

Federated K-Means Clustering

Garst, Swier; Reinders, Marcel

DOI

[10.1007/978-3-031-78166-7_8](https://doi.org/10.1007/978-3-031-78166-7_8)

Publication date

2025

Document Version

Final published version

Published in

Pattern Recognition - 27th International Conference, ICPR 2024, Proceedings

Citation (APA)

Garst, S., & Reinders, M. (2025). Federated K-Means Clustering. In A. Antonacopoulos, S. Chaudhuri, R. Chellappa, C.-L. Liu, S. Bhattacharya, & U. Pal (Eds.), *Pattern Recognition - 27th International Conference, ICPR 2024, Proceedings* (pp. 107-122). (Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics); Vol. 15302 LNCS). Springer. https://doi.org/10.1007/978-3-031-78166-7_8

Important note

To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright

Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy

Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

Green Open Access added to TU Delft Institutional Repository

'You share, we take care!' - Taverne project

<https://www.openaccess.nl/en/you-share-we-take-care>

Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public.



Federated K-Means Clustering

Swier Garst^(✉)  and Marcel Reinders 

Delft University of Technology, Delft, The Netherlands
{S.J.F.Garst,M.J.T.Reinders}@tudelft.nl

Abstract. Federated learning is a technique that enables the use of distributed datasets for machine learning purposes without requiring data to be pooled, thereby better preserving privacy and ownership of the data. While supervised FL research has grown substantially over the last years, unsupervised FL methods remain scarce. This work introduces an algorithm which implements K-means clustering in a federated manner, addressing the challenges of varying number of clusters between centers, as well as convergence on less separable datasets.

Keywords: Federated Learning · K-Means clustering · Distributed machine learning

1 Introduction

Nowadays, lots of data is being generated in a distributed fashion. Mobile phones and other personal devices such as smart watches enable the collection of massive amounts of data. If made accessible, this data could prove useful for improving the performance of the services provided by these devices. However, due to a growing concern on data privacy, more and more users of these devices are hesitant in sharing their data. Furthermore, regulations such as the General Data Protection and Regulation (GDPR) act prevent the collection of data of this kind in bulk. Federated learning (FL) ([15]) was introduced as a solution to this problem. In short, instead of pooling data to train a single model, instances of a model are being shared to data owners (clients), which then train the model on their local data. Then, these trained models are sent back to the central server, which aggregates the results. Next, a new round begins with the server sending out the updated models. This cycle continues until convergence.

Over the past couple of years, research has shown FL to be a promising technique, reaching performances comparable to a central approach in which all data of the clients is pooled at a single location [11, 17]. The vast majority of the federated learning research has been focusing on the supervised learning paradigm. Little work has been done on unsupervised federated learning methods, even less so when specifically looking into clustering techniques [12, 13]. One of these

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-78166-7_8.

clustering techniques is k-means clustering [7]. In a federated learning setting, k-means clustering can be described as trying to find overarching cluster means over data which is distributed among different datasets (clients).

Prior work has been done on creating federated k-means clustering algorithms [5, 8, 10, 14, 18]. We focus on solving the issue of a variable amount of local clusters. When data distributions between clients differ, it is likely that not all clients share data from all global clusters. When this is the case, the amount of clusters present per client can differ. This complicates federated clustering in two ways. First, matching cluster means between clients becomes less straightforward, as there is no one-to-one matching anymore. Second, when the local k does not correspond to the global k , the problem of finding an optimal k scales linearly with the amount of clients. Furthermore, when clients hold only part of the data, it can be hard to distinguish between outliers and samples of a different cluster, without the knowledge of the data distribution of other clients. Altogether, this makes manually determining a value for k on each client locally complicated, if not infeasible without loss of performance.

We propose an iterative federated k-means clustering algorithm (FKM) that automatically determines the local value for k . By iteratively aggregating local cluster means, and running k-means locally on all clients in parallel, we are able to create a clustering that in many cases corresponds to a central clustering, i.e. the k-means clustering that would occur if all data was pooled together. By pruning empty global clusters on local clients, we are able to deal with a variable amount of local clusters between clients, without having to set the values for k locally on each client.

2 Related Work

Since its inception in 2017, federated learning has been applied in various scenarios. The most well known taxonomy of federated learning systems is the split between the cross-device and cross-silo settings [12]. In brief, in the cross-device setting, many, in the order of thousands or more, devices are connected. However their connection is usually unstable, and cross-device federations will have to deal with users dropping out or joining later throughout the process. This setting is mostly applicable to federations with end-user devices, such as mobile phones. On the other hand, in the cross-silo setting, there are usually only a couple to tens of clients. These clients are usually quite stable and can be assumed to be connected from the beginning of the process all the way to the end. This setting is more applicable to health care centers or companies learning a shared model without sharing their data due to privacy or competition reasons.

Different challenges emerge based on whether the federation is cross-device or cross-silo and as such different algorithms have emerged for either setting. For the cross-device setting, Liu et al. introduced a method for federated k-means in a setting where each client is seen as a single sample [14]. More specifically, they tackle the issue of proactive caching in cellular networks, i.e. trying to predict what data to keep local based on popularity. Their method is robust

against user change, and assumes a tiered architecture with base stations and sub stations, as can be found in next-generation cellular networks. Kumar et al. also propose a federated k-means clustering algorithm that can be applied for the cross-device setting [10]. Their method assumes a central dataset available at the server, on which a k-means clustering is pre-trained. It is then distributed across all clients who update the clustering based on their local data, after which the clustering is aggregated again on the server. Finally, Hou et al. created a k-means clustering algorithm based on homomorphic encryption and blockchain [8]. Although their method is not explicitly applied to the cross-device setting, it does share encrypted versions of the data, which is often still unacceptable for many cross-silo use-cases.

