Heterogeneity for the Win: Communication-Efficient Federated Clustering

Don Kurian Dennis
Carnegie Mellon University
`dondennis@cmu.edu`

Virginia Smith
Carnegie Mellon University
`smithv@cmu.edu`

Abstract

Networks of edge devices are collecting data at an ever-increasing rate. While data generated in these federated networks is commonly unlabeled, most research effort to date has instead focused on the problem of supervised federated learning (FL). In this work, we explore the unique challenges—and opportunities—of unsupervised learning in federated networks. In particular, we develop and analyze a communication-efficient federated clustering scheme, k-FED, based on the classical and 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 separation requirements of its centralized counterpart. Our analysis shows that in heterogeneous regimes, specifically 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 point of view, k-FED also has many properties that are desirable in the federated setting: it is compute-lite, communication-efficient, asynchronous, and can naturally handle node/network failures. We motivate our analysis with experiments on several image and text benchmarks.

1 Introduction

A growing prevalence of remote devices, coupled with privacy concerns over transmitting data, has led to interest in the area of federated learning, in which the aim is to perform distributed machine learning over large, heterogeneous networks of devices such as mobile phones or wearables [24]. In practice, much of the data generated in federated networks is likely to be unlabeled or weakly labeled. However, 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 [17, 20].

In this work, we show that unsupervised learning presents unique opportunities for federated learning. We focus on a defining characteristic of federated networks: *statistical heterogeneity*, i.e., that data distributions may differ across devices. In supervised learning, many works have highlighted detrimental effects of statistical heterogeneity, observing that heterogeneity can lead to poor convergence for federated optimization methods [21, 24], result in unfair models [26], or necessitate novel forms of personalization [23, 30]. In contrast to these works, we show that heterogeneity can in fact have measurable benefits for federated learning. We focus specifically on the role that heterogeneity plays in clustering, via the development and analysis of a communication-efficient data clustering scheme.

Clustering is a crucial first step in many learning tasks and is widely used for exploratory data analysis. While many works have explored techniques for distributed clustering (Section 2), most consider data that is randomly partitioned over the network, and aim to achieve a clustering that is similar in quality to the centralized setting. Instead, in this work, we study distributed clustering in settings where the

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

data is non-identically distributed across the network, and show that our federated clustering schemes can not only match, but also exceed, standard centralized baselines.

The method we propose, k-FED, carefully considers constraints of learning over a network of devices. k-FED works with a single round of communication with a centralized server. Each device z solves a local k^z-means problem and only communicates a message of size $O(dk^z)$ containing its local cluster means. This approach allows for device failures, only requiring that there are enough devices available in the network such that there are k target clusters in the available data. Even one device per cluster is sufficient for clustering data currently available on the network. Moreover, it is possible to cluster points in previously unavailable devices via a simple recomputation at the central server.

When analyzing k-FED, our focus is on a large network of edge devices where each device z only contains data from a small number of the total clusters, i.e., $k^z = O(\sqrt{k})$. Such settings are likely to occur in heterogeneous networks when the devices only observe or interact with a portion of the complete environment. In this regime, we show that our separation requirement (which is key in the analysis of clustering methods) is similar to that of the centralized counterpart. More interestingly, while the centralized setting requires all pairs of cluster centers to satisfy a $O(\sqrt{k})$ center separation requirement, the federated approach can get away with a large fraction of cluster pairs satisfying a weaker $O(k^{\frac{1}{4}})$ separation requirement. This is the first result we are aware of to analyze the benefits of structure and heterogeneity in the context of distributed 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 $O(\sqrt{k})$ center separation assumption. Moreover, in contrast to the centralized setting, we show that a large number of cluster pairs need only a $O(k^{\frac{1}{4}})$ weaker separation assumption in heterogeneous networks. Our work highlights that heterogeneity, if carefully considered, 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 methods, and has been extensively studied in both centralized and distributed settings. Although a variety of clustering methods exist, Lloyd's heuristic [22] remains popular due to its simplicity. In the worst case, Lloyd's heuristic is known to take superpolynomial time to converge [1], but under suitable separation assumptions, can converge in polynomial time depending on the initialization [1, 3, 19, 27]. The method we propose, k-FED (Section 3.3), is a simple, communication-efficient distributed variant of these classical techniques. k-FED runs 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 Lloyd's algorithm developed by Kumar and Kannan [19] and later improved in Awasthi and Sheffet [3]. These works develop a deterministic framework with no distributional assumptions on the data, which subsumes a number of previous results [e.g., 25, 27].

