IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 10 September 2024, accepted 17 September 2024, date of publication 24 September 2024, date of current version 4 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3467114

RESEARCH ARTICLE

Privacy-Preserving Continual Federated Clustering via Adaptive Resonance Theory

NAOKI MASUYAMA[1], (Member, IEEE), YUSUKE NOJIMA[1], (Member, IEEE), YUICHIRO TODA[2], (Member, IEEE), CHU KIONG LOO[3], (Senior Member, IEEE), HISAO ISHIBUCHI[4], (Fellow, IEEE), AND NAOYUKI KUBOTA[5], (Senior Member, IEEE)

[1]Department of Core Informatics, Graduate School of Informatics, Osaka Metropolitan University, Sakai, Osaka 599-8531, Japan
[2]Faculty of Environmental, Life, Natural Science and Technology, Okayama University, Okayama 700-8530, Japan
[3]Department of Artificial Intelligence, Faculty of Computer Science and Information Technology, Universiti Malaya, Kuala Lumpur 50603, Malaysia
[4]Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen 518055, China
[5]Graduate School of Systems Design, Department of Mechanical Systems Engineering, Tokyo Metropolitan University, Asahigaoka, Hino, Tokyo 191-0065, Japan

Corresponding author: Naoki Masuyama (masuyama@omu.ac.jp)

This work was supported in part by Japan Society for the Promotion of Science (JSPS) KAKENHI under Grant 22K12199 and Grant 22H03664, in part by the National Natural Science Foundation of China under Grant 62250710163 and Grant 62376115, in part by Guangdong Provincial Key Laboratory under Grant 2020B121201001, in part by the Japan Science and Technology (JST) [Moonshot Research and Development (RnD)] under Grant JP-MJMS2034, and in part by the Tokyo Metropolitan University (TMU) Local 5G Research Support.

ABSTRACT With the increasing importance of data privacy protection, various privacy-preserving machine learning methods have been proposed. In the clustering domain, various algorithms with a federated learning framework (i.e., federated clustering) have been actively studied and showed high clustering performance while preserving data privacy. However, most of the base clusterers (i.e., clustering algorithms) used in existing federated clustering algorithms need to specify the number of clusters in advance. These algorithms, therefore, are unable to deal with data whose distributions are unknown or continually changing. To tackle this problem, this paper proposes a privacy-preserving continual federated clustering algorithm. In the proposed algorithm, an adaptive resonance theory-based clustering algorithm capable of continual learning is used as a base clusterer. Therefore, the proposed algorithm inherits the ability of continual learning. Experimental results with synthetic and real-world datasets show that the proposed algorithm has superior clustering performance to state-of-the-art federated clustering algorithms while realizing data privacy protection and continual learning ability. The source code is available at https://github.com/Masuyama-lab/FCAC.

INDEX TERMS Self-organizing feature maps, adaptive resonance theory, continual learning, federated clustering, local ϵ-differential privacy.

I. INTRODUCTION

In a society with advanced information technology, privacy protection techniques in data utilization are becoming increasingly important. Various methods have been proposed to protect data privacy. Secure computation and secret sharing are widely used as cryptographic methods [1]. These methods realize the sharing of information while preserving anonymity and confidentiality by using encryption keys. As mathematical-based methods, anonymization improves data privacy by converting or removing personally identifiable information [2]. Differential privacy adds noise to training data or outputs of a trained model. This is to make it difficult to determine whether a particular individual exists in the training data [3], [4]. Specifically, the approach that adds noise to the training data is called local differential privacy, while the approach that adds noise to the output of the trained model is called global differential privacy. In the machine learning domain, federated learning is known as a privacy-preserving distributed learning method [5]. Federated learning performs learning from distributed data across multiple clients without

The associate editor coordinating the review of this manuscript and approving it for publication was Yunlong Cai.

aggregating the data on a single server. The parameters of a trained model in each client are then consolidated on a model in a server. In addition, federated learning can further improve data privacy by applying differential privacy [6], [7].

In recent years, the importance of data privacy has also increased in the clustering domain [2], [8], [9]. Among privacy-preserving clustering algorithms, federated clustering (i.e., an clustering algorithm applies a federated learning framework) has attracted much attention because of its usefulness in practical applications [10]. As a base clusterer (i.e., a clustering algorithm) of federated clustering, in general, centroid-based clustering algorithms such as k-means, fuzzy c-means, and Gaussian Mixture Model (GMM) are often used [10], [11], [12], [13]. Although centroid-based clustering algorithms are simple and highly applicable, these algorithms need to specify the number of centroids in advance. This drawback makes it difficult to apply these algorithms to data whose distributions are unknown and/or continually changing.

Among clustering algorithms, Adaptive Resonance Theory (ART)-based approaches are capable of continual learning without catastrophic forgetting by adaptively generating centroids (nodes) depending on the distributions of given data [14]. In particular, ART-based clustering algorithms with Correntropy-Induced Metric (CIM) [15] as a similarity measure show faster and more stable self-organizing ability than other clustering algorithms [16], [17], [18], [19], [20], [21]. In general, ART-based clustering algorithms generate a number of representative points (nodes) from given data. Thus, data privacy is more or less considered since the given data itself is not retained. However, the ability to protect data privacy is insufficient because no data privacy protection techniques are explicitly applied to those algorithms.

Recent growth of Internet of Things (IoT) technology has enabled the creation and acquisition of a wide variety of data that are generated in a distributed and continual manner. Such data are regarded as important economic resources that can be used for marketing, finance, and IoT solution development. In the rapid utilization of economic resources, a technique that satisfies privacy-preserving, no prior assumptions, and efficient continual learning is highly valuable. However, clustering algorithms that simultaneously realize privacy-preserving and efficient continual learning in situations where data are distributed have not been adequately discussed.

With the motivation of the above discussions, this paper proposes a new privacy-preserving federated clustering algorithm, called Federated Clustering via ART-based Clustering (FCAC), which applies local differential privacy to an ART-based clustering algorithm in a federated learning framework. FCAC performs clustering for distributed training data across multiple clients without aggregating the data on a single server while preserving data privacy. In FCAC, CIM-based ART with Edge (CAE) [21] after a minor modification, which is called CAE for Federated Clustering (CAE$_{\mathrm{FC}}$), is used as a base clusterer for a server. Although the original CAE is a state-of-the-art parameter-free ART-based topological clustering algorithm, CAE often generates a large number of nodes. Therefore, we introduce a minor modification for reducing the number of generated nodes. A base clusterer for each client of FCAC is CAE$_{\mathrm{FC}}$ without topology (i.e., edges), called CA+, which is first introduced in this paper. Since a trained model of a server is a clustering result, the server is required to continually generate well-separated clusters. In contrast, each client is required to generate a number of nodes that can continually and appropriately approximate the distributions of the training data in the client. This is because the generated nodes in each client are utilized as training data for the server. Therefore, we introduce CA+ as the base clusterer for each client, which only generates nodes from given data in each client. Thanks to CAE$_{\mathrm{FC}}$ and CA+, FCAC can adaptively, efficiently, and continually generate topological networks from the given data in each client. Note that continual learning is generally categorized into three scenarios: domain incremental learning, task incremental learning, and class incremental learning [22], [23]. Since CAE$_{\mathrm{FC}}$ and CA+ are capable of class incremental learning, FCAC inherits the ability of class incremental learning.

The contributions of this paper are summarized as follows:

(I) FCAC is proposed as a new privacy-preserving federated clustering algorithm capable of continual learning. FCAC explicitly considers the protection of data privacy by applying local differential privacy a federated learning framework. To the best of our knowledge, FCAC is the first ART-based privacy-preserving federated clustering algorithm.

(II) A new clustering algorithm called CA+, which is a variant of CAE$_{\mathrm{FC}}$, is introduced. The self-organizing ability of CA+ satisfies the demand of a client in FCAC, i.e., generated nodes by CA+ can continually and appropriately approximate the distributions of the training data in each client.

(III) Empirical studies show that FCAC has superior clustering performance to state-of-the-art algorithms while protecting data privacy and maintaining continual learning ability.

The paper is organized as follows. Section II presents a literature review for growing self-organizing clustering and privacy-preserving clustering algorithms. Section III presents the preliminary knowledge for CAE. Section IV explains the learning procedure of the proposed FCAC algorithm in detail. Section V presents extensive simulation experiments to evaluate its clustering performance by using synthetic and real-world datasets. Section VIII concludes this paper.

Table 1 summarizes the main notations used in FCAC and related functions/algorithms.

TABLE 1. Summary of notations.

Notation	Description
$\mathbf{x}$	d-dimensional data point
$\mathcal{X}_c$	set of data points for a client c ($\mathbf{x} \in \mathcal{X}_c$)
$\mathbf{y}_k$	k-th node
$\mathcal{Y}$	set of nodes ($\mathbf{y}_k \in \mathcal{Y}$)
$\lvert\mathcal{Y}\rvert$	number of nodes in $\mathcal{Y}$
$\mathbf{y}_{s_1}$	1st winner node
$\mathbf{y}_{s_2}$	2nd winner node
$\hat{C}(\cdot)$	correntropy
$\mathrm{CIM}(\cdot)$	correntropy-induced metric
σ	bandwidth of a kernel function in CIM
$\mathcal{S}$	set of bandwidths of a kernel function $\sigma \in \mathcal{S}$
$\mathcal{N}_k$	set of neighbor nodes of node $\mathbf{y}_k$
λ	number of active nodes
$\mathcal{A}$	set of active nodes
D	diversity of a set of active nodes
V_{s_1}	CIM value between $\mathbf{x}$ and $\mathbf{y}_{s_1}$
V_{s_2}	CIM value between $\mathbf{x}$ and $\mathbf{y}_{s_2}$
$\mathbf{R}$	matrix of pairwise similarities
$V_{\text{threshold}}$	similarity threshold (a vigilance parameter)
M_k	winning count of $\mathbf{y}_k$
$\mathcal{M}$	set of winning counts M_k ($M_k \in \mathcal{M}$)
$a(\mathbf{y}_k, \mathbf{y}_l)$	age of an edge between nodes $\mathbf{y}_k$ and $\mathbf{y}_l \in \mathcal{Y} \setminus \mathbf{y}_k$
$\mathcal{E}$	set of ages of edges ($a(\mathbf{y}_k, \mathbf{y}_l) \in \mathcal{E}$)
α_{del}	set of ages of deleted edges
$a_{\max}$	edge deletion threshold
CA+	base clusterer for a client
$\mathrm{CAE}_{\mathrm{FC}}$	base clusterer for a server
ϵ	privacy budget
Δf	local sensitivity
$L(\cdot)$	inverse cumulative density function
$\mathcal{X}'_c$	set of data points for a client c with differential privacy
C	number of clients for federatd clustering
$\mathbf{Y}^{\geq 75\text{th}}$	nodes above the 75th percentile of elements in $\mathcal{M}$
$\mathbf{Y}^{<75\text{th}}$	nodes below the 75th percentile of elements in $\mathcal{M}$
$\mathbf{Y}^{\text{CA+}}$	data points for a server (i.e., nodes generated by CA+)
$\mathcal{Y}^{\mathrm{CAE}_{\mathrm{FC}}}$	set of nodes generated by $\mathrm{CAE}_{\mathrm{FC}}$

II. LITERATURE REVIEW

A. GROWING SELF-ORGANIZING CLUSTERING ALGORITHMS

Clustering is useful in all fields that deal with data. Many clustering algorithms have been proposed [24], [25], [26], [27], [28] and applied to a variety of applications [29], [30], [31], [32], [33], [34], [35], [36]. Although many clustering algorithms have their success, some algorithms have a well-known drawback, namely the number of clusters/partitions has to be pre-specified. To solve this problem, growing self-organizing clustering algorithms such as Growing Neural Gas (GNG) [37] and Adjusted Self-Organizing Incremental Neural Network (ASOINN) [38] have been proposed. GNG and ASOINN adaptively generate topological networks (i.e., nodes and edges) for representing the distributions of given data. SOINN+ [39] is an ASOINN-based algorithm that can handle arbitrary data distributions in noisy data streams without any pre-defined parameters. However, since these algorithms permanently insert new nodes and edges for learning new information, there is a possibility of forgetting previously learned information (i.e., catastrophic forgetting). As a GNG-based algorithm, Grow When Required (GWR) [40] successfully avoids catastrophic forgetting by adding a node only when the state of the current network does not sufficiently match a new instance. One common problem of GWR and SOINN+ is that as the number of nodes in the topological network increases, the cost of calculating a threshold for each node increases, and therefore the learning efficiency decreases.

One promising approach for avoiding catastrophic forgetting is an ART-based algorithm [16], [17], [18], [41], [42]. In particular, algorithms that use CIM as a similarity measure have shown superior clustering performance to other clustering algorithms [14], [19], [43], [44]. A well-known drawback of ART-based algorithms is the specification of significantly data-dependent parameters such as a similarity threshold (i.e., a vigilance parameter). Several studies have proposed to avoid and/or suppress the effect of the above-mentioned drawback by using multiple vigilance values [45], by specifying the vigilance parameter indirectly [46], [47], and by adjusting some data-dependent parameters during the learning process [48]. A state-of-the-art parameter-free algorithm is CAE [21]. In CAE, a similarity threshold is calculated based on pairwise similarities by using well-diversified generated nodes. The diversified nodes are selected by a Determinantal Point Processes (DPP)-based criterion [49], [50] incorporating CIM.

B. PRIVACY-PRESERVING CLUSTERING ALGORITHMS

The protection of data privacy is generally realized by cryptograph-, mathematical-, and machine learning-based methods. Secure computation and secret sharing are widely used as cryptograph-based methods [1], [51], [52]. Secure computation is also known as secure multi-party computation, which allows multiple parties to compute a function over their data while keeping those data private. Secret sharing divides sensitive data into multiple parts to preserve data privacy. In general, secure computation is often time-consuming [53], while secret sharing needs to process the creation, distribution, and combination of secret shares [54].

As mathematical-based methods, anonymization provides a simple and efficient data privacy protection mechanism [2]. One problem with anonymization is that the original data can be estimated by combining it with specific information. Differential privacy provides mathematically-defined strict privacy protection [4]. Differential privacy protects sensitive information in training data by adding noise to the training data or the output of a trained model. In the clustering domain, local differential privacy is often applied thanks to its simple mechanism and mathematical guarantees for privacy protection [55], [56], [57], [58].

One recent successful machine learning-based method is federated learning [5], [7]. In the clustering domain, an algorithm with a federated learning framework is called federated clustering [10], [11], [59], [60]. k-FED [10] is a communication-efficient federated clustering algorithm that requires one-shot communication from clients to a server. In k-FED, each client performs k-means on local data and sends clustering results (i.e., centroids) to the server. The server performs k-means on the centroids sent from all clients. Federated Fuzzy c-Means (FedFCM) [12] is an algorithm similar to k-FED that utilizes fuzzy c-means [61]. Machine Unlearning Federated Clustering (MUFC) [62] introduced a novel sparse compressed multi-set aggregation scheme that satisfies a new privacy criterion. In MUFC, a server only receives the information on the centroids and the number of data points related to each centroid from each client. Thus, the server has no information on each data point.

