DIFFERENTIALLY PRIVATE FEDERATED k-MEANS WITH SERVER-SIDE DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Clustering has long been a cornerstone of data analysis. It is particularly suited to identifying coherent subgroups or substructures in unlabeled data, as are generated continuously in large amounts these days. However, in many cases traditional clustering methods are not applicable, because data are increasingly being produced and stored in a distributed way, e.g. on edge devices, and privacy concerns prevent it from being transferred to a central server. To address this challenge, we present FedDP-KMeans, a new algorithm for k-means clustering that is fully-federated as well as differentially private. Our approach leverages (potentially small and out-of-distribution) server-side data to overcome the primary challenge of differentially private clustering methods: the need for a good initialization. Combining our initialization with a simple federated DP-Lloyds algorithm we obtain an algorithm that achieves excellent results on synthetic and real-world benchmark tasks. We also provide a theoretical analysis of our method that provides bounds on the convergence speed and cluster identification success.

1 INTRODUCTION

Clustering has long been the technique of choice for understanding and identifying groups and structures in unlabeled data. Effective algorithms to cluster non-private centralized data have been around for decades (Lloyd, 1982; Shi & Malik, 2000; Ng et al., 2001). However, the major paradigm shift in how data are generated nowadays presents new challenges that often prevent the use of traditional methods. For instance, the proliferation of smart phones and other wearable devices, has led to large amounts of data being generated in a decentralized manner. Moreover, the nature of these devices means that the generated data are often highly sensitive to users and should remain private. While public data of the same kind usually exists, typically there is much less of it, and it does not follow the same data distribution as the private client data, meaning that it cannot be used to solve the clustering task directly.

These observations have triggered the development of techniques for learning from decentralized data, most popularly *federated learning (FL)* (McMahan et al., 2017). Originally proposed as an efficient means of training supervised models on data distributed over a large number of mobile devices (Hard et al., 2019), FL has become the de facto standard approach to distributed learning in a wide range of privacy-sensitive applications (Brisimi et al., 2018; Ramaswamy et al., 2019; Rieke et al., 2020; Kairouz et al., 2021). However, it has been observed that, on its own, FL is not sufficient to maintain the privacy of client data (Wang et al., 2019; Geiping et al., 2020; Boenisch et al., 2023). The reason is that information about the client data, or even some data items themselves, might be extractable from the learned model weights. This is most obvious in the case of clustering: imagine that a cluster emerges that consists of a single data point. Then, this data could be read off directly from the corresponding cluster center, even if FL was used for training. Therefore, in privacy-sensitive applications, it is essential to combine FL with other privacy preserving techniques. The most common among these is *differential privacy (DP)* (Dwork, 2006), which we introduce in Section 2. DP masks information about individual data points with carefully crafted noise. This can, however, lead to a reduction in the quality of the results, referred to as the privacy-utility trade-off.

Several methods have been proposed for clustering private data that are either federated, but not DP compatible, or which are DP but not adapted to work in FL settings, see Section 6. In this paper we close this gap by introducing FedDP-KMeans, a fully federated and differentially private

k-means clustering algorithm. Our main innovation is a new initialization method, FedDP-Init, that leverages (potentially small and out-of-distribution) public data to find good initial centers. These serve as input to FedDP-Lloyds, a simple federated and differentially private variant of Lloyds algorithm (Lloyd, 1982). As we expand upon in Section 2, a good initialization is critical to obtaining a good final clustering. While this is already true for non-private, centralized clustering, it is especially the case in the differentially private, federated setting, where we are further limited by privacy and communication constraints in the number of times we can access client data and thereby refine our initialization.

We report on experiments for synthetic as well as real datasets in two settings: when we wish to preserve individual *data point privacy*, as is common for cross-silo federated learning settings (Li et al., 2020), and *client-level privacy*, as is typically used in cross-device learning settings (McMahan et al., 2017). In both cases, FedDP-KMeans achieves clearly better results than all baseline techniques. We also provide a theoretical analysis, proving that under standard assumptions for the analysis of clustering algorithms (Gaussian mixture data with well-separated components), the cluster centers found by FedDP-KMeans converge exponentially fast to the true component means and the ground truth clusters are identified after only logarithmically many steps.

2 Background

k-Means Clustering Given a set of data point, $P = (p_1, \ldots, p_n)$ and any $2 \leq k \leq n$, the goal of k-means clustering is to find *cluster centers*, $\nu_1, \ldots, \nu_k$ that minimize the *k-means objective*,

$$\sum_{i=1}^{n} \min_{j=1,\ldots,k} \|p_i - \nu_j\|^2. \tag{1}$$

The cluster centers induce a partition of the data points: a point p belongs to cluster j, if $\|p - \nu_j\| \leq \|p - \nu_{j'}\|$ for all j, j', with ties broken arbitrarily (but deterministically). It is well established that solving the k-means problem optimally is NP-hard in general (Dasgupta, 2008). However, efficient approximate algorithms are available, the most popular being Lloyd's algorithm (Lloyd, 1982). Given an initial set of centers, it iteratively refines their positions until a local minimum of (1) has been found. A characteristic property of Lloyd's algorithms is that the number of steps required until convergence and the quality of the resulting solution depend strongly on the initialization: the most commonly used initialization is the k-means++ algorithm (Arthur & Vassilvitskii, 2007).

Federated Learning Federated learning is a design principle for training a joint model from data that is stored in a decentralized way on local clients, without those clients ever having to share their data with anybody else. The computation is coordinated by a central *server* which typically employs an iterative protocol: first, the server sends intermediate model parameters to the clients. Then, the clients compute local updates based on their own data. Finally, the updates are *aggregated*, e.g. as their sum across clients, either by a trusted intermediate or using cryptographic protocols, such as multi-party computation (Bonawitz et al., 2016; Talwar et al., 2024). The server receives the aggregate and uses it to improve the current model, then it starts the next iteration. Although this framework enables better privacy, by keeping client data stored locally, each iteration incurs significant communication costs. Consequently, to make FL practical, it is important to design algorithms that require as few such iterations as possible.

While the primary focus of FL is on decentralized client data, the server itself can also possess data of its own, though usually far less than the clients in total and not of the same data distribution. Such a setting is in fact common in practice, where e.g. data from public sources, anatomized data, or data from some consenting clients is available to the server (Hard et al., 2019; Dimitriadis et al., 2020; Gao et al., 2022; Scott & Cahill, 2024).

Differential Privacy (DP) DP is a mathematically rigorous framework for computing summary information about a dataset (for us, its cluster centers) in such a way that the privacy of individual data items is preserved. Formally, for any $\varepsilon, \delta > 0$, a (necessarily randomized) algorithm $\mathcal{A} : \mathcal{P} \to \mathcal{S}$ that takes as input a data collection $P \in \mathcal{P}$ and outputs some values in a space $\mathcal{S}$, is called (ϵ, δ) *differentially private*, if it fulfills that for every $S \subset \mathcal{S}$

$$\Pr[\mathcal{A}(P) \in S] \leq e^{\varepsilon} \Pr[\mathcal{A}(P') \in S] + \delta, \tag{2}$$

where P and P' are two arbitrary *neighboring* datasets.

We consider two notions of *neighboring* in this work: for standard *data-point-level privacy*, two datasets are neighbors if they are identical except that one of them contains an additional element compared to the other. In the more restrictive *client-level privacy*, we think of two datasets as a collection of per-client contributions, and we consider two datasets as neighbors if they are identical, except that all data points of one of the individual client are missing in one of them. Condition (2) then ensures that no individual data item (a data point or a client's data set) can influence the algorithm output very much. As a consequence, from the output of the algorithm it is not possible to reliably infer if any specific data item occurred in the client data or not.

An important property of DP is its *compositionality*: if algorithms $\mathcal{A}_1, \ldots, \mathcal{A}_t$ are DP with corresponding privacy parameters $(\varepsilon_1, \delta_1), \ldots, (\varepsilon_t, \delta_t)$, then any combination or concatenation of their outputs is DP at least with privacy parameters $(\sum_{s=1}^{t} \varepsilon_s, \sum_{s=1}^{t} \delta_s)$. In fact, stronger guarantees hold, which in addition allows trading off between ε and δ, see (Kairouz et al., 2015). These cannot, however, be be stated as easily in closed form. Due to compositionality, DP algorithms can be designed easily by designing individually private steps and composing them.

In this work, we employ two mechanisms for making computational steps differentially private: The *Laplace mechanism* (Dwork et al., 2006) achieves $(\varepsilon, 0)$ privacy by adding Laplace-distributed noise with scale parameter $\frac{S}{\varepsilon}$ to the output of the computation. Here, S is the *sensitivity* of the step, i.e. the maximal amount by which its output can change when operating on two neighboring datasets, measured by the L^1-distance. The *Gaussian mechanism* (Dwork & Roth, 2014) instead adds Gaussian noise of variance $\sigma_G^2(\varepsilon, \delta; S) = \frac{2\log(1.25/\delta)S^2}{\varepsilon^2}$ to ensure (ε, δ)-privacy[1] Here, the sensitivity, S, is measured with respect to the L^2-distance. The above formulas show that stronger privacy guarantees, i.e. a smaller *privacy budget* (ε, δ), require more noise to be added. This, however, might reduce the accuracy of the output. Additionally, the more processing steps there are that access private data, the smaller the privacy budget of each step has to be in order to not exceed an overall target budget. In combination, this causes a counter-intuitive trade-off for DP algorithms that does not exist in this form for ordinary algorithms: accessing the data more often, e.g. more rounds of Lloyd's algorithm, might lead to lower accuracy results, because the larger number of steps has to be compensated by more noise per step. Consequently, a careful analysis of the privacy-utility trade-off is crucial for practical DP algorithms. As a general guideline, however, algorithms are preferable that access the private data as rarely as possible. In the context of k-means clustering this means that one can only expect good results if one can avoid having to run many iterations of Lloyd's algorithm. Consequently, a good initialization will be crucial for achieving high accuracy.

3 METHOD

We assume a setting of m clients, where each client, j, possesses a dataset, $P^j \in \mathbb{R}^{n_j \times d}$. In addition, we assume that the server, also possesses some data, Q, which can freely be shared with the clients, but that potentially is small and *out-of-distribution* (i.e. not following the client data distribution). The goal is to determine a k-means clustering of the joint clients' dataset $P := \bigcup_{j=1}^{m} P_j$ in a *federated* and *differentially private* way.

We propose FedDP-KMeans, which solves this task in two stages. the first, FedDP-Init (Algorithm 1), is our main contribution: it constructs a strong initialization to the k-means clustering problem by exploiting server-side data. The second, FedDP-Lloyds (Algorithm 2), is a simple federated DP-Lloyds algorithm, which refines the initialization, if necessary.

3.1 FEDDP-INIT

Sketch: FedDP-Init has three steps: **Step 1** computes a projection matrix onto the space spanned by the top k singular vectors of the client data matrix P. **Step 2** projects the server data onto that subspace, and computes a weight for each server point q that reflects how many client points have q as their nearest neighbor. **Step 3** computes initial cluster centers in the original data space by first clustering the weighted server data in the projected space and then refining these centers by a

[1]For simplicity of exposition, we assume $\varepsilon \leq 1$ for all steps involving the Gaussian mechanism, as larger values require a different noise scaling. Note that the complete algorithm nevertheless can handle larger privacy budgets, as the overall privacy level is determined from the per-step levels as approximately their sum.

step resembling one step of Lloyd's algorithm on the clients, but with the similarity computed in the projected space. To ensure the privacy of the client data all above computations are performed with sufficient amounts of additive noise, and the server only ever receives noised aggregates of the computed quantities across all clients. Consequently, FedDP-Init is differentially private and fully compatible with standard FL and secure aggregation setups, as described in Section 2.

Intuitively, the goal of Step 1 is to project the data onto a lower-dimensional subspace that preserves the important variance (i.e. distance between the means) but reduces the variance in nuisance direction (in particular the intra-cluster variance). This construction is common for clustering algorithm that strive for theoretical guarantees, and was popularized by Kumar & Kannan (2010). Our key novelty lies in Step 2 and 3: here, we exploit the server data, essentially turning it into a proxy dataset on which the server can operate without any privacy cost. After one more interaction with the clients, the resulting cluster centers are typically so close to the optimal ones, that only very few (sometimes none at all) steps of Lloyd's algorithm will still be required afterwards to refine them. Our theoretical analysis (Section 4) quantifies this effect: for suitably separated Gaussian Mixture data, the necessary number of steps to identify the ground truth clusters is at most logarithmic in the total number of data points.

In the rest of this section, we describe the individual steps in more technical detail. For the sake of simpler exposition, we describe only the setting of data-point-level differential privacy. However, only minor changes are needed for client-level privacy, see Section 5. As private budget, we treat δ as fixed for all steps, and denote the individual budgets of the three steps as ε_1, ε_2 and ε_3. We provide recommendations how to set these values given an overall privacy budget in Appendix G.4.

Algorithm details – Step 1: The server aims to compute the top k eigenvectors of the clients' data outer product matrix $P^T P$. However, in the federated setup, it cannot do so directly because it does not have access to the matrix P. Instead, the algorithm exploits that the overall outer product matrix can be decomposed as the sum of the outer products of each client data matrix, i.e. $P^T P = \sum_{j=1}^{m} (P^j)^T P^j$. Therefore, each client can locally compute their outer product matrix and the server only receives their noisy across-client aggregate, $\widehat{P^T P}$. We ensure the privacy of this computation by the Gaussian mechanism. The associated sensitivity is the maximum squared norm of any single data point, which is upper bounded by the square of the dataset radius, Δ. Consequently, a noise variance of $\sigma_G^2(\varepsilon_1, \delta; \Delta^2)$ ensures (ε_1, δ)-privacy, as shown by Dwork et al. (2014).

The remaining operations the server can perform noise-free: it computes the top k eigenvectors of $\widehat{P^T P}$ and forms the matrix $\Pi \in \mathbb{R}^{d \times k}$ from them, which allows projecting to the subspace spanned by these vectors (which we call *data subspace*). The projection provides a data-adjusted way of reducing the dimension of data vectors from potentially large d to the typically much smaller k. This is an important ingredient to our algorithm, because in low dimension typically less noise is required to ensure privacy. The lower dimension also helps to keep the communication between server and client small. The dimension k is chosen, because for sufficiently separated clusters, one can then expect the subspace to align well with the subspace spanned by the cluster centers. In that case, the projection will preserve inter-cluster variance but reduce intra-cluster variance, thereby improving the signal-to-noise ratio of the data.

Step 2: Next, the server aims to compute per-point weights for its own data such that this can serve as a proxy for the data of the clients. The server shares with the clients the computed projection matrix Π, and its own projected dataset ΠQ. Each client uses Π to project its own data to the data subspace. Then, it computes a weight for each server point $q \in \Pi Q$ as,

$$w_q(\Pi P^j) \coloneqq \left|\{p \in \Pi P^j \mid \forall q' \in \Pi Q,\ \|p - q\| \leq \|p - q'\|\}\right|, \tag{3}$$

that is, the count of how many of the client's projected points are closer to q than to any other $q' \in \Pi Q$, breaking ties arbitrarily. The weights are sent to the server in aggregated and noised form. As an unnormalized histogram over the client data, the point weight has L^1-sensitivity 1. Therefore, the Laplace mechanism with noise scale $1/\varepsilon_2$ makes this step $(\varepsilon_2, 0)$-DP. The noisy total weights, $\widehat{w_q(\Pi P)}$ for $q \in \Pi Q$, provide the server with a (noisy) estimate of how many client data points each of its data points represents. It then runs k-means clustering on its projected data ΠQ, where each point q receives weight $\widehat{w_q(\Pi P)}$ in the k-means cost function, to obtain centers $\xi_1, \ldots, \xi_k$ in the data subspace.

