Cluster Merging in Adaptive Resonance Theory-based Clustering Guided by Mass and Distance Criteria

Shoki Inoue*, Naoki Masuyama*, Yusuke Nojima*,
Yuichiro Toda†, Zongying Liu‡, Chu Kiong Loo§, Wei Shiung Liew§
*Department of Core Informatics, Osaka Metropolitan University, Sakai, Japan
Emails: sp22896r@st.omu.ac.jp, masuyama@omu.ac.jp, nojima@omu.ac.jp
†Faculty of Environmental, Life, Natural Science and Technology, Okayama University, Okayama, Japan
Email: ytoda@okayama-u.ac.jp
‡Faculty of Navigation, Dalian Maritime University, Dalian, China
Email: liuzongying@dlmu.edu.cn
§Department of Artificial Intelligence, Universiti Malaya, Kuala Lumpur, Malaysia
Email: ckloo.um@um.edu.my, liew.wei.shiung@gmail.com

***Abstract*—There is a demand for clustering methods that can efficiently handle large-scale data. Adaptive Resonance Theory (ART)-based clustering algorithms adaptively generate prototype nodes that represent local data regions during learning. This mechanism allows ART-based methods to scale to large datasets. However, ART-based methods tend to generate an excessive number of nodes, which reduces scalability and often degrades clustering performance. This paper addresses these limitations by extracting an explicit cluster structure from the generated nodes through node grouping with a criterion that combines node importance and similarity. Experiments on 19 real-world datasets show that the proposed algorithm achieves superior clustering performance compared with state-of-the-art clustering methods.**

The source code is available at https://github.com/Masuyama-lab/CAplusTC

***Index Terms*—Clustering, Adaptive Resonance Theory, Hierarchical Clustering.**

I. Introduction

Clustering is a fundamental technique in data analysis. It groups data according to similarity and extracts latent structure and representative characteristics in an unsupervised manner. Representative clustering methods include the k-means algorithm [1] and the Gaussian Mixture Model [2]. However, these methods require the number of clusters k to be specified in advance.

The representative methods that do not require the number of clusters in advance include Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [3], First Integer Neighbor Clustering Hierarchy (FINCH) [4], and Torque Clustering (TC) [5]. DBSCAN performs density-based clustering and identifies dense regions in the data space, and FINCH performs hierarchical clustering based on distances among data points. TC is a state-of-the-art hierarchical clustering method that merges clusters by considering both inter-cluster distance and cluster mass, where the mass is defined as the number of samples in each cluster. TC defines torque using the mass and the distance between two clusters, and it merges clusters based on this torque. However, TC requires an $n \times n$ distance matrix over data points, and the memory and time costs grow rapidly with the dataset size. This computational burden makes TC difficult to apply to large-scale datasets.

Adaptive Resonance Theory (ART)-based clustering can retain previously learned knowledge and adapt to newly arriving data [6]–[8]. ART-based methods form clusters by measuring the similarity between an input sample and existing prototype nodes. Among the similarity measures used in these methods, the Correntropy-Induced Metric (CIM) provides a localized, kernel-based notion of similarity that can be less sensitive to the curse of dimensionality in high-dimensional data. CIM-based ART plus (CA+) [9] automatically sets a vigilance parameter based on pairwise similarities among active nodes, and it uses this similarity threshold to decide whether to update an existing prototype node or create a new node. However, CA+ does not explicitly construct a cluster structure, which often results in a larger number of clusters than the number of classes in the dataset.

This paper proposes CA+TC, a method that constructs a clearer cluster structure from the nodes generated by CA+ and extends the practical applicability of TC to large-scale datasets. CA+ structures the data distribution by representing local density with a set of prototype nodes whose size is much smaller than the original number of data points. TC then structures these nodes into clusters using the inter-node distance matrix. This combination allows the two methods to compensate for each other's weaknesses. The main contributions of this paper are summarized as follows.

This work was supported by the Japan Society for the Promotion of Science (JSPS) KAKENHI Grant Number JP25K03187. An AI language model (OpenAI ChatGPT, GPT-5.2) was used to support improvements in English expression and to clarify selected explanations in this manuscript. The technical content, experimental design, and conclusions were conceived, produced, and verified entirely by the authors.

(i) CA+TC is proposed, in which CA+ generates nodes and TC extracts the cluster structure from the inter-node relationships by applying it to the generated nodes.
(ii) CA+TC constructs a clearer cluster structure from the nodes generated by CA+ via torque-based node grouping and improves ARI and AMI.
(iii) Experiments on real-world datasets in an online setting with a time-invariant data stream demonstrate the effectiveness of the proposed method.

