)
	Wine	3.9 (1.4)	41.6 (4.7)	35.0 (3.5)	21.9 (3.4)	28.4 (8.8)

The standard deviation is indicated in parentheses.
N/A indicates that an algorithm could not build a predictive model.

TABLE 7. Results of training time on CPU [sec] in the stationary environment.

Type	Dataset	GHNG	GH-EXIN	HFTCA	CAEA	HCAEA
Synthetic	Aggregation	17.078 (1.408)	1.112 (0.093)	16.860 (1.656)	7.736 (0.512)	16.709 (2.077)
	Compound	0.709 (0.070)	2.512 (0.199)	3.306 (0.209)	0.558 (0.064)	0.571 (0.049)
	HardDistribution	0.124 (0.009)	193.311 (13.438)	42.863 (5.481)	1.679 (0.101)	1.703 (0.091)
	Jain	3.012 (0.263)	0.824 (0.192)	2.951 (0.212)	1.624 (0.121)	3.275 (0.248)
	Pathbased	60.086 (4.993)	6.375 (1.199)	2.109 (0.168)	27.056 (2.211)	71.756 (12.270)
Real-world	Breast Cancer	0.241 (0.019)	0.267 (0.023)	0.657 (0.087)	0.221 (0.023)	0.309 (0.028)
	COIL20	1.667 (0.187)	500.936 (116.089)	1.720 (0.100)	11.124 (0.532)	11.120 (0.518)
	Iris	2.761 (0.287)	0.727 (0.066)	0.287 (0.029)	1.489 (0.194)	3.415 (0.322)
	Isolet	34.098 (2.078)	173.905 (66.630)	9.656 (0.731)	38.382 (4.096)	60.154 (7.300)
	OptDigits	2.082 (0.203)	0.804 (0.325)	236.825 (13.707)	1.293 (0.087)	2.374 (0.135)
	Seeds	0.268 (0.034)	0.432 (0.034)	0.273 (0.019)	0.227 (0.018)	0.377 (0.114)
	Sonar	0.030 (0.003)	1.011 (0.104)	0.040 (0.002)	0.055 (0.003)	0.051 (0.002)
	TOX171	0.002 (0.000)	182.100 (29.984)	N/A	0.024 (0.001)	0.021 (0.002)
	Wine	0.105 (0.015)	0.388 (0.037)	0.108 (0.006)	0.103 (0.031)	0.096 (0.008)

The standard deviation is indicated in parentheses.
N/A indicates that an algorithm could not build a predictive model.

3) NON-STATIONARY ENVIRONMENT

In the case of the non-stationary environment, a continual learning ability is required to maintain the classification/clustering performance because the training data points are randomly selected from a specific class of the dataset, and the class is shifted sequentially.

In general, whether an environment is stationary or non-stationary is unknown and dynamic. In other words, an algorithm should have characteristics that do not deteriorate its classification/clustering performance in any environment. For this reason, the parameter specifications in the non-stationary environment are the same as in the stationary environment, i.e., Tables 3 and 4.

Table 8 shows the results of the classification performance in the non-stationary environment. The best value in each metric is indicated by bold for each dataset. The standard deviation is indicated in parentheses. N/A indicates that an algorithm could not build a predictive model. Similar to the results in the stationary environment, the classification performance of GH-EXIN and HCAEA is generally better

than CAEA, GHNG and HFTCA. Here, the null hypothesis is rejected on the Friedman test over all the evaluation metrics and datasets. Thus, we apply the Nemenyi post-hoc analysis. Fig. 2 shows a critical difference diagram based on the classification performance including all the evaluation metrics and datasets. Similar to the results in the stationary environment, GH-EXIN shows the best performance but there is no statistically significant difference from HCAEA. Regarding HCAEA and CAEA, there is a statistically significant difference between these two algorithms. This result indicates that HCAEA has the stable and superior information extraction performance than CAEA in the non-stationary environments.

Table 9 shows the number of generated nodes by each algorithm in the non-stationary environment. The general trend is the same as in the stationary environment, where GH-EXIN has a large number of generated nodes and GHNG has a small number of them. Regarding HCAEA, this algorithm generates a single layer for COIL20, Isolet, Sonar, TOX171, and Wine datasets, which are the same as in the stationary environment.

TABLE 8. Results of the classification performance in the non-stationary environment.

