Federated c-means and Fuzzy c-means Clustering Algorithms for Horizontally and Vertically Partitioned Data

José Luis Corcuera Bárcena, Francesco Marcelloni, *Member, IEEE,* Alessandro Renda, Alessio Bechini, Pietro Ducange

***Abstract*—Federated clustering lets multiple data owners collaborate in discovering patterns from distributed data without violating privacy requirements. The federated versions of traditional clustering algorithms proposed so far are, however, "lossy" since they fail to identify exactly the same clusters as the original versions executed on the merged data stored in a centralized server, as would happen if no privacy constraint occurred. In this paper, we propose federated procedures for losslessly executing the C-Means (CM) and the Fuzzy C-Means (FCM) algorithms in both horizontally and vertically partitioned data scenarios, while preserving data privacy. We formally prove that the proposed federated procedures identify the same clusters determined by applying the algorithms to the union of all local data. Further, we present an extensive experimental analysis for characterizing the behavior of the proposed approach in a typical federated learning scenario, that is, as the fraction of participants in the federation changes. We focus on the federated FCM and the horizontally partitioned data, which is the most interesting scenario. We show that the proposed procedure is effective and is able to achieve competitive performance with respect to two recently proposed versions of federated FCM for horizontally partitioned data.**

***Impact Statement*—Data access limitations in a distributed setting, often due to privacy needs, represent an obstacle to extracting knowledge from data belonging to different owners: to overcome such an issue, Federated Learning (FL) has been proposed. So far, unsupervised FL, and federated clustering in particular, has received less attention than supervised FL, although several interesting applications cannot rely on labelled data. In addition, the few federated versions of traditional clustering algorithms proposed in the literature do not faithfully reproduce the behavior of the centralized versions applied to all merged data. This paper provides data scientists with horizontal and vertical federated versions, behaving exactly like the original ones, of the well-known and popular CM and FCM algorithms, thus avoiding possible misleading interpretations. The proposed versions can be effectively used in sensitive domains like healthcare, finance, and telecommunications, yet complying with stringent data regulations and policies.**

***Index Terms*—Federated Clustering, Federated Learning, fuzzy c-means, k-means,**

I. Introduction

FEDERATED Learning (FL) [1] has been recently proposed as a solution for the collaborative training of an ML model, to overcome the shortcomings of data centralization. In an FL system, a shared global model is learned by aggregating locally-computed updates from remote data owners, with no need to expose raw private data across parties.

Algorithmic challenges in FL depend on data distribution patterns [1], [2]: in *horizontally* partitioned data, objects are spread over multiple distinct nodes, and the same set of features describes all of them. Conversely, in a *vertical* partitioning scenario the information for a single object is split across multiple nodes, each with a partial view (a subset of features) of all the objects.

Since the introduction of FL by Google [3], [4], most of the work on FL has addressed the horizontal setting, revolving around the centralized *federated averaging* (FedAvg) protocol for model aggregation [3]. FedAvg is a round-based, collaborative Stochastic Gradient Descent (SGD) optimization procedure, with the following steps performed in each round: (i) the server sends out the current global model to data owners (or a subset thereof); (ii) each selected data owner updates the local model via SGD on its examples; (iii) each selected data owner sends back its updated model to the server; (iv) the server obtains a new global model by averaging the locally updated models, weighted according to the number of examples.

Several extensions of FedAvg have been proposed to adapt FL to heterogeneous settings [5]–[8]. Mainstream FL typically makes use of SGD-optimized models. Indeed, FL has been primarily employed for the collaborative training of deep neural networks for supervised learning, e.g., in image classification and language modelling tasks. Other ML techniques, and clustering in particular, have been much less explored in FL [9]. Nonetheless, clustering plays a pivotal role in several real-world applications. In the healthcare domain, for instance, it is used for medical diagnosis, biological data analysis, and medical image segmentation [10]. However, privacy issues hamper the sharing of medical data, typically kept in isolated data centers: such a setting prevents the use of traditional clustering algorithms and asks for novel paradigms properly designed for a distributed environment. Another application is the recognition of driving styles for real-world drivers, which offers benefits like safer driving and cost efficiency [11]. Typically, clustering methods group drivers based on trajectory data. However, a traditional, centralized approach involves collecting and sharing private drivers' data. Federated clustering can overcome such a privacy problem.

Clustering is useful not only for the identification of groups within a dataset: it is also frequently applied as a numerosity

All the authors are with the Department of Information Engineering of the University of Pisa, Largo Lucio Lazzarino 1, 56122 Pisa, Italy. (e-mail: jose.corcuera@ing.unipi.it, {francesco.marcelloni, alessandro.renda, alessio.bechini, pietro.ducange}@unipi.it)

reduction technique [12] or as a preprocessing step in data mining pipelines, e.g., for the estimation of antecedent parameters in Fuzzy Rule-based Systems [13]. Thus, it is crucial to revisit the most popular clustering algorithms to adapt them to the FL setting.

The overarching goal of this work is the proposal of a unifying framework for *federated clustering* (i.e., identification of clusters of unlabelled data instances spread over multiple, distributed clients, with no sharing of raw data), for both horizontally and vertically partitioned data. This ensures privacy preservation, a fundamental requirement toward trustworthy AI [14], urgently advocated by governmental entities and public opinion. As privacy enforcement impedes the collection of distributed data for centralized processing, clustering might be carried out locally, reaching results different from what could be obtained working on all data. Federated clustering allows for uncovering "global" patterns, ensuring users' privacy preservation as well. The main challenge of this task consists of revisiting the traditional optimization procedures of clustering algorithms to work over a federated setting, possibly generating the same clusters obtainable by centralizing all the data. Furthermore, other issues must be addressed: the number of clients can grow fast, their participation in the federation may be unstable, and their data may differ in terms of distribution and volumes.

In this paper, we propose an unifying framework for the FL execution of the traditional c-means (CM) [15] and fuzzy c-means (FCM) [16] algorithms for both horizontally and vertically partitioned data[1], stemming from initial partial results presented in one of our previous works [17]. We mainly choose these two algorithms because of their popularity, due to their intuitive operation and computational efficiency. Moreover, with the coverage of the fuzzy variant besides the crisp one, we make federated clustering applicable also in scenarios requiring uncertain attribution of objects to clusters. Even if, for the fuzziness factor set to 1, the traditional FCM reduces to FC, its time complexity is higher than that of CM, since it is quadratic in the number of clusters instead of linear. Inherent limitations of CM and FCM, typical of partitioning methods, have been studied in the literature: e.g., we can recall the sensitivity to outliers and the inability to capture non-convex clusters. Federated variants of other clustering approaches (e.g., hierarchical, density-based) may equally deserve attention, but they do not fit into the same unifying framework and require ad-hoc methods, and are thus beyond the scope of this paper.

Although both CM and FCM aim to minimise a differentiable global objective function, neither FedAvg nor a gradient-descent approach is adopted in our proposal. Instead, we define a strategy to alternately execute the object assignment to clusters and the centres' update in a collaborative way. Such steps correspond to those of the traditional algorithms for a global dataset stored in a central server. We theoretically prove that, given the same initial centroids, our proposed federated versions obtain the same results as the centralised counterparts, yet enabling privacy-preserving clustering among scattered data owners.

[1]Source code available: https://github.com/Unipisa/FederatedClustering

The main contributions of our work can be summarized as follows:

- novel federated versions of CM and FCM (named LLF-CM and LLF-FCM, respectively) are proposed for both horizontally and vertically partitioned data, presented according to a general and unifying federated framework, named LLF (*LossLess Federation)*;
- the LLF framework enables federated clustering over decentralized data, thus preserving the privacy of data owners; furthermore, we formally show that the behavior of federated clustering algorithms is equivalent to the one of the respective centralized versions applied to the merged data partitions;
- an extensive experimental analysis carried out on horizontal LLF-FCM shows the robustness of the approach with respect to the number of contributing clients, and its competitive performance compared to existing works on federated FCM;
- an approach for carefully initialising the horizontal LLF-FCM (in a privacy-preserving manner) is introduced and shown to improve the clustering accuracy.

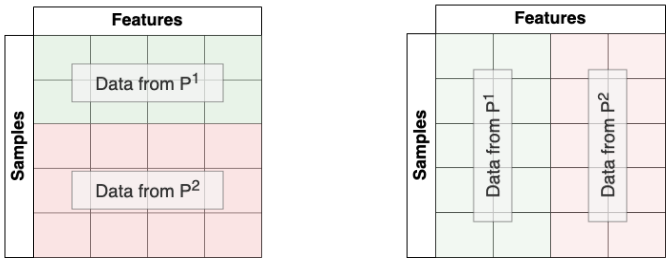
The rest of the paper is organized as follows: Section II discusses related works. Section III describes the background regarding FL, and the general setting of our investigation. Sections IV and V introduce our federated versions of CM and FCM over horizontally and vertically partitioned data, respectively. In Section VI we discuss some experimental results. Finally, Section VII reports some concluding remarks.

II. Related Works

The CM (a.k.a. k-means) clustering algorithm partitions the dataset objects into C clusters (each indicated by Γ_c, with $c = \{1, \ldots, C\}$) in which each object belongs to the cluster with the nearest mean (*cluster prototype*); the traditional batch optimization procedure is known as Lloyd's algorithm [15]. CM is among the most popular clustering algorithms, and results in a *crisp* or *hard* partitioning, where each object belongs only to one single cluster. The FCM clustering algorithm [16] leverages the fuzzy set theory, extensively used in cluster analysis [18], and allows for *fuzzy* partitioning: an object may belong to multiple clusters with a membership degree in $[0, 1]$. The number of clusters is an input parameter in both algorithms.

Many adaptations of CM and FCM have been proposed for different application scenarios. In this section, we discuss approaches concerning the decentralized setting, and able to comply with the privacy needs of data owners. Notably, FL is known to address such requirements, but it has been scarcely investigated for unsupervised learning. However, relevant extensions of CM and FCM have been proposed in other research areas that share some common traits with FL.

In a distributed setting, data are spread across multiple distinct parties (or *data owners*) usually according to some given pattern, indicated as *data partitioning*. In practice, the most reasonable and frequently occurring ones are the *horizontal* and *vertical* schemes (see Fig. 1b), which ask for different FL



(a) Horizontal FL (b) Vertical FL

Fig. 1. Data partitioning patterns. Example with two data owners P^1 and P^2: (a) Horizontal and (b) Vertical.

algorithms. Let M be the number of data owners; the feature space, the local dataset, and the sample ID space of the m-th data owner are indicated by F^m, $\mathbf{X}^m$, and I^m, respectively. Horizontal partitioning is formally described as follows:

$$F^m = F^n, \quad I^m \neq I^n \quad \forall\, \mathbf{X}^m, \mathbf{X}^n \text{ with } m \neq n \tag{1}$$

In this case, the dataset is said to be *sample-partitioned*. Similarly, the vertical partitioning (Fig. 1b) is defined as

$$F^m \neq F^n, \quad I^m = I^n \quad \forall\, \mathbf{X}^m, \mathbf{X}^n \text{ with } m \neq n \tag{2}$$

The dataset is said to be *feature-partitioned*: each data owner holds the values for a specific subset of the features only, across all the samples.

The field of *distributed ML* has similarities with horizontal FL [1], but it just exploits multiple nodes to boost processing power [19]. The MapReduce programming model [20] has been used for a structured parallelization of the CM algorithm [21]. For example, Balcan et al. [22] propose distributed coreset-based solutions for c-means and c-median clustering with no centralized coordinator. Besides the focus on privacy preservation, a set of key properties can tell apart FL from distributed ML [3]: in FL, data from different participants may have different volumes and distribution (non-i.i.d.); furthermore, the number of involved clients may be arbitrarily large and variable due to limited and unstable communication.

The area of Secure Multi-Party Computation (SMC), pioneered by Yao in the 1980s [23], provides a framework to enable multiple parties to compute an agreed-upon function on their private input [9]. Cryptography allows SMC to achieve complete zero-knowledge (each party knows nothing except its input and output) without sacrificing accuracy [1], [24]. Several contributions on privacy-preserving CM clustering algorithms have been presented [25]: although the iterative nature of CM makes zero-knowledge unattainable, private implementations have been proposed both for horizontally [26] and for vertically [27] partitioned data. However, cryptography and SMC tools introduce severe computation overheads and hamper the scalability of the approaches [24], [28].

