Differentially-Private Clustering of Easy Instances

Edith Cohen [1 2] **Haim Kaplan** [1 2] **Yishay Mansour** [1 2] **Uri Stemmer** [1 3] **Eliad Tsfadia** [1 2]

Abstract

Clustering is a fundamental problem in data analysis. In differentially private clustering, the goal is to identify k cluster centers without disclosing information on individual data points. Despite significant research progress, the problem had so far resisted practical solutions. In this work we aim at providing simple implementable differentially private clustering algorithms that provide utility when the data is "easy," e.g., when there exists a significant separation between the clusters.

We propose a framework that allows us to apply non-private clustering algorithms to the easy instances and privately combine the results. We are able to get improved sample complexity bounds in some cases of Gaussian mixtures and k-means. We complement our theoretical analysis with an empirical evaluation on synthetic data.

1. Introduction

Differential privacy (Dwork et al., 2006b) is a mathematical definition of privacy, that aims to enable statistical analyses of databases while providing strong guarantees that individual-level information does not leak. Privacy is achieved in differentially private algorithms through randomization and the introduction of "noise" to obscure the effect of each individual, and thus differentially private algorithms can be less accurate than their non-private analogues. In most cases, this loss in accuracy is studied theoretically, using asymptotic tools. As a result, there is currently a significant gap between what is known to be possible *theoretically* and what can be done *in practice* with differential privacy. In this work we take an important step towards bridging this gap in the context of *clustering related tasks*.

The construction of differentially private clustering algorithms has attracted a lot of attention over the last decade, and many different algorithms have been suggested.[1] However, to the best of our knowledge, none of these algorithms have been implemented: They are not particularly simple and suffer from large hidden constants that translate to a significant loss in utility, compared to non-private implementations.

Question 1.1. *How hard is it to cluster privately with a practical implementation?*

We take an important step in this direction using the following approach. Instead of directly tackling "standard" clustering tasks, such as k-means clustering, we begin by identifying a very simple clustering problem that still seems to capture many of the challenges of practical implementations (we remark that this problem is completely trivial without privacy requirements). We then design effective (private) algorithms for this simple problem. Finally, we reduce "standard" clustering tasks to this simple problem, thereby obtaining private algorithms for other tasks.

In more detail, we introduce the following problem, called the *k-tuple clustering* problem.

Definition 1.2 (informal, revised in Definition 3.6)**.** *An instance of the* k-tuple clustering *problem is a collection of k-tuples. Assuming that the input tuples can be partitioned into k "obvious clusters", each consisting of one point of each tuple, then the goal is to report k "cluster-centers" that correctly partition the input tuples into clusters. If this assumption on the input structure does not hold, then the outcome is not restricted.*

Remark 1.3.

1. *By "obvious clusters" we mean clusters which are far away from each other.*
2. *The input tuples are* unordered*. This means, e.g., that the "correct" clustering might place the first point of one tuple with the fifth point of another tuple.*
3. *Of course, we want to solve this problem while guaranteeing differential privacy. Intuitively, this means*

[*]Equal contribution [1]Google Research [2]Blavatnik School of Computer Science, Tel Aviv University [3]Ben-Gurion University. Correspondence to: Eliad Tsfadia <eliadtsfadia@gmail.com>.

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

[1](Blum et al., 2005; Nissim et al., 2007; Feldman et al., 2009; McSherry, 2009; Gupta et al., 2010; Mohan et al., 2012; Wang et al., 2015; Nock et al., 2016; Su et al., 2016; Nissim et al., 2016; Feldman et al., 2017; Balcan et al., 2017; Nissim & Stemmer, 2018; Huang & Liu, 2018; Kaplan & Stemmer, 2018; Stemmer, 2020; Shechner et al., 2020; Ghazi et al., 2020; Nguyen, 2020)

that the outcome of our algorithm should not be significantly effected when arbitrarily modifying one of the input tuples.

Observe that without the privacy requirement this task is trivial: We can just take one arbitrary input tuple $(x_1, ..., x_k)$ and report it. With the privacy requirement, this task turns out to be non-trivial. It's not that this problem cannot be solved with differential privacy. It can. It's not even that the problem requires large amounts of data asymptotically. It does not. However, it turns out that designing an implementation with a practical privacy-utility tradeoff, that is effective on finite datasets (of reasonable size), is quite challenging.

1.1. Our algorithms for the k-tuple problem

We present two (differentially private) algorithms for the k-tuple clustering problem, which we call $\mathrm{PrivatekAverages}$ and $\mathrm{PrivatekNoisyCenters}$. Both algorithms first privately test if indeed the input is partitioned into k obvious clusters and quit otherwise. They differ by the way they compute the centers in case this test passes. Algorithm $\mathrm{PrivatekAverages}$ privately averages each identified cluster. Algorithm $\mathrm{PrivatekNoisyCenters}$, on the other hand, does not operate by averaging clusters. Instead, it selects one of the input k-tuples, and then adds a (relatively small) Gaussian noise to every point in this tuple. We prove that this is private if indeed there are k obvious clusters in the input. We evaluate these two algorithms empirically, and show that, while algorithm $\mathrm{PrivatekAverages}$ is "better in theory", algorithm $\mathrm{PrivatekNoisyCenters}$ is much more practical for some interesting regimes of parameters.

We now give a simplified overview of the ideas behind our algorithms. For concreteness, we focus here on $\mathrm{PrivatekAverages}$. Recall that in the k-tuple clustering problem, we are only required to produce a good output assuming the data is "nice" in the sense that the input tuples can be clustered into k "far clusters" such that every cluster contains exactly one point from every tuple. However, with differential privacy we are "forced" to produce good outputs even when this niceness assumption does not hold. This happens because if the input data is "almost nice" (in the sense that modifying a small number of tuples makes it nice) then differential privacy states that the outcome of the computation should be close to what it is when the input data is nice.

So, the definition of differential privacy forces us to cope with "almost nice" datasets. Therefore, the niceness test that we start with has to be a bit clever and "soft" and succeed with some probability also for data which is "almost nice". Then, in order to achieve good performances, we have to utilize the assumption that the data is "almost nice" when we compute the private centers. To compute these centers, Algorithm $\mathrm{PrivatekAverages}$ determines (*non-privately*) a clustering of the input tuples, and then averages (with noise) each of the clusters. The conceptual challenge here is to show that even though the clustering of the data is done non-privately, it is stable enough such that the outcome of this algorithm still preserves privacy.

1.2. Applications

The significance of algorithms $\mathrm{PrivatekAverages}$ and $\mathrm{PrivatekNoisyCenters}$ is that many clustering related tasks can be privately solved by a reduction to the k-tuple clustering problem. In this work we explore two important use-cases: (1) Privately approximating the k-means under stability assumption, and (2) Privately learning the parameters of a mixture of well-separated Gaussians.

k-Means Clustering

In k-means clustering, we are given a database $\mathcal{P}$ of n input points in $\mathbb{R}^d$, and the goal is to identify a set C of k *centers* in $\mathbb{R}^d$ that minimizes the sum of squared distances from each input point to its nearest center. This problem is NP-hard to solve exactly, and even NP-hard to approximate to within a multiplicative factor smaller than 1.0013 (Lee et al., 2017). The current (non-private) state-of-the-art algorithm achieves a multiplicative error of 6.357 (Ahmadian et al., 2019).

One avenue that has been very fruitful in obtaining more accurate algorithms (non-privately) is to look beyond worst-case analysis (Ostrovsky et al., 2012; Awasthi et al., 2010; 2012; Balcan et al., 2009; Bilu & Linial, 2012; Kumar & Kannan, 2010). In more details, instead of constructing algorithms which are guaranteed to produce an approximate clustering for any instance, works in this vain give stronger accuracy guarantees by focusing only on instances that adhere to certain "nice" properties (sometimes called stability assumptions or separation conditions). The above mentioned works showed that such "nice" inputs can be clustered much better than what is possible in the worst-case (i.e., without assumptions on the data).

