Heterogeneity for the Win: One-Shot Federated Clustering

Don Kurian Dennis [1] **Tian Li** [1] **Virginia Smith** [1]

Abstract

In this work, we explore the unique challenges—and opportunities—of unsupervised federated learning (FL). We develop and analyze a one-shot federated clustering scheme, k-FED, based on the widely-used Lloyd's method for k-means clustering. In contrast to many supervised problems, we show that the issue of statistical heterogeneity in federated networks can in fact benefit our analysis. We analyse k-FED under a center separation assumption and compare it to the best known requirements of its centralized counterpart. Our analysis shows that in heterogeneous regimes where the number of clusters per device (k') is smaller than the total number of clusters over the network k, $(k' \leqslant \sqrt{k})$, we can use heterogeneity to our advantage—significantly weakening the cluster separation requirements for k-FED. From a practical viewpoint, k-FED also has many desirable properties: it requires only one round of communication, can run asynchronously, and can handle partial participation or node/network failures. We motivate our analysis with experiments on common FL benchmarks, and highlight the practical utility of one-shot clustering through use-cases in personalized FL and device sampling.

1. Introduction

Federated learning (FL) aims to perform machine learning over large, heterogeneous networks of devices such as mobile phones or wearables (McMahan et al., 2017). While significant attention has been given to the problem of supervised learning in such settings, the problem of unsupervised federated learning has been relatively unexplored (Kairouz et al., 2019). In this work, we show that unsupervised learning presents unique opportunities for FL, specifically for the task of clustering data that resides in a federated network.

Clustering is a crucial first step in many learning tasks. In the case of federated learning, clustering has found applications in client-selection (Cho et al., 2020), personalization (Ghosh et al., 2020) and exploratory data analysis. While many works have explored techniques for distributed clustering (Section 2), most do not take into account the unique challenges of federated learning, such as statistical heterogeneity, systems heterogeneity, and stringent communication constraints (Li et al., 2020a)[1]. These challenges can complicate analyses, reduce efficiency, and lead to practical issues with stragglers and device failures. In this work, we study communication-efficient distributed clustering in settings where the data is non-identically distributed across the network (i.e., heterogeneous), and devices can join and leave the network abruptly. For such settings, we develop and analyse a one-shot clustering scheme, k-FED, based on the classical Lloyd's heuristic (Lloyd, 1982) for clustering.

The method we propose, k-FED, requires only one round of communication with a central server. Each device, indexed by z, solves a local $k^{(z)}$-means problem and then communicates its local cluster means via a message of size $O(dk^{(z)})$. As we show in Section 3, this allows for device failures, only requiring that there are enough devices available in the network such that k target clusters exist in the data. Moreover, it is possible to cluster points in previously unavailable devices via a simple recomputation at the central server.

Beyond the practical benefits of k-FED, our work is unique in rigorously demonstrating a problem setting where possible benefits of statistical heterogeneity exist for federated learning. In particular, in supervised learning, many works have highlighted detrimental effects of statistical heterogeneity, observing that heterogeneity can lead to poor convergence for federated optimization methods (McMahan et al., 2017; Li et al., 2020b), result in unfair models (Mohri et al., 2019), or necessitate novel forms of personalization (Smith et al., 2017; Mansour et al., 2020). In contrast to these works, we show that for the specific notion of heterogeneity considered herein (provided in Definition 3.2 and motivated by the application of clustering), heterogeneity can in fact have measurable benefits for our approach.

[1]Carnegie Mellon University, Pittsburgh, PA, USA. Correspondence to: Don Dennis <dondennis@cmu.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

[1]Privacy, while an important concern for many federated applications, is not the main focus of our work. However, a possible benefit of the one-shot nature of k-FED is that it requires significantly fewer messages to be shared over the network relative to standard iterative techniques such as distributed k-means.

More specifically, similar to many works in clustering (Kumar & Kannan (2010); Awasthi & Sheffet (2012) and references therein), we analyse k-FED under a center-separation assumption; that is, we assume that the mean of the clusters are well separated. We also consider a specific notion of heterogeneity: given a target clustering with k clusters that we wish to recover from the data, we assume that each device contains data from only $k' \leqslant \sqrt{k}$ of these target clusters. For instance, for clustering data generated by a mixture of k well separated Gaussians, we assume that each device contains data from $k' \leqslant \sqrt{k}$ component Gaussians. In this regime, we show that our separation requirement is similar to that of the centralized counterpart. Further, while the centralized setting requires all pairs of cluster centers to satisfy a $\Omega(\sqrt{k})$ center separation requirement, the federated approach can handle a large fraction of cluster pairs only satisfying a weaker $\Omega(k^{\frac{1}{4}})$ separation requirement. This is the first result we are aware of that analyzes the benefits of heterogeneity in the context of federated clustering.

Contributions. We propose and analyze a one-shot communication scheme for federated clustering. Our proposed method, k-FED, addresses common practical concerns in federated settings, such as high communication costs, stragglers, and device failures. Theoretically, we show that k-FED performs similarly to centralized clustering in regimes where each device only has data from at most $\sqrt{k}$ clusters with a similar $\Omega(\sqrt{k})$ center separation requirement. Moreover, in contrast to the centralized setting, we show that a large number of cluster pairs need only a $\Omega(k^{\frac{1}{4}})$ weaker separation assumption in heterogeneous networks, thus allowing a broader class of problems to be solved in this setting compared with centralized clustering. We demonstrate our method through experiments on common FL benchmarks, and explore the applicability of k-FED to problems in personalized federated learning and device sampling. Our work highlights that heterogeneity can have distinct benefits for a subset of problems in federated learning.

2. Background and Related Work

Centralized Clustering. Clustering is one of the most widely-used unsupervised learning tasks, and has been extensively studied in both centralized and distributed settings. Although a variety of clustering methods exist, Lloyd's heuristic (Lloyd, 1982) remains popular due in part to its simplicity. In Lloyd's method, we start with an initial set of k centers. We then assign each point to its nearest center and reassign the centers to be the mean of all the points assigned to it, continuing this process till termination. While it is easy to show that this method terminates, it is also known that this process can take superpolynomial time to converge (Arthur & Vassilvitskii, 2006). However, under suitable assumptions and careful choice of the initial centers, it can be shown to converge in polynomial time (Arthur & Vassilvitskii, 2006; Ostrovsky et al., 2013; Kumar & Kannan, 2010; Awasthi & Sheffet, 2012).