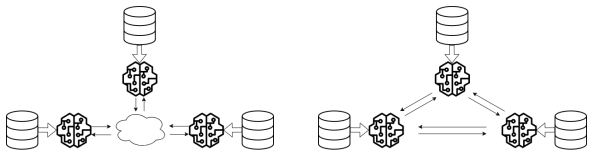
Recently, Lyu et al. [24] have addressed privacy-preserving collaborative fuzzy clustering by preceding the clustering stage (based on FCM) with non-linear transformation and random projection of local raw data onto a lower dimensional space. The approach is shown to be effective, at a limited cost in terms of accuracy, in mitigating specific attacks carried out by the central server, which may also collude with some of the participants to violate user privacy.

Unlike the works mentioned above, however, in this paper we assume a *semi-honest* central server [29]. Despite its relevance in real-world applications, especially for the horizontal partitioning [1], clustering under such *weak* privacy model has not been adequately investigated in the FL literature. Conversely, privacy issues are neglected in a recent work focused on the federated implementation of the CM algorithm for heterogeneous environments [30].

To the best of our knowledge, only two recent works [31], [32] on federated clustering assume the same privacy model and communication topology (i.e., orchestration by a central server), and reference algorithms as we do. The first work describes a horizontal federated version of FCM that exploits a gradient-descent optimization procedure [31]. More specifically, each FL round encompasses the following steps: (i) the server shares the prototypes with the clients, (ii) each client locally executes several iterations of the traditional FCM procedure by alternately updating the partition matrix and the prototypes; then it computes the gradient of the FCM cost function with respect to the prototypes based on local data and sends it back to the server; (iii) the server evaluates the average of the gradients received by the clients, weighted with number of training samples on each node and, similarly to FedSGD [3], updates the prototype through a step of gradient-descent optimization. In our paper we refer to this algorithm as GDF-FCM (after "Gradient Descent Federation"). Its results are shown to be close to, but not equal, those of the traditional FCM algorithm applied to the global dataset. As described, the GDF-FCM optimization does not follow the traditional Lloyd's procedure, and may be subject to instability.

A more recent proposal for federated FCM adopts a round-based optimization as well [32]. Each round is composed of the following steps: (i) the server shares the prototypes with the clients, (ii) each client executes locally the traditional FCM procedure up to convergence, and returns its (locally computed) "candidate" prototypes to the server; (iii) the server updates the global prototypes through CM applied on the set of candidate prototypes received from the clients.. The k-means++ initialization [33] is used to improve both the CM accuracy and speed. In our paper we refer to this algorithm as C2F-FCM (after "Clustering of Clusterings Federation"). As it happened for GBD-FCM [31], also C2F-FCM does not guarantee the obtained fuzzy partition to be the same as in the centralized case. Furthermore, the execution of each round is computationally heavy: each client must run a complete FCM execution, and the server must run CM on the candidate prototypes returned by the clients.

We say that a federated version of a clustering algorithm is *lossless* with respect to the traditional version of the same algorithm applied to the global dataset (obtained by merging the local data) if both obtain exactly the same clusters. Unlike the mentioned state-of-art approaches, our LLF versions closely adhere to the Lloyd's optimization procedure and are shown to be lossless, yet preserving the participants' privacy. Further, our LLF approach covers the case of vertically partitioned data, while the two others do not.



(a) Centralized FL (b) Fully-decentralized FL

Fig. 2. FL categorization based on communication topology: (a) Centralized FL and (b) Fully-decentralized FL.

III. Federated Learning Background

Several recent surveys presented thorough overviews of FL [1], [2], [9], [28], [34], [35]. In this section, we introduce the used notation and recall our proposal's most important aspects.

Let $P^1, P^2, \ldots, P^M$ be M parties, also referred to as *clients* or *data owners*, and $\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^M$ their relative local datasets. The *global dataset* is the union of all the local datasets: $\mathbf{X}_{sum} = \bigcup_{m=1}^{M} \mathbf{X}^m$. The local dataset of the m-th party is $\mathbf{X}^m = \{\mathbf{x}_1^m, \mathbf{x}_2^m, \ldots, \mathbf{x}_{N_m}^m\}$: it contains N_m objects, each indicated as $\mathbf{x}_j^m$.

An FL system is a learning process whose participants collaboratively train a model $Model_{fed}$ without exposing their (private) data to others. Let $Model_{sum}$ be the model trained on $\mathbf{X}_{sum}$: such global dataset can be obtained only by gathering all data in a *centralized* location, clearly violating privacy requirements. $Model_{fed}$ is learned with the federated version of the ordinary ML algorithm used for $Model_{sum}$. We expect $Model_{fed}$ to closely trails $Model_{sum}$. In clustering, the clusters generated by the federated versions ($Model_{fed}$) are expected to closely approximate those by the centralised versions ($Model_{sum}$). We will prove that our proposed federated versions of the clustering algorithms are lossless.

The FL paradigm may be used in scenarios characterized by different communication topologies and scales of federation [28], [35]. E.g., depending on the *communication topology*, we may have *centralized* and *fully-decentralized* FL systems. In the former (Fig. 2a), a central server orchestrates the learning process and aggregates the model updates received from the parties. In the latter (Fig. 2b), information across the parties is shared peer-to-peer, with no need for centralized coordination.

Scale of federation refers to the number of involved parties and data availability: cross-silo and cross-device FL are the two most common settings. In *cross-silo* FL, the clients are organizations or data centers, and only a few parties are typically present (ranging from two to a few tens), each with a relatively large amount of data and computational power. Conversely, in *cross-device* FL clients may be much more, and each party has limited data and computational power.

In this paper, we introduce federated versions of the CM and FCM clustering algorithms for both vertically and horizontally partitioned data in a centralized communication topology, with no limitation on the number of participants. Vertical FL typically adopts entity alignment techniques [28], aimed at spotting out the overlapped samples among the participants to the federation [36]; such alignment can be accomplished without compromising user privacy, as in the case of a downstream federated gradient tree-boosting over vertically partitioned data [37]. However, for the purpose of this work and with no loss of generality, we assume the samples to be aligned across the parties. Concerning the privacy model, we observe that the server assumes a key role in privacy issues in a centralized communication topology. As it is typical in FL [1], we assume honest participants and a *semi-honest*, or *honest-but-curious*, central server: it gathers information without modifying the FL protocol but may exploit it to uncover private raw data. Privacy is violated if the server (or, in general, any party with root access to the server and/or to its input and output messages) can unambiguously derive an instance of private raw data from a participant [29]. This can occur for instance when a data owner sends to the server statistics related to a unique instance in the case of the horizontal federated CM algorithm. We have extensively discussed privacy aspects in Sections IV-B and V-B for the horizontal and vertical scenarios, respectively.

Each party can benefit from participating in the federation since the clustering result is obtained collaboratively, exploiting information from all the involved local datasets.

The distributed nature of an FL algorithm asks for the adoption of proper communication and synchronization across the involved parties. For the sake of conciseness and clearness, the pseudocodes reported hereafter provide a general vision, with no explicit breakdown of the code lines between server and data owners, as would be expected; moreover, data transmissions are indicated without recurring to explicit primitives.

IV. Federated Clustering over Horizontally Partitioned Data

In this section, we focus on the scenario of *horizontal FL* with a *centralized* communication topology. M parties wish to obtain a partitioning of their data, taking advantage of a clustering model to be built collaboratively without sharing private local data. Let $\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^M$ be the M distinct private datasets, each with a variable number N_m of objects over the same F-dimensional feature space: $\mathbf{x}_j^m = \left\{x_{j,1}^m, x_{j,2}^m, \ldots, x_{j,F}^m\right\}$.

The federated versions of the CM and FCM clustering algorithms (namely, LLF-CM and LLF-FCM) are derived, respectively, from the traditional batch optimization [15] and the FCM version that exploits a cluster center updating procedure [38], which is more efficient than the original one proposed by Bezdek [16].

CM and FCM adopt the same optimization scheme, thus a unique horizontal federated clustering framework can be defined for both algorithms, whose steps are summarized in the sequence diagram in Fig. 3: it reports the actions of the server and each data owner in successive rounds of the federated algorithm. The complete pseudocode is reported in Algorithm 1 for the Horizontal LLF-CM and in Algorithms 2 and 3 for the Horizontal LLF-FCM.

The server is in charge of the initialization stage: in LLF-FCM, the fuzziness factor λ is selected and shared with each data owner. Furthermore C cluster centers are initialized (line 2 of both algorithms). If the server knows the domain of the features, it can randomly generate prototypes in that space: this strategy, adopted for example in [31], fits all applications

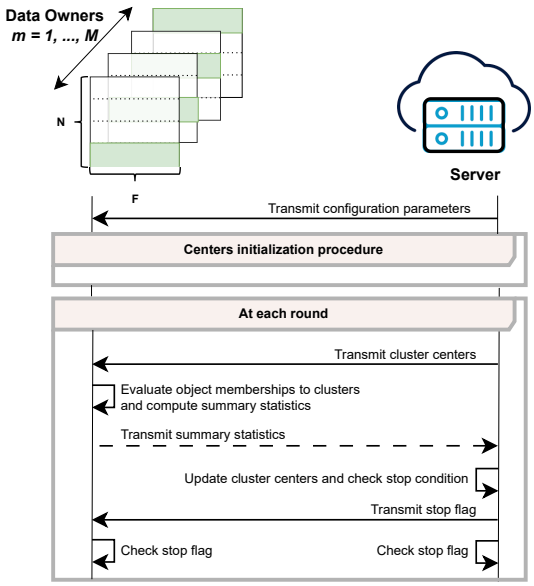


Fig. 3. Sequence diagram summarizing the execution of the federated CM and FCM algorithms over horizontally partitioned data.

Algorithm 1 Horizontal Federated c-means (LLF-CM). C: number of clusters, $\varepsilon > 0$: tolerance value for the stop condition. T: maximum number of rounds

Initialization stage
Server:
stop_flag = FALSE
Initialization procedure for C cluster centers $\mathbf{V}^{(0)} = \left\{\mathbf{v}_1^{(0)}, \ldots, \mathbf{v}_C^{(0)}\right\}$

Execution stage
At each round t, with t starting from 0:
Cluster assignment
Server:
Transmit $\mathbf{V}^{(t)}$ to each data owner
Each data owner P^m:
Evaluate $q_j^{(t),m} \in \{1, \ldots, C\}$ for each object j, as its own cluster assignment according to the nearest center $\mathbf{v}_c^{(t)}$
Centers update
Each data owner P^m:
for each cluster Γ_c **do**
  $n_c^{(t),m} \leftarrow$ count of the number of objects in cluster Γ_c
  if $n_c^{(t),m} > 1$ **then**
    compute $\mathbf{Ls}_c^{(t),m}$ as per Eq. 3
  else
    $(\mathbf{Ls}_c^{(t),m}, n_c^{(t),m}) \leftarrow (\mathbf{0}, 0)$
  end if
end for
Transmit to the server all the pairs $(\mathbf{Ls}_c^{(t),m}, n_c^{(t),m})$ calculated above
Server:
Update cluster centers evaluating $\mathbf{V}^{(t+1)}$ as per Eq. 5.
Termination
Server:
if NOT $\left(||\mathbf{V}^{(t+1)} - \mathbf{V}^{(t)}||_F < \varepsilon \quad \text{OR} \quad t > T\right)$ **then**
  stop_flag = TRUE
end if
Transmit *stop_flag* to each data owner
Each data owner P^m & Server:
if NOT *stop_flag* **then**
  Proceed with the next round (line 4)
end if
⟨Termination⟩

Algorithm 2 Horizontal Federated Fuzzy c-means (LLF-FCM). C: number of clusters, $\lambda > 1$: fuzziness factor, $\varepsilon > 0$: tolerance value for the stop condition. T: maximum number of rounds

Initialization stage
Server:
stop_flag = FALSE
Initialization procedure for C cluster centers $\mathbf{V}^{(0)} = \left\{\mathbf{v}_1^{(0)}, \ldots, \mathbf{v}_C^{(0)}\right\}$
Transmit the fuzziness factor λ to each data owner

