Journal of Machine Learning Research 22 (2021) 1-30 Submitted 7/20; Revised 4/21; Published 5/21

Locally Private k-Means Clustering

Uri Stemmer u@uri.co.il
Ben-Gurion University, Beer-Sheva, Israel
Google Research, Tel Aviv, Israel

Editor: Mehryar Mohri

Abstract

We design a new algorithm for the Euclidean k-means problem that operates in the local model of differential privacy. Unlike in the non-private literature, differentially private algorithms for the k-means objective incur both additive and multiplicative errors. Our algorithm significantly reduces the additive error while keeping the multiplicative error the same as in previous state-of-the-art results. Specifically, on a database of size n, our algorithm guarantees $O(1)$ multiplicative error and $\approx n^{1/2+a}$ additive error for an arbitrarily small constant $a > 0$. All previous algorithms in the local model had additive error $\approx n^{2/3+a}$. Our techniques extend to k-median clustering.

We show that the additive error we obtain is almost optimal in terms of its dependency on the database size n. Specifically, we give a simple lower bound showing that every locally-private algorithm for the k-means objective must have additive error at least $\approx \sqrt{n}$.

Keywords: Differential privacy, local model, clustering, k-means, k-median.

1. Introduction

In center-based clustering, we aim to find a "best" set of centers (w.r.t. some cost function), and then partition the data points into clusters by assigning each data point to its nearest center. With over 60 years of research, center-based clustering is an intensively-studied key-problem in unsupervised learning (see Hartigan (1975) for a textbook). One of the most well-studied problems in this context is the *Euclidean k-means problem*. In this problem we are given a set of input points $S \subseteq \mathbb{R}^d$ and our goal is to identify a set C of k *centers* in $\mathbb{R}^d$, approximately minimizing the sum of squared distances from each input point to its nearest center. This quantity is referred to as the *cost* of the centers w.r.t. the set of points, denoted as $\text{cost}_S(C) = \sum_{x \in S} \min_{c \in C} \|x - c\|^2$.

The huge applicability of k-means clustering, together with the increasing awareness and demand for user privacy, motivated a long line of research on *privacy preserving* k-means clustering. In this work we study the Euclidean k-means problem in the *local model of differential privacy (LDP)*. Differentially private algorithms work in two main modalities: *trusted-curator* and *local*. The *trusted-curator* model assumes a trusted curator that collects all the personal information and then analyzes it. The privacy guarantee in this model is that the *outcome* of the analysis "hides" the information of any single individual, but this information is not hidden from the trusted curator. In contrast, the *local* model of differential privacy, which is the model we consider in this work, does not involve a trusted curator. In this model, there are n users and an untrusted server, where each user i is holding a private input item x_i (a point in $\mathbb{R}^d$ in our case), and the server's goal is to

Reference	# Rounds	Multiplicative Error	Additive Error
Nissim and Stemmer (2018)	$O(k \log n)$	$O(k)$	$\tilde{O}\left(n^{2/3+a} \cdot k^{4/3} \cdot d^{1/3}\right)$
Kaplan and Stemmer (2018)	$O(1)$	$O(1)$	$\tilde{O}\left(n^{2/3+a} \cdot k^{2} \cdot d^{1/3}\right)$
This work	$O(1)$	$O(1)$	$\tilde{O}\left(n^{1/2+a} \cdot k \cdot \sqrt{d}\right)$

Table 1: Locally-private algorithms for k-means in the d-dimensional Euclidean space. Here n is the number of input points, k is the number of desired centers, d is the dimension, and $a > 0$ is an arbitrarily small constant (the constant hiding in the multiplicative error depends on a). We assume that input points come from the unit ball. For simplicity, we use the $\tilde{O}$ notation to hide logarithmic factors in k, n, d, as well as the dependency on the privacy parameters ε, δ and the failure probability β.

compute some function of the inputs (approximate the k-means objective in our case). However, in this model, the users do not send their data as is to the server. Instead, every user randomizes her data locally, and only sends noisy reports to the server, who aggregates all the reports. Informally, the privacy requirement is that the input of user i has almost no effect on the distribution on the messages that user i sends to the server. We refer to the collection of user inputs $S = (x_1, \ldots, x_n)$ as a "distributed database" (as it is not stored in one location, and every x_i is only held locally by user i). This model is used in practice by large corporations to ensure that private data never reaches their servers in the clear.

As minimizing the k-means objective objective is NP-hard (even without privacy constraints), the literature has focused on approximation algorithms, with the current (non-private) state-of-the-art construction of Ahmadian et al. (2017) achieving a multiplicative error of 6.357. That is, the algorithm of Ahmadian et al. (2017) identifies a set of k centers whose cost is no more than $6.357 \cdot \mathrm{OPT}_S(k)$, where $\mathrm{OPT}_S(k)$ denotes the lowest possible cost. Unlike in the non-private literature, it is known that every *differentially private* algorithm for approximating the k-means objective must have an *additive* error, which scales with the diameter of the input space Kaplan and Stemmer (2018). This is true both in the local model and in the trusted-curator model, even for computationally unbounded algorithms. Hence, a standard assumption for private k-means is that the input points come from the d-dimensional ball of radius Λ around the origin $\mathcal{B}(0, \Lambda)$. This is the setting we consider in this work, where we assume that $\Lambda = 1$ in the introduction.

There has been a significant amount of work aimed at constructing differentially private k-means algorithms that work in the *trusted-curator* model.[1] The current state-of-the-art construction by Ghazi et al. (2020) obtains an $O(1)$ multiplicative error (which can be made arbitrarily close to the best non-private error) and $\mathrm{poly}(\log(n), k, d)$ additive error. That is, given a set of n input points $S \in (\mathbb{R}^d)^n$, the algorithm of Ghazi et al. (2020) privately identifies a set of k centers with cost at most $O(1) \cdot \mathrm{OPT}_S(k) + \mathrm{poly}(\log(n), k, d)$.

On the other hand, for the *local* model of differential privacy, only two constructions are available (with provable utility guarantees). The first construction, by Nissim and Stemmer

1. Blum et al. (2005); Nissim et al. (2007); Feldman et al. (2009); McSherry (2010); 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 and Stemmer (2018); Huang and Liu (2018); Kaplan and Stemmer (2018); Shechner et al. (2020); Ghazi et al. (2020); Cohen et al. (2021)

(2018), obtains $O(k)$ multiplicative error and $\approx n^{2/3}$ additive error.[2] In addition to the relatively large multiplicative and additive errors, another downside of the algorithm of Nissim and Stemmer (2018) is that it requires $O(k \cdot \log(n))$ rounds of *interaction* between the users and the untrusted server. Following the work of Nissim and Stemmer (2018), an improved locally-private k-means algorithm was presented by Kaplan and Stemmer (2018), which requires only $O(1)$ rounds of interaction and guarantees a multiplicative error of $O(1)$ and an additive error of $\approx n^{2/3}$. That is, the algorithm of Kaplan and Stemmer (2018) reduced the number of interaction rounds while at the same time reducing the multiplicative error to a constant. However, the additive error still remained large. In this work we reduce the additive error to $\approx \sqrt{n}$ while keeping all other complexities the same, i.e., with $O(1)$ rounds of interaction and with $O(1)$ multiplicative error.[3]

We remark that additive error of $\sqrt{n}$ is what one would expect in the local model of differential privacy, as this turned out to be the correct dependency of the error in n for many other problems, including the *heavy-hitters* problem, *median* estimations, answering *counting queries*, and more. Indeed, in Section 5 we show that every locally-private algorithm for the k-means must have additive error $\Omega(\sqrt{n})$. Hence, our positive result is almost optimal in terms of the dependency of the additive error in the database size n.

1.1 Existing Techniques

Before presenting the new ideas of this work, we need to understand the reasons for why the previous results only achieved an additive error of $\approx n^{2/3}$. To that end, we give here an informal overview of the construction of Kaplan and Stemmer (2018). Let $S = (x_1, \ldots, x_n) \in (\mathbb{R}^d)^n$ be a (distributed) database. At a high level, the algorithm of Kaplan and Stemmer (2018) can be summarized as follows:

1. Privately identify a set of *candidate centers* $Y \subseteq \mathbb{R}^d$ that contains a subset $Y^* \subseteq Y$ of size k with low cost, say $\text{cost}_S(Y^*) \leq O(1) \cdot \text{OPT}_S(k) + \Gamma$ for some error parameter Γ.
2. For every $y \in Y$, let $\#_S(y)$ denote the number of input points $x_i \in S$ such that y is their nearest candidate center, that is
$$\#_S(y) = |\{x \in S : y = \text{argmin}_{y' \in Y} \|x - y'\|\}|,$$
and let $\hat{\#}_S(y)$ be a noisy estimation of $\#_S(y)$, satisfying LDP.
3. Post-process the set of candidate centers and the noisy counts to identify a set C of k centers that approximately minimizes
$$\text{cost}_{Y,\hat{\#}}(C) = \sum\nolimits_{y \in Y} \hat{\#}_S(y) \cdot \min_{c \in C} \cdot \|y - c\|^2.$$

Figure 1: High level overview of the construction of Kaplan and Stemmer (2018).

2. The error bounds stated throughout the introduction are simplified; see Table 1 for more details.
3. Throughout the introduction, we write $\approx \sqrt{n}$ and $\approx n^{2/3}$ to mean $O(n^{1/2+a})$ and $O(n^{2/3+a})$, respectively.

Step 2 is done using standard LDP counting tools (to be surveyed in Section 2). Step 1 is more involved, and we will elaborate on it later. For now, it suffices to say that for any $\sqrt{n} \lesssim \Gamma \lesssim n$, there is an LDP algorithm that is capable of identifying a set Y of size $|Y| \approx n/\Gamma$ that contains a subset of k centers $Y^* \subseteq Y$ such that $\text{cost}_S(Y^*) \lesssim O(1) \cdot \text{OPT}_S(k) + \Gamma$. The analysis then goes by arguing that for every set of k centers D we have that

$$\text{cost}_{Y,\hat{\#}}(D) \approx \text{cost}_{Y,\#}(D) \approx \text{cost}_S(D),$$

where $\text{cost}_{Y,\#}(D)$ is the same as $\text{cost}_{Y,\hat{\#}}(D)$ but with the "true" counts $\#_S(y)$ instead of the estimated counts $\hat{\#}_S(y)$. This means that the set C computed in Step 3 also has a low k-means cost w.r.t. the input points S, and is hence a good output. The main question is how *tight* are these connections. Using the fact that there is a subset $Y^* \subseteq Y$ with low k-means cost w.r.t. S, one can show that the connection $\text{cost}_{Y,\#}(D) \approx \text{cost}_S(D)$ holds, informally, up to an additive error of $O(\Gamma)$.