Given the success of non-private stability-based clustering, it is not surprising that such stability assumptions were also utilized in the privacy literature, specifically by Nissim et al. (2007); Wang et al. (2015); Huang & Liu (2018); Shechner et al. (2020). While several interesting concepts arise from these four works, none of their algorithms have been implemented, their algorithms are relatively complex, and their practicability on finite datasets is not clear.

We show that the problem of stability-based clustering (with privacy) can be reduced to the k-tuple clustering problem. Instantiating this reduction with our algorithms for the k-tuple clustering problem, we obtain a simple and practical

algorithm for clustering "nice" k-means instances privately.

Learning Mixtures of Gaussians. Consider the task of *privately* learning the parameters of an unknown mixtures of Gaussians given i.i.d. samples from it. By now, there are various private algorithms that learn the parameters of a *single* Gaussian (Karwa & Vadhan, 2018; Kamath et al., 2019a; Cai et al., 2019; Bun & Steinke, 2019; Kamath et al., 2020; Biswas et al., 2020). Recently, (Kamath et al., 2019b) presented a private algorithm for learning mixtures of well-separated (and bounded) Gaussians. We remark, however, that besides the result of (Biswas et al., 2020), which is a practical algorithm for learning a single Gaussian, all the other results are primarily theoretical.

By a reduction to the k-tuples clustering problem, we present a simple algorithm that privately learns the parameters of a separated (and bounded) *mixture* of k Gaussians. From a practical perspective, compared with the construction of (Kamath et al., 2019b), our algorithm is simple and implementable. From a theoretical perspective, our algorithm offers reduced sample complexity, weaker separation assumption, and modularity.

Our results for stability-based clustering and for learning mixtures of Gaussians, as well as an extended discussion on related works, are given in the supplementary material.

2. Preliminaries

2.1. Notation

In this work, a k-tuple $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$ is an *unordered* set of k vectors $\mathbf{x}_i \in \mathbb{R}^d$. For $\mathbf{x} \in \mathbb{R}^d$, we denote by $\|\mathbf{x}\|$ the ℓ_2 norm of $\mathbf{x}$. For $\mathbf{c} \in \mathbb{R}^d$ and $r > 0$, we denote $B(\mathbf{c}, r) := \{\mathbf{x} \in \mathbb{R}^d \colon \|\mathbf{x} - \mathbf{c}\| \leq r\}$. For a multiset $\mathcal{P} \in (\mathbb{R}^d)^*$ we denote by $\mathrm{Avg}(\mathcal{P}) := \frac{1}{|\mathcal{P}|} \cdot \sum_{\mathbf{x} \in \mathcal{P}} \mathbf{x}$ the average of all points in $\mathcal{P}$. Throughout this work, a database $\mathcal{D}$ is a multiset. For two multisets $\mathcal{D} = \{x_1, \ldots, x_n\}$ and $\mathcal{D}' = \{x'_1, \ldots, x'_m\}$, we let $\mathcal{D} \cup \mathcal{D}'$ be the multiset $\{x_1, \ldots, x_n, x'_1, \ldots, x'_m\}$. For a multiset $\mathcal{D} = \{x_1, \ldots, x_n\}$ and a set S, we let $\mathcal{M} \cap S$ be the multiset $\{x_i\}_{i \in \mathcal{I}}$ where $\mathcal{I} = \{i \in [n] \colon x_i \in S\}$. All logarithms considered here are natural logarithms (i.e., in base e).

2.2. Indistinguishability and Differential Privacy

Definition 2.1 (Neighboring databases)**.** *Let $\mathcal{D} = \{x_1, \ldots, x_n\}$ and $\mathcal{D}' = \{x'_1, \ldots, x'_n\}$ be two databases over a domain $\mathcal{X}$. We say that $\mathcal{D}$ and $\mathcal{D}'$ are **neighboring** if there is exactly one index $i \in [n]$ with $x_i \neq x'_i$.*

Definition 2.2 ((ε, δ)-indistinguishable)**.** *Two random variable X, X' over a domain $\mathcal{X}$ are called (ε, δ)-indistinguishable, iff for any event $T \subseteq \mathcal{X}$, it holds that $\Pr[X \in T] \leq e^{\varepsilon} \cdot \Pr[X' \in T] + \delta$. If $\delta = 0$, we say that X and X' are ε-indistinguishable.*

Definition 2.3 ((ε, δ)-differential privacy (Dwork et al., 2006b))**.** *An algorithm $\mathcal{A}$ is called (ε, δ)-differentially private, if for any two neighboring databases $\mathcal{D}, \mathcal{D}'$ it holds that $\mathcal{A}(\mathcal{D})$ and $\mathcal{A}(\mathcal{D}')$ are (ε, δ)-indistinguishable. If $\delta = 0$ (i.e., pure privacy), we say that $\mathcal{A}$ is ε-differentially private.*

Lemma 2.4 ((Bun & Steinke, 2016))**.** *Two random variable X, X' over a domain $\mathcal{X}$ are (ε, δ)-indistinguishable, iff there exist events $E, E' \subseteq \mathcal{X}$ with $\Pr[X \in E], \Pr[X' \in E'] \geq 1 - \delta$ such that $X|_E$ and $X'|_{E'}$ are ε-indistinguishable.*

2.2.1. The Laplace Mechanism

Definition 2.5 (Sensitivity)**.** *We say that a function $f \colon \mathcal{U}^n \to \mathbb{R}$ has sensitivity λ, if for all neighboring databases $\mathcal{S}, \mathcal{S}'$ it holds that $|f(\mathcal{S}) - f(\mathcal{S}')| \leq \lambda$.*

Theorem 2.6 (The Laplace Mechanism (Dwork et al., 2006b))**.** *Let $\varepsilon > 0$, and assume $f \colon \mathcal{U}^n \to \mathbb{R}$ has sensitivity λ. Then the mechanism that on input $\mathcal{S} \in \mathcal{U}^n$ outputs $f(\mathcal{S}) + \mathrm{Lap}(\lambda/\varepsilon)$ is ε-differentially private.*

2.2.2. The Gaussian Mechanism

Definition 2.7 (ℓ_2-sensitivity)**.** *We say that a function $f \colon \mathcal{U}^n \to \mathbb{R}^d$ has ℓ_2-sensitivity λ if for all neigboring databases $\mathcal{S}, \mathcal{S}'$ it holds that $\|f(\mathcal{S}) - f(\mathcal{S}')\| \leq \lambda$.*

Theorem 2.8 (The Gaussian Mechanism (Dwork et al., 2006a))**.** *Let $\varepsilon, \delta \in (0, 1)$, and assume $f \colon \mathcal{U}^n \to \mathbb{R}^d$ has ℓ_2-sensitivity λ. Let $\sigma \geq \frac{\lambda}{\varepsilon}\sqrt{2\log(1.25/\delta)}$. Then the mechanism that on input $\mathcal{S} \in \mathcal{U}^n$ outputs $f(\mathcal{S}) + \left(\mathcal{N}(0, \sigma^2)\right)^d$ is (ε, δ)-differentially private.*

2.2.3. Estimating the Average of Points

The Gaussian mechanism (Theorem 2.8) allows for privately estimating the average of points in $B(\mathbf{0}, \Lambda) \subseteq \mathbb{R}^d$ within ℓ_2 error of $\approx \frac{\Lambda\sqrt{d}}{\varepsilon n}$. In some cases, we could relax the dependency on Λ. For example, using the following proposition.