Execution stage
At each round t, with t starting from 0:
Cluster assignment - Centers update
Server:
Transmit $\mathbf{V}^{(t)}$ to each data owner
Each data owner P^m:
$\mathbf{u}^{(t),m}, \mathbf{WS}^{(t),m} = \text{LocalSums}(\mathbf{V}^{(t)}, \mathbf{X}^m, \lambda)$
Transmit $(\mathbf{u}^{(t),m}, \mathbf{WS}^{(t),m})$ to the server
Server:
Update cluster centers evaluating $\mathbf{V}^{(t+1)}$ as per Eq. 7.
Termination
Server:
if NOT $\left(||\mathbf{V}^{(t+1)} - \mathbf{V}^{(t)}||_F < \varepsilon \quad \text{OR} \quad t > T\right)$ **then**
  stop_flag = TRUE
end if
Transmit *stop_flag* to each data owner
Each data owner P^m & Server:
if NOT *stop_flag* **then**
  Proceed with the next round (line 5)
end if
⟨Termination⟩

Algorithm 3 LocalSums($\mathbf{V}^{(t)}$, $\mathbf{X}^m$, λ)

Given: $\mathbf{V}^{(t)}$ - array of C cluster centers
Given: $\mathbf{X}^m$ - m-th dataset
Given: λ - fuzziness factor
$\mathbf{WS}^{(t),m} \leftarrow zeros(C \times F)$
$\mathbf{u}^{(t),m} \leftarrow zeros(C)$
for each $\mathbf{x}_j^m \in \mathbf{X}^m$ **do**
  $denom = 0$
  for each cluster c **do**
    $numer_c = \|\mathbf{x}_j^m - \mathbf{v}_c^{(t)}\|^{\frac{2}{\lambda-1}}$
    $denom = denom + \frac{1}{numer_c}$
  end for
  for each cluster c **do**
    $\mu_{c,j} = (numer_c * denom)^{-1}$
    $\mathbf{ws}_c^{(t),m} = \mathbf{ws}_c^{(t),m} + \mu_{c,j}^{\lambda} \mathbf{x}_j^m$
    $u_c^{(t),m} = u_c^{(t),m} + \mu_{c,j}^{\lambda}$
  end for
end for
return $\mathbf{u}^{(t),m}, \mathbf{WS}^{(t),m}$

where the ranges of the features are known or can be estimated, but is not viable in cases where they are unknown. An alternative strategy would consists in letting one of the participants initialize the centers and send them to the server: this ensures that the starting centroids are reasonable within the feature domain and that there is no violation of privacy (as they are randomly generated). An orthogonal challenge concerns the setting of parameter C, i.e. the number of clusters. In our experiments, we will let the centers be initialized by a random participant and we will set C as the number of classes in each dataset. Notably, the problem of selecting the number of clusters affects partitional clustering algorithms regardless of the learning setting (i.e., centralized or federated) and is typically addressed with heuristic approaches, such as the well-known elbow method. A heuristic based on the Soft Davies

Bouldin (SDB) index [39] compliant with the FL setting has been proposed in [32].

The execution stage is iterated until the stop condition occurs. At each round t, the server transmits the current array of cluster centers to each data owner. In the following, we provide details about the specific steps by highlighting the differences between LLF-CM and LLF-FCM.

LLF-CM: Each data owner P^m assigns each object in the local dataset to the cluster with the nearest center. Let $\mathbf{q}^{(t),m} = \left[q_1^{(t),m}, q_2^{(t),m}, \ldots, q_{N_m}^{(t),m}\right]$, $q_j^{(t),m} \in \{1, \ldots, C\}$, be the vector that, for each object $\mathbf{x}_j^m$, indicates the cluster it is assigned to. Then, for each cluster Γ_c, each data owner counts the relative local cardinality $n_c^{(t),m}$ and, if $n_c^{(t),m} > 1$, it computes

$$\mathbf{Ls}_c^{(t),m} = \sum_{\mathbf{x}_j^m \in \Gamma_c} \mathbf{x}_j^m \tag{3}$$

as the linear sum of the objects in cluster Γ_c, otherwise it sets both $\mathbf{Ls}_c^{(t),m}$ and $n_c^{(t),m}$ to 0 (this is done to enforce privacy preservation). The following information is then transmitted to the server:

$$(\mathbf{Ls}_c^{(t),m}, n_c^{(t),m}) \quad \forall\, c \in \{1, \ldots, C\} \tag{4}$$

The server, upon receiving it from all the data owners, can proceed to update the cluster centers:

$$\mathbf{v}_c^{(t+1)} = \frac{\sum_{m=1}^{M} \mathbf{Ls}_c^{(t),m}}{\sum_{m=1}^{M} n_c^{(t),m}}, \quad \forall\, c \in \{1, \ldots, C\} \tag{5}$$

LLF-FCM: The cluster assignment step consists of evaluating the membership degree of each object to each cluster as follows:

$$\mu_{c,j} = \left(\sum_{l=1}^{C} \left(\frac{\|\mathbf{x}_j^m - \mathbf{v}_c\|}{\|\mathbf{x}_j^m - \mathbf{v}_l\|} \right)^{\frac{2}{\lambda - 1}} \right)^{-1} \quad \forall\, j, c. \tag{6}$$

In the used version of FCM [38], storing the membership matrix $[\mu_{c,j}]$ is not necessary, with a significant reduction of the asymptotic runtime. Specifically, each data owner P^m computes both $\mathbf{u}^{(t),m}$ and $\mathbf{WS}^{(t),m}$ as described in Algorithm 3. $\mathbf{u}^m$ is an array with C elements, whose c-th element is the sum of the membership degrees of the objects in $\mathbf{X}_m$ to cluster Γ_c, each raised to the λ-th power. More formally, $\mathbf{u}^m = \{u_1^m, u_2^m, \ldots, u_C^m\}$ and $u_c^m = \sum_{j=1}^{N_m} \mu_{c,j}^{\lambda}$. $\mathbf{WS}^m$ is a $C \times F$ matrix, whose c-th row is the sum of the data objects, weighted by the membership degree of the objects to cluster Γ_c. More formally, $\mathbf{WS}^m = \{\mathbf{ws}_1^m, \mathbf{ws}_2^m, \ldots, \mathbf{ws}_C^m\}$ and $\mathbf{ws}_c^m = \sum_{j=1}^{N_m} \mu_{c,j}^{\lambda} \mathbf{x}_j^m$. Then, each data owner transmits $\mathbf{u}^{(t),m}$ and $\mathbf{WS}^{(t),m}$ to the server, so that it can update the coordinates of the centers as follows:

$$\mathbf{v}_c^{(t+1)} = \frac{\sum_{m=1}^{M} \mathbf{ws}_c^{(t),m}}{\sum_{m=1}^{M} u_c^{(t),m}} \quad \forall\, c \in \{1, \ldots, C\} \tag{7}$$

In both the LLF-CM and LLF-FCM cases, the server checks the stop condition, i.e. whether the centroids move less than a given tolerance (the threshold ε). Formally, the tolerance index is expressed as the Frobenius norm of the difference in the cluster centers between two consecutive rounds. If the stop condition is met, the execution terminates; otherwise, the server initiates the next round, by transmitting the new centers to the data owners. A maximum number of rounds can be expressed to force termination as well.

A. Equivalence with the traditional algorithms executed on the global dataset

Equivalence between the federated approaches and the corresponding original algorithms is demonstrated by showing that the center update step in the federated setting ($Model_{fed}$) leads to the same result of the centralized setting ($Model_{sum}$). Obviously, the equivalence holds as long as all clients participate in the federation. It is also worth noting that the termination condition depends on the overall number of rounds and the position of the cluster centers, as it evaluates their evolution over two consecutive rounds; thus, it can be applied equivalently in the case of centralized and federated settings.

1) LLF-CM: Given the same initial cluster centers randomly generated, Algorithm 1 produces the same clusters that would be produced by the traditional CM applied to the union $\mathbf{X}_{sum}$ of the local datasets. Indeed, assuming that all data are stored in the central server, at each iteration the cluster centers would be updated by the traditional CM as follows:

$$\mathbf{v}_c^{(t+1)} = \frac{\sum_{\mathbf{x}_j^{(t),sum} \in \Gamma_c} \mathbf{x}_j^{(t),sum}}{n_c^{(t),sum}} \quad \forall\, c \in \{1, \ldots, C\} \tag{8}$$

Notably, assuming that $n_c^{(t),m} > 1\ \forall\, c, m$, Eq. 8 leads to the same result of Eq. 5 since, for each cluster c:

$$\sum_{m=1}^{M} \mathbf{Ls}_c^{(t),m} = \sum_{\mathbf{x}_j^{(t),sum} \in \Gamma_c} \mathbf{x}_j^{(t),sum} \tag{9}$$

and

$$\sum_{m=1}^{M} n_c^{(t),m} = n_c^{(t),sum}. \tag{10}$$

In the centralized setting, each sum is carried out in a single step whereas in the federated setting, each sum is performed first on a per-owner basis and then aggregated by the central server.

2) LLF-FCM: Analogous considerations apply to LLF-FCM (Algorithms 2,3). Under the assumption that all data are stored in the central server, cluster centers would be updated by the traditional FCM as follows:

$$\mathbf{v}_c^{(t+1)} = \frac{\sum_{j=1}^{N} \mu_{c,j}^{\lambda} \mathbf{x}_j^{(t),sum}}{\sum_{j=1}^{N} \mu_{c,j}^{\lambda}}, \quad \forall\, c \in \{1, \ldots, C\} \tag{11}$$

The center of the c-th cluster is updated as the weighted average of the objects assigned to the cluster: the weights are the membership degrees of the objects to the cluster ($\mu_{c,j}$), raised to the λ-th power. Equation 11 (centralized setting) and Eq. 7 (federated setting) lead to the same result. In fact, for each cluster c,

$$\sum_{m=1}^{M} \mathbf{ws}_c^{(t),m} = \sum_{j=1}^{N} \mu_{c,j}^{\lambda} \mathbf{x}_j^{(t),sum} \tag{12}$$

and

$$\sum_{m=1}^{M} u_c^{(t),m} = \sum_{j=1}^{N} \mu_{c,j}^{\lambda} \tag{13}$$

We have proved that the horizontal federated versions of the algorithms are equivalent, and therefore lossless, compared to the traditional versions applied to the global dataset.

B. Privacy analysis

1) LLF-CM: The analysis of Algorithm 1 highlights that raw data are not exposed while learning the cluster representation, thus protecting the privacy of data owners. Furthermore, since each data owner only transmits the number of objects and the linear sum of data objects assigned to each cluster, the server cannot infer the exact value of the features of any single object. The only exception arises when, for a given m and a given c, we have $n_c^{(t),m} = 1$ (i.e., Γ_c is a "singleton cluster"): in this case the vector $\mathbf{Ls}_c^{(t),m}$ holds the exact coordinates of the only object $\mathbf{x}_j^m$ in cluster c for the owner P^m. This information disclosure can be avoided using a simple trick to prevent local singleton clusters from contributing to the update of the relative centers: whenever $n_c^{(t),m} = 1$, both the values for $\mathbf{Ls}_c^{(t),m}$ and $n_c^{(t),m}$ are forced to 0 before transmitting them (see line 11). It is worth noticing that client P^m still participates in the federation and actually shares information for all the not-singleton clusters. A very unlikely pathological situation might occur when the server receives no information on one cluster, but it can be handled with a server-side random perturbation of the cluster center. In practice, in all our experiments this situation has never shown up. When $n_c^{t,m} > 1$, the information shared with the server does not reveal the raw data since for each feature f it results in an indeterminate equation with infinitely many solutions. In the case $n_c^{t,m} = 2$, for example, the indeterminate equation is in the form $x_{1,f}^m + x_{2,f}^m = Ls_{c,f}^{t,m}$, where $x_{1,f}^m$ and $x_{2,f}^m$ are the values of the generic feature f for the two instance assigned to cluster c on client m.