The paper is organized as follows. Section II describes CA+ and TC as related methods. Section III presents the proposed method. Section IV reports experimental results. Section V concludes this study.

II. Preliminaries

This section reviews the components used in the proposed method. We first introduce correntropy and the Correntropy-Induced Metric (CIM), and then describe the bandwidth selection based on Silverman's rule. Finally, we summarize the learning procedures of CA+ and TC.

A. Correntropy and CIM

As the similarity measure, correntropy [10] is used as a kernel-based similarity between two data points $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_d)$ as

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

In this equation, $\kappa_\sigma(\cdot)$ denotes a positive definite kernel with bandwidth σ. This study adopts the Gaussian kernel

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

CIM is defined from correntropy [10] as

$$\mathrm{CIM}(\mathbf{x}, \mathbf{y}, \sigma) = \left[1 - \hat{C}(\mathbf{x}, \mathbf{y}, \sigma) \right]^{\frac{1}{2}} . \tag{3}$$

The Gaussian kernel in (2) is used without the normalization constant $\frac{1}{\sqrt{2\pi}\sigma}$, therefore $\mathrm{CIM}(\mathbf{x}, \mathbf{y}, \sigma) \in [0, 1]$.

B. Kernel Density Estimation by the Silverman's rule

The bandwidth matrix of the kernel function is determined by Silverman's rule [11] as

$$\mathbf{H} = \left(\frac{4}{2+d} \right)^{\frac{1}{4+d}} \mathbf{\Gamma}\, \lambda^{-\frac{1}{4+d}} , \tag{4}$$

where $\mathbf{\Gamma}$ is a d-dimensional rescaling vector computed from the per-dimension standard deviations over λ samples. The bandwidth for CIM is set to

$$\sigma = \mathrm{median}(\mathbf{H}) . \tag{5}$$

Algorithm 1: Learning procedure of CA+

Input:
a set of data points $\mathcal{X}_c$
Output:
a set of nodes $\mathcal{Y}$
a set of winning counts $\mathcal{M}$

while existing data points to be trained **do**
  Input a data point $\mathbf{x}$ ($\mathbf{x} \in \mathcal{X}_c$).
  if the number of active nodes λ is not defined **or** $|\mathcal{Y}| < \lambda/2$ **or** $V_{\text{threshold}}$ is not calculated **then**
    Create a node as $\mathbf{y}_{|\mathcal{Y}|+1} = \mathbf{x}$ and set $M_{|\mathcal{Y}|+1} = 1$.
    Update $\mathcal{Y}$, $\mathcal{M}$, and $\mathcal{A}$ by inserting the new node.
    Calculate $\sigma_{|\mathcal{Y}|+1}$ with $\mathcal{A}$ by (5) and update $\mathcal{S}$.
    Calculate $\mathbf{R}$ in (7) and $D = \det(\mathbf{R})$ by (6).
    if $D < 1.0\mathrm{e}{-6}$ **then**
      Calculate the number of active nodes as $\lambda = 2|\mathcal{A}|$ and $V_{\text{threshold}}$ by (8).
    else
      Set the number of active nodes as $\lambda = \infty$.
  else
    Select the 1st and 2nd winner indices by (9) and (10) and calculate V_{s_1} and V_{s_2} in (11) and (12).
    if $V_{\text{threshold}} < V_{s_1}$ **then**
      /* Case I */
      Create a node at $\mathbf{x}$ and initialize it in the same manner as above.
    else
      /* Case II and Case III */
      Update M_{s_1} by (16) and update $\mathbf{y}_{s_1}$ by (17).
      Update $\mathcal{A}$ by inserting the 1st winner node.
      if $V_{s_2} \leq V_{\text{threshold}}$ **then**
        Update $\mathbf{y}_{s_2}$ by (18).
    if $\lambda < \infty$ **and** $|\mathcal{A}| > \lambda/2$ **then**
      Delete the oldest active node from $\mathcal{A}$.

C. CA+

CA+ is an ART-based clustering algorithm that supports online learning. During learning, a node set $\mathcal{Y}$ and a winning count set $\mathcal{M}$ are updated in an incremental manner. CA+ also estimates a similarity threshold $V_{\text{threshold}}$ from training data, and manual tuning of the vigilance parameter is not required. The overall learning procedure of CA+ is described in Algorithm 1.