Parallel and Distributed Clustering. Many works have explored parallel or distributed implementations of centralized clustering techniques [5, 9, 10, 31, 33]. Unlike the one-shot communication scheme explored herein, these methods are typically direct parallel implementations of methods such as Lloyd's heuristic or DBSCAN [12], 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 [4, 6, 7, 13, 16, 18]. These works are mostly empirical, in that there are no provable guarantees on the approximation quality of the distributed schemes. The works of [4, 6, 7] 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 in their analyses.

Federated Clustering. Several works have previously explored clustering in the context of supervised federated learning, as a way to better model non-IID data [14, 15, 29, 30]. These works differ from our own by clustering specifically in terms of devices, focusing on the downstream supervised learning task, and using either iterative [15, 29, 30] or centralized [14] clustering schemes. More recently, a distributed matrix factorization based clustering approach was explored in [32] 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 Preliminaries and Main Results

In this section, we begin by discussing some preliminaries and existing results in clustering related to Lloyd-type methods (Section 3.1). We then provide intuition and motivation for heterogeneous clustering through an empirical study in Section 3.2. Finally, in Section 3.3 we present our method k-FED and state our theoretical results. 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, $\mathcal{T} = (T_1, \dots, T_k)$, 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.

```
Algorithm 1: Local $k^z$-means
Input: The matrix of data points $A^z$ and an integer $k^z$.
1 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 $(\nu_1, \nu_2, \dots, \nu_{k^z})$.
2 Set $S_r \leftarrow \{i : \|\hat{A}^z_i - \nu_r\|_2 \leqslant \frac{1}{3}\|\hat{A}^z_i - \nu_s\|_2,$ for every $s\}$ and $\theta_r \leftarrow \mu(S_r)$.
3 Run Lloyd steps until convergence
      $U_r \leftarrow \{i : \|A^z_i - \theta_r\|_2 \leqslant \|A^z_i - \theta_s\|_2,$ for every $s\}$
      $\theta_r \leftarrow \mu(U_r)$.
Result: Cluster assignments $(U_1, U_2, \dots, U_{k^z})$ and their means $\Theta' = (\theta_1, \dots, \theta_{k'})$.
```

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 the clustering $T_1, \dots, T_k$ 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, $A_i \in T_{c(A_i)}$. 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. The quantity $(\tilde{\Delta}_r + \tilde{\Delta}_s)$ is useful when reasoning about the separation between clusters T_r and T_s, which is a key quantity in analyzing the difficulty of k-means clustering. 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:

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

Thus, two clusters are well separated if their means are c-standard-deviations apart. Using the center separation assumption in (3), Awasthi and Sheffet [3] show that the Lloyd-like algorithm defined in Algorithm-1 correctly clusters all but a small fraction of the data points. We will 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. Let

$$g_{r,s} = \left(\frac{1}{\sqrt{n_r}} + \frac{1}{\sqrt{n_s}} \right) \|A - C\| \,.$$

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*

$$\left\|\bar{A}_i - \mu_r\right\|_2 - \left\|\bar{A}_i - \mu_s\right\|_2 \geqslant g_{r,s} \,.$$

We refer to points that do not satisfy the proximity condition as 'bad points'. We now state the main result from [3] in the following lemma.

Lemma 1 (Awasthi-Sheffet, 2011)**.** *Assume there exists a well separated clustering $\mathcal{T} = (T_1, \ldots, T_k)$. 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 missclassify, we mean with respect to $\mathcal{T}$ and upto 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 we do not make any distributional assumption in this framework; all relevant quantities are defined in terms of the data matrix A and $\mathcal{T}$. Our result will follow the same framework.

3.2 *k*-FED: Motivation

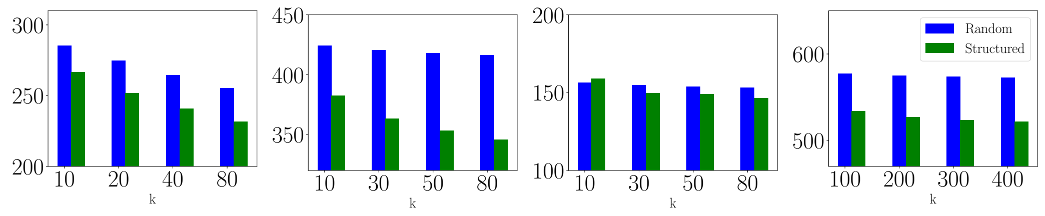
In the federated setting, the data points both *originate* and *reside* on devices in the network. Similar to the centralized setting, our objective is to classify the data on the entire network into k clusters such that each device knows the cluster assignments for its local dataset. We assume that each device can communicate to a central server.