2) LLF-FCM: The privacy analysis on Horizontal Federated FCM has already been discussed in a previous paper [17]: for the sake of comprehensiveness, it is summarized hereafter as well. The server can infer the value of the features of each single sample under certain conditions. At each round, each data owner discloses the following information: for each cluster c, the sum $u_c^{(t),m}$ of the membership degrees, each raised to the λ-th power, and the array $\mathbf{ws}_c^{(t),m}$ of the weighted sum of the objects, according to their degree of membership to cluster c. Sum $u_c^{(t),m}$ can be represented by the following C equations:

$$\begin{cases} \mu_{1,1}^{\lambda} + \mu_{1,2}^{\lambda} + \cdots + \mu_{1,N_m}^{\lambda} = \alpha_1 \\ \mu_{2,1}^{\lambda} + \mu_{2,2}^{\lambda} + \cdots + \mu_{2,N_m}^{\lambda} = \alpha_2 \\ \cdots \\ \mu_{C,1}^{\lambda} + \mu_{C,2}^{\lambda} + \cdots + \mu_{C,N_m}^{\lambda} = \alpha_C \end{cases} \tag{14}$$

The array $\mathbf{ws}_c^{(t),m}$ corresponds to the following C equations for each feature f:

$$\begin{cases} x_{1,f}\mu_{1,1}^{\lambda} + x_{2,f}\mu_{1,2}^{\lambda} + \cdots + x_{N_m,f}\mu_{1,N_m}^{\lambda} = \beta_{1,f} \\ x_{1,f}\mu_{2,1}^{\lambda} + x_{2,f}\mu_{2,2}^{\lambda} + \cdots + x_{N_m,f}\mu_{2,N_m}^{\lambda} = \beta_{2,f} \\ \cdots \\ x_{1,f}\mu_{C,1}^{\lambda} + x_{2,f}\mu_{C,2}^{\lambda} + \cdots + x_{N_m,f}\mu_{C,N_m}^{\lambda} = \beta_{C,f} \end{cases} \tag{15}$$

The server receives from each data owner m the α_c value for each cluster c, and the $\beta_{c,f}$ value for each cluster c and each feature f. As membership degrees ($\mu_{c,j}$) of objects to clusters are computed according to the definition in Eq. 6, it turns out that the only unknown variables are the coordinates $x_{j,f}^m$. To derive these coordinates for the objects stored in m, the server needs to solve the overall system of equations, composed by Eqs. 14 and 15, replacing the values of $\mu_{c,j}$ by using Eq. 6. The number of unknown variables is $N_m \times F$ and the number of equations is $C + C \times F$. If the server knew the number of objects N_m, it could derive the solution unless the number of unknown variables was greater than the number of equations, that is, if $N_m > \frac{C \times (F+1)}{F}$. We highlight that the server does not know N_m. Further, the condition does not appear too restrictive since it requires that the number of objects is greater than the number of clusters plus C/F. Therefore, we might decide to let the data owner transmit the locally aggregated data (Algorithm 2, line 7) only if $N_m > \frac{C \times (F+1)}{F}$. Although the server may be able to retrieve some data information from the aggregated measurements submitted by each data owner, it cannot determine the exact raw data values.

V. Federated Clustering over Vertically Partitioned Data

Unlike the horizontal partitioning setting, in the *vertical* partitioning setting the M participants have the same number of samples N and mutually disjoint subsets of the features for all samples. Let $\mathbf{x}_j^m$ be the projection of the j-th sample of the dataset onto the F^m features over the m-th participant; the complete value of the j-th sample corresponds to the concatenation of the projections available on all the data owners: $\mathbf{x}_j = \left\{\mathbf{x}_j^1, \mathbf{x}_j^2, \ldots, \mathbf{x}_j^M\right\}$. The dimension of this vector is $F = \sum_{m=1}^{M} F^m$, i.e., the overall number of features.

The federated versions of CM and FCM clustering algorithms are derived from the original versions of the two algorithms by Lloyd [15] and Bezdek [16], respectively.

Similarly to the *horizontal* setting, Fig. 4 shows the sequence diagram of a generic round-based scheme for vertical federated clustering, with the actions of the server and each data owner. Since LLF-CM and LLF-FCM share the same overall scheme, we provide a unique pseudocode for both as reported in Algorithm 4.

The clustering parameters (number of clusters C, and fuzziness factor λ, for LLF-FCM) are initialized on the server. Notably, the heuristic proposed in [32] for deciding the number of clusters can be adapted to the vertical setting as the computation of the SDB index on the server does not violate the privacy of raw data. The number of clusters is shared with each

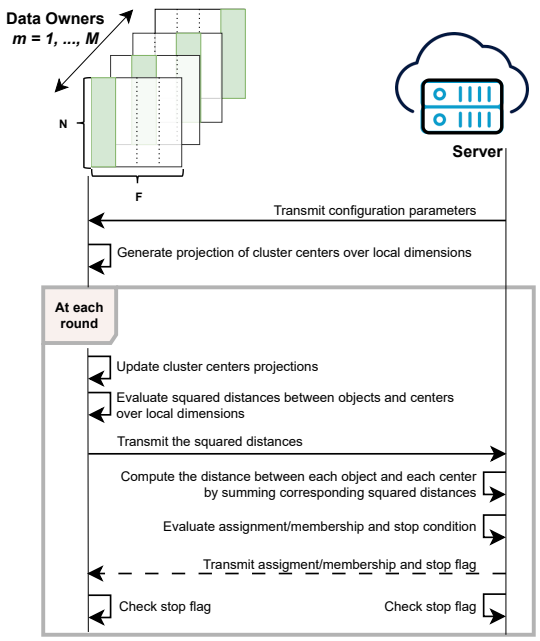


Fig. 4. Sequence diagram summarizing the execution of the federated CM and FCM algorithms over vertically partitioned data.

Algorithm 4 Vertical Federated c-means (LLF-CM) and Fuzzy c-means (LLF-FCM). C: number of clusters, $\varepsilon > 0$: tolerance value for the stop condition. T: maximum number of rounds

Initialization stage

Server:

1: Transmit the number of clusters C to each data owner
2: *stop_flag* = FALSE

Each data owner P^m:

3: Randomly generate the projections of the cluster centers over the features defined on P^m: $\mathbf{V}_c^{(0),m} = \left\{\mathbf{v}_1^{(0),m}, \mathbf{v}_2^{(0),m}, \ldots, \mathbf{v}_C^{(0),m}\right\}$

Execution stage

4: At each round t, with t starting from 0:

Centers update

Each data owner P^m:

5: **if** NOT *stop_flag* AND $t > 0$ **then**
6: Evaluate $\mathbf{v}_c^{(t),m}$ for each cluster Γ_c (as per Eq. 16 in CM, as per Eq. 17 in FCM)
7: **end if**

Cluster assignment

Each data owner P^m:

8: Evaluate $d_{j,c}^{(t),m} = \sum_{f=1}^{F^m} \left(x_{j,f}^m - v_{c,f}^{(t),m}\right)^2$ for each object j and cluster Γ_c
9: Transmit the $N \times C$ matrix $\mathbf{D}^{(t),m}$ to the server

Server:

10: Evaluate $d_{j,c}^{(t)} = \sqrt{\sum_{m=1}^{M} d_{j,c}^{(t),m}}$ for each object j and cluster Γ_c
11: Evaluate assignment/membership for each object j and cluster Γ_c (i.e., $q_j^{(t)}$ and $n_c^{(t)}$ in CM, $\mathbf{U}^{(t)} = \left[\mu_{c,j}^{(t)}\right]$ as per Eq. 18 in FCM)
12: **if** $t \geq 1$ AND $\left(||\mathbf{D}^{(t)} - \mathbf{D}^{(t-1)}||_F < \varepsilon \text{ OR } t > T\right)$ **then**
13: *stop_flag* = TRUE
14: **end if**
15: Transmit object assignment/membership to clusters and *stop_flag* to each data owner

Termination

Each data owner P^m & Server:

16: **if** NOT *stop_flag* **then**
17: Proceed with the next round (line 5)
18: **end if**
19: ⟨Termination⟩

data owner. At initialization time each data owner P^m randomly generates the projections of the cluster centers onto the components for P^m, $\mathbf{V}_c^{(0),m} = \left\{\mathbf{v}_1^{(0),m}, \mathbf{v}_2^{(0),m}, \ldots, \mathbf{v}_C^{(0),m}\right\}$.

The execution stage is subsequently iterated until the occurrence of the stop condition. At each round t (except the first one) each data owner computes the new projections $\mathbf{v}_c^{(t),m}$ of the centers $\mathbf{v}_c^{(t)}$ based on the most recent information obtained from the server. In the case of LLF-CM, $\mathbf{v}_c^{(t),m}$ is evaluated as the mean of the objects assigned to each cluster:

$$\mathbf{v}_c^{(t),m} = \frac{\sum_{\mathbf{x}_j^m \in \Gamma_c} \mathbf{x}_j^m}{n_c^{(t-1)}} \quad \forall\, c \in \{1, \ldots, C\}. \tag{16}$$

In the case of LLF-FCM, $\mathbf{v}_c^{(t),m}$ is evaluated based on the current $\mathbf{U}^{(t)}$ matrix:

$$\mathbf{v}_c^{(t),m} = \frac{\sum_{j=1}^{N} \left(u_{c,j}^{(t-1)}\right)^{\lambda} \mathbf{x}_j^m}{\sum_{j=1}^{N} \left(u_{c,j}^{(t-1)}\right)^{\lambda}} \quad \forall\, c \in \{1, \ldots, C\}. \tag{17}$$

Then, each data owner P^m evaluates the squared distance between each object $\mathbf{x}_j^m$ and each cluster center $\mathbf{v}_c^{(t)}$, i.e., the local contribution $d_{j,c}^{(t),m}$ for the evaluation of the overall distance of $\mathbf{x}_j$ from $\mathbf{v}_c^{(t)}$. The $N \times C$ matrix $\mathbf{D}^{(t),m}$ is sent to the server. The server aggregates all the local contributions to evaluate the overall distance of each object from each cluster center. The execution proceeds by assessing the cluster assignment/membership of each object. In LLF-CM, the server assigns each object in the dataset to one cluster, according to the nearest cluster center principle. Let $\mathbf{q}^{(t)}$ be the vector that, for each object $\mathbf{x}_j$, reports the cluster the object belongs to, and $\mathbf{n}^{(t)}$ be the vector that, for each cluster c, reports the number of objects assigned to the cluster. Conversely, in LLF-FCM, the server can compute the membership degree of each object j to each cluster Γ_c as follows:

$$\mu_{c,j}^{(t)} = \left(\sum_{l=1}^{C} \left(\frac{d_{j,c}^{(t)}}{d_{j,l}^{(t)}}\right)^{\frac{2}{\lambda-1}}\right)^{-1} \quad \forall\, j, c. \tag{18}$$

The termination condition we chose to implement for both algorithms is the logical disjunction of two predicates, the exceeded threshold T for the number of rounds (only for the horizontal version), and the exceeded threshold ε of centroids' displacement in adjacent rounds (evaluated as the Frobenius norm of the difference of the distance matrices of centroids).

The server transmits object membership information (i.e., $\mathbf{q}^{(t)}$ and $\mathbf{n}^{(t)}$ in LLF-CM, $\mathbf{U}^{(t)}$ in LLF-FCM), along with the indication about termination, to the data owners for executing the update of centers in the next round (if any).

A. Equivalence with the traditional algorithms executed on the global dataset

The vertical variants of LLF-CM and LLF-FCM (algorithm 4) produce the same clusters that would be obtained by the traditional CM and FCM, respectively, applied to the global dataset $\mathbf{X}_{sum}$, assuming the same initial cluster centers. Indeed, the update step for cluster centers is exactly equivalent

to the *centralized* case. As per the cluster assignment step, the equivalence derives from the way the squared distance is computed:

$$\sum_{m=1}^{M}\sum_{f=1}^{F^m}\left(x_{j,f}^{m}-v_{c,f}^{(t),m}\right)^2=\sum_{f=1}^{F}\left(x_{j,f}^{sum}-v_{c,f}^{(t)}\right)^2 \qquad (19)$$

The left-hand side of Eq. 19 corresponds to the evaluation as carried out in the vertical federated setting, whereas the right-hand side represents the evaluation when all the object features are directly available at the central server. The equivalence holds only when all M parties provide their updates, i.e. mutually disjoint subsets of the features for all the N samples.

Finally, the stop conditions in Algorithm 4 (line 12) can be applied equivalently in the case of a centralized dataset. We have proved that the vertical federated versions of the algorithms are equivalent, and therefore lossless, compared to the traditional versions applied to the global dataset.

B. Privacy analysis

The analysis of Algorithm 4 highlights that raw data are not exposed while learning the cluster representation, thus protecting the privacy of data owners. We point out that, during the entire execution of both algorithms, no one knows the complete set of coordinates of the cluster centers: such knowledge is split among data owners, since each one holds only the information relative to its local components. The server, in particular, receives the M distance matrices $\mathbf{D}^{(t),m}$ from the data owners but it cannot infer the true coordinates of the objects since it ignores the position of the cluster centers.

