FedHC: Learning Imbalanced Clusters via Federated Hierarchical Clustering

Yue Zhang[1,4(✉)], Xinfa Liao[1], Qingsheng Chen[2], Haotian Wu[3], and Yiqun Zhang[2]

[1] School of Computer Science, Guangdong Polytechnic Normal University, Guangzhou, China
[2] School of Computer Science and Technology, Guangdong University of Technology, Guangzhou, China
[3] Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou, China
[4] Guangdong Provincial Key Laboratory of Intellectual Property and Big Data, Guangzhou, China
zhangyue@gpnu.edu.cn, L593486501@gmail.com, 2112205080@mail2.gdut.edu.cn, wuht@gzhu.edu.cn, yqzhang@gdut.edu.cn

Abstract. Federated learning has been widely studied in recent years, which acts to avoid the privacy leakage problem while ensuring co-learning among clients. Most existing federated clustering methods are mainly focusing on extending the learning strategy of k-means to a federated scenario. This unavoidably makes the clustering inherit the inherent limitations brought by k-means, i.e., each learning client requires the number of true clusters k to be given in advance, which is not always the case in reality. Moreover, the "uniform effect" that prevents the existing clustering methods from effectively partitioning imbalanced clusters also remains unsolved for federated clustering. It is worth noting that a too-small k also contributes to the "uniform effect" as the granularity that will be searched by the clustering algorithm is fixed to be relatively large. To solve the above problems, we propose the Federated Hierarchical Clustering (FedHC) method to explore subclusters at each client and then merge them at the server to form a sought number of clusters. Such a process simultaneously protects privacy and aggregates the cluster distribution information that is finely learned by each client without requiring a pre-set "true" local k. Since we do not force each client to search for the same and small number of clusters, they can provide rich micro-cluster distribution information to the server, and the server hierarchically merges closely distributed subclusters to avoid the "uniform effect". It turns out that FedHC can effectively explore imbalanced clusters without passing the privacy information. Several experiments have been conducted to illustrate the efficacy of FedHC.

Keywords: Federated learning · hierarchical clustering · imbalanced data · number of clusters.

1 Introduction

Federated learning is an approach for distributed learning while protecting the privacy among several cooperated clients, which was originally introduced by Google in 2016 [7]. It allows for model training on the local device and then building global models through joint learning of model parameters. To keep privacy confidential, the data of each client will not be directly uploaded to the server. However, this learning constraint poses great challenges to unsupervised cluster learning as the data label is unavailable on both the server and the clients, preventing the learning from being consensus and complementary. More specifically, in most real cases, the number of sought clusters k is unknown by the clients for local learning, and thus the learned information propagated to the server cannot well-aggregate for exploring appropriate cluster distributions, especially when the global cluster distribution is imbalanced.

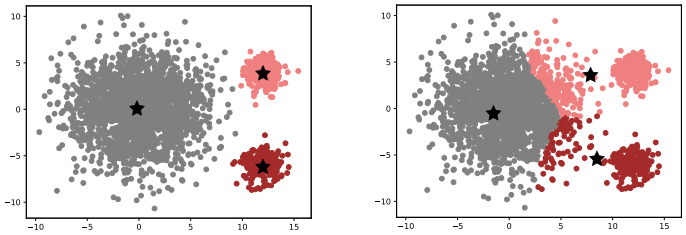

Fig. 1. (a) is the correct clustering result and (b) is the result of the "uniform effect" occurring during clustering. The star symbol represents the corresponding centroid.

The current federated clustering approaches share incomplete data information with the server, primarily following a k-means-like approach [4]. These methods essentially integrate centralized clustering methods into federated learning frameworks. The core idea is to perform initial clustering at the clients and then upload the cluster centroids to the server for further global clustering. However, these methods often encounter challenges in handling imbalanced datasets, leading to the occurrence of a "uniform effect". This effect refers to the tendency of cluster centroids to gradually shift towards regions of higher-density data during the partitioning process, resulting in the misclassification of data points originally belonging to high-density clusters into low-density clusters, as shown in Fig. 1.

In general, the main shortcomings of existing methods are as follows: Most of them are based on partition-based clustering, which performs poorly on handling imbalanced data; clustering at the client side requires knowledge of the cluster number k, but sometimes we only know the total number of clusters in the entire dataset and cannot determine how many clusters these data points at the client side actually belong to. Therefore, we propose a federated clustering method, attempting to address the above shortcomings. Unlike traditional federated clustering methods resembling k-means, our approach is based on the idea

of merging similar clusters. Initially, data is clustered into several subclusters on the client side, and then some parameters of these subclusters are uploaded to the server. Finally, on the server side, these parameters are used to merge subclusters from different clients to complete global clustering. This approach is hierarchical clustering rather than partition-based clustering. Compared to existing federated clustering methods, our proposed method shows improvement in the task of clustering imbalanced data. The main contributions are summarized below:

- This paper deals for the first time with the ubiquitous but not yet well-resolved federated clustering of imbalanced data, providing an effective federated clustering paradigm for subsequent research.
- We propose to partition the local client data into micro-clusters, which are then merged hierarchically according to their proxy uploaded to the server, so as to avoid overlooking the minor clusters, and avoid the "uniform effect" that may occur in the divisional clustering used in the existing methods. Therefore, our algorithm will have a better result in the clustering of imbalanced datasets.
- Unlike existing k-means-like federated clustering algorithms, on the client side, we do not rely on a given k. This largely eliminates assumptions about the cluster distribution at the clients, giving them greater freedom to search for cluster distributions that are closer to reality.