Before we formally present our theoretical results, we first build some intuition for our method and analyses. Our focus is on a large network of devices where, while over the entire network, the number of clusters k can be quite large, each device z only contains data from a small $k^z = O(\sqrt{k})$ number of clusters. We will argue that, for such settings, if a target $\mathcal{T}$ over the entire data exists satisfying certain separation assumptions, then solving the 'easier' sub-problems on individual devices and combining the result is more advantageous than the alternate approach of combining all the (anonymized) data into one place and solving a single instance of the clustering problem.

In general, these settings are applicable to a variety of practical problems of interest, most often in cases where the data is generated on the end-points themselves. For instance, consider identifying interest 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. Observe how the partition of the data among devices in the network is 'structured' in this sense. Such structures may not exist in classical distributed settings, where the data is the often randomly partitioned among worker nodes. Moreover, even in the federated setting we lose this structure if we were to anonymize and combine the data.

To highlight this advantage further, we compare the clustering performance of a naive two-stage Lloyd's algorithm on two different partitions of data: (i) one with *IID random* partitions, and (ii) another with *structured* partitions. In particular, each device $z \in [Z]$ first performs a local clustering step using Lloyd's algorithm and returns the cluster centers $\Theta^z = (\theta_1^z, \ldots, \theta_{k^z}^z)$ corresponding to the local clustering $\mathcal{T}^z = (T_1^z, T_2^z, \ldots, T_{k^z}^z)$ to the central node. The central node then runs Lloyd's algorithm on these cluster centers and partitions them into k clusters. This second stage induces a partition on the data points on each node; for devices z_1 and z_2, if $\theta_i^{z_1}$ and $\theta_j^{z_2}$ are assigned to the same cluster at the central node, say r, then points $T_i^{z_1}$ and $T_j^{z_2}$ are assigned to cluster r.

While several benchmarks exist for supervised federated learning, there is a lack of publicly available data related to clustering for federated learning. We thus use two sources of data in our experiments: We first look at two common benchmarks for supervised federated settings, MNIST and EMNIST,



(a) MNIST (b) EMNIST (c) NeurIPS (d) Reddit

Figure 1: Comparing the performance of a naive extension of Lloyd's method on two types of splits (partitions) of data on the network. On the y axis we have the k-means cost ϕ (scaled by 10^{-3}) for various values of k on the x-axis.

and assign 'clusters' according to class labels. We then look at two classic clustering datasets, NeurIPS [28] and Reddit [11]. For both datasets, we create partitions either randomly or by assigning only a small number of clusters to each device; see Appendix B for details. Figure 1 presents the clustering cost ϕ of the final assignments on four different datasets. Across these datasets, we see that the clustering on structured splits outperforms clustering on random splits in all but one case in terms of ϕ. We build on this intuition to analyze heterogeneous clustering more rigorously in Section 3.3.

3.3 *k*-FED: Method and Main Result

We now present our method k-FED, described in Algorithm 2, and our main results. In the first step of k-FED, 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 local cluster centers and cluster assignments for each device. Note that even though each device has classified its own points into clusters, we do not have a matching for points across devices. The central server attempts to create this matching by aggregating the local cluster centers and separating them into k sets. As before, these sets induce a partition of the data on the network. If the number of devices is Z, and $k' = \max_z k^z$, then this step runs in roughly $O((Zk')^2 \log Zk')$ time.

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. 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_i^z could be empty. Let k^z be the number of non-empty subsets on that this device. For simplicity we assume k^z is identical for each device $z \in [Z]$ and denote it by k'. For the general case we can have $k' = \max_z k^z$, and our proofs will follow. Similar to the centralized case, 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 cluster T_r on some device and with $n_r^z = |T_r^z|$. We say that T_r^z is a *large* cluster, for a constant $m_0 > 1$, $n_r^z \geqslant \frac{1}{m_0} n_r$. We will use this notion to ensure that individual devices have 'enough' points. Let,

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

Here $n_{\min} = \min_r |T_r|$, is the size of the smallest cluster.