Type	Dataset	Metric	GHNG	GH-EXIN	HFTCA	CAEA	HCAEA
Synthetic	Aggregation	Accuracy	0.886 (0.043)	0.992 (0.010)	0.478 (0.073)	0.979 (0.026)	**0.996 (0.006)**
		NMI	0.858 (0.045)	0.986 (0.016)	0.362 (0.043)	0.964 (0.035)	**0.992 (0.012)**
		ARI	0.795 (0.065)	0.984 (0.019)	0.237 (0.062)	0.956 (0.041)	**0.993 (0.012)**
		macro-F_1	0.693 (0.100)	0.984 (0.034)	0.345 (0.072)	0.962 (0.059)	**0.993 (0.013)**
	Compound	Accuracy	0.781 (0.094)	0.949 (0.035)	0.560 (0.061)	0.936 (0.060)	**0.959 (0.035)**
		NMI	0.791 (0.086)	0.919 (0.048)	0.430 (0.079)	0.909 (0.070)	**0.935 (0.050)**
		ARI	0.701 (0.131)	0.905 (0.068)	0.354 (0.093)	0.890 (0.098)	**0.922 (0.066)**
		macro-F_1	0.583 (0.142)	0.931 (0.054)	0.344 (0.064)	0.913 (0.089)	**0.952 (0.042)**
	Hard Distribution	Accuracy	0.987 (0.016)	**0.991 (0.007)**	0.986 (0.010)	0.988 (0.010)	0.984 (0.012)
		NMI	0.951 (0.051)	**0.961 (0.030)**	0.945 (0.037)	0.951 (0.038)	0.936 (0.043)
		ARI	0.961 (0.049)	**0.973 (0.022)**	0.959 (0.029)	0.965 (0.028)	0.953 (0.034)
		macro-F_1	0.987 (0.016)	**0.991 (0.007)**	0.986 (0.010)	0.988 (0.010)	0.984 (0.012)
	Jain	Accuracy	0.914 (0.056)	**1.000 (0.000)**	0.851 (0.046)	0.992 (0.026)	**1.000 (0.000)**
		NMI	0.605 (0.160)	**1.000 (0.000)**	0.329 (0.127)	0.964 (0.116)	**1.000 (0.000)**
		ARI	0.673 (0.177)	**1.000 (0.000)**	0.451 (0.130)	0.969 (0.102)	**1.000 (0.000)**
		macro-F_1	0.888 (0.067)	**1.000 (0.000)**	0.796 (0.061)	0.990 (0.035)	**1.000 (0.000)**
	Pathbased	Accuracy	0.715 (0.109)	0.945 (0.049)	0.528 (0.093)	0.895 (0.081)	**0.965 (0.048)**
		NMI	0.510 (0.115)	0.844 (0.120)	0.152 (0.089)	0.765 (0.155)	**0.904 (0.119)**
		ARI	0.422 (0.178)	0.837 (0.129)	0.097 (0.095)	0.718 (0.202)	**0.895 (0.134)**
		macro-F_1	0.668 (0.105)	0.943 (0.056)	0.511 (0.086)	0.894 (0.082)	**0.965 (0.050)**
Real-world	Breast Cancer	Accuracy	0.897 (0.044)	**0.923 (0.031)**	0.789 (0.272)	0.911 (0.040)	0.913 (0.041)
		NMI	0.549 (0.139)	**0.625 (0.105)**	0.420 (0.185)	0.581 (0.144)	0.596 (0.149)
		ARI	0.626 (0.140)	**0.710 (0.102)**	0.506 (0.208)	0.674 (0.133)	0.680 (0.135)
		macro-F_1	0.885 (0.048)	**0.915 (0.033)**	0.779 (0.270)	0.903 (0.044)	0.905 (0.045)
	COIL20	Accuracy	0.416 (0.051)	**0.832 (0.054)**	0.797 (0.065)	0.646 (0.076)	0.646 (0.076)
		NMI	0.707 (0.042)	0.887 (0.028)	**0.891 (0.032)**	0.806 (0.037)	0.806 (0.037)
		ARI	0.401 (0.054)	**0.756 (0.063)**	0.741 (0.078)	0.558 (0.065)	0.558 (0.065)
