

# A Federated Fuzzy $c$ -means Clustering Algorithm

José Luis Corcuera Bárcena<sup>1</sup>, Francesco Marcelloni<sup>1</sup>, Alessandro Renda<sup>1</sup>,  
Alessio Bechini<sup>1</sup> and Pietro Ducange<sup>1</sup>

<sup>1</sup>Department of Information Engineering, University of Pisa, Largo Lucio Lazzarino 1, 56122 Pisa, Italy

## Abstract

Traditional clustering algorithms require data to be centralized on a single machine or in a datacenter. Due to privacy issues and traffic limitations, in several real applications data cannot be transferred, thus hampering the effectiveness of traditional clustering algorithms, which can operate only on locally stored data. In the last years a new paradigm has been gaining popularity: Federated Learning (FL). FL enables the collaborative training of data mining models and, at the same time, preserves data locally at the data owners' places, decoupling the ability to perform machine learning from the need to transfer data. In this context, we propose the federated version of the popular fuzzy  $c$ -means clustering algorithm. We first describe this version through pseudo-code and then demonstrate that the clusters obtained by the federated approach coincide with those generated by the classical algorithm executed on the union of all the local datasets. We also present an analysis on how privacy is preserved. Finally, we show some experimental results on the performance of the federated version when only a number of clients are involved in the clustering process.

## Keywords

Federated Learning, Federated Clustering, Federated Fuzzy  $c$ -Means

## 1. Introduction

The performance of a machine learning (ML) model may benefit from the exploitation of data from multiple sources. However, the conventional approach of collecting data and storing them in a centralised server introduces severe communication overheads, and, most importantly, violates the privacy and security requirements that are often paramount to data owners [1].

As an alternative paradigm to data centralization, Federated Learning (FL) [2] has recently been proposed for the collaborative training of an ML model, and it can represent a key enabler in the framework of computational collective intelligence. In an FL system, data owners are not required to expose their own data to other parties; instead, they learn a shared model via the aggregation of locally-computed updates.

Early works that introduced the concept of FL [1, 3] primarily focused on supervised learning approaches. For example, the seminal algorithm *federated averaging* (FedAvg) [1] allows for the collaborative training of deep neural networks for image classification and language modeling

---


WILF'21: The 13th International Workshop on Fuzzy Logic and Applications

✉ joseluis.corcuera@phd.unipi.it (J. Corcuera Bárcena); francesco.marcelloni@unipi.it (F. Marcelloni);  
alessandro.renda@ing.unipi.it (A. Renda); alessio.bechini@unipi.it (A. Bechini); pietro.ducange@unipi.it  
(P. Ducange)

🆔 0000-0002-9984-1904 (J. Corcuera Bárcena); 0000-0002-5895-876X (F. Marcelloni); 0000-0002-0482-5048  
(A. Renda); 0000-0002-5951-1265 (A. Bechini); 0000-0003-4510-1350 (P. Ducange)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

by iterating over the following steps: (i) the server sends out the global model to the data owners; (ii) each data owner updates the model using its local data and sends it back to the server; (iii) the server takes the average of the locally updated models, weighted according to the number of examples, to obtain a new global model.

The unprecedented performance levels achieved by deep neural networks on a variety of supervised learning tasks likely motivates the emphasis on adapting such models to the federated environment; conversely, much less consideration has been devoted to other ML techniques, and specifically in the field of *cluster analysis* [4]. However, there exist several applications that require to determine groups of objects without sharing local data with a central server: this setting prevents the use of classical clustering algorithms, and asks for novel algorithms properly designed for a federated environment. Thus, the reshaping of the most effective clustering algorithms to reap the benefits of FL is of particular importance from a practical perspective.

A recent review [5] has highlighted how the mathematical representation of uncertainty provided by fuzzy set theory has historically found considerable popularity in cluster analysis. The Fuzzy  $c$ -means (FCM) [6] algorithm certainly is the most popular *fuzzy* clustering method: it partitions data objects into  $c$  clusters, where  $c$  is fixed by the user.

In this paper we discuss the adaptation of FCM algorithm to the federated setting, considering the scenario of horizontally partitioned data, where objects are spread over multiple distinct nodes, and all of them are described by the same set of attributes.

The paper is organized as follows: Section 2 reports related works. Section 3 describes the background related to FL and the general setting of our investigation. Section 4 introduces our Federated FCM. Section 5 describes the experimental setup and results. Section 6 draw concluding remarks.