For our analysis, we require two different separation assumptions. We refer to them as *active* and *inactive* separation and introduce them through the following two definitions.

Definition 3.2 (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 3.3. *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 $\|\mu_r - \mu_s\|_2 \geqslant 10\sqrt{m_0}\lambda$.*

Note that the inactive separation requirement is much weaker than the active separation requirement for the smallest cluster. We now state our main theorem, which characterizes the performance of our two stage algorithm. We provide a detailed proof in Appendix A.

Theorem 3.1. *Assume there exists a target clustering $\mathcal{T}$ of only large clusters such that all active cluster pairs satisfy the active separation requirement and all inactive pairs satisfy the inactive separation requirement. Then, at termination of Algorithm-2 all but $O(\frac{1}{c^2})n$ points are correctly classified. Moreover, if data points for the local clustering problem on each device satisfies the proximity condition, then all points are classified correctly.*

As before, by classified we mean that the clustering produced by k-FED and $\mathcal{T}$ agree on all but $O(\frac{1}{c^2})n$ points, up to permutation of labels. Note that the above theorem holds for any $k^z \leqslant k$. Thus, our active separation requirement is stricter than that required in centralized clustering in general when $k' \approx k$. 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 poor dependency on $\sqrt{m_0}$.

Algorithm 2: k-FED

```
1  On each device $z \in [Z]$, run Algorithm-1 with local data $A^z$ and $k'$ and obtain local cluster
     centers $\Theta^z = (\theta^z_1, \dots, \theta^z_{k'})$
2  For each $\theta^z_j$, $(z, j) \in [k'] \times [Z]$, MAKE-SET($\theta^z_j$).
3  for each pair of centers $(\theta^z_j, \theta^z_t)$ ordered in increasing distance $\|\theta^z_j - \theta^z_t\|_2$ do
       // UNION() and FIND() methods from the union-find data structure.
4      if FIND($\theta^z_j$) $\neq$ FIND($\theta^z_t$) then
5          UNION(FIND($\theta^z_j$), FIND($\theta^z_t$))
6      end
7      if Number of sets $\leqslant k - 1$ then
8          Discard last union and return the $k$-sets.
9      end
10 end
   Result: Local cluster centers partitioned into $k$ sets.
```

However, as discussed previously, the data residing on the devices in federated networks are generated locally, and in a large number of practical cases, the partition is heterogeneous: number of subsets of target clusters that reside on a device is much smaller than the total number of cluster. For $k' \leqslant \sqrt{k}$, 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 $k^{1/4}$.

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

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

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

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

The cluster means of inactive cluster pairs can be much closer to each other than active pairs due to the weaker separation requirement. Since the centralized clustering setup requires all pairs of cluster means to satisfy the stricter active separation requirement, these datasets do not have the clustering guarantees of [3]. Clustering in the distributed setting retains these.

4 Conclusion and Future Work

In this work, we highlight how heterogeneity in federated networks can be beneficial by rigorously analyzing the effects of heterogeneity on a simple, communication-efficient variant of Lloyd's algorithm for distributed clustering. Our initial results reveal several interesting questions worthy of further study: (1) Can we remove our dependency on $\sqrt{m_0}$, perhaps with additional distributional assumptions? (2) Is it possible to refine our inactive analysis by characterizing the number of inactive cluster pairs? (3) Although we assume k and k' are known, can these determine be determined from the data itself? More generally, 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 a direction we hope to continue exploring in future work.

References

[1] D. Arthur and S. Vassilvitskii. How slow is the k-means method? In *Proceedings of the twenty-second annual symposium on Computational geometry*, 2006.

[2] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

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

[4] O. Bachem, M. Lucic, and A. Krause. Scalable k-means clustering via lightweight coresets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018.

[5] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii. Scalable k-means++. *VLDB*, 2012.

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

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

[8] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. Emnist: an extension of mnist to handwritten letters, 2017.

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

[10] I. S. Dhillon and D. S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Large-scale parallel data mining*. 2002.

[11] D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[12] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 1996.

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

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

[15] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran. An efficient framework for clustered federated learning. *arXiv:2006.04088*, 2020.

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

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

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

[19] A. Kumar and R. Kannan. Clustering with spectral norm and the k-means algorithm. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, 2010.