Most of base clusterers used in existing federated clustering (and also cryptograph- and mathematical-based methods) are centroid-based algorithms such as GMM [13], k-means [10], [62], [63], [64], [65], fuzzy c-means [11], [12], and spectral clustering [66]. Although centroid-based clustering algorithms are simple and highly applicable, these algorithms require the number of clusters to be specified in advance. In a society with rapidly growing and diverse data, it is difficult to know the true number of clusters of the data a priori. Moreover, there is a possibility that the distributions of data and the number of clusters are frequently changed dynamically. The above-mentioned difficulty in each existing method emphasizes the significance of developing federated clustering algorithms capable of continual learning.

III. PRELIMINARY KNOWLEDGE

This section provides preliminary knowledge related to FCAC. First, local differential privacy is explained. Next, a similarity measure and a kernel density estimator used in CAE are explained. Last, the learning procedure of CAE is explained in detail.

A. LOCAL DIFFERENTIAL PRIVACY

Local differential privacy includes two approaches: randomized response and adding noise to the data itself [3], [4], [67]. In general, the former is applied to discrete values, while the latter is applied to continuous values. Since FCAC is a clustering algorithm, the latter approach is applied, i.e., adding noise to the data itself.

The definition of local differential privacy is as follows. An algorithm Ψ satisfies ϵ-differential privacy ($\epsilon > 0$) if and only if for any data points $\mathbf{x}$ and $\mathbf{x}'$ ($\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$), the following relation holds:

$$\Pr\left[\Psi(\mathbf{x}) = \mathbf{y}\right] \leq e^{\epsilon} \Pr\left[\Psi(\mathbf{x}') = \mathbf{y}\right], \ \forall \mathbf{y} \in \mathrm{Range}(\Psi), \quad (1)$$

where $\Pr[\cdot]$ is a probability, $\mathrm{Range}(\Psi)$ is every possible output of the algorithm Ψ, ad ϵ is a privacy budget. In general, $\epsilon = 0$ means perfect privacy, while $\epsilon = \infty$ means no privacy guarantee.

In this paper, local ϵ-differential privacy is realized by using a Laplace mechanism [3]. More specifically, the inverse cumulative density function of Laplace distribution [68] is used. Suppose that a set of data points $\mathcal{X}_c = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ is given, where $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d})$. A method of adding a noise to the jth dimension of a data point $\mathbf{x}_i$ is as follows:

$$x'_{i,j} = x_{i,j} + L(v_{i,j}) \quad (j = 1, 2, \ldots, d), \quad (2)$$

where

$$L(v_{i,j}) = \mu - \frac{\Delta f_j}{\epsilon} \mathrm{sgn}(v_{i,j}) \ln(1 - 2|v_{i,j}|), \quad (3)$$

here, $v_{i,j}$ is the random variable sampled from a uniform distribution $U(-0.5, 0.5)$, $\mathrm{sgn}(\cdot)$ is a signum function, $\ln(\cdot)$ is a natural logarithm function, μ is the mean of a Laplace distribution, and Δf_j is a local sensitivity of differential privacy for $x_{i,j}$.

The local sensitivity Δf_j is defined as follows:

$$\Delta f_j = \max_{n=1,2,\ldots,N} (x_{n,j}) - \min_{n=1,2,\ldots,N} (x_{n,j}). \quad (4)$$

B. CORRENTROPY AND CORRENTROPY-INDUCED METRIC

Correntropy [15] 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:

$$\hat{C}(\mathbf{x}, \mathbf{y}, \sigma) = \frac{1}{d} \sum_{i=1}^{d} \kappa_\sigma \left(x_i, y_i\right). \quad (5)$$

where $\kappa_\sigma(\cdot)$ denotes a positive definite kernel with a bandwidth σ. In this paper, we use the following Gaussian kernel in the correntropy:

$$\kappa_\sigma \left(x_i, y_i\right) = \exp\left[-\frac{(x_i - y_i)^2}{2\sigma^2}\right]. \quad (6)$$

A nonlinear metric called CIM is derived from the correntropy [15]. CIM quantifies the similarity between two data points $\mathbf{x}$ and $\mathbf{y}$ as follows:

$$\mathrm{CIM}\left(\mathbf{x}, \mathbf{y}, \sigma\right) = \left[1 - \hat{C}(\mathbf{x}, \mathbf{y}, \sigma)\right]^{\frac{1}{2}}, \quad (7)$$

here, since the Gaussian kernel in (6) does not have the coefficient $\frac{1}{\sqrt{2\pi}\sigma}$, the range of CIM is limited to [0, 1].

C. KERNEL DENSITY ESTIMATOR

In general, the bandwidth of a kernel function can be estimated from λ instances belonging to a certain distribution [69], which is defined as follows:

$$\mathbf{\Sigma} = U(F_\nu)\mathbf{\Gamma}\lambda^{-\frac{1}{2\nu+d}}, \quad (8)$$

$$U(F_\nu) = \left(\frac{\pi^{d/2} 2^{d+\nu-1} (\nu!)^2 R(F)^d}{\nu \kappa_\nu^2(F) \left[(2\nu)!! + (d-1)(\nu!!)^2\right]}\right)^{\frac{1}{2\nu+d}}, \quad (9)$$

where $\mathbf{\Gamma}$ denotes a rescale operator (d-dimensional vector) which is defined by a standard deviation of each of the d attributes among λ instances, ν is the order of a kernel, the single factorial of ν is calculated by the product of integer numbers from 1 to ν, the double factorial notation is defined as $(2\nu)!! = (2\nu-1)\cdot 5\cdot 3\cdot 1$ (commonly known as the odd factorial), $R(F)$ is a roughness function, and $\kappa_\nu(F)$ is the moment of a kernel. The details of the derivation of (8) and (9) can be found in [69]. In this paper, we use the Gaussian kernel for CIM. Therefore, $\nu = 2$, $R(F) = (2\sqrt{\pi})^{-1}$, and $\kappa_\nu^2(F) = 1$. Then, (9) is rewritten as follows:

$$\mathbf{H} = \left(\frac{4}{2+d}\right)^{\frac{1}{4+d}} \mathbf{\Gamma}\lambda^{-\frac{1}{4+d}}. \tag{10}$$

Equation (10) is known as the Silverman's rule [70]. Here, $\mathbf{H}$ contains the bandwidth of a kernel function in CIM.

D. CIM-BASED ART WITH EDGE: CAE

CAE is a parameter-free ART-based topological clustering algorithm capable of continual learning [21]. In general, ART-based algorithms have a data-dependent parameter such as a vigilance parameter (similarity threshold). In CAE, a similarity threshold is calculated based on a pairwise similarities among a certain number of nodes. The sufficient number of nodes for calculating the similarity threshold is estimated by a Determinantal Point Processes (DPP)-based criterion [49], [50]. In addition, an edge deletion threshold is estimated based on the age of each edge, which is inspired by the edge deletion mechanism of SOINN+ [39]. Empirical studies with the synthetic and real-world datasets showed that the clustering performance of CAE is superior to state-of-the-art parameter-free/fixed algorithms [21].

In CAE, the value of the similarity threshold has a significant impact on the clustering performance. In [21], the validity of the estimated similarity threshold is experimentally evaluated, and it has been confirmed that CAE shows high clustering performance on various datasets in both stationary and non-stationary environments by using the estimated threshold. Please refer to Section V-D in [21] for further information.

The following sections provide the learning processes of CAE step by step. Algorithm 1 summarizes the entire learning procedure of CAE.

1) ESTIMATION OF DIVERSITY OF NODES

In CAE, a similarity threshold is defined by a pairwise similarities among nodes (i.e., $\mathcal{Y}$). Therefore, the diversity of nodes for calculating the similarity threshold is important to obtain an appropriate threshold value, which leads to good clustering performance.

The diversity D of the active node set $\mathcal{A}$ is estimated by a DPP-based criterion [49], [50] incorporating CIM as follows:

$$D = \det(\mathbf{R}), \tag{11}$$

where

$$\mathbf{R} = \left[\exp\left(1 - \text{CIM}(\mathbf{y}_i, \mathbf{y}_j, \sigma)\right)\right]_{1\le i,j\le|\mathcal{A}|}. \tag{12}$$

Here, $\det(\mathbf{R})$ is the determinant of the matrix $\mathbf{R}$, and $\mathbf{R}$ is a matrix of pairwise similarities between nodes in $\mathcal{A}$. A bandwidth σ for CIM is calculated from $\mathbf{H}$ in (10) by using the node set $\mathcal{A}$. As in (10), $\mathbf{H}$ contains the bandwidth of a kernel function in CIM. In this paper, the median of $\mathbf{H}$ is used as the bandwidth of the Gaussian kernel in CIM, i.e.,

$$\sigma = \text{median}(\mathbf{H}). \tag{13}$$

In general, the diversity $D = 0$ means that the node set $\mathcal{A}$ is not diverse while $D > 0$ means $\mathcal{A}$ is diverse. In other words, the value of D becomes close to zero when a new node is created around the existing nodes.

In CAE, the value of λ is set as the two times of the number of nodes (i.e., $2|\mathcal{A}|$) when the diversity D satisfies $D < 1.0\text{e}{-6}$. If the number of nodes becomes smaller than $\lambda/2$ after a node deletion process, λ is calculated again.

As shown in a line 3 of Algorithm 1, the first $\lambda/2$ data points (i.e., $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_{\lambda/2}\}$) directly become nodes, i.e., $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{\lambda/2}\}$ where $\mathbf{y}_k = \mathbf{x}_k$ $(k = 1, 2, \ldots, \lambda/2)$. In addition, the bandwidth for the Gaussian kernel in CIM is assigned to each node, i.e., $\mathcal{S} = \{\sigma_1, \sigma_2, \ldots, \sigma_{\lambda/2}\}$ where $\sigma_1 = \sigma_2 = \cdots = \sigma_{\lambda/2}$.

The value of λ is automatically updated in the proposed CAE algorithm. In an active node set $\mathcal{A}$, λ nodes are stored. When a new node is added to $\mathcal{A}$, an old node is removed to maintain the active node set size as λ. The addition of a new node and the removal of an old node are explained later.

2) CALCULATION OF SIMILARITY THRESHOLD

The similarity threshold $V_{\text{threshold}}$ is calculated by the average of the minimum pairwise CIM values in the active node set $\mathcal{A}$ as follows:

$$V_{\text{threshold}} = \frac{1}{\lambda}\sum_{\mathbf{y}_i\in\mathcal{A}} \min_{\mathbf{y}_j\in\mathcal{A}\backslash\mathbf{y}_i} \left[\text{CIM}\left(\mathbf{y}_i, \mathbf{y}_j, \text{mean}(\mathcal{S})\right)\right], \tag{14}$$

where $\mathcal{S}$ is a set of bandwidths of the Gaussian kernel in CIM for $\mathcal{A}$. The bandwidth of each node in $\mathcal{A}$ is calculated by using (10) and (13) when a new node is created.

3) SELECTION OF WINNER NODES

During the learning process of CAE, every time a data point $\mathbf{x}$ is given, two nodes that have a similar state to $\mathbf{x}$ are selected from $\mathcal{Y}$, namely the 1st winner node $\mathbf{y}_{s_1}$ and the 2nd winner node $\mathbf{y}_{s_2}$. The winner nodes are determined based on the value of CIM as follows:

$$s_1 = \mathop{\arg\min}_{\mathbf{y}_i\in\mathcal{Y}} \left[\text{CIM}\left(\mathbf{x}, \mathbf{y}_i, \text{mean}(\mathcal{S})\right)\right], \tag{15}$$

$$s_2 = \mathop{\arg\min}_{\mathbf{y}_i\in\mathcal{Y}\backslash\{\mathbf{y}_{s_1}\}} \left[\text{CIM}\left(\mathbf{x}, \mathbf{y}_i, \text{mean}(\mathcal{S})\right)\right], \tag{16}$$

where s_1 and s_2 denote the indexes of the 1st and 2nd winner nodes, respectively. $\mathcal{S} = \{\sigma_1, \sigma_2, \ldots, \sigma_{|\mathcal{Y}|}\}$ is a set of bandwidths of the Gaussian kernel in CIM corresponding to a node set $\mathcal{Y}$.

Note that the 1st winner node $\mathbf{y}_{s_1}$ becomes a new active node, and the oldest node in the active node set $\mathcal{A}$ (i.e., λ nodes in $\mathcal{Y}$) is replaced by the new one.

Algorithm 1 Learning Procedure of CAE

```
Input:
a set of data points $\mathcal{X}_c$
Output:
a set of nodes $\mathcal{Y}$
a set of winning counts $\mathcal{M}$
a set of ages of edges $\mathcal{E}$
1  while existing data points to be trained do
2      Input a data point $\mathbf{x}$ ($\mathbf{x} \in \mathcal{X}_c$).
3      if the number of active nodes $\lambda$ is not defined or the number of nodes $|\mathcal{Y}|$ is smaller than $\lambda/2$ or a similarity threshold $V_{\text{threshold}}$ is not calculated then
4          Create a node as $\mathbf{y}_{|\mathcal{Y}|+1} = \mathbf{x}$, and update a set of nodes as $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{\mathbf{y}_{|\mathcal{Y}|+1}\}$.
5          Define a winning count as $M_{|\mathcal{Y}|+1} = 1$, and update a set of winning counts as $\mathcal{M} \leftarrow \mathcal{M} \cup \{M_{|\mathcal{Y}|+1}\}$.
6          Update a set of active nodes as $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathbf{y}_{|\mathcal{Y}|+1}\}$.
7          Calculate $\sigma_{|\mathcal{Y}|+1}$ with a set of active nodes $\mathcal{A}$, and update a set of bandwidths $\mathcal{S} \leftarrow \mathcal{S} \cup \{\sigma_{|\mathcal{Y}|+1}\}$.   // (10),(13)
           /* Estimation of Diversity of Nodes */
8          Calculate a pairwise similarity matrix $\mathbf{R}$.   // (12)
9          Calculate the diversity as $D = \det(\mathbf{R})$.   // (11)
10         if $D < 1.0\text{e}{-6}$ then
11             Calculate the number of active nodes as $\lambda = 2|\mathcal{A}|$.
               /* Calculation of Similarity Threshold */
12             Calculate a similarity threshold $V_{\text{threshold}}$.   // (14)
13         else
14             Set the number of active nodes as $\lambda = \infty$.
15     else
16         Select the 1st and 2nd nearest nodes from $\mathbf{x}$ (i.e., $\mathbf{y}_{s_1}$ and $\mathbf{y}_{s_2}$) based on CIM.   // (15),(16)
17         Calculate similarities between $\mathbf{x}$ and $\mathbf{y}_{s_1}, \mathbf{y}_{s_2}$ (i.e., $V_{s_1}$ and $V_{s_2}$).   // (17),(18)
           /* Vigilance Test and Creation/Update of Nodes and Edges */
18         if $V_{\text{threshold}} < V_{s_1}$ then
               /* Case I */
19             Create a node as $\mathbf{y}_{|\mathcal{Y}|+1} = \mathbf{x}$, and update a set of nodes as $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{\mathbf{y}_{|\mathcal{Y}|+1}\}$.
20             Define a winning count as $M_{|\mathcal{Y}|+1} = 1$, and update a set of winning counts as $\mathcal{M} \leftarrow \mathcal{M} \cup \{M_{|\mathcal{Y}|+1}\}$.
21             Update a set of active nodes as $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathbf{y}_{|\mathcal{Y}|+1}\}$.
22             Calculate $\sigma_{|\mathcal{Y}|+1}$ with a set of active nodes $\mathcal{A}$, and update a set of bandwidths $\mathcal{S} \leftarrow \mathcal{S} \cup \{\sigma_{|\mathcal{Y}|+1}\}$.   // (10),(13)
23         else
               /* Case II */
24             Update a winning count of the $s_1$ node as $M_{s_1} \leftarrow M_{s_1} + 1$.   // (22)
25             Update a node as $\mathbf{y}_{s_1} \leftarrow \mathbf{y}_{s_1} + \frac{1}{M_{s_1}}(\mathbf{x} - \mathbf{y}_{s_1})$.   // (23)
26             Update a set of active nodes as $\mathcal{A} \leftarrow \mathcal{A} \cup \{\mathbf{y}_{|\mathcal{Y}|+1}\}$.
27             for $\mathbf{y}_k \in \mathcal{N}_{s_1}$ do
28                 Update an age of an edge as $a(\mathbf{y}_{s_1}, \mathbf{y}_k) \leftarrow a(\mathbf{y}_{s_1}, \mathbf{y}_k) + 1$.   // (24)
29             if $V_{s_2} \leq V_{\text{threshold}}$ then
                   /* Case III */
30                 Reset an age of an edge as $a(\mathbf{y}_{s_1}, \mathbf{y}_{s_2}) \leftarrow 1$.   // (25)
31                 for $\mathbf{y}_k \in \mathcal{N}_{s_1}$ do
32                     Update a node as $\mathbf{y}_k \leftarrow \mathbf{y}_k + \frac{1}{10M_k}(\mathbf{x} - \mathbf{y}_k)$.   // (26)
           /* Estimation of Edge Deletion Threshold */
33         Update $\alpha \leftarrow$ a set of ages of edges which connect to $\mathbf{y}_{s_1}$.
34         Update $\alpha_{0.75} \leftarrow$ the 75th percentile of elements in $\alpha$.
35         Calculate a coefficient as $a_{\text{thr}} = \alpha_{0.75} + \text{IQR}(\alpha)$.   // (28)
36         Calculate an edge deletion threshold as $a_{\max} = \bar{\alpha}_{\text{del}} \frac{|\alpha_{\text{del}}|}{|\alpha_{\text{del}}|+|\alpha|} + a_{\text{thr}} \left(1 - \frac{|\alpha_{\text{del}}|}{|\alpha_{\text{del}}|+|\alpha|}\right)$.   // (27)
           /* Deletion of Edges */
37         for $\mathbf{y}_k \in \mathcal{N}_{s_1}$ do
38             if $a(\mathbf{y}_{s_1}, \mathbf{y}_k) > a_{max}$ then
39                 Update a set of ages of deleted edges as $\alpha_{\text{del}} \leftarrow \alpha_{\text{del}} \cup \{a(\mathbf{y}_{s_1}, \mathbf{y}_k)\}$.
40                 Delete the edge between $\mathbf{y}_{s_1}$ and $\mathbf{y}_k$.
41     if the number of presented data points is a multiple of $\lambda$ then
42         Delete isolated nodes.
```