1) Diversity and the Active Node Set: CA+ keeps an active node set $\mathcal{A} \subseteq \mathcal{Y}$. The size of $\mathcal{A}$ determines the recalculation interval λ used for updating $V_{\text{threshold}}$. The diversity of $\mathcal{A}$ is assessed by a DPP-based criterion [12]. The diversity is defined as

$$D = \det(\mathbf{R}), \tag{6}$$

where

$$\mathbf{R} = \left[\exp\left(1 - \mathrm{CIM}(\mathbf{y}_i, \mathbf{y}_j, \sigma)\right) \right]_{1 \leq i,j \leq |\mathcal{A}|} . \tag{7}$$

CA+ sets $\lambda = 2|\mathcal{A}|$ when $D < 1.0\mathrm{e}{-6}$. Otherwise, CA+ does not fix λ and continues to expand $\mathcal{A}$.

2) Similarity threshold: After λ is determined, CA+ calculates the similarity threshold from pairwise CIM values within $\mathcal{A}$ as

$$V_{\text{threshold}} = \frac{1}{\lambda} \sum_{\mathbf{y}_i \in \mathcal{A}} \min_{\mathbf{y}_j \in \mathcal{A} \setminus \mathbf{y}_i} \left[\text{CIM} \left(\mathbf{y}_i, \mathbf{y}_j, \text{mean}(\mathcal{S}) \right) \right]. \quad (8)$$

Here, $\mathcal{S} = \{\sigma_1, \ldots, \sigma_{|\mathcal{Y}|}\}$ stores the bandwidth values for CIM. Each σ_i is computed by (4) and (5) when a new node is added.

3) Winner selection and vigilance test: For each input $\mathbf{x}$, CA+ computes the CIM dissimilarity to all nodes in $\mathcal{Y}$ and it selects the 1st and 2nd winner indices as

$$s_1 = \arg\min_{\mathbf{y}_i \in \mathcal{Y}} \left[\text{CIM} \left(\mathbf{x}, \mathbf{y}_i, \text{mean}(\mathcal{S}) \right) \right], \quad (9)$$

$$s_2 = \arg\min_{\mathbf{y}_i \in \mathcal{Y} \setminus \{\mathbf{y}_{s_1}\}} \left[\text{CIM} \left(\mathbf{x}, \mathbf{y}_i, \text{mean}(\mathcal{S}) \right) \right]. \quad (10)$$

Let V_{s_1} and V_{s_2} denote the CIM dissimilarities between $\mathbf{x}$ and the corresponding winner nodes $\mathbf{y}_{s_1}$ and $\mathbf{y}_{s_2}$ as

$$V_{s_1} = \text{CIM} \left(\mathbf{x}, \mathbf{y}_{s_1}, \text{mean}(\mathcal{S}) \right), \quad (11)$$

$$V_{s_2} = \text{CIM} \left(\mathbf{x}, \mathbf{y}_{s_2}, \text{mean}(\mathcal{S}) \right). \quad (12)$$

The vigilance test compares V_{s_1} and V_{s_2} with $V_{\text{threshold}}$ and decides whether to create a new node or to update existing nodes. The following three cases are considered:

$$\text{Case I} \quad : \; V_{\text{threshold}} < V_{s_1} \leq V_{s_2}, \quad (13)$$

$$\text{Case II} \quad : \; V_{s_1} \leq V_{\text{threshold}} < V_{s_2}, \quad (14)$$

$$\text{Case III} : \; V_{s_1} \leq V_{s_2} \leq V_{\text{threshold}}. \quad (15)$$

Case I indicates insufficient match to existing nodes and it triggers node creation. Case II updates only the 1st winner node. Case III updates both winner nodes with different learning rates.

4) Creation and update of nodes: In Case I, CA+ creates a new node $\mathbf{y}_{|\mathcal{Y}|+1} = \mathbf{x}$ and initializes its winning count as $M_{|\mathcal{Y}|+1} = 1$.

In Case II, CA+ increments the winning count of the 1st winner node and update the node as

$$M_{s_1} \leftarrow M_{s_1} + 1, \quad (16)$$

$$\mathbf{y}_{s_1} \leftarrow \mathbf{y}_{s_1} + \frac{1}{M_{s_1}} \left(\mathbf{x} - \mathbf{y}_{s_1} \right). \quad (17)$$

In Case III, CA+ applies the Case II update and the 2nd winner node is additionally updated as

$$\mathbf{y}_{s_2} \leftarrow \mathbf{y}_{s_2} + \frac{1}{100 M_{s_2}} \left(\mathbf{x} - \mathbf{y}_{s_2} \right). \quad (18)$$

CA+ updates the active node set $\mathcal{A}$ by inserting the newly created node or the updated winner node. After λ is fixed, CA+ keeps $|\mathcal{A}| = \lambda/2$ by removing the oldest active node when necessary.