The method we propose, k-FED (Section 3.2), is a simple, communication-efficient distributed variant of these classical techniques. k-FED runs a variant of Lloyd's method for k-means clustering locally on each device, and then performs one round of communication to aggregate and assign clusters. Our work builds on the analysis of a variant of Lloyd's algorithm developed by Kumar & Kannan (2010) and later improved in Awasthi & Sheffet (2012) for the problem of clustering data from mixture distributions and other related results (e.g., McSherry, 2001; Ostrovsky et al., 2013). These works develop a deterministic framework with no generative assumptions on the data. Our analysis follows this framework and does not make any generative assumptions on the data.

Parallel and Distributed Clustering. Many works have explored parallel or distributed implementations of centralized clustering techniques (Dhillon & Modha, 2002; Tasoulis & Vrahatis, 2004; Datta et al., 2005; Bahmani et al., 2012; Xu et al., 1999). Unlike the one-shot communication scheme explored herein, these methods are typically direct parallel implementations of methods such as Lloyd's heuristic or DBSCAN (Ester et al., 1996), and require numerous rounds of communication. Another line of work has considered communication-efficient distributed clustering variants that require only one or two rounds of communication (e.g., Kargupta et al., 2001; Januzaj et al., 2004; Feldman et al., 2012; Balcan et al., 2013; Bateni et al., 2014; Bachem et al., 2018). These works are mostly empirical, in that there are no provable guarantees on the approximation quality of the distributed schemes; the works of Balcan et al. (2013); Bateni et al. (2014); Bachem et al. (2018) differ by providing communication-efficient distributed coreset methods for clustering, along with provable approximation guarantees. However, these works do not explore the federated setting or potential benefits of heterogeneity.

Federated Clustering. Several works have explored clustering in the context of supervised FL as a way to better model non-IID data (Smith et al., 2017; Ghosh et al., 2019; 2020; Sattler et al., 2020). These works differ from our own by clustering specifically in terms of devices, focusing on the downstream supervised learning task, and using either iterative (Smith et al., 2017; Ghosh et al., 2020; Sattler et al., 2020) or centralized (Ghosh et al., 2019) clustering schemes. Though not the main of focus of our work, in Section 4 we demonstrate the applicability of one-shot clustering by showing how k-FED can be used as a simple pre-processing step to deliver personalized federated learning—achieving similar or superior performance relative to the recent iterative approach for clustered FL proposed in Ghosh et al. (2020).

More recently, a distributed matrix factorization based clustering approach was explored in Wang & Chang (2020) for the purposes of unsupervised learning. However, while the authors consider the impact of statistical heterogeneity on their convergence guarantees, the focus is not on one-shot clustering or on showing distinct benefits of heterogeneity in their analyses.

3. k-FED: Preliminaries and Main Results

In this section, we begin by discussing some preliminaries and existing results in clustering related to Lloyd-type methods. In Section 3.1, we present the deterministic framework of Awasthi & Sheffet (2012) for centralized clustering, which we build upon. We present our method k-FED and state our theoretical results in Section 3.2. We provide detailed proofs in Appendix A.

3.1. Centralized k-means

In the standard (centralized) k-means problem, we are given a matrix $A \in \mathbb{R}^{n\times d}$ where each row A_i is a data point in $\mathbb{R}^d$. We are also given a fixed positive integer $k \leqslant n$, and our objective is to partition the data points into k disjoint partitions, $\mathcal{T} = (T_1, \ldots, T_k)$, so as to minimize the k-means cost:

$$\phi(\mathcal{T}) = \sum_{j=1}^{k} \sum_{i\in T_j} \|A_i - \mu(T_j)\|_2^2 . \tag{1}$$

Here we use $\mu(S)$ as an operator to indicate the mean of the points indexed by S, i.e., $\mu(S) = \frac{1}{|S|}\sum_{i\in S} A_i$. To ease notation, we simplify this as $\mu_r := \mu(T_r)$, when T_r is unambiguous.

While the k-means problem as stated here does not specify any generative model for the data points A_i, a popular setting to consider is when the data is sampled from a mixture of k-distributions in d-dimensions ($k \ll d$). For instance, we could imagine the data points as being sampled from a mixture of k Gaussian distributions. This generative model also introduces a notion of a *target clustering*, $\mathcal{T} = (T_1, \ldots, T_k)$ where the set T_i contains all points generated by the i-th component distribution. Many distribution dependent results are known for the problem of clustering distributions (see Kumar & Kannan (2010)). In general, they can be stated as: If the means of the distributions are poly(k) standard deviations apart, then we can cluster the data in polynomial time. Kumar & Kannan (2010) introduce a deterministic (distribution independent) framework that encompasses many of these known results. This work was later simplified and improved by Awasthi & Sheffet (2012). We state the main results of this framework here, after stating the notation we use. We emphasis that in our analysis we make no assumptions on how the data is generated; all relevant quantities only depend on the provided data.

Algorithm 1 Local $k^{(z)}$-means (Awasthi & Sheffet, 2012)

1: **Input:** On device indexed by z, the matrix of data points $A^{(z)}$, integer $k^{(z)}$;
2: Project $A^{(z)}$ onto the subspace spanned by the top $k^{(z)}$ singular vectors to get $\hat{A}^{(z)}$. Run any standard 10-approximation algorithm on the projected data and estimate $k^{(z)}$ centers $(\nu_1, \nu_2, \ldots, \nu_{k^{(z)}})$.
3: Set

$$S_r \leftarrow \{i : \|\hat{A}_i^{(z)} - \nu_r\|_2 \leqslant \frac{1}{3}\|\hat{A}_i^{(z)} - \nu_s\|_2, \text{ for every } s\}$$

and $\theta_r^{(z)} \leftarrow \mu(S_r)$
4: Run Lloyd steps until convergence

$$U_r^{(z)} \leftarrow \{i : \|A_i^{(z)} - \theta_r^{(z)}\|_2 \leqslant \|A_i^{(z)} - \theta_s^{(z)}\|_2, \forall s\}$$

and $\theta_r^{(z)} \leftarrow \mu(U_r^{(z)})$.
5: **Return:** Cluster assignments $(U_1^{(z)}, U_2^{(z)}, \ldots, U_{k^{(z)}}^{(z)})$ and their means $\Theta^{(z)} = (\theta_1^{(z)}, \ldots, \theta_{k^{(z)}})$.