VI. Experimental Analysis

It has been shown that the clustering results of the proposed LLF-CM and LLF-FCM, and their traditional centralized versions, are the same (given the same initial centers). Thus, there is no need for an experimental comparison of their outcomes. Instead, it can be worth empirically investigating an aspect peculiar to federated settings: due to different reasons (unreliable communication, client failure, temporal unavailability, deliberate exclusion, etc.) some updates from clients to the server may be missed. So, the effect of restricting the number of clients that provide an update can be evaluated for different fractions γ of the total number of clients.

Our experimental analysis has been mainly focused on horizontal LLF-FCM for different reasons. First, the analysis of results and convergence, depending on the fraction γ of involved clients, is relevant only in the *horizontal* setting: in a vertical partition, as each client holds a subset of features, the absence of a single client's contribution leads to a clustering outcome necessarily different from what the original algorithm obtains on the entire dataset. Second, we do not aim to compare the performance of CM and FCM, and both our LLF-CM and LLF-FCM share the same federation scheme (see Fig. 3): due to space limitations, we chose to focus only on LLF-FCM, because of the possibility of comparing our approach with the recently proposed horizontal versions GDF-FCM [31] and C2F-FCM [32]. For the sake of completeness, we also report one experiment to show the equivalence of the vertical LLF-FCM to the centralized one as the overall number of clients varies. To the best of our knowledge, none of the existing FL libraries support federated clustering. Thus, we implemented the federated versions of CM and FCM from scratch in Python.

A. Horizontal Federated FCM: Setup

We consider eight real-world and synthetic datasets with available ground-truth labels, namely *xclara*, *s-set1* and *s-set2* from the Clustering Benchmark repository[2], *waveform v1*, *pendigits*, *rice*, and *statlog* from the UCI Machine Learning repository[3] and *phoneme* from the Keel repository[4] [40]. The properties of each dataset are summarized in Table I.

TABLE I
DATASETS USED IN EXPERIMENTS

Dataset	DOI	Classes	Samples	Features
xclara (XC)	10.18637/jss.v001.i04	3	3000	2
s-set1 (S1)	10.1016/j.patcog.2005.09.012	15	5000	2
s-set2 (S2)	10.1016/j.patcog.2005.09.012	15	5000	2
pendigits (PE)	10.24432/C5MG6K	10	10992	16
waveform-v1 (WA)	10.24432/C5CS3C	3	5000	21
phoneme (PH)	10.1609/aaai.v33i01.33015117	2	5404	5
rice (RI)	10.24432/C5MW4Z	2	3810	7
statlog (ST)	10.24432/C55887	6	6435	16

The number of clusters C was set to match the number of classes of each dataset; Furthermore, we chose the fuzziness factor $\lambda = 2$. The convergence of the algorithms is an important aspect to investigate: thus, we set the maximum number of rounds $T = 30$ as the unique stopping condition, to uncover the algorithms' behaviour through successive rounds.

We considered a federated scenario with $M = 20$ clients and data randomly and evenly distributed among the clients, with roughly the same amount and distribution of instances per client (i.i.d. setting). Moreover, to simulate scenarios with partial contributions by clients, at the beginning of each round, we randomly select only a fraction γ of the clients to contribute to the global model update. In our experiments, we varied γ in the set $\{0.25, 0.5, 0.75, 1\}$, with $\gamma = 1$ indicating that all the clients are selected.

B. Horizontal Federated FCM: Evaluation strategy

We evaluated our horizontal LLF-FCM from multiple points of view. First, we assessed the consistency of the clustering results of the *federated* setting with those of the *centralized* setting, i.e., with the original FCM executed on the global dataset, for different values of γ. This is accomplished (i) by evaluating the distance between $\mathbf{V}_{fed}$ and $\mathbf{V}_{sum}$, i.e., the cluster centers obtained by the federated and the centralized approaches, respectively, and (ii) by adopting a metric to compare the cluster assignment of objects between the two approaches. Specifically, let $\mathbf{y}_{fed} = [y_{fed,1}, y_{fed,2}, \ldots, y_{fed,N}]$, $y_{fed,j} \in \{1, \ldots, C\}$, and $\mathbf{y}_{sum} =$

[2]https://github.com/deric/clustering-benchmark
[3]https://archive.ics.uci.edu/
[4]https://sci2s.ugr.es/keel/datasets.php

$[y_{sum,1}, y_{sum,2}, \ldots, y_{sum,N}]$, $y_{sum,j} \in \{1, \ldots, C\}$, be the vectors of such cluster assignment for each object, after defuzzification. In the federated setting, such a vector is obtained, purely for evaluation purposes, by concatenating the results obtained locally by each data owner, including those that did not contribute to the federated procedure but could still benefit from the aggregated model. For the traditional centralized FCM, we choose the implementation in the scikit-fuzzy library [5]. The consistency between partitions has been evaluated through the adjusted rand index (ARI), a normalized version of the Rand Index, corrected by the expected agreements obtained by chance. ARI is a widely accepted clustering metric: $ARI = 1$ and $ARI = 0$ indicate respectively a perfect and random agreement between the two clusterings.

Second, we applied ARI as an extrinsic evaluation metric to compare $\mathbf{y}_{fed}$ with the ground truth $\mathbf{y}_{true}$. Furthermore, we adopted two popular intrinsic evaluation metrics, i.e., Xie Beni (XB) [41] and Soft Davies Bouldin (SDB) [39], which has been recently proposed as a fuzzy adaptation of the well-known Davies Bouldin separation measure [42].

Third, we compared our approach with the state-of-art approaches for Federated FCM [31], [32] discussed in Section II, in two distinct experiments. In the first setting, we compared our LLF-FCM with GDF-FCM [31] starting from random prototype initialization, as reported in the original proposal [31]. In the second setting we extended our LLF-FCM to support careful seeding through a variant of k-means++, for the sake of a fair comparison with C2F-FCM [32]. We used the same configuration parameters for our algorithm and the others: namely, the stopping condition ($T = 30$), the number of clusters (matching the number of classes in each dataset), and the fuzziness factor ($\lambda = 2$). In the implementation of GDF-FCM, we followed the author's suggestion for the number of local iterations ($local\_iter = 10$); anyway, we noticed that the default value of the learning rate $\alpha = 0.05$ does not secure convergence for six of the eight datasets. Thus, for a fair comparison, we tuned the learning rate to a reasonably lower value ($\alpha = 0.005$), ensuring convergence for all datasets. The default parameter configuration has been used for C2F-FCM [32].

C. Horizontal Federated FCM: Results

The analysis of the convergence with respect to the number of rounds for our horizontal LLF-FCM and the state-of-art GDF-FCM [31] are reported in the plots of Table II for each dataset. We performed 10 runs of the algorithms by varying the seed for the center initialization and data owners sampling. For each execution, we adopted the same center initialization for both the algorithms to compare. More specifically, we plot the *tolerance index*, i.e. the Frobenius norm of the difference of the cluster centers $\mathbf{V}_{fed}$ between two consecutive rounds, averaged over the 10 executions.

In general, both approaches converge within the first 10 rounds. Interestingly, the trend of the tolerance index with respect to the number of rounds is not particularly affected by the fraction of clients involved in the federation. However, the index value converges to higher values for smaller γ values. This is sensible because, when few participants are randomly sampled at each round, the probability of selecting the same participants in consecutive rounds is low and consequently the centers are constantly being tweaked. The values of standard deviation (shaded regions) of GDF-FCM are in general lower compared to those of LLF-FCM. In the former, in fact, the evolution of centers depends on the gradient of the cost function and is modulated by the learning rate: a lower learning rate generally leads to smaller updates of the centers, leading to lower values of standard deviations.

Table III shows the numerical results of our experiments. The measure of agreement with the ground-truth partition, i.e., $ARI(\mathbf{y}_{fed}(\gamma), \mathbf{y}_{true})$, gives us a general insight of the soundness of the clustering algorithm. It is worth recalling that we are not concerned with demonstrating the suitability of the FCM algorithm itself, but rather in validating the federated approach; nonetheless, it can be observed that ARI values of our LLF-FCM are always positive indicating that it consistently outperforms random partitioning for each dataset. Interestingly, LLF-FCM and GDF-FCM achieve very similar figures of clustering quality. Furthermore, the ARI computed with respect to the ground truth is not particularly affected by the parameter γ: a reasonable partition is found even with a low fraction of contributing participants.

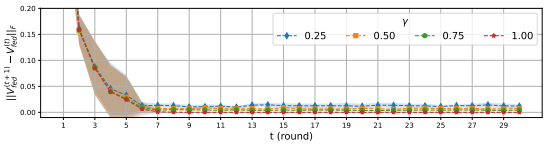
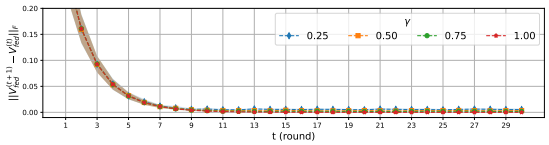
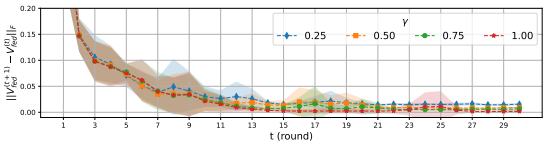
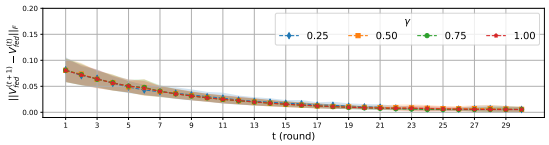
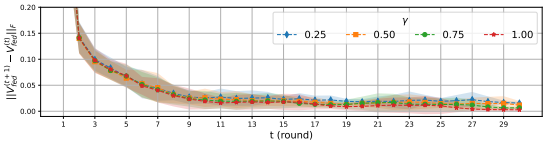
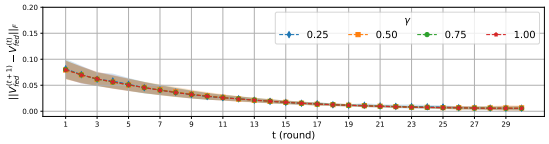
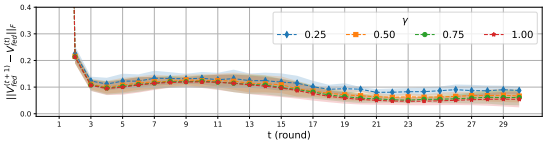
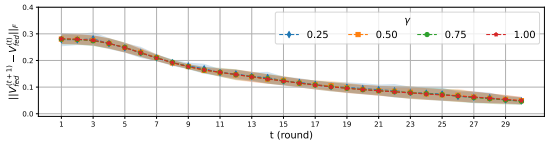
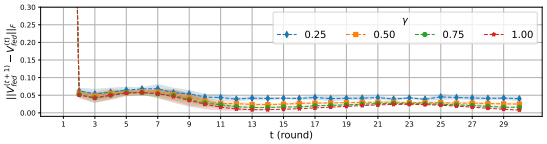
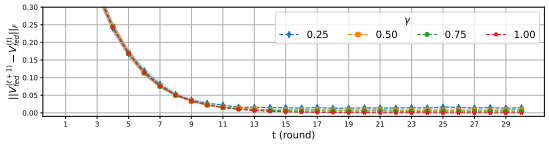
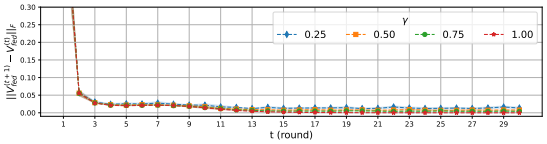
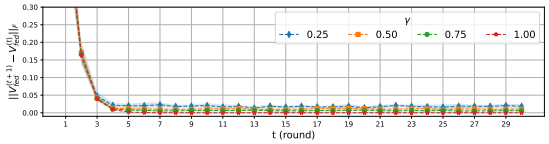
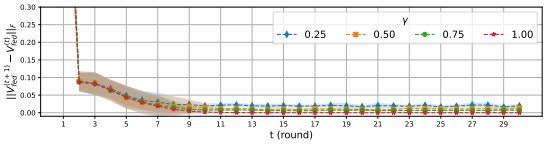
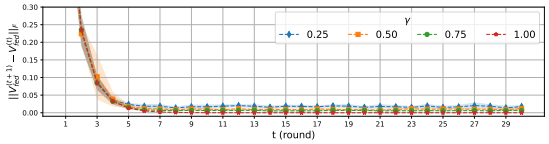
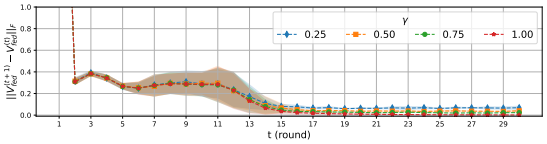
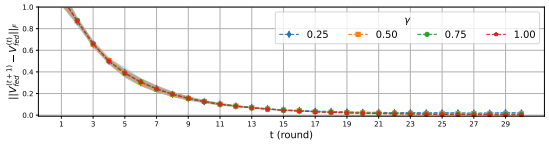
The two horizontal federated FCM algorithms achieve very similar results also in terms of intrinsic evaluation metrics. The discrepancy between the two is significant only for the *pendigits* dataset, where GDF-FCM achieves lower values of XB and SDB compared to LLF-FCM. In these cases the optimization strategy proposed in [31] is able to find clusters that are more compact and separated from each other than those discovered in our approach, even in the case of $\gamma = 1$ for which LLF-FCM coincides with the traditional FCM.