4) VIGILANCE TEST

Similarities between the data point $\mathbf{x}$ and each of the 1st and 2nd winner nodes are defined in lines 13-14 of Algorithm 1 as follows:

$$V_{s_1} = \mathrm{CIM}\left(\mathbf{x}, \mathbf{y}_{s_1}, \mathrm{mean}(\mathcal{S})\right), \tag{17}$$
$$V_{s_2} = \mathrm{CIM}\left(\mathbf{x}, \mathbf{y}_{s_2}, \mathrm{mean}(\mathcal{S})\right). \tag{18}$$

The vigilance test classifies the relationship between the data point $\mathbf{x}$ and the two winner nodes into three cases by using the similarity threshold $V_{\text{threshold}}$, i.e.,

- Case I
  The similarity between $\mathbf{x}$ and the 1st winner node $\mathbf{y}_{s_1}$ is larger (i.e., less similar) than $V_{\text{threshold}}$, namely:

$$V_{\text{threshold}} < V_{s_1} \leq V_{s_2}. \tag{19}$$

- Case II
  The similarity between $\mathbf{x}$ and the 1st winner node $\mathbf{y}_{s_1}$ is smaller (i.e., more similar) than $V_{\text{threshold}}$, and the similarity between $\mathbf{x}$ and the 2nd winner node $\mathbf{y}_{s_2}$ is larger (i.e., less similar) than $V_{\text{threshold}}$, namely:

$$V_{s_1} \leq V_{\text{threshold}} < V_{s_2}. \tag{20}$$

- Case III
  The similarities between $\mathbf{x}$ and the 1st and 2nd winner nodes (i.e., $\mathbf{y}_{s_1}$ and $\mathbf{y}_{s_2}$) are both smaller (i.e., more similar) than $V_{\text{threshold}}$, namely:

$$V_{s_1} \leq V_{s_2} \leq V_{\text{threshold}}. \tag{21}$$

5) CREATION/UPDATE OF NODES AND EDGES

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

If the data point $\mathbf{x}$ is classified as Case I by the vigilance test (i.e., (19) is satisfied), a new node is created as $\mathbf{y}_{|\mathcal{Y}|+1} = \mathbf{x}$, and updated a node set as $\mathcal{Y} \leftarrow \mathcal{Y} \cup \{\mathbf{y}_{|\mathcal{Y}|+1}\}$. Here, the node $\mathbf{y}_{|\mathcal{Y}|+1}$ becomes a new active node, and the oldest node in the active node set $\mathcal{A}$ (i.e., λ nodes in $\mathcal{Y}$) is replaced by the new one. In addition, a bandwidth $\sigma_{|\mathcal{Y}|+1}$ for $\mathbf{y}_{|\mathcal{Y}|+1}$ is calculated by (10) and (13) with the active node set $\mathcal{A}$, and the winning count of $\mathbf{y}_{|\mathcal{Y}|+1}$ is initialized as $M_{|\mathcal{Y}|+1} = 1$.

If the data point $\mathbf{x}$ is classified as Case II by the vigilance test (i.e., (20) is satisfied), first, the winning count of $\mathbf{y}_{s_1}$ is updated as follows:

$$M_{s_1} \leftarrow M_{s_1} + 1. \tag{22}$$

Then, $\mathbf{y}_{s_1}$ is updated as follows:

$$\mathbf{y}_{s_1} \leftarrow \mathbf{y}_{s_1} + \frac{1}{M_{s_1}}\left(\mathbf{x} - \mathbf{y}_{s_1}\right), \tag{23}$$

here, the node $\mathbf{y}_{s_1}$ becomes a new active node, and the oldest node in the active node set $\mathcal{A}$ (i.e., λ nodes in $\mathcal{Y}$) is replaced by the new one.

When updating the node, the difference between $\mathbf{x}$ and $\mathbf{y}_{s_1}$ is divided by M_{s_1}. Thus, the change of the node position is smaller when M_{s_1} is larger. This is because the information around the node, where data points are frequently given, is important and should be held by the node.

The age of each edge connected to the 1st winner node $\mathbf{y}_{s_1}$ is also updated as follows:

$$a(\mathbf{y}_{s_1}, \mathbf{y}_k) \leftarrow a(\mathbf{y}_{s_1}, \mathbf{y}_k) + 1 \quad (\mathbf{y}_k \in \mathcal{N}_{s_1}), \tag{24}$$

where $\mathcal{N}_{s_1}$ is a set of all neighbor nodes of the node $\mathbf{y}_{s_1}$.

If the data point $\mathbf{x}$ is classified as Case III by the vigilance test (i.e., (21) is satisfied), the same operations as Case II (i.e., (22)-(24)) are performed. In addition, if there is an edge between $\mathbf{y}_{s_1}$ and $\mathbf{y}_{s_2}$, an age of the edge is reset as follows:

$$a(\mathbf{y}_{s_1}, \mathbf{y}_{s_2}) \leftarrow 1. \tag{25}$$

In the case that there is no edge between $\mathbf{y}_{s_1}$ and $\mathbf{y}_{s_2}$, a new edge is defined with an age of the edge by (25).

After updated the edge information, the neighbor nodes of $\mathbf{y}_{s_1}$ are updated as follows:

$$\mathbf{y}_k \leftarrow \mathbf{y}_k + \frac{1}{10M_k}\left(\mathbf{x} - \mathbf{y}_k\right) \quad \left(\mathbf{y}_k \in \mathcal{N}_{s_1}\right). \tag{26}$$

Apart from the above operations in Cases I-III, the nodes with no edges are deleted (and removed from the active node set $\mathcal{A}$) every λ data points for the noise reduction purpose (i.e., the node deletion interval is the presentation of λ data points), which is performed in lines 38-39 of Algorithm 1.

With respect to the active node set $\mathcal{A}$, its update rules are summarized as follows. In Case I, a new node is directly created by the data point $\mathbf{x}$ and added to $\mathcal{A}$. In Case II and Case III, the updated winner node in (23) is added to $\mathcal{A}$. In all cases, the oldest active node is removed from $\mathcal{A}$. Then, in lines 38-39 of Algorithm 1, all active nodes with no edges are removed. After this removal procedure, the number of active nodes can be smaller than λ.

6) ESTIMATION OF EDGE DELETION THRESHOLD

CAE estimates an edge deletion threshold based on the ages of the current edges and the deleted edges, which is inspired by the edge deletion mechanism of SOINN+ [39].

The edge deletion threshold $a_{\max}$ is defined as follows:

$$a_{\max} = \bar{\alpha}_{\text{del}} \frac{|\alpha_{\text{del}}|}{|\alpha_{\text{del}}| + |\alpha|} + a_{\text{thr}} \left(1 - \frac{|\alpha_{\text{del}}|}{|\alpha_{\text{del}}| + |\alpha|}\right), \tag{27}$$

where α_{del} is the set of ages of all the deleted edges during the learning process, $|\alpha_{\text{del}}|$ is the number of elements in α_{del}, $\bar{\alpha}_{\text{del}}$ is the arithmetic mean of α_{del}, α is the set of ages of edges which connect to $\mathbf{y}_{s_1}$ ($\alpha \subset \mathcal{E}$), and $|\alpha|$ is the number of elements in α. The coefficient a_{thr} is defined as follows:

$$a_{\text{thr}} = \alpha_{0.75} + \mathrm{IQR}(\alpha), \tag{28}$$

where $\alpha_{0.75}$ is the 75th percentile of elements in α, and $\mathrm{IQR}(\alpha)$ is the interquartile range.

The edge deletion threshold $a_{\max}$ is updated each time the age of an edge increases, which is performed in a line 33 of Algorithm 1.

7) DELETION OF EDGES

If there is an edge whose age is greater than the edge deletion threshold $a_{\max}$, the edge is deleted and the set of ages of deleted edges α_{del} is updated, which are performed in lines 36-37 of Algorithm 1.

IV. PROPOSED ALGORITHMS

This section explains the proposed algorithms in detail: CAE with a minor modification (i.e., $\mathrm{CAE_{FC}}$), CA+, and FCAC.

A. $\mathrm{CAE_{FC}}$

Although CAE [21] is a state-of-the-art parameter-free ART-based topological clustering algorithm, CAE tends to generate a large number of nodes. The main reason for this phenomenon is that the diversity D of the node set $\mathcal{A}$ defined in (11) is unlikely to be $D < 1.0\mathrm{e}{-6}$. In other words, the pairwise similarity matrix $\mathbf{R}$ defined in (12) is not appropriate in the case where a large number of nodes is not preferable. In $\mathrm{CAE_{FC}}$, a correntropy-based pairwise similarity matrix $\mathbf{R}$ is used, which is defined as follows:

$$\mathbf{R} = \left[\exp\left(\hat{C}(\mathbf{y}_i, \mathbf{y}_j, \sigma)\right)\right]_{1\le i,j\le|\mathcal{A}|}, \tag{29}$$

where $\hat{C}(\cdot)$ is correntropy which is defined in (5).

In comparison to the definition in (12), the definition in equation (29) tends to have a larger difference in the values of the elements of $\mathbf{R}$. As a result, the value of D is close to zero (i.e., $D < 1.0\mathrm{e}{-6}$) when the diversity of the node set $\mathcal{A}$ is small.

The rest of the learning procedure of $\mathrm{CAE_{FC}}$ is completely the same as CAE (see Algorithm 1).

For qualitative comparison between CAE and $\mathrm{CAE_{FC}}$, we perform a clustering task by using a simple dataset in Fig. 1. The dataset consists of 16,000 data points generated from a 2D Gaussian distribution ($\mu = (0, 0), \Sigma = [1, 0; 0, 1]$) and scaled to [0, 1]. Each data point is given to CAE and $\mathrm{CAE_{FC}}$ only once. Figs. 1b and 1c show the results of the clustering task by CAE and $\mathrm{CAE_{FC}}$, respectively. Note that in Fig. 1b, three isolated nodes are shown by different colors. Obviously, CAE generates a large number of nodes, while $\mathrm{CAE_{FC}}$ generates a small number of nodes.

If the objective of the clustering task is efficient information extraction, it is required to approximate a given data set with fewer nodes. From this perspective, $\mathrm{CAE_{FC}}$ is preferable to CAE. Note that the desired number of nodes depends on the purpose of the application and therefore CAE may be preferable in some other cases.

B. CA+

CA+ is a variant of $\mathrm{CAE_{FC}}$, i.e., $\mathrm{CAE_{FC}}$ without topology. Thus, the majority of the learning procedure is the same as $\mathrm{CAE_{FC}}$. The differences between CA+ and $\mathrm{CAE_{FC}}$ are summarized as follows:

- For CA+, the learning processes related to the edge information are removed from Algorithm 1, namely lines 24-25, a line 27, and lines 30-37. As a result, the output of CA+ is only a node set $\mathcal{Y}$.
- Since all the nodes of CA+ are isolated (i.e., no node has edges), a process for deleting isolated nodes (lines 38-39 in Algorithm 1) is removed.
- In $\mathrm{CAE_{FC}}$, the weight updating rule in (26) uses the edge information for defining the neighbor nodes $\mathcal{N}_{s_1}$ (see also lines 28-29 in Algorithm 1). Since CA+ has no edges, CA+ updates only the 2nd winner node $\mathbf{y}_{s_2}$ as follows:

$$\mathbf{y}_{s_2} \leftarrow \mathbf{y}_{s_2} + \frac{1}{100M_{s_2}}\left(\mathbf{x} - \mathbf{y}_{s_2}\right), \tag{30}$$

where M_{s_2} is the winning count of $\mathbf{y}_{s_2}$.
In CA+, $\mathbf{y}_{s_2}$ does not need to be moved significantly because more nodes are generated than $\mathrm{CAE_{FC}}$. Therefore, the coefficient in (30) is set to $1/100$ in CA+ whereas it is $1/10$ in $\mathrm{CAE_{FC}}$.

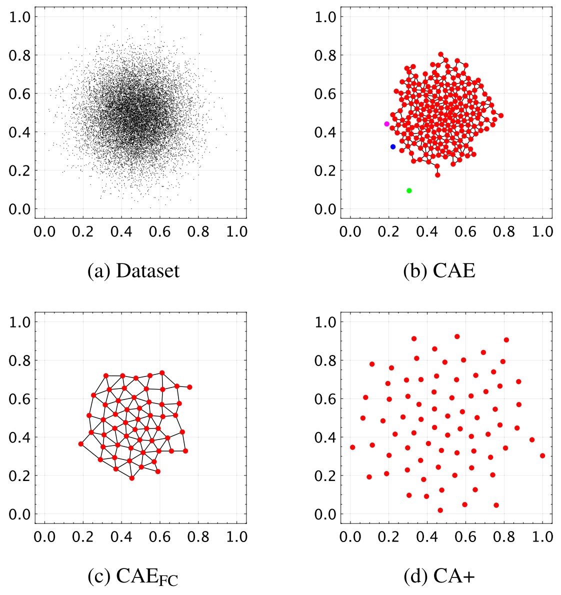


FIGURE 1. Clustering results for qualitative comparison of CAE, $\mathrm{CAE_{FC}}$, and CA+. (a) The dataset consists of 16,000 data points generated from a 2D Gaussian distribution ($\mu = (0, 0)$, $\Sigma = [1, 0; 0, 1]$) and scaled to [0, 1]. (b) Clustering result of CAE. (c) Clustering result of $\mathrm{CAE_{FC}}$. (d) Clustering result of CA+.