		macro-F_1	0.305 (0.048)	**0.820 (0.057)**	0.766 (0.080)	0.579 (0.085)	0.579 (0.085)
	Iris	Accuracy	0.907 (0.063)	**0.933 (0.048)**	**0.933 (0.061)**	0.813 (0.146)	0.913 (0.108)
		NMI	0.840 (0.076)	**0.864 (0.085)**	0.860 (0.113)	0.787 (0.131)	0.861 (0.121)
		ARI	0.800 (0.107)	0.830 (0.116)	**0.841 (0.137)**	0.701 (0.196)	0.826 (0.165)
		macro-F_1	0.868 (0.113)	**0.913 (0.074)**	0.904 (0.100)	0.759 (0.187)	0.890 (0.137)
	Isolet	Accuracy	0.543 (0.072)	**0.779 (0.031)**	0.535 (0.031)	0.137 (0.034)	0.137 (0.034)
		NMI	0.691 (0.036)	**0.803 (0.021)**	0.683 (0.017)	0.347 (0.040)	0.347 (0.040)
		ARI	0.443 (0.056)	**0.646 (0.041)**	0.421 (0.027)	0.066 (0.022)	0.066 (0.022)
		macro-F_1	0.504 (0.083)	**0.771 (0.033)**	0.494 (0.046)	0.053 (0.023)	0.053 (0.023)
	OptDigits	Accuracy	0.899 (0.018)	**0.955 (0.009)**	0.794 (0.038)	0.683 (0.221)	0.681 (0.219)
		NMI	0.830 (0.025)	**0.914 (0.017)**	0.701 (0.032)	0.636 (0.171)	0.633 (0.168)
		ARI	0.796 (0.034)	**0.904 (0.020)**	0.587 (0.076)	0.530 (0.231)	0.526 (0.227)
		macro-F_1	0.899 (0.018)	**0.955 (0.009)**	0.802 (0.034)	0.633 (0.270)	0.631 (0.269)
	Seeds	Accuracy	0.867 (0.067)	0.871 (0.054)	0.869 (0.072)	0.826 (0.147)	**0.888 (0.070)**
		NMI	0.707 (0.119)	0.699 (0.105)	0.692 (0.153)	0.680 (0.154)	**0.736 (0.132)**
		ARI	0.647 (0.155)	0.657 (0.119)	0.661 (0.180)	0.624 (0.195)	**0.699 (0.176)**
		macro-F_1	0.856 (0.082)	0.858 (0.069)	0.851 (0.074)	0.809 (0.148)	**0.876 (0.075)**
	Sonar	Accuracy	0.486 (0.069)	0.599 (0.091)	0.650 (0.122)	**0.671 (0.144)**	**0.671 (0.144)**
		NMI	0.031 (0.042)	0.066 (0.069)	0.137 (0.203)	**0.241 (0.140)**	**0.241 (0.140)**
		ARI	0.005 (0.015)	0.043 (0.065)	0.118 (0.209)	**0.164 (0.179)**	**0.164 (0.179)**
		macro-F_1	0.402 (0.098)	0.582 (0.099)	0.631 (0.135)	**0.635 (0.157)**	**0.635 (0.157)**
	TOX171	Accuracy	0.336 (0.098)	0.406 (0.103)	N/A	**0.508 (0.124)**	**0.508 (0.124)**
		NMI	0.218 (0.120)	0.290 (0.098)	N/A	**0.439 (0.155)**	**0.439 (0.155)**
		ARI	0.061 (0.073)	0.066 (0.087)	N/A	**0.223 (0.132)**	**0.223 (0.132)**
		macro-F_1	0.247 (0.090)	0.356 (0.122)	N/A	**0.409 (0.155)**	**0.409 (0.155)**
	Wine	Accuracy	0.637 (0.096)	0.652 (0.099)	**0.866 (0.113)**	0.777 (0.148)	0.767 (0.188)
		NMI	0.460 (0.139)	0.484 (0.111)	**0.711 (0.173)**	0.604 (0.181)	0.604 (0.223)
		ARI	0.328 (0.160)	0.345 (0.146)	**0.632 (0.212)**	0.486 (0.237)	0.502 (0.278)
		macro-F_1	0.570 (0.092)	0.624 (0.092)	**0.870 (0.114)**	0.763 (0.164)	0.752 (0.191)