The values of $ARI(\mathbf{y}_{fed}(\gamma), \mathbf{y}_{sum})$ indicate the agreement between the results of the federated and the traditional centralized FCM. Such an agreement is always very high with $ARI > 0.9$ for all the datasets for both the approaches except for GDF-FCM on the *pendigits* dataset ($ARI = 0.79$), in which the algorithm evidently converges to a configuration slightly different from the centralized one. The reason for such disagreement arguably lies in the optimization strategy of C2F-FCM: in fact, it exploits the traditional FCM optimization at the client level and the gradient-descent optimization at the server level (based on the weighted average of locally computed gradients). These two optimizations may work in opposite directions, especially in the case of non-i.i.d. datasets, as we will highlight at the end of this subsection.

Also in the case of the agreement between the federated and the centralized FCM, the ARI is quite robust to γ variations, showing a slight increasing tendency when the fraction of participants increases. For $\gamma = 1$, when all clients are involved, our LLF-FCM always obtains $ARI(\mathbf{y}_{fed}, \mathbf{y}_{sum}) = 1$, as experimental evidence of the equivalence that we have also demonstrated theoretically in Section IV-A. Although even GDF-FCM can achieve ARI values close to 1 (and actually equal to 1 for *XC*, *PH* and *RI*), an exact match is not guaranteed and the approach cannot be defined lossless.

[5]https://pythonhosted.org/scikit-fuzzy/

TABLE II
HORIZONTAL FEDERATED FCM: AVERAGE VALUES OF TOLERANCE INDEXES. SHADED REGIONS REPRESENT THE STANDARD DEVIATIONS.

Dataset	LLF-FCM (ours)	GDF-FCM [31]
XC		
S1		
S2		
PE		
WA		
PH		
RI		
ST		

The consistency between federated and centralized approaches is also analyzed through the displacements of the centers found in the two cases. The average displacements are reported in the left part of Table III and plotted in Figure 5. Notably, we consider the permutation of the centers in $\mathbf{V}_{fed}(\gamma)$ that minimizes the norm $||\mathbf{V}_{fed}(\gamma) - \mathbf{V}_{sum}||_F$. Coherently with the increasing trend of ARI, the norm decreases as γ grows, ultimately reaching the value 0 in our approach for $\gamma = 1$. It is worth underlining that, although the federated centers deviate from those in the centralized case, this has little impact on the cluster assignment of objects, as indicated by the rather small variations in $ARI(\mathbf{y}_{fed}(\gamma), \mathbf{y}_{true})$ values. The displacement degree is comparable for the two approaches, except for *pendigits*, where it is significantly higher for GDF-FCM. We also verified that this does not depend on the maximum number of rounds: since the convergence on *pendigits* is relatively slower than in the other datasets (see relevant figure in Table II), we increased the number of rounds from 30 to 50, noticing that the final displacement of centers becomes even more pronounced. As stated above, this suggests that the algorithm converges to a configuration slightly different from the one in the centralized scenario.

In general, the results show that both proposals yield reliable and appropriate results for the clustering problem in the i.i.d. horizontal federated settings, albeit pursuing different approaches. Our LLF-FCM, however, guarantees to be lossless when all clients collaborate, whereas GDF-FCM slightly deviates from the result of the original FCM. This is motivated by the adoption of the gradient descent as the server-side optimization strategy and by the related parameterization: it

TABLE III
CLUSTERING RESULTS OF OUR LLF-FCM (TOP) AND GDF-FCM [31] (BOTTOM) WITH RESPECT TO INCREASING PERCENTAGES OF PARTICIPANTS TO THE FEDERATION. AVERAGE VALUES OF: I) THE DIFFERENCE OF THE PROTOTYPES BETWEEN *federated* AND *centralized* VERSIONS, II) ARI BETWEEN *federated* AND *centralized* VERSIONS, III) ARI BETWEEN *federated* AND *ground truth* PARTITIONS, IV) XB INDEX AND V) SDB INDEX. THE BEST VALUE BETWEEN THE TWO APPROACHES IS HIGHLIGHTED IN BOLD.

		$\|\|\mathbf{V}_{fed}(\gamma) - \mathbf{V}_{sum}\|\|_F$				$ARI(\mathbf{y}_{fed}(\gamma), \mathbf{y}_{sum})$				$ARI(\mathbf{y}_{fed}(\gamma), \mathbf{y}_{true})$				XB				SDB			
		0.25	0.50	0.75	1.00	0.25	0.50	0.75	1.00	0.25	0.50	0.75	1.00	0.25	0.50	0.75	1.00	0.25	0.50	0.75	1.00
LLF-FCM (ours)	XC	**0.01**	0.01	**0.00**	**0.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.99**	**0.99**	**0.99**	**0.99**	**0.05**	**0.05**	**0.05**	**0.05**	**0.26**	**0.26**	**0.26**	**0.26**
	S1	**0.14**	**0.10**	**0.01**	**0.00**	**0.96**	**0.98**	**1.00**	**1.00**	**0.91**	**0.90**	**0.90**	**0.90**	**0.73**	**0.83**	**0.89**	**0.94**	**0.74**	**0.76**	**0.78**	**0.79**
	S2	**0.06**	**0.05**	**0.04**	**0.00**	**0.98**	**0.98**	**0.98**	**1.00**	**0.91**	**0.91**	**0.91**	**0.90**	0.42	**0.31**	**0.36**	**0.52**	**0.60**	**0.58**	**0.58**	**0.61**
	PE	**0.12**	**0.07**	**0.05**	**0.00**	**0.96**	**0.98**	**0.98**	**1.00**	**0.48**	**0.48**	**0.48**	**0.48**	11.26	9.95	8.93	8.79	1.48	1.41	1.39	1.37
	WA	0.04	**0.02**	**0.01**	**0.00**	0.97	**0.98**	**0.99**	**1.00**	**0.24**	**0.24**	**0.24**	**0.24**	0.92	0.86	**0.82**	**0.79**	0.54	0.53	0.53	0.53
	PH	**0.01**	**0.01**	**0.00**	**0.00**	**0.98**	**0.99**	**0.99**	**1.00**	**0.15**	**0.15**	**0.15**	**0.15**	0.97	**0.97**	0.97	0.97	**0.21**	**0.21**	**0.21**	**0.21**
	RI	**0.02**	**0.01**	**0.01**	**0.00**	**0.99**	**0.99**	**0.99**	**1.00**	**0.68**	**0.68**	**0.68**	**0.68**	**0.22**	**0.22**	**0.22**	**0.22**	**0.25**	**0.25**	**0.25**	**0.25**
	ST	**0.06**	**0.03**	**0.02**	**0.00**	**0.98**	**0.99**	**0.99**	**1.00**	**0.53**	**0.53**	**0.53**	**0.53**	**0.30**	**0.31**	**0.31**	**0.31**	**0.78**	**0.78**	**0.78**	**0.78**
GDF-FCM [31]	XC	**0.01**	**0.00**	**0.00**	**0.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.99**	**0.99**	**0.99**	**0.99**	**0.05**	**0.05**	**0.05**	**0.05**	**0.26**	**0.26**	**0.26**	**0.26**
	S1	0.16	0.16	0.18	0.18	**0.96**	0.96	0.95	0.95	0.90	0.89	0.89	0.89	0.95	1.01	1.02	**0.94**	0.83	0.85	0.86	0.84
	S2	0.19	0.19	0.20	0.20	0.91	0.91	0.91	0.91	0.86	0.85	0.85	0.85	**0.36**	0.38	0.39	0.40	0.63	0.64	0.64	0.64
	PE	0.82	0.80	0.81	0.79	0.79	0.79	0.79	0.79	**0.48**	**0.48**	0.45	**0.48**	**4.25**	**3.99**	**3.95**	**3.87**	**0.88**	**0.87**	**0.87**	**0.87**
	WA	**0.03**	**0.03**	**0.03**	0.03	**0.98**	**0.98**	0.98	0.98	**0.24**	**0.24**	**0.24**	**0.24**	**0.83**	**0.82**	**0.82**	0.82	**0.52**	**0.52**	**0.52**	**0.52**
	PH	**0.01**	**0.01**	**0.00**	**0.00**	0.97	0.98	**0.99**	**1.00**	**0.15**	**0.15**	**0.15**	**0.15**	**0.94**	**0.97**	**0.96**	**0.96**	**0.21**	**0.21**	**0.21**	**0.21**
	RI	**0.02**	**0.01**	**0.01**	**0.00**	**0.99**	**0.99**	**0.99**	**1.00**	**0.68**	**0.68**	**0.68**	**0.68**	**0.22**	**0.22**	**0.22**	**0.22**	**0.25**	**0.25**	**0.25**	**0.25**
	ST	0.03	0.02	0.02	0.02	**0.98**	**0.99**	**0.99**	0.99	**0.53**	**0.53**	**0.53**	**0.53**	0.31	**0.31**	0.32	**0.31**	**0.78**	**0.78**	**0.78**	**0.78**

is clear that the convergence differs from the original one depending on the additional parameters α, i.e., the learning rate, and number of local iterations.

The gradient-descent strategy of GDF-FCM, however, is not effective in the non-i.i.d. setting, due to the interplay between the server-level and the client-level optimizations. To analyse this aspect, we also considered the case of label imbalance, in which data are partitioned following a probability distribution whereby the classes are represented differently in different clients. Following the relevant literature [43], we exploited the Dirichlet distribution to allocate a proportion of the samples of each class k, i.e., $p_k \sim Dir(\beta)$: the smaller the concentration parameter β, the more unbalanced the distribution. In our experiments, we set $\beta = 0.5$ as in [43]. In Table IV, we report the values of $ARI(\mathbf{y}_{fed}(\gamma), \mathbf{y}_{true})$ as a measure of agreement with the ground-truth partition for $\gamma = 1$.

TABLE IV
ARI BETWEEN *federated* AND *ground truth* PARTITIONS WITH $\gamma = 1$ IN THE NON-I.I.D. SETTING.

	XC	S1	S2	PE	WA	PH	RI	ST
LLF-FCM (ours)	0.99	**0.90**	**0.90**	**0.48**	0.24	**0.15**	**0.68**	**0.53**
GDF-FCM [31]	**1.00**	0.83	0.74	0.41	**0.27**	0.13	0.32	0.40

Results suggest that the clustering results obtained with GDF-FCM are significantly impacted by heterogeneity in data distribution. For all datasets except *xclara* and *waveform* the values of ARI are lower than those reported in Table III and than those obtained with our LLF-FCM. Notably, our approach achieves the exact same results as the i.i.d. case: the equivalence with the traditional algorithm does not depend on the distribution of data across clients.

Finally, from a communication point of view, our LLF-FCM is slightly more costly compared to GDF-FCM [31] which requires that, at each round, the clients upload the gradient matrix ($C \times F$) to the server and download the centroid matrix ($C \times F$) from the server. In LLF-FCM the download cost is the same, but each client needs to transmit to the server both $\mathbf{WS}^m$ ($C \times F$), i.e. the sum of the data objects, weighted by the membership degree of the objects to cluster c, and $\mathbf{u}^m$, i.e. the array with C elements, where the c-th element is the sum of the membership degrees of the objects to cluster c.