2 Related Work

2.1 Federated Clustering

The current work on clustering under federated learning is roughly as follows:

Clustered Federated Learning. Because of the similarity in terminology, let's start by describing these works and pointing out how they differ from ours. These works explore some ways to perform clustering in supervised federated learning for handling non-independently identically distributed data better [1]. These are typically implemented through clustering clients, focusing on downstream supervised learning tasks and using iterative or centralized clustering schemes [17,6].

Federated Clustering. These methods initially focus on identifying global clusters in distributed data without sharing data. It has been explored to extend the k-mean algorithm to the federated setting, proposing a global average function to update the global cluster center [8]. In addition, there is a fuzzy version that uses fuzzy assignments as weights [16]. The method has not been fully explored but has shown potential for identifying reasonable clusters in some experiments. Later, Li et al [9] proposed a federated clustering method through sharing data, their approach primarily involves encrypting local data at the client before sending it to the server for clustering.

One-Shot Federated Clustering. Some studies have demonstrated the potential of one-shot clustering, utilizing it as a simple preprocessing step to

achieve personalized federated learning [4]. The federated learning approach will increase communication overheads to some extent [10], this method [4] not only achieves performance similar to or superior to the latest iterative methods but also does not require multiple rounds of communication [6]. In addition, the processing of high-dimensional data through federated clustering was recently explored by Xie et al [19], they proposed a federated subspace clustering method to solve the problem that k-means clustering, such as [4], underperforms in high-dimensional space.

These studies explore various aspects of clustering in federated learning, providing multiple methods and insights for addressing clustering problems in the distributed computing environment.

2.2 Imbalance Clustering

In recent years, researchers have extensively explored the problem of imbalanced clustering. However, much of the attention has been focused on class imbalance issues, as evidenced by the work of Lin et al [12], the scenario of imbalanced clusters, where the number of samples within clusters is uneven, has not yet received sufficient attention. Currently, there are numerous studies exploring clustering on imbalanced data. For instance, by investigating the performance of k-means on imbalanced data, Xiong et al [20] proposed the concept of uniform effect and introduced the coefficient of variance (CV) as an evaluation metric. Liang et al [11] proposed an algorithm based on fuzzy k-means clustering, which conducts clustering on imbalanced data through a three-stage process. However, it faces issues such as predefined prototype numbers and manual selection of cluster quantities. Furthermore, some studies have employed ensemble clustering or spectral clustering methods to address the issue of clustering on imbalanced data. However, these methods have not adequately addressed the scenario of imbalanced clusters. In this background, Lu et al [14]proposed a new method, which is based on [3,22,21], named "Self-Adaptive Multiprototype-Based Competitive Learning"(SMCL), this method has been one of the more effective approaches for handling imbalanced data in recent years, in addition, this work is further investigated in [23]. Regarding clustering on imbalanced data, most of the existing methods proposed are traditional centralized clustering approaches. However, in the realm of federated clustering, this issue has not yet been adequately addressed.

3 FedHC: Federated Hierarchical Clustering Method

The problem we aim to address is the clustering of imbalanced datasets distributed across multiple clients in a federated learning environment. The data originates from multiple clients, each client potentially containing a varying number of samples, with each sample represented by a set of attributes. These clients may exhibit different data distributions and scales, necessitating consideration

of data imbalances during clustering. Our objective is to obtain a global distribution of sample clusters at the server while prohibiting the transmission of raw sample data.

FedHC mainly consists of two stages, they are local clustering on the client side and global clustering on the server side. To effectively handle imbalanced data, we adopt a method of merging subclusters. This approach, which merges subclusters from fine to coarse granularity, can avoid missing the detection of small clusters and also mitigate the problem of centroid deviation that occurs in partition-based clustering methods used by existing approaches. The main process of FedHC is as follows: 1. Partitioning the data at each client-side into several subclusters; 2. Calculating the parameters such as radius, cluster size, standard deviation, etc., of these subclusters at each client side and uploading them to the server; 3. The server uses the merge subcluster algorithm to merge all subclusters from clients based on the obtained parameters to obtain the final clustering result.

3.1 Basic Notations

Firstly, we introduce some of the basic symbols that will be used. Let C be a cluster and c be the centroid of the cluster C. Let $X_C = \{x_1^{(C)}, ..., x_{n_c}^{(C)}\}$ be a set of the data points from cluster C, thus, n_c is the number of all data points from cluster C. Let $dist(x_i, x_j)$ be the Euclidean distance between data point x_i to data point x_j. And then, we index individual devices by $z \in [Z]$. Let r_C be the radius of cluster C and s_C be the standard deviation of cluster C. These are the main basic notations involved in our proposed approach.