## 2. Related Works

The FCM algorithm and its crisp counterpart  $c$ -means [7] are among the most popular and widely used clustering algorithms. Since their introduction, a number of works have extended and adapted the original algorithms to different application scenarios. This section can only provide some insights by describing how some more recent works relate to our proposal.

As discussed in a survey paper [8], several works have provided contributions around privacy-preserving  $c$ -means clustering algorithm: although complete zero-knowledge (i.e., each party knows nothing except its input and output) cannot be achieved due to the iterative nature of the algorithm, private implementations have been proposed both for horizontally [9] and vertically [10] partitioned data. However, the adoption of cryptography and Secure Multi-Party Computation primitives introduces a severe computation overhead and hinders the scalability of the approaches [11]. In a recent remarkable proposal for privacy-preserving collaborative fuzzy clustering [11], each data owner transforms its own data by first applying a non linear function and then performing a random projection onto a lower dimensional space. This perturbation is independent of the subsequent clustering algorithm (specifically, FCM is used). Furthermore, it is assumed that the central server may collude with some of the participants and may conduct specific attacks to violate user's privacy. The authors show how resistance or mitigation of these attacks is achieved at a limited cost in terms of accuracy loss.

Compared to the representative works mentioned above, we assume a *semi-honest* central server: the server can try to retrieve private raw data based on the updates communicated by the data owners, but it does so by adhering to the protocol defined for the execution of the ML algorithm. This is a typical assumption for the horizontal partitioning [2]; however, clustering under such *weak* privacy model has not been extensively investigated yet in FL literature. To the best of our knowledge, only one recent work [12] follows an approach related to ours, from the point of view of privacy model, data partitioning, and communication topology: in a federated version of FCM, the author proposes to determine the cluster centers by means of a gradient-based optimization procedure. Although it is shown that such a federated version of FCM obtains similar results (and not equal, as in our case) compared to the traditional FCM algorithm applied to the overall dataset, the gradient-based optimization procedure differs from the classical, iterative, two-step minimization, in which the membership matrix and the centroids are alternately optimized. On the contrary, our proposal is inspired by the optimization procedure adopted in the traditional FCM algorithm: by only communicating aggregated data, it enables federated clustering without violating the privacy of participants.

### 3. Background and Problem Statement

A thorough overview on FL has recently been presented in several surveys, such as [2, 4, 13]. A general definition of FL can be found in [2]. Let  $\{P^1, P^2, \dots, P^M\}$  be  $M$  parties, i.e. data owners, who wish to train an ML model by consolidating their respective data  $\{X^1, X^2, \dots, X^M\}$ , where  $X^m = \{\mathbf{x}_1^m, \mathbf{x}_2^m, \dots, \mathbf{x}_{N_j}^m\}$  and  $\mathbf{x}_j^m$  is an object stored in the  $m$ -th party. In an FL process the parties collaboratively learn a model  $Model_{fed}$  without exposing their private data to others. The accuracy of  $Model_{fed}$  should be close to the one achieved by a model  $Model_{sum}$  learned on the union of the local datasets  $X_{sum} = \bigcup_{m=1}^M X^m$ . Specifically, given a non-negative real number  $\delta$ , the FL algorithm is said to have  $\delta$ -accuracy loss if  $|Acc_{fed} - Acc_{sum}| < \delta$ , where  $Acc_{fed}$  and  $Acc_{sum}$  are the accuracies of  $Model_{fed}$  and  $Model_{sum}$ , respectively.

The above definition is quite general and covers a wide spectrum of FL applications. Indeed, FL systems can be coarsely categorized based on two aspects: data partitioning and communication topology [13]. *Data partitioning* relates to how data are distributed across the various parties and can be categorized in *horizontal* and *vertical* FL. Let  $F^i$  be the feature space and  $I^i$  the sample ID space of the data  $X^i$  held by the  $i$ -th data owner. As per horizontal FL, the dataset is said to be *sample-partitioned*. Horizontal FL can be formalized as follows:

$$F^i = F^j, \quad I^i \neq I^j, \quad \forall X^i, X^j, i \neq j \quad (1)$$

As per Vertical FL, the dataset is said to be *feature-partitioned*. Vertical FL can be formalized as follows:

$$F^i \neq F^j, \quad I^i = I^j, \quad \forall X^i, X^j, i \neq j \quad (2)$$

FL systems can be categorized according to their *communication topology*, thus discriminating between *centralized* and *fully-decentralized* FL. The former entails a server that orchestrates

the learning process by aggregating the updates computed by different parties; most FL implementations assume this communication topology. The latter does not require the presence of a central server: information is shared in a peer-to-peer fashion.