Proposition 2.9 (Estimating the Average of Bounded Points in $\mathbb{R}^d$)**.** *Let $\varepsilon \in (0, 1)$, $d, \Lambda > 0$ and let $r_{\min} \in [0, \Lambda]$. There exists an efficient (ε, δ)-differentially private algorithm that takes an n-size database $\mathcal{S}$ of points inside the ball $B(\mathbf{0}, \Lambda)$ in $\mathbb{R}^d$ and satisfy the following utility guarantee: Let $r > 0$ be the minimal radius of a d-dimensional ball that contains all points in $\mathcal{S}$. Then with probability $1 - \beta$, the algorithm outputs $\hat{\mathbf{a}} \in \mathbb{R}^d$ such that*

$$\|\hat{\mathbf{a}} - \mathrm{Avg}(\mathcal{S})\| \leq \tilde{O}\left(\max\{r, r_{\min}\} \frac{d \log\left(\frac{d}{\delta}\right) \log\left(\frac{\Lambda}{r_{\min}\beta}\right)}{\varepsilon n}\right),$$

where the $\tilde{O}$ hides a $\sqrt{\log(d/\beta)}$ factor. The algorithm runs in time $\tilde{O}(dn)$ (ignoring logarithmic factors).

Proposition 2.9 can be seen as a simplified variant of (Nissim

et al., 2016)'s private average algorithm. The main difference is that (Nissim et al., 2016) first uses the Johnson Lindenstrauss (JL) transform (Johnson & Lindenstrauss, 1984) to randomly embed the input points in $\mathbb{R}^{d'}$ for $d' \approx \log n$, and then estimates the average of the points in each axis of $\mathbb{R}^{d'}$. As a result, they manage to save a factor of $\sqrt{d}$ upon Proposition 2.9 (at the cost of paying a factor of $\log(n)$ instead). However, for simplifying the construction and the implementation, we chose to omit the JL transform step, and we directly estimate the average along each axis of $\mathbb{R}^d$. For completeness, we present the full details of Proposition 2.9 in the supplementary material.

2.2.4. Sub-Sampling

Lemma 2.10 ((Beimel et al., 2010; Kasiviswanathan et al., 2011)). *Let $\mathcal{A}$ be an $(\varepsilon^*, \delta^*)$-differentially private algorithm operating on databases of size m. Fix $\varepsilon \leq 1$, and denote $n = \frac{m}{\varepsilon}(3 + \exp(\varepsilon^*))$. Construct an algorithm $\mathcal{B}$ that on an input database $\mathcal{D} = (z_i)_{i=1}^n$, uniformly at random selects a subset $\mathcal{I} \subseteq [n]$ of size m, and executes $\mathcal{A}$ on the multiset $\mathcal{D}_{\mathcal{I}} = (z_i)_{i \in \mathcal{I}}$. Then $\mathcal{B}$ is (ε, δ)-differentially private, where $\delta = \frac{n}{4m} \cdot \delta^*$.*

3. k-Tuples Clustering

We first introduce a new property of a collection of (unordered) k-tuples $\{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \in (\mathbb{R}^d)^k$, which we call *partitioned by Δ-far balls*.

Definition 3.1 (Δ-far balls). *A set of k balls $\mathcal{B} = \{B_i = B(\mathbf{c}_i, r_i)\}_{i=1}^k$ over $\mathbb{R}^d$ is called **Δ-far balls**, if for every $i \in [k]$ it holds that $\|\mathbf{c}_i - \mathbf{c}_j\| \geq \Delta \cdot \max\{r_i, r_j\}$ (i.e., the balls are relatively far from each other).*

Definition 3.2 (partitioned by Δ-far balls). *A tuple $X \in (\mathbb{R}^d)^k$ is partitioned by a given set of k Δ-far balls $\mathcal{B} = \{B_1, \ldots, B_k\}$, if for every $i \in [k]$ it holds that $|X \cap B_i| = 1$. A multiset of k-tuples $\mathcal{T} \in ((\mathbb{R}^d)^k)^*$ is **partitioned by $\mathcal{B}$**, if all $X \in \mathcal{T}$ are partitioned by $\mathcal{B}$. We say that $\mathcal{T}$ is **partitioned by Δ-far balls** if such a set $\mathcal{B}$ of k Δ-far balls exists.*

In some cases we want to use a notion of *almost* partitioned property of a database of k-tuples $\mathcal{T}$. This is defined below using the additional parameter ℓ.

Definition 3.3 (ℓ-nearly partitioned by Δ-far balls). *A multiset $\mathcal{T} \in ((\mathbb{R}^d)^k)^*$ is **ℓ-nearly partitioned by** a given set of Δ-far balls $\mathcal{B} = \{B_1, \ldots, B_k\}$, if there are at most ℓ tuples in $\mathcal{T}$ that are not partitioned by $\mathcal{B}$. We say that $\mathcal{T}$ is **ℓ-nearly partitioned by Δ-far balls** if such a set of Δ-far balls $\mathcal{B} = \{B_1, \ldots, B_k\}$ exists.*

For a database of k-tuples $\mathcal{T} \in ((\mathbb{R}^d)^k)^n$, we let $\mathrm{Points}(\mathcal{T})$ be the collection of all the points in all the k-tuples in $\mathcal{T}$.

Definition 3.4 (The points in a collection of k-tuples). *For $\mathcal{T} = \{\{\mathbf{x}_{1,j}\}_{j=1}^k, \ldots, \{\mathbf{x}_{n,j}\}_{j=1}^k\} \in ((\mathbb{R}^d)^k)^n$, we define $\mathrm{Points}(\mathcal{T}) = \{\mathbf{x}_{i,j}\}_{i \in [n], j \in [k]} \in (\mathbb{R}^d)^{kn}$.*

We next define $\mathrm{Partition}(\mathcal{T})$ of a database $\mathcal{T} \in ((\mathbb{R}^d)^k)^*$ which is partitioned by Δ-far balls for $\Delta > 2$.

Definition 3.5 ($\mathrm{Partition}(\mathcal{T})$). *Given a multiset $\mathcal{T} \in ((\mathbb{R}^d)^k)^*$ which is partitioned by Δ-far balls for $\Delta > 2$, we define the partition of $\mathcal{T}$, which we denote by $\mathrm{Partition}(\mathcal{T}) = \{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$, by fixing an (arbitrary) k-tuple $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \in \mathcal{T}$ and setting $\mathcal{P}_i = \{\mathbf{x} \in \mathrm{Points}(\mathcal{T}) \colon i = \mathrm{argmin}_{j \in [k]} \|\mathbf{x} - \mathbf{x}_j\|\}$.*

In the supplementary material we prove that $\mathrm{Partition}(\mathcal{T})$ is uniquely defined.

We now define the k-tuple clustering problem.

Definition 3.6 (k-tuple clustering). *The input to the problem is a database $\mathcal{T} \in ((\mathbb{R}^d)^k)^n$ and a parameter $\Delta > 2$. The goal is to output a k-tuple $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_k\} \in (\mathbb{R}^d)^k$ such that the following holds: If $\mathcal{T}$ is partitioned by Δ-far balls, then for every $i \in [k]$, there exists a cluster in $\mathrm{Partition}(\mathcal{T})$ (call it $\mathcal{P}_i$) such that $\mathcal{P}_i = \{\mathbf{x} \in \mathrm{Points}(\mathcal{T}) \colon i = \mathrm{argmin}_{j \in [k]} \|\mathbf{x} - \mathbf{y}_j\|\}$.*

Namely, in the k-tuple clustering problem, the goal is to output a k-tuple Y that partitions $\mathcal{T}$ correctly. We remark that for applications, we are also interested in the quality of the solution. Namely, how small is the distance between $\mathbf{y}_i$ and $\mathcal{P}_i$, compared to the other clusters in $\mathrm{Partition}(\mathcal{T})$. We also remark that without privacy, the problem is trivial, since any k-tuple $X \in \mathcal{T}$ is a good solution by definition.