3.2 ClientMP: Automated Micro-Partition at Clients

In FedHC, firstly, we perform preliminary clustering on the client side to obtain several subclusters. At each client $z \in [Z]$, we note the data it has as $A^{(z)}$, we can use many existing methods to partitioning $A^{(z)}$, and denote the partitioning method as "P-Method", P-Method can be but not limited to DBSCAN [18], "Affinity Propagation" algorithm [5], "Mean Shift" algorithm [2]. Then, we will get some subclusters $\{C_1, ..., C_{k'}\}$ and the corresponding set of centroids($centroids = \{c_1, ..., c_{k'}\}$), k' denotes the number of subclusters. In this process, if the obtained subclusters have more similar sizes, we can achieve better results in the subsequent merging process, ultimately leading to a good clustering result.

After completing preliminary clustering on the client side, we obtain several subclusters. To ensure the transmission of data, obtained from the client, to the server while avoiding leakage of sample privacy, we designed a transmission strategy, that computes the radius, size, and standard deviation of the subclusters obtained from the client side, and then transmits these three parameters instead of the original data to the server for the next stage of clustering work. We calculate the relevant parameters as follows: Radius of a subcluster is the

average distance from all points within a cluster to the centroid, calculated by

$$r_C = \frac{\sum_{i=1}^{n_C} dist(c, x_i^C)}{n_C}. \tag{1}$$

Size is the number of data points within the cluster. Standard deviation is the standard deviation of data points within the cluster, calculated by

$$s_C = \sqrt{\frac{\sum_{i=1}^{n_C}(x_i - \overline{x})^2}{n_C - 1}}, \overline{x} = \frac{\sum_{j=1}^{n_C} x_j}{n_C}. \tag{2}$$

If the data is multidimensional, it is sufficient to calculate the standard deviation of each dimension and compose it into a vector. After completing the calculations of $R = \{r_{C_1}, ..., r_{C_{k'}}\}$, $N = \{n_{C_1}, ..., n_{C_{k'}}\}$ and $S = \{s_{C_1}, ..., s_{C_{k'}}\}$, we upload them to the server.

3.3 ServerHM: Hierarchical Merging at Server

The Merge Subcluster (referred to as MS hereafter) is a merging method proposed by us for performing global clustering on the server side. Its core idea is to continuously merge the subclusters obtained through preliminary clustering at the client side based on the similarity between clusters until the specified number of K clusters remains. To evaluate the similarity between two subclusters, we have defined a distance equation, denoted as

$$d_{C_i,C_j} = dist(c_i, c_j) * o_{C_i,C_j} * sim^{\beta}_{C_i,C_j}. \tag{3}$$

When the value of d is smaller, it indicates a smaller special distance between the corresponding two subclusters, implying greater similarity.

The o coefficient represents the degree of overlap between two subclusters. Let's consider two circular clusters, where r_1 and r_2 are the radii of these two circles, and d is the distance between the centers of the circles. Then, $r_1 + r_2$ is a fixed value, while the ratio of d to $r_1 + r_2$ varies with the degree of overlap between these two circles. When the circles do not overlap, this ratio is greater than 1; when the circles overlap, the ratio is less than 1, and the greater the degree of overlap, the smaller this ratio becomes. Consequently, the value of Eq. (3) decreases, indicating a higher likelihood that the subclusters corresponding to these two circles belong to the same cluster. Therefore, we define o as

$$o_{C_i,C_j} = \frac{dist(c_i, c_j)}{r_{C_i} + r_{C_j}}. \tag{4}$$

Although this idea is simple, subsequent experimental results show that it is effective. Further, we can optimize this equation in future work to improve the effectiveness of the algorithm.

The *sim* denotes a similarity metric between two subclusters, as shown in

$$\mathrm{sim}^{\beta}_{\mathrm{C_i,C_j}} = |s_{C_i} - s_{C_j}|^{\beta}, \tag{5}$$

Algorithm: FedHC

At each client $z \in [Z]$, run P-Method with $A^{(z)}$
and obtain some subclusters $centroids = \{C_1, ..., C_{k'}\}$.
Calculate the sets R, N and S corresponding to $centroids$ by Eq. (1), (2).
Upload $centroids$, R, N and S to server.
Then, at server, input k and β.
Initialize current number of clusters counter k_0 and the set of centroids T:
k_0 = size of $centroids$, $T = \{c_1, ..., c_{k_0}\}$;
while $k_0 > k$:
    Calculate d_{C_i,C_j} between each subcluster by Eq. (3) - (5) ;
    Select C_a, C_b by Eq. (10) and merge them as a new cluster by Eq. (11);
    Update relevant parameters of the new cluster by Eq. (6) - (9) ;
    $k_0 = k_0$ - 1;
    $T = \{c_1, ..., c_{k_0}\}$;
end while
Output: centroids set $T = \{c_1, ..., c_k\}$.

where we currently use the difference between the standard deviations of the two subclusters(s_{C_i} and s_{C_j} represent the standard deviation of the two subclusters respectively.). The parameter β is used to control the influence of the similarity metric between subclusters on the calculation of the distance coefficient. The value of β should be greater than or equal to 0, the larger β is, the greater the influence of sim in calculating d. It should be noted that if β is too large, the value of d will reflect the similarity between two subclusters, neglecting the distance and degree of overlap between centroids. Conversely, if β is too small, the sim will approach 1, meaning that d will reflect a special distance calculated based on the distance and degree of overlap between centroids. So, the value of β usually starts at 1 and is then fine-tuned according to datasets.