Algorithm 1 FedDP-Init

Input: Client data sets $P^1, \ldots, P^m$, # of clusters k, privacy parameters $\varepsilon_1, \varepsilon_2, \varepsilon_{3G}, \varepsilon_{3L}, \delta$
Step 1: *// Compute projection onto top k singular vectors of P*
for client $j = 1, \ldots, m$ **do**
  Client j computes outer product $(P^j)^T P^j$
end for
Server receives noisy aggregate $\widehat{P^T P} = \sum_{j=1}^m (P^j)^T P^j + \mathcal{N}_{d\times d}(0, \sigma^2(\varepsilon_1, \delta; \Delta^2))$
Server forms a projection matrix Π from top k eigenvectors of $\widehat{P^T P}$
Step 2: *// Determine importance weights*
for client $j = 1, \ldots, m$ **do**
  Client j receives Π and ΠQ from server
  for every point $q \in \Pi Q$ **do**
    Client j computes weight $w_q(\Pi P^j) := \left|\{p \in \Pi P^j \mid \forall q' \in \Pi Q,\ \|p - q\| \leq \|p - q'\|\}\right|$
  end for
end for
Server receives noisy aggregate $\widehat{w_q(\Pi P)} = \sum_{j=1}^m w_q(\Pi P^j) + \mathrm{Lap}(0, \frac{1}{\varepsilon_2})$ for each $q \in \Pi Q$
Step 3: *// Cluster projected server points and initialize centers*
Server computes cluster centers $\xi_1, .., \xi_k$ by running k-means clustering of ΠQ with per-sample weights $\widehat{w_q(\Pi P)}$
for client $j = 1, \ldots, m$ **do**
  Client j receives $\xi_1, .., \xi_k$ from server
  Client j computes $S_r^j = \{p \in P^j : \forall s, \|\Pi p - \xi_r\| \leq \|\Pi p - \xi_s\|\}$
  Client j computes $m_r^j = \sum_{p \in S_r^j} p$ and $n_j^r = |S_r^j|$
end for
Server receives noisy aggregates $\widehat{m_r} = \sum_{j=1}^m m_r^j + \mathcal{N}_d(0, \sigma^2(\varepsilon_{3G}, \delta; \Delta))$ and $\widehat{n_r} = \sum_{j=1}^m n_r^j + \mathrm{Lap}(0, \frac{1}{\varepsilon_{3L}})$
Server computes initial centers $\nu_r = \widehat{m_r}/\widehat{n_r}$ for $r = 1, \ldots, k$
Output: Initial cluster centers $\nu_1, .., \nu_k$

Step 3: In the final step the server constructs centers in the original space. For this, it sends the projected centers $\xi_1, \ldots, \xi_k$ to the clients. For each projected cluster center ξ_r, each client j computes the set of all points $p \in P^j$ whose closest center in the projected space is ξ_r, i.e. $S_r^j := \{p \in P^j : \forall s, \|\Pi p - \xi_r\| \leq \|\Pi p - \xi_s\|\}$. For any r, the union of these sets across all clients would form a cluster in the client data. We want the mean vector of this to constitute the r-th initialization center. For this, each client j computes the sum of their points in each cluster, $m_r^j = \sum_{p \in S_r^j} p$, and the number of points in of each of their clusters, $n_j^r = |S_r^j|$. Aggregated across all clients one obtains the global sum and count of the points in each cluster: $m_r = \sum_{j=1}^m m_r^j$ and $n_r = \sum_{j=1}^m n_r^j$. To make this step private, we first split $\varepsilon_3 = \varepsilon_{3G} + \varepsilon_{3L}$. For m_r^j, which has L^2-sensitivity Δ, we apply the Gaussian mechanism with variance $\sigma^2(\varepsilon_{3G}, \delta; \Delta)$. For n_r, which has the L^1-sensitivity is 1, we use the Laplace mechanisms with scale $1/\varepsilon_{3L}$. This ensures $(\varepsilon_{3G}, \delta)$ and $(\varepsilon_{3L}, 0)$ privacy, respectively, and therefore (at least) (ε, δ) privacy overall for this step. Finally, the server uses the noisy estimates of the total sums and counts, $\widehat{m_r}$ and $\widehat{n_r}$, to compute approximate means $\nu_r = \widehat{m_r}/\widehat{n_r}$, and outputs these as initial centers.

3.2 FEDDP-LLOYDS

The second step of FedDP-KMeans is a variant of Lloyd's algorithm that we adapt to a private federated setting. The basic observation here is that a step of Lloyd's algorithm can be expressed only as summations and counts of data points. Consequently, all quantities that the server requires can be expressed as aggregates over client statistics which allows us to preserve user privacy with secure aggregation and differential privacy, as described in Section 2.

Specifically, assume that we are given initial centers $\nu_1^0, \ldots, \nu_k^0$, and a privacy budget $(\varepsilon_4, \delta_4)$, which we split as $\varepsilon_4 = \varepsilon_{4G} + \varepsilon_{4L}$. For rounds $t = 1, \ldots, T$, we repeat the following steps. The server

Algorithm 2 FedDP-Lloyds

Input: Initial centers $\nu_1^0, \ldots, \nu_k^0$, P, steps T, privacy parameters $\varepsilon_G, \varepsilon_L, \delta$
for $t = 1, \ldots, T$ **do**
  for client $j = 1, \ldots, m$ **do**
    Client j receives $\nu_1^{t-1}, \ldots, \nu_k^{t-1}$ from Server
    for $r = 1, \ldots, k$ **do**
      Client j computes $S_r^j := \{p \in P^j : \forall s, \|p - \nu_r^{t-1}\| \leq \|p - \nu_s^{t-1}\|\}$
      Client j computes $m_r^j = \sum_{p \in S_r^j} p$ and $n_j^r = |S_r^j|$
    end for
  end for
  Server receives $\widehat{m_r} = \sum_{j=1}^m m_r^j + \mathcal{N}_d(0, T\Delta^2\sigma^2(\varepsilon_G/T, \delta))$ and $\widehat{n_r} = \sum_{j=1}^m n_r^j + \mathrm{Lap}(0, \frac{T}{\varepsilon_L})$
  Server computes next centers $\nu_r^t = \widehat{m_r}/\widehat{n_r}$ for $r = 1, \ldots, k$
end for
Output: Final cluster centers $\nu_1^T, .., \nu_k^T$

sends the latest estimate of the centers to the clients. Each client j computes, for $r = 1, \ldots, k$, $S_r^j := \{p \in P^j : \forall s, \|p - \nu_r^{t-1}\| \leq \|p - \nu_s^{t-1}\|\}$, the set of points whose closest center is ν_r^{t-1}. Note that in contrast to the initialization, the distance is measured in the full data space here, not the data subspace. The remaining steps coincide with the end of Step 3 above. Each client j computes the summations and counts of their points in each cluster: $m_r^j = \sum_{p \in S_r^j} p$ and $n_j^r = |S_r^j|$. These quantities are aggregated to $m_r = \sum_{j=1}^m m_r^j$ and $n_r = \sum_{j=1}^m n_r^j$, and made private by the Gaussian mechanisms with variance $\sigma^2(\varepsilon_{4G}/T, \delta/T, \Delta)$ and the Laplacian mechanism with scale T/ε_{4L}, respectively. The server receives the noisy total sums and counts $\widehat{m_r}$ and $\widehat{n_r}$, and it updates its estimate of the centers as $\nu_r^t = \widehat{m_r}/\widehat{n_r}$. Overall, the composition property of DP ensures that FedDP-Lloyds is at least (ε_4, δ)-private.

4 Theoretical analysis

We analyze the theoretical properties of FedDP-KMeans in the standard setting of data from a k-component Gaussian mixture, i.e. the data P is sampled from a distribution $\mathcal{D}(x) = \sum_{j=1}^k w_j \mathcal{G}_j(x)$ with means μ_j, covariance matrix Σ_j and cluster weight w_j. The data is partitioned arbitrarily among the clients, i.e. each clients data is not necessarily distributed according to $\mathcal{D}$ itself. We denote by G_j the set of samples from the j-th component $\mathcal{G}_j$: the goal is to recover the clustering $G_1, ..., G_k$. The server data, $Q \subset \mathbb{R}^d$, can be small and not of the same distribution as P.

Our main result is Theorem 2, which states that FedDP-KMeans successfully clusters such data, in the sense that the cluster centers it computes converge towards the ground truth cluster centers, i.e. the means of the Gaussian parameters, and the induced clustering becomes the ground truth one. In doing so, the algorithm respects data-point differential privacy. For this result to hold, a *separation condition* is required (Definition 1), which ensures that the ground truth cluster centers are separated far enough from each other to be identifiable. In the following, we first introduce and discuss the separation condition and then state the theorem. The proof is provided in Appendix E and F.

Definition 1 (Separation Condition)**.** *For a constant c, a Gaussian mixture $\big((\mu_i, \Sigma_i, w_i)\big)_{i=1,\ldots,k}$ with n samples is called c-*separated *if*

$$\forall i \neq j, \|\mu_i - \mu_j\| \geq c\sqrt{\frac{k}{w_i}}\sigma_{max}\log(n),$$

where σ_{max} is the maximum variance of any Gaussian along any direction. For some large enough constant c fixed independently of the input, we simply say that the mixture is separated[2].

Note that the dependency in $\log(n)$ is unavoidable, because with growing n also the chance grows that outliers occur from the Gaussian distributions: assigning each data point to its nearest mean would not be identical to the ground truth clustering anymore.

[2]This constant c is determined by prior works: our analysis uses results from Awasthi & Sheffet (2012), which did not specify exactly the value of the constant nor tried to optimize it.

To prove the main theorem, two additional assumptions on P are required: (1) the diameter of the dataset is bounded by $\Delta := O\big(\frac{k \log^2(n)\sqrt{d}\sigma_{\max}}{\varepsilon w_{\min}}\big)$ – so that the noise added to compute a private SVD preserves enough signal. (2): there is not too many server data, namely $|Q| \leq \frac{\varepsilon n k \sigma_{\max}^2}{\Delta^2}$. This ensures the noise added Step 2 is not overwhelming compared to the signal. Note that conditions (1) and (2) can always be enforced by two preprocessing steps, which we present as part of the proof in Appendix E. In practice, however, they are typically satisfied automatically – as we observe in Appendix G.2 – thereby allowing use of the algorithm directly as stated.

Theorem 2. *Suppose that the client dataset P is generated from a separated Gaussian mixtures with $n \geq \zeta_1 \frac{k \log^3 n \sqrt{d}\sigma_{max}}{\varepsilon^2 w_{min}^2}$ samples, where ζ_1 is some universal constant, and that Q contains a least one sample from each component of the mixture. Then, FedDP-KMeans followed with FedDP-Lloyds is (ε, δ)-DP for $\varepsilon = \varepsilon_1 + \varepsilon_2 + \varepsilon_{3G} + \varepsilon_{3L} + \varepsilon_{4G} + \varepsilon_{4L}$, and there is a constant ζ_2 such that, under assumptions (1) and (2), the centers $\nu_1, ..., \nu_k$ that are computed after T steps of FedDP-Lloyds satisfy with high probability*

$$\|\mu_i - \nu_i\| \leq \zeta_2 \cdot \left(2^{-T} \cdot \sqrt{\frac{n\sigma_{max}^2}{|G_i|}} + \frac{T\Delta \log(n)}{\varepsilon n w_{min}}\right). \tag{4}$$

Furthermore, there is a constant ζ_3 such that, after $\zeta_3 \log(n)$ rounds of communication, the clustering induced by $\nu_1, ..., \nu_k$ is the ground-truth clustering $G_1, ..., G_k$.

Note that assumption (1) implies that $\frac{\Delta \log(n)}{\varepsilon w_{\min}}$ is negligible compared to n. That means, the estimated centers converge exponentially fast towards the ground truth.

5 Experiments

We now present our empirical evaluation of FedDP-KMeans, which we implemented using the `pfl-research` framework (Granqvist et al., 2024). To verify the broad applicability of our method we run experiments in both the setting of data-point-level privacy, see Section 5.1, and client-level privacy, see Section 5.2. The appropriate level of privacy in FL is typically determined by which data unit corresponds to a human. In *cross-silo* FL we typically have a smaller number of large clients, e.g. hospitals, with each data point corresponding to some individual, so data point-level privacy is appropriate. In *cross-device* FL, we typically have a large number of clients, where each client is a user device such as a smartphone, so client-level privacy is preferable. Our chosen evaluation datasets reflect these dynamics.

Baselines As natural alternatives to FedDP-KMeans we consider different ways of initializing the k-means problem and combine these with FedDP-Lloyds. Two baseline methods use the server data to produce initialization: *ServerKMeans++* runs k-means++ (Arthur & Vassilvitskii, 2007) on the server data, while *ServerLloyds* runs a full k-means clustering of the server data. The baselines can be expected to work well when the server data is large and of the same distribution as the client data. This, however, is exactly the situation where the server data would suffice anyway, so any following FL would be wasteful. In the more realistic setting where the server data is small and/or out-of-distribution, the baselines might produce biased and therefore suboptimal results. As a third baseline, we include the *SpherePacking* initialization of (Su et al., 2017). This data-independent technique constructs initial centroids that are suitably spaced out and cover the data space, see Appendix G.3 for details. None of the above baselines use client data for initialization. Therefore, they consume none of their privacy budget for this step, leaving all of it for the subsequent FedDP-Lloyds.

In addition to the above ones, we also report results for two methods that do not actually adhere to the differentially-private federated paradigm. *k-FED* (Dennis et al., 2021) is the most popular federated k-means algorithm. As we will discuss in Section 6 it does not exploit server data and it does not offer privacy guarantees. *Optimal* we call the method of transferring all client data to a central location and running non-private k-means clustering with *kmeans++* initialization. This provides neither the guarantees of federated learning nor of differential privacy, but it serves as a lower bound on the achievable k-means cost for all other methods.

Evaluation Procedure We compare FedDP-KMeans with the baselines over a range of privacy budgets. Specifically, if a method has s steps that are each $(\varepsilon_1, \delta), \ldots, (\varepsilon_s, \delta)$ DP then the total privacy cost of the method is computed as $(\varepsilon_{\text{total}}, \delta)$ by strong composition using Google's

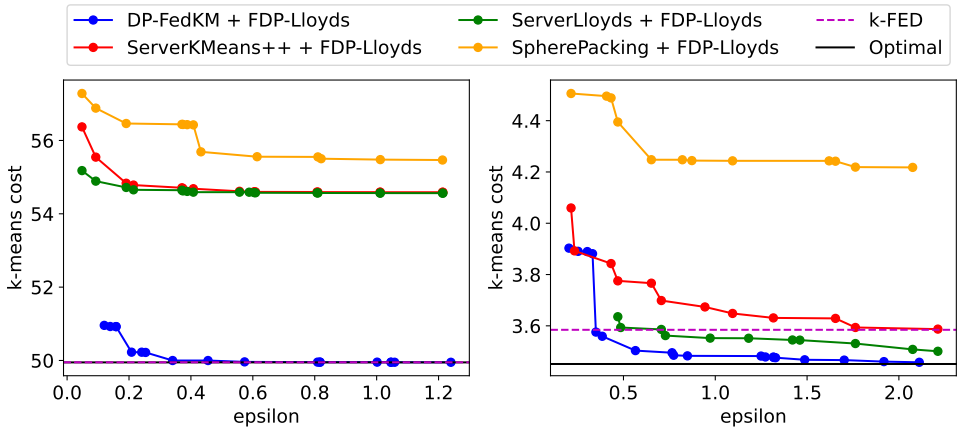

Figure 1: Results with data-point-level privacy ($k = 10$). Left: synthetic mixture of Gaussians data with 100 clients. Right: US census dataset. The 51 clients are US states, each client has the data of individuals with employment type "Federal government employee".

`dp_accounting` library [3]. We fix $\delta = 10^{-6}$ for all steps and for the total privacy costs. We vary the ε_i of individual steps as well as other hyperparameters of the algorithms, e.g. the number of steps of FedDP-Lloyds, and we measure the k-means cost of the computed clustering. For each method we plot the Pareto front of the results in the (k-means cost, $\varepsilon_{\text{total}}$) space. When plotting we scale the k-means cost by the dataset size, so the value computed in Equation 1 is scaled by $1/n$. This evaluation procedure gives us a good overview of the performance of each method at a range of different privacy budgets. However, on its own it does not tell us how to set hyperparameters for FedDP-KMeans, such as the amount of privacy budget to allocate to each step. Knowing how to set the hyperparameters is important for applying FedDP-KMeans in practice and we address this in Appendix G.4.

5.1 Data-point-level Privacy Experiments

Privacy Implementation details In our theoretical discussion we assumed that no individual data point has norm larger than Δ in order to compute the sensitivity of certain steps. As Δ is typically not known in practice, in our experiments we ensure the desired sensitivity by clipping the norm of each data point to be at most Δ, before using it in any computation. Δ is therefore a hyperparameter of the algorithm, which we set to be the radius of the server dataset.

Datasets We evaluate on both synthetic and real federated datasets that resemble a cross-silo federated setting. Our synthetic data comes from a mixture of Gaussians distribution, as assumed for our theoretical results in Section 4. The client data is of this mixture distribution while the server data consists to two thirds of data from the true mixture and to one third of data that is uniformly distributed, to simulate related but out-of-distribution data. We additionally evaluate on US census data using the folktables (Ding et al., 2021) package. The dataset has 51 clients, each corresponding to a US state. Each data point contains information about an individual in the census. For full details on the datasets and our preprocessing steps see Appendix G.1.