Similar to the qualitative comparison between CAE and $\mathrm{CAE_{FC}}$, CA+ and $\mathrm{CAE_{FC}}$ are also compared by the same clustering task with the dataset in Fig. 1a. Each data point in Fig. 1a is given to CA+ and $\mathrm{CAE_{FC}}$ only once. Figs. 1c and 1d show the results of the clustering task by $\mathrm{CAE_{FC}}$ and CA+, respectively. $\mathrm{CAE_{FC}}$ does not generate any nodes on the outer edges of the Gaussian distribution, while CA+ generates some nodes to cover the entire distribution. As a result, more nodes are generated by CA+ (75 nodes) than $\mathrm{CAE_{FC}}$ (55 nodes) for the task in Fig. 1a. This property of CA+ is preferable for a client in federated clustering because the generated nodes in each client are utilized as training data for the server.

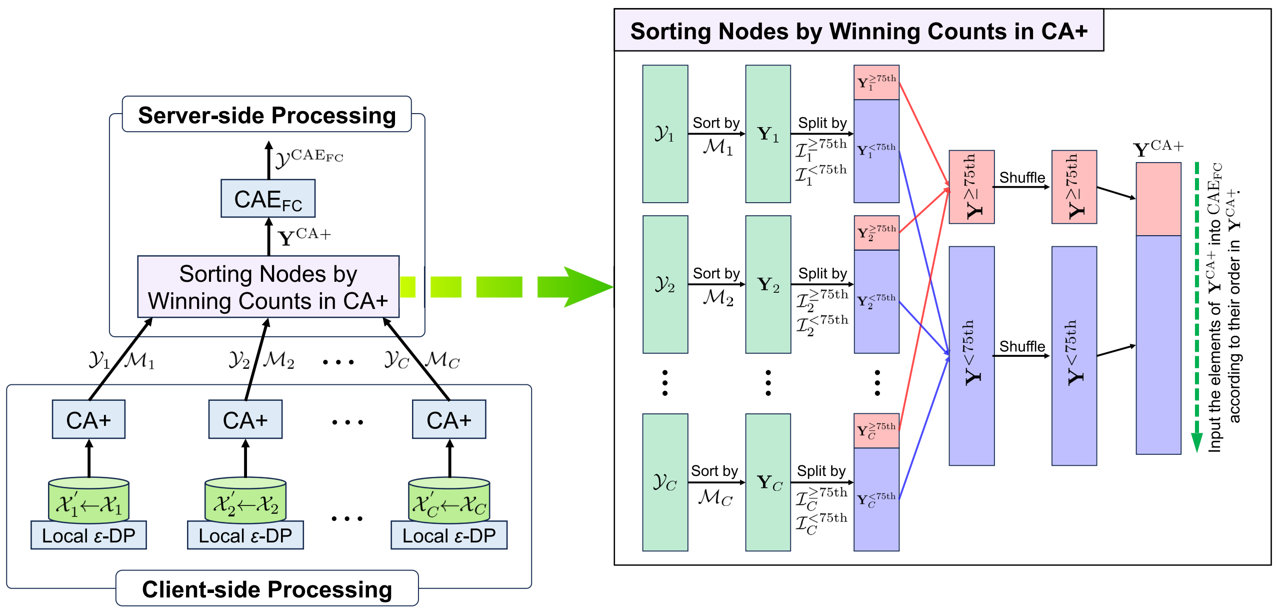

Server-side Processing
$\mathcal{Y}^{\mathrm{CAE_{FC}}}$
CAE$_{\mathrm{FC}}$
$\mathbf{Y}^{\mathrm{CA+}}$
Sorting Nodes by Winning Counts in CA+
$\mathcal{Y}_1$ $\mathcal{M}_1$
$\mathcal{Y}_2$ $\mathcal{M}_2$
$\mathcal{Y}_C$ $\mathcal{M}_C$
CA+
$\mathcal{X}'_1 \leftarrow \mathcal{X}_1$
$\mathcal{X}'_2 \leftarrow \mathcal{X}_2$
$\mathcal{X}'_C \leftarrow \mathcal{X}_C$
Local ε-DP
Client-side Processing
Sort by
Split by
$\mathcal{I}_1^{\geq 75\mathrm{th}}$ $\mathcal{I}_1^{<75\mathrm{th}}$
$\mathbf{Y}^{\geq 75\mathrm{th}}$
$\mathbf{Y}^{<75\mathrm{th}}$
Shuffle
Input the elements of $\mathbf{Y}^{\mathrm{CA+}}$ into CAE$_{\mathrm{FC}}$ according to their order in $\mathbf{Y}^{\mathrm{CA+}}$.

FIGURE 2. Overview of FCAC. FCAC performs clustering for distributed training data across multiple clients without aggregating the data on a single server while preserving data privacy. In the client-side processing, local ϵ-differential privacy is applied to each data point, and then CA+ is performed in parallel. In the server-side processing, the nodes generated by CA+ are fed to CAE$_{\mathrm{FC}}$ to obtain an aggregated clustering result.

C. FCAC

1) OVERVIEW

Fig. 2 shows the outline of FCAC. Similar to k-FED [10] and MUFC [62], FCAC is a one-shot federated clustering algorithm that does not require multiple communication rounds between the server and clients. In addition, since FCAC uses ART-based clustering algorithms, FCAC can perform continual learning while avoiding catastrophic forgetting. Furthermore, FCAC does not require any iterative learning process for convergence unlike k-FED and MUFC with k-means as a base clusterer.

As a unique learning process of FCAC, the training data of the server (i.e., generated nodes in each client) are re-ordered according to the importance of each node (i.e., a winning count $\mathcal{M}$), and then nodes with higher importance are fed to CAE$_{\mathrm{FC}}$ first one by one in the order of their importance (see Sorting Nodes by Winning Counts in CA+ in Fig. 2). Since the similarity threshold $V_{\mathrm{threshold}}$, which is an important parameter for achieving high clustering performance, is calculated based on a certain number of initial training data points, this process can be expected to improve the stability of network creation for CAE$_{\mathrm{FC}}$ in the early stage of learning.

2) LEARNING PROCEDURE

Algorithm 2 summarizes the learning procedure of FCAC. In the client-side processing, local ϵ-differential privacy is applied to each data point, and then CA+ is performed in parallel for obtaining $\mathcal{Y}_c$ and $\mathcal{M}_c$ $(c = 1, 2, \ldots, C)$.

In the server-side processing, as a first step, the nodes from each client are re-ordered based on a winning count of each node. Algorithm 3 summarizes the sorting mechanism of Fig. 2 in detail. First, the nodes in a client $\mathcal{Y}_c$ split into two ordered groups based on the winning counts $\mathcal{M}_c$, namely $\mathbf{Y}_c^{\geq 75\mathrm{th}}$ and $\mathbf{Y}_c^{<75\mathrm{th}}$ (lines 2-6). Here, $\mathbf{Y}_c^{\geq 75\mathrm{th}}$ consists of the nodes above the 75th percentile of elements in $\mathcal{M}_c$, i.e., a group of nodes with high winning counts $\mathcal{M}_c$, while $\mathbf{Y}_c^{<75\mathrm{th}}$ consists of the nodes below the 75th percentile of elements in $\mathcal{M}_c$, i.e., a group of nodes with low winning counts $\mathcal{M}_c$. Then, the order of the nodes is randomly shuffled in each

Algorithm 2 Learning Procedure of FCAC

```
Input:
a privacy budget $\epsilon$
the number of clients $C$
a set of data points for each client $\mathcal{X}_c$
$(c = 1, 2, \ldots, C)$
Output:
a set of nodes $\mathcal{Y}^{\mathrm{CAE_{FC}}}$
/* Client-side Processing */
1 for $c = \{1, \ldots, C\}$ do in parallel
2     Generate a set of privacy-preserving data points
      $\mathcal{X}'_c$ by local $\epsilon$-differential privacy.
      // (2), (3), (4)
3     Create nodes in each client as
      $\mathcal{Y}_c \leftarrow$ perform CA+ by using $\mathcal{X}'_c$.
/* Server-side Processing */
4 Update $\mathbf{Y}^{\mathrm{CA+}} \leftarrow$ sorting nodes by winning counts in
  CA+.                                   // Algorithm 3
5 Update $\mathcal{Y}^{\mathrm{CAE_{FC}}} \leftarrow$ perform CAE$_{\mathrm{FC}}$ by using $\mathbf{Y}^{\mathrm{CA+}}$.
```

Algorithm 3 Sorting Nodes by Winning Counts in CA+

```
Input:
  node sets $\{\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_C\}$
  winning count sets $\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_C\}$
Output:
  sorted nodes $\mathbf{Y}^{\text{CA+}}$
1 Initialize $\mathbf{Y}^{\geq 75\text{th}} \leftarrow$ null, and $\mathbf{Y}^{<75\text{th}} \leftarrow$ null.
2 for $c = 1, \ldots, C$ do
3     Extract indices $\mathcal{I}_c^{\geq 75\text{th}}$ and $\mathcal{I}_c^{<75\text{th}}$ which are above and below the 75th percentile of elements in $\mathcal{M}_c$, respectively.
4     Extract $\mathcal{Y}_c^{\geq 75\text{th}}$ and $\mathcal{Y}_c^{<75\text{th}}$ from $\mathcal{Y}_c$ corresponding to $\mathcal{I}_c^{\geq 75\text{th}}$ and $\mathcal{I}_c^{<75\text{th}}$, respectively.
5     Update $\mathbf{Y}^{\geq 75\text{th}} \leftarrow \mathbf{Y}^{\geq 75\text{th}}$.append($\mathcal{Y}_c^{\geq 75\text{th}}$).
6     Update $\mathbf{Y}^{<75\text{th}} \leftarrow \mathbf{Y}^{<75\text{th}}$.append($\mathcal{Y}_c^{<75\text{th}}$).
7 Update $\mathbf{Y}^{\geq 75\text{th}} \leftarrow$ Shuffle($\mathbf{Y}^{\geq 75\text{th}}$).
8 Update $\mathbf{Y}^{<75\text{th}} \leftarrow$ Shuffle($\mathbf{Y}^{<75\text{th}}$).
9 Update $\mathbf{Y}^{\text{CA+}} \leftarrow [\mathbf{Y}^{\geq 75\text{th}}; \mathbf{Y}^{<75\text{th}}]$.
```

group (lines 7-8). Last, the shuffled node groups $\mathbf{Y}_c^{\geq 75\text{th}}$ and $\mathbf{Y}_c^{<75\text{th}}$ are marged into $\mathbf{Y}^{\text{CA+}}$ while preserving their ordering (a line 9).

After preparing sorted nodes $\mathbf{Y}^{\text{CA+}}$, the elements of $\mathbf{Y}^{\text{CA+}}$ is fed to CAE$_{\text{FC}}$ according to their order in $\mathbf{Y}^{\text{CA+}}$ (a line 5 in Algorithm 2).

V. SIMULATION EXPERIMENTS

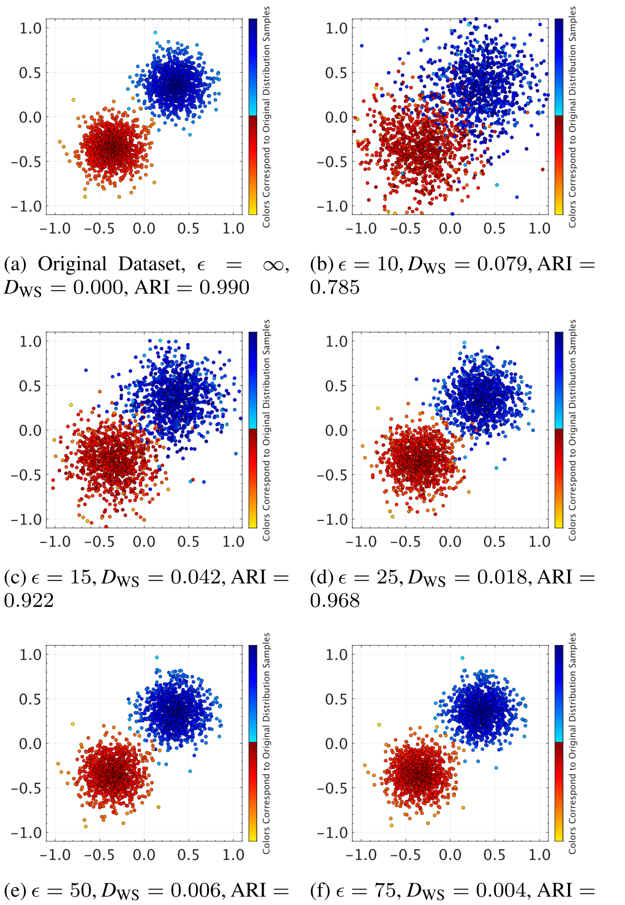
In this section, first, the effects of local ϵ-differential privacy is qualitatively analyzed. Next, the continual learning ability of FCAC is demonstrated by using a synthetic dataset. Then, the clustering performance of FCAC is quantitatively evaluated by using real-world datasets and compared with state-of-the-art federated clustering algorithms. Last, we analyze the computational complexity of FCAC.

Note that all experiments are carried out on Matlab 2023b and Python 3.10 with the Apple M1 Ultra processor and 128GB RAM.

A. EFFECTS OF LOCAL ϵ-DIFFERENTIAL PRIVACY ON DATA DISTRIBUTIONS

Apart from the abilities of FCAC, an intuitive understanding of the effects of local ϵ-differential privacy on a dataset is necessary to discuss the clustering performance of FCAC. In general, the value of ϵ controls the degree of data privacy protection, i.e., $\epsilon = 0$ means perfect privacy, while $\epsilon = \infty$ means no privacy guarantee. However, the relationship between the value of ϵ and the degree of data privacy protection is difficult to define quantitatively because it depends on the data sensitivity, the purpose of data analysis, and applications [67]. In this paper, therefore, we do not focus on the theoretical analysis of privacy protection.

Here, we qualitatively and quantitatively investigate the effects of local ϵ-difference privacy (i.e., noise) on data distributions using a synthetic data set which is shown in Fig. 3. The dataset consists of two Gaussian distributions (i.e., two clusters) with 1,000 data points each. In Fig. 3, a red data point is darker if it is closer to (-0.35, -0.35), while a blue data point is darker if it is closer to (0.35, 0.35). The noise as local ϵ-differential privacy for each data point is generated by using (3) with $\mu = 0$. The value of ϵ is set as $\{10, 15, 25, 50, 75\}$. These ϵ values were selected after investigating the case of generating large noise ($\epsilon = 10$) and the case of generating small noise ($\epsilon = 75$) from prior preliminary experiments. To quantitatively measure the changes in the distribution from the original dataset to the noised one, we apply the 1-Wasserstein Distance (D_{WS}) which arises from the idea of optimal transport [71]. Note that D_{WS} represents the amount of change in the data distribution, not represents the degree of data privacy protection. In addition, we apply k-means ($k = 2$) to each dataset in Fig. 3, and then calculate the Adjusted Rand Index (ARI) [72] to qualitatively measure the effects of local ϵ-differential privacy on clustering performance.