The best value in each metric is indicated by bold. The standard deviation is indicated in parentheses.
N/A indicates that an algorithm could not build a predictive model.

TABLE 9. Results of the number of leaf nodes in the non-stationary environment.

Type	Dataset	GHNG	GH-EXIN	HFTCA	CAEA	HCAEA
Synthetic	Aggregation	23.9 (2.3)	139.1 (8.7)	107.1 (9.9)	74.6 (6.5)	498.0 (75.2)
	Compound	14.5 (1.6)	130.1 (11.3)	63.0 (5.5)	29.9 (4.1)	50.6 (24.0)
	Hard Distribution	14.0 (0.0)	292.9 (34.1)	71.7 (14.9)	87.3 (9.3)	87.3 (9.3)
	Jain	11.6 (1.6)	93.4 (7.8)	58.4 (6.6)	39.0 (3.9)	278.6 (32.5)
	Pathbased	39.5 (2.3)	305.6 (11.6)	46.4 (5.1)	66.8 (9.6)	559.9 (130.1)
Real-world	Breast Cancer	3.0 (0.0)	33.6 (3.3)	48.4 (17.4)	27.4 (3.4)	40.4 (4.8)
	COIL20	71.3 (12.4)	1758.5 (238.6)	66.7 (4.8)	39.8 (2.3)	39.8 (2.3)
	Iris	10.1 (1.4)	87.0 (4.5)	25.5 (3.7)	38.8 (4.4)	231.5 (35.2)
	Isolet	155.2 (8.1)	1265.2 (97.7)	56.3 (11.0)	100.8 (51.5)	110.8 (65.3)
	OptDigits	9.4 (1.2)	74.5 (6.0)	1091.4 (13.1)	35.9 (3.5)	191.6 (22.2)
	Seeds	3.5 (1.1)	48.1 (3.9)	43.5 (5.3)	27.9 (3.0)	50.8 (23.6)
	Sonar	3.5 (1.1)	46.7 (5.9)	16.8 (2.6)	23.1 (2.0)	23.1 (2.0)
	TOX171	3.0 (0.0)	33.2 (8.4)	N/A	27.5 (3.6)	27.5 (3.6)
	Wine	3.5 (1.1)	42.8 (3.1)	31.7 (3.3)	24.6 (3.7)	27.4 (4.0)

The standard deviation is indicated in parentheses.
N/A indicates that an algorithm could not build a predictive model.

TABLE 10. Results of training time on CPU [sec] in the non-stationary environment.

Type	Dataset	GHNG	GH-EXIN	HFTCA	CAEA	HCAEA
Synthetic	Aggregation	43.125 (7.952)	2.097 (0.320)	110.278 (13.641)	64.900 (2.274)	76.470 (6.296)
	Compound	0.759 (0.066)	2.638 (0.206)	3.633 (0.276)	0.817 (0.157)	0.958 (0.241)
	Hard Distribution	0.158 (0.021)	330.892 (151.078)	46.797 (8.806)	2.529 (0.229)	2.573 (0.242)
	Jain	3.096 (0.293)	0.918 (0.322)	3.304 (0.237)	4.443 (0.501)	4.470 (0.931)
	Pathbased	63.483 (3.684)	6.247 (0.645)	2.199 (0.172)	36.101 (2.311)	60.177 (7.249)
Real-world	Breast Cancer	0.244 (0.018)	0.276 (0.025)	0.759 (0.268)	0.270 (0.037)	0.301 (0.040)
	COIL20	1.665 (0.134)	379.113 (43.146)	1.992 (0.182)	12.973 (0.422)	13.036 (0.629)
	Iris	2.961 (0.399)	0.784 (0.073)	0.316 (0.036)	2.299 (0.718)	4.003 (0.648)
	Isolet	36.363 (1.707)	119.191 (10.419)	7.894 (0.587)	57.970 (10.203)	60.253 (12.377)
	OptDigits	2.220 (0.182)	0.791 (0.263)	347.531 (15.500)	1.355 (0.294)	2.687 (0.396)
	Seeds	0.273 (0.030)	0.428 (0.039)	0.300 (0.021)	0.257 (0.086)	0.377 (0.162)
	Sonar	0.029 (0.003)	0.964 (0.114)	0.042 (0.002)	0.058 (0.010)	0.056 (0.007)
	TOX171	0.002 (0.000)	204.276 (29.950)	N/A	0.023 (0.002)	0.024 (0.002)
	Wine	0.105 (0.014)	0.383 (0.033)	0.110 (0.013)	0.101 (0.026)	0.104 (0.026)

The standard deviation is indicated in parentheses.
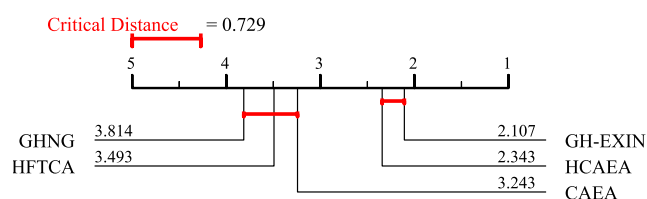N/A indicates that an algorithm could not build a predictive model.

FIGURE 2. Critical difference diagram of classification tasks in the non-stationary environment.

Table 10 summarizes the results of training time in the non-stationary environment. The general trend of the computation speed is also the same as in the stationary environment. Namely, GHNG is faster, the computation speed of GH-EXIN depends on the number of attributes, and the computation speed of HCAEA and CAEA depends on the number of generated nodes.

D. PARAMETER SENSITIVITY

Fig. 3 shows box plots with all the results of NMI obtained by grid search in Section V-B for each dataset in the stationary environment. Depending on the values of the parameters in the grid range, each algorithm shows high/low NMI. By observing the variations of NMI, the sensitivity of an algorithm to each parameter specification can be evaluated.