When merging two subclusters, some parameters of the new clusters obtained after merging need to be calculated. Calculating the size(denoted as n) and standard deviation(denoted as s) of the new cluster is relatively straightforward, we can directly use

$$n' = n_{C_1} + n_{C_2} \tag{6}$$

and

$$s' = \sqrt{\frac{(n_{C_1} - 1)s_{C_1}^2 + (n_{C_2} - 1)s_{C_2}^2}{n_{C_1} + n_{C_2} - 2}}. \tag{7}$$

Here, n represents the number of data points in the cluster.

However, calculating the radius(denoted as r) and centroid(denoted as c) of the new cluster is more complex because, on the server side, we do not have the specific values of the data points within these subclusters. Therefore, we can only estimate their radius and centroid. We estimate the radius and centroid of the new cluster based on a weighted approach, as described in

$$c' = \frac{n_{C_1}}{(n_{C_1} + n_{C_2})}c_1 + \frac{n_{C_2}}{(n_{C_1} + n_{C_2})}c_2 \tag{8}$$

and

$$r' = \frac{n_{C_1}}{n_{C_1} + n_{C_2}}[dist(c', c_1) + r_{C_1}] + \frac{n_{C_2}}{n_{C_1} + n_{C_2}}[dist(c', c_2) + r_{C_2}], \tag{9}$$

where the weight assignment depends on the size of the cluster. Here, $dist(a, b)$ in Eq. (9) represents the centroid distance from cluster a to cluster b.

After the definition of these equations, we design a subcluster merging method MS. The process description of the MS algorithm running on the server is as follows: Firstly, we compute the total number of clusters k_0 at all clients and initialize the centroids set T. Then, we enter a loop to merge subclusters. In each iteration of the loop, we calculate pairwise distances between all subclusters using Eq. (3) - (5) (where distances mentioned in this paper are all computed using Euclidean distance). We then select two subclusters C_a, C_b with the smallest distance by Eq. (3) and

$$a, b = \underset{1 \leq i,j \leq k_0, i \neq j}{\operatorname{argmin}} d_{C_i, C_j}, \tag{10}$$

k_0 is the number of centroids at the server. Then merging them as a new cluster by

$$C' = C_a \cup C_b. \tag{11}$$

After merging, we compute the centroid, radius, size, and standard deviation of the new cluster by Eq. (6) - (9), and update the corresponding values for the next merging operation.

After $k_0 - k$ iterations, we obtain the centroids set T corresponding to the k clusters at the server. Finally, the centroids set T can be transmitted back to the clients to determine the assignment of each data point at the client to a specific cluster.

3.4 Discussion

By avoiding the occurrence of the "uniform effect", FedHC results in relatively good clustering performance for imbalanced datasets. We use the algorithm PNS [14] as the P-Method, this allows us to adaptively partition the data into several small clusters of similar size at each client and there is no need to know the value of k in advance, we consider the similarity between subclusters when merging, if the sizes of subclusters are similar, the similarity between subclusters belonging to different clusters will be significantly smaller, therefore, the PNS algorithm is a relatively suitable "P-Method". The computational cost of FedHC at the server is $O[(k_0 - k) * k_0^2]$, this is similar to the state-of-the-art method KFed. But the cost of FedHC at the client, cause using PNS as P-Mehtod, is higher than KFed, which uses KMeans at the client. Overall, the computational cost of our proposed method is higher than KFed, but the focus of our work is not to optimize the computational cost, we will improve this problem in the future. Moreover, our approach has limitations: 1) Joint learning among clients could be improved, as there is no iterative interaction between clients and server, which is a common drawback of existing methods; 2) Divisive and agglomerative clustering methods used by client and server respectively result in high computational costs. Yet, this paper prioritizes addressing issues with imbalanced data and clients lacking a reasonable k, rather than optimizing computational efficiency.

Table 1. Information of Datasets

Dataset	Feature	Data	Cluster size
ids2	2	3200	$\{2000, 400, 400, 200, 200\}$
gaussian	2	2000	$\{61, 1212, 606, 121\}$
ids2_22	2	400	$\{200, 200\}$
ids2_2k22	2	2400	$\{2000, 200, 200\}$
pageblock	10	5357	$\{4913, 329, 115\}$
yeast	6	1394	$\{463, 429, 244, 163, 51, 44\}$
abalone	7	1262	$\{115, 391, 689, 67\}$
breast	9	683	$\{444, 239\}$
iris	4	150	$\{50, 50, 50\}$
seed	7	210	$\{70, 70, 70\}$
wireless	7	2000	$\{500, 500, 500, 500\}$

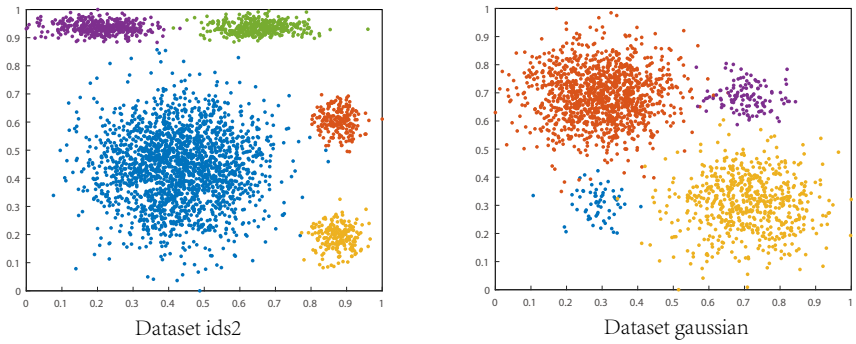

Fig. 2. Demonstration of two synthetic datasets.