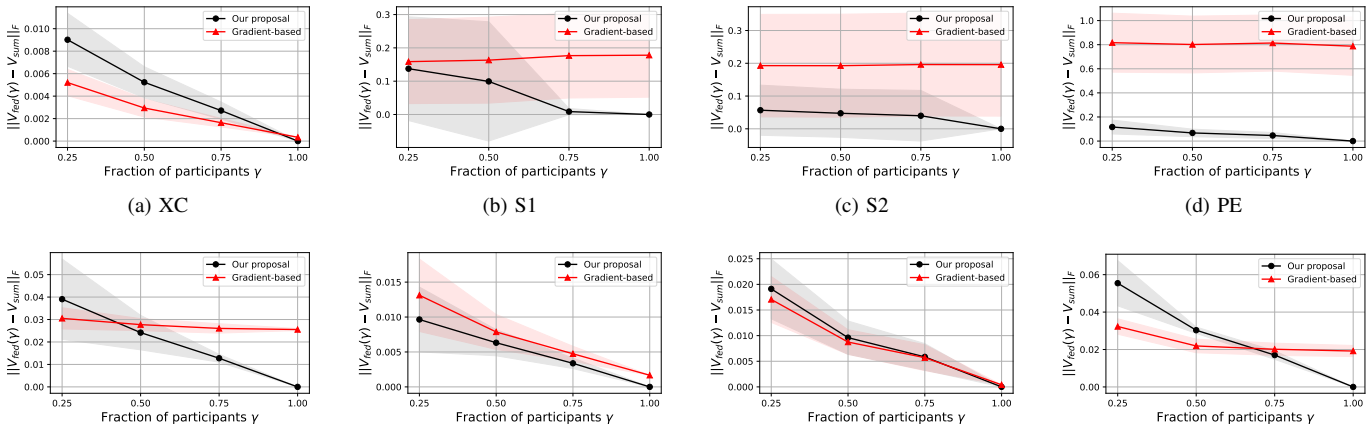
Further data and charts relative to the experimental results are available as Supplementary Material.

D. Horizontal Federated FCM with Careful Seeding

In the previous experimental section, we compared our LLF-FCM with GDF-FCM with a random selection of the initial prototypes [31]. The CM and FCM algorithms, however, are known to be sensitive to initialization, with the risk of getting stuck in local sub-optima. As described in Section II, C2F-FCM [32] leverages a careful seeding policy based on k-means++, as ordinarily done in many popular library implementations of k-means[6]: for the sake of a fair comparison, we introduced a careful seeding also in our LLF-FCM. In particular, at the beginning, each client generates a set of candidate centers and shares them with the server. The server executes the CM procedure on the overall candidate centers, getting to the C centers for initializing the FCM procedure (hence executed as described in Section IV). The candidate centers evaluation procedure on each client is based on k-means++ [33], suitably adapted to avoid the transmission of raw data: whenever a point of the dataset is selected as candidate center, the average of its 5 nearest neighbours is considered rather than the point itself. Clearly, this procedure generally leads to initial centers different from those obtained by applying the k-means++ algorithm in the centralized setting.

Table V reports the results. The ARI values (w.r.t. ground truth partition) obtained by our LLF-FCM after careful seeding are comparable to, or higher than, those reported in Table III (major gain is observed for *s-set1* and *pendigits*). Our LLF-FCM and C2F-FCM [32] achieve quite similar results, in terms of agreement with the ground truth partition and

[6]e.g. in Scikit-learn https://scikit-learn.org/ and Matlab https://mathworks.com/ in "Statistics and Machine Learning Toolbox"



(a) XC (b) S1 (c) S2 (d) PE

(e) WA (f) PH (g) RI (h) ST

Fig. 5. Average distance between cluster centers computed by the Horizontal Federated FCM and the traditional FCM on the global merged dataset $\mathbf{X}_{sum}$. Our LLF-FCM (black) is compared with GDF-FCM [31] (red). Best viewed in color.

TABLE V

CLUSTERING RESULTS OF OUR LLF-FCM (TOP) AND C2F-FCM [32] (BOTTOM) WITH RESPECT TO INCREASING PERCENTAGES OF PARTICIPANTS TO THE FEDERATION, BOTH EMPLOYING CAREFUL SEEDING BASED ON k-MEANS++. AVERAGE VALUES OF: I) THE DIFFERENCE OF THE PROTOTYPES BETWEEN *federated* AND *centralized* VERSIONS, II) ARI BETWEEN *federated* AND *centralized* VERSIONS, III) ARI BETWEEN *federated* AND *ground truth* PARTITIONS, IV) XB INDEX AND V) SDB INDEX. THE BEST VALUE BETWEEN THE TWO APPROACHES IS HIGHLIGHTED IN BOLD.

		$\|\|\mathbf{V}_{fed}(\gamma)-\mathbf{V}_{sum}\|\|_F$				$ARI(\mathbf{y}_{fed}(\gamma),\mathbf{y}_{sum})$				$ARI(\mathbf{y}_{fed}(\gamma),\mathbf{y}_{true})$				XB				SDB			
		0.25	0.50	0.75	1.00	0.25	0.50	0.75	1.00	0.25	0.50	0.75	1.00	0.25	0.50	0.75	1.00	0.25	0.50	0.75	1.00
LLF-FCM (ours)	XC	**0.01**	**0.01**	**0.00**	**0.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.99**	**0.99**	**0.99**	**0.99**	**0.05**	**0.05**	**0.05**	**0.05**	**0.26**	**0.26**	**0.26**	**0.26**
	S1	**0.05**	**0.08**	**0.04**	**0.04**	**0.99**	**0.98**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.04**	**0.06**	**0.04**	**0.04**	**0.54**	**0.55**	**0.54**	**0.54**
	S2	**0.10**	**0.10**	**0.05**	**0.09**	**0.98**	**0.98**	**0.99**	**0.98**	**0.95**	**0.95**	**0.96**	**0.95**	**0.12**	0.14	**0.07**	**0.13**	**0.51**	**0.52**	**0.49**	**0.51**
	PE	**0.37**	**0.39**	**0.40**	**0.39**	**0.91**	**0.90**	**0.90**	**0.91**	**0.56**	**0.56**	**0.56**	**0.57**	2.95	2.04	1.68	1.43	0.80	0.76	0.73	0.72
	WA	**0.03**	**0.02**	**0.01**	**0.00**	**0.99**	**0.99**	**1.00**	**1.00**	**0.24**	**0.24**	**0.24**	**0.24**	**0.83**	0.85	0.85	0.85	**0.53**	**0.53**	**0.53**	**0.53**
	PH	**0.01**	**0.01**	**0.00**	**0.00**	**0.98**	**0.99**	**0.99**	**1.00**	**0.15**	**0.15**	**0.15**	**0.15**	**1.00**	**0.97**	**0.97**	**0.97**	0.22	**0.21**	**0.21**	**0.21**
	RI	**0.02**	**0.01**	**0.01**	**0.00**	**0.99**	**0.99**	**1.00**	**1.00**	**0.68**	**0.68**	**0.68**	**0.68**	**0.21**	0.22	**0.22**	**0.22**	**0.25**	**0.25**	**0.25**	**0.25**
	ST	**0.06**	**0.04**	**0.02**	**0.01**	**0.98**	**0.98**	**0.99**	**0.99**	0.53	**0.53**	**0.53**	**0.53**	**0.31**	**0.32**	**0.31**	**0.31**	**0.78**	**0.78**	**0.78**	**0.78**
C2F-FCM [32]	XC	**0.01**	**0.01**	**0.00**	**0.00**	**1.00**	**1.00**	**1.00**	**1.00**	**0.99**	**0.99**	**0.99**	**0.99**	**0.05**	**0.05**	**0.05**	**0.05**	**0.26**	**0.26**	**0.26**	**0.26**
	S1	0.17	0.25	0.17	0.13	0.97	0.95	0.96	0.97	0.97	0.94	0.96	0.97	0.16	0.31	0.32	0.27	0.58	0.61	0.60	0.59
	S2	0.19	0.24	0.23	0.19	0.95	0.94	0.94	0.94	0.92	0.92	0.92	0.92	0.13	**0.13**	0.16	0.14	0.53	0.53	0.53	0.53
	PE	1.41	1.04	0.73	0.72	0.84	0.86	0.88	0.89	0.54	0.54	0.55	0.55	**1.51**	**1.22**	**1.55**	**1.26**	**0.78**	**0.72**	**0.70**	**0.69**
	WA	**0.03**	**0.02**	**0.01**	0.01	0.98	**0.99**	0.99	0.99	**0.24**	**0.24**	**0.24**	**0.24**	0.85	**0.84**	**0.84**	**0.83**	**0.53**	**0.53**	**0.53**	**0.53**
	PH	0.02	**0.01**	0.01	**0.00**	0.96	0.97	0.98	0.99	**0.15**	**0.15**	**0.15**	**0.15**	**1.00**	1.00	0.98	**0.97**	**0.21**	**0.21**	**0.21**	**0.21**
	RI	**0.02**	**0.01**	**0.01**	**0.00**	0.98	**0.99**	0.99	**1.00**	**0.68**	**0.68**	**0.68**	**0.68**	0.22	**0.21**	**0.22**	**0.22**	**0.25**	**0.25**	**0.25**	**0.25**
	ST	0.10	0.06	0.04	0.02	0.96	0.97	0.98	**0.99**	**0.54**	**0.53**	**0.53**	**0.53**	**0.31**	**0.32**	0.33	0.33	**0.78**	0.79	0.79	0.79

in terms of the internal metrics XB and SDB. Furthermore, as already noted for the random initialization, ARI values do not vary particularly with the number of participating clients. Regarding the ARIs between corresponding partitions and the displacements between centres obtained by the federated and centralized approaches, respectively, our LLF-FCM leads to a clustering that is systematically more consistent with the centralized one. It should be noted, however, that the federated initialization procedure is not equivalent to the adoption of k-means++ in the centralized setting, and therefore the equivalence of the resulting partitions may be lost. For half of the datasets it holds $ARI(\mathbf{y}_{fed},\mathbf{y}_{sum}) \neq 1$, even when all clients are involved ($\gamma = 1$). Notably, C2F-FCM introduces a higher computational complexity than LLF-FCM, as it features two optimization procedures at each round (FCM, locally, and CM at the server side). At the same time, however, the cost in terms of communication is slightly reduced because only cluster centers are shared from each client to the server and vice versa.

E. Vertical Federated FCM

Replicating the analysis carried out for the *horizontal* case in the *vertical* setting would be pointless: indeed, varying the fraction of clients would entail changing the feature space where the samples are defined, and thus the convergence analysis of the federated algorithm becomes not very meaningful. However, to experimentally validate the theoretical evidence of equivalence with the centralized setting, we report the results obtained with our Vertical LLF-FCM on the *pendigits dataset* as the overall number of clients varies. Such a dataset has been selected due to the relatively high dimensionality (16 features), which enables different vertical data partitioning schemes: 16 clients each with 1 feature, 8 clients each with 2 features, 4 clients each with 4 features, and 2 clients each with 8 features. We also considered an uneven split, with 5 clients having 1, 2, 3, 4, and 6 features, respectively. Figure 6 shows the convergence analysis, measured as the Frobenius norm of the difference in consecutive rounds of the distance matrices $\mathbf{D}^{(t+1)}$ and $\mathbf{D}^{(t)}$, in which the element $d_{j,c}$ represents the

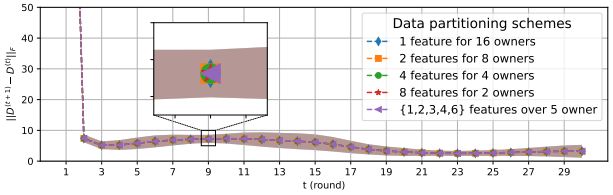


Fig. 6. Vertical LLF-FCM: Average values of Frobenius norm of the difference in the distance matrix between consecutive rounds for different data partitioning schemes on the *pendigits dataset*. Shaded regions represent the standard deviations.

distance between object j and center c.

Unsurprisingly, all curves show exactly the same pattern, as can be clearly seen in the zoomed-in region: when all the clients contribute to the FL process, our vertical LLF-FCM is lossless compared to the traditional centralized FCM. Furthermore, we verified that all the data partitioning schemes achieve the same final clustering result as the centralized case, with $ARI(\mathbf{y}_{fed}(\gamma), \mathbf{y}_{true}) = 0.48$ equal to that obtained by Horizontal LLF-FCM with $\gamma = 1$ for the same dataset.