Results In Figure 1 we report the outcomes. The left panel shows the results for the synthetic Gaussian mixture and the right panel for the US census dataset when the clients hold the data of federal employees. The other two categories are shown in Figures 3 and 4 of Appendix H. On the synthetic data, FedDP-KMeans outperforms all private baselines by a wide margin. These baselines, are unable to overcome their poor initialization, with performance plateauing even as the privacy budget increases. In contrast FedDP-KMeans is able to match the optimal (non-private) performance at a low privacy budget of around $\varepsilon_{\text{total}} = 0.4$. The non-private k-FED also performs optimally in this setting as is to be expected given that the synthetic data distribution fulfills the conditions assumed by Dennis et al. (2021). On the US census datasets we observe a more interesting picture. Across

[3] https://github.com/google/differential-privacy/tree/main/python/dp_accounting

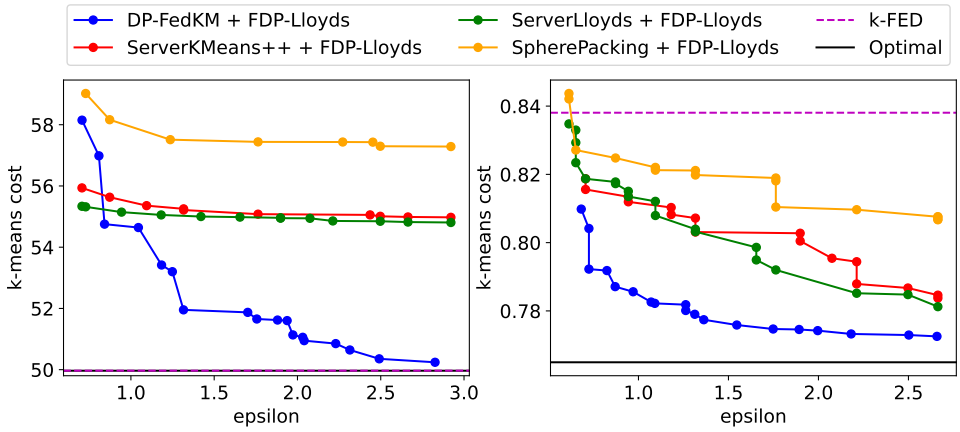

Figure 2: Results with client-level privacy ($k = 10$). Left: synthetic mixture of Gaussians data with 2000 clients. Right: stackoverflow dataset with 9237 clients, topic tags github and pdf.

all three settings FedDP-KMeans outperforms all baselines, except in the very low privacy budget regime. The latter is to be expected, because for sufficiently low privacy budget any client-based initialization will become very noisy, whereas the initialization with only server data (which requires no privacy budget) stays reasonable. With a high enough privacy budget FedDP-KMeans is able to recover the optimal non-private clustering. Among the baselines we observe similar performance between the two methods that initialize using server data, with ServerLloyds performing slightly better across the board. The data independent SpherePacking initialization performs very poorly, emphasizing the importance of leveraging related server data to initialize.

We attribute FedDP-KMeans's good performance predominantly to the excellent quality of its initialization. As evidence, Table 4 in Appendix G shows how many steps of Lloyd's algorithm had to be performed for Pareto-optimal behavior: this is never more than 2, and often none at all.

5.2 Client-level Privacy Experiments

Privacy Implementation details Moving to client-level differential privacy changes the sensitivities of the steps of our algorithms, which now depend not only on the maximum norm of a client data point norm, but also on the maximum number of data points a client has. Rather than placing assumptions or restrictions on this, and deriving corresponding bounds on the sensitivity of each step, we instead simply enforce sensitivity by clipping client statistics prior to aggregations. This is a standard technique to enforce a given sensitivity in private FL, where it is typically applied to clipping client model/gradient updates. For full details on our implementation in the client-level privacy setting see Appendix G.5

Datasets We evaluate on both synthetic and real federated datasets, this time in a cross-device federated setting. For synthetic data we again use a mixture of Gaussians, but with more clients than in Section 5.1. We also use the Stack Overflow dataset provided by Tensorflow Federated[4]. This is a large scale text dataset of questions posted by users on stackoverflow.com. We preprocess this dataset by embedding it with a pre-trainined sentence embedding model. Thus each client dataset consists of small number of text embedding vectors. The server data consists of embedding vectors from questions asked about different topics to the client data. See Appendix G.1 for full details.

Results In Figure 2 we report the outcomes. The left panel shows results for the synthetic Gaussian mixture dataset with 2000 clients, and the right panel for the stackoverflow dataset, with topics github and pdf. Further results can be found in Appendix H: synthetic data with 1000 and 5000 clients in Figures 5 and 6, and the other stackoverflow topics are shown in Figures 7, 8 and 9.

For the synthetic data we again observe that the baselines that use only server data are unable to overcome their poor initialization, even with more generous privacy budgets. As the total number of

[4] https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/stackoverflow

clients grows, from 1000 to 2000 to 5000, FedDP-KMeans exhibits better performance for the same privacy budget and the budget at which FedDP-KMeans outperforms server initialization becomes smaller. This is to be expected since the impact of the noise will be lower the more clients we are able to aggregate over. For stackoverflow we again observe that FedDP-KMeans exhibits the best performance, except for in a few cases in the low privacy budget regime. k-FED performs quite poorly across the board, tending to be outperformed by the private baselines.

As in data-point-level privacy, we find the quality of FedDP-Init's initialization to be excellent: very few, if any, Lloyd's steps are required for Pareto-optimality (see Table 5 in Appendix G).

6 Related Work

In the context of FL, clustering appears primarily for the purpose of grouping clients together. Such *clustered FL* techniques jointly find a clustering of the clients while training a separate ML model on each cluster (Sattler et al., 2020; Ghosh et al., 2020; Xia et al., 2020). In contrast, in this work we are interested in the task of clustering the clients' data points, rather than the clients. In Dennis et al. (2021), the one-shot scheme k-Fed is proposed for this task: first all clients cluster their data locally. Then, they share their cluster centers with the server, which clusters the set of client centers to obtain a global clustering of the data. However, due to the absence of aggregation of the quantities that clients share with the server, the method has no privacy guarantees. Liu et al. (2020) propose using *federated averaging* (McMahan et al., 2017) to minimize the k-means objective in combination with multi-party computation. Similarly, Mohassel et al. (2020) describe an efficient multi-party computation technique for distance computations. This will avoid the server seeing individual client contributions before aggregation, but the resulting clustering might still expose private information.

For privacy-preserving clustering, many methods have been proposed based on variants of DPLloyd's (Blum et al., 2005), i.e. Lloyd's algorithm with suitable noise added to intermediate steps. The methods differ typically in the data representation and initialization. For example, Su et al. (2016) creates and clusters a proxy dataset by binning the data space. This, however, is tractable only in very low-dimensional settings. Chang et al. (2021) also works with a proxy dataset, which it constructs in a private way from client data points. Ren et al. (2017) chooses initial center points by forming subsets of the original data and clustering those. Zhang et al. (2022) initializes with randomly selected data points and then uses multi-party computation to securely aggregate client contributions. None of the methods are compatible with the FL setting, though.

To our knowledge, only two prior works combined the advantages of DP and FL so far. Li et al. (2023) is orthogonal to our work, as it targets *vertical FL*, in which all clients posses the same data points, only different subsets of their features. Diaa et al. (2024) studies the same problem as we do, but they propose a custom aggregation scheme that does not fit standard security requirements of FL. For initialization, it uses SpherePacking, which in our experiments led to rather poor results.

7 Conclusion

In this paper we presented FedDP-KMeans, a fully federated and differentially private k-means clustering algorithm. FedDP-KMeans makes use of out-of-distribution server-side data to obtain a good initialization to the k-means problem. Combined with a simple federated, differentially private, variant of Lloyd's algorithm we obtain an efficient and practical clustering algorithm. We demonstrate that FedDP-KMeans performs well in practice under both data-point-level and client-level privacy models. FedDP-KMeans also comes with theoretical guarantees that show exponential convergence to the true cluster centers in the Gaussian mixture setting.

A remaining shortcoming of our method is the need to choose hyperparameters, which is known to be difficult when privacy is meant to be ensured. While we provide heuristics for this in Appendix G.4, a more principled solution would be preferable. It would also be interesting to explore if the server-side data could be replaced with a suitably private mechanism based on client data, and if a variant of FedDP-Init is possible that adjusts to very small privacy budgets.

REFERENCES

Dimitris Achlioptas and Frank McSherry. On spectral learning of mixtures of distributions. In *Conference on Computational Learning Theory (COLT)*, 2005.

David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *Symposium on Discrete Algorithms (SODA)*, 2007.

Hassan Ashtiani, Shai Ben-David, Nicholas JA Harvey, Christopher Liaw, Abbas Mehrabian, and Yaniv Plan. Near-optimal sample complexity bounds for robust learning of Gaussian mixtures via compression schemes. *Journal of the ACM (JACM)*, 67(6):1–42, 2020.

Pranjal Awasthi and Or Sheffet. Improved spectral-norm bounds for clustering. In *International Workshop on Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX)*, 2012.

Alex Bie, Gautam Kamath, and Vikrant Singhal. Private estimation with public data. *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the SuLQ framework. In *Symposium on Principles of Database Systems (PODS)*, 2005.

Franziska Boenisch, Adam Dziedzic, Roei Schuster, Ali Shahin Shamsabadi, Ilia Shumailov, and Nicolas Papernot. When the curious abandon honesty: Federated learning is not private. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2023.

K. A. Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for federated learning on user-held data. In *NIPS Workshop on Private Multi-Party Machine Learning*, 2016. URL https://arxiv.org/abs/1611.04482.

Theodora S. Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch. Paschalidis, and Wei Shi. Federated learning of predictive models from federated electronic health records. *International Journal of Medical Informatics*, 112:59–67, 2018.

Alisa Chang and Pritish Kamath. Practical differentially private clustering. https://research.google/blog/practical-differentially-private-clustering/, 2021. Accessed: 2024-09-23.

Alisa Chang, Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. Locally private k-means in one round. In *International Conference on Machine Learning (ICML)*, 2021.

Edith Cohen, Haim Kaplan, Yishay Mansour, Uri Stemmer, and Eliad Tsfadia. Differentially-private clustering of easy instances. In *International Conference on Machine Learning (ICML)*, 2021.

S. Dasgupta. The hardness of k-means clustering. Technical report, University of California, Berkeley, 2008.

Don Kurian Dennis, Tian Li, and Virginia Smith. Heterogeneity for the win: One-shot federated clustering. In *International Conference on Machine Learning (ICML)*, 2021.

Abdulrahman Diaa, Thomas Humphries, and Florian Kerschbaum. FastLloyd: Federated, accurate, secure, and tunable k-means clustering with differential privacy, 2024. URL https://arxiv.org/abs/2405.02437.

Ilias Diakonikolas, Daniel M. Kane, Daniel Kongsgaard, Jerry Li, and Kevin Tian. Clustering mixture models in almost-linear time via list-decodable mean estimation. In *Symposium on Theory of Computing (STOC)*, 2022.

Dimitrios Dimitriadis, Kenichi Kumatani, Robert Gmyr, Yashesh Gaur, and Sefik Emre Eskimez. A federated approach in training acoustic models. In *Interspeech*, 2020.

Frances Ding, Moritz Hardt, John Miller, and Ludwig Schmidt. Retiring adult: New datasets for fair machine learning. *Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

Max Dupré la Tour, Monika Henzinger, and David Saulpic. Making old things new: A unified algorithm for differentially private clustering. In *International Conference on Machine Learning (ICML)*, 2024.

Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming*, pp. 1–12, 2006.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam D. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference (TTC)*, 2006.

Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze Gauss: optimal bounds for privacy-preserving principal component analysis. In *Symposium on Theory of Computing (STOC)*, 2014.

Yan Gao, Titouan Parcollet, Salah Zaiem, Javier Fernández-Marqués, Pedro P. B. de Gusmao, Daniel J. Beutel, and Nicholas D. Lane. End-to-end speech recognition from federated acoustic models. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

Jonas Geiping, Hartmut Bauermeister, Hannah Dröge, and Michael Moeller. Inverting gradients – how easy is it to break privacy in federated learning? In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Avishek Ghosh, Jichan Chung, Dong Yin, and Kannan Ramchandran. An efficient framework for clustered federated learning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Filip Granqvist, Congzheng Song, Áine Cahill, Rogier van Dalen, Martin Pelikan, Yi Sheng Chan, Xiaojun Feng, Natarajan Krishnaswami, Vojta Jina, and Mona Chitnis. pfl-research: simulation framework for accelerating research in private federated learning, 2024. URL `https://arxiv.org/abs/2404.06430`.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction, 2019. URL `https://arxiv.org/abs/1811.03604`.

Samuel B. Hopkins and Jerry Li. Mixture models, robustness, and sum of squares proofs. In Ilias Diakonikolas, David Kempe, and Monika Henzinger (eds.), *Symposium on Theory of Computing (STOC)*, 2018.

Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *International Conference on Machine Learning (ICML)*, 2015.

Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14, 2021.

Gautam Kamath, Or Sheffet, Vikrant Singhal, and Jonathan R. Ullman. Differentially private algorithms for learning mixtures of separated gaussians. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Pravesh K. Kothari, Jacob Steinhardt, and David Steurer. Robust moment estimation and improved clustering via sum of squares. In *Symposium on Theory of Computing (STOC)*, 2018.

Amit Kumar and Ravindran Kannan. Clustering with spectral norm and the k-means algorithm. In *Foundations of Computer Science (FOCS)*, 2010.

Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020.

Zitao Li, Tianhao Wang, and Ninghui Li. Differentially private vertical federated clustering. *Proceedings of the VLDB Endowment*, 16(6):1277–1290, 2023.

Allen Liu and Jerry Li. Clustering mixtures with almost optimal separation in polynomial time. In *Symposium on Theory of Computing (STOC)*, 2022.

Yang Liu, Zhuo Ma, Zheng Yan, Zhuzhu Wang, Ximeng Liu, and Jianfeng Ma. Privacy-preserving federated k-means for proactive caching in next generation cellular networks. *Information Sciences*, 521:14–31, 2020.

Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory (TIT)*, 28(2):129–136, 1982.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

Frank McSherry. Differential privacy for measure concentration. `https://windowsontheory.org/2014/02/04/differential-privacy-for-measure-concentration/`, 2014. Accessed: 2024-09-23.

Payman Mohassel, Mike Rosulek, and Ni Trieu. Practical privacy-preserving k-means clustering. In *Privacy Enhancing Technologies Symposium*, 2020.

Ankur Moitra and Gregory Valiant. Settling the polynomial learnability of mixtures of Gaussians. In *Foundations of Computer Science (FOCS)*, 2010.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Conference on Neural Information Processing Systems (NeurIPS)*, pp. 849–856. MIT Press, 2001.

Karl Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London. A*, 185, 1894.

Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. Federated learning for emoji prediction in a mobile keyboard, 2019. URL `https://arxiv.org/abs/1906.04329`.

Oded Regev and Aravindan Vijayaraghavan. On learning mixtures of well-separated Gaussians. In *Foundations of Computer Science (FOCS)*, 2017.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

Jun Ren, Jinbo Xiong, Zhiqiang Yao, Rong Ma, and Mingwei Lin. DPLK-means: A novel differential privacy k-means mechanism. In *International Conference on Data Science in Cyberspace (DSC)*, 2017.

Nicola Rieke, Jonny Hancox, Wenqi Li, Fausto Milletarì, Holger R. Roth, Shadi Albarqouni, Spyridon Bakas, Mathieu N. Galtier, Bennett A. Landman, Klaus Maier-Hein, Sébastien Ourselin, Micah Sheller, Ronald M. Summers, Andrew Trask, Daguang Xu, Maximilian Baust, and M. Jorge Cardoso. The future of digital health with federated learning. *npj Digital Medicine*, 3(1):119, 2020.

Felix Sattler, Klaus-Robert Müller, and Wojciech Samek. Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, 32(8):3710–3722, 2020.

Jonathan Scott and Áine Cahill. Improved modelling of federated datasets using mixtures-of-Dirichlet-multinomials. In *International Conference on Machine Learning (ICML)*, 2024.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(8):888–905, 2000.

David Steurer and Stefan Tiegel. SoS degree reduction with applications to clustering and robust moment estimation. In Dániel Marx (ed.), *Symposium on Discrete Algorithms (SODA)*, 2021.

Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. Differentially private k-means clustering. In *ACM Conference on Data and Application Security and Privacy*, 2016.

Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, Min Lyu, and Hongxia Jin. Differentially private k-means clustering and a hybrid approach to private optimization. *ACM Transactions of Privacy and Security (TOPS)*, 20:1–33, 2017.

Kunal Talwar, Shan Wang, Audra McMillan, Vojta Jina, Vitaly Feldman, Pansy Bansal, Bailey Basile, Aine Cahill, Yi Sheng Chan, Mike Chatzidakis, Junye Chen, Oliver Chick, Mona Chitnis, Suman Ganta, Yusuf Goren, Filip Granqvist, Kristine Guo, Frederic Jacobs, Omid Javidbakht, Albert Liu, Richard Low, Dan Mascenik, Steve Myers, David Park, Wonhee Park, Gianni Parsa, Tommy Pauly, Christian Priebe, Rehan Rishi, Guy Rothblum, Michael Scaria, Linmao Song, Congzheng Song, Karl Tarbe, Sebastian Vogt, Luke Winstrom, and Shundong Zhou. Samplable anonymous aggregation for private federated data analysis, 2024. URL `https://arxiv.org/abs/2307.15017`.

Eliad Tsfadia, Edith Cohen, Haim Kaplan, Yishay Mansour, and Uri Stemmer. FriendlyCore: Practical differentially private aggregation. In *International Conference on Machine Learning (ICML)*, 2022.

Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang, and Hairong Qi. Beyond inferring class representatives: User-level privacy leakage from federated learning. In *IEEE Conference on Computer Communications (INFOCOM)*, 2019.