5) CA+ Prediction Procedure: The prediction procedure assigns a label to an unlabeled sample based on the generated nodes. Given the unlabeled sample $\mathbf{x}$, CA+ identifies the most similar node and assigns the corresponding node index as the predicted label.

D. Torque Clustering

Torque Clustering (TC) is a hierarchical clustering method that first constructs a hierarchical structure and then automatically determines the final partition and the number of clusters by cutting connections based on torque. In TC, the number of data points within a cluster is regarded as its mass, and torque is defined using both inter-cluster distances and these masses. Torque is used to evaluate the abnormality of connections. The number of clusters can be determined automatically, and it can also be specified by stopping the cutting process at the desired number.

1) Definition and Initialization of Mass: When the input is a set of data points, each data point is initialized as a cluster with mass $\theta_i = 1$. Thereafter, the mass of a cluster is updated as the number of data points contained in the cluster. A mass value can also be assigned to each node.

2) Nearest-Neighbor Cluster Identification and Connection Generation: For each cluster C_i in $\mathbf{\Gamma} = (C_1, C_2, ..., C_K)$, its nearest neighbor N_i is determined. Let ζ_i be the representative point of cluster C_i. Distances to the other clusters are computed using the representative points:

$$N_i = \arg \min_{j \in \{1,\ldots,K\} \setminus \{i\}} \left\| \boldsymbol{\zeta}_i - \boldsymbol{\zeta}_j \right\|_2. \quad (19)$$

After determining N_i, a connection is created between clusters C_i and C_{N_i}, and the connection graph G is updated.

Then, the connected components of the constructed connection graph are computed, and each connected component is merged into a single cluster. After the merging step, the mass of each cluster is aggregated according to (20), and a new cluster set $\mathbf{\Gamma}'$ is defined by (21).

$$\begin{cases} \theta_i \leftarrow \theta_i + \theta_{N_i}, & \text{if } \theta_i \leq \theta_{N_i}, \\ \theta_{N_i} \leftarrow \theta_i + \theta_{N_i}, & \text{if } \theta_i > \theta_{N_i}. \end{cases} \quad (20)$$

$$\mathbf{\Gamma}' = \Phi(G). \quad (21)$$

$\Phi(\cdot)$ is a function that treats each connected component of a graph as a single cluster. The same procedures in Step 2 are repeatedly applied to the new cluster set $\mathbf{\Gamma}'$, and the hierarchy is constructed until the number of clusters becomes 1 at the top level.

3) Detection and Removal of Abnormal Connections Based on Torque: For each connection generated during the hierarchical construction process, the torque τ_i is computed based on the product of the masses of the two endpoint clusters and the squared inter-cluster distance by (22).

$$\tau_i = M_i \times D_i, \quad (22)$$

$$M_i = \theta_i \times \theta_{N_i}, \quad (23)$$

$$D_i = \left\{ d(\zeta_i, \zeta_{N_i}) \right\}^2. \quad (24)$$

A list is created by sorting all connections in descending order of torque, and the torque difference between adjacent connections, TGap_i, is computed as (25).

$$\text{TGap}_i = \frac{\tau_i}{\tau_{i+1}}. \quad (25)$$

Among the connections that satisfy the conditions in (26)–(28) the position at which TGap_i is maximized is used as the boundary, and the higher-ranked connections are removed as abnormal connections.

$$M_i \geq \mathrm{mean}(\boldsymbol{M}_{\mathbf{all}}), \tag{26}$$

$$D_i \geq \mathrm{mean}(\boldsymbol{D}_{\mathbf{all}}), \tag{27}$$

$$\tau_i \geq \mathrm{mean}(\boldsymbol{\tau}_{\mathbf{all}}). \tag{28}$$

Here, $\boldsymbol{M}_{\mathbf{all}}$, $\boldsymbol{D}_{\mathbf{all}}$, and $\boldsymbol{\tau}_{\mathbf{all}}$ store the mass term, the distance term, and the torque values for all connections, respectively.

Finally, the connected components of the remaining connection graph are output as the final clusters $\boldsymbol{\Gamma}$.