Notation. We now introduce several definitions and notations that will be used throughout the paper. Let $\|A\|$ denote the spectral norm of a matrix A, defined as $\|A\| = \max_{u:\|u\|_2=1}\|Au\|_2$, and let $\|A_i\|_2$ denote the ℓ_2 norm of a vector A_i. For consistency, we index individual rows of A with i and j. Moreover, when a target clustering $T_1, \ldots, T_k$ is fixed, we index clusters with r, s, e.g., A_r is the matrix of points indexed by T_r. For notational convenience, we let $c(A_i)$ to denote the cluster index for data point A_i such that, $A_i \in T_{c(A_i)}$. For some set of points M, and another point say x, let $d_M(x)$ denote the distance of x to the set M, defined as $d_M(x) = \min_{y\in M}\|x - y\|_2$. Finally, let C be a $n \times d$ matrix with each row $C_i = \mu_{c(A_i)}$. For cluster T_r with $n_r = |T_r|$, we define

$$\tilde{\Delta}_r := \sqrt{k}\frac{\|A - C\|}{\sqrt{n_r}}. \tag{2}$$

Here the quantity $\|A - C\|/\sqrt{n_r}$ can be thought of as a deterministic analogue of the standard deviation; it measures the maximum average variance along any direction. Thus instead of reasoning about the separation between two clusters T_r and T_s in terms of the standard deviation, we will use $(\tilde{\Delta}_r + \tilde{\Delta}_s)$. In particular, we say that the two clusters T_r and T_s are *well separated* if for large enough constant c, their means satisfy:

$$\|\mu_r - \mu_s\|_2 \geqslant c(\tilde{\Delta}_r + \tilde{\Delta}_s) . \tag{3}$$

Again, we can interpret this as saying that two clusters are well separated if their means are c-standard-deviations apart.[2] Using the center separation assumption in (3),

[2] Any $c \geqslant 100$ is sufficient for our arguments (see Lemma 5).

Awasthi & Sheffet (2012) show that for a target clustering $T_1, T_2, \ldots, T_k$ satisfying the separation assumption, the variant of Lloyd's algorithm presented in Algorithm-1 when applied to the centralized clustering problem correctly clusters all but a small fraction of the data points. We state their result formally in Lemma 1, but before that we define a *proximity condition*, that will be used to precisely characterize the misclassified points.

Definition 3.1. *A point A_i for some $i \in T_s$ is said to satisfy the proximity condition, if for every $r \neq s$, the projection of A_i onto the line connecting μ_r and μ_s, denoted by $\bar{A}_i$ satisfies*

$$\left\|\bar{A}_i - \mu_r\right\|_2 - \left\|\bar{A}_i - \mu_s\right\|_2 \geqslant \left(\frac{1}{\sqrt{n_r}} + \frac{1}{\sqrt{n_s}}\right)\|A - C\|.$$

Thus a point A_i for $i \in T_s$ satisfies the proximity condition if its projection on the line connecting μ_r and μ_s is closer to μ_s by $\|A - C\|(\frac{1}{\sqrt{n_r}} + \frac{1}{\sqrt{n_s}})$. We refer to points that do not satisfy the proximity condition as 'bad points'. We now state the main result from Awasthi & Sheffet (2012) in the following lemma.

Lemma 1 (Awasthi-Sheffet, 2011)**.** *Let $\mathcal{T} = (T_1, \ldots, T_k)$ be the target clustering. Assume that each pair of clusters T_r and T_s are well separated. Then, after step 2 of Algorithm-1, for every r, it holds that $\|\mu(S_r) - \mu_r\|_2 \leqslant \frac{25}{c}\frac{1}{\sqrt{n_r}}\|A - C\|$. Moreover, if the number of bad points is ϵn, then (a) the clustering $\{U_1, U_2, \ldots, U_k\}$ misclassifies no more than $(\epsilon + O(1)c^{-4})n$ points and (b) $\epsilon < O((c - \frac{1}{\sqrt{k}})^{-2})$. Finally, if $\epsilon = 0$ then all points are correctly assigned.*

When we say misclassify, we mean with respect to $\mathcal{T}$ and up to a permutation of labels. Lemma 1 tells us that the cluster means, $\mu(S_r)$, are not very far away from the target cluster means, μ_r. Note that there are no distribution dependent terms in this statement; all relevant quantities are defined in terms of the data matrix A and $\mathcal{T}$.

3.2. k-FED: Method and Main Result

We now turn our attention to clustering data in a federated network. In our setting, we assume that all the devices in the network can communicate with a central server. Our clustering method k-FED, described in Algorithm 2, can be thought of as working in two stages. In the first stage, each device solves a local clustering subproblem and computes the cluster means for this subproblem. In the second stage, the central server accumulates and aggregates the results to compute the final clustering.

Notation. Let A be an $n \times d$ data matrix of all the data points in our network. We index individual devices by $z \in [Z]$ and thus, we denote the data-matrix for any particular device by $A^{(z)} \in \mathbb{R}^{n^{(z)} \times d}$, where $n^{(z)}$ is the number of data points on the device. Let $n_{\min} = \min_z n^{(z)}$. Note that $A^{(z)}$ is some subset of rows of A. Let $\mathcal{T} = (T_1, \ldots, T_k)$ be a clustering of all the data, referred to as a target clustering. For a fixed $\mathcal{T}$, let $\mathcal{T}^{(z)} = (T_1^{(z)}, T_2^{(z)}, \ldots, T_k^{(z)})$ be subsets of our target clustering that reside on a device z. Note that some $T_r^{(z)}$ could be empty. Let $k^{(z)}$ be the number of non-empty subsets on device z and let $k' = \max_z k^{(z)}$. Our notion of heterogeneity is formally defined based on the value of k', as described below.