Although the median of NMI is different, as a general tendency, the interquartile ranges are similar for each algorithm on each dataset except for small datasets (e.g., Jain and Wine) and high-dimensional datasets (e.g., COIL20 and Isolet). In regard to the parameters specified by grid search (see Table 4), those values for GHNG, GH-EXIN, and HFTCA are selected from the entire grid range. In contrast, the parameters of CAEA and HCAEA are mostly selected from a certain range, i.e., {24, 26, 28, 30}, although the candidate of grid values are {10, 12, . . . , 30}. This observation suggests that CAEA and HCAEA can apply the same parameter setting to a wide variety of datasets.

E. COMPUTATIONAL COMPLEXITY

This section presents the computational complexity of CAEA, HCAEA, and comparison algorithms. Specifically, CAEA and HCAEA are analyzed in detail.
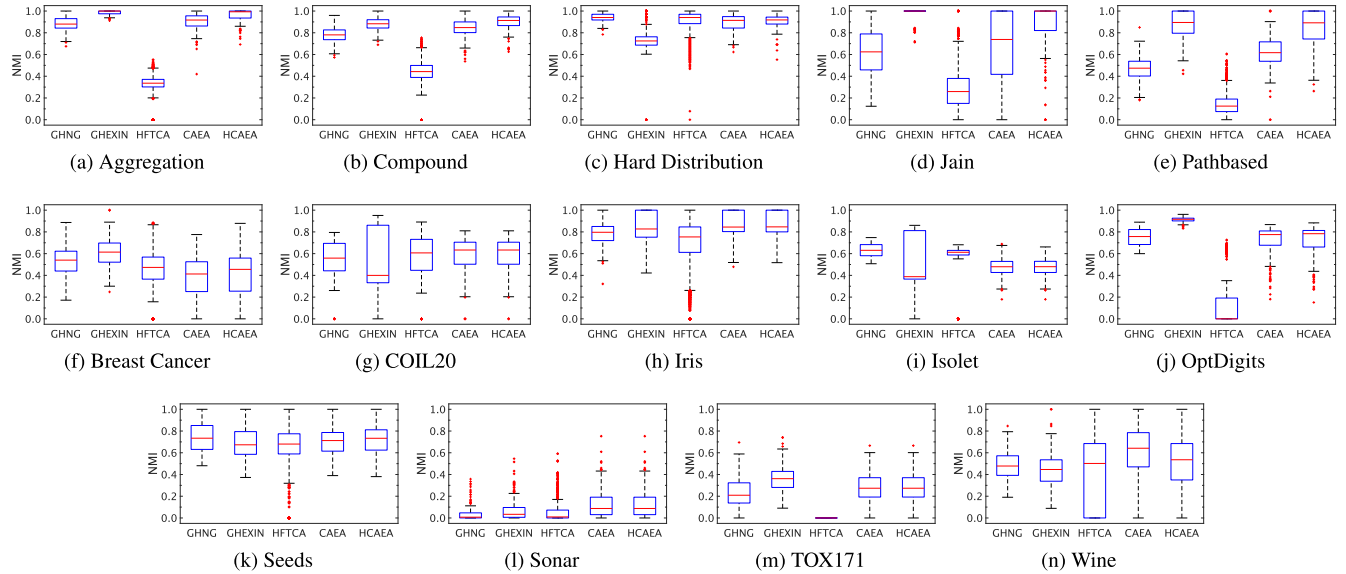
FIGURE 3. Box plot with NMI obtained by grid search in the stationary environment.

Here, assuming n is the number of data points, λ is an interval for adapting σ, d is a dimension of a data point and a node, and K is the number of generated nodes. In CAEA, the computational complexity of each process is as follows: Computing a bandwidth of a kernel function in the CIM is $\mathcal{O}(\frac{n}{\lambda}d)$ (line 5 in Alg. 1). Computing the CIM is $\mathcal{O}(nd\frac{\lambda}{2})$ (line 7 in Alg. 1), and for sorting the result of the CIM is $\mathcal{O}(\frac{\lambda}{2}\log\frac{\lambda}{2})$ (line 7 in Alg. 1). Similarly, for computing the CIM in line 9 is $\mathcal{O}(ndK)$ and for sorting the result of the CIM in line 9 is $\mathcal{O}(K\log K)$. In general, $\lambda \ll n$ and $K \ll n$. Thus, the total computational complexity of CAEA is $\mathcal{O}(nd(\lambda + K))$.