(a) Original Dataset, $\epsilon = \infty$, $D_{\text{WS}} = 0.000$, ARI = 0.990 (b) $\epsilon = 10$, $D_{\text{WS}} = 0.079$, ARI = 0.785 (c) $\epsilon = 15$, $D_{\text{WS}} = 0.042$, ARI = 0.922 (d) $\epsilon = 25$, $D_{\text{WS}} = 0.018$, ARI = 0.968 (e) $\epsilon = 50$, $D_{\text{WS}} = 0.006$, ARI = 0.986 (f) $\epsilon = 75$, $D_{\text{WS}} = 0.004$, ARI = 0.986

FIGURE 3. Effects of local ϵ-differential privacy on data distributions. (a) The dataset consists of two Gaussian distributions (i.e., two clusters) which have 1,000 data points each. (b)-(f) The results after adding some noise generated by the local ϵ-differential privacy.

Figs. 3b-3f show the effects of local ϵ-differential privacy on a 2D Gaussian distribution corresponding to each ϵ value.

The corresponding D_{WS} and ARI values for each distribution are shown next to the ϵ value. In Figs. 3b-3f, the color of each data point corresponds to the color of the data point before adding noise (i.e., the data point in Fig. 3a).

In Fig. 3b ($\epsilon = 10$), the noised dataset shows a significantly different distribution from the one in Fig. 3a. This indicates that the data utility is too low although the data privacy protection would be high. In fact, D_{WS} shows a larger value and ARI shows a lower value than others (i.e., $\epsilon = 15, 25, 50, 75$). Comparing to Fig. 3b ($\epsilon = 10$), the disturbance of the distribution is observed in Fig. 3c ($\epsilon = 15$) is small. In fact, D_{WS} is relatively smaller, and ARI is much higher than Fig. 3b ($\epsilon = 10$). A similar trend is observed in Fig. 3d ($\epsilon = 25$). In Figs. 3e and 3f ($\epsilon = 50, 75$), the effects of local ϵ-difference privacy is quite limited because D_{WS} is small and ARI is nearly 1.0. In summary, the results in Fig. 3 show that the clustering performance deteriorates when the value of ϵ is small because the data points move significantly, and the data distributions tend to overlap.

From the above-mentioned observations, we only consider $\epsilon = 15, 25, 50, 75$ for local ϵ-differential privacy in the subsequent sections.

B. CONTINUAL LEARNING ABILITY

This section demonstrates the continual learning ability of FCAC. Here, we consider that FCAC has two clients (i.e., client #1, client #2) and a server. The entire dataset used in this experiment is shown in Fig. 4. Each distribution in the entire dataset consists of 15,000 data points generated from the 2D Gaussian distribution and 150 data points (i.e., 1 % noise) sampled from a uniform distribution.

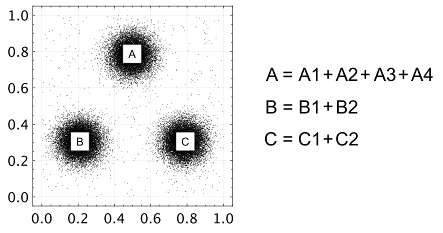


FIGURE 4. Entire dataset for the continual learning. Each distribution in the entire dataset consists of 15,000 data points generated from the 2D Gaussian distribution and 150 data points (i.e., 1 % noise) sampled from a uniform distribution.

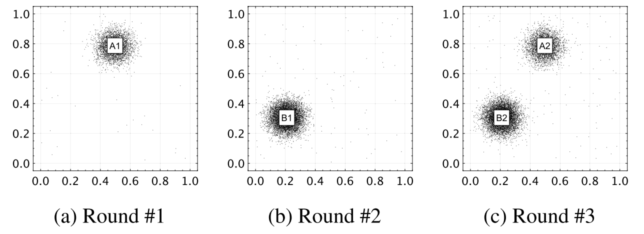


FIGURE 5. Visualization of the synthetic dataset for the client #1 in sequential order (i.e., (a) to (c)).

In order to perform continual learning in the practical environment, the entire dataset is divided into eight subsets without duplication as shown in Figs. 5 and 6 (i.e., A1, A2, A3, A4, B1, B2, C1, C2). The subsets A1, A2, A3, and A4 consist of 3,750 data points each, while the subsets B1, B2, C1, and C2 consist of 7,500 data points each. Each data point in each subset is given to FCAC only once. Throughout the round #1 to #3, CA+ in each client and CAE_{FC} in the server are continually updated without being initialized.

Fig. 7 shows the clustering results of the client #1, the client #2, and the server in the round #1. In the client-side

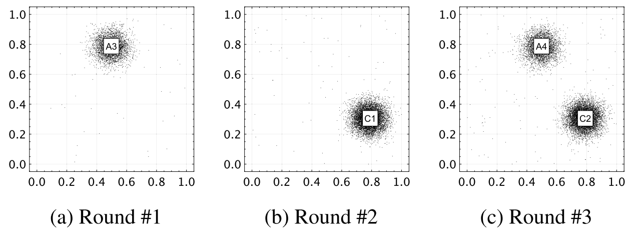


FIGURE 6. Visualization of the synthetic dataset for the client #2 in sequential order (i.e., (a) to (c)).

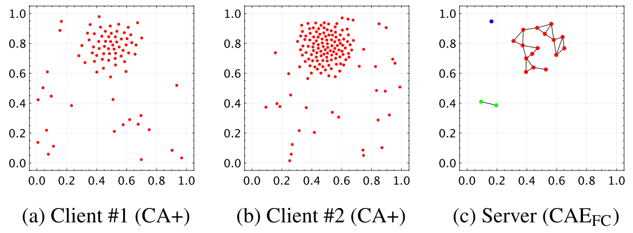


FIGURE 7. Clustering results of the client #1, the client #2, and the server in the round #1. In the client-side processing, some nodes are generated from noise data points because CA+ does not have noise reduction ability. In contrast on the server-side processing, CAE_{FC} generates a well-organized cluster by reducing the effect of noise data points thanks to its functionality.

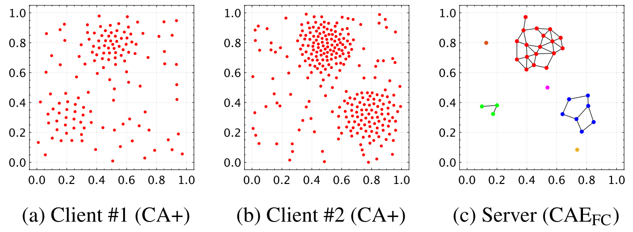


FIGURE 8. Clustering results of the client #1, the client #2, and the server in the round #2. Since each client and the server perform continual learning, the result of each client in the round #1 remains to the round #2.

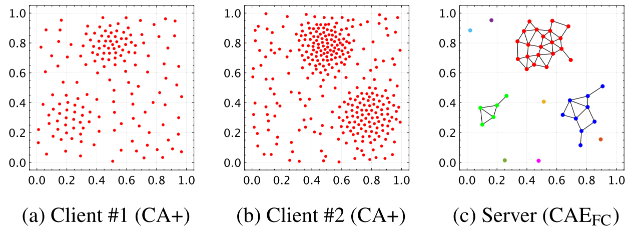


FIGURE 9. Clustering results in the round #3. The generated clusters in the round #3 are well-organized by updating the results in the round #2.

processing, some nodes are generated from noise data points because CA+ does not have noise reduction ability. In contrast on the server-side processing, CAE$_{FC}$ generates a well-organized cluster by reducing the effects of noise data points thanks to its functionality.

Fig. 8 shows the results in the round #2. Since each client and the server perform continual learning, the result of each client in the round #1 remains to the round #2 (see Figs. 8a and 8b). In Fig. 8c, therefore, the cluster with red-colored nodes in the server is organized by updating the small clusters of the server in the round #1 (i.e., Fig. 7c).

Fig. 9 shows the results in the round #3. It can be seen that the generated clusters in the round #3 are well-organized by updating the results in the round #2. It is noteworthy that there is no significant change in the respective distributions of nodes between rounds #2 and #3 (i.e., Figs. 8 and 9). This indicates that the distribution of the training data points is well-approximated with only a single learning epoch in rounds #1 and #2.

In the server-side processing, CAE$_{FC}$ generates some isolated nodes (see Figs. 7c, 8c, and 9c). This is because the number of presented data points is, in general, not a multiple of the deletion cycle (i.e., because the deletion process in line 42 of Algorithm 1 is, in general, not applied just before the termination of the algorithm for performance evaluation). A simple solution for avoiding this phenomenon is to delete isolated nodes after the learning procedure. However, since FCAC aims for continual learning, FCAC prepares for future learning without removing isolated nodes after the current learning procedure. Therefore, we consider that this is not a drawback of FCAC.

From the above-mentioned observations, we regard that FCAC can continually perform federated clustering without catastrophic forgetting.

C. CLUSTERING PERFORMANCE ON REAL-WORLD DATASETS

In this section, we evaluate the clustering performance of FCAC compared to state-of-the-art federated clustering algorithms by using real-world datasets.

1) COMPARED ALGORITHMS

As federated clustering algorithms, k-FED [10], FedFCM [12], and MUFC [62] are selected as compared algorithms. Note that although MUFC is originally proposed in the machine unlearning domain, MUFC is treated as a compared algorithm because it has a federated clustering mechanism [62]. In addition, we include k-means [24] as a reference although k-means is not a federated clustering algorithm.

Since a base clusterer of all the compared algorithms is a centroid-based algorithm, the number of clusters and the number of iterations for convergence are required. Throughout our experiments in this section, both in clients and a server, we set the number of clusters as equal to the number of classes in each dataset, and the number of iterations for convergence is 100. Moreover, the initialization of centroids in k-means is performed 10 times in the same way as in k-means++. In contrast, FCAC has no parameters to be specified in advance, the number of iterations for convergence is 1, and no centroid initialization process is need. Note that, therefore, FCAC has much higher applicability than the other compared algorithms.

The source code of k-FED,[1] FedFCM,[2] and MUFC[3] are obtained from the publicly available implementations. The source code of FCAC is available at GitHub.[4]

2) DATASET

We use 10 real-world datasets from public repositories [73]. Table 2 summarizes statistics of the 10 real-world datasets.

TABLE 2. Statistics of real-world datasets.

Dataset	# of Instances	# of Features	# of Classes	# of Clients
Hill-Valley	1,212	100	2	5
Ozone	2,534	73	2	5
Bioresponse	3,751	1,777	2	10
Phoneme	5,404	5	2	10
Optdigits	5,620	64	10	50
Pendigits	10,992	16	10	50
Mozilla4	15,545	5	2	10
Magic	19,020	11	2	50
FMNIST	70,000	784	10	100
Skin	245,057	3	2	100

Ozone stands for Ozone-Level-8hr. Magic stands for Magic-Telescope. FMNIST stands for Fashion-MNIST.

To perform federated clustering, each dataset is split into an arbitrary number of clients by using codes from the personalized federated learning platform repository.[5] In our experiments, each algorithm is evaluated by two conditions, namely an Independent and Identically Distributed (IID) scenario and a non-IID scenario. In the IID scenario, the number of data points in each client is the same, and the data distribution for each client is consistent with the entire dataset. In the non-IID scenario, on the other hand, the number of data points in each client is different. Moreover, the data distribution for each client is not consistent with the entire dataset (as a result, each client has a different data distribution). In the case of the non-IID scenario, each of the 10 datasets in Table 2 is divided into the same number of subsets as the number of clients by using Dirichlet distribution-based splitting approach with a parameter $\alpha = 0.5$ [74].

After splitting each of the 10 datasets, local ϵ-differential privacy is applied to each data point in each client. In our experiments, we first examine $\epsilon = \infty$ to evaluate the general clustering performance of each algorithm although $\epsilon = \infty$ means no privacy guarantee (i.e., a data point has no noise).

[1]https://github.com/metastableB/kfed/
[2]https://github.com/stallmo/federated_clustering
[3]https://github.com/thupchnsky/mufc
[4]https://github.com/Masuyama-lab/FCAC
[5]https://github.com/TsingZ0/PFL-Non-IID

Then, we examine four values of ϵ (i.e., 15, 25, 50, 75) which are determined based on the observations in Section V-A.

During experiments, all data points in each dataset are presented to each algorithm in random order. Since all algorithms are clustering algorithms, we use the same data points for training and testing, i.e., an algorithm is trained by all data points in each dataset and tested by the same data points as the training data.

Note that since k-means is not a federated clustering algorithm, the datasets split into each client are merged, and then k-means performs clustering on the merged dataset.

3) RESULTS OF GENERAL CLUSTERING PERFORMANCE

Tables 3 and 4 show the results of clustering performance on the 10 real-world datasets in the IID and non-IID scenarios with $\epsilon = \infty$, respectively. The clustering performance of each algorithm is measured by the ARI [72], the Adjusted Mutual Information (AMI) [75], and the Normalized Mutual Information (NMI) [76]. With respect to FCAC, the number of nodes and clusters in the server (i.e., CAE_{FC}) are also shown. We repeat the evaluation 20 times with different random seeds for obtaining consistent averaging results. The best value in each metric is indicated in bold, and the values in parentheses indicate the standard deviation. A number to the right of each evaluation metric is the rank of an algorithm corresponding to the metric value. The smaller the rank, the better the metric score. In addition, a darker tone in a cell corresponds to a smaller rank (i.e., better evaluation).

As general trends, FCAC shows better clustering performance than the other algorithms in both scenarios, and MUFC and k-means show better clustering performance than k-FED and FedFCM. In particular, FCAC shows high clustering performance in the Magic and Skin datasets. Moreover, although FCAC shows a low rank in the Phoneme, Optdigits, and FMNIST datasets, the values of ARI, AMI, and NMI are not extremely low compared to the other algorithms. As mentioned in Section V-C1, FCAC has no parameters to be specified in advance, the number of iterations for convergence is 1, and no centroid initialization process is needed. In contrast, all the compared algorithms have a parameter to be specified in advance, and require a number of iterative processes for good clustering performance. This clearly highlights the advantages of FCAC for situations where the distribution of a dataset is unknown and/or the size of a dataset is large.the number of iterations for convergence is 1, and no centroid initialization process is needed. In contrast, all the compared algorithms have a parameter to be specified in advance, and require a number of iterative processes for good clustering performance. This clearly highlights the advantages of FCAC for situations where the distribution of a dataset is unknown and/or the size of a dataset is large.

With respect to the number of clusters of FCAC, FCAC tends to generate a large number of clusters compared to the true number of classes in each dataset as shown in Tables 3 and 4. This property has positive impacts on clustering performance in many cases. Compared to k-FED, FedFCM, and MUFC, since a large number of nodes are generated near the boundary regions between clusters, the boundaries of clusters become clearer. As a result, it is considered that the decrease in the value of the clustering evaluation metric is suppressed. Note that, in general, it is difficult to discuss the relation between the number of clusters and clustering performance in the case of self-organizing algorithms that adaptively generate nodes corresponding to data points sampled from an unknown data distribution, such as GNG-, and ART-based clustering algorithms.