Definition 3.2 (Heterogeneity of Clustering). *In the context of clustering, we say that a federated network with sufficient data is heterogeneous if $k' \leqslant \sqrt{k}$. The lower the ratio between k' and $\sqrt{k}$, the more heterogeneity exists in the network.*

Intuitively, this definition of heterogeneity states that—in contrast to the data from the k total clusters being partitioned in an IID fashion across the network—the data are partitioned in an non-IID fashion, such that only data from a small number of clusters (at most k') exists on each device. Such non-IID partitioning is reasonable to expect in heterogeneous federated networks with a large number of clusters, since the distribution of data on each device may differ, and it is not possible to actively re-distribute data across the network. For instance, consider identifying interests of mobile phone users based on the interaction data on an application. Here the interaction data is generated by the user on their particular device, and will reflect the tastes of individual. While the total number of 'tastes' (clusters) over the entire network could be quite large, a typical user will be interested in only a small number of them. With this definition in mind, we next describe our one-shot clustering method, k-FED, and analyze it in heterogeneous regimes.

Method Description. Similar to the centralized case (Section 3.1), let $C^{(z)}$ be a $n^{(z)} \times d$ matrix of the local cluster means, i.e. of $\mathcal{T}^{(z)}$. Consider a non-empty susbset $T_r^{(z)}$ of cluster T_r on some device and let $n_r^{(z)} = |T_r^{(z)}|$. We assume that there is a constant $m_0 > 1$, such that $n_r^{(z)} \geqslant \frac{1}{m_0} n_r$ for all r. We will use this quantity to ensure that individual devices have 'enough' points. Let,

$$\Delta_r = k' \frac{\|A - C\|}{\sqrt{n_r}}, \quad \text{and} \quad \lambda = \sqrt{k'}\left(\frac{\|A - C\|}{\sqrt{n_{\min}}}\right). \quad (4)$$

In the first step of k-FED (Algorithm-2), each (available) device $z \in [Z]$ runs Algorithm-1 locally and solves a local clustering problem with their local dataset $A^{(z)}$ and parameter $k^{(z)}$. We assume that $k^{(z)}$ is known. This stage outputs *device cluster centers* $\Theta^{(z)} = (\theta_1^{(z)}, \ldots, \theta_{k^{(z)}}^{(z)})$ and cluster assignments, $U_1^{(z)}, \ldots, U_{k^{(z)}}^{(z)}$ for each device z. At this stage, note that even though each device has classified its own points into clusters, we do not yet have a clustering for

Algorithm 2 k-FED

1: On each device $z \in [Z]$, run Algorithm-1 with local data $A^{(z)}$ and $k^{(z)}$ and obtain device cluster centers $\Theta^{(z)} = (\theta_1^{(z)}, \ldots, \theta_{k^{(z)}}^{(z)})$ at the central node.
2: Pick any $z \in [Z]$ and let $M \leftarrow \Theta^{(z)}$.
3: **repeat**
4: Let $\bar{\theta} \leftarrow \arg\max_{z\in[Z], i\in[k]} d_M(\theta_i^z)$. That is, the farthest $\theta_i^{(z)}$ from the set M.
5: $M \leftarrow M \cup \{\bar{\theta}\}$.
6: **until** there are k points in M, i.e. $|M| = k$
7: Run one round of Lloyd's heuristic to cluster points $\theta_i^{(z)}$, $z \in [Z], i \in [k]$ into k sets/clusters, $(\tau_1, \tau_2, \ldots, \tau_k)$. Use points in M as initial centers.
8: **Return:** the clustering $(\tau_1, \tau_2, \ldots, \tau_k)$ of the device cluster centers and the corresponding k-FED induced clustering (Definition 3.3).

points across devices. The central server attempts to create this clustering by aggregating the device cluster centers and separating them into k sets, $\tau_1, \ldots, \tau_k$. These sets *induce a clustering* of the data on the network as defined here:

Definition 3.3 (k-FED induced clustering)**.** *Let $\tau_1, \tau_2, \ldots, \tau_k$ be the clustering of device centers returned by Algorithm 2. Define,*

$$T'_r = \{i : A_i^{(z)} \in U_s^{(z)} \text{ and } \theta_s^{(z)} \in \tau_r, z \in [Z], s \in [k^{(z)}]\}.$$

Then, $\mathcal{T}' = (T'_1, \ldots, T'_k)$ form a disjoint partition of the entire data, called the k-FED induced clustering.

For our analysis comparing the quality of the k-FED induced clustering, $\mathcal{T}'$, to our target clustering $\mathcal{T}$, we require two different separation assumptions. We refer to them as *active* and *inactive* separation and introduce them through the following two definitions.

Definition 3.4 (Active/Inactive cluster pairs)**.** *A pair of clusters (T_r, T_s) are said to be an active pair if there exists at least one device that contains data points from both T_r and T_s. If no device has data points from both clusters T_r and T_s, we refer to the cluster pair (T_r, T_s) as an inactive pair.*

Definition 3.5. *We say that two clusters T_r and T_s satisfies the active separation requirement if, $\|\mu_r - \mu_s\|_2 \geqslant 2c\sqrt{m_0}(\Delta_r + \Delta_s)$, for some large enough constant c. Similarly, we say that they satisfy the inactive separation requirement if $\|\mu_r - \mu_s\|_2 \geqslant 10\sqrt{m_0}(\lambda_r + \lambda_s)$.*

Intuitively, these notions capture the difficulty in clustering two different types of clusters pairs—active and inactive cluster pairs. If no device has data from both T_r and T_s (i.e. an inactive pair), then the clustering sub-problems individual devices have to solve is easier since they never involve data from both of these clusters simultaneously. Thus the separation requirement for inactive cluster pairs is weaker than that for an active cluster pair. We now state our main theorem, which characterizes the performance of k-FED. We provide a detailed proof in Appendix A.