The difficulty lies in the connection $\text{cost}_{Y,\hat{\#}}(D) \approx \text{cost}_{Y,\#}(D)$. As we mentioned, it is known that estimating counts under LDP generally incurs an additive error of $\Theta(\sqrt{n})$ (ignoring the dependency on all other parameters). As a result, for every $y \in Y$ the estimation error $\left|\hat{\#}_S(y) - \#_S(y)\right|$ might be as big as $\sqrt{n}$. Moreover, when comparing $\text{cost}_{Y,\hat{\#}}(D)$ to $\text{cost}_{Y,\#}(D)$, the different noises "add up". To illustrate this point, let D be a possible set of centers, and observe that

$$\begin{aligned}
\text{cost}_{Y,\hat{\#}}(D) &= \sum_{y \in Y} \hat{\#}_S(y) \cdot \min_{d \in D} \|y - d\|^2 \\
&\lesssim \sum_{y \in Y} \left(\#_S(y) + \sqrt{n}\right) \cdot \min_{d \in D} \|y - d\|^2 \\
&\leq \text{cost}_{Y,\#}(D) + |Y| \cdot \sqrt{n}. \qquad (1)
\end{aligned}$$

Actually, it can be shown that the additive error only increases proportionally to $\sqrt{|Y| \cdot n}$, because of noise cancellations (as the sum of $|Y|$ independent noises only scales with $\sqrt{|Y|}$). In any case, at least with this type of an analysis, the error in the connection $\text{cost}_{Y,\hat{\#}}(D) \approx \text{cost}_{Y,\#}(D)$ scales with $\sqrt{|Y| \cdot n}$. Recall that the error in the other connection $\text{cost}_{Y,\#}(D) \approx \text{cost}_S(D)$ scales with $\Gamma \approx \frac{n}{|Y|}$. That is, the error in one of the two connections grows with $|Y|$, and the error in the second connection decreases with $|Y|$. These two requirements balance at $|Y| \approx n^{1/3}$, which results in an additive error of $\sqrt{n^{1/3} \cdot n} = n/n^{1/3} = n^{2/3}$. In a nutshell, this is the main reason for the large additive error in the construction of Kaplan and Stemmer (2018). The construction of Nissim and Stemmer (2018) suffered from similar issues (although their algorithm is different).

1.2 Our Contributions

The takeaway from the above discussion is that in order to obtain an algorithm with small additive error, for every $y \in Y$, it suffices to ensure that $\hat{\#}_S(y)$ approximates $\#_S(y)$ to within a constant *multiplicative* factor. Indeed, in such a case Inequality (1) would be

replaced with

$$\begin{aligned}\mathrm{cost}_{Y,\hat{\#}}(D) &= \sum_{y\in Y} \hat{\#}_S(y) \cdot \min_{d\in D} \|y-d\|^2 \\ &\leq \sum_{y\in Y} O(1)\cdot \#_S(y) \cdot \min_{d\in D} \|y-d\|^2 \\ &= O(1)\cdot \mathrm{cost}_{Y,\#}(D),\end{aligned}$$

which is acceptable (since we are aiming for a construction with a constant multiplicative error anyways). Observe that, as our noisy estimations are accurate to within an additive error of $\approx \sqrt{n}$, for every $y \in Y$ such that $\#_S(y) \gtrsim \sqrt{n}$ we already have that $\hat{\#}_S(y)$ approximates $\#_S(y)$ to within a constant multiplicative factor. However, this is not the case for candidate centers $y \in Y$ such that $\#_S(y) \ll \sqrt{n}$, and there could be many such candidates.

To summarize our discussion so far, we would like to identify a set Y of candidate centers that satisfies the following two conditions.

Condition 1: $\exists Y^* \subseteq Y$ of size k such that

$$\mathrm{cost}_S(Y^*) \lesssim O(1)\cdot \mathrm{OPT}_S(k) + \sqrt{n}.$$

Condition 2: $\forall y \in Y$ we have $\#_S(y) \gtrsim \sqrt{n}$.

While the set of candidate centers Y computed in the previous works of Nissim and Stemmer (2018); Kaplan and Stemmer (2018) is guaranteed to satisfy the first condition above, we do not have any guarantee w.r.t. the second condition.

First Attempt. One might try to achieve the second condition above by simply deleting every $y \in Y$ such that $\hat{\#}_S(y) \lesssim \sqrt{n}$. This would indeed mean that, after the deletions, for every $y \in Y$ we have that $\#_S(y)$ is at least $\sqrt{n}$. However, this might break condition 1. To see how this could happen, suppose that $k = d = 2$, and consider a collection of points $p_1, \ldots, p_{\sqrt{n}}$ around the point $(1, 0)$, where every two points p_i, p_j are at pairwise distance $\approx \rho$ (infinitely small), and all of them are within distance $\approx \rho$ to the point $(1, 0)$. Now consider a database containing $(n - n^{3/4})$ copies of the point $(0, 0)$, and $n^{1/4}$ copies of every p_i (so that S is of size n). Now suppose that $Y = \{(0, 0), (0, 1), p_1, \ldots, p_{\sqrt{n}}\}$. Since our count estimations are only accurate up to an error of $\sqrt{n}$, we will have that $\hat{\#}_S(0, 0) \approx n - n^{3/4}$, and that $\hat{\#}_S(y) \approx 0$ for every other candidate center in Y. Hence, if we were to delete every $y \in Y$ with a small estimated count, then we would be left only with the point $(0, 0)$, that misses the cluster around $(0, 1)$, and hence $\mathrm{cost}_S(Y) \gtrsim n^{3/4}$, even though $\mathrm{OPT}_S(k) \approx 0$.

Resolution. To overcome this challenge, we revisit the way in which the set of candidate centers Y is constructed. Recall that the construction of Kaplan and Stemmer (2018) (described in Figure 1) is oblivious to the way in which the set of candidate centers Y is constructed (the only requirement is that Y contains a subset Y^* with low k-means cost). For our new construction, we will need to identify additional properties of these candidate centers, that are specific to the way in which they are constructed.

In more details, the set of candidate centers Y is constructed in $\log(n)$ iterations, where during the ith iteration we identify "large" subsets of input points (at least $\gtrsim \sqrt{n}$ points) that can be enclosed in a ball of radius $r = 2^{-i}$, and add to Y a (privacy preserving) estimation for the average of every such subset of clustered input points. As the previous works of Nissim and Stemmer (2018); Kaplan and Stemmer (2018) showed, the set Y constructed in this process contains (w.h.p.) a subset of k centers with low k-means cost. Informally, the additional property that we will leverage is that for every candidate center y that is constructed during the iteration with parameter r there are $\gtrsim \sqrt{n}$ input points within distance $\lesssim r$ to y, which is true by the way in which the candidate centers are constructed.

We will say that such a candidate center y was *created for* the radius r. Moreover, we will say that an input point $x \in S$ *created* y, if x was one of the points that y was computed as their noisy average. So every candidate center has "a lot" ($\gtrsim \sqrt{n}$) input points who created it. Observe that this still does not guarantee that the resulting set Y is such that every $y \in Y$ has "a lot" of neighbors in S, which is what we really wanted. This can happen, e.g., if some (or all) of the points who created the candidate center y have a different candidate center that is closer to them.

To overcome this issue, we assign input points to candidate centers in a different way – not by assigning every input point to its nearest candidate center. Informally, when assigning an input point $x \in S$ to a candidate center we give a slight preference to the candidate centers that x created. We then estimate the *weight* of every candidate center $y \in Y$ according to this new assignment (where the *weight* of a candidate center is the number of input points assigned to it). We show that these new weights still allow us to estimate the cost of every set of centers w.r.t. the input points. In addition, with these new weights, we show that it is possible to delete every candidate center whose weight is lower than $\approx \sqrt{n}$. While this deletion step might delete many centers from Y, and in particular, might even delete the best k centers from Y, we show that for every deleted candidate center y there is a sequence of alternative candidate centers $y_1, y_2, \ldots, y_w$ such that each of them could be a "good substitute" for y and such that at least one of them is *not* deleted from Y. We obtain the following theorem (the details are given in Sections 3 and 4; see Theorem 4.11 for the formal statement).

Theorem 1.1 (**informal**) *There exists an LDP algorithm such that the following holds. When executed on a (distributed) database S containing n points in the d-dimensional unit ball, the algorithm returns a set K of k centers such that with high probability we have*

$$\mathrm{cost}_S(K) \leq O(1) \cdot \mathrm{OPT}_S(k) + \tilde{O}\left(k \cdot \sqrt{d} \cdot n^{0.5+a}\right),$$

where $a > 0$ is an arbitrarily small constant (the constant hiding in the multiplicative error depends on a).

We remark that all of our techniques extend to k-median clustering (this will be made precise in the technical sections). In Section 5 we show that the additive error achieved by our construction is almost optimal. Specifically, we present a lower bound showing that every LDP algorithm for the k-means must have additive error $\Omega(\sqrt{n})$. This lower bound follows from a simple reduction from a task (related to) *counting bits* to the task of

approximating the k-means objective of the data, together with known lower bounds for counting bits under LDP. We obtain the following theorem (see Theorem 5.3 for the formal statement).

Theorem 1.2 (**informal**) *Every LDP algorithm for approximating the k-means objective of a (distributed) database of size n must have additive error $\Omega(\sqrt{n})$.*

1.3 Followup Work

In this work, we design a new locally-private algorithm for the Euclidean k-means and k-median problems. Our algorithms require a constant number of interaction rounds between the users and the untrusted server, and guarantee $O(1)$ multiplicative error and $\approx n^{1/2+a}$ additive error for an arbitrarily small constant $a > 0$. Following our work, Chang et al. (2021) presented a different algorithm that uses only a single round of interaction and obtains improved multiplicative and additive error guarantees. In particular, the algorithm of Chang et al. (2021) obtains additive error $\tilde{O}(\sqrt{n})$ rather than $O\left(n^{1/2+a}\right)$ as in our result.