For the cross-silo setting, Servetnyk et al. proposed a federated k-means clustering algorithm based on dual averaging and self-organising maps [18]. Although their algorithm is capable of dealing with heterogeneous data, they do not explicitly address the challenge of a variable local k , or cluster alignment. Finally, Dennis et al. propose a one-shot federated k-means clustering algorithm, which only needs one local clustering, as well as one global aggregation step [5]. As we focus mostly on the cross-silo setting, our algorithms bears the most resemblance to the method by Dennis et al. (kFed). The key differences are that our method does not require setting a local value for k on each client, which, as we argue in the introduction, can be difficult if not impossible without loss of performance. This also allows us to have a different k for each client, further improving performance. Finally, we show that, for less separable datasets, there is a substantial performance gain in iterating between the local data and global aggregation more than once, as is done by Dennis et al.

Table 1. Notations used

symbol	description
X_i	data on client i
K_g	global number of clusters
K_i	number of clusters on client i
C_g	global cluster means
C_i	cluster means of client i
M	total amount (sum) of local clusters
S_i	amount of samples for each cluster on client i
N	total amount of clients

3 Methods

Notation used throughout this section is found in Table 1. The pseudocode for our proposed federated k-means algorithm (FKM) can be found in Algorithm 1.

Algorithm 1. The federated kmeans algorithm**Input:** K_g

```

1: Init:
2: on each client  $i \in N$  do:
3:    $K_i = K_g$ 
4:    $S_i, C_i = \text{kmeans++\_init}(X_i, K_i)$   $\triangleright$  get cluster means using kmeans++
      initialization
5:   send  $S_i, C_i$  to server
6: For each round  $r$  do:
7:   On server do:
8:      $C_l = [C_1|C_2|\dots|C_M]$   $\triangleright$  Concatenate all local cluster means
9:      $S_l = [S_1|S_2|\dots|S_M]$   $\triangleright$  Repeat for sample amounts per cluster
10:     $C_g = \text{kmeans}(C_l, K_g, \text{weights} = S_l)$   $\triangleright$  Obtain new global clusters using kmeans
11:    send  $C_g$  to all clients
12:   On each client  $i \in N$  do:
13:      $C_i = C_g$ 
14:      $S_i = \text{kmeans\_assign}(X_i, C_i)$   $\triangleright$  Determine empty clusters
15:      $C_i = C_i[s \neq 0 \text{ for } s \text{ in } S_i]$   $\triangleright$  Drop empty global clusters
16:      $K_i = \text{size}(C_i)$ 
17:      $S_i, C_i = \text{kmeans}(X_i, K_i, \text{init} = C_i)$   $\triangleright$  run kmeans from remaining global cluster
      means
18:   send  $S_i, C_i$  to server

```

The algorithm can be divided into two parts: an initialization step, in which we generate initial cluster means on each client using k-means++ initialization [1], and an iterative k-means step in which clients communicate their cluster means to the server, which aggregates these means into a ‘global’ set of means, which then gets redistributed to the clients for the next k-means iteration. See supplement A for background on k-means and k-means++.

3.1 Determining the Amount of Local Clusters

While the global amount of clusters is set (main parameter k of the k-means procedure), it is not a given that each client has data for each of these clusters. In other words, the number of clusters between clients can differ, and is not necessarily equal to the number of clusters in the pooled data. In order to solve this problem, we determine which global clusters correspond to local data in each round on each client. Before a client applies a new k-means step locally, it assigns its data to the global cluster means it has received (line 14). Next, clients check if there are empty clusters, i.e. cluster means which did not get any points assigned to them. If so, clients discard these empty clusters (line 15). The remaining (global) cluster means are then used as initialization for the next local k-means step (line 16). This way, k can locally become smaller when running k-means on the clients. Since this pruning step only happens after global aggregation, we guarantee that the discarded clusters are indeed corresponding to clusters on other clients.

3.2 Cluster Alignment

After each client has calculated one iteration of k-means (not until convergence, to avoid local minima) on their local data (each with their own amount of local clusters), they send their cluster means as well as the amount of samples per cluster back to the server. The server then concatenates all cluster means, and aggregates them. It does so by running a k-means clustering on the received local means until convergence (using the global k parameter), to align clusters from different clients to each other. This global k-means is weighted by the amount of samples per cluster found, such that a cluster with lots of samples in it will have a bigger impact on the aggregation step compared to a cluster with fewer samples. That is, we modify the k-means objective function (see supplement A for the original) into:

$$F_{km} = \sum_{j=0}^M \min_{C_i \in C_g} (S_j ||C_j - C_i||^2) \quad (1)$$

where S_j is the amount of samples corresponding to local cluster C_j . Note that the *cluster means* sent back by the clients are at the server used as the *samples* for clustering using k-means. Doing the aggregation with a k-means clustering, we solve the cluster alignment problem, since similar clusters will be close to each other and thus merged by the global k-means step.