4 Experiments

In this section, we mainly conducted the following experiments: 1) Clustering on IID imbalanced data, FedHC works better; 2) Clustering on IID balanced data, FedHC works best on synthetic datasets but ordinary on real datasets; 3) Clustering on Non-IID imbalanced data, FedHC works best.

4.1 Datasets and Compared Methods

To evaluate the clustering performance, we compared FedHC with four other federated clustering algorithms on four synthetic datasets and seven real datasets. Two of the synthetic datasets, *ids*2 and *gaussian*(Fig. 2), are generated from a mixture of bivariate Gaussian density functions as same as in [14], other two datasets, *ids*2_22 and *ids*2_2*k*22, are modified from *ids*2. The seven real datasets are sourced from the UCI Machine Learning Repository [15], which is a publicly available machine learning database. In the experiment, the features of all datasets were normalized to [0, 1]. The information of these datasets is listed in Table 1.

In the experiments, a total of four comparative methods were used. One of them is KFed [4], which is a state-of-the-art federated clustering method based on k-means. Additionally, due to the scarcity of existing federated clustering

Table 2. Result of the IID imbalanced data.

Dataset	Measurement	Ours	KFed	SF-AP	SF-MS	SF-KDPC
pageblock	F-measure	**0.9042**	0.8514	0.8877	0.8869	0.8219
	Accuracy	0.9601	0.9252	0.9985	**0.9989**	0.8391
	NMI	**0.1550**	0.0390	0.0891	0.0744	0.0202
	ARI	**0.2960**	0.0260	0.0312	0.0234	0.0502
	DCV	**0.1106**	0.1793	0.2103	0.2112	0.1837
yeast	F-measure	**0.5431**	0.4826	0.4657	0.4125	0.5229
	Accuracy	0.7288	0.5882	0.6819	**0.9620**	0.5911
	NMI	**0.2720**	0.2191	0.2102	0.1046	0.2535
	ARI	**0.2165**	0.1459	0.1166	0.0239	0.1918
	DCV	0.4891	0.2933	0.5242	1.5558	**0.0203**
abalone	F-measure	**0.6500**	0.5334	0.5461	0.5155	0.5632
	Accuracy	0.6395	0.5171	0.5103	**0.7124**	0.5380
	NMI	**0.2835**	0.2553	0.2703	0.1525	0.2735
	ARI	**0.2434**	0.1682	0.1849	0.0214	0.1922
	DCV	**0.2660**	0.6048	0.3272	0.3446	0.6363
breast	F-measure	**0.9680**	0.9580	0.7957	0.6888	0.9663
	Accuracy	**0.9678**	0.9582	0.8386	0.9444	0.9663
	NMI	**0.7985**	0.7368	0.3867	0.1437	0.7749
	ARI	**0.8744**	0.8426	0.3769	0.1002	0.8686
	DCV	0.0663	0.0352	0.5176	0.8324	**0.0041**
gaussian (synthetic)	F-measure	**0.9863**	0.8468	0.8212	0.7086	0.9181
	Accuracy	**0.9860**	0.8167	0.7775	0.6350	0.9013
	NMI	**0.9233**	0.6751	0.6745	0.5184	0.7646
	ARI	**0.9676**	0.6825	0.5710	0.4682	0.7761
	DCV	**0.0217**	0.3149	0.5901	0.5396	0.2614
ids2 (synthetic)	F-measure	**0.9922**	0.7708	0.7586	0.5950	0.7558
	Accuracy	**0.9922**	0.7440	0.7068	0.5544	0.6973
	NMI	**0.9619**	0.6488	0.7694	0.3145	0.7859
	ARI	**0.9779**	0.5550	0.5347	0.2916	0.5506
	DCV	**0.0209**	0.4341	0.6998	0.2495	0.6954
Avg. Rank		**1.2667**	3.2000	3.6000	4.3000	2.6333

methods, we separately developed three federated clustering methods for comparison in the experiments. The design principles of these three methods are similar, and they share the same configuration as our proposed FedHC (there is no number of clusters, usually marked as k, at the client, but there is a k on the server). All of them first perform local clustering at the client side using methods that do not require the k (specifically, affinity propagation algorithm, mean shift algorithm, and k-nearest neighbor-based DPC algorithm [13]), then upload several parameters to the server for global clustering using k-means. We respectively denote these three methods as SF-AP, SF-MS, and SF-KDPC.