In this paper, 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, but with no need to share their private raw data. Let  $X^1 = \{\mathbf{x}_1^1, \mathbf{x}_2^1, \dots, \mathbf{x}_{N_1}^1\}$ ,  $X^2 = \{\mathbf{x}_1^2, \mathbf{x}_2^2, \dots, \mathbf{x}_{N_2}^2\}$ ,  $\dots$ ,  $X^M = \{\mathbf{x}_1^M, \mathbf{x}_2^M, \dots, \mathbf{x}_{N_M}^M\}$  be the  $M$  private datasets we are considering, each of them with a variable number of objects. Instances from all the datasets are represented in the same  $F$ -dimensional attribute space:  $\mathbf{x}_j^m = \{x_{j,1}^m, x_{j,2}^m, \dots, x_{j,F}^m\}$ . We assume that both the number of clusters and the domain of definition of the attributes are known a-priori, and, specifically, they are also known to the server. As per the privacy model, we assume honest participants and a *semi-honest*, or *honest-but-curious*, central server [2].

## 4. Federated FCM

The original FCM algorithm by Bezdek [6] has been later reworked to improve its efficiency [14]. Our federated proposal for FCM stems from such a version [14], which adopts an efficient reorganization of the update procedure for the cluster centers: no storing of the membership matrix (i.e. the matrix that contains the degree of membership of each object to each cluster) is required, thus the asymptotic runtime is significantly reduced.

Algorithm 1: **Federated FCM**( $C, \lambda, \varepsilon$ ).

**Given:**  $C$  - number of clusters  
**Given:**  $\lambda$  - fuzziness factor  
**Given:**  $\varepsilon$  - tolerance value for the stop condition

**Initialization stage**  
*Server:*  
1: Randomly selects  $C$  cluster centers.  $V^{(0)} = \{\mathbf{v}_1^{(0)}, \mathbf{v}_2^{(0)}, \dots, \mathbf{v}_C^{(0)}\}$   
2: Transmits the fuzziness factor  $\lambda$  to each data owner

**Execution stage**  
3: At each round  $t$ , with  $t$  starting from 0  
*Server:*  
4: Transmits  $V^{(t)}$  to each data owner  
*Each data owner  $m$ :*  
5:  $U^{(t),m}, WS^{(t),m} = \text{LocalSums}(V^{(t)}, X^m, \lambda)$   
6: Transmit  $(U^{(t),m}, WS^{(t),m})$  to the server  
*Server:*  
7: Update cluster centers evaluating  $V^{(t+1)}$  as per Eq. 3.  
8: **if**  $\|V^{(t+1)} - V^{(t)}\|_F < \varepsilon$  **then**  
9:    Terminate  
10: **else**  
11:    Proceed with the next round (line 4)  
12: **end if**

Algorithm 2: **LocalSums**( $V^{(t)}, X^m, \lambda$ )

**Given:**  $V^{(t)}$  - array of  $C$  cluster centers  
**Given:**  $X^m$  -  $m$ -th dataset  
**Given:**  $\lambda$  - fuzziness factor

- 1:  $WS^{(t),m} \leftarrow \text{zeros}(C \times F)$
- 2:  $U^{(t),m} \leftarrow \text{zeros}(C)$
- 3: **for**  $\mathbf{x}_j^m \in X^m$  **do**
- 4:     $denom = 0$
- 5:    **for** each cluster  $c$  **do**:
- 6:        $numer_c = \|\mathbf{x}_j^m - \mathbf{v}_c^{(t)}\|^{\frac{2}{\lambda-1}}$
- 7:        $denom = denom + \frac{1}{numer_c}$
- 8:    **end for**
- 9:    **for** each cluster  $c$  **do**:
- 10:        $\mu_{c,j} = (numer_c * denom)^{-1}$
- 11:        $\mathbf{ws}_c^{(t),m} = \mathbf{ws}_c^{(t),m} + \mu_{c,j}^\lambda \mathbf{x}_j^m$
- 12:        $U_c^{(t),m} = U_c^{(t),m} + \mu_{c,j}^\lambda$
- 13:    **end for**
- 14: **end for**
- 15: **return**  $U^{(t),m}, WS^{(t),m}$

The pseudocode of the Federated FCM algorithm is reported in Algorithms 1 and 2. Federated FCM is structured in successive rounds. For the sake of clarity, the communication and synchronization details for this round-based organization is omitted.