Because the amount of samples have to be reported to the central server, there exists a privacy risk if a client finds a cluster with only one sample in it. To prevent this, any clusters holding less than p samples (we used $p = 2$ throughout this work) are simply omitted from the list of means sent to the server.

4 Results

We compared our federated k-means (FKM) with a k-means clustering that is executed on all data centrally, as well as to one-shot the method of Dennis et al. [5]. Our first set of experiments is on simulated data, such that ground truth labels of the cluster centers is known. We therefore calculate the Adjusted Rand Index (ARI) for both central and federated approaches with respect to the labelled samples data. Since there are no labels for the clustering in the FEMNIST experiment, the silhouette score was used instead. In some cases, we added an “informed” setting for Dennis et al., in which we set K_l such as to achieve the highest ARI score by exhaustive search. In all other cases, we run their method using $K_l = K_g$, as the ARI score is only available when ground truth labels are known, which is not always the case.

4.1 Clients Holding Different Parts of the Data

In order to validate the FKM algorithm, a synthetic two-dimensional dataset was generated. The generation procedure is taken from Servetnyk et al. [18]. Sixteen

cluster centers were chosen with an equal distance (here 5) from one another, see Fig. 1a. Then, 50 data points were sampled around each cluster center using a normal distribution (with variance 1). This data was then distributed among four clients in the following way: First, each client is assigned a ‘location’ within the field ($X_1, X_2 \in (-12.5, 12.5)$). From there, the probability P that a data point would be assigned to a certain client scales inversely with the euclidean distance d to that datapoint:

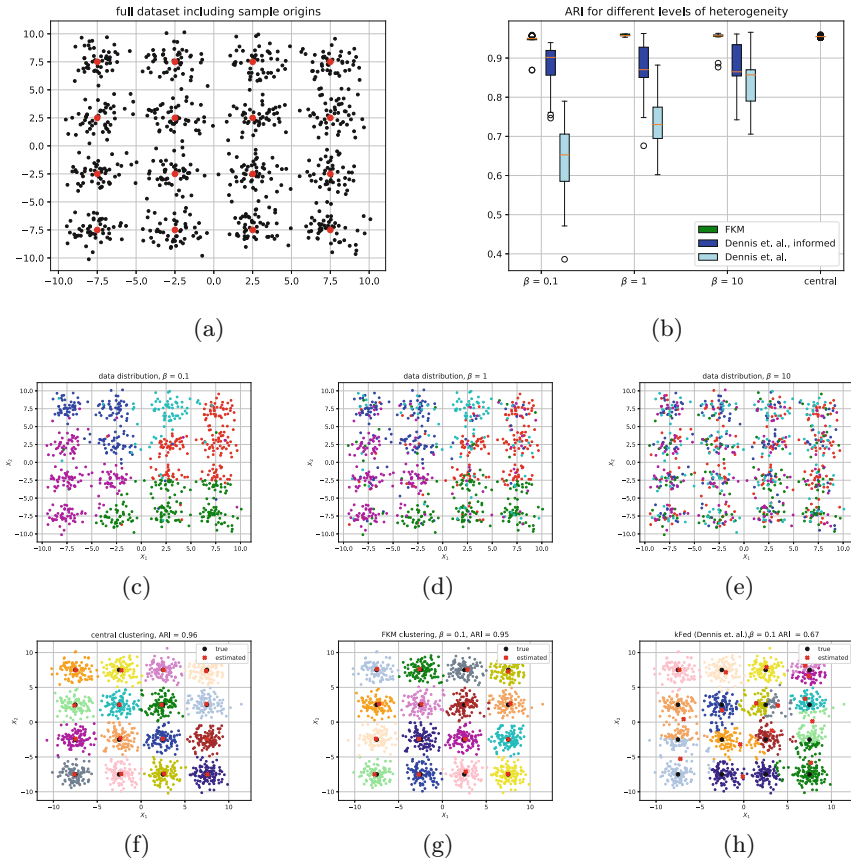


Fig. 1. The regular synthetic datasets. (a) shows the original sampling of the regular synthetic dataset, with the defined cluster means (from which the data are generated using a normal distribution $N(0,1)$) in red. (b) shows ARI results on all three datasets. (c) until (e) shows how this dataset is distributed over five different clients using different values of β . Different colors indicate the different clients. (f) to (h) show examples of a clustering on (c) as given by a centralized k-means, FKM (ours), and kFed (Dennis et al.), respectively. Note that different colors in the last three plots denote different cluster assignments instead of different clients. (Color figure online)

$$P = 1 - \exp\left(-\frac{\beta}{d}\right) \quad (2)$$

where β is a parameter which can be tuned to promote more or less heterogeneity in the data separation. Differing from [18], if a data point happens to be assigned to multiple clients, it instead gets assigned at random.

We wanted to explore the influence of data heterogeneity, i.e. a varying amount of clusters per client. To do so, we generated three versions of this dataset, with $\beta = 0.1, 1, 10$. See Fig. 1 c-e for the final distributions. Note that β only changes which points get assigned to which client, meaning that it does not influence the performance for the central case. Figure 1b shows that our method is able to attain performance similar to a centralized k-means clustering, while outperforming Dennis et al., regardless of tuning of the K_l parameter. Performance of our FKM approach seems to be independent of β (in contrast to the method of Dennis et al.), meaning that our algorithm is robust to having varying cluster amounts between clients.