4.2 Evaluation Metrics

For numerical comparison, we used five evaluation metrics to evaluate the clustering results: 1) F-measure; 2) Accuracy; 3) Normalized Mutual Information (NMI); 4) Adjusted Rand Index(ARI); and 5) DCV. The first four are common

Table 3. Results of the IID balanced data.

Dataset	Measurement	Ours	KFed	SF-AP	SF-MS	SF-KDPC
ids2_22 (balanced)	F-measure	**1.0000**	**1.0000**	0.6017	0.6656	**1.0000**
	Accuracy	**1.0000**	**1.0000**	0.6700	0.9950	**1.0000**
	NMI	**1.0000**	**1.0000**	0.0261	0.0235	**1.0000**
	ARI	**1.0000**	**1.0000**	0.0331	0.0001	**1.0000**
	DCV	**0.0000**	**0.0000**	0.3492	1.4001	**0.0000**
ids2_2k22 (imbalanced)	F-measure	**0.9959**	0.6942	0.6647	0.8963	0.7672
	Accuracy	**0.9958**	0.6414	0.6091	0.9288	0.7197
	NMI	**0.9575**	0.2944	0.3396	0.4945	0.4204
	ARI	**0.9817**	0.2418	0.1691	0.6309	0.2942
	DCV	**0.0108**	0.7460	0.8348	0.0361	0.6474
Avg. Rank		**1.0000**	2.6000	3.4000	2.3000	2.2000
iris (balanced)	F-measure	0.7620	0.7408	**0.8399**	0.7729	0.8260
	Accuracy	0.9467	0.7553	0.8400	**0.9867**	0.8267
	NMI	0.6818	0.5652	0.6659	**0.7220**	0.6550
	ARI	0.5509	0.4864	**0.6303**	0.5558	0.6109
	DCV	0.8502	0.3645	**0.0400**	0.9702	0.1200
seed (balanced)	F-measure	0.8849	0.7260	0.8787	0.6196	**0.9246**
	Accuracy	0.8857	0.7309	0.8810	0.6190	**0.9238**
	NMI	0.6670	0.4624	0.6512	0.3016	**0.7452**
	ARI	0.6958	0.4317	0.6872	0.2602	**0.7832**
	DCV	0.1116	0.2601	0.1116	0.1505	**0.1030**
wireless (balanced)	F-measure	0.9251	0.8306	0.9278	0.3977	**0.9451**
	Accuracy	0.9235	0.8327	0.9264	**0.9720**	0.9445
	NMI	0.8157	0.6492	0.8143	0.0816	**0.8378**
	ARI	0.8067	0.6370	0.8141	0.0021	**0.8586**
	DCV	0.1846	0.2140	0.1567	1.9250	**0.0798**
Avg. Rank		2.6667	4.2667	2.4000	3.8000	**1.6000**

evaluation metrics for clustering, while DCV [11] is designed specifically to evaluate clustering performance on imbalanced datasets, when the value of DCV is small, it does not necessarily indicate good clustering performance, but a large value often implies relatively poor clustering performance. In addition, we give an average ranking of each method based on its clustering results on different datasets.

4.3 Federated Clustering Performance Evaluation.

In the federated learning environment, the distribution of data across different clients is complex and diverse. It's not practical to design specific data distributions. To ensure a fair comparison, we shuffled the data and distributed it to each client uniformly for federated clustering experiments, that is, the data are independently and identically distributed(IID data). The experimental results are shown in Table 2, and the best and second best results are shown in boldface and underlined, respectively. We conducted experiments on four real datasets and two synthetic datasets. Considering that some algorithms have inherent randomness, such as KFed, whose clustering results are influenced by the initial cluster centroids, we repeated each experiment 10 times for each algorithm and

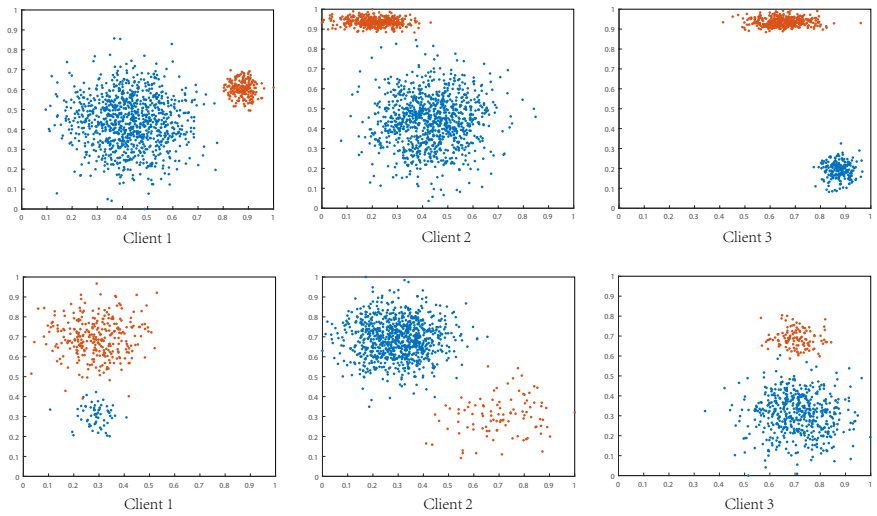

Fig. 4. Non-IID data of *Gaussian* across three clients.

averaged the results. From the experimental results, our algorithm achieved the best results in most metrics. The algorithm with the second-best performance was SF-KDPC, while the remaining three methods performed differently on different datasets. In other words, for federated clustering on imbalanced datasets, our method outperforms other comparative methods.