Theorem 3.1 (Main theorem)**.** *Let $\mathcal{T} = (T_1, T_2, \ldots, T_k)$ be a fixed target clustering of the data on a federated network. Let $m_0 > 1$ be such that, $|T_r^{(z)}| \geqslant \frac{1}{m_0}|T_r|$ for all r, s and for all $z \in [Z]$. Assume that each active cluster pairs T_r and T_s satisfy the active separation requirement, i.e.,*

$$\|\mu(T_r) - \mu(T_s)\|_2 \geqslant c\sqrt{m_0}(\Delta_r + \Delta_s).$$

Further, assume that for each inactive cluster pairs T_r, T_s,

$$\|\mu(T_r) - \mu(T_s)\|_2 \geqslant 10\sqrt{m_0}\lambda\,.$$

Then, at termination of k-FED all but $O(\frac{1}{c^2})n$ points are correctly classified. Moreover, if for each device z, the data points $A^{(z)}$ satisfy the proximity condition (Definition 3.1) for its local problem, then all points are classified correctly.

As before, by classified we mean that the clustering $\mathcal{T}'$ produced by k-FED and $\mathcal{T}$ agree on all but $O(\frac{1}{c^2})n$ points, up to permutation of labels of $\mathcal{T}$. Note that when $k' \approx k$, our active separation requirement is stricter than that required in centralized clustering ($\Omega(k)$ vs $\Omega(\sqrt{k})$). Further, as one would expect, as the number of points per cluster on each device decreases, the local clustering becomes harder. This is highlighted by our adverse dependency on $\sqrt{m_0}$.

However, in contrast to the general distributed learning framework where each device typically has a random subset of the data, the data residing on the devices in federated networks are typically generated locally and thus the partition of data among the devices is non-identically distributed. Specifically, in practice, the number of subsets of target clusters that reside on a device may be much smaller than the total number of clusters. Thus, as outlined in Definition 3.2, we look at the cases where $k' \leqslant \sqrt{k}$. Observe that in such settings, our active separation requirement reduces to that of the centralized k-means problem (with an additional $\sqrt{m_0}$ penalty) and our inactive separation requirement weakens to $k^{1/4}$. We state this formally in Corollary 1.1.

Corollary 1.1. *Assuming $k' \leqslant \sqrt{k}$, an active cluster pair (T_r, T_s) satisfies the active separation requirement if*

$$\begin{aligned}\|\mu_r - \mu_s\|_2 &\geqslant c\sqrt{m_0 k}\left(\frac{\|A - C\|}{\sqrt{n_r}} + \frac{\|A - C\|}{\sqrt{n_s}}\right)\\ &= c\sqrt{m_0}(\Delta_r + \Delta_s).\end{aligned}$$

Similarly, an inactive cluster pair (T_r, T_s) satisfies the inactive separation requirement if

$$\|\mu_r - \mu_s\|_2 \geqslant 10\sqrt{m_0}k^{\frac{1}{4}}\left(\frac{\|A - C\|}{\sqrt{n_r}} + \frac{\|A - C\|}{\sqrt{n_s}}\right).$$

Thus in this setting of $k' < \sqrt{k}$, k-FED recovers the target partitions in only one round of communication. Moreover, inactive cluster pairs need only satisfy our $\Omega(k^{\frac{1}{4}})$ separation requirement as opposed to the $\Omega(\sqrt{k})$ separation that all cluster pairs need to satisfy in the centralized setting for Lemma 1 to hold. This highlights that there exists a benefit of heterogeneity in the context of running k-FED over federated networks.

Practical benefits of k-FED. Finally, we highlight several practical benefits of the k-FED method:

- *One-shot:* k-FED only requires one round of communication for each device: one outgoing message to send the local clustering results and one incoming message to receive cluster identity information.
- *No network-wide synchronization:* Classical parallel implementations of Lloyd's heuristic and variants (e.g., Dhillon & Modha, 2002), require a network wide synchronization/initialization step. Unlike these methods, each device in k-FED works independently does not require an initialization/synchronization step.
- *New devices/Device Failures:* Assuming we have already performed clustering on the current network, for any new device entering the network, either from a previous failure or as a new participant, computing the clustering information can be done without involving any other device in the network. As we show in Theorem 3.2 (below), simply assigning any new local cluster center $\theta_i^{(z)}$ from the new device z, to the nearest device cluster mean in M sufficient. The central server only has to maintain k cluster means $\mu(\tau_1), \ldots, \mu(\tau_k)$ to perform this update.

Theorem 3.2. *Steps 2-8 of k-FED take $O(Zk' \cdot k^2)$ pairwise distance computations to terminate. Further, after the set M in Step 6 has been computed, new local cluster centers $\Theta^{(z)}$ from a yet unseen device z can be correctly assigned in $O(k' \cdot k)$ distance computations.*

As we show in Section 4, these properties of k-FED make it an ideal candidate for being used as an *inexpensive heuristic* for clustering in federated networks, either for data exploration or as part of a preprocessing step for another algorithm, even in settings where the separation requirements are not formally satisfied.

4. Applications and Experiments

We now present experimental evaluation of k-FED. We first specialize the theory to the special case where data is drawn from a mixture of k Gaussians in Section 4.1 to validate our theory on synthetic data. In Section 4.2, we evaluate k-FED on real datasets—presenting experimental evidence that highlights the benefit of heterogeneity and the communication efficiency of k-FED. We further present two applications of k-FED, in client selection as well as personalization. The dataset details for each experiment can be found in the corresponding section. Implementation of k-FED and experimental setup details can be found at: http://github.com/metastableB/kfed/.

4.1. Separating Mixture of Gaussians

We first specialize our theorem to the case of separating data generated from a mixture of k Gaussians $F_1, F_2, \ldots, F_k$. Let $\mu_r = \mu(F_r)$ be the mean of the mixture component F_r and let $w_1, w_2, \ldots, w_k$ be the mixing weights. Finally, let $w_{\min} = \min_r w_r$ be the minimum mixing weight. Let $\sigma_{\max}$ be the maximum variance along any direction among all the component distributions. Assume this data resides over our devices such that no single device has data from more than $k' < \sqrt{k}$ components. We state the following theorem (proved in Appendix A) that specifies the conditions required for this setup to satisfy our separation assumptions:

Theorem 4.1. *Let the total number of data pints, $n = poly\left(\frac{d}{w_{\min}}\right)$. Then any active cluster pairs r, s satisfy the active separation requirement with high probability if;*

$$\|\mu_r - \mu_s\|_2 \geqslant \frac{c\sqrt{km_0}\sigma_{\max}}{\sqrt{w_{\min}}} polylog\left(\frac{d}{w_{\min}}\right).$$

Further, an inactive cluster pairs r', s' satisfy the inactive separation requirement with high probability if

$$\|\mu_{r'} - \mu_{s'}\|_2 \geqslant \frac{c\sqrt{m_0}k^{\frac{1}{4}}\sigma_{\max}}{\sqrt{w_{\min}}} polylog\left(\frac{d}{w_{\min}}\right).$$

Finally, with this separation in place, all points satisfy the proximity condition with high probability.