In HCAEA, assuming that the number of data points in the m_lth partition of the lth layer of HCAEA is n_{lm_l}, then the computational complexity is $\mathcal{O}(\frac{n_{lm_l}}{\lambda}d + n_{lm_l}d\frac{\lambda}{2} + \frac{\lambda}{2}\log\frac{\lambda}{2} + n_{lm_l}dK_{lm_l} + K_{lm_l}\log K_{lm_l})$. Here, K_{lm_l} is the number of generated nodes in the m_lth partition of the lth layer. Thus, the total computational complexity from the 2nd to the Lth layer is $\mathcal{O}(\sum_{l=2}^{L}\sum_{i=1}^{m_l}(\frac{n_{li}}{\lambda}d + n_{li}d\frac{\lambda}{2} + \frac{\lambda}{2}\log\frac{\lambda}{2} + n_{li}dK_{li} + K_{li}\log K_{li}))$. In general, $\lambda \ll n$ and $K \ll n$. As a result, the total computational complexity of HCAEA is $\mathcal{O}(nd(\lambda + K) + \sum_{l=2}^{L}\sum_{i=1}^{m_l}n_{li}d(\lambda + m_l))$.

A self-organizing process of GHNG uses GNG but its hierarchical approach is similar to HCAEA. The computational complexity of GNG is $\mathcal{O}(nK)$. Thus, the total computational complexity from the 2nd to the Lth layer is $\mathcal{O}(\sum_{l=2}^{L}\sum_{i=1}^{m_l}n_{li}K_{li})$. As a result, the total computational complexity of GHNG is $\mathcal{O}(nK + \sum_{l=2}^{L}\sum_{i=1}^{m_l}n_{li}K_{li})$.

Regarding GH-EXIN, its computational complexity is analyzed as $\mathcal{O}(bJn\log_b n)$ [29] where b is the average branching factor, J is the average number of epochs, and n is the number of data points. Note that further information can be found in [29].

A learning algorithm and a hierarchical approach of HFTCA is the same with HCAEA except for a calculation

of the vigilance parameter. The computational complexity of the calculation of the vigilance parameter can be ignored here. Thus, the computational complexity of HFTCA is $\mathcal{O}(nd(\lambda + K) + \sum_{l=2}^{L}\sum_{i=1}^{m_l}n_{li}d(\lambda + m_l))$.

The computational complexity for each algorithm is summarized in Table 11.

TABLE 11. Summary of computational complexity.

Algorithm	Computational Complexity
GHNG	$\mathcal{O}(nK + \sum_{l=2}^{L}\sum_{i=1}^{m_l}n_{li}K_{li})$
GH-EXIN	$\mathcal{O}(bJn\log_b n)$
HFTCA	$\mathcal{O}(nd(\lambda + K) + \sum_{l=2}^{L}\sum_{i=1}^{m_l}n_{li}d(\lambda + m_l))$
CAEA	$\mathcal{O}(nd(\lambda + K))$
HCAEA	$\mathcal{O}(nd(\lambda + K) + \sum_{l=2}^{L}\sum_{i=1}^{m_l}n_{li}d(\lambda + m_l))$

VI. DISCUSSION

This section summarizes the characteristics of each algorithm based on the results in Sections V-C2 and V-C3 for emphasizing properties of HCAEA and CAEA.

In regard to GHNG, the computation speed is faster than all the other algorithms. However, the information extraction performance is inferior to the others. One possible reason for its poor performance is that a network generated by GNG does not adequately approximate the distribution of the data points in each dataset. This drawback can be resolved by increasing the number of training epochs, but this is a major disadvantage in terms of the concept of continual learning.

GH-EXIN maintains the superior information extraction performance both in the stationary and non-stationary environments. However, GH-EXIN tends to generate a large number of nodes, and the computation speed of GH-EXIN

depends on the number of attributes in the dataset. Although GH-EXIN shows superior information extraction performance, the above characteristics become disadvantages when the algorithm is applied to a real-world dataset with a large number of attributes. The common disadvantage of the GNG-based algorithms (i.e., GHNG and GH-EXIN), is that there are a lot of parameters to be specified in advance (see Table 3). Moreover, the specified parameter values are significantly for different datasets (see Table 4).

Regarding HFTCA, its information extraction performance is similar to GHNG, and inferior to GH-EXIN, CAEA, and HCAEA. In addition, the vigilance parameter V, which is a predefined parameter, has to be specified to each layer in advance. If the value of the vigilance parameter V is not properly defined, the algorithm cannot generate a sufficient number of nodes to approximate the distribution of data points. Specifically, as shown in Section IV-C, since TOX171 is a high-dimensional data with a small number of data points, HFTCA could not build the predictive model because generated nodes are deleted as isolated nodes before the network (i.e., edge connections) is fully constructed. This problem may be avoided by giving training data points multiple times until the network is sufficiently constructed.