Although our goal is to explore federated clustering on imbalanced data, there are also many real-world scenarios where balanced data needs to be handled. Therefore, we cannot ignore the clustering performance of FedHC on balanced data. Accordingly, we conducted some experiments on balanced data, and the results are shown in Table 3. Looking at the experiments on three real balanced datasets (iris, seed, and wireless), SF-KDPC performs the best, followed by our method and SF-AP. Although our method does not achieve the best clustering performance on these three datasets, it still slightly outperforms the state-of-the-art method KFed. One possible reason for the poor results of our method on real balanced datasets compared to synthetic datasets is that data is divided into multiple subclusters at the client, but since the data is balanced, these subclusters are more similar to each other, and then when merging at the server, there is a possibility that two subclusters which do not belong to the same cluster may be incorrectly merged together due to their comparative similarity.

4.4 Performance Evaluation for Clients with Non-IID Distributions.

To test whether these methods can identify all clusters distributed across different clients on the server, we conducted a cluster absence experiment. In this experiment, each client only has data from a subset of clusters, that is, the data are Non-IID, representing a simple and intuitive scenario for federated clustering. We divided the synthetic datasets ids2 and Gaussian among three clients for experimentation, with each client having data points from only two clusters

Table 4. Result of the Non-IID imbalanced data.

Dataset	Measurement	Ours	KFed	SF-AP	SF-MS	SF-KDPC
ids2	F-measure	**0.9981**	0.9840	0.7177	0.9871	0.7696
	Accuracy	**0.9981**	0.9836	0.6806	0.9869	0.7163
	NMI	**0.9898**	0.9411	0.6799	0.9496	0.7770
	ARI	**0.9949**	0.9534	0.4571	0.9624	0.5467
	DCV	**0.0041**	0.0461	0.7686	0.0369	0.6942
gaussian	F-measure	**0.9975**	0.9873	0.8923	0.9921	0.8352
	Accuracy	**0.9975**	0.9870	0.8689	0.9920	0.7859
	NMI	**0.9850**	0.9410	0.7081	0.9584	0.7172
	ARI	**0.9947**	0.9742	0.7434	0.9835	0.6022
	DCV	**0.0055**	0.0234	0.2821	0.0141	0.5478
Avg. Rank		**1.0000**	3.0000	4.6000	2.0000	4.4000

in the dataset. The specific data distribution is illustrated in Fig. 3 and Fig. 4. The experimental results are summarized in Table 4. Our proposed FedHC performed the best in this experiment, followed by SF-MS, with KFed closely behind. The remaining two methods performed relatively poorly. This experiment demonstrates that our method can correctly identify clusters distributed across multiple clients, with performance comparable to the mainstream method KFed. Additionally, according to the DCV metric, the value corresponding to FedHC is significantly smaller than that of other methods, indicating that FedHC performs better in handling imbalanced data compared to other methods. Overall, from the experiments conducted, our proposed FedHC not only performs comparably to the state-of-the-art method KFed in clustering balanced datasets but, more importantly, it outperforms KFed and other compared methods in clustering imbalanced datasets. This indicates that our proposed method has achieved the intended goal.

4.5 Ablation Studies

We conducted a simple ablation study on data. The experiments on datasets $ids2\_22$ and $ids2\_2k22$ were conducted to demonstrate the difference in clustering performance of each method on balanced and imbalanced two-dimensional synthetic data. The relevant information for these two datasets is mentioned in Table 1, where $ids2\_22$ is a balanced dataset with only two clusters, and $ids2\_2k22$ is an imbalanced dataset formed by adding a relatively large cluster to the former. From the results in Table 3, it can be observed that on the balanced dataset $ids2\_22$, our method, KFed, and SF-KDPC exhibit better clustering performance. However, when the data becomes imbalanced, i.e., on the imbalanced dataset $ids2\_2k22$, only our method maintains good clustering performance and outperforms the other methods, making it the best-performing method among them. This suggests that our method performs better in handling imbalanced data compared to the other methods. This observation is consistent with the results of the previous experiments on imbalanced datasets, where FedHC also achieved the best results. In addition, ablation studies on methods will be further explored in future work.

5 Concluding Remarks

In this paper, we explore a new federated clustering framework called FedHC to cope with two close-to-reality but usually overlooked issues: (1) the number of "true" clusters is unknown by each client, and (2) the global cluster distributions that will be searched by the server is imbalanced. Accordingly, a micro-partition strategy is proposed to automatically partition data into small subclusters without requiring the number of clusters k to be given at each client. The subcluster distribution information is then efficiently summarized to avoid privacy leakage and passed to the server. The server is designed to hierarchically merge the subclusters by working on their abstracted information. The above design effectively circumvents the "uniform effect" of partitional clustering approaches and is thus competent in exploring imbalanced clusters. Through several experiments, is demonstrated that the proposed FedHC is effective in handling imbalanced data even for the more challenging case that each client is with non-IID data.