4.2 Increasing Levels of Noise

Next, we explored the effect of having noisier clusters. We recreated the regular synthetic dataset, but varied the standard deviation from which samples are being generated, from 1 to 1.5 (original used 1). Figure 2 shows the effect. We generated these datasets twice, once with 50 points per cluster and once with 200 points per cluster.

Results on these datasets are shown in Fig. 3. For both central and federated clustering, the ARI scores go down for higher noise levels. This is expected, as

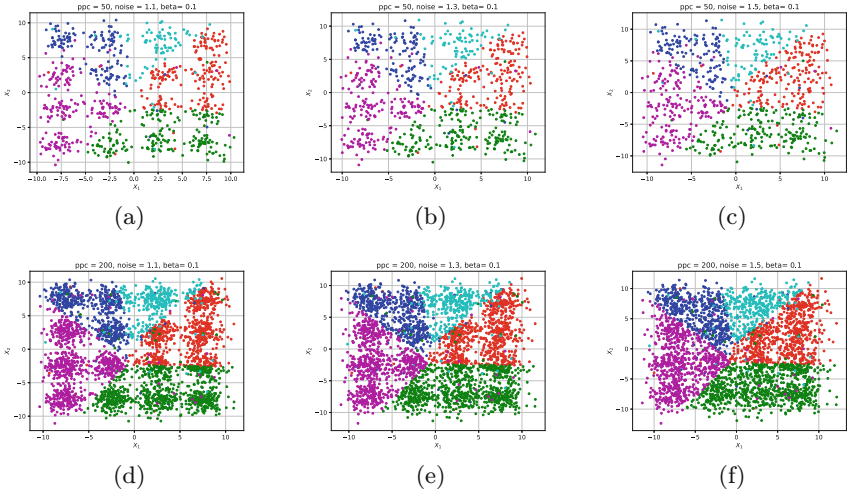


Fig. 2. Some of the data distributions of the simulated datasets with increasing levels of noise (columns), using 50 or 200 points per cluster (rows).

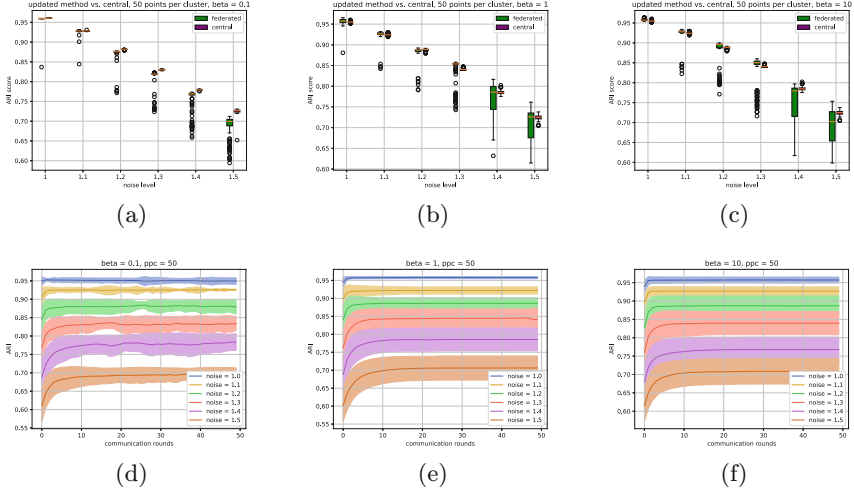


Fig. 3. Clustering results on the synthetic dataset when using different levels of noise for different values of β . (a) to (c) show the final ARI scores for $\beta = 0.1$, 1 and 10, respectively. (d) to (f) show how the ARI score for FKM converges over time, each corresponding to the figure above it.

there will be more points ending up closer to the cluster they did not originally belong to, meaning that even if kmeans finds the original cluster means perfectly, the label assignment will be off. Therefore, the relative difference between federated and central clustering is more important than the absolute ARI scores. Our method attains a similar average performance; however, variance seems to increase compared to centralized clustering. Furthermore, for $\beta = 0.1$, mean ARI decreases compared to central clustering at high noise levels, meaning that a setting with high noise as well as high cluster variability is still a hard challenge for our federated k-means algorithm.

Regardless, performance does seem to increase significantly as compared to the method of Dennis et al. [5]. This can partly be due to our ability to iterate. Figure 3a to 3b shows that, especially for noisier datasets, there is a large benefit in being able to iterate more often. The amount of points per cluster does not seem to influence ARI score significantly, see supplement B.

4.3 Ablation Study

To explore the importance of several parts of our algorithm, we perform an ablation study on the two dimensional synthetic data introduced in Sect. 4.1 with a noise parameter of 1. We make five separate ablations, as well as one setting in which all five modifications are included:

- **Retain empty clusters:** For this ablation, we skip the step where we prune the empty clusters, effectively fixing K_l equal to K_g in all clients.

- **Initialization:** Instead of initializing local cluster using k-means++, we initialize local clusters at random.
- **100 local iterations:** Instead of performing one iteration of k-means locally, we perform 100 iterations.
- **Unweighted aggregation:** When aggregating cluster means on the server, instead of weighting the cluster means by the amount of samples corresponding to said cluster locally, we simply give every cluster a weight of 1 (Note that we explored several values for the amount of local iterations, however we saw little difference between those values, so for sake of brevity we report only the results using the largest value that we explored (100)).