CAEA maintains better information extraction performance than GHNG and HFTCA both in the stationary and non-stationary environments. Moreover, CAEA and HCAEA can apply the same parameter setting to a wide variety of datasets. One disadvantage of HCAEA and CAEA is that these algorithms tends to have poor classification performance over comparison algorithms on datasets with a large number of classes (i.e., COIL20, Isolet, and OptDigits). This is because that CAEA (and also HCAEA) estimates the similarity threshold V and a bandwidth σ for a kernel function from a very small number of training data points compared to whole data points, the similarity threshold V and the bandwidth σ are not suitable for the distribution of data points if the number of classes is large. This problem could be solved by adapting a large value of λ. As shown in Figs. 1 and 2, a unique drawback of CAEA is that the information extraction performance is significantly different between the stationary and non-stationary environments. By introducing a hierarchical structure to CAEA, HCAEA successfully solves the disadvantage of CAEA. In general, HCAEA shows better information extraction performance than CAEA and comparable performance to GH-EXIN without the specification of a large number of parameters. One problem of HCAEA is that the algorithm tends to generate excessive nodes in the case of Iris and Pathbased datasets. However, this problem can be solved by implementing a mechanism to delete unnecessary nodes in HCAEA.

VII. CONCLUSION
This paper proposed an ART-based topological clustering algorithm, called CAEA. CAEA automatically estimates a similarity threshold, i.e., a vigilance parameter, from the distributions of data points. In addition, a divisive hierarchical

clustering algorithm capable of continual learning, called HCAEA, was also proposed by applying a hierarchial structure to CAEA. The experimental results showed that the hierarchical structure of HCAEA improves the information extraction performance while solving the disadvantage of CAEA, i.e., the information extraction performance is significantly different between the stationary and non-stationary environments. Moreover, it has been shown that HCAEA has various advantages (i.e. a small number of parameters, an automated mechanism for hierarchical structure design) and comparable clustering performance to recently-proposed state-of-the-art hierarchical clustering algorithms.

In this paper, we focused only on the avoidance of catastrophic forgetting in order to achieve stable continual learning. However, dealing with concept drift, i.e., the change of the concepts in the learned information, is also important for maintaining a continual learning ability [37]. Therefore, as a future research topic, we will consider dealing with concept drift in HCAEA in order to extend the functionality of the algorithm.

REFERENCES

[1] A. Rauber, D. Merkl, and M. Dittenbach, "The growing hierarchical self-organizing map: Exploratory analysis of high-dimensional data," *IEEE Trans. Neural Netw.*, vol. 13, no. 6, pp. 1331–1341, Nov. 2002.

[2] E. J. Palomo and E. López-Rubio, "The growing hierarchical neural gas self-organizing neural network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 9, pp. 2000–2009, Sep. 2016.

[3] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biol. Cybern.*, vol. 43, pp. 59–69, Jan. 1982.

[4] B. Fritzke, "A growing neural gas network learns topologies," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 7, 1995, pp. 625–632.

[5] G. A. Carpenter and S. Grossberg, "The ART of adaptive pattern recognition by a self-organizing neural network," *Computer*, vol. 21, no. 3, pp. 77–88, Mar. 1988.

[6] S. Grossberg, "Competitive learning: From interactive activation to adaptive resonance," *Cognit. Sci.*, vol. 11, no. 1, pp. 23–63, 1987.

[7] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural Netw.*, vol. 4, no. 6, pp. 759–771, 1991.

[8] B. Vigdor and B. Lerner, "The Bayesian ARTMAP," *IEEE Trans. Neural Netw.*, vol. 18, no. 6, pp. 1628–1644, Nov. 2007.

[9] M. Tscherepanow, "TopoART: A topology learning hierarchical ART network," in *Proc. Int. Conf. Artif. Neural Netw.*, Springer: Berlin, Germany, 2010, pp. 157–167.

[10] N. Masuyama, C. K. Loo, and F. Dawood, "Kernel Bayesian ART and ARTMAP," *Neural Netw.*, vol. 98, pp. 76–86, Feb. 2018.

[11] N. Masuyama, C. K. Loo, H. Ishibuchi, N. Kubota, Y. Nojima, and Y. Liu, "Topological clustering via adaptive resonance theory with information theoretic learning," *IEEE Access*, vol. 7, pp. 76920–76936, 2019.

[12] N. Masuyama, N. Amako, Y. Nojima, Y. Liu, C. K. Loo, and H. Ishibuchi, "Fast topological adaptive resonance theory based on correntropy induced metric," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Dec. 2019, pp. 2215–2221.

[13] W. Liu, P. P. Pokharel, and J. C. Principe, "Correntropy: Properties and applications in non-Gaussian signal processing," *IEEE Trans. Signal Process.*, vol. 55, no. 11, pp. 5286–5298, Nov. 2007.