To assess the statistical differences between the results shown in Tables 3 and 4, we employ the Friedman test and the Nemenyi post-hoc analysis [77]. The Friedman test is utilized for testing a null hypothesis that the performance of all algorithms is equal. Upon rejection of the null hypothesis, the Nemenyi post-hoc analysis is conducted. The Nemenyi post-hoc analysis involves comparing every pair of algorithms, considering their performance ranks across all datasets for each evaluation metric. In this case, both the Friedman test and Nemenyi post-hoc analysis identify significant differences at a significance level of 0.05.

Fig. 10 shows critical difference diagrams based on the results of ARI, AMI, and NMI by each algorithm, which are defined by the Nemenyi post-hoc analysis. A better result has a lower average rank, i.e., on the right side of each diagram. In theory, algorithms within a critical distance (i.e., a red line) do not have a statistically significance difference [77]. Fig. 10a shows a critical difference diagram based on the overall results (i.e., all the results of ARI, AMI, and NMI in the IID and non-IID scenarios). FCAC is the lowest rank (i.e., best) algorithm with a statistically significant difference from k-FED and FedFCM. Figs. 10b and 10c are critical difference diagrams correspond to the results in Tables 3 and 4, respectively. The ranks of MUFC and FedFCM differ depending on the scenario (i.e., IID or non-IID), which implies the instability of their learning.

The above-mentioned observations suggest that FCAC has superior clustering performance to state-of-the-art algorithms on various datasets with $\epsilon = \infty$.

4) RESULTS OF CLUSTERING PERFORMANCE ON DATASETS WITH LOCAL ϵ-DIFFERENTIAL PRIVACY

For the comparisons of clustering performance on the 10 datasets with local ϵ-differential privacy, we set $\epsilon = 15, 25, 50, 75$ and then conduct the same experiments as Section V-C3 for obtaining ARI, AMI, and NMI. As mentioned in Section III-A, the value of ϵ controls the degree of data privacy protection, i.e., the smaller ϵ value provides higher data privacy, while the larger ϵ value provides lower data privacy.

Similar to Section V-C3, the Friedman test and Nemenyi post-hoc analysis are used. The Friedman test is used to test the null hypothesis that all algorithms perform equally. If the

TABLE 3. Results of quantitative comparisons on 10 real-world datasets in the IID scenario with $\epsilon = \infty$. As general trends, FCAC shows better clustering performance than the other algorithms, and MUFC and k-means show better clustering performance than k-FED and FedFCM.

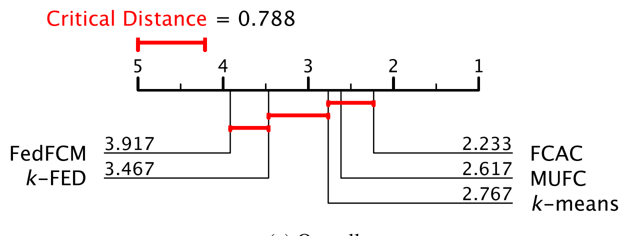
Dataset	Metric	k-means	k-FED	FedFCM	MUFC	FCAC
Hill-Valley	ARI	-0.0001 (0.0000) 3	-0.0001 (0.0000) 4	-0.0001 (0.0001) 5	0.0000 (0.0003) 2	0.0011 (0.0014) 1
	AMI	-0.0003 (0.0000) 3	-0.0003 (0.0001) 5	-0.0003 (0.0002) 4	-0.0002 (0.0009) 2	0.0025 (0.0022) 1
	NMI	0.0005 (0.0000) 3	0.0005 (0.0001) 5	0.0005 (0.0002) 4	0.0007 (0.0008) 2	0.0076 (0.0037) 1
	# of Nodes	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	19.9 (4.0)
	# of Clusters	—	—	—	—	11.6 (4.2)
Ozone	ARI	-0.0311 (0.0000) 5	-0.0308 (0.0008) 4	-0.0098 (0.0012) 1	-0.0219 (0.0224) 2	-0.0246 (0.0173) 3
	AMI	0.0187 (0.0000) 3	0.0186 (0.0005) 4	0.0189 (0.0045) 2	0.0175 (0.0055) 5	0.0221 (0.0049) 1
	NMI	0.0191 (0.0000) 3	0.0190 (0.0005) 4	0.0194 (0.0045) 2	0.0180 (0.0055) 5	0.0268 (0.0039) 1
	# of Nodes	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	45.9 (13.1)
	# of Clusters	—	—	—	—	29.5 (11.6)
Bioresponse	ARI	-0.0010 (0.0000) 5	-0.0010 (0.0001) 4	0.0101 (0.0025) 1	0.0079 (0.0067) 2	0.0003 (0.0063) 3
	AMI	0.0033 (0.0000) 5	0.0033 (0.0001) 4	0.0060 (0.0006) 2	0.0056 (0.0027) 3	0.0117 (0.0067) 1
	NMI	0.0035 (0.0000) 5	0.0035 (0.0001) 4	0.0062 (0.0006) 2	0.0058 (0.0027) 3	0.0151 (0.0075) 1
	# of Nodes	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	215.0 (14.9)
	# of Clusters	—	—	—	—	18.6 (11.1)
Phoneme	ARI	0.1195 (0.0008) 3	0.1155 (0.0320) 4	0.1397 (0.0019) 2	0.1536 (0.0631) 1	0.0659 (0.0312) 5
	AMI	0.1756 (0.0007) 1	0.1658 (0.0147) 2	0.1608 (0.0016) 3	0.1417 (0.0462) 4	0.1136 (0.0115) 5
	NMI	0.1758 (0.0007) 1	0.1659 (0.0147) 2	0.1610 (0.0016) 3	0.1418 (0.0462) 4	0.1149 (0.0113) 5
	# of Nodes	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	46.8 (10.5)
	# of Clusters	—	—	—	—	26.9 (10.2)
Optdigits	ARI	0.6719 (0.0028) 2	0.5836 (0.0279) 3	0.2224 (0.0322) 5	0.6922 (0.0355) 1	0.4425 (0.0625) 4
	AMI	0.7547 (0.0035) 1	0.6957 (0.0140) 3	0.3737 (0.0278) 5	0.7506 (0.0188) 2	0.5937 (0.0352) 4
	NMI	0.7555 (0.0035) 1	0.6966 (0.0139) 3	0.3755 (0.0277) 5	0.7514 (0.0187) 2	0.5986 (0.0351) 4
	# of Nodes	10.0 (0.0)	10.0 (0.0)	10.0 (0.0)	10.0 (0.0)	72.4 (21.1)
	# of Clusters	—	—	—	—	41.0 (18.9)
Pendigits	ARI	0.5449 (0.0268) 3	0.5245 (0.0298) 4	0.3428 (0.0321) 5	0.5645 (0.0284) 2	0.6185 (0.0507) 1
	AMI	0.6833 (0.0045) 2	0.6613 (0.0135) 4	0.5192 (0.0301) 5	0.6766 (0.0151) 3	0.7180 (0.0275) 1
	NMI	0.6838 (0.0045) 2	0.6619 (0.0135) 4	0.5201 (0.0300) 5	0.6772 (0.0150) 3	0.7195 (0.0275) 1
	# of Nodes	10.0 (0.0)	10.0 (0.0)	10.0 (0.0)	10.0 (0.0)	106.2 (16.6)
	# of Clusters	—	—	—	—	34.4 (13.4)
Mozilla4	ARI	-0.0065 (0.0003) 1	-0.0078 (0.0004) 3	-0.0075 (0.0000) 2	-0.0109 (0.0168) 5	-0.0106 (0.0312) 4
	AMI	0.0456 (0.0001) 4	0.0456 (0.0023) 5	0.0575 (0.0000) 2	0.0526 (0.0099) 3	0.1003 (0.0156) 1
	NMI	0.0457 (0.0001) 4	0.0457 (0.0023) 5	0.0576 (0.0000) 2	0.0526 (0.0099) 3	0.1026 (0.0162) 1
	# of Nodes	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	297.3 (91.7)
	# of Clusters	—	—	—	—	162.6 (72.8)
Magic	ARI	0.0594 (0.0000) 2	0.0543 (0.0018) 3	0.0201 (0.0005) 5	0.0217 (0.0145) 4	0.0993 (0.0442) 1
	AMI	0.0209 (0.0000) 2	0.0182 (0.0009) 3	0.0070 (0.0001) 5	0.0070 (0.0038) 4	0.0900 (0.0299) 1
	NMI	0.0210 (0.0000) 2	0.0183 (0.0009) 3	0.0071 (0.0001) 5	0.0071 (0.0038) 4	0.0908 (0.0302) 1
	# of Nodes	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	158.1 (42.1)
	# of Clusters	—	—	—	—	27.1 (15.2)
FMNIST	ARI	0.3596 (0.0155) 3	0.3620 (0.0193) 2	0.1701 (0.0029) 5	0.3655 (0.0249) 1	0.2415 (0.0675) 4
	AMI	0.5154 (0.0067) 1	0.5115 (0.0120) 2	0.2549 (0.0043) 5	0.5092 (0.0133) 3	0.4607 (0.0375) 4
	NMI	0.5155 (0.0067) 1	0.5117 (0.0120) 2	0.2551 (0.0043) 5	0.5093 (0.0133) 3	0.4613 (0.0375) 4
	# of Nodes	10.0 (0.0)	10.0 (0.0)	10.0 (0.0)	10.0 (0.0)	83.6 (20.8)
	# of Clusters	—	—	—	—	35.3 (14.9)
Skin	ARI	-0.0397 (0.0001) 4	-0.0396 (0.0001) 3	-0.0329 (0.0155) 2	-0.0409 (0.0094) 5	0.1276 (0.2579) 1
	AMI	0.0234 (0.0001) 3	0.0233 (0.0001) 4	0.0210 (0.0019) 5	0.0265 (0.0107) 2	0.2127 (0.2002) 1
	NMI	0.0234 (0.0001) 3	0.0233 (0.0001) 4	0.0210 (0.0019) 5	0.0265 (0.0107) 2	0.2127 (0.2002) 1
	# of Nodes	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	2.0 (0.0)	80.3 (12.2)
	# of Clusters	—	—	—	—	14.9 (8.4)
	Average ARI Rank	3.100	3.300	3.400	2.500	2.700
	Average AMI Rank	2.500	3.800	3.600	3.100	2.000
	Average NMI Rank	2.500	3.800	3.600	3.100	2.000
	Average Total Rank	2.700	3.633	3.533	2.900	2.233

Ozone stands for Ozone-Level-8hr. Magic stands for Magic-Telescope. FMNIST stands for Fashion-MNIST.
The best value in each metric is indicated in bold. The values in parentheses indicate the standard deviation.
A number to the right of a metric value is the rank of an algorithm corresponding to the metric value.
The smaller the rank, the better the metric score. A darker tone in a cell corresponds to a smaller rank.

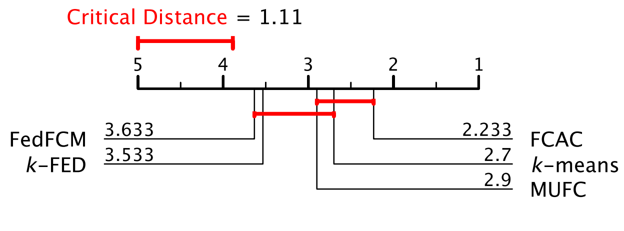
TABLE 4. Results of quantitative comparisons on 10 real-world datasets in the non-IID scenario with $\epsilon = \infty$. As general trends, FCAC shows better clustering performance than the other algorithms, and MUFC and k-means show better clustering performance than k-FED and FedFCM.