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

[21] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith. Federated optimization in heterogeneous networks. In *MLSys*, 2020.

[22] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2): 129–137, 1982.

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

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

[25] F. McSherry. Spectral partitioning of random graphs. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, 2001.

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

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

[28] V. Perrone, P. A. Jenkins, D. Spano, and Y. W. Teh. Poisson random fields for dynamic feature models. *arXiv:1611.07460*, 2016.

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

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

[31] D. K. Tasoulis and M. N. Vrahatis. Unsupervised distributed clustering. In *Parallel and distributed computing and networks*, 2004.

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

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

A Proofs

Before we can proceed to proving theorem 3.1, we need to establish a few preliminary results. Let T_r^z be the subset of points of a cluster T_r on device z. Recall that we use i to index points and r, s to index clusters. For any point, A_i^z on device z, let $c(A_i^z)$ denote the index of the cluster it belongs to. That is,

$$A_i^z \in T^z_{c(A_i^z)} \subseteq T_{c(A_i^z)}.$$

Our first lemma bounds how far the 'local' cluster mean $\mu(T_r^z)$ can deviate from $\mu(T_r)$.

Lemma 2 (Lemma 5.2 in [19]). *Let T_r^z be a subset of T_r on device z. Let $\mu(T_r^z)$ denote the mean of the points indexed by T_r^z. Then,*

$$\|\mu(T_r^z) - \mu(T_r)\|_2 \leqslant \frac{\|A - C\|}{\sqrt{|T_r^z|}}.$$

Proof. Let A^z be the sub-matrix of A on device z and let $\tilde{C}^z$ be the corresponding sub-matrix of our matrix of means C. Let u be an indicator vector for points in T_r^z. Observe that,

$$\|\ |T_r^z|(\mu(T_r^z) - \mu_r)\ \|_2 = \|(A^z - \tilde{C}^z) \cdot u\|_2 \leqslant \|A^z - \tilde{C}^z\| \|u\|_2 \leqslant \|A - C\| \sqrt{|T_r^z|}.$$

Here, for the last inequality, we note that $(A^z - \tilde{C}^z)$ contains a subset of rows of $(A - C)$. □

Now consider the local clustering problem on each device z. The device has a data matrix A^z, whose rows are a subset of A. Let $T_1^z, T_2^z, \ldots, T_k^z$ be subsets of $T_1, T_2, \ldots, T_k$ on this device, such that no more than k' of them are non-empty. Construct a matrix C^z, of the same dimensions as A^z where for each row of A^z, the corresponding row of C^z contains the mean of the local cluster the point belongs to. That is, i-th row of C^z contains $\mu(T^z_{c(A_i^z)})$.

Using this next lemma, we bound the operator norm of the matrix $(A^z - C)$, in terms of $(A - C)$.

Lemma 3. *Let $T_1^z, T_2^z, \ldots T_k^z$ be subsets of target cluster that reside on a device such that k' of them are non-empty. Let A^z be the corresponding $n^z \times d$ data matrix. Let C^z be the corresponding matrix of means; that is each row $C_i^z = \mu(T^z_{c(A_i^z)})$. Then,*

$$\|A^z - C^z\| \leqslant 2\sqrt{k'}\|A - C\|$$

Proof. Let $\tilde{C}^z$ be an $n^z \times d$ matrix where $\tilde{C}_i^z = \mu(T_{c(A_i^z)})$. First, consider a unit vector u along the top singular direction and observe that,

$$\|\tilde{C}^z - C^z\|^2 = \sum_{r=1}^{k} |T_r^z| \Big(\big(\mu(T_r^z) - \mu(T_r)\big) \cdot u \Big)^2 \leqslant \sum_{r=1}^{k} |T_r^z| \|\mu(T_r^z) - \mu(T_r)\|_2^2 \leqslant_{(a)} k' \|A - C\|^2.$$

Here for inequality (a) we invoke Lemma 2. Also note that $\|A^z - \tilde{C}^z\| \leqslant \|A - C\|$ Thus, we have

$$\|A^z - C^z\| \leqslant \|A^z - \tilde{C}\| + \|\tilde{C} - C^z\| \leqslant (1 + \sqrt{k'})\|A - C\| \leqslant 2\sqrt{k'}\|A - C\|.$$

□

Now using Lemma 2 and 3, we translate our active separation condition to the separation condition of [3] required for Algorithm-1. Since Algorithm-1 is run locally on each node, it is unaffected by the passive separation condition, as by definition, subsets of only active cluster pairs exist on each device.