It is also worth noting that some aspects of FedHC still need improvement, such as its dependency on cluster number k to be known by the server, and potential performance degradation when dealing with high-dimensional or large-scale data. In summary, we have taken a step forward in exploring federated clustering on imbalanced data and have seen promising results, indicating that this area deserves further in-depth research in future work.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under grant 62172112, the National Key Research and Development Program of China under grant 2022YFE0112200, and the Guangdong Provincial Key Laboratory of Intellectual Property and Big Data under grant 2018B030322016.

References

1. Briggs, C., Fan, Z., Andras, P.: Federated learning with hierarchical clustering of local updates to improve training on non-iid data. In: International Joint Conference on Neural Networks. pp. 1–9. IEEE (2020)
2. Carreira-Perpinán, M.A.: A review of mean-shift algorithms for clustering. arXiv preprint arXiv:1503.00687 (2015)
3. Cheung, Y.m., Zhang, Y.: Fast and accurate hierarchical clustering based on growing multilayer topology training. IEEE Transactions on Neural Networks and Learning Systems **30**(3), 876–890 (2019)
4. Dennis, D.K., Li, T., Smith, V.: Heterogeneity for the win: One-shot federated clustering. In: International Conference on Machine Learning. pp. 2611–2620. PMLR (2021)
5. Frey, B.J., Dueck, D.: Clustering by passing messages between data points. science **315**(5814), 972–976 (2007)
6. Ghosh, A., Chung, J., Yin, D., Ramchandran, K.: An efficient framework for clustered federated learning. Advances in Neural Information Processing Systems **33**, 19586–19597 (2020)

7. Konečnỳ, J., McMahan, H.B., Ramage, D., Richtárik, P.: Federated optimization: Distributed machine learning for on-device intelligence. arXiv preprint arXiv:1610.02527 (2016)
8. Kumar, H.H., Karthik, V., Nair, M.K.: Federated k-means clustering: A novel edge ai based approach for privacy preservation. In: IEEE International Conference on Cloud Computing in Emerging Markets. pp. 52–56. IEEE (2020)
9. Li, S., Hou, S., Buyukates, B., Avestimehr, S.: Secure federated clustering. arXiv preprint arXiv:2205.15564 (2022)
10. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine **37**(3), 50–60 (2020)
11. Liang, J., Bai, L., Dang, C., Cao, F.: The k-means-type algorithms versus imbalanced data distributions. IEEE Transactions on Fuzzy Systems **20**(4), 728–745 (2012)
12. Lin, W.C., Tsai, C.F., Hu, Y.H., Jhang, J.S.: Clustering-based undersampling in class-imbalanced data. Information Sciences **409**, 17–26 (2017)
13. Long, Z., Gao, Y., Meng, H., Yao, Y., Li, T.: Clustering based on local density peaks and graph cut. Information Sciences **600**, 263–286 (2022)
14. Lu, Y., Cheung, Y.M., Tang, Y.Y.: Self-adaptive multiprototype-based competitive learning approach: A k-means-type algorithm for imbalanced data clustering. IEEE Transactions on Cybernetics **51**(3), 1598–1612 (2019)
15. Markelle Kelly, Rachel Longjohn, K.N.: The uci machine learning repository, https://archive.ics.uci.edu
16. Pedrycz, W.: Federated fcm: clustering under privacy requirements. IEEE Transactions on Fuzzy Systems **30**(8), 3384–3388 (2021)
17. Sattler, F., Müller, K.R., Samek, W.: Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. IEEE Transactions on Neural Networks and Learning Systems **32**(8), 3710–3722 (2020)
18. Schubert, E., Sander, J., Ester, M., Kriegel, H.P., Xu, X.: Dbscan revisited, revisited: why and how you should (still) use dbscan. ACM Transactions on Database Systems (TODS) **42**(3), 1–21 (2017)
19. Xie, S., Wu, Y., Liao, K., Chen, L., Liu, C., Shen, H., Tang, M., Sun, L.: Fed-sc: One-shot federated subspace clustering over high-dimensional data. In: 2023 IEEE 39th International Conference on Data Engineering (ICDE). pp. 2905–2918. IEEE (2023)
20. Xiong, H., Wu, J., Chen, J.: K-means clustering versus validation measures: A data-distribution perspective. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **39**(2), 318–331 (2009)
21. Zhang, Y., Cheung, Y.m.: A fast hierarchical clustering approach based on partition and merging scheme. In: 2018 Tenth International Conference on Advanced Computational Intelligence (ICACI). pp. 846–851 (2018)
22. Zhang, Y., Cheung, Y.m., Liu, Y.: Quality preserved data summarization for fast hierarchical clustering. In: 2016 International Joint Conference on Neural Networks (IJCNN). pp. 4139–4146 (2016)
23. Zhao, M., Zhang, Y., Ji, Y., Lu, Y.: Unsupervised concept drift detection via imbalanced cluster discriminator learning. In: Liu, Q., Wang, H., Ma, Z., Zheng, W., Zha, H., Chen, X., Wang, L., Ji, R. (eds.) Pattern Recognition and Computer Vision. pp. 31–43. Springer Nature Singapore, Singapore (2024)