[14] Y. Yamada, N. Masuyama, N. Amako, Y. Nojima, C. K. Loo, and H. Ishibuchi, "Divisive hierarchical clustering based on adaptive resonance theory," in *Proc. Int. Symp. Community-Centric Syst. (CcS)*, Sep. 2020, pp. 1–6, doi: 10.1109/CcS49175.2020.9231474.

[15] G. J. McLachlan, S. X. Lee, and S. I. Rathnayake, "Finite mixture models," *Annu. Rev. Statist. Appl.*, vol. 6, pp. 355–378, Jan. 2019.

[16] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. IT-28, no. 2, pp. 129–137, Mar. 1982.

[17] S. Furao and O. Hasegawa, "An incremental network for on-line unsupervised classification and topology learning," *Neural Netw.*, vol. 19, no. 1, pp. 90–106, Jan. 2006.

[18] S. Marsland, J. Shapiro, and U. Nehmzow, "A self-organising network that grows when required," *Neural Netw.*, vol. 15, nos. 8–9, pp. 1041–1058, 2002.

[19] G. I. Parisi, J. Tani, C. Weber, and S. Wermter, "Lifelong learning of human actions with deep neural network self-organization," *Neural Netw.*, vol. 96, pp. 137–149, Dec. 2017.

[20] C. Wiwatcharakoses and D. Berrar, "SOINN+, a self-organizing incremental neural network for unsupervised learning from noisy data streams," *Exp. Syst. Appl.*, vol. 143, Apr. 2020, Art. no. 113069.

[21] C. Wiwatcharakoses and D. Berrar, "A self-organizing incremental neural network for continual supervised learning," *Exp. Syst. Appl.*, vol. 185, Dec. 2021, Art. no. 115662.

[22] S. C. Tan, J. Watada, Z. Ibrahim, and M. Khalid, "Evolutionary fuzzy ARTMAP neural networks for classification of semiconductor defects," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 933–950, May 2014.

[23] A. L. S. Matias and A. R. R. Neto, "OnARTMAP: A fuzzy ARTMAP-based architecture," *Neural Netw.*, vol. 98, pp. 236–250, Feb. 2018.

[24] A. L. S. Matias, A. R. R. Neto, C. L. C. Mattos, and J. P. P. Gomes, "A novel fuzzy ARTMAP with area of influence," *Neurocomputing*, vol. 432, pp. 80–90, Apr. 2021.

[25] L. Wang, H. Zhu, J. Meng, and W. He, "Incremental local distribution-based clustering using Bayesian adaptive resonance theory," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3496–3504, Nov. 2019.

[26] L. E. Brito da Silva, I. Elnabarawy, and D. C. Wunsch, "Distributed dual vigilance fuzzy adaptive resonance theory learns online, retrieves arbitrarily-shaped clusters, and mitigates order dependence," *Neural Netw.*, vol. 121, pp. 208–228, Jan. 2020.

[27] A. Saxena, M. Prasad, A. Gupta, N. Bharill, O. P. Patel, A. Tiwari, M. J. Er, W. Ding, and C.-T. Lin, "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, pp. 664–681, Dec. 2017.

[28] Y. Wang and B. Moseley, "An objective for hierarchical clustering in Euclidean space and its connection to bisecting *k*-means," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 4, pp. 6307–6314.

[29] G. Cirrincione, G. Ciravegna, P. Barbiero, V. Randazzo, and E. Pasero, "The GH-EXIN neural network for hierarchical clustering," *Neural Netw.*, vol. 121, pp. 57–73, Jan. 2020.

[30] D. J. Henderson and C. F. Parmeter, "Normal reference bandwidths for the general order, multivariate kernel density derivative estimator," *Statist. & Probab. Lett.*, vol. 82, no. 12, pp. 2198–2205, Dec. 2012.

[31] P. Fränti and S. Sieranoja, "K-means properties on six clustering benchmark datasets," *Appl. Intell.*, vol. 48, no. 12, pp. 4743–4759, 2018.

[32] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, Apr. 2017.

[33] D. Dua and C. Graff, "UCI machine learning repository," School Inf. Comput. Sci., Univ. California, Irvine, Tech. Rep., 2019. [Online]. Available: http://archive.ics.uci.edu/ml/citation_policy.html

[34] A. Strehl and J. Ghosh, "Cluster ensembles—A knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, pp. 583–617, Dec. 2002.

[35] L. Hubert and P. Arabie, "Comparing partitions," *J. Classification*, vol. 2, no. 1, pp. 193–218, 1985.

[36] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[37] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 12, pp. 2346–2363, Dec. 2019.

• • •