2. Preliminaries

In k-means clustering we aim to partition n points into k clusters in which each point x belongs to the cluster whose mean is closest to x. We will also consider the k-median clustering objective, where we aim to place centroids at the *median* of every cluster (rather than the mean). Formally, for a set of points $S \in (\mathbb{R}^d)^n$ and a set of centers $C \subseteq \mathbb{R}^d$, the *k-means cost* of C w.r.t. the points S is defined as

$$\mathrm{cost}_S^2(C) = \sum_{x \in S} \min_{c \in C} \|x - c\|^2,$$

and the *k-median cost* is defined as

$$\mathrm{cost}_S^1(C) = \sum_{x \in S} \min_{c \in C} \|x - c\|.$$

For $p \in \{1, 2\}$ and for a *weighted* set $S = \{(x_1, \alpha_1), \ldots, (x_n, \alpha_n)\} \in (\mathbb{R}^d \times \mathbb{R})^n$, the *weighted cost* is

$$\mathrm{cost}_S^p(C) = \sum_{(x,\alpha) \in S} \alpha \cdot \min_{c \in C} \|x - c\|^p.$$

We use $\mathrm{OPT}_S^p(k)$ to denote the lowest possible cost of k centers w.r.t. S. That is,

$$\mathrm{OPT}_S^p(k) = \min_{C \subseteq \mathbb{R}^d,\, |C| = k} \{\mathrm{cost}_S^p(C)\}.$$

2.1 Local Differential Privacy

The local model of differential privacy was formally defined by Dwork et al. (2006b) and Kasiviswanathan et al. (2011). We give here the formulation presented by Vadhan (2016). Consider n parties $P_1, \ldots, P_n$, where each party is holding a data item x_i. We denote $S = (x_1, \ldots, x_n)$ and refer to S as a *distributed database*. A *protocol* proceeds in a sequence

of rounds until all (honest) parties terminate. Informally, in each round, each party selects a message to be broadcast based on its input, internal coin tosses, and all messages received in previous rounds. The *output* of the protocol is specified by a deterministic function of the transcript of messages exchanged. For some $j \in [n]$, we consider an *adversary* controlling all parties other than P_j. Given a particular adversary strategy A, we write $\text{View}_A((A \leftrightarrow (P_1, \ldots, P_n))(S))$ for the random variable that includes everything that A sees when participating in the protocol $(P_1, \ldots, P_n)$ on input $S = (x_1, \ldots, x_n)$.

Definition 2.1 (**Dwork et al. (2006b); Kasiviswanathan et al. (2011); Beimel et al. (2008); Vadhan (2016)**) *A protocol $P = (P_1, \ldots, P_n)$ satisfies (ε, δ)-local differential privacy (LDP) if, for every $j \in [n]$, for every adversary A controlling all parties other than P_j, for every two datasets S, S' that differ on P_j's input (and are equal otherwise), the following holds for every set T:*

$$\Pr[\text{View}_A((A \leftrightarrow (P_1, \ldots, P_n))(S)) \in T] \leq e^{\varepsilon} \cdot \Pr[\text{View}_A((A \leftrightarrow (P_1, \ldots, P_n))(S')) \in T] + \delta.$$

As is standard in the literature on local differential privacy, we will consider protocols in which there is a unique player, called *the server*, which has no inputs. All other players are called *users*. Typically, users do not communicate with other users directly, only with the server.

2.1.1 Counting Queries and Histograms

For a database $S = (x_1, \ldots, x_n) \in X^n$ and a domain element $x \in X$, we use $f_S(x)$ to denote the multiplicity of x in S, i.e., $f_S(x) = |\{x_i \in S : x_i = x\}|$. One of the most basic tasks in the local model of differential privacy is computing *histograms*, in which the goal is to estimate $f_S(x)$ for every domain element x.

Theorem 2.2 (**Hsu et al. (2012); Bassily and Smith (2015); Bassily et al. (2017); Bun et al. (2018)**) *Fix $\beta, \varepsilon \leq 1$. There exists a non-interactive $(\varepsilon, 0)$-LDP protocol that operates on a (distributed) database $S \in X^n$ for some finite set X, and returns a mapping $\hat{f} : X \to \mathbb{R}$ such that the following holds. For every choice of $x \in X$, with probability at least $1 - \beta$, we have that*

$$\left|\hat{f}(x) - f_S(x)\right| \leq \frac{3}{\varepsilon} \cdot \sqrt{n \cdot \log\left(\frac{4}{\beta}\right)}.$$

2.1.2 Composition and Post-Processing

We will later present algorithms (or protocols) that instantiate several differentially private algorithms (or protocols). We will use the following theorems.

Theorem 1 (**Dwork et al. (2006b)**) *Let Π be an (ε, δ)-LDP protocol, and let Π' be a protocol that invokes Π and output an arbitrary function of its output. Then Π' is (ε, δ)-LDP.*

Theorem 2 (**Dwork et al. (2006a, 2010)**) *Let Π be a protocol that consists of k (adaptive) executions of (ε, δ)-LDP protocols. Then Π is $(k\varepsilon, k\delta)$-LDP.*

We remark that stronger composition theorems exist (in terms of the dependency of the resulting privacy guarantees in k), and refer the reader to Dwork et al. (2010) for more details.

3. Candidates with Additional Properties

As we explained in the introduction, the first step in our construction is to privately identify a set Y of *candidate centers*. Our construction makes use of an LDP tool for this task, called `GoodCenters`. This tool, in its original form, was presented by Nissim et al. (2016) for the *trusted-curator* model, and was refined and extended to the local model by Nissim and Stemmer (2018). At a high level, algorithm `GoodCenters` takes a parameter r and works by hashing input points using a *locality sensitive hash function*, that aims to maximize the probability of a collision for "close" items (within distance $\leq r$), while minimizing the probability of collision for "far" items (at distance $\gg r$). If there is a ball of radius r that encloses "a lot" ($\gtrsim \sqrt{n}$) of input points, then we expect that "a lot" of them will be hashed into the same hash value, which would allow us to isolate them and estimate their average with small error.

We identify additional properties of algorithm `GoodCenters`, which will be useful in the following section. Our contribution here is mostly conceptual – in identifying the necessary properties and in showing that they are achieved by the algorithm. Most of the technical details in the construction and in the analysis of `GoodCenters` have already appeared in the works of Nissim and Stemmer (2018); Kaplan and Stemmer (2018). Therefore, here we only state the properties of the algorithm, and present the formal details in the appendix. Our modification to `GoodCenters` is captured by Item 1 in the following theorem.

Theorem 3.1 (**Algorithm `GoodCenters` Nissim et al. (2016); Nissim and Stemmer (2018); Kaplan and Stemmer (2018)**) *For every two constants $a > b > 0$ there exists a constant $c = c(a,b)$ such that the following holds. Let $\beta, \varepsilon, \delta, n, d, \Lambda, r$ be such that $\Lambda/r \leq \mathrm{poly}(n)$ and such that $t \geq O\left(\frac{n^{0.5+a+b}\cdot\sqrt{d}}{\varepsilon}\log(\frac{1}{\beta})\log\left(\frac{dn}{\beta\delta}\right)\right)$. Algorithm `GoodCenters` satisfies (ε,δ)-LDP. Furthermore, let $S = (x_1,\ldots,x_n)$ be a distributed database where every x_i is a point in the d-dimensional ball $\mathcal{B}(0,\Lambda)$, and let `GoodCenters` be executed on S with parameters $r, t, \beta, \varepsilon, \delta$. Denote $M = 4n^a\ln(\frac{1}{\beta})$. The algorithm outputs a partition $I_1,\ldots,I_M \subseteq [n]$, hash functions $h_1,\ldots,h_M$, lists of hash values $L_1,\ldots,L_M$, and sets of centers $Y_1,\ldots,Y_M$, where for every $m \in [M]$ and $u \in L_m$ the set Y_m contains a center $\hat{y}_{m,u}$. In addition,*

1. *With probability at least $1-\beta$, for every $m \in [M]$ and every $u \in L_m$ we have*

$$\left|\left\{ i \in I_m : \begin{array}{c} h_m(x_i) = u \\ \text{and} \\ \|x_i - \hat{y}_{m,u}\| \leq 5cr \end{array} \right\}\right| \geq \frac{t\cdot n^{-b}}{16M}.$$

2. *Denote $Y = \bigcup_{m\in[M]} Y_m$. Then*

$$|Y| \leq \frac{512\cdot n^{1+a+b}}{t}\ln\left(\frac{1}{\beta}\right).$$

3. *Let $P \subseteq S$ be a set of t points that can be enclosed in a ball of radius r. With probability at least $1-\beta$ there exists $\hat{y} \in Y$ such that the ball of radius $5cr$ around $\hat{y}$ contains all of P.*

4. Algorithm `WeightedCenters`

In this section we present our main construction – algorithm `WeightedCenters`. In order to identify the set Y of candidate centers, the algorithm begins by executing algorithm `GoodCenters` on the (distributed) database S multiple times with exponentially growing choices for the parameter r. Recall that an execution of `GoodCenters` with parameter r returns a partition $I_1^r, \ldots, I_M^r \subseteq [n]$, hash functions $h_1^r, \ldots, h_M^r$, lists of hash values $L_1^r, \ldots, L_M^r$, and sets of centers $Y_1^r, \ldots, Y_M^r$, where for every $m \in [M]$ and $u \in L_m^r$ the set Y_m^r contains a center $\hat{y}_{m,u}^r$. By the properties of algorithm `GoodCenters`, with high probability, for every $m \in [M]$ and every $u \in L_m^r$ we have that

$$\left| \left\{ i \in I_m^r : \begin{array}{c} h_m^r(x_i) = u \\ \text{and} \\ \|x_i - \hat{y}_{m,u}^r\| \leq 5cr \end{array} \right\} \right| \geq \frac{t}{16M} \cdot n^{-b}.$$

We introduce the following notation.

Notation 4.1 *Given the outcomes of `GoodCenters` (with parameter r), we say that a point $x_i \in S$ (or, alternatively, that the i^{th} user)* creates *a center $\hat{y}_{m,u}^r \in Y_m^r$ if $i \in I_m^r$ and $h_m^r(x_i) = u$ and $\|x_i - \hat{y}_{m,u}^r\| \leq 5cr$. Observe that a point $x_i \in S$ creates at most one center in $Y^r = Y_1^r \cup \cdots \cup Y_M^r$. If $x_i \in S$ creates a center in Y^r, then we say that x_i* creates a center for the radius r.

Remark 4.2 *Algorithm `GoodCenters` constructs the centers in Y^r by averaging (with noise) subsets of input points. Informally, we think of the set of points who "create" a center $\hat{y} \in Y^r$ as the set of points s.t. $\hat{y}$ was computed as their (noisy) average. The actual definition, however, is a bit different (as stated above).*

After the set Y of candidate centers is constructed, algorithm `WeightedCenters` proceeds by assigning input points to the centers in Y (in a certain way) and estimating the *weight* of every candidate center (where the *weight* of a candidate center is the number of input points assigned to it). Then, the algorithm *re-assigns* the input points to the candidate canters, and *re-estimates* the weights of the candidates. This second iteration of re-assigning points to centers (and re-estimating candidate weights) is done in order to "eliminate" candidates with low weights. Finally, the algorithm post-processes the weighted set of candidate centers in order to produce the final set of centers (this post-processing is done with a non-private algorithm for approximating either the k-means or the k-median cost objectives, as required).