Chang Xia, Jingyu Hua, Wei Tong, and Sheng Zhong. Distributed k-means clustering guaranteeing local differential privacy. *Computers & Security*, 90:101699, 2020.

En Zhang, Huimin Li, Yuchen Huang, Shuangxi Hong, Le Zhao, and Congmin Ji. Practical multi-party private collaborative k-means clustering. *Neurocomputing*, 467:256–265, 2022.

A Extended Related work

Clustering Gaussian Mixture The problem of clustering Gaussian mixtures is a fundamental of statistics, perhaps dating back from the work of Pearson (1894).

Estimating the parameters of the mixture, as we are trying to in this paper, has a rich history. Moitra & Valiant (2010) showed that, even non-privately, the sample complexity has to be exponential in k; the standard way to bypass this hardness is to require some separation between the means of the different components. If this separation is $o(\sqrt{\log k})$, then any algorithm still requires a non-polynomial number of samples (Regev & Vijayaraghavan, 2017). When the separation is just above this threshold, namely $O(\log(k)^{1/2+c})$, Liu & Li (2022) present a polynomial-time algorithm based on Sum-of-Squares to recover the means of *spherical* Gaussians.

For clustering general Gaussians, the historical approach is based solely on statistical properties of the data, and requires a separation $\Omega(\sqrt{k})$ times the maximal variance of each component (Achlioptas & McSherry, 2005; Awasthi & Sheffet, 2012). This separation is necessary for accurate clustering, namely, if one aims at determining from which component each samples is from (Diakonikolas et al., 2022). This approach has been implemented privately by Kamath et al. (2019) (with the additional assumption that the input is in a bounded area): this is the one we follow, as the simplicity of the algorithms allows to have efficient implementation in a Federated Learning environment. Bie et al. (2022) studied how public data can improve performances of this private algorithm: they assume access to a small set of samples from the distribution, which improves the sample complexity and allows them to remove the assumption that the input lies in a bounded area.

We note that both private works of Kamath et al. (2019) and Bie et al. (2022) have a separation condition that grows with $\log n$, as ours.

To only recover the means of the Gaussians, and not the full clustering, a separation of k^α (for any $\alpha > 0$) is enough (Hopkins & Li, 2018; Kothari et al., 2018; Steurer & Tiegel, 2021). This is also doable privately (when additionally the input has bounded diameter) using the approach of Cohen et al. (2021) and Tsfadia et al. (2022). Those works are hard to implement efficiently in our FL framework for two reasons: first, they rely on Sum-of-Square mechanisms, which does not appear easy to implement efficiently. Second, they use Single Linkage as a subroutine: this does not seem possible to implement in FL. Therefore, some new ideas would be necessary to get efficient algorithm for FL based on this approach.

A different and orthogonal way of approaching the problem of clustering Gaussian mixtures is to recover a distribution that is ε-close to the mixture in total variation distance, in which case the algorithm of Ashtiani et al. (2020) has optimal sample complexity $\tilde{O}(kd^2/\varepsilon^2)$ – albeit with a running time $\omega(\exp(kd^2))$.

On Private k-means Clustering The private k-means algorithm of Dupré la Tour et al. (2024), implemented in our FL setting, would require either $\Omega(k)$ rounds of communication with the clients (for simulating their algorithm for central DP algorithm), or a a very large amount of additive noise $k^{O(1)}$ (for their local DP algorithm, with an unspecified exponent in k). Furthermore, the algorithm requires to compute a net of the underlying Euclidean space, which has size exponential in the dimension, and does not seem implementable. To the best of our knowledge, the state-of-the-art implementation of k-means clustering is from Chang & Kamath (2021): however, it has no theoretical guarantee, and is not tailored to FL.

B Technical preliminaries

B.1 Differential Privacy Definitions and Basics

As mentioned in introduction, one of the most important properties of Differential Privacy is the ability to compose mechanisms. There are two ways of doing so. First, parallel composition: if an (ε, δ)-DP algorithms is applied on two distinct datasets, then the union of the two output is also (ε, δ)-DP. Formally, the mechanism that takes as input two elements $P_1, P_2 \in \mathcal{P}$ and outputs $(\mathcal{A}(P_1), \mathcal{A}(P_2))$ is (ε, δ)-DP.

The second property is sequential composition: applying an (ε, δ)-DP algorithm to the output of another (ε, δ)-DP algorithm is $(2\varepsilon, 2\delta)$-DP. Formally: if $\mathcal{A} : \mathcal{P} \to \mathcal{S}_A$ is $(\varepsilon_A, \delta_A)$-DP and $\mathcal{B} : \mathcal{P} \times \mathcal{S}_A \to \mathcal{S}_B$ is $(\varepsilon_B, \delta_B)$-DP, then $\mathcal{B}(\mathcal{A}(\cdot), \cdot) : \mathcal{P} \to \mathcal{S}_B$ is $(\varepsilon_A + \varepsilon_B, \delta_A + \delta_B)$-DP.

Those are the composition theorem that we use for the theoretical analysis. However, in practice, better bounds can be computed – although they don't have closed-form expression. We use a standard algorithm to estimate more precise upper-bounds on the privacy parameters of our algorithms (Kairouz et al., 2015).

The sensitivity of a function is a key element to know how much noise is needed to add in order to make the function DP. Informally, the sensitivity measures how much the function can change between two neighboring datasets. Formally, we have the following definition.

Definition 3 (Sensitivity)**.** *Given a norm $\ell : \mathbb{R}^d \to \mathbb{R}$, the ℓ-sensitivity of a function $f : \mathcal{X}^n \to \mathbb{R}^d$ is defined as*

$$\sup_{x \sim X' \in \mathcal{X}^n} \ell(f(X) - f(X')),$$

where $X \sim X'$ means that X and X' are neighboring datasets.

The two most basic private mechanism are the Laplace and Gaussian mechanism, that make a query private by adding a simple noise. We use the Laplace mechanism for counting:

Lemma 4 (Laplace Mechanism for Counting.)**.** *Let X be a dataset. Then, the mechanism $M(X) = |X| + \mathrm{Lap}(1/\varepsilon)$ is $(\varepsilon, 0)$-DP, where $\mathrm{Lap}(1/\varepsilon)$ is a variable following a Laplace distribution with variance $1/\varepsilon$.*

We use the Gaussian mechanism for more general purposes (e.g., the PCA step). It is defined as follows:

Lemma 5. *Gaussian Mechanism Let $f : \mathcal{X} \to \mathbb{R}^n$ be a function with ℓ_2-sensitivity $\Delta_{f,2}$. Then, for $\sigma(\varepsilon, \delta) = \frac{\sqrt{2\log(2/\delta)}}{\varepsilon}$ the Gaussian mechanism $M(X) = f(X) + \mathcal{N}_d\left(0, \Delta_{f,2}^2 \sigma(\varepsilon, \delta)^2\right)$ is (ε, δ)-DP, where $\mathcal{N}_d(0, \sigma^2)$ is a d-dimensional Gaussian random variable, where each dimension is independent with mean 0 and variance σ^2.*

Combining those two mechanisms gives a private and accurate estimate for the average of a dataset

Lemma 6 (Private averaging)**.** *For dataset X in the ball $B(0, \Delta)$, the mechanism $M(X) := \frac{\sum_{x \in X} X + \mathcal{N}_d(0, \Delta_{f,2}\sigma^2(\varepsilon/2, \delta))}{|X| + \mathrm{Lap}(2/\varepsilon)}$ is (ε, δ)-DP. Additionally, $|X| \geq$, then it holds with probability $1 - \beta$ that $\|M(X) - \mu(X)\|_2 \leq \frac{\Delta \ln(2/\beta)}{|X|\varepsilon} + \frac{\Delta\sigma(\varepsilon/2, \delta)\sqrt{\ln(2/\beta)}}{|X|}$.*

B.2 Differential Privacy for Gaussian Mixtures

First, we review some properties of the private rank-k approximation: this algorithm was analyzed by Dwork et al. (2014), and its properties when applied on Gaussian mixtures by Kamath et al. (2019). The guarantee that is verified by the projection onto the noisy eigenvectors is the following:

Definition 7. *Fix a matrix $X \in \mathbb{R}^{n \times d}$, and let Π_k be the projection matrix onto the principal rank-k subspace of $X^T X$. For some $B \geq 0$, we say that a matrix Π is a B-almost k-PCA of X if Π is a projection such that:*

- $\|X^T X - (\Pi X)^T (\Pi X)\|_2 \leq \|X^T X - (\Pi_k X)^T (\Pi_k X)\|_2 + B$, *and*
- $\|X^T X - (\Pi X)^T (\Pi X)\|_F \leq \|X^T X - (\Pi_k X)^T (\Pi_k X)\|_F + kB$.

Dwork et al. (2014) shows how to compute a B-almost k-PCA, with a guarantee on B that depends on the diameter of the dataset:

Theorem 8 (Theorem 9 of Dwork et al. (2014))**.** *Let $X \in \mathbb{R}^{n \times d}$ such that $\|X_i\|_2 \leq 1$, and fix $\sigma(\varepsilon, \delta) = \sqrt{2\ln(2/\delta)}/\varepsilon$. Let $E \in R^{d \times d}$ be a symmetric matrix, where each entry $E_{i,j}$ with $j \geq i$ is an independent draw from $\mathcal{N}(0, \sigma(\varepsilon, \delta)^2)$. Let Π_k be the rank-k approximation of $X^T X + E$.*

Then, Π_k is a $O(\sqrt{d} \cdot \sigma(\varepsilon, \delta))$-almost k-PCA of X, and is (ε, δ)-DP.

Kamath et al. (2019) shows crucial properties of Gaussian mixtures: first, the projection of each empirical mean with a B-almost k-PCA is close to the empirical mean:

Lemma 9 (Lemma 3.1 in Kamath et al. (2019)). *Let $X \in \mathbb{R}^{n\times d}$ be a collection of points from k clusters centered at $\mu_1, ..., \mu_k$. Let C be the cluster matrix, namely $C_j = \mu_i$ if X_j belongs to the i-th cluster, and G_i be the i-th cluster.*

Let Π_k be a B-almost k-PCA, and denote $\bar{\mu_1}, ..., \bar{\mu_k}$ the empirical means of each cluster, and $\tilde{\mu_1}, ..., \tilde{\mu_k}$ the projected empirical means.

Then, $\|\bar{\mu_i} - \tilde{\mu_i}\| \leq \frac{1}{\sqrt{|G_i|}}\|X - C\|_2 + \sqrt{\frac{B}{|G_i|}}$.

Second – and this helps bounding the above – they provide bounds on the spectral norm of the clustering matrix $X - C$:

Lemma 10 (Lemma 3.2 in Kamath et al. (2019)). *Let $X \in \mathbb{R}^{n\times d}$ be a set of n samples from a mixture of k Gaussians. Let σ_i be the maximal unidirectional variance of the i-th Gaussian, and $\sigma_{max} = \max \sigma_i$. Let C be the cluster matrix, namely $C_j = \mu_i$ if X_j is sampled from $\mathcal{N}(\mu_i, \Sigma_i)$.*

If $n \geq \frac{1}{w_{min}}(\zeta_1 d + \zeta_2 \log_2(k/\beta))$, where ζ_1, ζ_2 are some universal constants, then with probability $1 - \beta$ it holds that

$$\frac{\sqrt{nw_{min}}\sigma_{max}}{4} \leq \|X - C\|_2 \leq 4\sqrt{n\sum_{i=1}^{k} w_i\sigma_i^2}.$$

B.3 Properties of Gaussian Mixtures

Lemma 11. *Consider a set P of n samples from a Gaussian mixtures $\{(\mu_i, \Sigma_i, w_i)\}_{i\in[k]}$. Let G_i be the set of points sampled from the i-th component. If $n \geq \frac{24\log(k)}{w_{min}}$, then with probability 0.99 it holds that $\forall i, |G_i| \geq nw_i/2$*

Proof. This is a direct application of Chernoff bounds: each sample s is in G_i with probability w_i. Therefore, the expected size of G_i is nw_i, and with probability at least $1 - 2\exp(-nw_i/12)$ it holds that $\big||G_i| - nw_i\big| \leq nw_i/2$: for $n \geq 24\log(k)/w_{\min}$, the probability is at least $1 - 2/k^2$. A union-bound over all i concludes. □

B.4 Clustering preliminaries

Our algorithm first replaces the full dataset P with a weighted version of Q, and then computes a k-means solution on this dataset. The next lemma shows that, if $\mathrm{cost}(P, Q)$ is small, then the k-means solution on the weighted Q is a good solution for P:

Lemma 12. *Let $P, C_1 \subset \mathbb{R}^d$, and $f : P \to C_1$ be a mapping with $\Gamma := \sum_{p\in P}\|p - f(p)\|^2$. Let w_ν be such that $|w_\nu - |f^{-1}(\nu)|| \leq |f^{-1}(\nu)|/2$. Let $\tilde{P}$ be the multiset where each $\nu \in C_1$ appears w_ν many times, . Let C_2 be such that $\mathrm{cost}(\tilde{P}, C_2) \leq \alpha \mathit{OPT}(\tilde{P})$. Then,*

$$\mathrm{cost}(P, C_2) \leq (2 + 12\alpha)\Gamma + 12\alpha \mathit{OPT}(P).$$

Proof. Recall that $C_2(p)$ is the closest point in C_2 to p. We have, using triangle inequality:

$$\begin{aligned}
\text{cost}(P, C_2) &= \sum_{p \in P} \|p - C_2(p)\|^2 \\
&\leq \sum_{p \in P} \|p - C_2(f(p))\|^2 \\
&\leq \sum_{p \in P} \left(\|p - f(p)\| + \|f(p) - C_2(f(p))\|\right)^2 \\
&\leq \sum_{p \in P} 2\|p - f(p)\|^2 + 2\|f(p) - C_2(f(p))\|^2 \\
&\leq 2\Gamma + 2 \sum_{\nu \in C_1} |f^{-1}(\nu)| \|\nu - C_2(\nu)\|^2 \\
&\leq 2\Gamma + 4 \sum_{\nu \in C_1} w_\nu \|\nu - C_2(\nu)\|^2 \\
&\leq 2\Gamma + 4\alpha \text{OPT}(\tilde{P}).
\end{aligned}$$

A similar argument bounds $\text{OPT}(\tilde{P})$: let C^* be the optimal solution for P, then, for any point p we have $\|f(p) - C^*(f(p))\| \leq \|f(p) - C^*(p)\| \leq \|f(p) - p\| + \|p - C^*(p)\|$. Therefore,

$$\begin{aligned}
\text{OPT}(\tilde{P}) &\leq \sum_{\nu \in C_1} w_\nu \|\nu - C*(\nu)\|^2 \\
&\leq \frac{3}{2} \sum_{\nu \in C_1} |f^{-1}(\nu)| \|\nu - C*(\nu)\|^2 \\
&\leq 3 \sum_{p \in P} 2\|C_1(p) - p\|^2 + 2\|p - C^*(p)\|^2 \\
&\leq 3\Gamma + 3\text{OPT}(P).
\end{aligned}$$

Combining those two inequalities concludes the lemma. □

C THE NON-PRIVATE, NON-FEDERATED ALGORITHM OF AWASTHI & SHEFFET (2012)

The algorithm we take inspiration from is the following, from Awasthi & Sheffet (2012) and inspired by Kumar & Kannan (2010): first, project the dataset onto the top-k eigenvectors of the dataset, and compute a constant-factor approximation to k-means (e.g., using local search). Then, improve iteratively the solution with Lloyd's steps. The pseudo-code of this algorithm is given in Algorithm 3, and the main result of Awasthi & Sheffet (2012) is the following theorem:

Theorem 13 (Awasthi & Sheffet (2012))**.** *For a separated Gaussian mixture, Algorithm 3 correctly classifies all point w.h.p.*

Their result is more general, as they do not require the input to be randomly generated, and only requires a strict separation between the clusters. In this paper, we focus specifically on Gaussian mixtures.

D OUR RESULT

Our main theoretical results is to adapt Algorithm 3 to a private and federated setting. We show the following theorem:

Theorem 14. *Suppose that the client dataset P is generated from a separated Gaussian mixtures with $n \geq \zeta_1 \frac{kdT \log^2 n \cdot \sqrt{\ln(1/\delta)}}{\varepsilon^2 w_{min}^2}$ samples, where ζ_1 is some universal constant, and that Q contains a least one sample from component of the mixture.*

Algorithm 3 Cluster(P)

1: **Part 1:** find initial Clusters
   a) Compute $\hat{P}$ the projection of P onto the subspace spanned by the top k singular vectors of P.
   b) Run a c-approximation algorithm for the k-means problem on $\hat{P}$ to obtain centers $\nu_1, ..., \nu_k$.
2: **Part 2:** For $r = 1, ...k$, set $S_r \leftarrow \{i : \forall s, \|\hat{P}_i - \nu_r\| \leq \frac{1}{3}\|\hat{P}_i - \nu_s\|\}$ and $\theta_r \leftarrow \mu(S_r)$
3: **Part 3:** Repeat Lloyd's steps until convergence:
   for $r = 1, ...k$, set $C(\nu_r) \leftarrow \{i : \forall s, \|P_i - \nu_r\| < \|P_i - \nu_s\|\}$, and $\theta_r \leftarrow \mu(C(\nu_r))$

Then, there is an (ε, δ)-DP algorithm that computes centers $\nu_1, ..., \nu_k$ such that, for some universal constants ζ_2, ζ_3, after $T + \zeta_2 \log \frac{\sigma_{max} \log |Q|}{\varepsilon w_{min}}$ rounds of communications, it holds with high probability that:

$$\|\mu_i - \nu_i\| \leq \zeta_3 \max\left(\frac{1}{2^T}, \frac{kdT \log^2 n\sigma_{max}\sqrt{\ln(T/\delta)}}{n\varepsilon^2 w_{min}^2}\right).$$

Note that the precision increases with the number of samples: if n is larger than $\frac{2^T \log(\sigma_{\max}/w_{\min}) kd \log^2 n\sigma_{\max}}{\varepsilon^2 w_{\min}^2}$, then the dominating term is $1/2^T$.