VII. Conclusion

In this paper, we have proposed federated versions of the c-means (CM) and fuzzy CM (FCM) clustering algorithms over horizontally and vertically partitioned datasets. For both algorithms, the optimization procedure is based on alternately updating the assignment of objects to clusters and cluster centers. We have shown that these steps can still be carried out in the federated setting, where multiple data owners wish to collaboratively evaluate a global clustering model without disclosing their private raw data, but sharing only locally aggregated statistics with an orchestrating central server. We have proved that our federated versions achieve exactly the same results as the traditional clustering algorithms when applied to the union of the local datasets. Our versions are applicable in scenarios with relaxed privacy requirements: we thoroughly analyzed privacy issues in the case of an honest-but-curious central server, which, although adhering to the agreed federated protocol, can try to retrieve private raw data from the updates communicated by the data owners.

The effectiveness of the proposed federated versions is presented through an extensive experimental analysis based on FCM: besides experimentally verifying equivalence with the centralized algorithm for both data partitioning schemes, we deepened the analysis of horizontal federated FCM convergence as the fraction of clients participating in the federation varies. Results are evaluated on several synthetic and real-world datasets and compared with those obtained by two recently proposed horizontal federated FCM approaches. Experimental analysis yields several insights: first, our approach converges in a few FL rounds. Second, with random initialization, our federated FCM and the gradient-descent one achieve very similar results in terms of ARI, even if only ours proves to be lossless and robust to the non-i.i.d. setting. This is accomplished at a slightly additional communication cost, in comparison with the gradient-descent approach. With careful seeding, although the exact equivalence with the centralized setting is lost, clustering accuracy is improved. Third, our horizontal federated FCM is shown to be robust concerning the fraction of clients participating in the federation in the case of independent and identically distributed (i.i.d.) datasets. This is testified by the high agreement between the partition obtained with federated clustering and the *centralized* one ($ARI > 0.9$) and by the small displacements between corresponding cluster centers. Despite the wide scope of the presented study on federated clustering, it is possible to identify two interesting further developments, which we intend to address in future works. First, we would like to study federated versions of other popular clustering algorithms (for instance, the DBSCAN algorithm). Second, the study presented in this paper can be extended by relaxing the assumption that the datasets are *statically* available over the several parties, but rather produced as *data streams*: indeed, ad-hoc approaches should be devised for collaboratively updating the federated clustering based on newly collected data.

Acknowledgments

This work has been partly funded by the PON 2014-2021 "Research and Innovation", DM MUR 1062/2021, Project title: "Progettazione e sperimentazione di algoritmi di federated learning per data stream mining", by the PNRR - M4C2 - Investimento 1.3, Partenariato Esteso PE00000013 - "FAIR - Future Artificial Intelligence Research" - Spoke 1 "Human-centered AI" and the PNRR "Tuscany Health Ecosystem" (THE) (Ecosistemi dell'Innovazione) - Spoke 6 - Precision Medicine & Personalized Healthcare (CUP I53C22000780001) under the NextGeneration EU programme, and by the Italian Ministry of University and Research (MUR) in the framework of the FoReLab and CrossLab projects (Departments of Excellence).

References

[1] Q. Yang, Y. Liu *et al.*, "Federated machine learning: Concept and applications," *ACM T Intel Syst Tec*, vol. 10, no. 2, pp. 1–19, 2019.

[2] C. Zhang, Y. Xie *et al.*, "A survey on federated learning," *Knowl-Based Syst*, vol. 216, p. 106775, 2021.

[3] B. McMahan, E. Moore *et al.*, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[4] J. Konečnỳ, H. B. McMahan *et al.*, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[5] H. Zhu, J. Xu *et al.*, "Federated learning on non-IID data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, 2021.

[6] R. Gosselin, L. Vieu *et al.*, "Privacy and security in federated learning: A survey," *Applied Sciences*, vol. 12, no. 19, 2022.

[7] M. Morafah, W. Wang *et al.*, "A practical recipe for federated learning under statistical heterogeneity experimental design," *IEEE T Artificial Intelligence*, pp. 1–14, 2023.

[8] S. Vahidian, M. Morafah *et al.*, "Rethinking data heterogeneity in federated learning: Introducing a new notion and standard benchmarks," *IEEE T Artificial Intelligence*, pp. 1–13, 2023.

[9] P. Kairouz, H. B. McMahan *et al.*, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[10] S. Khanmohammadi, N. Adibeig *et al.*, "An improved overlapping k-means clustering method for medical applications," *Expert Syst. Appl.*, vol. 67, pp. 12–18, 2017.

[11] L. Lu, Y. Lin *et al.*, "Federated clustering for recognizing driving styles from private trajectories," *Eng Appl Artif Intel*, vol. 118, p. 105714, 2023.

[12] J. Han, M. Kamber *et al.*, "3 - data preprocessing," in *Data Mining*, 3rd ed., ser. The Morgan Kaufmann Series in Data Management Systems. Boston: Morgan Kaufmann, 2012, pp. 83–124.
[13] J. M. C. Sousa and U. Kaymak, *Fuzzy decision making in modeling and control*. World Scientific, 2002, vol. 27.
[14] E. Commission, C. Directorate-General for Communications Networks *et al.*, *Ethics guidelines for trustworthy AI*. Publications Office, 2019.
[15] S. Lloyd, "Least squares quantization in PCM," *IEEE T Inform Theory*, vol. 28, no. 2, pp. 129–137, 1982.
[16] J. C. Bezdek, "Fuzzy mathematics in pattern classification," *Ph. D. Dissertation, Applied Mathematics, Cornell University*, 1973.
[17] J. L. Corcuera Bárcena, F. Marcelloni *et al.*, "A federated fuzzy c-means clustering algorithm," in *Int'l Workshop on Fuzzy Logic and Applications 2021*, vol. 3074, 2021, pp. 1–9.
[18] E. H. Ruspini, J. C. Bezdek *et al.*, "Fuzzy clustering: A historical perspective," *IEEE Comput Intel M*, vol. 14, no. 1, pp. 45–55, 2019.
[19] K. Chandiramani, D. Garg *et al.*, "Performance analysis of distributed and federated learning models on private data," *Procedia Comput Sci*, vol. 165, pp. 349–355, 2019.
[20] J. Dean and S. Ghemawat, "Mapreduce: A flexible data processing tool," *Commun. ACM*, vol. 53, no. 1, p. 72–77, jan 2010.
[21] W. Zhao, H. Ma *et al.*, "Parallel k-means clustering based on mapreduce," in *Cloud Computing*, M. G. Jaatun, G. Zhao *et al.*, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 674–679.
[22] M.-F. F. Balcan, S. Ehrlich *et al.*, "Distributed k-means and k-median Clustering on General Topologies," *Adv Neur In*, vol. 26, pp. 1995–2003, 2013.
[23] A. C.-C. Yao, "How to generate and exchange secrets," in *27th Ann IEEE Symp Found (sfcs 1986)*. IEEE, 1986, pp. 162–167.
[24] L. Lyu, J. C. Bezdek *et al.*, "Privacy-preserving collaborative fuzzy clustering," *Data Knowl Eng*, vol. 116, pp. 21–41, 2018.
[25] F. Meskine and S. N. Bahloul, "Privacy preserving k-means clustering: a survey research." *Int. Arab J. Inf. Technol.*, vol. 9, no. 2, pp. 194–200, 2012.
[26] S. Jha, L. Kruger *et al.*, "Privacy preserving clustering," in *European symposium on research in computer security*. Springer, 2005, pp. 397–417.
[27] J. Vaidya and C. Clifton, "Privacy-preserving k-means clustering over vertically partitioned data," in *Proc of the 9th ACM SIGKDD Int'l Conf on Knowledge discovery and data mining*, 2003, pp. 206–215.
[28] Q. Li, Z. Wen *et al.*, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE T Knowl Data En*, vol. 35, no. 4, pp. 3347–3366, 2023.
[29] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.
[30] D. K. Dennis, T. Li *et al.*, "Heterogeneity for the win: One-shot federated clustering," in *Proc. of the 38th Int'l Conf on Machine Learning*, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 2611–2620.
[31] W. Pedrycz, "Federated FCM: Clustering Under Privacy Requirements," *IEEE T Fuzzy Syst*, pp. 1–1, 2021.
[32] M. Stallmann and A. Wilbik, "On a framework for federated cluster analysis," *Applied Sciences*, vol. 12, no. 20, 2022.
[33] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proc. of the Ann ACM-SIAM Sym on Discrete Algorithms*, vol. 07-09-January, 2007, Conference paper, p. 1027 – 1035.
[34] M. Aledhari, R. Razzak *et al.*, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140 699–140 725, 2020.
[35] V. Mothukuri, R. M. Parizi *et al.*, "A survey on security and privacy of federated learning," *Future Gener Comp Sy*, vol. 115, pp. 619–640, 2021.
[36] Z. Yan, L. Guoliang *et al.*, "A survey on entity alignment of knowledge base," *Journal of Computer Research and Development*, vol. 53, no. 1, p. 165, 2016.
[37] K. Cheng, T. Fan *et al.*, "Secureboost: A lossless federated learning framework," *IEEE Intell Syst*, vol. 36, no. 6, pp. 87–98, 2021.
[38] J. F. Kolen and T. Hutcheson, "Reducing the time complexity of the fuzzy c-means algorithm," *IEEE T Fuzzy Syst*, vol. 10, no. 2, pp. 263–267, 2002.
[39] A. A. Vergani and E. Binaghi, "A Soft Davies-Bouldin Separation Measure," in *2018 IEEE Int'l Conf. on Fuzzy Systems (FUZZ-IEEE)*, 2018, pp. 1–8.
[40] D. Dua and C. Graff, "UCI machine learning repository," 2017.
[41] X. Xie and G. Beni, "A validity measure for fuzzy clustering," *IEEE T Pattern Anal*, vol. 13, no. 8, pp. 841–847, 1991.
[42] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE T Pattern Anal*, vol. PAMI-1, no. 2, pp. 224–227, 1979.
[43] Q. Li, Y. Diao *et al.*, "Federated learning on non-iid data silos: An experimental study," in *2022 IEEE 38th Int'l Conf on Data Engineering (ICDE)*, 2022, pp. 965–978.

José Luis Corcuera Bárcena received the M.Sc. degree in Computer Engineering and the Ph.D. degree in Information Engineering from the University of Pisa in 2020 and in 2024, respectively. Currently, he is a research fellow at the Department of Information Engineering of the University of Pisa, Italy. His research interests include concurrent and distributed systems, machine learning, and competitive programming.

Francesco Marcelloni is currently a full professor at the Department of Information Engineering of the University of Pisa, Italy. His main research interests include federated learning, explainable AI, sentiment analysis and opinion mining, and fuzzy clustering algorithms. He has co-edited three volumes, four journal special issues, and is (co-)author of a book and of more than 250 papers in international journals, books and conference proceedings. He has received the 2021 IEEE TFS Outstanding Paper award and the 2022 IEEE CIM Outstanding Paper award. Currently, he serves as associate editor of IEEE Transactions on Fuzzy Systems, Information Sciences, and Soft Computing, and is on the editorial board of a number of other international journals. He has coordinated various research projects funded by both public and private entities.

Alessandro Renda received the M.Sc. degree in Biomedical Engineering from the University of Pisa, Italy, in 2017 and the Ph.D. degree in Smart Computing, jointly awarded by the Universities of Florence, Pisa and Siena, in 2021. Currently, he is an assistant professor at the Department of Information Engineering of the University of Pisa, Italy. His research interests include machine learning algorithms for data streams, federated learning and applications of deep learning methodologies.

Alessio Bechini earned both his Laurea degree in Electronics Engineering and his PhD degree in Information Engineering from the University of Pisa in 1996 and 2003. At present, he is an associate professor at the Department of Information Engineering of the University of Pisa. His research interests span the fields of concurrent and distributed systems, enterprise information systems, data management and integration, and bioinformatics. Recently, his work has been focused on issues in data mining for Big Data. He has served as TPC co-chair, general co-chair and program co-chair of ACM international conferences.

Pietro Ducange received the M.Sc. degree in Computer Engineering and the Ph.D. degree in Information Engineering from the University of Pisa in 2005 and 2009, respectively. Currently, he is an associate professor of Information Systems and Technologies at the University of Pisa. His main research interests include explainable AI, big data mining, social sensing, and sentiment analysis. He has been involved in a number of R&D projects in which data mining and computational intelligence algorithms have been successfully employed. He has co-authored over 80 papers in international journals and conference proceedings.