4. Our Algorithms

In this section we present two (ε, δ)-differentially private algorithms for the k-tuple clustering problem: PrivatekAverages and PrivatekNoisyCenters. Algorithm PrivatekAverages attempts to solve the problem by determining the clusters in $\mathrm{Partition}(\mathcal{T})$ and then privately estimating the average of each cluster using the algorithm from Proposition 2.9. Algorithm PrivatekNoisyCenters, on the other hand, does not operate by averaging clusters. Instead, it first selects one of the input tuples $X \in \mathcal{T}$ (in a special way), and then adds a (relatively small) Gaussian noise to this tuple.[2]

Both algorithms share the same first step, which is to call Algorithm PrivateTestPartition (Figure 2), that privately decides whether $\mathcal{T}$ is ℓ-nearly partitioned by Δ-far balls or not (for small ℓ). If so, the algorithm determines (non-

[2] We remind that all the tuples in this work are *unordered*, and indeed the privacy analysis of our algorithms relies on it (i.e., the domain of outputs that we consider is all the unordered k-tuples, and (ε, δ)-indistinguishability holds for each subset of this domain).

privately) a set of Δ-far balls $\mathcal{B} = \{B_1, \ldots, B_k\}$ that ℓ-nearly partitions $\mathcal{T}$.

All of the missing proofs are given in the supplementary material.

4.1. Algorithm PrivateTestPartition

Algorithm PrivateTestPartition is described in Figure 2. Its main component is Algorithm PrivateTestCloseTuples (Figure 1), which inputs two multisets of k-tuples $\mathcal{T}_1$ and $\mathcal{T}_2$ and privately checks whether the tuples in $\mathcal{T}_1$ are close to the tuples in $\mathcal{T}_2$. In the supplementary material, we prove that the value of $Status$ in PrivateTestCloseTuples preserves ε_1-DP w.r.t. $\mathcal{T}_1$, and preserves ε_2-DP w.r.t. $\mathcal{T}_2$.

Algorithm PrivateTestCloseTuples

Input: Multisets $\mathcal{T}_1 \in ((\mathbb{R}^d)^k)^m$ and $\mathcal{T}_2 \in ((\mathbb{R}^d)^k)^n$, a privacy parameter $\varepsilon_1 \in (0,1]$ for $\mathcal{T}_1$, a privacy parameter $\varepsilon_2 \in (0,1]$ for $\mathcal{T}_2$, a confidence parameter $\beta \in (0,1]$, and a separation parameter $\Delta > 6$.

1. For each $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\} \in \mathcal{T}_1$:
   (a) $\mathcal{B}_X = \{B_i^X = B(\mathbf{x}_i, r_i^X)\}_{i=1}^k$, where $r_i^X = \frac{1}{\Delta} \cdot \min_{j \neq i} \|\mathbf{x}_i - \mathbf{x}_j\|$.
   (b) $\ell_X = |\{Y \in \mathcal{T}_2 \colon Y \text{ is } \textbf{not} \text{ partitioned by } \mathcal{B}_X\}|$.
   (c) $\hat{\ell}_X = \ell_X + \mathrm{Lap}\left(\frac{m}{\varepsilon_2}\right)$.
   (d) $\mathrm{pass}_X = \begin{cases} 1 & \hat{\ell}_X \leq \frac{m}{\varepsilon_2} \cdot \log\left(\frac{m}{\beta}\right) \\ 0 & \text{otherwise} \end{cases}$.
2. $s = \sum_{X \in \mathcal{T}_1} \mathrm{pass}_X$ and $\hat{s} \leftarrow s + \mathrm{Lap}\left(\frac{1}{\varepsilon_1}\right)$.
3. If $\hat{s} < m - \frac{1}{\varepsilon_1} \log\left(\frac{1}{\beta}\right)$, set $Status$ = "Failure". Otherwise, set $Status$ = "Success".
4. If $Status$ = "Success" and $\mathrm{pass}_X = 1$ for at least one $X \in \mathcal{T}_1$, let X^* be the first tuple in $\mathcal{T}_1$ with $\mathrm{pass}_{X^*} = 1$ and set $\mathcal{B} = \mathcal{B}_{X^*}$. Otherwise, set $\mathcal{B}$ to be a set of k empty balls.
5. Output $(Status, \mathcal{B})$.

Figure 1: Algorithm PrivateTestCloseTuples for privately checking if Δ-far balls around each k-tuples in $\mathcal{T}_1$ partitions the tuples in $\mathcal{T}_2$.

In the following, we specify our choices of m and ε_1 (functions of $n, \varepsilon, \delta, \beta$) that are used by PrivateTestPartition.

Definition 4.1. *Let $m = m(n, \varepsilon, \delta, \beta)$ be the smallest integer that satisfies $m > \frac{1}{\varepsilon_1} \cdot (2\log(1/\delta) + \log(1/\beta))$, where $\varepsilon_1 = \log(\frac{\varepsilon n}{2m} - 3)$.*

The dependence between m and ε_1 for Algorithm PrivateTestPartition is due to the choice of $\mathcal{T}_1$ as an m-size random sample of $\mathcal{T}$. A smaller m allows for a larger value of ε_1 for the same overall privacy, by a sub-sampling argument (e.g., Lemma 2.10). We note that for $n \gg 1/\varepsilon$ and $\beta, \delta \geq \frac{1}{\mathrm{poly}(n)}$, we have $\varepsilon_1 = \Theta(\log n)$, which yields that $m = O(1)$. For smaller values of δ, we obtain that $m = O\left(\frac{\log(1/\delta)}{\log n}\right)$.

Algorithm PrivateTestPartition

Input: A multiset $\mathcal{T} \in ((\mathbb{R}^d)^k)^n$, privacy parameters $\varepsilon, \delta \in (0,1]$, confidence parameter $\beta \in (0,1]$, and separation parameter $\Delta > 6$.

1. Let m and ε_1 be the values from Definition 4.1 w.r.t. $n, \varepsilon, \delta, \beta$, and let $\varepsilon_2 = \varepsilon/2$.
2. Let $\mathcal{T}_1$ be a uniform sample of m k-tuples from $\mathcal{T}$ (without replacement), and let $\mathcal{T}_2 = \mathcal{T}$.
3. Output $(Status, \mathcal{B}) =$ PrivateTestCloseTuples$(\mathcal{T}_1, \mathcal{T}_2, \varepsilon_1, \varepsilon_2, \beta, \Delta)$.

Figure 2: Algorithm PrivateTestPartition for privately checking if $\mathcal{T}$ is partitioned by Δ-far balls.

4.1.1. Properties of PrivateTestPartition

Claim 4.2 (Correctness)**.** *Assume that $\mathcal{T}$ is partitioned by $(2\Delta+2)$-far balls. Then with probability $1-\beta$, when executing* PrivateTestPartition *on input $\mathcal{T}, \varepsilon, \delta, \beta, \Delta$, it outputs ("Success", $\mathcal{B}$), where $\mathcal{B}$ is a set of Δ-far balls that partitions $\mathcal{T}$.*

Claim 4.3 ($Status$ is private)**.** *Let $\mathcal{T}$ and $\mathcal{T}'$ be two neighboring databases, and consider two independent executions* PrivateTestPartition$(\mathcal{T})$ *and* PrivateTestPartition$(\mathcal{T}')$ *(with the same parameters $\varepsilon, \delta, \beta, \Delta$). Let $Status$ and $Status'$ be the status outcomes of the two executions (respectively). Then $Status$ and $Status'$ are ε-indistinguishable.*

The following claim states that when the tests succeed, then w.h.p., $\mathcal{T}$ is ℓ-nearly partitioned by $\mathcal{B}$, for the value of ℓ defined below.

Definition 4.4. *Let $\ell = \ell(n, \varepsilon, \delta, \beta) = \frac{2m}{\varepsilon} \cdot \log\left(\frac{m}{\beta\delta}\right)$, where $m = m(n, \varepsilon, \delta, \beta)$ is the value from Definition 4.1.*

We note that $\ell = O\left(\frac{\log^2(1/\delta)}{\varepsilon \log n}\right)$. When $\beta, \delta \geq 1/\mathrm{poly}(n)$, we have that $\ell = O\left(\frac{1}{\varepsilon}\log n\right)$.