Corollary 15. *Suppose that the client dataset P is generated from a separated Gaussian mixtures with $n = \Omega\left(\frac{k \log^2 n\sqrt{d}\sigma_{max}}{\varepsilon^2 w_{min}^2}\right)$ samples, that Q contains a least one sample from component of the mixture and at most n data points.*

Suppose that $n = \Omega\left(\frac{k \log^3 n\sqrt{d}\sigma_{max}}{\varepsilon^2 w_{min}^2}\right)$, and that $n = \Omega\left(\frac{\log(n)^6 \cdot kd^2}{\varepsilon^4 w_{min}^2}\right)$.

Then, there is an (ε, δ)-DP algorithm with $O(\log(n))$ rounds of communications that computes centers $\nu_1, ..., \nu_k$ such that, with high probability, the clustering induced by $\nu_1, ..., \nu_k$ is the partition $G_1, ..., G_k$.

Proof. Theorem 5.4 of Kumar & Kannan (2010) (applied to Gaussian mixtures) bounds the number of misclassified points in a given cluster in terms of the distance between ν_i and μ_i. Define, for any i, S_i as the cluster of ν_i, and $\delta_i = \|\mu_i - \nu_i\|$. Then, for $j \neq i$, Kumar & Kannan (2010) show that, for some constant c':

$$|G_i \cap S_j| \leq \frac{c' n w_{\min}(\delta_i^2 + \delta_j^2)}{\|\mu_i - \mu_j\|^2}$$ [5]

Since $\|\mu_i - \mu_j\|^2 \geq c^2 \frac{k\sigma_{\max}^2 \log(n)^2}{w_{\min}}$, we get that the number of points from G_i assigned to cluster j is at most $\frac{c' n w_{\min}^2(\delta_i^2 + \delta_j^2)}{k\sigma_{\max}^2 \log(n)^2}$.

We aim at bounding δ_i and δ_j using Theorem 14. For $T = \log\left(\frac{10c' n w_{\min}}{k\sigma_{\max}}\right)$, it holds that $\frac{1}{2^T} \leq \frac{\sqrt{k}\sigma_{\max}}{10c'\sqrt{n} w_{\min}}$.

In addition, for this value of T and a number of samples n at least $n \geq \frac{100c'^2 \log(n)^2 \cdot kd^2 \log(n)^4}{\varepsilon^4 w_{\min}^2}$, we also have $\frac{kdT \log^2 n\sigma_{\max}\sqrt{\ln(T/\delta)}}{n\varepsilon^2 w_{\min}^2} \leq \frac{\sqrt{k}\sigma_{\max}}{10c'\sqrt{n} w_{\min}}$.

Therefore, the upper bound on δ_i and δ_j from Theorem 14 after $T + \log(\sigma_{\max} \log|Q|/w_{\min}) = O(\log(n))$ rounds of communications ensure that there is no point misclassified. This which concludes the statement. □

[5] We simplified the original statement of Kumar & Kannan (2010) to directly adapt it to separated Gaussian mixtures: in this case, $\|P - C\|_2^2 \leq 4n\sigma_{\max}^2$, and $\Delta_{i,j}$ (defined in the original statement) is our separation value, $c\sqrt{k/w_{\min}}\sigma_{\max} \log(n)$.

In the case where the assumption of Theorem 2 are satisfied, namely, (1) the diameter is bounded and (2) the server data are well spread, then the algorithm of Theorem 14 reduces directly to Algorithm 1 followed with T steps of Algorithm 2, with only T rounds of communication. Indeed, the first $O\left(\log \frac{\sigma_{\max} \log |Q|}{\varepsilon w_{\min}}\right)$ rounds of the algorithm from Theorem 14 are dedicated to enforcing condition (1) and (2): if they are given, there is no need for those steps.

The organization of the proof is as follows. First, we give some standard technical preliminary tools about differential privacy and Gaussian mixtures. Then, we show how to implement Algorithm 3: the bulk of the work is in the implementation of its Part 1, computing a good solution for ΠP. The second part to iteratively improve the solution is very similar to the non-private part.

E Part 1: Computing centers close to the means

E.1 Reducing the diameter

Lemma 16. *There is an ε-DP algorithm with one communication round that, given w_{min} and σ_{max}, reduces the diameter of the input to $O\left(\frac{\log |Q| \log n\sqrt{d}\sigma_{max}}{\varepsilon w_{min}}\right)$.*

Proof. We fix a distance $D = 4 \log n\sqrt{d}\sigma_{\max}$. First, the server identifies regions that contains many server points: if q is such that $|Q \cap B(q, D)| \geq \frac{\varepsilon n w_{\min}}{200 \log |Q|}$, then q is marked *frozen*.

Then, each client assigns its points to their closest server point in Q, breaking ties arbitrarily. In one round of communication, the server learns, for each server point $q \in Q$, the noisy number of points assigned to q, namely $\hat{w}_q(P) = w_q(P) + \mathrm{Lap}(1/\varepsilon)$. For privacy, the noise added to each count follows a Laplace distribution with parameter $1/\varepsilon$. Hence, with high probability, the noise is at most $O\left(\frac{\log |Q|}{\varepsilon}\right)$ on each server data q.

With high probability on the samples, for all i the $B(\mu_i, D)$ contains all the $w_i n$ samples from $\mathcal{G}_i$. Therefore, any server point q sampled from $\mathcal{G}_i$ is either frozen, or the noisy count in $B(q, D)$ ball is at least $nw_{\min}/2 - |Q \cap B(\mu_i, D)| \cdot \frac{\log |Q|}{\varepsilon} \geq nw_{\min}/3$, using Lemma 11.

Consider now an arbitrary point $p \in \mathbb{R}^d$. Since G_i is fully contained in $B(\mu_i, D/2)$, either the ball $B(p, D/2)$ doesn't intersect with G_i, or $B(p, D)$ contains entirely G_i. Furthermore, by triangle inequality, for any $q \in G_i \cap Q$ the ball $B(q, D)$ contains entirely G_i: if q is not frozen, it has noisy count at least $nw_{\min}/3$, and therefore true count at least $nw_{\min}/6$.

To reduce the diameter, we first remove all points from Q that are not frozen and for which the ball $B(q, D)$ has noisy count less than $nw_{\min}/3$: by the previous discussion, those points are not sampled from any $\mathcal{G}_i$ and are part of the noise. In addition, connect any pair of points that are at distance less than D.

We claim that each connected component has diameter at most $O\left(\frac{\log^2 n\sqrt{d}\sigma_{\max}}{\varepsilon w_{\min}}\right)$.

To prove this claim, we fix such a component, and consider the following iterative procedure. Pick an arbitrary point from the component, and remove all points that are at distance $2D$. Repeat those two steps until there are no more points.

Let q be a point selected at some step of this procedure. First, note that $B(q, D)$ is disjoint from any ball $B(q', D)$, for q' previously selected – as $B(q', 2D)$ has been removed. Furthermore, either q is frozen and the ball contains $\frac{\varepsilon n w_{\min}}{200 \log |Q|}$ many points of Q, or q is not frozen and $B(q, D)$ contains at least $nw_{\min}/6$ points of P. Therefore, there are at most $t_{max} := \frac{6}{w_{\min}} + \frac{200 \log |Q|}{\varepsilon w_{\min}}$ iterations. So the connected component can be covered with t_{max} balls of radius $2D$. Additionally, since each edge has length at most D, the component has diameter at most $O(t_{max} D) = O\left(\frac{\log |Q| \log n\sqrt{d}\sigma_{\max}}{\varepsilon w_{\min}}\right)$. This concludes the claim.

The other key property of the connected component is that each G_i is fully contained in a single connected component, as all points of G_i are at distance at most D of each other.

Therefore, we can transform the space such that the connected components get closer but do not interact, so that the diameter reduces while the centers of Gaussians are still far apart. Formally, let D' be the maximum diameter of the connected components. Select an arbitrary representative in Q from each connected component, and apply a translation to the connected component such that its representative has coordinate $(100D' \cdot i, 0, 0, ..., 0)$. This affine transformation ensures that (1) within each connected component, all means are still separated and the points are still drawn from Gaussian with the same covariance matrix and (2) the separation between centers of different component is at least $50D'$.

Therefore, the instance constructed still satisfy the separation conditions of Definition 1, and has diameter at most $O(kD') = O\left(\frac{k \log n \log |Q| \sqrt{d} \sigma_{\max}}{\varepsilon w_{\min}}\right)$ □

E.2 A RELAXATION OF AWATHI-SHEFFET'S CONDITIONS

The result of Awasthi & Sheffet (2012), applied to Gaussian, requires a slightly weaker separation between the centers than what we enforce. They consider a dataset P sampled from a Gaussian mixtures, and with cluster matrix C (namely, $C_i = \mu_i$ if P_i is sampled from the i-th component). They define for each cluster $\Delta_i^{AS} := \frac{1}{\sqrt{|G_i|}} \min(\sqrt{k}\|P-C\|_2), \|P-C\|_F)$, and require $\|\mu_i - \mu_j\| \geq c(\Delta_i^{AS} + \Delta_j^{AS})$ for some large constant c.

In the Gaussian setting, we have $|G_i| \approx n w_i$ (Lemma 11), $\|P - C\|_2 = O(\sigma_{\max}\sqrt{n})$ Lemma 10 and $\|A - C\|_F = \Theta(\sqrt{nd}\sigma_{\max})$. Thus, in most cases, $\min\left(\sqrt{k}\|P - C\|_2, \|P - C\|_F\right) = \sqrt{nk}\sigma_{\max} \operatorname{polylog}(d/w_{\min})$, except in some degenerate cases – and we keep the minimum only to fit with the proof of Awasthi & Sheffet (2012).

We can define $\Delta_i = \frac{\sigma_{\max}\sqrt{n}}{\sqrt{|G_i|}} \min\left(\sqrt{k}\operatorname{polylog}(d/w_{\min}), \sqrt{d}\right)$: our separation condition Definition 1 ensures that $\|\mu_i - \mu_j\| \geq c(\Delta_i + \Delta_j)$, for some large c. We now show the two key lemmas from Awasthi & Sheffet (2012), adapted to our private algorithm.

Fact 17 (Fact 1.1 in Awasthi & Sheffet (2012))**.** *Let $P \in \mathbb{R}^{n \times d}$ be a set of n points sampled from a Gaussian mixtures, and let C be the cluster matrix, namely $C_j = \mu_i$ if X_j is sampled from $\mathcal{N}(\mu_i, \Sigma_i)$. Let Π be a B-approximate k-PCA for $P_1, ..., P_n$. Suppose that B satisfies $B \leq \frac{\sqrt{n w_{min}}\sigma_{max}}{4k}$. Then:*

$$\|\Pi P - C\|_F^2 \leq 20 \min(k\|A - C\|_2^2), \|A - C\|_F^2) (= n w_i \Delta_i^2).$$

Proof. First, since both ΠP and C have rank k, it holds that $\|\Pi P - C\|_F^2 \leq 2k\|\Pi P - C\|_2^2$. By triangle inequality, this is at most $2k\left(\|\Pi P - P\|_2 + \|P - C\|_2\right)^2$.

Now, we have that $\|\Pi P - P\|_2^2 = \|(\Pi P - P)(\Pi P - P)^T\|_2$: since Π is a B-approximate k-PCA, this is at most $\|(\Pi_k P - P)(\Pi_k P - P)^T\|_2 + B$, where $\Pi_k P$ is the best rank-k approximation to P. By definition of Π_k, this is equal to $\|P - \Pi_k P\|_2^2 + B \leq \|P - C\|_2^2 + B$.

Overall, we get using $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$:

$$\begin{aligned}
\|\Pi P - C\|_F^2 &\leq 2k\left(\|\Pi P - P\|_2 + \|P - C\|_2\right)^2 \\
&\leq 2k\left(2\|P - C\|_2 + \sqrt{B}\right)^2 \\
&\leq 16k\|P - C\|_2^2 + 4kB.
\end{aligned}$$

Using Lemma 10 and the assumption that $4kB \leq \sqrt{n w_{\min}}\sigma_{\max}$ concludes the first part of the lemma.

For the other term, we have $\|\Pi P - C\|_F \leq \|\Pi P - P\|_F + \|P - C\|_F$. The fact that Π is a B-approximate k-PCA ensures that $\|\Pi P - P\|_F^2 \leq \|P - C\|_F^2 + kB$; and the fact that $\|P - C\|_F^2 \geq \|P - C\|_2^2 \geq \frac{n w_{\min}\sigma_{\max}^2}{16} \geq B$ concludes (where the second inequality is from Lemma 10). □

Fact 18. *[Analogous to Fact 1.2 in Awasthi & Sheffet (2012)] Let $P \in \mathbb{R}^{n \times d}$ be a Gaussian mixtures, and Π be a B-approximate k-PCA for $P_1, ..., P_n$. Suppose that B satisfies $B^2 \leq n w_{min}\sigma_{max}^2$. Let $S = \{\nu_1, ..., \nu_k\}$ be centers such that $\operatorname{cost}(\Pi P, S) \leq nk\sigma_{max}^2 \cdot \log^2 n$.*

Then, for each μ_i, there exists j such that $\|\mu_i - \nu_j\| \leq 6\Delta_i$, so that we can match each μ_i to a unique ν_j.

Proof. The proof closely follows the one in Awasthi & Sheffet (2012). Assume by contradiction that there is a i such that $\forall j, \|\mu_i - \nu_j\| > 6\Delta_i$. For any point $p \in P$, let ν_p be its closest center. Then, the contribution of the points in G_i to the cost is at least

$$\sum_{p \in G_i} \|\mu_i - \nu_p + \Pi p - \mu_i\|^2 > \frac{|G_i|}{2}(6\Delta_i)^2 - \sum_{p \in G_i} \|\Pi p - \mu_i\|^2 \geq 18|G_i|\Delta_i^2 - \|\Pi P - C\|_F^2,$$

where the first inequality follows from $(a-b)^2 \geq \frac{a^2}{2} - b^2$. Using first that $|G_i|\Delta_i^2 = 100nk\sigma_{\max}^2 \log^2(n)$, then Fact 17 combined with Lemma 10 yields that $\sum_{p \in G_i} \|\Pi p - \nu_p\|^2 > 1800nk\sigma_{\max}^2 \log^2(n) - 16nk\sigma_{\max}^2$ This contradicts the assumption on the clustering cost. □

Assuming there is a matching as in Fact 18, the proof of Awasthi & Sheffet (2012) directly goes through (when the Lloyd steps in Parts 2 and 3 of the algorithm are implemented non-privately), and we can conclude in that case that the clustering computed by Algorithm 1 is correct. Therefore, we first show that our algorithm computes a set of centers satisfying the conditions of Fact 18; and will show afterwards that the remaining of the proof works even with the addition of private noise.

E.3 Computing a good k-means solution for ΠP

The goal of this section is to show the following lemma:

Lemma 19. *There is an ε-DP algorithm with $10 \log \frac{4 \log |Q|}{\varepsilon w_{min}}$ rounds of communications that computes a k-means solution S with*

$$\text{cost}(\Pi P, S) = O\left(n \cdot \log^2\left(\frac{1}{\varepsilon w_{min}}\right) \cdot k\sigma_{max}^2 \log n\right).$$

The proof of this lemma is divided into several parts: first, we show that the means of the projected Gaussians $\Pi\mu_1, ..., \Pi\mu_k$ would be a satisfactory clustering. As points in Q are drawn independently from Π, there are points ΠQ close to each center $\Pi\mu_i$: our second step is therefore an algorithm that finds those points, in few communications rounds.

Lemma 20. *Let Π be the private projection computed by the algorithm. With high probability, clustering the projected set ΠG_i to the projected mean $\Pi\mu_i$ has cost $|G_i| \log n \cdot k\sigma_{max}^2$.*

Proof. We focus on a single Gaussian $\mathcal{G}_i$, and denote for simplicity $\mu := \mu_i$ its center and $\hat{\Sigma} := \Pi\Sigma_i$ the covariance matrix of $\Pi\mathcal{G}_i$. Standard arguments (see Proof of Corollary 5.15 in Kamath et al. (2019), or the blog post from McSherry (2014)) show that, with high probability, for all point it holds that $\|\Pi(p-\mu)\|_2^2 \leq \sqrt{k \log(n)}\sigma_{\max}$.