III. Proposed Algorithm

The proposed method, CA+TC, is a two-stage clustering framework that combines CA+ and TC. CA+ first generates a set of representative nodes from an input dataset. TC then groups these nodes by constructing a hierarchical structure based on inter-node relationships. In the final step, connections are ranked in descending order of torque, and a fixed fraction of the highest-torque connections is removed to determine the final clusters.

A set of data points $\mathcal{X}_c$ is first processed by CA+ to generate a node set $\mathcal{Y}$. CA+ also provides the winning count set $\mathcal{M}$, where each element represents how often the corresponding node becomes the winner during CA+ learning. TC is then applied to $\mathcal{Y}$ with node masses specified by $\mathcal{M}$. The hierarchical construction and merging procedures follow the standard TC process, and this process generates the set of connections for torque-based cutting.

A torque value is computed for each connection generated during the hierarchy construction. All connections are collected and sorted in descending order of torque. Let m denote the total number of connections, and let $\rho \in (0, 1)$ denote the cutting ratio. The number of connections to be removed is defined as

$$L = \lceil \rho m \rceil. \tag{29}$$

Let τ_L denote the L-th largest torque in the sorted list. All connections with torque values not smaller than τ_L are removed. This rule replaces the original torque-gap criterion in TC. Finally, connected components of the remaining graph are extracted, and they are output as the final clusters. The overall learning procedure of the proposed method is summarized in Algorithm 2.

Algorithm 2: Learning procedure of CA+TC

Input:
a set of data points $\mathcal{X}_c \in \mathbb{R}^{n \times D}$, the cutting ratio $\rho \in (0, 1)$.
Output:
Cluster $\boldsymbol{\Gamma} = (C_1, C_2, ..., C_K)$.

Initialize $\mathcal{Y} \leftarrow \emptyset$ and $\mathcal{M} \leftarrow \emptyset$.
for $i = 1$ **to** n **do**
  Update $\mathcal{Y}$ and $\mathcal{M}$ by the CA+ learning procedure with input $\mathbf{x}_i$.
Initializing connected graph G.
Constructing cluster set $\boldsymbol{\Gamma} = \{\zeta_i\}_{i=1}^{l}$ from $\mathcal{Y}$, where each node $\mathbf{y}_i \in \mathcal{Y}$ is regarded as an initial cluster ζ_i and $l = |\mathcal{Y}|$.
Set the cluster mass of each ζ_i to $\theta_i \leftarrow M_i$ using $\mathcal{M}$, where $\Theta = \{\theta_i\}_{i=1}^{l}$.
Construct $\boldsymbol{D_{all}} \in \mathbb{R}^{l \times l}$ from $\mathcal{Y}$ as (24).
Initialize $\boldsymbol{M}_{\mathbf{all}} \leftarrow \emptyset$ and $\boldsymbol{D}_{\mathbf{all}} \leftarrow \emptyset$.
while $|\Gamma| > 1$ **do**
  Computing the mass θ_i of each cluster ζ_i in $\boldsymbol{\Gamma}$, where $\Theta = \{\theta_i\}_{i=1}^{|\Gamma|}$.
  Searching the nearest cluster of ζ_{N_i} according to $\boldsymbol{D_{all}}$ or by (19)
  Generating the connections, updating G, and updating the cluster mass by summation in (20).
  Computing the two properties M_i and D_i by (23) and (24), and saving these to $\boldsymbol{M}_{\mathbf{all}}$ and $\boldsymbol{D}_{\mathbf{all}}$, respectively.
  Computing the connected components of G to update the cluster set $\boldsymbol{\Gamma}$ by (21).
Computing the torque values $\boldsymbol{\tau_{all}}$ based on $\boldsymbol{M}_{\mathbf{all}}$ and $\boldsymbol{D}_{\mathbf{all}}$ by (22).
Set L by (29) and let τ_L be the L-th largest value in $\boldsymbol{\tau_{all}}$.
Updating G by removing all connections whose torque satisfies $\tau_i \geq \tau_{\mathrm{L}}$.
Computing the connected components of G to obtain the final cluster $\boldsymbol{\Gamma} = (C_1, C_2, ..., C_K)$.

IV. Simulation Experiments

CA+TC is evaluated under different cutting ratios ρ, and the best-performing setting is then compared with the baseline methods.

A. Dataset

In this experiment, evaluations were conducted on 19 real-world datasets from public repositories. Table I summarizes the dataset statistics in terms of the number of samples, attributes, and classes. The selected datasets cover a wide range of sample sizes, dimensionalities, and class counts, which enables a comprehensive comparison across diverse clustering scenarios.