Claim 4.5 (On success, $\mathcal{B}$ almost partitions $\mathcal{T}$)**.** *Let $\mathcal{T} \in ((\mathbb{R}^d)^k)^n$ and $\delta > 0$. Consider a random execution of*

$\text{PrivateTestPartition}(\mathcal{T}, \varepsilon, \delta, \beta)$*, and let* $(Status, \mathcal{B})$ *be the outcome of the execution. Let* S *be the event that* $Status =$ *"Success", and let* $E \subseteq S$ *be the event that* $\mathcal{T}$ *is* ℓ*-nearly partitioned by* $\mathcal{B}$*, where* $\ell = \ell(n, \varepsilon, \delta, \beta)$ *is the value from Definition 4.4. Then the following holds: If* $\Pr[S] \geq \delta$*, then* $\Pr[E \mid S] \geq 1 - \delta$*.*

Recall that Algorithm PrivateTestPartition has two outputs: A bit $Status$ and a set of balls $\mathcal{B}$. As we stated in Claim 4.3, the bit $Status$ preserves privacy. The set of balls $\mathcal{B}$, however, does *not*. Still, in the following sections we use Algorithm PrivateTestPartition as a subroutine in our two main algorithms PrivatekAverages and PrivatekNoisyCenters. To argue about the privacy properties of these algorithms, we rely on the following key property of algorithm PrivateTestPartition.

Claim 4.6. *Let* $\mathcal{A}^*$ *be an algorithm that gets as input a multiset* $\mathcal{T} \in ((\mathbb{R}^d)^k)^n$ *and a set of balls* $\mathcal{B} = \{B_1, \ldots, B_k\}$*, and let* $\ell = \ell(n, \varepsilon/2, \delta/4, \beta/2)$ *be the value from Definition 4.4. Assume that* $\mathcal{A}^*$ *has the property that for any neighboring multisets* $\mathcal{T}, \mathcal{T}'$ *and any sets of* Δ*-far balls* $\mathcal{B}, \mathcal{B}'$ *that* ℓ*-nearly partitions* $\mathcal{T}$ *and* $\mathcal{T}'$ *(respectively), it holds that* $\mathcal{A}^*(\mathcal{T}, \mathcal{B})$ *and* $\mathcal{A}^*(\mathcal{T}', \mathcal{B}')$ *are* $(\varepsilon^*, \delta/4)$*-indistinguishable. Let* $\mathcal{A}$ *be the algorithm that on input* $\mathcal{T}$*, does the following steps: (1) Compute* $(Status, \mathcal{B}) = \text{PrivateTestPartition}(\mathcal{T}, \varepsilon/2, \delta/4, \beta/2, \Delta)$*, and (2) If* $Status =$ *"Failure", output* $\perp$ *and abort, and otherwise output* $\mathcal{A}^*(\mathcal{T}, \mathcal{B})$*. Then* $\mathcal{A}$ *is* $(\varepsilon/2 + \varepsilon^*, \delta)$*-differentially private.*

Proof sketch. Consider two executions over two neighboring databases. If the probability of success is smaller than δ in the two executions, then the outputs are $(0, \delta)$-indistinguishable by Lemma 2.4. Otherwise, Claim 4.5 tells us that when the tests succeed, then w.h.p. the sets of Δ-far balls ℓ-nearly partitions the databases in both executions. Hence, the proof holds by the assumption on $\mathcal{A}^*$. □

Remark 4.7. *Note that* PrivateTestPartition *runs in time* $O(mdk^2n) = \tilde{O}(dk^2n)$ *since for each iteration* $X \in \mathcal{T}_1$ *in* PrivateTestCloseTuples*, Step 1a takes* $O(dk^2)$ *time, and Step 1b takes* $O(dk^2n)$ *times.*

4.2. Algorithm PrivatekAverages

In this section we describe and state the properties of Algorithm PrivatekAverages (Figure 3). The properties are given in the following theorems.

Theorem 4.8 (Utility of PrivatekAverages)**.** *Let* $d, k, \Lambda > 0$*,* $r_{\min} \in [0, \Lambda]$*,* $\varepsilon, \delta, \beta \in (0, 1]$*, and let* $\mathcal{T} \in \big(B(0, \Lambda)^k\big)^n \subseteq ((\mathbb{R}^d)^k)^n$*. Assume that* $\mathcal{T}$ *is partitioned by* Δ*-far balls for* $\Delta = 16$*, let* $\{\mathcal{P}_1, \ldots, \mathcal{P}_k\} = \text{Partition}(\mathcal{T})$ *(according to Definition 3.5), let* r_i *be the radius of the ball that contains* $\mathcal{P}_i$*. Then w.p.* $1 - \beta$*, algorithm* PrivatekAverages *on inputs* $\mathcal{T}, r_{\min}, \varepsilon, \delta, k$*, outputs* k *points* $\hat{A} = \{\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_k\}$ *such that for any* $i \in [k]$*, there exists a cluster (call it* $\mathcal{P}_i$*) with*

$$\|\hat{\mathbf{a}}_i - \mathbf{a}_i\| \leq \tilde{O}\left(\max\{r_i, r_{\min}\} \cdot \frac{dk\ell \log\left(\frac{\ell}{\delta}\right) \log\left(\frac{\Lambda}{r_{\min}\beta}\right)}{\varepsilon n}\right),$$

where $\mathbf{a}_i = \text{Avg}(\mathcal{P}_i)$ *and* $\ell = \ell(n, \frac{\varepsilon}{2}, \frac{\delta}{4}, \frac{\beta}{2})$ *is the value from Definition 4.4.*

Algorithm PrivatekAverages

Input: A multiset $\mathcal{T} \in \big(B(0, \Lambda)^k\big)^n \subseteq ((\mathbb{R}^d)^k)^n$, privacy parameters $\varepsilon, \delta \in (0, 1]$, a confidence parameter $\beta \in (0, 1]$, and a lower bound on the radii $r_{\min} \in [0, \Lambda]$.

1. Compute $(Status, \mathcal{B} = \{B_1, \ldots, B_k\}) = \text{PrivateTestPartition}(\mathcal{T}, \varepsilon/2, \delta/4, \beta/2, \Delta = 7)$.
2. If $Status =$ "Failure", output $\perp$ and abort.
3. Let $\mathbf{c}_1, \ldots, \mathbf{c}_k$ be the centers of $B_1, \ldots, B_k$ (respectively), and let $\mathcal{Q}_i = \{\mathbf{x} \in \text{Points}(\mathcal{T}) \colon i = \text{argmin}_j \|\mathbf{x} - \mathbf{c}_j\|\}$.
4. Let $\ell = \ell(n, \varepsilon/2, \delta/4, \beta/2)$ be the value from Definition 4.4.
5. For $i = 1$ to k:
   - Compute a noisy average $\hat{\mathbf{a}}_i$ of $\mathcal{Q}_i$ using the algorithm from Proposition 2.9 with parameters $\Lambda, r_{\min}, \hat{\beta} = \frac{\beta}{2k}, \hat{\varepsilon} = \frac{\varepsilon}{4k(\ell+1)}, \hat{\delta} = \frac{\delta}{8k \exp(\varepsilon/2)(\ell+1)}$.
6. Output $\hat{A} = \{\hat{\mathbf{a}}_1, \ldots, \hat{\mathbf{a}}_k\}$.

Figure 3: Algorithm PrivatekAverages for privately estimating the k averages.