We considered the same values of β as earlier described in Sect. 4.1, i.e. $\beta \in \{0.1, 1, 10\}$, as shown in Fig. 4. This figure shows that the degree to which different parameters are important depend on β , i.e. the heterogeneity of the data. For $\beta = 0.1$ the largest impact can be seen for doing weighted aggregation and local pruning. When data is homogeneously distributed ($\beta = 10$), i.e. every client has data for each cluster, the proposed ablations seem to have little impact on the algorithm. In fact, increasing the amount of local iterations could even be beneficial in a completely heterogeneous case. This is in line with literature on supervised federated learning, where increasing the amount of local epochs can increase performance [15]. However, under heterogeneous circumstances, in supervised federated learning, clients might move too far into local optima before aggregation, decreasing performance with more local iterations [20]. We therefore hypothesize that in even more heterogeneous circumstances, a lower amount of local iterations could still be beneficial for our method as well.

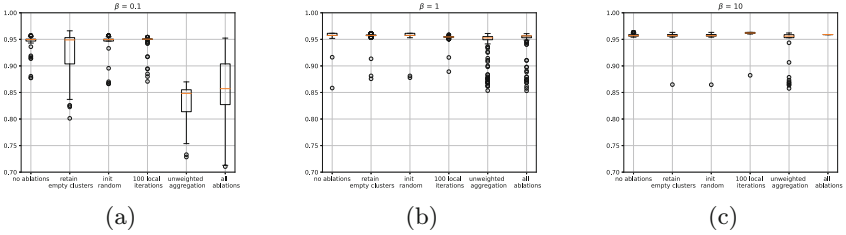


Fig. 4. Results of the ablation study. (a), (b) and (c) show results for $\beta = 0.1, 1$ and 10 , respectively. (a) shows that, for a highly heterogeneous dataset, both dropping empty local clusters and especially weighted aggregation have a large impact on model performance. However, as the data becomes more homogeneously distributed, (subfigures (b)–(c)), these factors become less important.

4.4 High Variability in Number of Local Clusters

Next we wanted to explore the effect of having an even more variable local k . We used the same data as generated for the regular synthetic dataset, but distributed

even more heterogeneously, such that each client only had data from 1, 4, 7, 10 or 16 clusters, respectively. See Fig. 5a.

Figure 5b shows that our method attains a similar average performance as compared to the central case, however with a larger variation. This is probably caused by differences in initializations. If the algorithm initializes in such a way that clients assign data to more clusters than what is being present in their data, the algorithm has a hard time correcting for that. Furthermore, it does not help that one client only has ten datapoints in total, meaning it initializes ten clusters of size one, of which none are being send to the central server due to privacy issues. Regardless, our method does outperform the algorithm from Dennis et al. [5]. This is likely due to our algorithm’s ability to change the value of k for its local k -means step between clients.

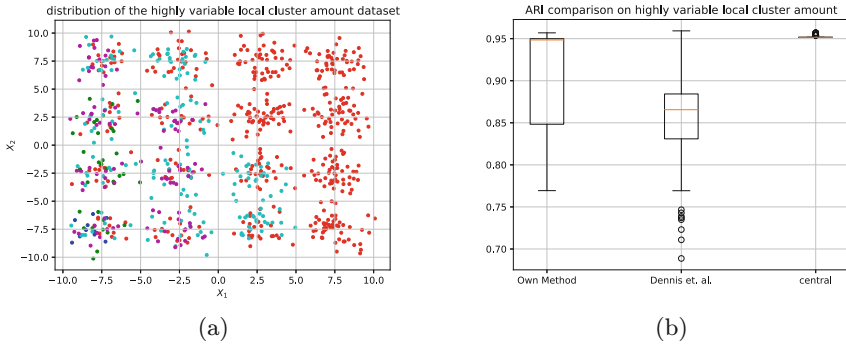


Fig. 5. Assessment of the method on data with a large variability of local clusters per client. (a) shows the distribution per client, (b) the ARI results for different methods.

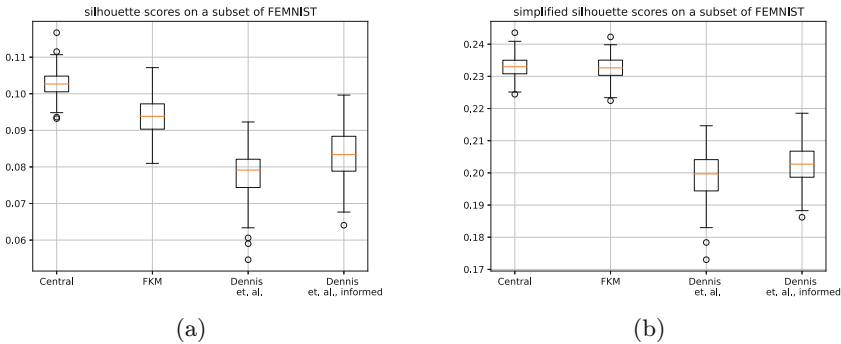


Fig. 6. Results on (a subset of) FEMNIST. (a) shows the silhouette score, (b) the simplified silhouette score.