For a sketch of that argument, notice that if the projection Π was fixed independently of the samples, this inequality is direct from the concentration of Gaussians around their means, as the projection of $\mathcal{G}_i$ via Π is still a Gaussian, with maximal unidirectional variance at most $\sigma_{\max}$. This does not stay true when Π depends on the sample; however, since Π is computed via a private mechanism, the dependency between Π and any fixed sample is limited, and we can show the concentration.

Combined with the fact that there are $|G_i|$ samples from $\mathcal{G}_i$, this concludes. □

Lemma 19 in particular ensures that clustering ΠP to the full set ΠQ yields a cost $nk\sigma_{\max}^2 \cdot \log^2 n$. Therefore, if we could compute for each $q \in Q$ the size $w_q(\Pi P)$ of Πq's cluster in ΠP, namely, the number of points in ΠP closer to Πq than to any other point in ΠQ (breaking ties arbitrarily), then Lemma 12 would ensure that computing an $O(1)$-approximation to k-means on this weighted set yields a solution to k-means on ΠP with cost $O\left(nk\sigma_{\max}^2 \cdot \log^2 n\right)$.

However, the privacy constraint forbids to compute $w_q(\Pi P)$ exactly, and the server only receives a noisy version $\widehat{w_q(\Pi P)}$ – with a noise following a Laplace noise with parameter $1/\varepsilon$. Hence, for all points $q \in Q$, the noise added is at most $\frac{\log n}{\varepsilon}$ with high probability.

E.4 If assumption (2) is satisfied: the noise is negligible

Assumption (2) can be used to bound the total amount of noise added to the server data: we can show that the total contribution of the noise is small compared to the actual k-means cost, in which case solving k-means on the noisy data set yields a valid solution. We show the next lemma:

Lemma 21. *For any set of k centers S, it holds that* $\left|\sum_q w_q(\Pi P)\operatorname{cost}(p,S) - \sum_q \widehat{w_q(\Pi P)}\operatorname{cost}(p,S)\right| \leq \frac{|Q|\log|Q|\Delta^2}{\varepsilon}$

Proof.

$$\left|\sum_q w_q(\Pi P)\operatorname{cost}(q,S) - \sum_q \widehat{w_q(\Pi P)}\operatorname{cost}(q,S)\right| = \left|\sum_q \operatorname{Lap}(1/\varepsilon)\operatorname{cost}(q,S)\right|$$

With high probability, each of the $|Q|$ Laplace law is smaller than $\frac{\log|Q|}{\varepsilon}$. In this case, we get $\left|\sum_q \operatorname{Lap}(1/\varepsilon)\operatorname{cost}(q,S)\right| \leq \frac{|Q|\log|Q|\Delta^2}{\varepsilon}$ Therefore, the gap between the solution evaluated with true weight $w_q(\Pi P)$ and noisy weight $\widehat{w_q(\Pi P)}$ is at most $\frac{|Q|\log|Q|\Delta^2}{\varepsilon}$ □

Using $|Q| \leq n$, the assumption $|Q| \leq \frac{\varepsilon nk\sigma_{\max}^2}{\Delta^2}$ therefore ensures that the upper bound of the previous lemma is at most $nk\log(n)\sigma_{\max}^2$.

Hence, if S is a solution that has cost $O(1)$ times optimal on the noisy projected server data, it has cost $O(nk\sigma_{\max}^2\log(n))$ on the projected server data. Combining this result with Lemma 12 concludes: $\operatorname{cost}(\Pi P, S) = O\left(nk\sigma_{\max}^2\log(n)\right)$.

E.5 Enforcing Assumption (2)

In order to get rid of Assumption (2), we view the problem slightly differently: we will not try to reduce the number of points in Q to the precise upper-bound, but will nonetheless manage to control the noise and show Lemma 19.

Indeed, if all points of Q get assigned more than $2\log n/\varepsilon$ many input points, then the estimates of w_q are correct up to a factor 2, and Lemma 12 shows that a k-means solution S for the dataset consisting of ΠQ with the noisy weights satisfies $\operatorname{cost}(\Pi P, S) = O\left(nk\sigma_{\max}^2 \cdot \log^2 n\right)$. However, it may be that some points of Q get assigned less than $2\log n/\varepsilon$ points, in which case the noise would dominate the signal and Lemma 12 becomes inapplicable. Our first goal is therefore to preprocess the set of hings Q to get $\hat{Q}$ such that :

1. for each cluster, $\Pi\hat{Q}$ still contains one good center, and
2. $\forall q \in \hat{Q}, \hat{w}_q \geq 2\log n/\varepsilon$ (where the weight $\hat{w}$ is computed by assigning each data point to its closest center of $\hat{Q}$)

The first item ensures that $\operatorname{cost}(\Pi P, \Pi\hat{Q}) = O\left(nk\sigma_{\max}^2 \cdot \log^2 n\right)$; the second one that the size of each cluster is well approximated, even after adding noise.

Our intuition is the following. Removing all points $q \in Q$ with estimated weight less than $2\log n/\varepsilon$ is too brutal: indeed, it may be that one cluster is so over-represented in Q that all its points get assigned less than $2\log n/\varepsilon$ points from P. However, in that case, there are many points in the cluster and in ΠQ: we can therefore remove each point with probability $1/2$ and preserve (roughly) the property that there is a good center in ΠQ. Repeating this intuition, we obtain the algorithm described in Algorithm 4.

We sketch briefly the properties of algorithm 4, before diving into details of the proof. First, the algorithm is ε-DP, as each of the T steps is ε/T-DP.

Then, points in F are *frozen*: even after adding noise, their weight is well approximated. We will show by induction on the time t that, for any cluster i that does not contain any frozen point at time t, then $Q_t \cap B(\mu_i, 2t\cdot\sqrt{k\log n}\sigma_{\max})$ contains many points: more precisely, $|Q_t \cap B(\mu_i, 2t\cdot$

Algorithm 4 SimplifyServerData

1: **Input:** Server data Q, client datasets $P^1, ..., P^m$, a projection matrix Π computed from $P^1, ..., P^m$, and privacy parameter ε
2: Let $F \leftarrow \emptyset, Q_0 \leftarrow Q, T = 10 \log\left(\frac{4 \log |Q|}{\varepsilon w_{\min}}\right)$
3: **for** $t = 0$ to T **do**
4: Let $C = F \cup Q_t$
5: for each $q \in C$, the server receives a noisy estimate $\hat{w}_q^{(t)}$ of $w_{\Pi q}(\Pi Q_t)$, with noise $\mathrm{Lap}(T/\varepsilon)$.
6: Server computes $L := \{q \in C : \hat{w}_q^{(t)} \leq 2 \log n/\varepsilon\}$.
7: $F \leftarrow F \cup (Q_t \setminus L)$.
8: Server computes Q_{t+1}, a subset of L where each point is sampled with probability $1/2$.
9: **end for**
10: **Return:** F

$\sqrt{k \log n}\sigma_{\max})| \geq \varepsilon |G_i|/2$. Since at each time step only half of the points in L are preserved in Q_{t+1} (line 7 of the algorithm), it implies that, at the beginning, $|Q \cap B(\mu_i, 2t \cdot \sqrt{k \log n}\sigma_{\max})| \gtrsim 2^t \varepsilon/T |G_i|$. Therefore, for $t = \log(1/(\varepsilon w_{\min}))$, we have for each cluster that either it contains a frozen point, or $|Q \cap B(\mu_i, 2t \cdot \sqrt{k \log n}\sigma_{\max})| \geq \frac{|G_i|}{w_{\min}} > n$: as the second option is not possible, all clusters contains a frozen point, which is a good center for that cluster.

Our next goal is to formalize the argument above, and show:

Lemma 22. *Let F be the output of Algorithm 4. Then, for each cluster i, there is a point $\nu_i \in F$ such that $\|\Pi(\mu_i - \nu_i)\| \leq \log\left(\frac{4 \log |Q|}{\varepsilon w_{min}}\right) \cdot \sqrt{k \log n}\sigma_{max}$.*

Furthermore, for each $q \in F$, define w_q as the number of points closest to q than any other point in F : it holds that $w_q \geq 2 \log n/\varepsilon$.

For simplicity, we define $\Delta_C := \sqrt{k \log n}\sigma_{\max}$. To prove this lemma, we show inductively that after t iterations of the loop in the algorithm, then either $B(\Pi\mu_i, 2t\Delta_C)$ contains a frozen point, or $|B(\Pi\mu_i, (t+1)\Delta_C) \cap \Pi Q_t| \geq \varepsilon |G_i|/2$. Since the number of points in ΠQ_t is divided by roughly 2 at every time step, the latter condition implies that there was initially at least $\approx 2^t \varepsilon |G_i|$ points in $B(\Pi\mu_i, (t+1)\Delta_C) \cap \Pi Q$. For $t \approx \log(1/(\varepsilon w_{\min})$, this is bigger than n and we get a contradiction: the ball contains therefore a frozen point.

Our first observation to show this claim is that many points of P are close to μ_i:

Fact 23. *With high probability on the samples, it holds that $\left|B(\Pi\mu_i, \sqrt{k \log n}\sigma_{max}) \cap \Pi P_i\right| \geq |G_i|$*

Proof. As in the proof of Lemma 20, the fact that Π is computed privately ensures that, with high probability, all points $p \in G_i$ satisfy $\|\Pi(p - \mu_i)\| \leq \sqrt{k \log n}\sigma_{\max}$. Thus, $\left|B(\Pi\mu_i, \sqrt{k \log n}\sigma_{\max}) \cap \Pi P_i\right| \geq |G_i|$. □

For the initial time step $t = 0$ we actually provide a weaker statement to initialize the induction, and show that there is at least one point in $B(\Pi\mu_i, \sqrt{k \log n}\sigma_{\max}) \cap \Pi Q_t$. This will be enough for the induction step.

Fact 24 (Initialization of the induction)**.** *With high probability, $\exists q \in Q, \|\Pi(\mu_i - q)\| \leq \sqrt{k \log n}\sigma_{max}$.*

Proof. This directly stems from the fact that there is some point $q \in Q$ that is sampled according to $\mathcal{G}_i$, and that Π is independent of that point. Therefore, Πq follows the Gaussian law $\Pi\mathcal{G}_i$, which is in a k dimensional space and has maximal unidirectional variance $\sigma_{\max}$. Concentration of Gaussian random variables conclude. □

To show our induction, the key lemma is the following:

Lemma 25. *After t iteration of the loop, either $B\left(\Pi\mu_i, (t+1)\sqrt{k \log n}\sigma_{max}\right)$ contains a frozen point, or $\left|\Pi Q_t \cap B(\Pi\mu_i, 2(t+1)\sqrt{k \log n}\sigma_{max})\right| \geq \frac{\varepsilon |G_i|}{4T \log(T|Q|)}$.*

Proof. Let $\Delta_C := \sqrt{k \log n}\sigma_{\max}$.

First, it holds with high probability that all the noise added Line 5 satisfy $\left|\hat{w}_q^{(t)} - w_{\Pi q}(\Pi Q_t)\right| \leq T \log(T|Q|)/\varepsilon$. This directly stems from concentration of Laplace random variables, and the fact that there are $T|Q|$ many of them.

We prove the claim by induction. Fix a $t \geq 0$. The induction statement at time t ensures that either there is a point frozen in $B(\Pi\mu_i, (t+1)\Delta_C)$, in which case we are done, or there is at least one point in $\Pi Q_t \cap B(\Pi\mu_i, (t+1)\Delta_C)$ (note that this statement holds for $t = 0$ by Fact 24).

By triangle inequality, this means that all points of $\Pi G_i \cap B(\Pi\mu_i, \Delta_C)$ are assigned at time $t+1$ to a point in $B(\Pi\mu_i, (t+2)\Delta_C)$ (in line 4 of Algorithm 4). Therefore, by Fact 23, $\sum_{q:\Pi q \in \Pi Q_t \cap B(\Pi\mu_i,(t+2)\Delta_C)} w_q^t \geq |G_i|$.

Then, either ΠQ_t contains less than $\frac{\varepsilon|G_i|}{2T\log(T|Q|)}$ many points from $B\left(\Pi\mu_i, (t+2)\Delta_C\right)$, and we are done, as one of them must have $w_{\Pi q}(\Pi Q_{t+1}) \geq 2T\log(|Q|T)/\varepsilon$ and will be frozen – as in this case $\hat{w}_q^{(t)} \geq T\log(|Q|T)/\varepsilon$. Or, there are more than $\frac{\varepsilon|G_i|}{2T\log(T|Q|)}$ points, and they all have $w_q^{t+1} \leq 2T\log(T|Q|)/\varepsilon$: Chernoff bounds ensure that, with high probability, at least $\frac{\varepsilon|G_i|}{4T\log(T|Q|)}$ will be sampled in the set Q_{t+1}, which concludes the lemma. □

Lemma 22 is a mere corollary of those results:

Proof of Lemma 22. Again, we define $\Delta_C := \sqrt{k \log n}\sigma_{\max}$. At the end of Lemma 22, all points in F are frozen: let $f : P \to F$ such that $f(p) = \arg\min_{q\in F} \|\Pi(p-q)\|$, breaking ties arbitrarily. Since all points are frozen, it holds that for all q, $|f^{-1}(q)| \geq 2\log n/\varepsilon$: therefore, their noisy weight $\hat{w}_q$ satisfy $|\hat{w}_q - |f^{-1}(q)| \leq \frac{|f^{-1}(q)|}{2}$.

Furthermore, for T large enough it holds that $T \geq \log\left(\frac{4T\log(T|Q|)}{\varepsilon w_{\min}}\right)$: this holds e.g. for $T = 10\log\left(\frac{4\log(|Q|)}{\varepsilon w_{\min}}\right)$.

Lemma 25 ensures that either $B(\Pi\mu_i, (T+1)\Delta_C)$ contains a frozen point, or $|\Pi Q_T \cap B(\Pi\mu_i, 2(T+1)\Delta_C)| \geq \frac{\varepsilon|G_i|}{4T\log(T|Q|)}$.

Suppose by contradiction that we are in the latter case. Since, at each time step, every point in Q is preserved with probability $1/2$, it holds with high probability that $|\Pi Q \cap B(\Pi\mu_i, 2(T+1)\Delta_C)| \geq \varepsilon 2^T \cdot |G_i|$. Indeed, all points of that ball are still present in Q_T with probability $1/T^t$: Chernoff bounds ensure that there must be initially at least $2^T \cdot \frac{\varepsilon|G_i|}{4T\log(T|Q|)}$ points in that ball in order to preserve $\frac{\varepsilon|G_i|}{4T\log(T|Q|)}$ of them after the sampling. With our choice of T, this means $|\Pi Q \cap B(\Pi\mu_i, 2(t+1)\Delta_C)| > |Q|$, which is impossible.

Therefore, it must be that $B(\Pi\mu_i, (T+1)\Delta_C)$ contains a frozen point, which concludes the proof. □

Proof of Lemma 19 We now have all the ingredients necessary to the proof of Lemma 19. The algorithm is a mere combination of the previous results:

- Use Algorithm 4 to compute a set F.
- Server sends F to the clients, who define $f : P \to F$ such that $f(p) = \arg\min_{q\in F} \|\Pi(p-q)\|$, breaking ties arbitrarily.
- Client i sends $w_{\Pi q}(\Pi P^i) := \left|\{p \in P^i : f(p) = q\}\right|$.
- Server receives $\hat{w}_q$, a noisy version of $w_q := \sum_i w_q^i$.
- Server computes an $O(1)$-approximation S to k-means on the dataset ΠF with weights $\hat{w}_q$.

To show that S has the desired clustering cost, we aim at applying Lemma 12. For this, we first bound $\sum_p \|\Pi(p-f(p))\|^2$. For each cluster i, let ν_i be the point from F as defined in Lemma 22. We have, using the definition of f and triangle inequality:

$$\sum_p \|\Pi(p-f(p))\|^2 \leq \sum_i \sum_{p\in G_i} \|\Pi(p-\nu_i)\|^2 \leq 2\sum_i \sum_{p\in G_i} \|\Pi(p-\mu_i)\|^2 + \|\Pi(\mu_i-\nu_i)\|^2.$$

From Lemma 20, we know that $\sum_i \sum_{p\in G_i} \|\Pi(p-\mu_i)\|^2 = O(n\log n \cdot k\sigma_{\max}^2)$. The guarantee of ν_i in Lemma 22 ensures $\sum_i \sum_{p\in G_i} \|\Pi(\mu_i-\nu_i)\|^2 = O\left(n\cdot\log^2\left(\frac{1}{\varepsilon w_{\min}}\right)\cdot k\sigma_{\max}^2\log n\right)$.

Thus, $\sum_p \|\Pi(p-f(p))\|^2 = O\left(n\cdot\log^2\left(\frac{1}{\varepsilon w_{\min}}\right)\cdot k\sigma_{\max}^2\log n\right)$.