Theorem 4.9 (Privacy of PrivatekAverages)**.** *Let* $d, k, \Lambda > 0$*,* $r_{\min} \in [0, \Lambda]$*,* $\varepsilon, \delta, \beta \in (0, 1]$*. Then for any integer* $n \geq 2 \cdot \ell(n, \varepsilon/2, \delta/4, \beta/2) + 2$ *(where* ℓ *is the function from Definition 4.4), algorithm* $\text{PrivatekAverages}(\cdot, \varepsilon, \delta, \beta, r_{\min})$ *is* (ε, δ)*-differentially private for databases* $\mathcal{T} \in (B(\mathbf{0}, \Lambda)^k)^n \subseteq ((\mathbb{R}^d)^k)^n$*.*

Proof Sketch. Consider two independent executions over neighboring databases $\mathcal{T}$ and $\mathcal{T}'$, and let $\mathcal{B}$ and $\mathcal{B}'$ be the balls from Step 1, respectively. By Claim 4.6, if we treat Step 3 to 6 as algorithm $\mathcal{A}^*$ of the claim, then it is enough to prove indistinguishability only in the case that $\mathcal{T}$ and $\mathcal{T}'$ are ℓ-nearly partitioned by $\mathcal{B}$ and $\mathcal{B}'$, respectively. In this case, since $\mathcal{T}$ and $\mathcal{T}'$ are neighboring and $n \geq 2\ell + 2$, there exists at least one k-tuples that is partitioned by both $\mathcal{B}$ and $\mathcal{B}'$,

yielding that each ball B_i in $\mathcal{B}$ intersects a single ball in $\mathcal{B}'$ (call it B'_i w.l.o.g.). Since $\mathcal{B}$ and $\mathcal{B}'$ are sets of Δ-far balls for large enough Δ ($\Delta = 7$, as defined in the algorithm, is sufficient), then each area $\mathcal{B}_i \cup \mathcal{B}'_i$ is "far away" from each $\mathcal{B}_j \cup \mathcal{B}'_j$ for $j \neq i$. This yields that the partition into $\{\mathcal{Q}_1, \ldots, \mathcal{Q}_k\}$ in Step 3 of both executions, agree on all the points that belong to $B_i \cup B'_i$ for some i. Therefore, this partition disagree on at most $k(\ell+1)$ points, which are the points of the (at most) $\ell+1$ tuples that are not partitioned by $\mathcal{B}$ or $\mathcal{B}'$. Hence, the privacy now holds by Proposition 2.9 with group privacy of size $k(\ell+1)$. □

Remark 4.10 (Run time of PrivatekAverages)**.** *Step 1 of* PrivatekAverages *takes* $\tilde{O}(dk^2n)$ *time (see Remark 4.7). By Proposition 2.9, the* k *executions of Step 5 take time* $\sum_{i=1}^{k} \tilde{O}(|\mathcal{T}_i|) = \tilde{O}(dkn)$*. Overall, the running time of* PrivatekAverages *is* $\tilde{O}(dk^2n)$.

4.2.1. Reducing the dependency in d

Algorithm PrivatekAverages estimates the average of each cluster $\mathcal{P}_i$ with radius r_i up to an additive error of $\tilde{O}\left(\frac{d}{n} \cdot r_i\right)$ (ignoring $\mathrm{poly}(k, 1/\varepsilon)$ and $\mathrm{polylog}(n, \delta, \beta, \Lambda, 1/r_{\min})$ factors). Yet, we can easily reduce the d into $\sqrt{d}$ by replacing in Step 5 the average algorithm of Proposition 2.9 by the average algorithm of (Nissim et al., 2016) (see Section 2.2.3).

4.3. Algorithm PrivatekNoisyCenters

In this section we describe and state the properties of Algorithm PrivatekNoisyCenters (Figure 4).

The properties of PrivatekNoisyCenters are given in the following theorems.

Theorem 4.11 (Utility of PrivatekNoisyCenters)**.** *Let* $d, k > 0$*,* $\varepsilon, \beta, \delta \in (0,1]$ *with* $\delta < \beta$*, let* $\mathcal{T} \in ((\mathbb{R}^d)^k)^n$*, and assume that* $\mathcal{T}$ *is partitioned by* $(2\Delta+2)$*-far balls, for* $\Delta = \Omega\left(\frac{k\log(k/\delta)\sqrt{\log(k/\beta)}}{\varepsilon}\right)$*. Then when executing* PrivatekNoisyCenters$(\mathcal{T}, \varepsilon, \delta, \beta, \Delta)$*, with probability* $1-\beta$*, the output* $\hat{C} = \{\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_k\}$ *satisfy for every* i *and* $j \neq i$ *that* $\|\hat{\mathbf{c}}_i - \mathbf{c}_i\| < \|\hat{\mathbf{c}}_i - \mathbf{c}_j\|$.

We remark that the k factor in the Δ in Theorem 4.11, comes from applying basic composition over the k noisy centers $\hat{C}$. This however can be reduced to $\tilde{O}(\sqrt{k})$ factor by applying advanced composition (Dwork et al., 2010).

Theorem 4.12 (Privacy of PrivatekNoisyCenters)**.** *Let* $d, k > 0$*,* $\varepsilon, \beta \in (0,1]$*,* $\delta \in (0, 1/2]$*,* $\Delta > 6$*. Then for any integer* $n \geq 2 \cdot \ell(n, \varepsilon/2, \delta/4, \beta/2) + 2$ *(where* ℓ *is the function from Definition 4.4),* PrivatekNoisyCenters$(\cdot, \varepsilon, \delta, \beta, \Delta)$ *is* $(\varepsilon + \delta/4, \delta)$*-differentially private for databases* $\mathcal{T} \in ((\mathbb{R}^d)^k)^n$.

Proof sketch. Consider two independent executions over neighboring databases $\mathcal{T}$ and $\mathcal{T}'$, and let $\mathcal{B}$ and $\mathcal{B}'$ be the balls from Step 1, respectively. Similarly to the proof sketch of Theorem 4.9 (privacy of PrivatekAverages), it is enough to prove indistinguishability only in the case that $\mathcal{T}$ and $\mathcal{T}'$ are ℓ-nearly partitioned by $\mathcal{B}$ and $\mathcal{B}'$, where in this case, each ball B_i in $\mathcal{B}$ intersects a single ball in $\mathcal{B}'$ (call it B'_i w.l.o.g.). Since $\mathcal{B}$ and $\mathcal{B}'$ are Δ-far balls, this yields that the centers of $\mathcal{B}$ and $\mathcal{B}'$ are relatively close, i.e., $\|\mathbf{c}_i - \mathbf{c}'_i\|$ is bounded by (approximately) $\frac{2}{\Delta} \cdot \min_{j\neq i}\|\mathbf{c}_i - \mathbf{c}_j\| \approx \frac{2}{\Delta} \cdot \min_{j\neq i}\|\mathbf{c}'_i - \mathbf{c}'_j\|$. Therefore, we deduce by the properties of the Gaussian mechanism that the outputs are indistinguishable. □

Algorithm PrivatekNoisyCenters

Input: A multiset $\mathcal{T} \in ((\mathbb{R}^d)^k)^n$, privacy parameters $\varepsilon \in (0,1]$, $\delta \in (0, 1/2]$, a confidence parameter $\beta \in (0,1]$, and a separation parameter $\Delta \gg 6$.

1. Compute $(Status, \mathcal{B} = \{B_1, \ldots, B_k\}) =$ PrivateTestPartition$(\mathcal{T}, \varepsilon/2, \delta/4, \beta/2, \Delta)$.
2. If $Status =$ "Failure", output $\perp$ and abort.
3. Let $\mathbf{c}_1, \ldots, \mathbf{c}_k$ be the centers of $B_1, \ldots, B_k$ (respectively).
4. For $i = 1$ to k:
   (a) $\gamma_i = \frac{4}{\Delta-2} \cdot \left(\mathrm{Lap}(4k/\varepsilon) + \frac{4k}{\varepsilon}\log(4k/\delta) + 1\right)$
   (b) $\lambda_i = \frac{2}{\Delta}(1+\gamma_i)\min_{j\neq i}\|\mathbf{c}_i - \mathbf{c}_j\|$.
   (c) $\hat{\mathbf{c}}_i = \mathbf{c}_i + (\mathcal{N}(\mathbf{0}, \sigma_i^2))^d$, where $\sigma_i = \frac{4k\lambda_i}{\varepsilon}\sqrt{2\log(10k/\delta)}$.
5. Output $\hat{C} = \{\hat{\mathbf{c}}_1, \ldots, \hat{\mathbf{c}}_k\}$.