Dataset	Metric	k-means		k-FED		FedFCM		MUFC		FCAC	
Hill-Valley	ARI	-0.0001 (0.0000)	5	-0.0001 (0.0001)	3	-0.0001 (0.0001)	4	0.0000 (0.0003)	2	0.0006 (0.0011)	1
	AMI	-0.0003 (0.0000)	5	-0.0002 (0.0003)	3	-0.0003 (0.0002)	4	-0.0002 (0.0008)	2	0.0013 (0.0024)	1
	NMI	0.0005 (0.0000)	5	0.0006 (0.0003)	3	0.0006 (0.0002)	4	0.0006 (0.0007)	2	0.0057 (0.0031)	1
	# of Nodes	2.0 (0.0)		2.0 (0.0)		2.0 (0.0)		2.0 (0.0)		18.2 (3.9)	
	# of Clusters	—		—		—		—		9.3 (3.3)	
Ozone	ARI	-0.0311 (0.0000)	5	-0.0271 (0.0021)	3	-0.0046 (0.0001)	1	-0.0149 (0.0230)	2	-0.0283 (0.0189)	4
	AMI	0.0187 (0.0000)	2	0.0177 (0.0007)	4	0.0048 (0.0003)	5	0.0180 (0.0063)	3	0.0202 (0.0049)	1
	NMI	0.0192 (0.0000)	2	0.0182 (0.0007)	4	0.0052 (0.0003)	5	0.0185 (0.0063)	3	0.0254 (0.0045)	1
	# of Nodes	2.0 (0.0)		2.0 (0.0)		2.0 (0.0)		2.0 (0.0)		48.8 (20.0)	
	# of Clusters	—		—		—		—		32.4 (17.5)	
Bioresponse	ARI	-0.0010 (0.0000)	4	-0.0010 (0.0000)	3	0.0111 (0.0003)	1	0.0059 (0.0075)	2	-0.0014 (0.0024)	5
	AMI	0.0033 (0.0000)	4	0.0033 (0.0000)	5	0.0064 (0.0002)	3	0.0068 (0.0078)	2	0.0122 (0.0070)	1
	NMI	0.0035 (0.0000)	4	0.0035 (0.0000)	5	0.0066 (0.0002)	3	0.0071 (0.0078)	2	0.0157 (0.0080)	1
	# of Nodes	2.0 (0.0)		2.0 (0.0)		2.0 (0.0)		2.0 (0.0)		185.6 (20.1)	
	# of Clusters	—		—		—		—		20.8 (12.0)	
Phoneme	ARI	0.1194 (0.0008)	2	0.1574 (0.0566)	1	0.0775 (0.0214)	4	0.0918 (0.1104)	3	0.0513 (0.0204)	5
	AMI	0.1757 (0.0007)	1	0.0764 (0.0282)	5	0.1030 (0.0260)	4	0.1066 (0.0520)	2	0.1050 (0.0084)	3
	NMI	0.1758 (0.0007)	1	0.0766 (0.0282)	5	0.1031 (0.0260)	4	0.1067 (0.0520)	2	0.1063 (0.0082)	3
	# of Nodes	2.0 (0.0)		2.0 (0.0)		2.0 (0.0)		2.0 (0.0)		45.3 (9.0)	
	# of Clusters	—		—		—		—		27.5 (10.3)	
Optdigits	ARI	0.6709 (0.0008)	2	0.5360 (0.0421)	3	0.2647 (0.0269)	5	0.6754 (0.0311)	1	0.4089 (0.0469)	4
	AMI	0.7563 (0.0005)	1	0.6739 (0.0273)	3	0.4036 (0.0223)	5	0.7449 (0.0167)	2	0.5669 (0.0281)	4
	NMI	0.7570 (0.0005)	1	0.6750 (0.0272)	3	0.4057 (0.0222)	5	0.7457 (0.0167)	2	0.5725 (0.0288)	4
	# of Nodes	10.0 (0.0)		10.0 (0.0)		10.0 (0.0)		10.0 (0.0)		69.3 (22.8)	
	# of Clusters	—		—		—		—		44.9 (23.6)	
Pendigits	ARI	0.5478 (0.0280)	3	0.5040 (0.0236)	4	0.3178 (0.0180)	5	0.5672 (0.0290)	2	0.5836 (0.0656)	1
	AMI	0.6837 (0.0045)	2	0.6512 (0.0105)	4	0.4867 (0.0135)	5	0.6820 (0.0115)	3	0.6994 (0.0323)	1
	NMI	0.6842 (0.0045)	2	0.6517 (0.0105)	4	0.4876 (0.0135)	5	0.6825 (0.0115)	3	0.7010 (0.0322)	1
	# of Nodes	10.0 (0.0)		10.0 (0.0)		10.0 (0.0)		10.0 (0.0)		101.2 (19.1)	
	# of Clusters	—		—		—		—		32.7 (11.2)	
Mozilla4	ARI	-0.0065 (0.0002)	3	-0.0021 (0.0011)	1	-0.0078 (0.0004)	4	-0.0047 (0.0322)	2	-0.0269 (0.0421)	5
	AMI	0.0457 (0.0001)	3	0.0444 (0.0005)	5	0.0447 (0.0003)	4	0.0557 (0.0111)	2	0.0945 (0.0198)	1
	NMI	0.0457 (0.0001)	3	0.0444 (0.0005)	5	0.0448 (0.0003)	4	0.0557 (0.0111)	2	0.0969 (0.0200)	1
	# of Nodes	2.0 (0.0)		2.0 (0.0)		2.0 (0.0)		2.0 (0.0)		295.8 (66.2)	
	# of Clusters	—		—		—		—		151.8 (60.1)	
Magic	ARI	0.0594 (0.0000)	2	0.0582 (0.0276)	3	0.0234 (0.0002)	5	0.0255 (0.0203)	4	0.1268 (0.0259)	1
	AMI	0.0209 (0.0000)	3	0.0289 (0.0197)	2	0.0075 (0.0001)	5	0.0117 (0.0150)	4	0.1072 (0.0139)	1
	NMI	0.0210 (0.0000)	3	0.0290 (0.0197)	2	0.0076 (0.0001)	5	0.0117 (0.0150)	4	0.1079 (0.0140)	1
	# of Nodes	2.0 (0.0)		2.0 (0.0)		2.0 (0.0)		2.0 (0.0)		143.3 (24.3)	
	# of Clusters	—		—		—		—		22.3 (9.8)	
FMNIST	ARI	0.3535 (0.0103)	3	0.3602 (0.0227)	2	0.2080 (0.0062)	5	0.3616 (0.0240)	1	0.2877 (0.0590)	4
	AMI	0.5117 (0.0010)	2	0.5047 (0.0255)	3	0.3313 (0.0054)	5	0.5143 (0.0167)	1	0.4846 (0.0305)	4
	NMI	0.5118 (0.0010)	2	0.5049 (0.0255)	3	0.3315 (0.0054)	5	0.5144 (0.0167)	1	0.4852 (0.0305)	4
	# of Nodes	10.0 (0.0)		10.0 (0.0)		10.0 (0.0)		10.0 (0.0)		77.4 (23.3)	
	# of Clusters	—		—		—		—		41.9 (18.1)	
Skin	ARI	-0.0397 (0.0001)	4	-0.0308 (0.0007)	3	-0.0286 (0.0018)	2	-0.0435 (0.0122)	5	0.2245 (0.2451)	1
	AMI	0.0234 (0.0001)	3	0.0179 (0.0004)	4	0.0161 (0.0007)	5	0.0299 (0.0180)	2	0.3054 (0.1916)	1
	NMI	0.0234 (0.0001)	3	0.0179 (0.0004)	4	0.0161 (0.0007)	5	0.0299 (0.0180)	2	0.3054 (0.1916)	1
	# of Nodes	2.0 (0.0)		2.0 (0.0)		2.0 (0.0)		2.0 (0.0)		72.2 (11.9)	
	# of Clusters	—		—		—		—		18.7 (10.2)	
	Average ARI Rank	3.300		3.600		2.600		2.400		3.100	
	Average AMI Rank	2.600		4.500		3.800		2.300		1.800	
	Average NMI Rank	2.600		4.500		3.800		2.300		1.800	
	Average Total Rank	2.833		4.200		3.400		2.333		2.233	

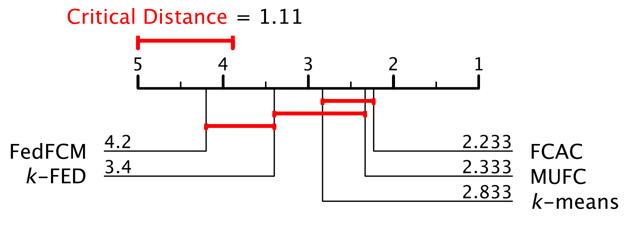
Ozone stands for Ozone-Level-8hr. Magic stands for Magic-Telescope. FMNIST stands for Fashion-MNIST.
The best value in each metric is indicated in bold. The values in parentheses indicate the standard deviation.
A number to the right of a metric value is the rank of an algorithm corresponding to the metric value.
The smaller the rank, the better the metric score. A darker tone in a cell corresponds to a smaller rank.



(a) Overall



(b) IID scenario



(c) non-IID scenario

FIGURE 10. Critical difference diagram based on the results of ARI, AMI, and NMI with $\epsilon = \infty$. FCAC shows the lowest rank (i.e., best) algorithm with a statistically significant difference from k-FED and FedFCM. The ranks of MUFC and FedFCM differ depending on the scenario (i.e., IID or non-IID), which implies the instability of their learning.

null hypothesis is rejected, the Nemenyi post-hoc analysis is then conducted. Here, the null hypothesis is rejected at the significance level of 0.05 both in the Friedman test and the Nemenyi post-hoc analysis.

In this section, due to page limitations, we only show the critical difference diagrams based on the overall results of ARI, AMI, and NMI. Fig. 11 shows the critical difference diagram corresponding to $\epsilon = 15, 25, 50, 75$. Except for $\epsilon = 15$, FCAC is the lowest rank (i.e., best) algorithm with a statistically significant difference from k-FED and FedFCM. This indicates that FCAC can maintain higher clustering performance for various privacy-preserving datasets than the other state-of-the-art algorithms.

VI. LIMITATIONS OF FCAC

In this section, we discuss the limitations of FCAC. First, we focus on the robustness against noise data. In general, the clustering/classification performance of ART-based algorithms is largely affected by noisy data, but FCAC shows a certain noise reduction ability as shown in Figs. 7-9. Although CAE_{FC} does not perfectly avoid the effect of noise data points, this is a positive observation. However, the datasets contains only 1 % noise data points. Therefore, We need to consider the further improvement of the noise reduction ability of FCAC to handle more practical data (e.g., EEG and sensory data).

Next, we discuss why the clustering performance of FCAC is inferior to the compared algorithms in Phoneme, Optdigits, and FMNIST datasets which are shown in Tables 3 and 4. In FCAC, the similarity threshold, which has a significant impact on the clustering performance, is calculated based on the diversity of the nodes using the DPP-based criterion. (see Section III-D1). With respect to the Optdigits and FMNIST datasets, their features consist of pixel values in images and are largely dominated by the black-colored background. Since the Phoneme dataset consists of features extracted from the speech signal, there is a large overlap between the features. It is difficult to calculate an appropriate similarity threshold when there is a large number of data with similar features in the dataset. Note that while this may be one of the limitations of FCAC, we believe that this can be addressed by using appropriate dimensionality reduction methods and/or features of the data extracted by an auto-encoder.

Last, we focus on the increase in the complexity of the clustering results in FCAC. Because FCAC is a centroid-based (i.e., a node-based) algorithm such as k-means, FCAC may not extract information effectively if the distribution of newly given data overlaps with a region where nodes already exist. In such a case, it is necessary to generate enough nodes to aggregate and maintain new and learned information, which may increase the complexity of the model. A simple solution is to merge and delete nodes while avoiding catastrophic forgetting. Alternatively, a selective forgetting mechanism (e.g., machine unlearning [62]) could be employed. By incorporating the above-mentioned mechanisms, we can expect further improvements in the continual learning ability of FCAC.

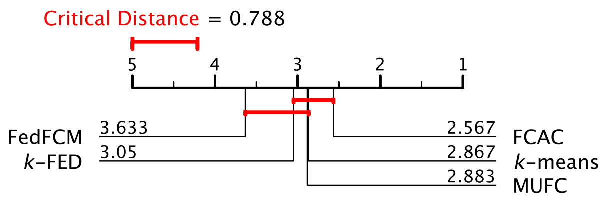
VII. COMPUTATIONAL COMPLEXITY

This section presents the computational complexity of FCAC. In this section, we use the notations in Table 1, namely d is the dimensionality of a data point, n is the number of data points, K is the number of nodes, $\mathcal{M}$ is a set of winning counts, λ is the number of active nodes, and $|\mathcal{E}|$ is the number of elements in the ages of edges set $\mathcal{E}$.

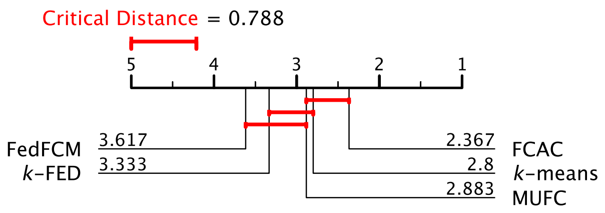
In FCAC, the computations on client-side can be performed in parallel, and therefore computational complexity is defined by the learning procedure of CA+, the re-ordering of training data points for CAE_{FC}, and the learning procedure of CAE_{FC}. Furthermore, since CA+ is a variant of CAE_{FC}, i.e., CAE_{FC} without topology, we only consider the computational complexity of the re-ordering of training data points for CAE_{FC} and the learning procedure of CAE_{FC}.

The computational complexity of the re-ordering of training data points for CAE_{FC} is as follows: for finding the 75th percentile of elements in $\mathcal{M}$ is $\mathcal{O}(K)$ (a line 3 in Alg. 3), for splitting a node set is $\mathcal{O}(K)$ (a line 4 in Alg. 3), and for shuffling the splitted nodes is $\mathcal{O}(K^{\geq 75\text{th}})$ and $\mathcal{O}(K^{<75\text{th}})$ (lines 7-8 in Alg. 3).

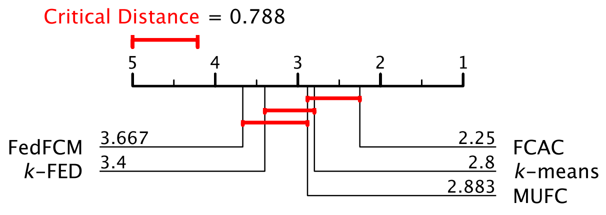
The computational complexity of the learning procedure of CAE_{FC} is as follows: for computing a bandwidth of a kernel function in CIM is $\mathcal{O}(d)$, for calculating a pairwise similarity



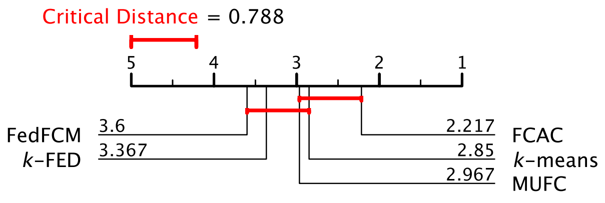
(a) $\epsilon = 15$



(b) $\epsilon = 25$



(c) $\epsilon = 50$



(d) $\epsilon = 75$

FIGURE 11. Critical difference diagram based on the overall results of ARI, AMI, and NMI with $\epsilon = 15, 25, 50, 75$. As general trends, FCAC shows the lowest rank (i.e., best) with a statistically significant difference from k-FED and FedFCM (except for $\epsilon = 15$). This indicates that FCAC can maintain higher clustering performance for various privacy-preserving datasets than the other state-of-the-art algorithms.

matrix by using CIM is $\mathcal{O}((\frac{\lambda}{2})^2 dK)$ (a line 8 in Alg. 1), for calculating determinant of the pairwise similarity matrix is $\mathcal{O}((\frac{\lambda}{2})^3)$ (a line 9 in Alg. 1), for computing CIM is $\mathcal{O}(ndK)$ (a line 16 in Alg. 1), for finding nodes which have the 1st and 2nd smallest CIM value is $\mathcal{O}(K)$ (a line 17 in Alg. 1), and for estimating the edge deletion threshold is $\mathcal{O}(|\mathcal{E}| \log |\mathcal{E}|)$ (lines 33-36 in Alg. 1).

In general, $n < \lambda^3$, $K \ll n$, and $\lambda < K$. As a result, the computational complexity of FCAC is $\mathcal{O}(\lambda^3 dK)$.

VIII. CONCLUDING REMARKS

In this paper, we introduced FCAC, a novel privacy-preserving federated clustering algorithm that leverages an ART-based clustering algorithm capable of continual learning. FCAC addresses the limitations of conventional federated clustering algorithms, thus making it highly effective for data with unknown or continually changing distributions. Empirical studies with synthetic and real-world datasets showed that the clustering performance of FCAC is superior to state-of-the-art federated clustering algorithms while maintaining data privacy protection and continual learning ability.

As a practical clustering algorithm, FCAC provides strong data privacy protection and facilitates rapid adaptation to new data. FCAC also enables effective data integration, making it a potentially useful tool in data-driven industries.

A future research topic will focus on integrating deep learning techniques with FCAC to further enhance its clustering performance and applicability to more complex and high-dimensional datasets. Additionally, exploring the potential of FCAC in various industry-specific contexts, such as healthcare and finance, could provide valuable insights.

REFERENCES

[1] P. Sun, "Security and privacy protection in cloud computing: Discussions and challenges," *J. Netw. Comput. Appl.*, vol. 160, Jun. 2020, Art. no. 102642.

[2] A. Majeed, S. Khan, and S. O. Hwang, "Toward privacy preservation using clustering based anonymization: Recent advances and future research outlook," *IEEE Access*, vol. 10, pp. 53066–53097, 2022.

[3] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Proc. Theory Cryptogr. Conf.* New York, NY, USA: Springer, 2006, pp. 265–284.

[4] C. Dwork, "Differential privacy," in *Proc. Int. Colloq. Automata, Lang., Program.* Cham, Switzerland: Springer, 2006, pp. 1–12.

[5] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, vol. 54, 2017, pp. 1273–1282.

[6] Y. Zhao, J. Zhao, M. Yang, T. Wang, N. Wang, L. Lyu, D. Niyato, and K.-Y. Lam, "Local differential privacy-based federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 8836–8853, Jun. 2021.

[7] Q. Li, Z. Wen, Z. Wu, S. Hu, N. Wang, Y. Li, X. Liu, and B. He, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3347–3366, Apr. 2023.

[8] C. Biswas, D. Ganguly, D. Roy, and U. Bhattacharya, "Privacy preserving approximate K-means clustering," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 1321–1330.

[9] A. M. Ikotun, A. E. Ezugwu, L. Abualigah, B. Abuhaija, and J. Heming, "K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data," *Inf. Sci.*, vol. 622, pp. 178–210, Apr. 2023.

[10] D. K. Dennis, T. Li, and V. Smith, "Heterogeneity for the win: One-shot federated clustering," in *Proc. 38th Int. Conf. Mach. Learn.*, vol. 139, Jul. 2021, pp. 2611–2620.