Since all points in F have an estimated that satisfies $\left|\hat{w}_q - |f^{-1}(q)|\right| \leq \frac{|f^{-1}(q)|}{2}$, we can apply Lemma 12: the solution computed by the above algorithm on the dataset ΠF with weights $\hat{w}_q$ has cost at most $O\left(n\cdot\log^2\left(\frac{1}{\varepsilon w_{\min}}\right)\cdot k\sigma_{\max}^2\log n\right) + O(\text{OPT}(\Pi P)) = O\left(n\cdot\log^2\left(\frac{1}{\varepsilon w_{\min}}\right)\cdot k\sigma_{\max}^2\log n\right)$.

This concludes the proof of Lemma 19.

F Part 2: Improving iteratively the solution

Our global algorithm is described in Algorithm 5: first, we use Lemma 16 to reduce the diameter of the input; then, we compute a good initial solution using Lemma 19. Then, we implement privately Part 2 and Part 3 of Algorithm 3, using private mean estimation.

Algorithm 5 Cluster

1: **Input:** Server data Q, client datasets $P^1,...,P^m$, and privacy parameters ε,δ
2: Process the input to reduce the diameter to Δ using Lemma 16, with privacy parameter $\varepsilon/4$.
3: In one round of communication, compute a $O(\sqrt{d}\Delta\cdot\sigma(\varepsilon/4,\delta))$-almost k-PCA using Theorem 8.
4: **Part 1:** find initial centers $\nu_1^{(1)},...,\nu_k^{(1)}$ using Lemma 19, with privacy parameter $\varepsilon/4$
5: **Part 2:**
   a) Server sends $\nu_1^{(1)},..,\nu_k^{(1)}$ to clients, and client c computes $S_r^c := \{P_i\in P^c : \forall s, \|\hat{P}_i-\nu_r\| \leq \frac{1}{3}\|\hat{P}_i-\nu_s\|\}$.
   b) Server receives, for all cluster r, $\nu_r^{(2)} := \frac{1}{\sum_{\text{client } c}|S_r^c|+\text{Lap}(T/\varepsilon)}\left(\sum_{\text{client } c}\sum_{P_i\in S_r^c} P_i + \mathcal{N}_d\left(0,\frac{2T^2\Delta\log(2T/\delta)}{\varepsilon^2}\right)\right)$
6: **Part 3:** Repeat Lloyd's steps for T steps, with privacy parameter $(\varepsilon/T,\delta/T)$:
   a) Server sends $\nu_1^{(t)},..,\nu_k^{(t)}$ to clients, and client c computes $S_r^c := \{P_i\in P^c : \forall s, \|\hat{P}_i-\nu_r\| \leq \|\hat{P}_i-\nu_s\|\}$.
   b) Server receives, for all cluster r, $\nu_r^{(t+1)} := \frac{1}{\sum_{\text{client } c}|S_r^c|+\text{Lap}(T/\varepsilon)}\left(\sum_{\text{client } c}\sum_{P_i\in S_r^c} P_i + \mathcal{N}_d\left(0,\frac{2T^2\Delta\log(2T/\delta)}{\varepsilon^2}\right)\right)$

Given the mapping of Fact 18, the main result of Awasthi & Sheffet (2012) is that step 2 of the algorithm computes centers that are very close to the μ_i:[6]

Theorem 26 (Theorem 4.1 in Awasthi & Sheffet (2012))**.** *Suppose that the solution $\nu_1,...,\nu_k$ is as in Fact 18, namely, for each μ_i, it holds that $\|\mu_i-\nu_i\| \leq 6\Delta_i$. Denote $S_i = \{j : \forall r\neq i, \|\Pi P_j-\nu_i\| \leq$*

[6]Note that the original theorem of Awasthi & Sheffet (2012) is stated slightly differently: however, their proof only requires Fact 17 and the matching provided by Fact 18, and we modified the statement to fit our purposes.

$\frac{1}{3}\|\Pi P_j - \nu_r\|\}$. *Then, for every* $i \in [k]$ *it holds that*

$$\|\mu(S_i) - \mu_i\| = O\left(\frac{1}{c\sqrt{|G_i|}} \cdot \|P - C\|_2\right),$$

where c *is the separation constant from Definition 1.*

Finally, the next result from Kumar & Kannan (2010) shows that the Lloyd's steps converge towards the true means:

Theorem 27 (theorem 5.5 in Kumar & Kannan (2010))**.** *If, for all* i *and a parameter* $\gamma \leq ck/50$*,*

$$\|\mu_i - \nu_i\| \leq \frac{\gamma\|P - C\|_2}{\sqrt{|G_i|}},$$

then

$$\|\mu_i - \mu(C(\nu_i))\| \leq \frac{\gamma\|P - C\|}{2\sqrt{|G_i|}},$$

where $C(\nu_i)$ *is the set of points closer to* ν_i *than to any other* ν_j*.*

This allows us to conclude the accuracy proof of Theorem 14

Proof of Theorem 14. The algorithm is (ε, δ)-DP: each of the 4 steps step – reducing the diameter, computing a PCA, finding a good initial solution and running T Lloyd's steps – is $(\varepsilon/4, \delta/4)$-DP, and private composition concludes.

The first three steps require a total of $2 + 10\log\frac{4\log|Q|}{\varepsilon w_{\min}}$ many rounds of communication, the last one requires $T + \log\frac{\sigma_{\max}^2}{w_{\min}}$ rounds. This simplifies to $T + \zeta_2 \log\frac{\sigma_{\max}\log|Q|}{\varepsilon w_{\min}}$, for some constant ζ_2.

The first step reduces the diameter to $\Delta = O\left(\frac{k\log^2 n\sqrt{d}\sigma_{\max}}{\varepsilon w_{\min}}\right)$; therefore, Lemma 19 combined with Fact 18 ensures that $\nu_1^{(1)}, ..., \nu_k^{(1)}$ satisfies the condition of Theorem 26. In addition, Lemma 4.2 of Awasthi & Sheffet (2012) ensures that the size of each cluster $|S_r|$ is at least $\frac{|G_i|}{2}$ at every time step.

Therefore, the private noise $\frac{\mathcal{N}_d\left(\Delta^2\sigma^2(\varepsilon',\delta')\right)}{|S_r^c|}$ is bounded with high probability by $\eta := O\left(\frac{\Delta\sqrt{d}\sigma(\varepsilon/T,\delta/T)}{|S_r^c|}\right) = O\left(\frac{kdT\log^2 n\sigma_{\max}\cdot\sqrt{\ln(1/\delta)}}{n\varepsilon^2 w_{\min}^2}\right)$, which for and $n = \Omega\left(\frac{kdT\log^2 n\cdot\sqrt{\ln(1/\delta)}}{\varepsilon^2 w_{\min}^2}\right)$ is smaller than $\Delta_i = \frac{\sigma_{\max}}{\sqrt{w_i}}\min\left(\sqrt{k}\,\mathrm{polylog}(d/w_{\min}), d\right)$.

Hence, the conditions of Theorem 26 and Theorem 27 are still satisfied after adding noise, and the latter implies that the noisy Lloyd steps converge exponentially fast towards $B(\mu_i, \eta)$.

More precisely, it holds with probability $1 - 1/k^2$ that $\left\|\mu_i - \nu_i^{T+\log\frac{\sigma_{\max}^2}{w_{\min}}}\right\| = O\left(\frac{1}{c2^{T+\log\frac{\sigma_{\max}^2}{w_{\min}}}}\right) \cdot \frac{\|P-C\|_2}{\sqrt{|G_i|}} + \eta$.

From Lemma 10 ensures $\|P - C\| \leq O(\sqrt{n}\sigma_{\max})$. Since $|G_i| \geq nw_{\min}/2$, the first term is at most $O\left(\frac{1}{2^T}\right)$.

Therefore,

$$\left\|\mu_i - \nu_i^{T+\log\frac{\sigma_{\max}^2}{w_{\min}}}\right\| = O\left(\max\left(\frac{1}{2^T}, \frac{kdT\log^2 n\sigma_{\max}\sqrt{\ln(T/\delta)}}{n\varepsilon^2 w_{\min}^2}\right)\right).$$

□

G EXPERIMENT DETAILS

G.1 DATASET DETAILS

Mixture of Gaussians Datasets We generate a mixture of Gaussians in the following way. We set the data dimension to $d = 100$ and we generate $k = 10$ mixtures by uniformly randomly sampling k means $\{\mu_1, \dots \mu_k\}$ from $[0, 1]^d$. Each mixture has diagonal covariance matrix $\Sigma_i = 0.5 I_d$ and equal mixture weights $w_i = 1/k$. The server data is generated by combining samples from the true mixture distribution together with additional data sampled uniformly randomly from $[0, 1]^d$ representing related but out-of-distribution data. We sample 20 points from each mixture component, for a total of $20 \times k = 200$ in distribution points and sample an additional 100 uniform points. For Section 5.1 we simulate a cross-silo setting with 100 clients, with each client having 1000 datapoints sampled i.i.d from the Gaussian mixture. For Section 5.2 we simulate a cross-device setting with 1000, 2000 and 5000 clients, each client having 50 points i.i.d sampled from the Gaussian mixture distribution. The server data is identical in both cases.

US Census Datasets We create individual datapoints coming from the ACSIncome task in folktables. Thus each datapoint consists of $d = 819$ binary features describing an individual in the census, including details such as employment type, sex, race etc. In order to create a realistic server dataset (of related but not not in-distribution data) we filter the client datasets to contain only individuals of a given employment type. The server then receives a small amount (20) of datapoints with the chosen employment type, and a larger amount (1000) of datapoints sampled i.i.d from the set of individuals with a different employment type. We do this for 3 different employment types, namely "Employee of a private not-for-profit, tax-exempt, or charitable organization", "Federal government employee" and "Self-employed in own not incorporated business, professional practice, or farm". These give us three different federated datasets, each with 51 clients, with total dataset sizes of 127491, 44720 and 98475 points respectively.

Stack Overflow Datasets Each client in the dataset is a stackoverflow user, with the data of each user being the questions they posted. Each question also has a number of tags associated with it, describing the broad topic area under which the question falls. We first preprocess the user questions by embedding them using a pre-trained sentence embedding model (Reimers & Gurevych, 2019). Thus a user datapoint is now a $d = 384$ text embedding. Now we again wish to create a scenario where the server can receive related but out of distribution data. We follow a similar approach to the creation of the US census datasets. We select two tag topics and filter our clients to consist of only those users that have at least one question that was tagged with one of the selected topics. For those clients we retain only the questions tagged with one of the chosen topics. The server then receives 1000 randomly sampled questions with topic tags that do not overlap with the selected client tags as well as 20 questions with the selected tags, 10 of each one. For our experiments we use the following topic tag pairs to create clients [(machine-learning, math), (github, pdf), (facebook, hibernate), (plotting, cookies)]. These result in federated clustering problems with $[10394, 9237, 23266, 2720]$ clients respectively.

G.2 VERIFYING OUR ASSUMPTIONS

On each of the datasets used in our data-point-level experiments we compute the radius of the dataset Δ, shown in Table 1.

Dataset	Δ
Gaussian Mixture (100 clients)	10.57
US Census (Not-for-profit Employees)	2.65
US Census (Federal Employees)	2.65
US Census (Self-Employed)	2.65

Table 1: Radius of each dataset.

Assumption (1) requires $\Delta = O\left(\frac{k \log^2(n) \sigma_{\max} \sqrt{d}}{\varepsilon w_{\min}}\right)$. For the Gaussian mixture, $k = 10, d = 100, w_{\min} = 1/10, n = 10^6$ and $\sigma_{\max} = 0.5$: thus Δ clearly satisfies the condition.

For the US Census datasets, $k = 10, d = 819, n \in \{127491, 44720, 98475\}$. As we cannot estimate $\sigma_{\max}$ and $w_{\min}$(since the dataset is not Gaussian), we use an upper-bound $w_{\min} = 1$, and replace $\sigma_{\max}$ with a proxy based on the optimal k-means cost, $\sqrt{\text{OPT}/n}$: this is a priori a large upper-bound on the value of $\sigma_{\max}$, but it still gives an indication on the geometry of each cluster. As can be seen in Figure 1, Figure 3, the average optimal cost is about 3.5 : thus, $\sqrt{\text{OPT}/n} \approx 1.87$, and we estimate $\frac{k \log^2(n) \sigma_{\max} \sqrt{d}}{\varepsilon w_{\min}} \approx \frac{10 \cdot \log^2(10^5) \cdot 0.005 \cdot \sqrt{819}}{0.5} \approx 123000$. This indicates that Condition (1) is satisfied as well for this dataset.

Assumption 2 requires that the size of the server data is not too large: $|Q| \leq \frac{\varepsilon n k \log(n) \sigma_{\max}^2}{\Delta^2}$. In the Gaussian case, we have $|Q| = 300$, and the right-hand-side is about 29000.

In the US Census Dataset, we again upper-bound $\sigma_{\max}^2 = \frac{\text{OPT}}{n}$. In that case, the right-hand-side is about 620000, while there are 1020 server point. Although our estimate of $\sigma_{\max}$ is only an upper-bound, this indicates that assumption (2) is also satisfied.

G.3 Baseline Implementation Details

SpherePacking We implement the data independent initialization described in Su et al. (2017) as follows. We estimate the data radius Δ using the server dataset. We set $a = \Delta\sqrt{d}$, for $i = 1, \ldots, k$, we randomly sample a center ν_i in $[-\Delta, \Delta]^d$. If ν_i is at least distance a from the corners of the hypercube $[-\Delta, \Delta]^d$ and at least distance $2a$ away from all previously sampled centers $\nu_1, \ldots, \nu_{i-1}$, then we keep it. If not we resample ν_i. We allow 1000 attempts to sample ν_i, if we succeed with sampling all k centers then we call the given a feasible. If not then a is infeasible. We find the largest feasible a by binary search and use the corresponding centers as the initialization.

G.4 Setting hyperparameters of FedDP-KMeans

In this section we analyze the hyperparameter settings of FedDP-KMeans that produced the Pareto optimal results shown in the figures in Sections 5.1 and 5.2. These analyses give us some insights on the optimal ways to set the hyperparameters when using FedDP-KMeans in practice.

Distributing the privacy budget The most important parameters to set are the values of epsilon in Parts 1-3 of Algorithm 1. Here we discuss how to set these.

Let ε_1, ε_2, ε_{3G} and ε_{3L} denote the epsilon we allow for part 1, part 2, the Gaussian query in part 3 and the Laplace query in part 3 respectively. We let $\varepsilon_{\text{init}} = \varepsilon_1 + \varepsilon_2 + \varepsilon_{3G} + \varepsilon_{3L}$. By strong composition the initialization will have a lower overall budget than $\varepsilon_{\text{init}}$, however, it serves as a useful proxy to the overall budget as we can think of what proportion of $\varepsilon_{\text{init}}$ we are assigning to each step.

Shown in Tables 2 and 3 are the values from our experiments. Specifically, for each dataset we take the mean across the Pareto optimal results that we plotted of the ε values used for each step. We then express this as a fraction of $\varepsilon_{\text{init}}$. Loosely speaking, we interpret these values as answering "What fraction of our overall privacy budget should we assign to each step?"

The results paint a consistent picture when comparing values with the same unit-level of privacy with slight differences between the two levels. For datapoint level privacy, clearly the most important step in terms of assigning budget is to the Gaussian mechanism in Step 3 with the other steps being roughly even in term of importance. Therefore, as a rule of thumb we would recommend assigning budget using the following approximate proportions $[0.2, 0.2, 0.45, 0.15]$. For user level privacy we observe the same level of importance being placed on the Gaussian mechanism in Step 3 but additionally on the Gaussian mechanism in Step 1. Based on these results we would assign budget following approximate proportions $[0.35, 0.1, 0.45, 0.1]$. Clearly these are recommendations based only on the datasets we have experimented with and the optimal settings will vary from dataset to dataset, most notably based on the number of clients and the number of datapoints per client.

Number of steps of FedDP-Lloyds The other important parameter to set in FedDP-KMeans is the number of steps of FedDP-Lloyds to run following the initialization obtained by FedDP-Init. As discussed already, this comes with the inherent trade-off of number of iterations vs accuracy of each iteration. For a fixed overall budget, if we run many iterations, then each iteration will have a lower privacy budget and will therefore be noisier. Not only that, but in fact the question of whether we even want to run any iterations has the same trade-off. If we run no iterations of FedDP-Lloyds, then

Dataset	$\epsilon_1/\epsilon_{\text{init}}$	$\epsilon_2/\epsilon_{\text{init}}$	$\epsilon_{3G}/\epsilon_{\text{init}}$	$\epsilon_{3L}/\epsilon_{\text{init}}$
Gaussian Mixture (100 clients)	0.18	0.23	0.43	0.17
US Census (Not-for-profit Employees)	0.24	0.17	0.41	0.18
US Census (Federal Employees)	0.15	0.16	0.52	0.17
US Census (Self-Employed)	0.20	0.23	0.47	0.10

Table 2: Amount of privacy budget, as a fraction of $\varepsilon_{\text{init}}$, that is assigned to each step of FedDP-Init. Results shown are the mean of the Pareto optimal results plotted for each of the data-point-level experiments in Figures 1, 3 and 4.