Consider the execution of `WeightedCenters` on a database S. For $p \in \{1,2\}$, let $C^{\mathrm{opt}_p} = (c_1^{\mathrm{opt}_p}, \ldots, c_k^{\mathrm{opt}_p})$ denote an optimal set of centers for S, where $p=1$ corresponds to the k-median objective and $p=2$ corresponds to the k-means objective. Also, for $p \in \{1,2\}$, let

Algorithm `WeightedCenters`

Input: Failure probability β, privacy parameters ε, δ.

Setting: Each player $j \in [n]$ holds a value $x_j \in \mathcal{B}(0, \Lambda)$. Define $S = (x_1, \ldots, x_n)$.

1. ***Constructing candidate centers:*** Denote

$$t = O\left(\frac{1}{\varepsilon} \cdot n^{0.5+a+b} \cdot \sqrt{d} \cdot \log(\frac{\log n}{\beta}) \log\left(\frac{dn}{\beta\delta}\right)\right).$$

   For $r = \Lambda, \frac{\Lambda}{2}, \frac{\Lambda}{4}, \ldots, \frac{\Lambda}{n}$, execute (in parallel) algorithm `GoodCenters` on S with the parameter t and the radius r to obtain sets of centers $Y_1^r, \ldots, Y_M^r$, lists $L_1^r, \ldots, L_M^r$, hash functions $h_1^r, \ldots, h_M^r$, and a partition $I_1^r, \ldots, I_M^r \subseteq [n]$. Each execution of `GoodCenters` is done with privacy parameters $\frac{\varepsilon}{4\log(n)}, \frac{\delta}{\log(n)}$. Denote $Y = \bigcup_{r,m} Y_m^r$.

2. ***Assigning points to candidate centers:*** Define the following assignment of users to centers in Y, denoted as $a(i, x_i)$. To compute $a(i, x_i)$, let $r_i \in \{\Lambda, \frac{\Lambda}{2}, \ldots, \frac{\Lambda}{n}\}$ be the smallest such that x_i *creates* a center for r_i (see Notation 4.1), and let $y(i, x_i, r_i)$ denote this created center. Then, let y_i^* be a center with minimal distance to x_i from $\bigcup_{\substack{r<r_i \\ m\in[M]}} Y_m^r$. Now, if $\|x_i - y_i^*\| < \|x_i - y(i, x_i, r_i)\|$ then $a(i, x_i) = y_i^*$, and otherwise $a(i, x_i) = y(i, x_i, r_i)$.

   % Observe that each user i can compute $a(i, x_i)$ herself from i, x_i and from the publicly released sets of centers $Y_1^r, \ldots, Y_M^r$, lists $L_1^r, \ldots, L_M^r$, hash functions $h_1^r, \ldots, h_M^r$, and partition $I_1^r, \ldots, I_M^r \subseteq [n]$.

3. ***Estimating weights of candidate centers:*** Use an $\frac{\varepsilon}{4}$-LDP algorithm for histograms (see Theorem 2.2) to obtain for every $y \in Y$ an estimation

$$\hat{a}(y) \approx a(y) \triangleq |\{i : a(i, x_i) = y\}|.$$

4. ***Re-assigning points to candidate centers:*** Let

$$W = \left\{y \in Y : \hat{a}(y) \geq \Omega\left(\frac{\sqrt{d}}{\varepsilon} \cdot n^{0.5+a} \cdot \log(\frac{1}{\beta}) \log\left(\frac{dn}{\delta}\right)\right)\right\},$$

   and define $b(i, x_i) = a(i, x_i)$ if $a(i, x_i) \in W$, and otherwise define $b(i, x_i)$ to be an arbitrary center in W with minimal distance to x_i.

5. ***Re-estimating weights of candidate centers:*** Use an $\frac{\varepsilon}{4}$-LDP algorithm for histograms (see Theorem 2.2) to obtain for every $y \in W$ an estimation

$$\hat{b}(y) \approx b(y) \triangleq |\{i : b(i, x_i) = y\}|.$$

6. ***Output:*** Non-privately identify a subset $K \subseteq \mathbb{R}^d$ of size k with low cost w.r.t. the set W and the weights $\hat{b}$ (specifically, with cost at most $O(1)$ times the lowest possible cost).

$S_1^{\text{opt}_p}, \ldots, S_k^{\text{opt}_p}$ be the partition of S induced by these optimal clusters. For $\ell \in [k]$ let

$$r_\ell^{\text{opt}_1} = \frac{2}{|S_\ell^{\text{opt}_1}|} \sum_{x \in S_\ell^{\text{opt}_1}} \|x - c_\ell^{\text{opt}_1}\|,$$

and

$$r_\ell^{\text{opt}_2} = \sqrt{\frac{2}{|S_\ell^{\text{opt}_2}|} \sum_{x \in S_\ell^{\text{opt}_2}} \|x - c_\ell^{\text{opt}_2}\|^2}.$$

Now let $P_\ell^{\text{opt}_p} = S_\ell^{\text{opt}_p} \cap \mathcal{B}(c_\ell^{\text{opt}_p}, r_\ell^{\text{opt}_p})$. Note that for every $\ell \in [k]$ we have that $|P_\ell^{\text{opt}_p}| \geq \frac{1}{2}|S_\ell^{\text{opt}_p}|$, as otherwise less than half of the points in $S_\ell^{\text{opt}_p}$ are within distance $r_\ell^{\text{opt}_p}$ to $c_\ell^{\text{opt}_p}$, and so $\text{cost}^p_{S_\ell^{\text{opt}_p}}(\{c_\ell^{\text{opt}_p}\}) > \frac{|S_\ell^{\text{opt}_p}|}{2} \cdot (r_\ell^{\text{opt}_p})^p = \text{cost}^p_{S_\ell^{\text{opt}_p}}(\{c_\ell^{\text{opt}_p}\})$.

Notation 4.3 *We say that an optimal cluster $S_\ell^{\text{opt}_p}$ is* large *if $|S_\ell^{\text{opt}_p}| \geq \tilde{O}\left(\frac{1}{\varepsilon} \cdot n^{0.5+a+b} \cdot \sqrt{d}\right)$.*

We begin the utility analysis by defining the following two events. The first event states that the executions of `GoodCenters` (in Step 1 of `WeightedCenters`) succeed. Specifically, the set of candidate centers Y (resulting from Step 1) contains a "close enough" center for every *large* optimal cluster, and in addition, for every $y \in Y$ there are "a lot" of users who *created* y.

Event `CREATION` (over the executions of `GoodCenters`):

1. $\forall y \in Y = \bigcup_{r,m} Y_m^r$, there are at least $\tilde{t} = \Omega\left(\frac{\sqrt{nd}}{\varepsilon} \log\left(\frac{dn}{\beta\delta}\right)\right)$ users that create y.
2. For every *large* optimal cluster $S_\ell^{\text{opt}_p}$, the set Y contains a center $y_\ell^* \in Y$ that was created for a radius r_ℓ^* such that $r_\ell^* \leq \max\{2r_\ell^{\text{opt}_p}, \frac{\Lambda}{n}\}$ and $\|y_\ell^* - c_\ell^{\text{opt}_p}\| \leq O(r_\ell^*)$.

Claim 4.4 *Event `CREATION` occurs with probability at least $1 - \beta$.*

Proof Item 1 follows directly from the properties of algorithm `GoodCenters` and a union bound over the different choices for r. For item 2, fix $\ell \in [k]$ such that $S_\ell^{\text{opt}_p}$ is large, and let $r_\ell^* \in \{\Lambda, \frac{\Lambda}{2}, \ldots, \frac{\Lambda}{n}\}$ be the smallest such that $r_\ell^* \geq r_\ell^{\text{opt}_p}$. Note that $r_\ell^* \leq \max\{2r_\ell^{\text{opt}_p}, \frac{\Lambda}{n}\}$. By Theorem 3.1, the execution of `GoodCenters` with the radius r_ℓ^* (during Step 1 of algorithm `WeightedCenters`) identifies a center y_ℓ^* s.t. $\|y_\ell^* - c_\ell^{\text{opt}_p}\| \leq O(r_\ell^*)$ with probability at least $1 - \frac{\beta}{k}$. By a union bound, with probability at least $1 - \beta$, this happens for every large cluster $S_\ell^{\text{opt}_p}$. ■

The next event states that the two executions of LDP histograms (in Steps 3 and 5) succeed.

Event HISTOGRAMS (over the randomness in Steps 3 and 5):
All the estimates computed in Steps 3 and 5 are accurate to within error $O\left(\frac{1}{\varepsilon}\sqrt{n\log(\frac{n}{\beta})}\right)$.

By Theorem 2.2, Event HISTOGRAMS happens with probability at least $1-\beta$. We continue with the analysis assuming that Events CREATION and HISTOGRAMS occur. Recall that in Step 1 we generate the set of candidate centers Y and that in Step 4 we define the subset $W \subseteq Y$ that contains only centers with "large" weights. The next two claims show that for every center $y \in Y$ there exists a center $w \in W$ that is "close enough" to y (even if $y \notin W$). The first claim shows that if $y \in Y \setminus W$ then there is another center $y' \in Y$ that is close to y (but y' might also be missing from W). This will be leveraged in the claim that follows to identify a center *in* W that is close to y.

Claim 4.5 *Let $y \in Y$ be a center that was created with the radius r. If $y \notin W$, then there is another center $y' \in Y$ that was created with a strictly smaller radius $r' < r$ such that $\|y - y'\| \leq O(r)$.*

Proof By Event CREATION, there are at least $\tilde{t} = \Omega\left(\frac{\sqrt{nd}}{\varepsilon}\log\left(\frac{dn}{\beta\delta}\right)\right)$ users who created the center y. Now, since $y \notin W$, it must be that for at least one user i who created y, we have that $a(i, x_i) \neq y$, as otherwise $a(y)$ would be large and y would be in W (by Event HISTOGRAMS, the error in the estimation $\hat{a}(y) \approx a(y)$ is of a lower order). There could be two possible reasons for why $a(i, x_i) \neq y$:

Case (a): User i also created another center y' for a smaller radius $r' < r$. In this case, since user i created both y and y' we have that $\|x_i - y\| \leq O(r)$ and $\|x_i - y'\| \leq O(r')$, and hence $\|y - y'\| \leq O(r + r') = O(r)$ by the triangle inequality.