B. Experimental Conditions

In the experiments, CA+, CA+TC, Self-Organizing Incremental Neural Network+ (SOINN+) [13], and CIM-based ART Edge with Age (CAEA) [14] are online methods that process data points sequentially, whereas TC is a batch method.

ARI and AMI are used to evaluate within-cluster class homogeneity and class completeness. The ARI and AMI values are compared using a Critical Distance (CD) diagram. Each hyperparameter setting is executed 30 times, and the averages of ARI and AMI are computed.

C. Hyperparameter Settings in CA+TC

This section compares the ARI and AMI of CA+TC across different hyperparameter settings. For CA+TC, the cutting ratio ρ for removing connections among the merged clusters is varied from 0.1 to 0.9 with a step size of 0.1.

The clustering performance of CA+TC across different hyperparameter settings is further summarized by the CD diagrams for ARI and AMI in Fig. 1(a) and Fig. 1(b). Due to page limitations, this section reports only the results of the

TABLE I: Statistics of real-world datasets

Dataset	#Samples	#Attributes	#Classes
Iris	150	4	3
Wine	178	13	3
Seeds	210	7	2
Glass	214	9	6
Newthyroid	215	5	3
Dermatology	366	34	6
Pima	768	8	2
Mice Protein	1,077	77	8
Binalpha	1,404	320	36
Yeast	1,484	8	10
Semeion	1,593	256	10
MSRA25	1,799	64	12
Image Segmentation	2,310	19	7
Rice	3,810	7	2
TUANDROMD	4,464	241	2
Phoneme	5,404	5	2
Texture	5,500	40	11
OptDigits	5,620	64	10
Statlog	6,435	36	6

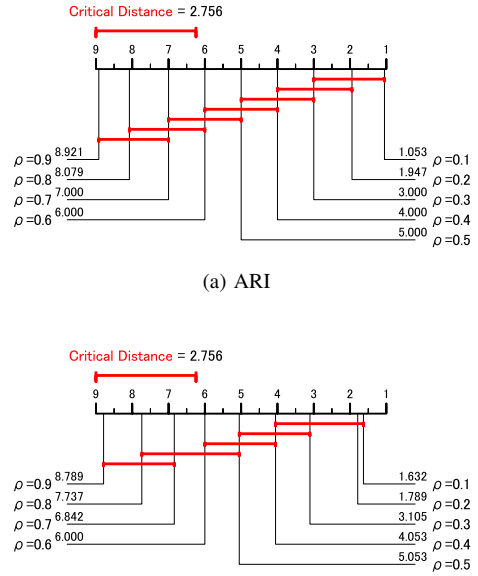
statistical significance tests based on the average ARI and AMI. For statistical comparisons, the Friedman test is applied at a significance level of 0.05 to evaluate whether all methods perform equally across datasets. When the Friedman test rejects this null hypothesis, pairwise differences are analyzed using the Nemenyi post-hoc test based on the average ranks. In the CD diagrams, methods connected by a red line do not show a statistically significant difference.

According to Fig. 1(a) and Fig. 1(b), the results for $\rho = 0.1$ achieve better ARI and AMI than those for the other hyperparameter settings. The performance decreases as ρ increases. When ρ is set to a small value, TC removes fewer connections than it does with a larger ρ, and the resulting clusters become more coherent. This tendency alleviates over-segmentation and leads to improved ARI and AMI.

D. Comparison with Other Methods

1) Compared Algorithms: The compared methods include TC [5], CA+ [9], SOINN+ [13] , CAEA [14]. SOINN+ is an incremental self-organizing network that constructs a topological graph of prototype nodes and edges. It can perform clustering without pre-specifying the number of clusters k, and it prunes irrelevant connections and removes weakly supported nodes, which improves robustness to noise and suppresses unnecessary prototypes. CAEA extends the CA algorithm by introducing edges and an edge-age based deletion mechanism. It uses CIM similarity and a vigilance threshold to decide whether to create a new node or update existing nodes and edges, and it removes isolated nodes every λ inputs to reduce noise.