[11] W. Pedrycz, "Federated FCM: Clustering under privacy requirements," *IEEE Trans. Fuzzy Syst.*, vol. 30, no. 8, pp. 3384–3388, Aug. 2022.

[12] M. Stallmann and A. Wilbik, "On a framework for federated cluster analysis," *Appl. Sci.*, vol. 12, no. 20, p. 10455, Oct. 2022.

[13] V. Pandhare, X. Jia, and J. Lee, "Collaborative prognostics for machine fleets using a novel federated baseline learner," in *Proc. Annu. Conf. PHM Soc.*, 2021, vol. 13, no. 1, pp. 1–10.

[14] 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.

[15] 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.

[16] 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, Jan. 1991.

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

[18] 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.

[19] 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.

[20] N. Masuyama, Y. Nojima, C. K. Loo, and H. Ishibuchi, "Multi-label classification via adaptive resonance theory-based clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 7, pp. 8696–8712, Jul. 2023.

[21] N. Masuyama, T. Takebayashi, Y. Nojima, C. Kiong Loo, H. Ishibuchi, and S. Wermter, "A parameter-free adaptive resonance theory-based topological clustering algorithm capable of continual learning," 2023, *arXiv:2305.01507*.

[22] G. M. van de Ven and A. S. Tolias, "Three scenarios for continual learning," 2019, *arXiv:1904.07734*.

[23] F. Wiewel and B. Yang, "Localizing catastrophic forgetting in neural networks," 2019, *arXiv:1906.02568*.

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

[25] 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.

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

[27] C. Zhang, H. Li, C. Chen, X. Jia, and C. Chen, "Low-rank tensor regularized views recovery for incomplete multiview clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 7, pp. 9312–9324, Dec. 2022.

[28] C. Zhang, D. Xu, X. Jia, C. Chen, and H. Li, "Continual multi-view clustering with consistent anchor guidance," in *Proc. 33rd Int. Joint Conf. Artif. Intell.*, Aug. 2024, pp. 5434–5442.

[29] S. Acikalin and H. H. Yildirim, "Comparison of banks with cluster analysis before and after Covid-19 pandemic," *J. Corporate Governance, Insurance Risk Manage.*, vol. 8, no. 2, pp. 170–184, May 2021.

[30] Y. Wang, Z. Wu, W. Fu, J. Du, and Y. Zhang, "Spatial and seasonal dynamics of air quality in the Beijing-Tianjin-Hebei region: An analysis using K-means clustering and BP neural networks," *Opportunities Challenges Sustainability*, vol. 2, no. 4, pp. 184–196, Nov. 2023.

[31] T. Wang, X. Wang, and H. Li, "Enhanced prediction accuracy in complex systems: An approach integrating fuzzy K-clustering and fuzzy neural network," *Int. J. Knowl. Innov. Stud.*, vol. 1, no. 1, pp. 30–43, Sep. 2023.

[32] F. Peovski, "The cryptocurrency market through the scope of volatility clustering and leverage effects," *Acadlore Trans. Appl. Math. Statist.*, vol. 1, no. 3, pp. 130–147, Nov. 2023.

[33] M. Imeni, Z. Bao, and V. Nozick, "Multiscale partial correlation analysis of Tehran stock market indices: Clustering and inter-index relationships," *J. Oper. Strategic Anal.*, vol. 2, no. 1, pp. 1–10, Mar. 2024.

[34] M. J. Hoque, M. S. Islam, and S. A. Mohtasim, "Optimizing decision-making through customer-centric market basket analysis," *J. Oper. Strategic Anal.*, vol. 2, no. 2, pp. 72–83, Apr. 2024.

[35] S. Sen, L. Sahoo, and S. L. Ghosh, "Lifetime extension of wireless sensor networks by perceptive selection of cluster head using K-means and Einstein weighted averaging aggregation operator under uncertainty," *J. Ind. Intell.*, vol. 2, no. 1, pp. 54–62, Mar. 2024.

[36] O. O. Olusanya, G. M. Adebajo, I. Giwa, K. Okokpujie, S. A. Daramola, and A. V. Akingunsoye, "Neuro-fuzzy logic controller for switching capacitor banks in power factor correction within the manufacturing industry," *J. Intell. Syst. Control*, vol. 3, no. 2, pp. 93–106, Jun. 2024.

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

[38] F. Shen and O. Hasegawa, "A fast nearest neighbor classifier based on self-organizing incremental neural network," *Neural Netw.*, vol. 21, no. 10, pp. 1537–1547, Dec. 2008.

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

[40] 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, Oct. 2002.

[41] 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.

[42] Y. Toda and N. Masuyama, "Adaptive resonance theory-based global topological map building for an autonomous mobile robot," *IEEE Access*, vol. 12, pp. 111371–111385, 2024.

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

[44] N. Masuyama, C. K. Loo, and S. Wermter, "A kernel Bayesian adaptive resonance theory with a topological structure," *Int. J. Neural Syst.*, vol. 29, no. 5, Jun. 2019, Art. no. 1850052.

[45] L. E. Brito da Silva, I. Elnabarawy, and D. C. Wunsch, "Dual vigilance fuzzy adaptive resonance theory," *Neural Netw.*, vol. 109, pp. 1–5, Jan. 2019.

[46] L. E. Brito da Silva and D. C. Wunsch, "Validity index-based vigilance test in adaptive resonance theory neural networks," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2017, pp. 1–8.

[47] N. Masuyama, N. Amako, Y. Yamada, Y. Nojima, and H. Ishibuchi, "Adaptive resonance theory-based topological clustering with a divisive hierarchical structure capable of continual learning," *IEEE Access*, vol. 10, pp. 68042–68056, 2022.

[48] L. Meng, A.-H. Tan, and D. C. Wunsch, "Adaptive scaling of cluster boundaries for large-scale social media data clustering," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 12, pp. 2656–2669, Dec. 2016.

[49] A. Kulesza, "Determinantal point processes for machine learning," *Found. Trends Mach. Learn.*, vol. 5, nos. 2–3, pp. 123–286, 2012.

[50] J. Parker-Holder, A. Pacchiano, K. M. Choromanski, and S. J. Roberts, "Effective diversity in population based reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33. Red Hook, NY, USA: Curran Associates, 2020, pp. 18050–18062.

[51] P. Bunn and R. Ostrovsky, "Secure two-party k-means clustering," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, Oct. 2007, pp. 486–497.

[52] E. Zhang, H. Li, Y. Huang, S. Hong, L. Zhao, and C. Ji, "Practical multi-party private collaborative k-means clustering," *Neurocomputing*, vol. 467, pp. 256–265, Jan. 2022.

[53] J. Anju and R. Shreelekshmi, "A faster secure content-based image retrieval using clustering for cloud," *Expert Syst. Appl.*, vol. 189, Mar. 2022, Art. no. 116070.

[54] A. Shivhare, M. K. Maurya, J. Sarif, and M. Kumar, "A secret sharing-based scheme for secure and energy efficient data transfer in sensor-based IoT," *J. Supercomput.*, vol. 78, no. 15, pp. 17132–17149, May 2022.

[55] U. Stemmer and H. Kaplan, "Differentially private k-means with constant multiplicative error," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31. Red Hook, NY, USA: Curran Associates, 2018, pp. 1–11.

[56] P. Tang, X. Cheng, S. Su, R. Chen, and H. Shao, "Differentially private publication of vertically partitioned data," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 2, pp. 780–795, Mar. 2021.

[57] F. Fioretto, P. Van Hentenryck, and K. Zhu, "Differential privacy of hierarchical census data: An optimization approach," *Artif. Intell.*, vol. 296, Jul. 2021, Art. no. 103475.

[58] M. Yang, I. Tjuawinata, and K.-Y. Lam, "K-means clustering with local d_x-privacy for privacy-preserving data analysis," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 2524–2537, 2022.

[59] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," *Adv. Neural Inf.*, vol. 33, no. 12, pp. 19586–19597, Dec. 2022.

[60] Y. Liu, Z. Ma, Z. Yan, Z. Wang, X. Liu, and J. Ma, "Privacy-preserving federated k-means for proactive caching in next generation cellular networks," *Inf. Sci.*, vol. 521, pp. 14–31, Jun. 2020.

[61] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy C-means clustering algorithm," *Comput. Geosci.*, vol. 10, nos. 2–3, pp. 191–203, 1984.

[62] C. Pan, J. Sima, S. Prakash, V. Rana, and O. Milenkovic, "Machine unlearning of federated clusters," in *Proc. 11th Int. Conf. Learn. Represent.*, 2023, pp. 1–27.

[63] J. Yuan and Y. Tian, "Practical privacy-preserving MapReduce based K-means clustering over large-scale dataset," *IEEE Trans. Cloud Comput.*, vol. 7, no. 2, pp. 568–579, Apr. 2019.

[64] A. Shewale, B. Keshavamurthy, and C. N. Modi, "An efficient approach for privacy preserving distributed K-means clustering in unsecured environment," in *Proc. 5th Recent Findings Intell. Comput. Techn.* Cham, Switzerland: Springer, 2019, pp. 425–431.

[65] Y. Wang, M. Jia, N. Gao, L. Von Krannichfeldt, M. Sun, and G. Hug, "Federated clustering for electricity consumption pattern extraction," *IEEE Trans. Smart Grid*, vol. 13, no. 3, pp. 2425–2439, May 2022.
[66] H. Wang, A. Li, B. Shen, Y. Sun, and H. Wang, "Federated multi-view spectral clustering," *IEEE Access*, vol. 8, pp. 202249–202259, 2020.
[67] C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Found. Trends Theor. Comput. Sci.*, vol. 9, nos. 3–4, pp. 211–407, 2013.
[68] L. Devroye, "Sample-based non-uniform random variate generation," in *Proc. 18th Conf. Winter Simulation (WSC)*, 1986, pp. 260–265.
[69] 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.
[70] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*. Evanston, IL, USA: Routledge, 2018.
[71] G. Peyré and M. Cuturi, "Computational optimal transport: With applications to data science," *Found. Trends Mach. Learn.*, vol. 11, nos. 5–6, pp. 355–607, 2019.
[72] L. Hubert and P. Arabie, "Comparing partitions," *J. Classification*, vol. 2, no. 1, pp. 193–218, Dec. 1985.
[73] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: Networked science in machine learning," *ACM SIGKDD Explorations Newslett.*, vol. 15, no. 2, pp. 49–60, Jun. 2014.
[74] T. Lin, L. Kong, S. U. Stich, and M. Jaggi, "Ensemble distillation for robust model fusion in federated learning," in *Proc. NIPS*, vol. 33, 2020, pp. 2351–2363.
[75] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance," *J. Mach. Learn. Res.*, vol. 11, pp. 2837–2854, Dec. 2010.
[76] A. Strehl and J. Ghosh, "Cluster ensembles—A knowledge reuse framework for combining multiple partitions," *J. Machince Learn. Res.*, vol. 3, pp. 583–617, Jun. 2002.
[77] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, no. 1, pp. 1–30, Dec. 2006.

NAOKI MASUYAMA (Member, IEEE) received the B.Eng. degree from Nihon University, Funabashi, Japan, in 2010, the M.E. degree from Tokyo Metropolitan University, Hino, Japan, in 2012, and the Ph.D. degree from the Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia, in 2016.

He is currently an Associate Professor with the Department of Core Informatics, Graduate School of Informatics, Osaka Metropolitan University, Sakai, Japan. His current research interests include clustering, data mining, and continual learning.

YUSUKE NOJIMA (Member, IEEE) received the B.S. and M.S. degrees in mechanical engineering from Osaka Institute of Technology, Osaka, Japan, in 1999 and 2001, respectively, and the Ph.D. degree in system function science from Kobe University, Hyogo, Japan, in 2004.

Since 2004, he has been with Osaka Prefecture University, Osaka, where he was a Professor with the Department of Computer Science and Intelligent Systems, in October 2020. Since April 2022, he has been a Professor with the Department of Core Informatics, Graduate School of Informatics, Osaka Metropolitan University. His research interests include evolutionary fuzzy systems, evolutionary multiobjective optimization, and multiobjective data mining.

Dr. Nojima was a Task Force Chair on Evolutionary Fuzzy Systems on the Fuzzy Systems Technical Committee of the IEEE Computational Intelligence Society. He was a guest editor of several special issues in international journals. He was an Associate Editor of *IEEE Computational Intelligence Magazine, from 2014 to 2019.*

YUICHIRO TODA (Member, IEEE) received the B.E., M.E., and Ph.D. degrees from Tokyo Metropolitan University, Hino, Japan, in 2011, 2013, and 2017, respectively.

He was an Assistant Professor with the Faculty of Engineering, Okayama University, Okayama, Japan, in 2018. He is currently an Associate Professor with the Faculty of Environmental, Life, Natural Science and Technology, Okayama University. His research interests include computational intelligence and intelligent robotics in unknown environments. He has published more than 80 refereed journal and conference papers in the interest research area.

CHU KIONG LOO (Senior Member, IEEE) received the B.Eng. degree (Hons.) in mechanical engineering from University Sains Malaysia and the Ph.D. degree from the University of Malaya.

He was a Design Engineer in various industrial firms and is currently the Founder of the Advanced Robotics Laboratory, University of Malaya, Malaysia. He has been involved in the application of research into Perus's Quantum Associative Model and Pribram's Holonomic Brain Model in humanoid vision projects. He is a Professor of computer science and information technology with the University of Malaya. He has led many projects funded by the Ministry of Science, Malaysia, and the High Impact Research Grant, Ministry of Higher Education, Malaysia. His research interests include brain-inspired quantum neural networks, constructivism-inspired neural networks, synergetic neural networks, and humanoid research.

HISAO ISHIBUCHI (Fellow, IEEE) received the B.S. and M.S. degrees in precision mechanics from Kyoto University, Kyoto, Japan, in 1985 and 1987, respectively, and the Ph.D. degree in computer science from Osaka Prefecture University, Sakai, Osaka, Japan, in 1992.

Since 1987, he has been with Osaka Prefecture University, for 30 years. He is currently a Chair Professor with the Department of Computer Science and Engineering, Southern University of Science Technology, Shenzhen, China. His current research interests include fuzzy rule-based classifiers, evolutionary multiobjective optimization, many-objective optimization, and memetic algorithms.

Dr. Ishibuchi was the IEEE Computational Intelligence Society (CIS) Vice President of Technical Activities from 2010 to 2013. He was an IEEE CIS AdCom Member from 2014 to 2019 and from 2021 to 2023, an IEEE CIS Distinguished Lecturer from 2015 to 2017 and from 2021 to 2023, and the Editor-in-Chief of *IEEE Computational Intelligence Magazine* from 2014 to 2019. He is also an Associate Editor of *ACM Computing Survey*, IEEE Transactions on Cybernetics, and IEEE Access.

NAOYUKI KUBOTA (Senior Member, IEEE) received the M.Eng. degree from Hokkaido University, Hokkaido, Japan, in 1994, and the D.E. degree from Nagoya University, Nagoya, Japan, in 1997.

He joined Osaka Institute of Technology, Osaka, Japan, in 1997. In 2000, he joined the Department of Human and Artificial Intelligence Systems, Fukui University, Fukui, Japan, as an Associate Professor. He joined the Department of Mechanical Engineering, Tokyo Metropolitan University, Tokyo, Japan, in 2004, where he is currently a Professor with the Department of System Design.