Case (b): User i did not create a center for any radius smaller than r, but there is a center y' created with radius $r' < r$ (that user i did not create) such that $\|x_i - y'\| < \|x_i - y\|$. Since user i did create y, we have that $\|x_i - y\| \leq O(r)$, and hence, we again have that $\|y - y'\| \leq O(r)$ by the triangle inequality. ■

The next claim applies the previous claim iteratively to identify a sequence of centers beginning from $y \in Y \setminus W$ and ending in a center $w \in W$ such that every two adjacent centers in this sequence are close.

Claim 4.6 *Let $y \in Y$ be a center that was created with the radius r. Then there is a center $w \in W$ such that $\|y - w\| \leq O(r)$.*

Proof First observe that, by the definition of $a(\cdot,\cdot)$, if a user i creates a center y' for $r = \frac{\Lambda}{n}$ (the smallest possible radius) then $a(i, x_i) = y'$. Therefore, for every center y' created with $r = \frac{\Lambda}{n}$ we have that $a(y')$ is large, and hence, y' appears also in W. Now consider a center $y \in Y$ that was created with the radius r. If $y \in W$ then the claim is trivial. Otherwise, by induction using Claim 4.5, there is a sequence of centers $y_1, y_2, \ldots, y_w$ such that

1. $y_1 = y$,
2. $y_w \in W$,
3. $\|y_1 - y_2\| \leq O(r)$ and for $i \geq 1$ we have
$$\|y_i - y_{i+1}\| \leq O\left(2^{-i} \cdot r\right),$$

where Item 3 holds since y_2 was created with a strictly smaller radius than y_1, and y_3 was created with a strictly smaller radius than y_2, and so on. Therefore,
$$\|y - y_w\| \leq O\left(r + \frac{r}{2} + \frac{r}{4} + \dots\right) = O(r).$$

■

The next claim shows that the set W contains a subset of k centers with low cost.

Claim 4.7 *$\exists W^* \subseteq W$ of size $|W^*| = k$ such that*
$$\mathrm{cost}_S^p(W^*) \leq O(1) \cdot \mathrm{OPT}_S^p(k) + \tilde{O}\left(\frac{k\sqrt{d}\Lambda^p}{\varepsilon} \cdot n^{0.5+a+b}\right).$$

Proof Recall that, by Event `CREATION`, for every *large* optimal cluster $S_\ell^{\mathrm{opt}_p}$, the set Y contains a center $y_\ell^* \in Y$, which was created for a radius r_ℓ^*, such that
$$\|y_\ell^* - c_\ell^{\mathrm{opt}_p}\| \leq O(r_\ell^*) = O\left(\max\left\{r_\ell^{\mathrm{opt}_p}\ ,\ \frac{\Lambda}{n}\right\}\right).$$
Let $y_1^*, \dots, y_k^* \in Y$ and $r_1^*, \dots, r_k^*$ denote the aforementioned centers and radiuses (ignoring small clusters). Now, by Claim 4.6, the set W contains centers $w_1^*, \dots, w_k^*$ such that for every large cluster $S_\ell^{\mathrm{opt}_p}$ we have $\|w_\ell^* - y_\ell^*\| \leq O(r_\ell^*) = O(\max\{r_\ell^{\mathrm{opt}_p}, \frac{\Lambda}{n}\})$. Hence, by the triangle inequality we have that $\|w_\ell^* - c_\ell^{\mathrm{opt}_p}\| \leq O(\max\{r_\ell^{\mathrm{opt}_p}, \frac{\Lambda}{n}\})$. Denote $W^* = \{w_1^*, \dots, w_k^*\}$. If it were the case that all of the clusters are large, then we would have that
$$\begin{aligned}
\mathrm{cost}_S^p(W^*) &= \sum_{x \in S} \min_{w \in W^*} \|x - w\|^p \\
&= \sum_{\ell \in [k]} \sum_{x \in S_\ell^{\mathrm{opt}_p}} \min_{w \in W^*} \|x - w\|^p \\
&\leq \sum_{\ell \in [k]} \sum_{x \in S_\ell^{\mathrm{opt}_p}} \|x - w_\ell^*\|^p \\
&\leq \sum_{\ell \in [k]} \sum_{x \in S_\ell^{\mathrm{opt}_p}} O\left(\|x - c_\ell^{\mathrm{opt}_p}\|^p + \|c_\ell^{\mathrm{opt}_p} - w_\ell^*\|^p\right) \\
&= O(1) \cdot \mathrm{OPT}_S^p(k) + O\left(\frac{\Lambda^p}{n^p}\right) + O(1) \cdot \sum_{\substack{\ell \in [k] \\ x \in S_\ell^{\mathrm{opt}_p}}} \left(r_\ell^{\mathrm{opt}_p}\right)^p \\
&= O(1) \cdot \mathrm{OPT}_S^p(k) + O\left(\frac{\Lambda^p}{n^p}\right).
\end{aligned}$$

Now, the cost of a small cluster is at most $\Gamma = O\left(\frac{\sqrt{d}\cdot\Lambda^p}{\varepsilon}\cdot n^{0.5+a+b}\cdot\log(\frac{k}{\beta})\log\left(\frac{dnk}{\beta\delta}\right)\log(n)\right)$, and there could be at most k such small clusters. Taking them into account, we have that

$$\mathrm{cost}_S^p(W^*) \leq O(1)\cdot\mathrm{OPT}_S^p(k) + O\left(k\cdot\Gamma\right).$$

■

In Step 4 of `WeightedCenters` we define an assignment $b(\cdot,\cdot)$ of the input points to the centers in W. If this assignment would simply assign each point to its nearest center in W, then (as W contains a good set of centers by the previous claim) this assignment would trivially have a low cost. However, the assignment $b(\cdot,\cdot)$ does not necessarily match every point to its nearest center. Nevertheless, as the next claim shows, this assignment still has low cost.

Claim 4.8

$$\sum_{i\in[n]}\|b(i,x_i)-x_i\|^p \leq O(1)\cdot\mathrm{OPT}_S^p(k)+O\left(\frac{k\sqrt{d}\cdot\Lambda^p}{\varepsilon}\cdot n^{0.5+a+b}\cdot\log\left(\frac{k}{\beta}\right)\log\left(\frac{dnk}{\beta\delta}\right)\log(n)\right).$$

Proof Fix a large optimal cluster $S_\ell^{\mathrm{opt}_p}$, let $c_\ell^{\mathrm{opt}_p}$ denote its center, and let $x_i \in S_\ell^{\mathrm{opt}_p}$. By Event `CREATION`, a center y_ℓ^* for the cluster $S_\ell^{\mathrm{opt}_p}$ is created with radius $r_\ell^* \leq \max\{2r_\ell^{\mathrm{opt}_p}, \frac{\Lambda}{n}\}$ such that $\|y_\ell^* - c_\ell^{\mathrm{opt}_p}\| \leq O(r_\ell^*)$. (But it is not necessarily the case that x_i created y_ℓ^*.) Let $y(x_i)\in Y$ denote the center with the smallest radius that was created by x_i, and let $r(x_i)$ denote the radius for which $y(x_i)$ was created. Note that $y(x_i)$ might not be in W. There are two cases:

Case (a): $r(x_i) > r_\ell^*$. Then $\|a(i,x_i)-x_i\| \leq \|x_i - y_\ell^*\|$, because $a(i,x_i)$ can take the value y_ℓ^* if it minimizes the distance to x_i. Now, we either have that $b(i,x_i) = a(i,x_i)$ if $a(i,x_i)\in W$, or else $b(i,x_i)$ is set to be the closest center in W to x_i, denoted as $W(x_i)$. So

$$\begin{aligned}
\|x_i - b(i,x_i)\| &\leq \|x_i - a(i,x_i)\| + \|x_i - W(x_i)\| \\
&\leq \|x_i - y_\ell^*\| + \|x_i - W(x_i)\| \\
&\leq \|x_i - c_\ell^{\mathrm{opt}_p}\| + \|c_\ell^{\mathrm{opt}_p} - y_\ell^*\| + \|x_i - W(x_i)\| \\
&\leq \|x_i - c_\ell^{\mathrm{opt}_p}\| + O\left(r_\ell^{\mathrm{opt}_p} + \frac{\Lambda}{n}\right) + \|x_i - W(x_i)\|.
\end{aligned}$$

Case (b): $r(x_i) \leq r_\ell^*$. Then, as x_i created $y(x_i)$,

$$\|x_i - a(i,x_i)\| \leq O(r(x_i)) = O(r_\ell^*)$$

Recall that $a(i,x_i)$ might be missing from W, and hence,

$$\|x_i - b(i,x_i)\| \leq O\left(r_\ell^{\mathrm{opt}_p} + \frac{\Lambda}{n}\right) + \|x_i - W(x_i)\|.$$

So, in any case, we have that

$$\|x_i - b(i,x_i)\| \leq \|x_i - c_\ell^{\mathrm{opt}_p}\| + O\left(r_\ell^{\mathrm{opt}_p} + \frac{\Lambda}{n}\right) + \|x_i - W(x_i)\|.$$

Therefore,

$$\begin{aligned}
\sum_{i\in[n]} \|b(i,x_i) - x_i\|^p &= \sum_{\ell\in[k]} \sum_{x_i \in S_\ell^{\mathrm{opt}_p}} \|b(i,x_i) - x_i\|^p \\
&\leq \sum_{\substack{\ell\in[k] \\ x_i\in S_\ell^{\mathrm{opt}_p}}} O\left(\|x_i - c_\ell^{\mathrm{opt}_p}\|^p + (r_\ell^{\mathrm{opt}_p})^p + \frac{\Lambda^p}{n^p} + \|x_i - W(x_i)\|^p\right) \\
&= O(1)\cdot \mathrm{OPT}_S^p(k) + O\left(\frac{\Lambda^p}{n^p}\right) + O(1)\cdot \mathrm{cost}_S^p(W) \\
&\leq O(1)\cdot \mathrm{OPT}_S^p(k) + O\left(\frac{\Lambda^p}{n^p}\right) + O(1)\cdot \mathrm{cost}_S^p(W^*) \\
&\leq O(1)\cdot \mathrm{OPT}_S^p(k) \\
&\qquad + O\left(\frac{k}{\varepsilon}\cdot n^{0.5+a+b}\cdot\sqrt{d}\cdot\log\left(\frac{k}{\beta}\right)\log\left(\frac{dnk}{\beta\delta}\right)\log(n)\cdot\Lambda^p\right),
\end{aligned}$$

where $W^* \subseteq W$ is a subset of size $|W^*| = k$ that minimizes $\mathrm{cost}_S^p(W^*)$, and where the last inequality follows from Claim 4.7. ■

Recall that we denote $b(w) \triangleq |\{i : b(i,x_i) = w\}|$, and that in Step 5 we obtain estimations $\hat{b}(w) \approx b(w)$ for every $w \in W$. We write B to denote the set W with weights $\{b(w)\}$. That is, B is a multiset of points containing $b(w)$ copies of every $w \in W$. Alternatively, B is the multiset $B = \{b(i,x_i) : i \in [n]\}$. We also write $\hat{B}$ to denote the set W with the noisy weights $\{\hat{b}(w)\}$. These noisy weights might not be integers (and, in principle, could also be negative, but this does not happen when Event `HISTOGRAMS` occurs). The next claim shows that for every set of centers D we have that $\mathrm{cost}_S^p(D) \approx \mathrm{cost}_B^p(D)$.