Concretely, in this setup k-FED recovers the target clustering exactly with high probability. To empirically evaluate our theory, we instantiate an simplified instance of the above setup as follows:

Setup. Again consider the Gaussian components $F_1, \ldots, F_k$, and define the set of integers $G_i = \{p \mid (i-1)\times \sqrt{k} \leqslant p \leqslant i \times \sqrt{k}\}$. These sets G_i thus can be used to index the Gaussian components $(F_{(i-1)\sqrt{k}}, \ldots, F_{i\sqrt{k}})$. For each G_i, construct a set of data points D_i by sampling poly(dk) samples from each component F_p for $p \in G_i$. Thus the set D_i contains $\sqrt{k} \cdot$ poly(dk) samples ($w_r = \frac{1}{k}, \forall r$). Pick m_0 and for each set of data points D_i, distribute the data among m_0 devices such that each device receives exactly $\frac{1}{m_0} \cdot$poly(dk) samples. We now run k-FED on this setup and measure the quality of the clustering averaged over 10 runs, (shown in Table 1). As one would expect, the clustering

Table 1. Clustering accuracy for clustering a mixture of Gaussians. Here for all instances we choose $k' = \sqrt{k}$. We can see that the one-shot clustering produced by k-FED agrees with the target clustering with high accuracy, particularly when k is relatively small compared to d.

Parameters	Accuracy
$(d = 100, k = 16, m_0 = 5, c = 100)$	100.00 ± 0.00
$(d = 100, k = 64, m_0 = 5, c = 100)$	98.82 ± 0.70
$(d = 300, k = 64, m_0 = 5, c = 100)$	99.27 ± 0.73
$(d = 300, k = 100, m_0 = 5, c = 100)$	98.40 ± 0.80
$(d = 300, k = 16, m_0 = 5, c = 100)$	100.00 ± 0.00

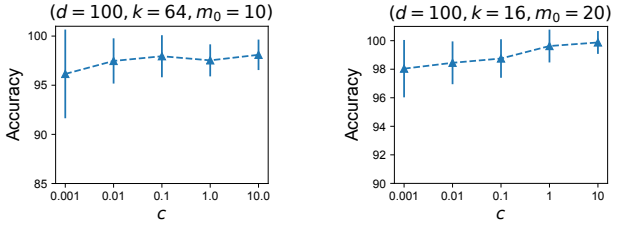

Figure 1. Impact of the separation constant c on the clustering accuracy when clustering a mixture of Gaussians. Even for relatively small values of c, for the case of data generated from a mixture of Gaussians, k-FED can recover highly accurate clustering with decreasing variance across runs.

produced by k-FED agrees strongly with the target clustering. Note that by construction all devices with data from the same set G_i contain data from the same set of Gaussian components. Further, devices with data from different sets G_i have no common Gaussian component. Thus all cluster pairs within the same set G_i are active cluster pairs and there are $\sqrt{k}\binom{\sqrt{k}}{2}$ such pairs. Moreover, any pair (r, s) such that $r \in G_i$, $s \in G_j$ $i \neq j$ form an inactive cluster pair and there are $\binom{k}{2} - \sqrt{k}\binom{\sqrt{k}}{2} = O(k^2)$ such pairs. These need only satisfy the weaker inactive separation requirement.

Note that while we prescribe $c \geqslant 100$ for our arguments to hold, Figure 1 demonstrates that clustering can be recovered even in settings where c is much smaller.

4.2. Empirical Evaluation on Real Data

In this section, we empirically explore k-FED and the related analyses from Section 3. First, we validate our theoretical results, showing that clustering over structured (heterogeneous) partitions can improve clustering performance relative to clustering over random, IID partitioned data. Second, we explore the effect of one-shot clustering relative to more communication-intensive baselines. Finally, we investigate practical applications of one-shot clustering in terms of client sampling and personalized federated learning.

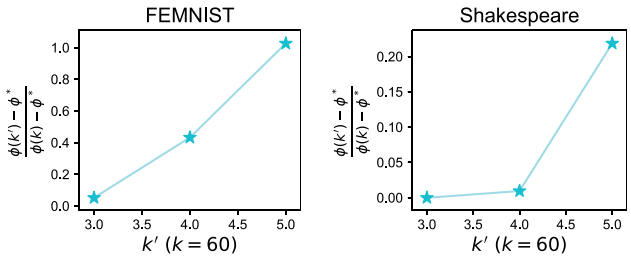

Figure 2. The k-means cost under structured partitions ($\phi(k')$) is closer to the cost of oracle clustering (ϕ^*) than that under random partitions ($\phi(k)$). As heterogeneity increases (k' decreases), the benefits of structured partitions are becoming more significant, with $\phi(k') - \phi^* \ll \phi(k) - \phi^*$.