The initialization stage of the Federated FCM algorithm is up to the server and involves the initialization of all the cluster centers. The server transmits to each data owner the coordinates of the centers and the fuzziness parameter  $\lambda$ . Subsequently, during the execution stage, at each round, each data owner computes both  $U^{(t),m}$  and  $WS^{(t),m}$  as described in Algorithm 2.  $U^m$  is an array with  $C$  elements, and the  $c$ -th element is the sum of the membership degrees of the objects in  $X_m$  to cluster  $c$ , each raised to the  $\lambda$ -th power. More formally,  $U^m = \{U_1^m, U_2^m, \dots, U_C^m\}$  and  $U_c^m = \sum_{j=1}^{N_m} \mu_{c,j}^\lambda$ .  $WS^m$  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  $c$ . More formally,  $WS^m = \{\mathbf{ws}_1^m, \mathbf{ws}_2^m, \dots, \mathbf{ws}_C^m\}$  and  $\mathbf{ws}_c^m = \sum_{j=1}^{N_m} \mu_{c,j}^\lambda \mathbf{x}_j^m$ . Then, each data owner transmits  $U^{(t),m}$  and  $WS^{(t),m}$  to the server, so that it can update the coordinates of the centers as follows:

$$\mathbf{v}_c^{(t+1)} = \frac{\sum_{m=1}^M \mathbf{ws}_c^{(t),m}}{\sum_{m=1}^M U_c^{(t),m}} \quad \forall c \in \{1, \dots, C\} \quad (3)$$

The server evaluates the stopping condition (Algorithm 1, line 8): if the Frobenius norm of the difference in the cluster centers between two consecutive rounds is lower than the given threshold  $\varepsilon$ , the execution terminates. In other words, we check whether the centers move less than the imposed tolerance. Otherwise, the server transmits the new centers to the data owners, thereby initiating the next round.

The rationale behind the LocalSums (Algorithm 2) is detailed in [14]; unlike the original paper, we are considering the scenario of horizontal FL, where data are scattered over multiple data owners. Notably, each data owner does not need to store the  $C \times N_m$  membership matrix.

#### 4.1. Equivalence with FCM executed on the overall dataset

In this sub-section, we show that the Federated FCM algorithm generates the same clusters that would be produced by the classical FCM algorithm applied to the overall dataset obtained by the union  $X_{sum}$  of the local datasets. Under the assumption that all data are stored in the central server, in the classical FCM algorithm at each iteration the cluster centers are updated as follows:

$$\mathbf{v}_c^{(t+1)} = \frac{\sum_{j=1}^N \mu_{c,j}^\lambda \mathbf{x}_j^{(t),sum}}{\sum_{j=1}^N \mu_{c,j}^\lambda}, \quad \forall c \in \{1, \dots, C\} \quad (4)$$

The updated center of the  $c$ -th cluster is the weighted average of the objects assigned to the cluster: the weights are the membership degree of the objects to the cluster, raised to the  $\lambda$ -th power. With  $M$  mutually disjoint sets, as it typically occurs in real-world horizontal data partitions, Eq. 4 (centralized setting) and Eq. 3 (federated setting) lead to the same result. In fact, for each cluster  $c$ ,

$$\sum_{j=1}^N \mu_{c,j}^\lambda \mathbf{x}_j^{(t),sum} = \sum_{m=1}^M \mathbf{ws}_c^{(t),m} \quad (5)$$

$$\sum_{j=1}^N \mu_{c,j}^\lambda = \sum_{m=1}^M U_c^{(t),m} \quad (6)$$

Specifically, both terms are sums of elements. In the federated setting each sum is performed in two steps: first, on a per-owner basis and then aggregated by the central server.

Finally, since the termination condition (Algorithm 1 line 8) depends only on the position of the cluster centers, it can be applied equivalently in the two scenarios.