Claim 4.9 *Denote*

$$\Gamma = \frac{k}{\varepsilon}\cdot n^{0.5+a+b}\cdot\sqrt{d}\cdot\log\left(\frac{k}{\beta}\right)\log\left(\frac{dnk}{\beta\delta}\right)\log(n)\cdot\Lambda^p.$$

For every set of centers $D \subseteq \mathbb{R}^d$ we have

$$\mathrm{cost}_B^p(D) \leq 2\cdot\mathrm{cost}_S^p(D) + O(1)\cdot\mathrm{OPT}_S^p(k) + O(\Gamma),$$

and,

$$\mathrm{cost}_S^p(D) \leq 2\cdot\mathrm{cost}_B^p(D) + O(1)\cdot\mathrm{OPT}_S^p(k) + O(\Gamma).$$

Proof For a set of centers C and a point x we write $C(x)$ to denote the closest neighbor of x in C. By Claim 4.8, for any set of centers $D \subseteq \mathbb{R}^d$ we have,

$$\begin{aligned}
\mathrm{cost}_B^p(D) &= \sum_{i\in[n]} \left\| b(i,x_i) - D\big(b(i,x_i)\big) \right\|^p \\
&\leq \sum_{i\in[n]} \|b(i,x_i) - D(x_i)\|^p \\
&\leq \sum_{i\in[n]} \left(2\cdot\|b(i,x_i) - x_i\|^p + 2\cdot\|x_i - D(x_i)\|^p\right) \\
&\leq O(1)\cdot \mathrm{OPT}_S^p(k) + O\left(\Gamma\right) + 2\cdot\mathrm{cost}_S^p(D),
\end{aligned}$$

and similarly,

$$\begin{aligned}
\mathrm{cost}_S^p(D) &= \sum_{i\in[n]} \|x - D(x)\|^p \\
&\leq \sum_{i\in[n]} \|x - D\big(b(i,x_i)\big)\|^p \\
&\leq \sum_{i\in[n]} \left(2\left\|x - b(i,x_i)\right\|^p + 2\left\|b(i,x_i) - D\big(b(i,x_i)\big)\right\|^p\right) \\
&\leq O(1)\cdot \mathrm{OPT}_S^p(k) + O\left(\Gamma\right) + 2\cdot\mathrm{cost}_B^p(D).
\end{aligned}$$

■

The next claim shows that for every set of centers D we have that $\mathrm{cost}_B^p(D) \approx \mathrm{cost}_{\hat{B}}^p(D)$.

Claim 4.10 *For every set of centers $D \subseteq \mathbb{R}^d$ we have $\frac{1}{2}\mathrm{cost}_B^p(D) \leq \mathrm{cost}_{\hat{B}}^p(D) \leq 2\mathrm{cost}_B^p(D)$.*

Proof First observe that, by Event `HISTOGRAMS`, for every $w \in W$ we have that $\frac{1}{2}b(w) \leq \hat{b}(w) \leq 2b(w)$. To see this, note that by the definition of the set W in Step 4, for every $w \in W$ we have that

$$b(w) \geq \Omega\left(\frac{1}{\varepsilon}\cdot n^{0.5+a}\cdot\sqrt{d}\cdot\log\left(\frac{1}{\beta}\right)\log\left(\frac{dn}{\delta}\right)\right).$$

In addition, by Event `HISTOGRAMS`, for every $w \in W$ we have that

$$|b(w) - \hat{b}(w)| \leq O\left(\frac{1}{\varepsilon}\sqrt{n\cdot\log(\frac{n}{\beta})}\right) \ll b(w),$$

and hence, for every $w \in W$ we have $\frac{1}{2}b(w) \leq \hat{b}(w) \leq 2b(w)$. Now let $D \subseteq \mathbb{R}^d$ be a set of centers. We have that

$$\begin{aligned}
\mathrm{cost}_B^p(D) &= \sum_{w\in W} b(w)\cdot\min_{d\in D}\|w-d\|^p \\
&\leq \sum_{w\in W} 2\cdot\hat{b}(w)\cdot\min_{d\in D}\|w-d\|^p \\
&= 2\cdot\mathrm{cost}_{\hat{B}}^p(D).
\end{aligned}$$

The other direction is symmetric. ■

So, by the last two claims, for every set of centers D we have that $\text{cost}_S^p(D) \approx \text{cost}_B^p(D) \approx \text{cost}_{\hat{B}}^p(D)$. Hence, we can use the (privately computed) weighted set $\hat{B}$ as a proxy in order to identify k centers with low cost w.r.t. S. This is formalized in the following theorem.

Theorem 4.11 *Let $p \in \{1,2\}$. Algorithm* `WeightedCenters` *satisfies (ε,δ)-LDP. In addition, when executed on a (distributed) database S containing n points in the d-dimensional ball $\mathcal{B}(0,\Lambda)$, the algorithm returns a set K of k centers such that with probability at least $1-O(\beta)$ we have*

$$\text{cost}_S^p(K) \leq O(1)\cdot \text{OPT}_S^p(k) + \tilde{O}\left(\frac{k\sqrt{d}\Lambda^p}{\varepsilon} n^{0.5+a+b}\right),$$

where $a > b > 0$ are arbitrarily small constants (the constant hiding in the multiplicative error depends on a and b).

Proof The privacy properties of `WeightedCenters` are straightforward (follow from composition and post-processing). We proceed with the utility analysis. Let $C_S^{\text{opt}_p} \subseteq \mathbb{R}^d$ be a subset of k centers minimizing $\text{cost}_S^p(\cdot)$, and let $C_{\hat{B}}^{\text{opt}_p} \subseteq \mathbb{R}^d$ be a subset of k centers minimizing $\text{cost}_{\hat{B}}^p(\cdot)$. The output of algorithm `WeightedCenters` is a set $K \subseteq \mathbb{R}^d$ of size $|K| = k$ such that $\text{cost}_{\hat{B}}^p(K) \leq O(1)\cdot \text{cost}_{\hat{B}}^p(C_{\hat{B}}^{\text{opt}_p})$. We now show that the set K has low cost w.r.t. S. Denote $\Gamma = \frac{k}{\varepsilon}\cdot n^{0.5+a+b}\cdot\sqrt{d}\cdot\log(\frac{k}{\beta})\log\left(\frac{dnk}{\beta\delta}\right)\log(n)\cdot\Lambda^p$. By Claims 4.9 and 4.10,

$$\text{cost}_S^p(K) \leq O(1)\cdot\text{cost}_{\hat{B}}^p(K) + O(1)\cdot\text{OPT}_S^p(k) + O(\Gamma).$$

Now, by the fact that K approximately minimizes $\text{cost}_{\hat{B}}^p(\cdot)$, the last expression is at most

$$\leq O(1)\cdot\text{cost}_{\hat{B}}^p(C_{\hat{B}}^{\text{opt}_p}) + O(1)\cdot\text{OPT}_S^p(k) + O(\Gamma).$$

Since $C_{\hat{B}}^{\text{opt}_p}$ minimizes $\text{cost}_{\hat{B}}^p(\cdot)$, the last expression is at most

$$\leq O(1)\cdot\text{cost}_{\hat{B}}^p(C_S^{\text{opt}_p}) + O(1)\cdot\text{OPT}_S^p(k) + O(\Gamma).$$

Finally, by using again Claims 4.9 and 4.10, we get that the last expression is at most

$$\begin{aligned}&\leq O(1)\cdot\text{cost}_S^p(C_S^{\text{opt}_p}) + O(1)\cdot\text{OPT}_S^p(k) + O(\Gamma)\\ &= O(1)\cdot\text{OPT}_S^p(k) + O(\Gamma).\end{aligned}$$

■

Protocol $\mathcal{B}$

Setting: Each user $i \in [n]$ holds a bit $b_i \in \{0,1\}$. Define $S = (b_1, \ldots, b_n)$.

Parameter: $r = \beta/4$, where β is the constant from Theorem 5.2.

1. **The server:** Let $R = \{[0,r],\ [r,2r], \ldots, [1-r,1]\}$, sample (uniformly) an interval $I \in R$, and let μ denote the center of I. Send μ to all the users.
2. **Every user i:** If $b_i = 0$ then ignore μ and set $x_i = 0$. Otherwise, set $x_i = \mu$.
2. **The server and the users:** Execute protocol $\mathcal{A}$ on the database $X = (x_1, \ldots, x_n)$ to obtain a set of centers $C = \{c_1, c_2\}$.
3. **The server:** If $c_1 \in I$ or $c_2 \in I$ then return 1. Otherwise return 0.

5. A Lower Bound on the Additive Error

In this section we present a simple lower bound on the error of every LDP algorithm for approximating the k-means (or the k-median) cost objective. To get our lower bound, we show a reduction from the following problem, called *Gap-Threshold*, to the k-means problem, and then use an existing lower bound for the Gap-Threshold problem.

Definition 5.1 (**Beimel et al. (2008)**) *For $\tau > 0$ and $x_1, \ldots, x_n \in \{0,1\}$,*

$$\text{GAP-TR}_\tau(x_1, \ldots, x_n) = \begin{cases} 0, & \text{If} \quad \sum_{i\in[n]} x_i = 0 \\ 1, & \text{If} \quad \sum_{i\in[n]} x_i \geq \tau \end{cases}$$

Note that $\text{GAP-TR}_\tau(x_1, \ldots, x_n)$ *is not defined when* $0 < \sum_{i\in[n]} x_i < \tau$.

Theorem 5.2 (**Beimel et al. (2008); Joseph et al. (2019)**) *There exist constants $0 < \varepsilon, \beta < 1$ such that the following holds. Let $\delta = o\left(\frac{1}{n^2 \log n}\right)$ and let $\mathcal{A}$ be an (ε,δ)-LDP protocol for computing* GAP-TR_τ *with success probability $1-\beta$. Then $\tau = \Omega(\sqrt{n})$.*

Theorem 5.2 is stated in (Beimel et al., 2008) for $(\varepsilon, 0)$-LDP protocols with a bounded number of interaction rounds. The extension to arbitrary LDP protocols follows from (Joseph et al., 2019, Theorem 5.3). We now show that this theorem implies a lower bound of $\Omega(\sqrt{n})$ on the additive error of LDP algorithms for the k-means or the k-median, even when the dimension d is 1 and $k = 2$.