Lemma 4. *Let (T_r, T_s) be cluster pairs such that, $\|\mu_r - \mu_s\|_2 \geqslant 2c\sqrt{m_0}(\Delta_r + \Delta_s)$. Let $T_r^z \subseteq T_r$ and $T_s^z \subseteq T_s$ be large subsets on device z. Then,*

$$\|\mu_r^z - \mu_s^z\|_2 \geqslant c\sqrt{m_0}(\Delta_r + \Delta_s)$$

Proof. Using the triangle inequality, we have

$$\begin{aligned}
\|\mu_r^z - \mu_s^z\|_2 &\geqslant \|\mu_r - \mu_s\|_2 - \|\mu_r^z - \mu_r\|_2 - \|\mu_s - \mu_s^z\|_2 \\
&\geqslant_{(a)} 2c\sqrt{m_0}(\Delta_r + \Delta_s) - \frac{\|A-C\|}{\sqrt{n_r^z}} - \frac{\|A-C\|}{\sqrt{n_s^z}} \\
&\geqslant 2c\sqrt{m_0}\left(k'\frac{\|A-C\|}{\sqrt{n_r}} + k'\frac{\|A-C\|}{\sqrt{n_s}}\right) - \frac{\|A-C\|}{\sqrt{n_r^z}} - \frac{\|A-C\|}{\sqrt{n_s^z}} \\
&\geqslant \left(2 - \frac{1}{ck'}\sqrt{\frac{n_r}{n_r^z m_0}}\right) c\sqrt{m_0}k'\frac{\|A-C\|}{\sqrt{n_r}} + \left(2 - \frac{1}{ck'}\sqrt{\frac{n_s}{n_s^z m_0}}\right) c\sqrt{m_0}k'\frac{\|A-C\|}{\sqrt{n_s}} \\
&\geqslant_{(b)} c\sqrt{m_0}k'\frac{\|A-C\|}{\sqrt{n_r}} + c\sqrt{m_0}k'\frac{\|A-C\|}{\sqrt{n_s}} \geqslant c\sqrt{m_0}(\Delta_r + \Delta_s).
\end{aligned}$$

Here (a) follows from Lemma 2. For (b) note that $n_r^z \geqslant \frac{1}{m_0}n_r$ and thus, $\frac{1}{ck'}\sqrt{\frac{n_r}{n_r^z m_0}} \leqslant 1$. □

Now we are ready to prove Theorem 3.1. We first use Lemma 4 followed by Lemma 1 to show that the local clustering problem on each device returns cluster means that are 'close' to the target cluster means and most of the points are classified correctly. We proceed to then use the inactive separation requirement to argue that Algorithm-2 correctly groups cluster means of subsets of the same target cluster. This will then give the required clustering over the entire network.

Proof. Consider a cluster T_r, and its subsets T_r^z, on devices $z \in [Z]$. Let θ_r^z be the mean returned by each of these devices returned by Algorithm-1. We now show that Algorithm-2 run on the central node, on termination, outputs k sets, one for each cluster, such that all θ_r^z belong to the same set. Note that this is sufficient to match correctly classified points across devices since this will induce a partition among the data points on each devices.

From Lemma 4 we see that, under the center separate assumptions, subsets of any two clusters T_r, T_s satisfy

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

Here, for inequaity (a), we used the fact that $\|A^z - C^z\| \geqslant \frac{1}{\sqrt{k'}}\|A-C\|$, from Lemma 3. Thus, the local clustering problem, with target clusters $T_1^z, T_2^z, \ldots, T_k^z$, satisfies the center separation condition of [3] specified in Lemma 1. Now, consider some device z. Let θ_r^z be the cluster mean of S_r^z corresponding to $T_r^z \subseteq T_r$. Then from Lemma 1 we have,

$$\|\theta_r^z - \mu(T_r^z)\|_2 \leqslant \frac{25}{c}\frac{\|A^z - C^z\|}{\sqrt{n_r^z}}.$$