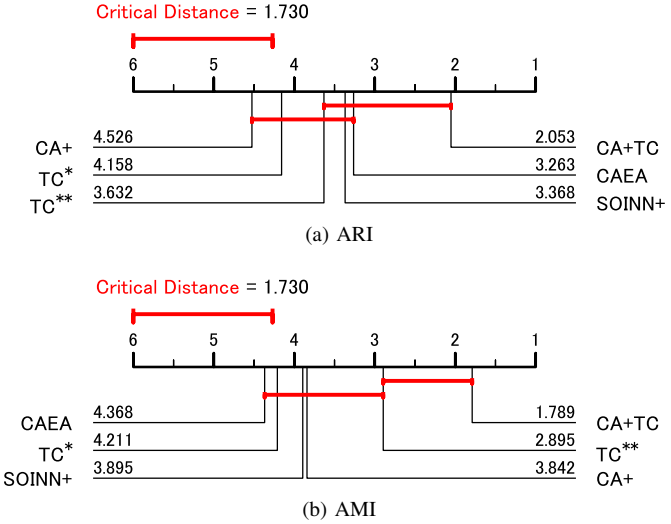
For the baseline methods, SOINN+ and CA+ are used without setting hyperparameters. For CAEA, the two integer hyperparameters, namely the maximum edge age before deletion $a_{\max}$ and the deletion interval for isolated nodes λ, are determined by Bayesian optimization. For TC, the number of clusters k is set to the number of classes in each dataset. For CA+TC, the cutting ratio ρ is fixed to 0.1, because



(a) ARI

(b) AMI

Fig. 1: CD diagrams for Hyperparameter Settings: (a) ARI and (b) AMI.



(a) ARI

(b) AMI

Fig. 2: CD diagrams for the compared methods: (a) ARI and (b) AMI.

$\rho = 0.1$ achieved the best ARI and AMI in the hyperparameter comparison.

2) Results: First, the clustering performance is compared.

Fig. 2(a) shows the CD diagram for ARI across the five methods, and Fig. 2(b) shows the CD diagram for AMI. In these figures, TC* denotes TC with k set to the number of classes in each dataset, whereas TC** denotes TC without specifying k. The CD diagrams indicate that CA+TC achieves the lowest average ranks for both ARI and AMI. For ARI, CA+TC is statistically significant against TC with k and CA+. For AMI, CA+TC is statistically significant against all the other methods except TC without k.

TABLE II: Number of Clusters for Each Method

Dataset	#Classes	CAEA	SOINN+	CA+	TC without k	CA+TC
Iris	3	10.2 (3.5)	9.2 (3.7)	47.3 (11.3)	3.1 (1.0)	4.2 (1.1)
Wine	3	2.2 (0.8)	8.3 (5.0)	122.4 (8.6)	14.7 (4.7)	11.8 (1.0)
Seeds	2	6.2 (2.8)	8.0 (4.3)	84.9 (18.8)	5.5 (1.0)	7.9 (1.9)
Glass	6	8.9 (1.9)	9.8 (4.2)	67.0 (10.2)	3.0 (2.0)	6.1 (1.1)
Newthyroid	3	7.9 (2.0)	11.4 (6.2)	64.0 (15.1)	5.3 (1.2)	5.9 (1.5)
Dermatology	6	8.8 (2.1)	16.5 (7.3)	119.1 (20.0)	7.3 (3.3)	11.4 (2.0)
Pima	2	5.3 (3.1)	12.1 (5.5)	213.2 (35.3)	32.6 (9.9)	20.7 (3.6)
Mice Protein	8	7.2 (2.5)	18.0 (7.6)	190.9 (31.3)	6.6 (1.2)	18.6 (3.2)
Binalpha	36	3.9 (1.9)	37.7 (13.4)	406.1 (38.2)	59.3 (27.0)	40.0 (3.9)
Yeast	10	7.4 (3.2)	6.3 (2.9)	270.6 (69.3)	24.5 (2.2)	26.4 (7.0)
Semeion	10	7.5 (5.7)	41.2 (9.6)	445.7 (44.3)	20.2 (9.3)	44.0 (4.5)
MSRA25	12	18.5 (4.0)	45.5 (12.4)	254.2 (55.7)	108.3 (36.7)	24.9 (5.6)
Image Segmentation	7	10.7 (4.8)	9.5 (5.6)	178.5 (50.8)	241.7 (142.1)	17.3 (5.0)
Rice	2	2.8 (3.7)	16.4 (6.1)	611.1 (40.5)	17.0 (0.0)	60.6 (4.1)
TUANDROMD	2	13.9 (2.8)	29.9 (10.9)	317.1 (84.7)	41.6 (18.9)	32.4 (8.2)
Phoneme	2	2.2 (1.9)	10.0 (5.2)	238.0 (59.1)	13.6 (5.3)	23.2 (5.9)
Texture	11	5.6 (1.6)	18.0 (6.7)	185.9 (53.5)	115.9 (34.5)	18.1 (5.4)
OptDigits	10	5.6 (6.0)	55.4 (17.0)	427.1 (143.2)	297.6 (181.2)	42.2 (14.4)
Statlog	6	46.7 (2.8)	39.5 (19.8)	408.0 (145.8)	53.1 (13.5)	40.3 (14.5)