Dataset	$\varepsilon_1/\varepsilon_{\text{init}}$	$\varepsilon_2/\varepsilon_{\text{init}}$	$\varepsilon_{3G}/\varepsilon_{\text{init}}$	$\varepsilon_{3L}/\varepsilon_{\text{init}}$
Gaussian Mixture (1000 clients)	0.38	0.09	0.42	0.10
Gaussian Mixture (2000 clients)	0.43	0.10	0.36	0.11
Gaussian Mixture (5000 clients)	0.43	0.09	0.37	0.11
Stack Overflow (facebook, hibernate)	0.29	0.15	0.42	0.15
Stack Overflow (github, pdf)	0.37	0.12	0.40	0.10
Stack Overflow (machine-learning, math)	0.29	0.14	0.45	0.13
Stack Overflow (plotting, cookies)	0.33	0.11	0.47	0.09

Table 3: Amount of privacy budget, as a fraction of $\varepsilon_{\text{init}}$, that is assigned to each step of FedDP-Init. Results shown are the mean of the Pareto optimal results plotted for each of the client-level experiments in Figures 2, 5, 6, 7 and 8.

we use none of our privacy budget here, and we have more available for FedDP-Init. To investigate this we do the following: for each dataset we compute, for each number of steps T of FedDP-Lloyds, the fraction of the Pareto optimal runs that used T steps.

Dataset	0 steps	1 step	2 steps
Gaussian Mixture (100 clients)	0.61	0.39	0
US Census (Not-for-profit Employees)	0.8	0.1	0.1
US Census (Federal Employees)	0.91	0.09	0
US Census (Self-Employed)	0.92	0.08	0

Table 4: Fraction of the Pareto optimal results that used a given number of steps of FedDP-Lloyds for the data-point-level experiments.

Dataset	0 steps	1 step	2 steps
Gaussian Mixture (1000 clients)	0.86	0.11	0.04
Gaussian Mixture (2000 clients)	0.8	0.17	0.03
Gaussian Mixture (5000 clients)	0.81	0.1	0.1
Stack Overflow (facebook, hibernate)	1.0	0	0
Stack Overflow (github, pdf)	1.0	0	0
Stack Overflow (machine-learning, math)	0.94	0.06	0
Stack Overflow (plotting, cookies)	0.96	0.04	0

Table 5: Fraction of the Pareto optimal results that used a given number of steps of FedDP-Lloyds for the client-level experiments.

The results, shown in Tables 4 and 5, are interesting. In all but one dataset more than 80% of the optimal runs used no steps of FedDP-Lloyds, with many of the datasets being over 90%. The preference was to instead use all the budget for the initialization. The reason for this is again the inherent trade-off between number of steps and accuracy of each step, with it clearly here being the case that fewer more accurate steps were better. One point to note here is that FedDP-Init essentially already has a step of Lloyds built into it, Step 3 is nearly identical to a Lloyds step but with points assigned by distance in the projected space. Running this step once and to a higher

degree of accuracy tended to outperform using more steps. This in fact highlights the point made in our motivation, about the importance of finding an initialization that is already very good, and does not require many follow up steps of Lloyds.

G.5 Adapting FedDP-KMeans to client-level privacy

As discussed in Section 5.2, moving to client-level DP changes the sensitivities of the algorithm steps that use client data. To calibrate the noise correctly we enforce the sensitivity of each step by clipping the quantities sent by each client to the server, prior to them being aggregated.

Concretely, suppose v_j is a vector quantity owned by client j, and the server wishes to compute the aggregate $v = \sum_j v_j$. Then prior to aggregation the client vector is clipped to have maximum norm B so that

$$\hat{v}_j = \begin{cases} \frac{B}{\|v_j\|} v_j, & \text{if } \|v_j\| > B \\ v_j, & \text{otherwise.} \end{cases}$$

The aggregate is then computed as $\hat{v} = \sum_j \hat{v}_j$. This query now has sensitivity B, and noise can be added accordingly. Each step of our algorithms can be expressed as such an aggregation over client statistics, the value of B for each step becomes a hyperparameter of the algorithm.

We make one additional modification to Step 3 of FedDP-Init to make it better suited to the client-level DP setting. In Algorithm 1 during Step 3 the clients compute the sum m_r^j and count n_r^j of the vectors in each cluster S_r^j. Rather than send these to the server to be aggregated the client instead computes their cluster means locally as

$$u_r^j = \begin{cases} \frac{m_r^j}{n_r^j}, & \text{if } n_r^j > 0 \\ 0, & \text{otherwise,} \end{cases}$$

as well as a histogram counting how many non-empty clusters the client has:

$$c_r^j = \begin{cases} 1, & \text{if } n_r^j > 0 \\ 0, & \text{otherwise.} \end{cases}$$

The server then receives the noised aggregates $\widehat{u_r}$ and $\widehat{c_r}$ and computes the initial cluster centers as $\nu_r = \widehat{u_r}/\widehat{c_r}$. In other words we use a mean of the means estimate of the true cluster mean.

H Additional Figures

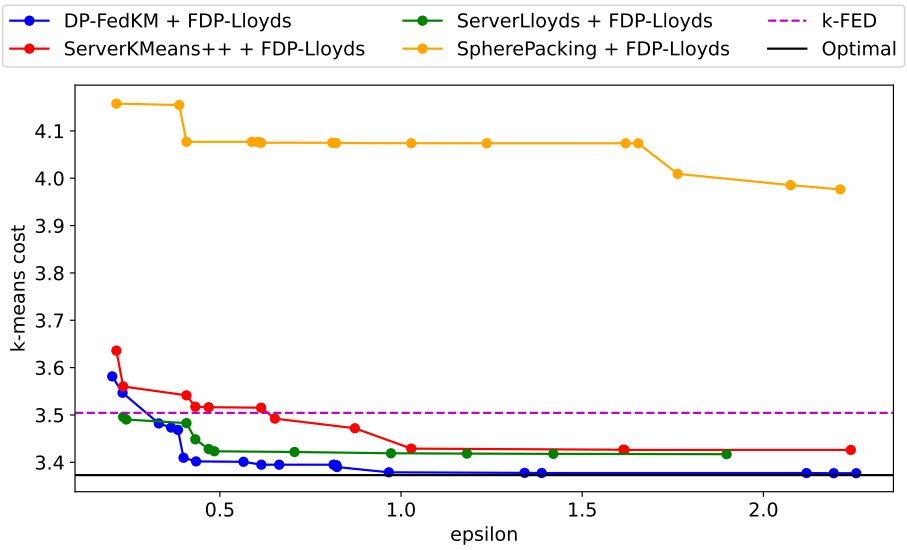

Figure 3: Results with data-point-level privacy on US census data. The 51 clients are US states, each client has the data of individuals with employment type "Employee of a private not-for-profit, tax-exempt, or charitable organization".

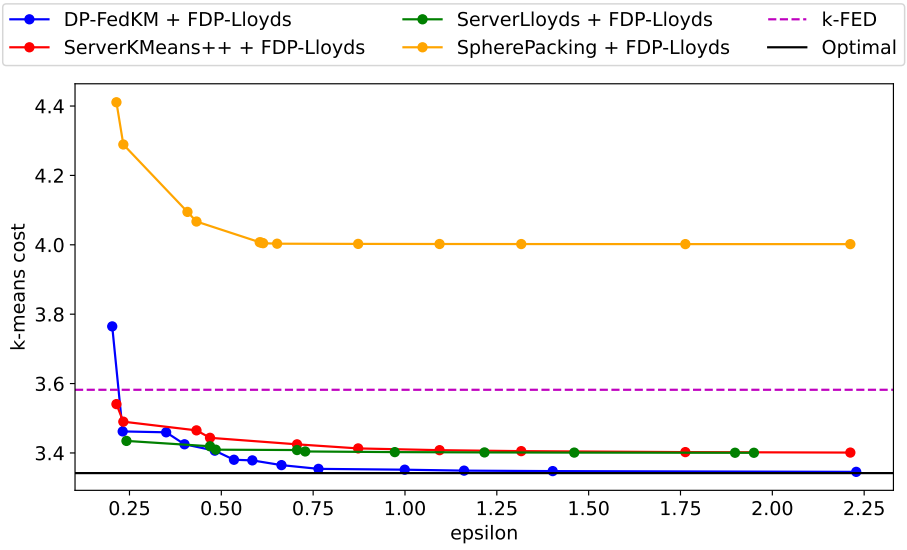

Figure 4: Results with data-point-level privacy on US census data. The 51 clients are US states, each client has the data of individuals with employment type "Self-employed in own not incorporated business, professional practice, or farm".

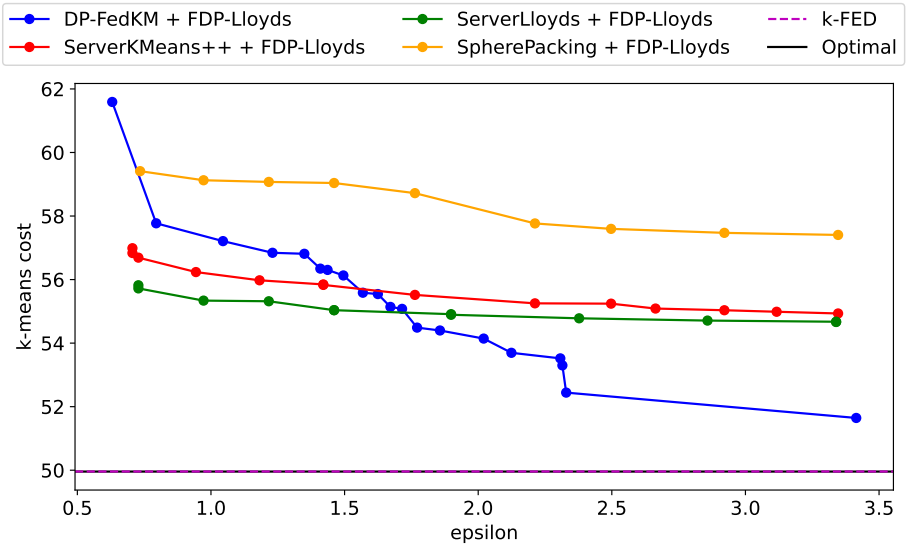

Figure 5: Results with client-level privacy on Synthetic mixture of Gaussians data with 1000 clients in total.

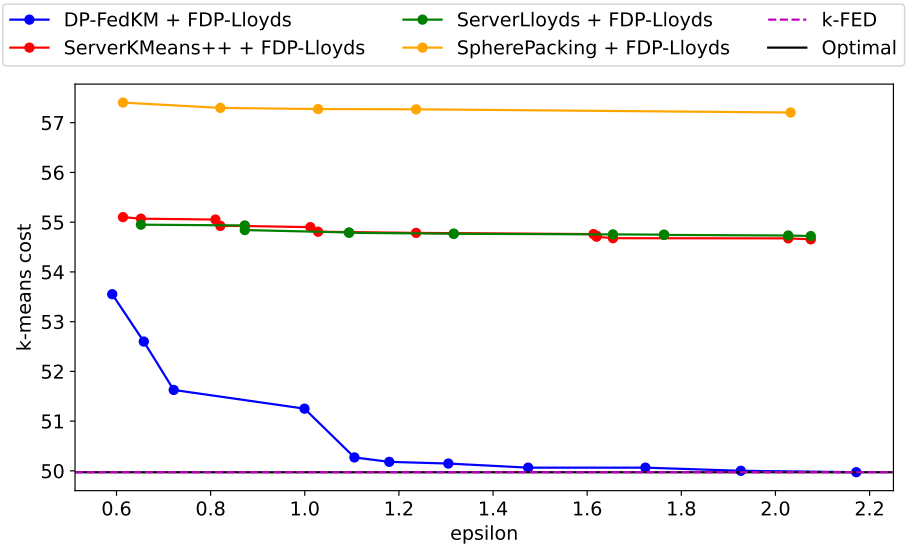

Figure 6: Results with client-level privacy on Synthetic mixture of Gaussians data with 5000 clients in total.

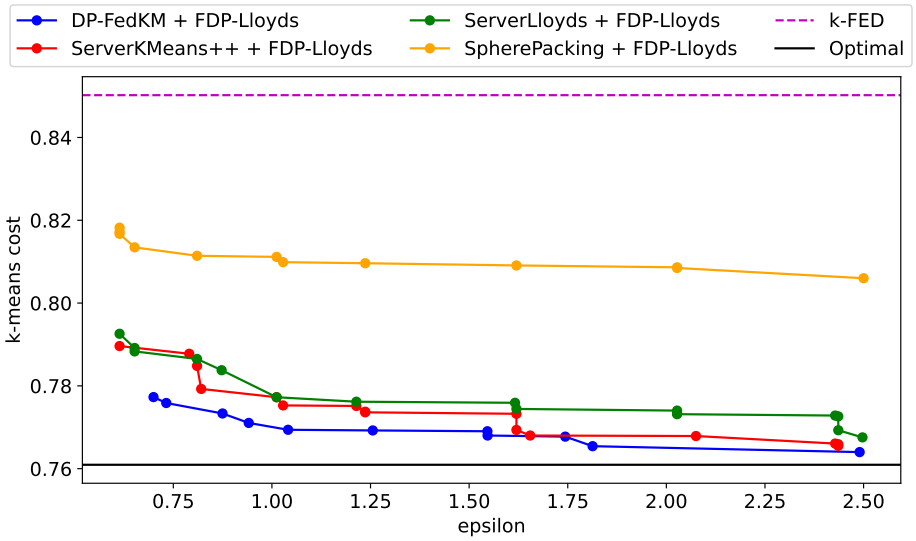

Figure 7: Results with client-level privacy on the stackoverflow dataset with 23266 clients with topic tags facebook and hibernate.

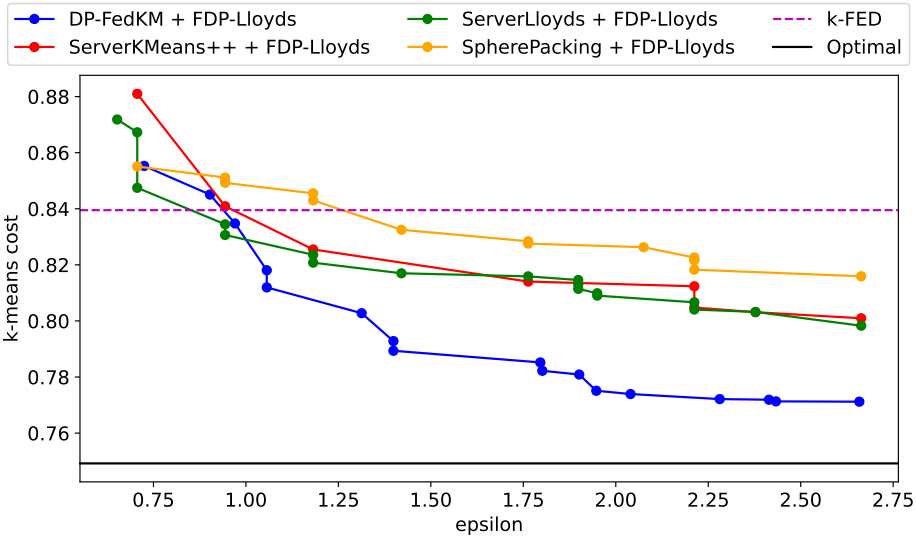

Figure 8: Results with client-level privacy on the stackoverflow dataset with 2720 clients with topic tags plotting and cookies.

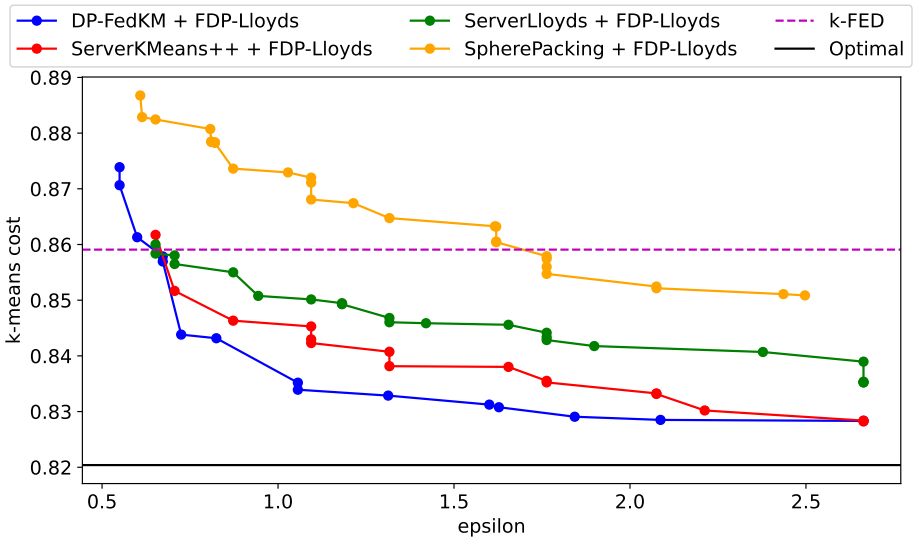

Figure 9: Results with client-level privacy on the stackoverflow dataset with 10394 clients with topic tags machine-learning and math.