Figure 4: Algorithm PrivatekNoisyCenters for privately finding the k centers.

Remark 4.13 (Run time of PrivatekNoisyCenters)**.** *Step 1 of* PrivatekNoisyCenters *takes* $\tilde{O}(dk^2n)$ *time (see Remark 4.7). The for-loop in Step 4 only takes* $O(dkn)$ *time. Overall, the running time of* PrivatekNoisyCenters *is* $\tilde{O}(dk^2n)$.

5. Empirical Results

We implemented in Python our two main algorithms for k-tuple clustering: PrivatekAverages and PrivatekNoisyCenters. We compared the two algorithms in terms of the sample complexity that is needed to privately separate the samples from a given mixture of Gaussians. Namely, how many k-tuples we need to sample such that, when executing

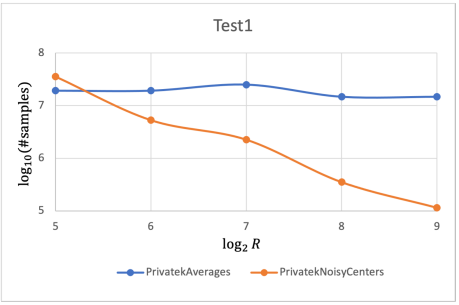

Figure 5: The case $d = 1$ and $k = 2$, for varies R.

Figure 6: The case $d = 4$ and $R = 512 \cdot k$, for varies k.

PrivatekAverages or PrivatekNoisyCenters, the resulting k-tuple $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_k\}$ satisfies the following requirement: For every $i \in [k]$, there exists a point in Y (call it $\mathbf{y}_i$), such that for every sample $\mathbf{x}$ that was drawn from the i'th Gaussian, it holds that $i = \text{argmin}_{j\in[k]}\|\mathbf{x} - \mathbf{y}_j\|$. We perform three tests, where in each test we considered a uniform mixture of k standard spherical Gaussians around the means $\{R \cdot \mathbf{e}_i, -R \cdot \mathbf{e}_i\}_{i=1}^{k/2}$, where $\mathbf{e}_i$ is the i'th standard basis vector. In all the tests, we generated each k-tuple by running algorithm k-means++ (Arthur & Vassilvitskii, 2007) over enough samples.

In Test1 (Figure 5) we examined the sample complexity in the case $d = 1$, $k = 2$, for $R \in \{2^5, 2^6, \ldots, 2^9\}$. In Test2 (Figure 6) we examined the case $d = 4$, $R = 512 \cdot k$, for $k \in \{2, 4, 6, 8\}$. In Test3 (Figure 7) we examined the case $k = 2$, $R = 256\sqrt{d}$, for $d \in \{4, 8, 12, 16\}$. In all the experiments we used privacy parameters $\varepsilon = 1$ and $\delta = e^{-28}$, and used $\beta = 0.05$. In all the tests of PrivatekNoisyCenters, we chose $\Delta = \frac{10}{\varepsilon} \cdot k \log(k/\delta)\sqrt{\log(k/\beta)}$, the number of k-tuples that we generated was exactly 3781 (the minimal value that is required for privacy), but the number of samples per k-tuple varied from test to test. In the tests of PrivatekAverages, we chose $\Lambda = 2^{10} \cdot k\sqrt{d}$ and $r_{\min} = 0.1$, we generated each k-tuple using $\approx 15 \cdot k$ samples, but the number of k-tuples varied from test to test.[3] All the experiments were tested in a MacBook Pro Laptop with 4-core Intel i7 CPU with 2.8GHz, and with 16GB RAM.

The graphs show the main bottleneck of Algorithm PrivatekAverages in practice. It requires only $O_{\varepsilon,\delta}(kd)$ tuples (or $O_{\varepsilon,\delta}(k\sqrt{d})$ for large values of d) in order to succeed, but the hidden constant is $\approx 500,000$ for our choice of ε and δ, and this does not improve even when the assumed separation R is very large. The cause of this large constant is the group privacy of size $O(k\ell)$ that we do in Step 5, where recall that $\ell = O\left(\frac{\log^2(1/\delta)}{\varepsilon \log n}\right)$ (Definition 4.4). While in theory this ℓ is relatively small, with our choice of parameters we get $\ell \approx 1000$. This means that we need to execute the private average algorithm with $\hat{\varepsilon} \approx \frac{\varepsilon}{4000k}$. Internally, this $\hat{\varepsilon}$ is shared between other private algorithms, and in particular, with an Interior Point algorithm that is one of the internal components of the average algorithm from Proposition 2.9. This algorithm is implemented using the exponential mechanism (McSherry & Talwar, 2007), which simply outputs a random noise when the number of points is too small.

We remark that prior work on differentially-private clustering, including in "easy" settings, is primarily theoretical. In particular, we are not aware of implemented methods that we could use as a baseline.[4] As a sanity check, we did consider the following naive baseline: For every sample point, add a Gaussian noise to make it private. Now, the resulting noisy samples are just samples from a new Gaussian mixture. Then, run an off-the-shelf non-private method to learn the parameters of this mixture. We tested this naive method on the simple case $d = 1$ and $k = 2$, where we generated samples from a mixture of standard Gaussians that are separated by $R = 512$. By the Gaussian mechanism, the noise magnitude that we need to add to each point for guaranteeing (ε, δ)-differential privacy, is

[3] By using $\tilde{\Omega}(kd)$ samples for creating each k-tuple, in Test3 (Figure 7) we could avoid the dependency of R in $\sqrt{d}$ (see the supplementary material for more details). However, since we only used $O(k)$ samples for each k-tuple when testing PrivatekAverages, then we could not avoid this dependency.

[4] We remark that in different settings, such as node, edge or weight-differential privacy, there exist some available implementations (e.g., (Pinot et al., 2018)).

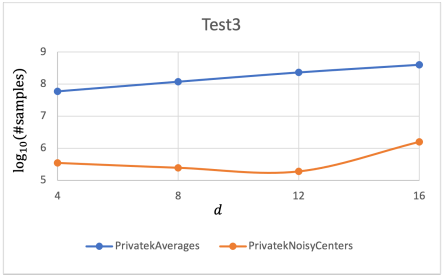

Figure 7: The case $k = 2$, $R = 256\sqrt{d}$, for varies d.

$\sigma \approx \frac{\Lambda}{\varepsilon}\sqrt{\log(1/\delta)} \gg 1$ for some $\Lambda > R$, meaning that the resulting mixture consists of very close Gaussians. We applied GaussianMixture from the package sklearn.mixture to learn this mixture, but it failed even when we used $100M$ samples, as this method is not intended for learning such close Gaussians.We remark that there are other non-private methods that are designed to learn any mixture of Gaussians (even very weakly separated ones) using enough samples (e.g., (Suresh et al., 2014)). The sample complexity and running time of these methods, however, are much worse than ours even asymptotically (e.g., the running time of (Suresh et al., 2014) is exponential in k), and moreover, we are not aware of any implementation we could use.[5]

6. Conclusion

We developed an approach to bridge the gap between the theory and practice of differentially private clustering methods. For future, we hope to further optimize the "constants" in the k-tuple clustering algorithms, making the approach practical for instances with lower separation. Tangentially, the inherent limitations of private versus non-private clustering suggest exploring different rigorous notions of privacy in the context of clustering.

Acknowledgements

Edith Cohen is supported by Israel Science Foundation grant no. 1595-19.

Haim Kaplan is supported by Israel Science Foundation grant no. 1595-19, and the Blavatnik Family Foundation.