4.5 Clustering Higher Dimensional Real Data

So far, all our experiments have been done on two dimensional, simulated data. For many use cases, however, data has a much higher dimensionality. In order to determine performance on a higher dimensional dataset, the Federated Extended MNIST (FEMNIST) from LEAF ([3]) (having a dimensionality of 784) was used, which separates the original Extended MNIST ([4]) handwritten numbers and letters based on the person who wrote them. FEMNIST has a dimensionality of 784. This leaves approximately 110 datapoints per client; see supplement C for the distribution. Only 10 clients were used from the original FEMNIST, as this drastically sped up the experiments, while keeping enough data for a meaningful assessment. We set $k = 60$, in line with earlier experiments from Dennis et al. [5]. Figure 6a shows that our method outperforms both settings of the method from Dennis et al. There is still a difference with a central clustering, however. This could be due to the relatively small amount of samples per client compared to the amount of dimensions, decreasing the quality of the local clusters.

The FEMNIST experiments use the silhouette score [16] as their performance metric. The silhouette score involves calculating distances from each point in a dataset to each other point in a dataset. This means that, to calculate a ‘global’ silhouette score, distances between datapoints from different clients need to be determined, something that can not be done in a straightforward federated manner. In our case, the simulated federated environment made it possible to calculate the silhouette score for evaluation purposes. In a real-life setting, the simplified silhouette score ([9]) could be a suitable alternative, as it only calculates distances between datapoints and cluster means, something which can be done on all clients separately.

We compare the simplified silhouette score with the silhouette score from the same experiments in Fig. 6b. There seems to be a high correlation between the two scores for a given method, which is in line with previous work [19].

4.6 Clustering Real Biological Data

Finally, we explore a common clustering task in bioinformatics, that of cell-type identification. We use data from Bouland et al. [2], specifically their dataset referred to as ‘four cancers’. This dataset consists of single-cell RNA sequence measurements from 12 different cancer patients with one of four cancer diagnoses: Lung, endo, colon and renal cancer. In total, there are 22815 genes (features) measured in 132549 cells (samples). These samples are either from tumor tissue or from normal tissue, adjacent to tumour tissue.

Before distributing the dataset over separate clients, we run a standard single-cell pipeline protocol¹ using the Seurat R library [6]. Briefly, we first filter out genes that have less than 200 or more than 2500 feature counts. We then log-normalize the data. Afterwards, we run the Seurat function “FindVariableFeatures” to find the top 500 genes with most variance. Then, we select only the

¹ https://satijalab.org/seurat/articles/pbmc3k_tutorial.html.

samples from tumor tissue, leaving a total of 68905 samples. Finally, we perform dimensionality reduction on these samples using the selected 500 genes, to reduce dimensionality to 5 (determined using the elbow method).

Data is grouped by patient, meaning that all data from a single patient will end up on a single client. Data is being distributed over three clients. Two different distributions of the data are considered, denoted as IID (Identically and Independently Distributed) or non-IID. For the IID-distribution, each client gets data from every cancer type, whereas for the non-IID data, each client gets data from only one cancer type, as well as from two out of six lung patients.

We run FKM, as well as central clustering and kFed (the method by Dennis et al. [5]), with a (global) K of 4, equivalent to the amount of cancers in the dataset. Figure 7 shows the silhouette and simplified silhouette scores for all algorithms. In the IID setting, our method slightly outperforms kFed, while there is still a slight gap with a centralized clustering. However, when considering the non-IID setting, we observe that the performance gap between FKM and kFed increases, whereas the difference between FKM and a centralized clustering is similar compared to the IID setting, indicating that FKM is more robust to various data distributions.

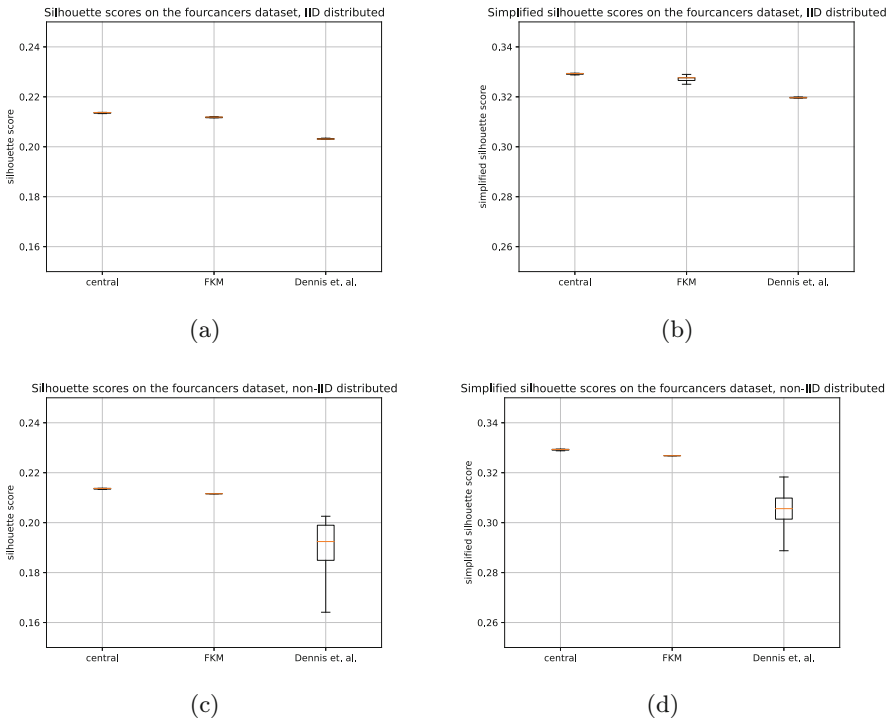


Fig. 7. Silhouette and simplified silhouette scores on the fourcancers dataset. Both the silhouette score in (a) as well as the simplified silhouette score in (b) show increased performance of FKM compared to kFed, though still slightly underperforming a centralized clustering. However, when the data is distributed non-IID (figure (c) and (d)), the gap between the federated methods seems to increase.