## 4.2. Privacy analysis

The *federated* approach allows the collaborative evaluation of the FCM algorithm with no need for individual data owners to expose their data. Nevertheless, the server can infer the value of the attributes of each single object under certain circumstances. At each round, each data owner reveals the following information: for each  $c$ , the sum  $\mathbf{U}_c^{(t),m}$  of the membership degrees, each raised to the  $\lambda$ -th power, and the array  $\mathbf{ws}_c^{(t),m}$  of the weighted sum of the objects assigned to  $c$ . The first information  $\mathbf{U}_c^{(t),m}$  can be represented by the following  $C$  equations:

$$\begin{cases} \mu_{1,1}^\lambda + \mu_{1,2}^\lambda + \cdots + \mu_{1,N_m}^\lambda = \alpha_1 \\ \mu_{2,1}^\lambda + \mu_{2,2}^\lambda + \cdots + \mu_{2,N_m}^\lambda = \alpha_2 \\ \cdots \\ \mu_{C,1}^\lambda + \mu_{C,2}^\lambda + \cdots + \mu_{C,N_m}^\lambda = \alpha_C \end{cases} \quad (7)$$

The second information  $\mathbf{ws}_c^{(t),m}$  leads to the following  $C$  equations for each attribute  $f$ :

$$\begin{cases} x_{1,f} \mu_{1,1}^\lambda + x_{2,f} \mu_{1,2}^\lambda + \cdots + x_{N_m,f} \mu_{1,N_m}^\lambda = \beta_{1,f} \\ x_{1,f} \mu_{2,1}^\lambda + x_{2,f} \mu_{2,2}^\lambda + \cdots + x_{N_m,f} \mu_{2,N_m}^\lambda = \beta_{2,f} \\ \cdots \\ x_{1,f} \mu_{C,1}^\lambda + x_{2,f} \mu_{C,2}^\lambda + \cdots + x_{N_m,f} \mu_{C,N_m}^\lambda = \beta_{C,f} \end{cases} \quad (8)$$

Specifically, the server receives from each data owner  $m$  the  $\alpha_c$  value for each cluster  $c$ , and the  $\beta_{c,f}$  value for each cluster  $c$  and for each feature  $f$ . By definition, we have that

$$\mu_{c,j} = \frac{1}{\sum_{l=1}^C \left( \frac{\|\mathbf{x}_j^m - \mathbf{v}_c\|}{\|\mathbf{x}_j^m - \mathbf{v}_l\|} \right)^{\frac{2}{\lambda-1}}} \quad (9)$$

Thus, we can consider that the only unknown variables are the coordinates  $x_{j,f}^m$ . To derive these coordinates for the objects stored in party  $m$ , the server must solve the overall system of equations, composed by Eqs. 7 and 8, replacing the values of  $\mu_{c,j}$  by using Eq. 9. The number of unknown variables is  $N_m \times F$  and the number of equations is  $C + C \times F$ . If the server was aware of the number of objects  $N_m$  that produced the statistics  $\mathbf{U}_c^{(t),m}$  and  $\mathbf{ws}_c^{(t),m}$ , it could empirically 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  $N_m$  and therefore this is a strong restrictive condition. Nevertheless, it does not appear too limiting 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 party transmit the locally aggregated data (Algorithm 1, line 6) 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.



## 5. Experimental setup and results discussion

A preliminary evaluation of the performance of Federated FCM has been carried out. We considered the following setting: all participants have roughly the same amount and the same distribution of objects (IID setting). The total number of participants  $M$  has been set to 20; however, as in the original FedAvg algorithm [1], we introduce a parameter  $\gamma$  (fraction of data owners) for random sampling among participants at the beginning of each FL round. The main objective of our analysis is to compare the results obtained with Federated FCM (with different values for  $\gamma$ ) and those obtained with classical FCM algorithm applied to the overall dataset  $X_{sum}$ . Specifically,  $V_{fed}(\gamma)$  and  $V_{sum}$  represents the final array of cluster centers obtained with Federated FCM and with classical FCM (centralized setting), respectively.

Overall, we employed four datasets: *xclara* and *s-set1* from the clustering benchmark repository<sup>1</sup>, *waveform v1* and *pendigits* from the UCI Machine Learning Repository<sup>2</sup>. For each dataset, the number of classes  $K$ , objects  $N$  and attributes  $F$  is reported in Table 1 (first column).

We set the fuzziness factor  $\lambda = 2$ , the tolerance value  $\varepsilon = 0.005$  and the maximum number of rounds to 30, as we observed that convergence generally requires far fewer rounds. Further the number  $C$  of clusters was set coherently with the number of classes  $K$ . We varied  $\gamma$  in  $\{0.25, 0.5, 0.75, 1\}$ : notably, in the case of  $\gamma = 1$  (i.e., all clients participate to FL procedure), we aim to experimentally verify the equivalence between federated and centralized settings, as theoretically demonstrated in Section 4.1. We performed 10 repetitions varying the seed for random center initialization and participant sampling; for each repetition, the same center initialization was used for the Federated FCM and the centralized FCM algorithms.