4.2.1. Properties of k-FED

Benefits of Heterogeneity (Def. 3.2). We compare the performance of k-FED on two different partitions of data among devices: (i) one with *IID random* partitions, and (ii) another with *structured* partitions. To generate the underlying structured partition for this experiment we use the following heuristic. First, we cluster all the data into k clusters for a range of values of k. For each k, we take the clustering we have as the target clustering $\mathcal{T}$, and construct the data matrix A and the matrix of centers C. Finally, for each pair of cluster means μ_r, μ_s, we compute the quantity $\frac{\|\mu_r - \mu_s\|}{2\sqrt{m_0}(\Delta_r + \Delta_s)}$, the ratio of the actual separation of the cluster mean to the required active separation. We pick a value of k at which a large number of clusters are reasonably well separated (see Appendix B, Figure 5). We call this our *oracle clustering*. Now to generate the IID partition for (i), we randomly distribute this data among Z devices. To generate the structured partition for (ii), we divide the data among Z devices such that each device receives only data from a random subset of no more than k' clusters. For each value of k', we cluster the data for both cases over the devices using k-FED and compute the k-means cost. Let ϕ^* denotes the k-means cost of the original oracle clustering. Let $\phi(k')$ denote the k-means cost when k' clusters are assigned to each device. Figure 2 presents the relative cost ratio between the cost change in structured partitions ($\phi(k') - \phi^*$) and random partitions ($\phi(k) - \phi^*$).

We perform this experiment on the FEMNIST and Shakespeare datasets (Caldas et al., 2018) (see Appendix B for details). It can be seen from the results plotted in Figure 2 that clustering on structured splits achieves a cost closer to that of the oracle partition compared to the cost achieved on the IID random partition. We note that the separation achieved in real datasets is much smaller than required even with this careful construction (Appendix B). Even still, our experiments demonstrate that heterogeneity can benefit federated clustering on common benchmarks.

Communication-Efficiency. One advantage of the proposed method is that it requires only a single round of communication. Given this, it is natural to wonder how the performance of k-FED would compare with other, more communication-intensive clustering baselines. In particular, a common way to solve k-means in distributed settings is to simply parallelize the cluster assignment and cluster mean calculations at each step. Here, we show that for different partitions of the dataset with multiple values of k', our one-shot method k-FED is able to produce similar clustering outputs (in terms of the k-means cost; lower is better) as naive distributed k-means, which requires multiple communication rounds. Here we use the same oracle clustering as the previous experiment to construct our device data.

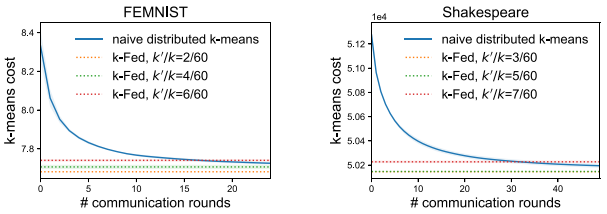

Figure 3. k-FED (using just one communication round) is able can provide similar clustering quality as naive distributed k-means.

4.2.2. Applications of k-Fed

Personalized FL. Compared with fitting a single global model to data across all device, jointly learning personalized (separate but related) models can boost the effective sample size while adapting to the heterogeneity in federated networks (e.g., Smith et al., 2017; Mansour et al., 2020).

Ghosh et al. (2020) recently proposed an algorithm to learn models over federated networks where devices are partitioned into clusters when the clustering information is unavailable. Consider a supervised learning problem that each cluster of devices want to solve and assume the number of clusters k is known. Their method, the Iterative Federated Clustering Algorithm (IFCA), in its first step initializes k models $(m_1, \ldots, m_k)$, one for each cluster. At the start of each round, all k models are sent to the devices. Each device picks the model that minimizes a loss function on its locally available data. The device can be configured to now either compute and transmit the gradient of the loss function of this model or it can perform a few model updates locally and send the updated model to the central server. As the last step of the round, for each model m_i $i \in [k]$, all the devices that picked this model are identified. All these devices are assigned cluster id i. Model m_i then is updated by either model averaging or gradient averaging using the information sent by devices in cluster i.

We instantiate IFCA on the problem of learning personalized models for clusters. As in (Ghosh et al., 2020), we use the MNIST dataset for this experiment. We construct $k = 4$ clusters by $0, 90, 180$ and 270 degree rotations and distribute them among devices. Note that in the setup for IFCA, each device only contains data from a single cluster (since we are clustering devices and not individual data points). Thus we set $k' = 1$ and compare IFCA with a simple k-FED based method: We first perform one-shot clustering to obtain an initial clustering and then we use FedAvg (McMahan et al., 2017) to learn one model per cluster. As a baseline, we also learn a single global model and include it for comparison. As can be seen from the test accuracies in Table 2 ($k' = 1$), k-FED is competitive with IFCA. Moreover, k-FED has the additional advantage that once the cluster identities have been assigned, we only need to transmit one model instead of the k models that are transmitted with IFCA.

Table 2. Test accuracy of rotated MNIST on three methods. Training personalized models based on the clustering information output by k-FED achieves the same performance of IFCA, without the high computation and communication overhead of IFCA when $k' = 1$. For $k' = 2$, the performance of k-FED degrades much less when compared to that of IFCA.

	Global	**IFCA**	**k-FED**
100 devices ($k' = 1$)	95.0	98.0	98.0
200 devices ($k' = 1$)	94.5	97.2	97.8
100 devices ($k' = 2$)	95.3	95.6	97.1
200 devices ($k' = 2$)	94.5	95.1	96.4

Since k-FED clusters data, the k-FED + FedAvg approach can also handle cases where there are data from multiple clusters on the same device. Table 2 ($k' = \sqrt{k} = 2$) shows the test accuracy on such a partition. Here we observe the performance of IFCA degrade when compared to k-FED.

Client Selection. Finally, we demonstrate that the clustering information produced by k-FED is a useful prior for client selection applications (Cho et al., 2020). In practice, cross-device federated optimization algorithms need to tolerate partial device participation (Kairouz et al., 2019). Intuitively, incorporating information from 'representative' devices at each communication round may speed up the convergence of learning tasks over federated networks as opposed to randomly sampling devices. When randomly sampling, similar and potentially redundant clients can be selected. A recent device selection method proposes to additionally select the devices with large training losses among those randomly-selected subset of devices (Cho et al., 2020) to help with convergence speed. We combine k-FED with this approach by further filtering out the devices coming from the same clusters. Note that k-FED does not add significant additional overhead to the baseline algorithm as it only requires running one-shot clustering before training.

The results are shown in Figure 4. We see that leveraging the underlying structure learnt by k-FED can boost convergence on these realistic federated benchmarks.