5 Discussion and Conclusion

This work describes the implementation and validation of a federated k-means clustering algorithm (FKM), enabling clustering over multiple datasets without sharing the underlying data. Our results show performances close to a central method, in which all data is brought into a single location. There are still some scenarios in which our method shows larger variability in performance as compared to a central clustering, however. These are mostly the more difficult scenarios, such as when there is an extreme distribution in the amount of cluster present on each client, or when the data has a high dimensionality as with the FEMNIST experiment. Assessment of our method on more heterogeneous and 'real life' datasets is therefore an important direction for future work. Nevertheless, FKM has shown to be a promising method in finding similarities among distributed datasets without the need of sharing any data.

6 Code Availability

The code to run FKM, as well as all experiments and generate the figures used throughout this manuscript, can be found at: <https://github.com/swiergarst/fedKMeans/>.

A Background on K-Means and K-Means++

A.1 K-Means Clustering

The objective of a clustering algorithm is to partition a given dataset into several subsets with similar features. The k-means clustering algorithm does so by trying to minimize the within cluster sum-of-squares criterion:

$$F_{km} = \sum_{j=0}^m \min_{C_i \in C} (||X_j - C_i||^2) \quad (3)$$

with m the amount of samples, C_i the cluster mean of cluster i , C the set of all cluster means and $X_{j,k}$ being data point j assigned to cluster k . The procedure in which the k-means algorithm tries to minimize Eq. 3 consists of two steps. First, all data points get assigned to the cluster mean according to the lowest euclidean distance. Then, the mean center point from all points assigned to a certain cluster is calculated. This is done for every cluster, creating a new set of means to start the next round with. This process is repeated until the change within these means is smaller than a certain threshold (and the algorithm has reached convergence) ([7]).

A.2 K-Means++

One of the drawbacks of classical k-means clustering is that its initialization is sampled uniformly from the underlying data. This means that having initial

cluster means that all come from the same cluster is as probable as having initial cluster means spread across all clusters. Although the K-means algorithm itself can somewhat compensate for this, it still leads to large variability in performance. Arthur and Vassilvitskii developed an initialization method for K-Means to combat this high variability, called k-means++ [1]. Instead of sampling K cluster means from the data with uniform probability, datapoints get weighted based on their distance to the closest already mean that is chosen, with larger distances giving larger weights. This results in (on average) initializations that are more distributed over the space, and prevents (on average) initial cluster means from starting very close to each other, decreasing k-means performance.

B Extra Results on Increasing Amount of Points per Cluster

See Fig. 8.

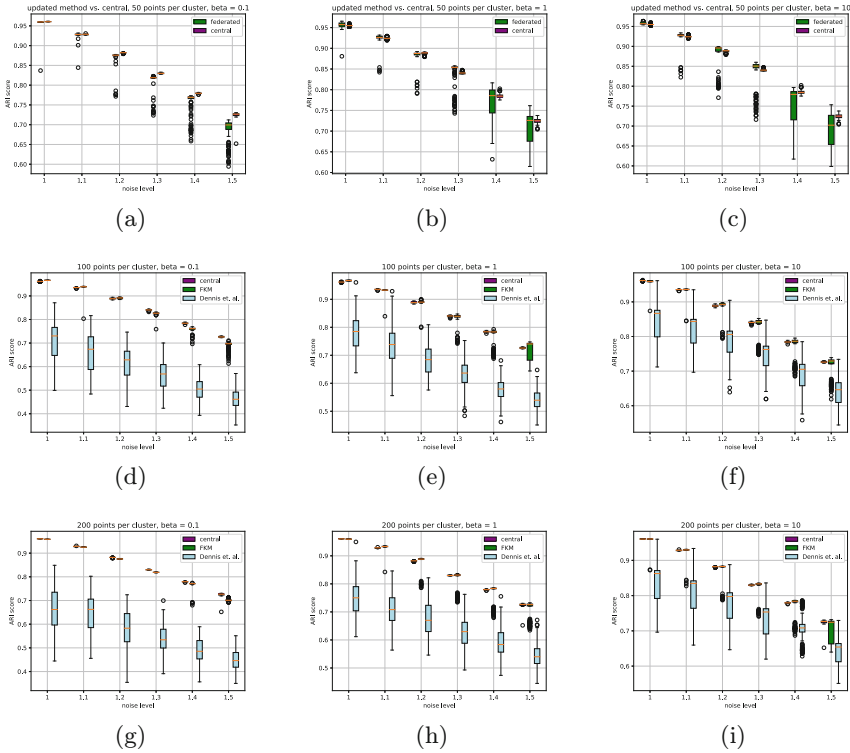


Fig. 8. results on using different levels of noise for different values of β , with differing amounts of points per cluster. From left to right, the columns correspond to $\beta = 0.1$, 1 and 10 respectively. From top to bottom, the rows correspond to 50, 100, and 200 points per cluster.

C FEMNIST Distribution

See Fig. 9.