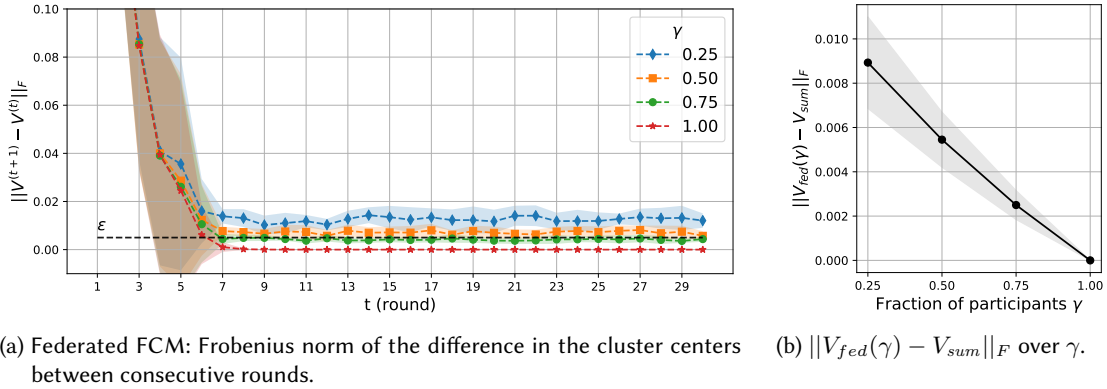
Due to the space limit, Fig. 1 reports only the results obtained on the *xclara* dataset. The convergence of Federated FCM (Algorithm 1, line 8) is reported in Fig. 1a: the Frobenius norm of the difference in the cluster centers between two consecutive rounds reaches a plateau after around 7 rounds, regardless of the value of  $\gamma$ . However, convergence w.r.t. the tolerance value  $\varepsilon$  is achieved only for high values of  $\gamma$  (namely, 0.75 and 1); intuitively, when fewer participants are sampled, the centers keep slightly shifting over time, since it is less likely that the same participants are selected in subsequent rounds.

Figure 1b shows the Frobenius norm  $\|V_{fed}(\gamma) - V_{sum}\|_F$  of the difference between cluster centers computed by Federated and centralized versions of FCM; notably, we consider the permutation of the centers in  $V_{fed}(\gamma)$  that minimizes  $\|V_{fed}(\gamma) - V_{sum}\|_F$ . When  $\gamma = 1$  (all 20 clients are involved in the procedure) the federated version of the algorithm is equivalent to the original one executed on the overall dataset, as previously demonstrated. As  $\gamma$  decreases, the distance between the centers increases. To assess whether this affects clustering performance, we apply an external measure of clustering quality, namely the Adjusted Rand Index (ARI) [15]. Given the centers  $V_{fed}(\gamma)$  computed by the federated version, we evaluated the fuzzy partition matrix on the dataset  $X_{sum}$ , assigned each object to the cluster with the highest membership degree, and computed ARI, which measures the consistency between clustering results and the available ground truth labels. Table 1 reports the results for all datasets, along with the value  $\|V_{fed}(\gamma) - V_{sum}\|_F$ . We observe that, in the case  $\gamma < 1$ , the centers computed in the

---

<sup>1</sup><https://github.com/deric/clustering-benchmark>

<sup>2</sup><https://archive.ics.uci.edu/ml/datasets.php>



**Figure 1:** Results on *xclara* dataset. Average values (shaded region indicates the standard deviation).

federated setting slightly deviates from those determined in the centralized case. As expected, this deviation decreases with the increase of  $\gamma$ . However, such deviation does not induce a significant variation in the ARI values, thus highlighting that the clusters generated by Federated and centralized versions are very similar even when only a fraction of participants is involved in the FL procedure. This outcome is particularly relevant in scenarios where clients have an unstable connection, which can prevent them to communicate with the central server at each round, or when it is mandatory to reduce the communication overhead over the network.

**Table 1**

ARIs obtained with centers  $V_{fed}(\gamma)$  on  $X_{sum}$  and Frobenius norm of the difference between  $V_{fed}(\gamma)$  and  $V_{sum}$ . Average values.