Theorem 5.3 *There exists constant $0 < \varepsilon, \beta < 1$ such that the following holds. Let $\delta = o\left(\frac{1}{n^2 \log n}\right)$, and let $\mathcal{A}$ be an (ε,δ)-LDP protocol such that for any (distributed) database $X \in ([0,1])^n$, with probability $1-\beta$ the algorithm outputs a set C of $k = 2$ centers satisfying $\text{cost}^p_X(C) \leq \gamma \cdot \text{OPT}^p_X(k) + \tau$, where $\gamma < \infty$ and $p \in \{1,2\}$. Then $\tau = \Omega(\sqrt{n})$.*

Proof Let β, ε be the constants from Theorem 5.2. Let $\mathcal{A}$ be an (ε,δ)-LDP protocol that operates on a (distributed) database $X \in ([0,1])^n$ and outputs a set C of size $k = 2$. We use $\mathcal{A}$ to construct a protocol for GAP-TR, described in protocol $\mathcal{B}$.

By Theorem 5.2, there exists an error parameter $\tau = \Omega(\sqrt{n})$, and a database $S^* \in \{0,1\}^n$ such that $\mathcal{B}(S^*) \neq \text{GAP-TR}_\tau(S^*)$ with probability at least β. We now show that S^* cannot

be the all zero database. To that end, observe that if the input S is the all zero database, then all of the users in the protocol $\mathcal{B}$ ignore μ, and hence, algorithm $\mathcal{A}$ gets no information about the selected interval I. In that case, the probability that one of c_1, c_2 falls in I is at most $2r$. That is, the probability that $\mathcal{B}(\vec{0}) \neq 0 = \text{GAP-TR}_\tau(\vec{0})$ is at most $2r = \beta/2$. Therefore, the database S^* (on which $\mathcal{B}$ errs with probability at least β) is *not* the all zero database $\vec{0}$. Hence, S^* contains at least τ ones. Therefore, whenever $\mathcal{B}$ errs on S^*, we have that $\text{cost}^p_X(C) \geq \left(\frac{r}{2}\right)^p \cdot \tau = \Omega(\sqrt{n})$, even though $\text{OPT}^p_X(k) = 0$. This happens with probability at least β, which completes the proof. ■

Acknowledgments

The author was partially supported by the Israel Science Foundation (grant No. 1871/19), and by the Cyber Security Research Center at Ben-Gurion University of the Negev. The author would like to thank Haim Kaplan and the anonymous reviewers for their helpful comments.

Appendix A.

In this section we provide the details that were omitted from Section 3, and prove Theorem 3.1.

A.1 Additional Preliminaries

A.1.1 Average of Vectors in $\mathbb{R}^d$ Under LDP

Consider a (distributed) database $X = (x_1, \ldots, x_n)$ where every user i is holding $x_i \in \mathbb{R}^d$. One of the most basic tasks we can apply under local differential privacy is to compute a noisy estimation for the sum (or the average) of vectors in X. Specifically, every user sends the server a noisy estimation of its vector (e.g., by adding independent Gaussian noise to each coordinate), and the server simply sums all of the noisy reports to obtain an estimation for the sum of X.

Theorem A.1 (**folklore**) *Consider a (distributed) database $X = (x_1, \ldots, x_n)$ where every user i is holding a point x_i in the d dimensional ball $\mathcal{B}(0, \Lambda)$. There exists an (ε, δ)-LDP protocol for computing an estimation a for the sum of the vectors in X, such that with probability at least $(1 - \beta)$ we have*

$$\left\| a - \sum_{i \in [n]} x_i \right\| \leq \frac{2\Lambda\sqrt{nd}\ln(\frac{2}{\beta\delta})}{\varepsilon}.$$

For our constructions we will need a tool for computing averages of *subsets* of X. Specifically, assume that there are n users, where user i is holding a point $x_i \in \mathbb{R}^d$. Moreover, assume that we have a fixed (publicly known) partition of $\mathbb{R}^d$ into a finite number of regions: $R_1, \ldots, R_T \subseteq \mathbb{R}^d$. For every region R_ℓ, we would like to obtain an estimation for the average

of the input points in that region. This can be done using the following simple protocol, called LDP-AVG.

Protocol LDP-AVG

Public parameters: Partition of $\mathbb{R}^d$ into t regions $R_1, \ldots, R_T$.

Setting: Each user $i \in [n]$ holds a point $x_i \in \mathbb{R}^d$. Define $X = (x_1, \ldots, x_n)$.

1. **Every user i:** Let $y_i = (y_{i,1}, \ldots, y_{i,T}) \in (\mathbb{R}^d)^T$ be a vector whose every coordinate is an independent Gaussian noise. Specifically, every $y_{i,t} \in \mathbb{R}^d$ is a vector whose every coordinate is sampled i.i.d. from $N(0, \sigma_t^2)$, for $\sigma_t = \frac{8 \cdot \mathrm{diam}(R_t)}{\varepsilon} \sqrt{\ln(1.25/\delta)}$. Let t be s.t. $x_i \in \mathbb{R}_t$. Add x_i to $y_{i,t}$. Send y_i to the server.
2. **The server and the users:** Run the protocol from Theorem 2.2 with privacy parameter $\frac{\varepsilon}{2}$. For every $t \in [T]$ the server obtains an estimation $\hat{r}_t \approx |\{i : x_i \in R_t\}| \triangleq r_t$.
3. **The server:** Output a vector $\hat{a} \in (\mathbb{R}^d)^T$, where $\hat{a}_t = \frac{1}{\hat{r}_t} \cdot \sum_{i \in [n]} y_{i,t}$.

Claim A.2 *LDP-AVG satisfies (ε, δ)-LDP. Moreover, with probability at least $(1 - \beta)$, for every $t \in [T]$ s.t. $r_t \geq \frac{12}{\varepsilon} \cdot \sqrt{n \cdot \log\left(\frac{4T}{\beta}\right)}$ we have that*

$$\left\| \frac{1}{\hat{r}_t} \cdot \sum_{i \in [n]} y_{i,t} - \frac{1}{r_t} \cdot \sum_{\substack{i \in [n]: \\ x_i \in R_t}} x_i \right\| \leq \frac{36\sqrt{dn} \cdot \ln(\frac{8dT}{\beta\delta})}{\varepsilon \cdot r_t} \cdot \mathrm{diam}(R_t).$$

Proof The privacy properties of LDP-AVG follow from the privacy properties of the Gaussian mechanism and the protocol from Theorem 2.2, together with composition. Observe that by Theorem 2.2, with probability at least $(1 - \frac{\beta}{2})$, for every $t \in [T]$ we have $|r_t - \hat{r}_t| \leq \frac{6}{\varepsilon} \cdot \sqrt{n \cdot \log\left(\frac{4T}{\beta}\right)}$. We continue with the analysis assuming that this is the case. Fix $t \in [T]$ such that $r_t \geq \frac{12}{\varepsilon} \cdot \sqrt{n \cdot \log\left(\frac{4T}{\beta}\right)}$, and observe that $\hat{r}_t \geq r_t/2$. Denote $\Delta_t = \mathrm{diam}(R_t)$. Using a standard tail bound for normal variables, with probability at least $1 - \frac{\beta}{2T}$ we have that $\left\| \sum_{i \in [n]} y_{i,t} - \sum_{\substack{i \in [n]: \\ x_i \in R_t}} x_i \right\| \leq \frac{12\sqrt{dn}\Lambda_t \cdot \ln(\frac{4dT}{\beta\delta})}{\varepsilon}$. Hence,

$$\left\| \frac{1}{\hat{r}_t} \cdot \sum_{i \in [n]} y_{i,t} - \frac{1}{r_t} \cdot \sum_{\substack{i \in [n]: \\ x_i \in R_t}} x_i \right\| \leq \left| \frac{1}{\hat{r}_t} - \frac{1}{r_t} \right| \cdot \left\| \sum_{i \in [n]} y_{i,t} \right\| + \left\| \frac{1}{r_t} \cdot \left(\sum_{i \in [n]} y_{i,t} - \sum_{\substack{i \in [n]: \\ x_i \in R_t}} x_i \right) \right\|$$

$$
\begin{aligned}
&\leq \left|\frac{1}{\hat{r}_t}-\frac{1}{r_t}\right| \cdot \left(\frac{12\sqrt{dn}\Lambda_t \cdot \ln(\frac{4dT}{\beta\delta})}{\varepsilon} + \left\| \sum_{\substack{i\in[n]:\\ x_i\in R_t}} x_i \right\| \right) + \frac{12\sqrt{dn}\Lambda_t \cdot \ln(\frac{4dT}{\beta\delta})}{\varepsilon \cdot r_t} \\
&= \left|\frac{1}{\hat{r}_t}-\frac{1}{r_t}\right| \cdot \frac{12\sqrt{dn}\Lambda_t \cdot \ln(\frac{4dT}{\beta\delta})}{\varepsilon} + \left|\frac{1}{\hat{r}_t}-\frac{1}{r_t}\right| \cdot \left\| \sum_{\substack{i\in[n]:\\ x_i\in R_t}} x_i \right\| + \frac{12\sqrt{dn}\Lambda_t \cdot \ln(\frac{4dT}{\beta\delta})}{\varepsilon \cdot r_t} \\
&\leq \frac{1}{r_t} \cdot \frac{12\sqrt{dn}\Lambda_t \cdot \ln(\frac{4dT}{\beta\delta})}{\varepsilon} + \frac{|r_t-\hat{r}_t|}{|r_t\cdot\hat{r}_t|} \cdot \sum_{\substack{i\in[n]:\\ x_i\in R_t}} \|x_i\| + \frac{12\sqrt{dn}\Lambda_t \cdot \ln(\frac{4dT}{\beta\delta})}{\varepsilon \cdot r_t} \\
&\leq \frac{24\sqrt{dn}\Lambda_t \cdot \ln(\frac{4dT}{\beta\delta})}{\varepsilon \cdot r_t} + \frac{|r_t-\hat{r}_t|}{|r_t\cdot\hat{r}_t|} \cdot \Lambda_t \cdot r_t \\
&\leq \frac{24\sqrt{dn}\Lambda_t \cdot \ln(\frac{4dT}{\beta\delta})}{\varepsilon \cdot r_t} + \frac{12\Lambda_t \cdot \sqrt{n \cdot \log\left(\frac{8T}{\beta}\right)}}{\varepsilon \cdot r_t} \\
&\leq \frac{36\sqrt{dn}\Lambda_t \cdot \ln(\frac{8dT}{\beta\delta})}{\varepsilon \cdot r_t}.
\end{aligned}
$$

The claim now follows from a union bound. ■

A.1.2 Random Rotation

We also use the following technical lemma to argue that if a set of points P is contained within a ball of radius r in $\mathbb{R}^d$, then by randomly rotating the Euclidean space we get that (w.h.p.) P is contained within an axis-aligned rectangle with side-length $\approx r/\sqrt{d}$.