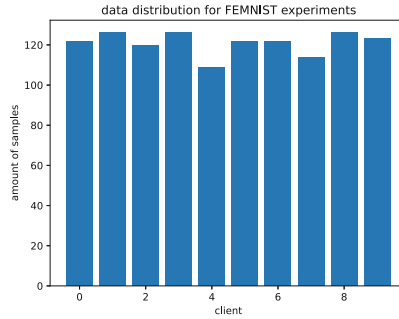


Fig. 9. Sample distribution for the FEMNIST dataset

References

1. Arthur, D., Vassilvitskii, S.: k-means++: the advantages of careful seeding. Technical Report 2006-13, Stanford InfoLab (2006). <http://ilpubs.stanford.edu:8090/778/>
2. Bouland, G.A., Mahfouz, A., Reinders, M.J.T.: Differential analysis of binarized single-cell RNA sequencing data captures biological variation. *NAR Genom. Bioinform.* **3**(4), lqab118 (2021). <https://doi.org/10.1093/nargab/lqab118>
3. Caldas, S., et al.: LEAF: a benchmark for federated settings. *CoRR* abs/1812.01097 (2018). <http://arxiv.org/abs/1812.01097>
4. Cohen, G., Afshar, S., Tapson, J., van Schaik, A.: EMNIST: an extension of MNIST to handwritten letters. *CoRR* abs/1702.05373 (2017). <http://arxiv.org/abs/1702.05373>
5. Dennis, D.K., Li, T., Smith, V.: Heterogeneity for the win: one-shot federated clustering. In: Meila, M., Zhang, T. (eds.) *Proceedings of the 38th International Conference on Machine Learning*. *Proceedings of Machine Learning Research*, vol. 139, pp. 2611–2620. PMLR (2021). <https://proceedings.mlr.press/v139/dennis21a.html>
6. Hao, Y., et al.: Dictionary learning for integrative, multimodal and scalable single-cell analysis. *Nat. Biotechnol.* (2023). <https://doi.org/10.1038/s41587-023-01767-y>
7. Hartigan, J., Wong, M.: Algorithm as 136: a k-means clustering algorithm. *J. Roy. Stat. Soc. Ser. C (Appl. Stat.)* **28**, 100–108 (1979)
8. Hou, R., Tang, F., Liang, S., Ling, G.: Multi-party verifiable privacy-preserving federated k-means clustering in outsourced environment. *Secur. Commun. Networks* **2021**(1), 3630312 (2021)
9. Hruschka, E., de Castro, L., Campello, R.: Evolutionary algorithms for clustering gene-expression data. In: *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pp. 403–406 (2004). <https://doi.org/10.1109/ICDM.2004.10073>

10. Kumar, H.H., Karthik, V.R., Nair, M.K.: Federated K-means clustering: a novel edge AI based approach for privacy preservation; federated k-means clustering: a novel edge AI based approach for privacy preservation. In: 2020 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM) (2020). <https://doi.org/10.1109/CCEM50674.2020.00021>
11. Lee, G.H., Shin, S.Y.: Federated learning on clinical benchmark data: performance assessment. *J. Med. Internet Res.* **22**(10) (2020). <https://doi.org/10.2196/20891>
12. Li, Q., et al.: A survey on federated learning systems: vision, hype and reality for data privacy and protection. *IEEE Tran. Knowl. Data Eng.* 1–1 (2021). <https://doi.org/10.1109/TKDE.2021.3124599>
13. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: challenges, methods, and future directions. *IEEE Signal Process. Mag.* **37**(3), 50–60 (2020)
14. Liu, Y., Ma, Z., Yan, Z., Wang, Z., Liu, X., Ma, J.: Privacy-preserving federated k-means for proactive caching in next generation cellular networks. *Inf. Sci.* **521**, 14–31 (2020). <https://doi.org/10.1016/J.INS.2020.02.042>
15. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.V.: Communication-efficient learning of deep networks from decentralized data. In: Singh, A., Zhu, J. (eds.) *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research*, vol. 54, pp. 1273–1282. PMLR (2017). <https://proceedings.mlr.press/v54/mcmahan17a.html>
16. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **20**, 53–65 (1987). [https://doi.org/10.1016/0377-0427\(87\)90125-7](https://doi.org/10.1016/0377-0427(87)90125-7). <https://www.sciencedirect.com/science/article/pii/0377042787901257>
17. Sadilek, A., et al.: Privacy-first health research with federated learning. *NPJ Digit. medicine* **4**(1), 132 (2021). <https://doi.org/10.1038/s41746-021-00489-2>, <http://www.ncbi.nlm.nih.gov/pubmed/34493770>, <http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC8423792>
18. Servetnyk, M., Fung, C.C., Han, Z.: Unsupervised federated learning for unbalanced data. *GLOBECOM 2020 - 2020 IEEE Global Communications Conference* (2020). <https://doi.org/10.1109/GLOBECOM42002.2020.9348203>
19. Wang, F., Franco-Penya, H.-H., Kelleher, J.D., Pugh, J., Ross, R.: An analysis of the application of simplified silhouette to the evaluation of k -means clustering validity. In: Perner, P. (ed.) *MLDM 2017. LNCS (LNAI)*, vol. 10358, pp. 291–305. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-62416-7_21
20. Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., Chandra, V.: Federated learning with non-IID data. *CoRR abs/1806.00582* (2018). <http://arxiv.org/abs/1806.00582>