Dataset ( $K, N, F$ ) \ $\gamma$	ARI( $\gamma$ )				$\ V_{fed}(\gamma) - V_{sum}\ _F$			
	0.25	0.50	0.75	1.00	0.25	0.50	0.75	1.00
<b>xclara</b> (3, 3000, 2)	0.99269	0.99279	0.99289	0.99289	0.00893	0.00545	0.00250	0.00000
<b>s-set1</b> (15, 5000, 2)	0.90418	0.90384	0.89645	0.89728	0.11640	0.09915	0.04865	0.00000
<b>pendigits</b> (10, 10992, 16)	0.42051	0.42115	0.42291	0.42468	0.11147	0.08261	0.03018	0.00000
<b>waveform v1</b> (3, 5000, 21)	0.24243	0.24366	0.24445	0.24359	0.03233	0.01877	0.02049	0.00000

## 6. Conclusion

While ML is widely employed in a great variety of application domains, there is a growing interest in the need to protect the privacy of data during the collaborative learning of ML models. Federated learning is emerging as one of the key paradigms to address this challenge. In this paper we have proposed a *federated* version of the popular fuzzy *c*-means clustering algorithm. We have focused on the scenario of horizontally partitioned data and relaxed privacy requirements. We have shown that our version achieves the same results obtained by the classical clustering algorithm applied to the overall merged datasets, while preserving privacy of data owners. Further, we carried out a preliminary experimental analysis: under the assumption



of IID data over clients, the federated version of the algorithm is substantially equivalent to the centralized one, even when only a fraction of clients participate to the FL process. Future works will investigate the selection of the proper number of clusters in the federated setting and the implementation of the algorithm for vertically partitioned data.

## Acknowledgments

This work has been partly funded by the European Commission through the H2020 project Hexa-X (Grant Agreement no. 101015956) and by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence).

## References

- [1] B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: *Artificial Intelligence and Statistics*, PMLR, 2017, pp. 1273–1282.
- [2] Q. Yang, Y. Liu, T. Chen, Y. Tong, Federated machine learning: Concept and applications, *ACM T. Intel. Sysy. Tec.* 10 (2019) 1–19.
- [3] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtarik, A. T. Suresh, D. Bacon, Federated Learning: Strategies for Improving Communication Efficiency, in: *NIPS Workshop on Private Multi-Party Machine Learning*, 2016.
- [4] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, et al., Advances and open problems in federated learning, *arXiv preprint arXiv:1912.04977* (2019).
- [5] E. H. Ruspini, J. C. Bezdek, J. M. Keller, Fuzzy clustering: A historical perspective, *IEEE Comput. Intell. M.* 14 (2019) 45–55.
- [6] J. C. Bezdek, Fuzzy mathematics in pattern classification, Ph. D. Dissertation, Applied Mathematics, Cornell University (1973).
- [7] S. Lloyd, Least squares quantization in PCM, *IEEE T. Inform. Theory* 28 (1982) 129–137.
- [8] F. Meskine, S. N. Bahloul, Privacy preserving k-means clustering: a survey research., *Int. Arab J. Inf. Technol.* 9 (2012) 194–200.
- [9] S. Jha, L. Kruger, P. McDaniel, Privacy preserving clustering, in: *European symposium on research in computer security*, Springer, 2005, pp. 397–417.
- [10] J. Vaidya, C. Clifton, Privacy-preserving k-means clustering over vertically partitioned data, in: *Proc. 9th ACM SIGKDD Int'l Conf. on KDD*, 2003, pp. 206–215.
- [11] L. Lyu, J. C. Bezdek, Y. W. Law, X. He, M. Palaniswami, Privacy-preserving collaborative fuzzy clustering, *Data Knowl. Eng.* 116 (2018) 21–41.
- [12] W. Pedrycz, Federated fcm: Clustering under privacy requirements, *IEEE T. Fuzzy Syst.* (2021) 1–1.
- [13] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. e. a. Dehghantanha, A survey on security and privacy of federated learning, *Future Gener. Comp. Sy.* 115 (2021) 619–640.
- [14] J. F. Kolen, T. Hutcheson, Reducing the time complexity of the fuzzy c-means algorithm, *IEEE T. Fuzzy Syst.* 10 (2002) 263–267.
- [15] L. Hubert, P. Arabie, Comparing partitions, *Journal of Classification* 2 (1985) 193–218.