Yishay Mansour has received funding from the European Research Council (ERC) under the European Union'sHorizon 2020 research and innovation program (grant agreement No. 882396), by the Israel Science Foundation (grant number 993/17) and the Yandex Initiative for Machine Learning at Tel Aviv University.

Uri Stemmer is partially supported by the Israel Science Foundation (grant 1871/19) and by the Cyber Security Research Center at Ben-Gurion University of the Negev.

References

Ahmadian, S., Norouzi-Fard, A., Svensson, O., and Ward, J. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. *SIAM Journal on Computing*, (0):FOCS17–97, 2019.

Arthur, D. and Vassilvitskii, S. k-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1027–1035, 2007.

Awasthi, P., Blum, A., and Sheffet, O. Stability yields a ptas for k-median and k-means clustering. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pp. 309–318. IEEE, 2010.

Awasthi, P., Blum, A., and Sheffet, O. Center-based clustering under perturbation stability. *Information Processing Letters*, 112(1-2):49–54, 2012.

Balcan, M.-F., Blum, A., and Gupta, A. Approximate clustering without the approximation. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pp. 1068–1077. SIAM, 2009.

Balcan, M.-F., Dick, T., Liang, Y., Mou, W., and Zhang, H. Differentially private clustering in high-dimensional Euclidean spaces. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 322–331, 2017.

Beimel, A., Kasiviswanathan, S. P., and Nissim, K. Bounds on the sample complexity for private learning and private data release. In *TCC*, volume 5978 of *LNCS*, pp. 437–454. Springer, 2010.

Bilu, Y. and Linial, N. Are stable instances easy? *Combinatorics, Probability and Computing*, 21(5):643–660, 2012.

Biswas, S., Dong, Y., Kamath, G., and Ullman, J. R. Coinpress: Practical private mean and covariance estimation. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS*, 2020.

Blum, A., Dwork, C., McSherry, F., and Nissim, K. Practical privacy: The SuLQ framework. In Li, C. (ed.), *PODS*, pp. 128–138. ACM, 2005.

Bun, M. and Steinke, T. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography - 14th International Conference, TCC 2016*, volume 9985, pp. 635–658, 2016.

Bun, M. and Steinke, T. Average-case averages: Private algorithms for smooth sensitivity and mean estimation. In *Advances in Neural Information Processing Systems*, pp. 181–191, 2019.

[5] Asymptotically, (Suresh et al., 2014) requires at least $\tilde{\Omega}(dk^9)$ samples, and runs in time $\tilde{\Omega}(n^2d+d^2(k^7\log d)^{k^2})$. For the setting of learning a mixture of k well-separated Gaussians, the approach of first adding noise to each point and then applying a non-private method such as (Suresh et al., 2014), results with much worse parameters than our result, which only requires $\tilde{O}(dk)$ samples and runs in time $\tilde{O}(dk^2n)$.

Cai, T. T., Wang, Y., and Zhang, L. The cost of privacy: Optimal rates of convergence for parameter estimation with differential privacy. *arXiv preprint arXiv:1902.04495*, 2019.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via distributed noise generation. In Vaudenay, S. (ed.), *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pp. 486–503. Springer, 2006a.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *TCC*, pp. 265–284. Springer, 2006b.

Dwork, C., Rothblum, G. N., and Vadhan, S. P. Boosting and differential privacy. In *FOCS*, pp. 51–60. IEEE Computer Society, 2010.

Feldman, D., Fiat, A., Kaplan, H., and Nissim, K. Private coresets. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009*, pp. 361–370, 2009.

Feldman, D., Xiang, C., Zhu, R., and Rus, D. Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks*, IPSN '17, pp. 3–15. ACM, 2017.

Ghazi, B., Kumar, R., and Manurangsi, P. Differentially private clustering: Tight approximation ratios. In *NeurIPS*, 2020.

Gupta, A., Ligett, K., McSherry, F., Roth, A., and Talwar, K. Differentially private combinatorial optimization. In *Proceedings of the Twenty-first Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '10, pp. 1106–1125, 2010.

Huang, Z. and Liu, J. Optimal differentially private algorithms for k-means clustering. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pp. 395–408, 2018.

Johnson, W. and Lindenstrauss, J. Extensions of lipschitz maps into a hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

Kamath, G., Li, J., Singhal, V., and Ullman, J. Privately learning high-dimensional distributions. In Beygelzimer, A. and Hsu, D. (eds.), *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, volume 99 of *Proceedings of Machine Learning Research*, pp. 1853–1902. PMLR, 2019a.

Kamath, G., Sheffet, O., Singhal, V., and Ullman, J. Differentially private algorithms for learning mixtures of separated gaussians. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019*, pp. 168–180, 2019b.

Kamath, G., Singhal, V., and Ullman, J. Private mean estimation of heavy-tailed distributions. *arXiv preprint arXiv:2002.09464*, 2020.

Kaplan, H. and Stemmer, U. Differentially private k-means with constant multiplicative error. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, pp. 5436–5446, 2018.

Karwa, V. and Vadhan, S. Finite sample differentially private confidence intervals. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, 2018.

Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. D. What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, 2011.

Kumar, A. and Kannan, R. Clustering with spectral norm and the k-means algorithm. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pp. 299–308. IEEE, 2010.

Lee, E., Schmidt, M., and Wright, J. Improved and simplified inapproximability for k-means. *Information Processing Letters*, 120:40–43, 2017.

McSherry, F. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009, Providence, Rhode Island, USA, June 29 - July 2, 2009*, pp. 19–30, 2009.

McSherry, F. and Talwar, K. Mechanism design via differential privacy. In *FOCS*, pp. 94–103. IEEE Computer Society, 2007.

Mohan, P., Thakurta, A., Shi, E., Song, D., and Culler, D. Gupt: Privacy preserving data analysis made easy. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, pp. 349–360, 2012.

Nguyen, H. L. A note on differentially private clustering with large additive error. *CoRR*, abs/2009.13317, 2020.

Nissim, K. and Stemmer, U. Clustering algorithms for the centralized and local models. In *Proceedings of Algorithmic Learning Theory*, volume 83 of *Proceedings of Machine Learning Research*, pp. 619–653, 2018.

Nissim, K., Raskhodnikova, S., and Smith, A. Smooth sensitivity and sampling in private data analysis. In *STOC*, pp. 75–84. ACM, 2007.

Nissim, K., Stemmer, U., and Vadhan, S. P. Locating a small cluster privately. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pp. 413–427, 2016.

Nock, R., Canyasse, R., Boreli, R., and Nielsen, F. k-variates++: more pluses in the k-means++. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016*, pp. 145–154, 2016.

Ostrovsky, R., Rabani, Y., Schulman, L. J., and Swamy, C. The effectiveness of lloyd-type methods for the k-means problem. *J. ACM*, 59(6):28:1–28:22, 2012.

Pinot, R., Morvan, A., Yger, F., Gouy-Pailler, C., and Atif, J. Graph-based clustering under differential privacy. In Globerson, A. and Silva, R. (eds.), *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018*, pp. 329–338, 2018.

Shechner, M., Sheffet, O., and Stemmer, U. Private k-means clustering with stability assumptions. In *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020*, volume 108 of *Proceedings of Machine Learning Research*, pp. 2518–2528, 2020.

Stemmer, U. Locally private k-means clustering. In *SODA*. SIAM, 2020.

Su, D., Cao, J., Li, N., Bertino, E., and Jin, H. Differentially private k-means clustering. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, CODASPY '16, pp. 26–37, 2016.

Suresh, A. T., Orlitsky, A., Acharya, J., and Jafarpour, A. Near-optimal-sample estimators for spherical gaussian mixtures. *Advances in Neural Information Processing Systems*, 27:1395–1403, 2014.

Wang, Y., Wang, Y.-X., and Singh, A. Differentially private subspace clustering. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pp. 1000–1008, 2015.