Next, the numbers of generated nodes in CAEA, SOINN+, CA+, and CA+TC are compared. Table II reports the mean and standard deviation of the cluster counts for each method. CA+ often generates a very large number of nodes, which exceeds the number of classes by a wide margin on many datasets. In contrast, CA+TC yields a smaller node count and shows lower variability than CA+. In addition, CA+TC does not generate fewer nodes than the number of classes in the evaluated datasets, which helps avoid performance degradation caused by overly aggressive merging.

Based on the above results, CA+TC improves clustering performance and mitigates over-segmentation. A plausible explanation for the observed improvement is the node grouping step. This step yields a clearer cluster structure by merging redundant nodes and alleviating over-segmentation in the node set generated by CA+. In addition, the mass definition in TC may reinforce this effect. When the mass is defined by the CA+ winning counts $\mathcal{M}$, the mass term becomes large for connections whose endpoints have high winning counts, and the corresponding torque τ becomes large. Since TC preferentially cuts connections with large torque as abnormal connections, such cutting tends to remove connections that bridge major structures in the data distribution. As a result, connections along the main structure of the data distribution are more likely to remain after cutting, which leads to a more coherent partition.

V. Conclusion

This paper proposes CA+TC, which addresses excessive node generation in CA+ and extends the applicability of TC to large-scale datasets. These improvements lead to better clustering performance than CA+. CA+TC outperforms the other methods in both ARI and AMI, and the CD diagram indicates a statistically significant improvement in AMI over the other methods. In addition, the table of the number of clusters shows that CA+TC reduces the number of clusters compared with CA+.

As future work, an algorithm will be developed to automatically estimate the cutting ratio by using an inter-cluster separability measure.

References

[1] Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[2] Geoffrey J McLachlan, Sharon X Lee, and Suren I Rathnayake. Finite mixture models. *Annual Review of Statistics and its Application*, 6:355–378, 2019.

[3] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of KDD*, volume 96, pages 226–231, 1996.

[4] Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8934–8943, 2019.

[5] Jie Yang and Chin-Teng Lin. Autonomous clustering by fast find of mass and distance peaks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 47(7):5336–5349, January 2025.

[6] Gail A Carpenter, Stephen Grossberg, and David B Rosen. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4(6):759–771, 1991.

[7] Boaz Vigdor and Boaz Lerner. The Bayesian ARTMAP. *IEEE Transactions on Neural Networks*, 18(6):1628–1644, 2007.

[8] Naoki Masuyama, Chu Kiong Loo, and Stefan Wermter. A kernel Bayesian adaptive resonance theory with a topological structure. *International Journal of Neural Systems*, 29(5):1850052 (20 pages), 2019.

[9] Naoki Masuyama, Yusuke Nojima, Yuichiro Toda, Chu Kiong Loo, Hisao Ishibuchi, and Naoyuki Kubota. Privacy-preserving continual federated clustering via adaptive resonance theory. *IEEE Access*, 12:139692–139710, September 2024.

[10] Weifeng Liu, Puskal P Pokharel, and José C Príncipe. Correntropy: Properties and applications in non-Gaussian signal processing. *IEEE Transactions on Signal Processing*, 55(11):5286–5298, 2007.

[11] Bernard W Silverman. *Density Estimation for Statistics and Data Analysis*, volume 26 of *Monographs on Statistics and Applied Probability*. Chapman and Hall, London, 1986.

[12] Alex Kulesza and Ben Taskar. Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286, 2012.

[13] Chayut Wiwatcharakoses and Daniel Berrar. SOINN+, a self-organizing incremental neural network for unsupervised learning from noisy data streams. *Expert Systems with Applications*, 143:113069, 2020.

[14] Naoki Masuyama, Narito Amako, Yuna Yamada, Yusuke Nojima, and Hisao Ishibuchi. Adaptive resonance theory-based topological clustering with a divisive hierarchical structure capable of continual learning. *IEEE Access*, 10:68042–68056, June 2022.