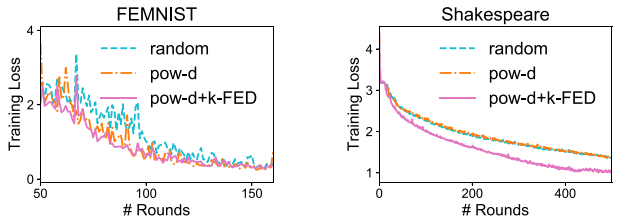

Figure 4. Additional clustering information provided by k-FED can help achieve faster convergence than recent client selection techniques pow-d (Cho et al., 2020).

Similar to Cho et al. (2020), we also observe that for the experiments in Figure 4, the variance of test performance across all devices has been reduced using client selection strategies favoring more informative (potentially more underrepresented) clients compared with that of random selection. For instance, on FEMNIST, the variance of final test accuracies is reduced by 35% when using k-fed combined with pow-d instead of random selection. This may be useful in scenarios where we wish to impose notions fairness for federated learning (Mohri et al., 2019; Li et al., 2020c).

5. Conclusion

In this work, we provide an example of how heterogeneity in federated networks can be beneficial, by rigorously analyzing the effects of heterogeneity on a simple, one-shot variant of Lloyd's algorithm for distributed clustering. Our proposed method, k-FED, addresses common practical concerns in federated settings, such as high communication costs, stragglers, and device failures. We believe that other, specific notions of heterogeneity—together with careful analyses—may provide benefits for a plethora of other problems in federated learning, which is an interesting direction of future work.

6. Acknowledgements

This work was supported in part by the National Science Foundation Grant IIS1838017, a Google Faculty Award, a Facebook Faculty Award, and the CONIX Research Center. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the National Science Foundation or any other funding agency.

References

Arthur, D. and Vassilvitskii, S. How slow is the k-means method? In *Proceedings of the Twenty-Second Annual Symposium on Computational Geometry*, 2006.

Awasthi, P. and Sheffet, O. Improved spectral-norm bounds for clustering. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*. 2012.

Bachem, O., Lucic, M., and Krause, A. Scalable k-means clustering via lightweight coresets. In *International Conference on Knowledge Discovery & Data Mining*, 2018.

Bahmani, B., Moseley, B., Vattani, A., Kumar, R., and Vassilvitskii, S. Scalable k-means+. *Proceedings of the VLDB Endowment*, 2012.

Balcan, M.-F., Ehrlich, S., and Liang, Y. Distributed k-means and k-median clustering on general topologies. In *Advances in Neural Information Processing Systems*, 2013.

Bateni, M., Bhaskara, A., Lattanzi, S., and Mirrokni, V. Distributed balanced clustering via mapping coresets. In *Advances in Neural Information Processing Systems*, 2014.

Caldas, S., Duddu, S. M. K., Wu, P., Li, T., Konečnỳ, J., McMahan, H. B., Smith, V., and Talwalkar, A. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.

Cho, Y. J., Wang, J., and Joshi, G. Client selection in federated learning: Convergence analysis and power-of-choice selection strategies. *arXiv preprint arXiv:2010.01243*, 2020.

Dasgupta, A., Hopcroft, J., Kannan, R., and Mitra, P. Spectral clustering with limited independence. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007.

Datta, S., Giannella, C., Kargupta, H., et al. K-means clustering over peer-to-peer networks. In *International Workshop on High Performance and Distributed Mining*, 2005.

Dhillon, I. S. and Modha, D. S. A data-clustering algorithm on distributed memory multiprocessors. In *Large-Scale Parallel Data Mining*. 2002.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *International Conference on Knowledge Discovery & Data Mining*, 1996.

Feldman, D., Sugaya, A., and Rus, D. An effective coreset compression algorithm for large scale sensor networks. In *International Conference on Information Processing in Sensor Networks*, 2012.

Ghosh, A., Hong, J., Yin, D., and Ramchandran, K. Robust federated learning in a heterogeneous environment. *arXiv preprint arXiv:1906.06629*, 2019.

Ghosh, A., Chung, J., Yin, D., and Ramchandran, K. An efficient framework for clustered federated learning. *Advances in Neural Information Processing Systems*, 2020.

Januzaj, E., Kriegel, H.-P., and Pfeifle, M. Dbdc: Density based distributed clustering. In *International Conference on Extending Database Technology*, 2004.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Kargupta, H., Huang, W., Sivakumar, K., and Johnson, E. Distributed clustering using collective principal component analysis. *Knowledge and Information Systems*, 2001.

Kumar, A. and Kannan, R. Clustering with spectral norm and the k-means algorithm. In *Annual Symposium on Foundations of Computer Science*, 2010.

Li, T., Sahu, A. K., Talwalkar, A., and Smith, V. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 2020a.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. Federated optimization in heterogeneous networks. In *Proceedings of Machine Learning and Systems*, 2020b.

Li, T., Sanjabi, M., Beirami, A., and Smith, V. Fair resource allocation in federated learning. In *International Conference on Learning Representations*, 2020c.

Lloyd, S. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 1982.

Mansour, Y., Mohri, M., Ro, J., and Suresh, A. T. Three approaches for personalization with applications to federated learning. *arXiv preprint arXiv:2002.10619*, 2020.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, 2017.

McSherry, F. Spectral partitioning of random graphs. In *Symposium on Foundations of Computer Science*, 2001.

Mohri, M., Sivek, G., and Suresh, A. T. Agnostic federated learning. In *International Conference on Machine Learning*, 2019.

Ostrovsky, R., Rabani, Y., Schulman, L. J., and Swamy, C. The effectiveness of lloyd-type methods for the k-means problem. *Journal of the ACM*, 2013.

Sattler, F., Müller, K.-R., and Samek, W. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. Federated multi-task learning. In *Advances in Neural Information Processing Systems*, 2017.

Tasoulis, D. K. and Vrahatis, M. N. Unsupervised distributed clustering. In *Parallel and Distributed Computing and Networks*, 2004.

Wang, S. and Chang, T.-H. Federated clustering via matrix factorization models: From model averaging to gradient sharing. *arXiv preprint arXiv:2002.04930*, 2020.

Xu, X., Jäger, J., and Kriegel, H.-P. A fast parallel clustering algorithm for large spatial databases. In *High Performance Data Mining*. 1999.