Thus,

$$\begin{aligned}
\|\theta_r^z - \mu(T_r)\|_2 \leqslant \|\theta_r^z - \mu(T_r^z)\|_2 + \|\mu(T_r^z) - \mu(T_r)\|_2 &\leqslant_{(a)} \frac{25}{c}\frac{\|A^z-C^z\|}{\sqrt{n_r^z}} + \frac{\|A-C\|}{\sqrt{n_r^z}} \\
&\leqslant \sqrt{k'}\left(\frac{50}{c} + \frac{1}{\sqrt{k'}}\right)\frac{\|A-C\|}{\sqrt{n_r^z}} \leqslant 2\sqrt{k'}\frac{\|A-C\|}{\sqrt{n_r^z}} \\
&\leqslant_{(b)} 2\sqrt{k'm_0}\frac{\|A-C\|}{\sqrt{n_r}} \leqslant 2\sqrt{m_0}\lambda. \qquad (5)
\end{aligned}$$

Here for inequality (a), we applied Lemma 2. For inequality (b), we used the fact that $n_r^z \geqslant \frac{1}{m_0}n_r \geqslant \frac{1}{m_0}n_{\min}$. Now we exploit our inactive separation assumption. Note that every active cluster pair

(T_r, T_s), also satisfy the inactive separation requirement as $2c\sqrt{m_0}(\Delta_r + \Delta_s) \geqslant 10\sqrt{m_0}\lambda$. Thus, for two clusters r and s, and $\theta_r^{z_1}$ and $\theta_s^{z_2}$ from two devices z_1 and z_2 we have,

$$\begin{aligned} \|\theta_r^{z_1} - \theta_s^{z_2}\|_2 &= \|(\mu_r - \mu_s) + (\theta_r^{z_1} - \mu_r) - (\theta_s^{z_2} - \mu_s)\|_2 \\ &\geqslant \|\mu_r - \mu_s\|_2 - \|\theta_r^{z_1} - \mu_r\|_2 - \|\theta_s^{z_2} - \mu_s\|_2 \\ &\geqslant 6\sqrt{m_0}\lambda. \end{aligned} \tag{6}$$

Now consider a fully connected weighted graph whose vertices represent θ_r^z for $z \in [Z]$ and $r \in [k]$. For two vertices $\theta_r^{z_1}$ and $\theta_s^{z_2}$, let the edge weights be $\|\theta_r^{z_1} - \theta_s^{z_2}\|_2$. We refer to an edge between $\theta_r^{z_1}$ and $\theta_s^{z_2}$ as an inter-cluster edge and similarly, and edge between $\theta_r^{z_1}$ and $\theta_r^{z_2}$ as an intra-cluster edge. Note from (5) and (6), that intra-cluster edges weights are at most 2λ while inter-cluster edges are at least 6λ. Thus step 2 of Algorithm-2, which iterates over the edges sorted in increasing order of edge weight, observes all intra-cluster edges before it sees any inter cluster edges. As the algorithm exhausts all its intra-cluster edges, we will have with precisely k connected components corresponding to the k partition we require. The first inter-cluster edge the algorithm encounters causes the number of connected components to become $k - 1$. We discard this edge and output the required k-partitions.

For the bound on the number of misclassified points note that for each device, no more than $O(c^{-2})n^z$ points are misclassified, from Lemma 1. Moreover, if the proximity condition is satisfied for each device, then no points are misclassified. Summing over all devices proves the result. Finally, to see that the algorithm takes $O((Zk')^2 \log(Zk'))$ time, note that the dominating step is sorting all the edges of the graph. There can be at most Zk' edges, leading to $O((Zk')^2 \log(Zk'))$ complexity for this sorting step. □

B Datasets

For our experiments we use four datasets—two standard classification datasets MNIST and EMNIST[8] and two clustering datasets, NeurIPS [28] and Reddit-Top from the UCI repository[11]. To generate the structured partitions we use for our experiments, we use two different heuristics. For classification datasets with L classes, we divide data points in each class into $\frac{k}{L}$ clusters. This gives us k clusters overall. We then divide this dataset among simulated devices such that each has data from no more than $\sqrt{k}$ of these clusters. Of course, we are implicitly assuming here that there is some clustering structure within each label.

For the unlabeled datasets Reddit-Top and NeurIPS, we use a different strategy. We first use D^2 sampling as in [2] to get k 'initial centers' for each of these datasets. We then assign each point to their nearest 'initial center' to create a clustering. As before, we partition this clustering among simulated devices such that each get atmost $\sqrt{k}$ clusters.