Lemma A.3 (**e.g., Vazirani and Rao**) *Let $P \in (\mathbb{R}^d)^m$ be a set of m points in the d dimensional Euclidean space, and let $Z = (z_1, \ldots, z_d)$ be a random orthonormal basis for $\mathbb{R}^d$. Then,*

$$\Pr_Z\left[\forall x,y\in P:\ \forall 1\leq i\leq d:\ |\langle x-y, z_i\rangle| \leq 2\sqrt{\ln(dm/\beta)/d}\cdot\|x-y\|\right] \geq 1-\beta.$$

A.1.3 Locality Sensitive Hashing

A locality sensitive hash function aims to maximize the probability of a collision for similar items, while minimizing the probability of collision for dissimilar items. Formally,

Definition A.4 (**Indyk and Motwani (1998)**) *Let $\mathcal{M}$ be a metric space, and let $r>0$, $c>1$, $0\leq q<p\leq 1$. A family $\mathcal{H}$ of functions mapping $\mathcal{M}$ into domain U is an (r, cr, p, q) locality sensitive hashing family (LSH) if for all $x, y \in \mathcal{M}$ (i) $\Pr_{h\in_R\mathcal{H}}[h(x)=h(y)] \geq p$ if $d_{\mathcal{M}}(x,y)\leq r$; and (ii) $\Pr_{h\in_R\mathcal{H}}[h(x)=h(y)]\leq q$ if $d_{\mathcal{M}}(x,y)\geq cr$.*

A.2 Algorithm `CentersProcedure`

Before presenting algorithm `GoodCenters` and its analysis, we introduce the following procedure, called `CentersProcedure`, which is the main ingredient in the construction of algorithm `GoodCenters`. This procedure identifies a set of candidate centers that, with noticeable probability, "captures" every large enough cluster of input points. In algorithm `GoodCenters`, we will later apply `CentersProcedure` multiple times to boost the success probability. The privacy properties of `CentersProcedure` are immediate (follow from composition), and are specified in the following observation.

In more detail, algorithm `CentersProcedure` first samples a locality sensitive hash function h, identifies (using standard LDP tools for histograms) a list of hash values that are "heavy" in the sense that many of the users' input are mapped to these values. Then, for every such heavy hash value, the algorithm averages (with noise to ensure LDP) all the input points that are mapped (by h) to this hash value. In order to reduce the noise incurred by averaging, before applying the LDP averaging tool, the algorithm encloses each cluster of input points that corresponds to a heavy hash value in a small box (with random rotation).

Observation A.5 *Algorithm `CentersProcedure` satisfies ε-LDP.*

We now proceed with the utility analysis of algorithm `CentersProcedure`. We will assume the existence of a family $\mathcal{H}$ of $(r, cr, p{=}n^{-b}, q{=}n^{-2-a})$-sensitive hash functions mapping $\mathbb{R}^d$ to a universe U, for some constants $a > b$, $r > 0$, and $c > 1$.

Lemma A.6 *Let $\beta, \varepsilon, \delta, n, d, \Lambda, r$ be such that $\Lambda/r \leq \mathrm{poly}(n)$ and $t \geq O\left(\frac{n^{0.5+b}\cdot\sqrt{d}}{\varepsilon}\log\left(\frac{dn}{\beta\delta}\right)\right)$ and $\beta \leq n^{-a}/28$. Let $S = (x_1, \ldots, x_n)$ be a distributed database where every x_i is a point in the d-dimensional ball $\mathcal{B}(0, \Lambda)$, and let `CentersProcedure` be executed on S with the family $\mathcal{H}$ and with parameters $r, t, \beta, \varepsilon, \delta$. The algorithm outputs a list of hash values L, a hash function h, and a set Y containing a center $\hat{y}_u$ for every $u \in L$, such that*

1. *The list L and the set Y are size at most $\frac{32\cdot n^{1+b}}{t}$ each.*
2. *With probability at least $1-\beta$, for every $u \in L$ we have*

$$|\{x \in S : h(x) = u \text{ and } \|x - \hat{y}_u\| \leq 5cr\}| \geq \frac{t}{8}\cdot n^{-b}.$$

3. *Let $P \subseteq S$ be a set of t points which can be enclosed in a ball of radius r. With probability at least $n^{-a}/4$ there exists $u^* \in L$ such that the ball of radius $3cr$ around $\hat{y}_{u^*} \in Y$ contains at least one point from P.*

Remark 3 *Lemma A.6 can be interpreted as follows. Item 3 states that, with noticeable probability, the set of candidate Y "captures" every large enough cluster P of input points that can be enclosed in a ball of radius r, in the sense that Y contains a candidate center that is close to this cluster. Item 1 states that the number of candidate centers (i.e., the size of Y) is not too big. Item 2 states that every candidate center $y \in Y$ corresponds to a hash value $u \in L$ such that there are "a lot" of input points that are hashed to u and are "close" to y.*

Algorithm `CentersProcedure`

Input: Radius r, target number of points t, failure probability β, privacy parameter ε.

Tool used: Family $\mathcal{H}$ of $(r, c \cdot r, p, q)$-locality sensitive hash functions mapping $\mathbb{R}^d$ to a universe U.

Setting: Each player $j \in [n]$ holds a value $x_j \in \mathcal{B}(0, \Lambda)$. Define $S = (x_1, \ldots, x_n)$.

1. Sample a hash function $h \in \mathcal{H}$ mapping $\mathbb{R}^d$ to U.
2. Use Theorem 2.2 with $\frac{\varepsilon}{4}$ to identify a list $L \subseteq U$ such that
   (a) Every $u \in U$ s.t. $|\{x \in S : h(x) = u\}| \geq \frac{t}{16} \cdot n^{-b}$ is in L,
   (b) For every $u \in L$ we have $|\{x \in S : h(x) = u\}| \geq \frac{t}{32} \cdot n^{-b}$,
   (c) The list L is of size at most $32n^{1+b}/t$.
3. Let $Z = (z_1, \ldots, z_d)$ be a random orthonormal basis of $\mathbb{R}^d$, and denote $p = 2rc\sqrt{\ln(\frac{dn}{\beta})/d}$. Also let $\mathcal{I} = \{I_1, I_2, \ldots, I_{4\Lambda/p}\}$ be a partition of $[-2\Lambda, 2\Lambda]$ into intervals of length p.
4. Randomly partition S into subsets $S^1, \ldots, S^d$ of size $|S^i| = \frac{n}{d}$. For every basis vector $z_i \in Z$, use Theorem 2.2 with $\frac{\varepsilon}{4}$ to obtain for every pair $(I, u) \in \mathcal{I} \times U$ an estimation $a_i(I, u)$ for
$$|\{x \in S^i : h(x) = u \text{ and } \langle x, z_i \rangle \in I\}|.$$
5. For every basis vector $z_i \in Z$ and for every hash value $u \in L$, denote $I(i, u) = \text{argmax}_{I \in \mathcal{I}}\{a_i(I, u)\}$, and define the interval $\hat{I}(i, u)$ by extending $I(i, u)$ by p to each direction (that is, $\hat{I}(i, u)$ is of length $3p$).
6. For every hash value $u \in L$, let $B(u)$ denote the box in $\mathbb{R}^d$ whose projection on every axis $z_i \in Z$ is $\hat{I}(i, u)$.
7. Use algorithm `LDP-AVG` to obtain, for every $u \in L$, an approximation $\hat{y}_u$ for the average of $\{x \in S : h(x) = u \text{ and } x \in B(u)\}$.
8. Use Theorem 2.2 with $\frac{\varepsilon}{4}$ to identify for every $u \in L$ an estimation $\hat{v}(u) \approx v(u) \triangleq |\{x \in S : h(x) = u \text{ and } \|x - \hat{y}_u\| \leq 5cr\}|$. Delete from L every element $u \in L$ such that $\hat{v}(u) \leq \frac{t}{4} \cdot n^{-b}$.
9. Output the set of centers $Y = \{\hat{y}_u : u \in L\}$, the list L, and the hash function h.

Proof Items 1 and 2 of the lemma follow from the fact that in Step 8 we delete from the list L every element u that does not satisfy the condition of item 2. Specifically, for $t \geq O\left(\frac{1}{\varepsilon} n^{0.5+b} \sqrt{\log(\frac{1}{\beta})}\right)$, our estimations in Step 8 are accurate enough such that item 2 holds with probability at least $1 - \beta$, in which case the list L is short (a longer list can be trimmed). We now proceed with the analysis of item 3.

First observe that, w.l.o.g., we can assume that the range U of every function in $\mathcal{H}$ is of size $|U| \leq n^3$. If this is not the case, then we can simply apply a (pairwise independent) hash function with range n^3 onto the output of the locally sensitive hash function. Clearly, this will not decrease the probability of collusion for "close" elements (within distance r),

and moreover, this can increase the probability of collusion for "non-close" elements (at distance at least cr) by at most $n^{-3} = o(n^{-2-a}) = o(q)$.

Now recall that by the properties of the family $\mathcal{H}$, for every $x, y \in \mathbb{R}^d$ s.t. $\|x-y\| \geq cr$ we have that $\Pr_{h\in\mathcal{H}}[h(x) = h(y)] \leq q = n^{-2-a}$. Using the union bound we get

$$\Pr_{h\in_R\mathcal{H}}[h(x) \neq h(y) \text{ for all } x, y \in S \text{ s.t. } \|x-y\| \geq cr] \geq (1 - n^{-a}/2).$$

Let $P \subseteq S$ denote the guaranteed set of t input points that are contained in a ball of radius r, and let $x \in P$ be an arbitrary point in P. By linearity of expectation, we have that

$$\mathop{\mathbb{E}}_{h\in\mathcal{H}}[|\{y \in P : h(y) \neq h(x)\}|] \leq t(1-p) = t(1-n^{-b}).$$

Hence, by Markov's inequality,

$$\Pr_{h\in\mathcal{H}}\left[|\{y \in P : h(y) \neq h(x)\}| \geq \frac{t(1-n^{-b})}{1-n^{-a}}\right] \leq 1 - n^{-a}.$$

So,

$$\Pr_{h\in\mathcal{H}}\left[|\{y \in P : h(y) = h(x)\}| \geq t\left(1 - \frac{1-n^{-b}}{1-n^{-a}}\right)\right] \geq n^{-a}.$$

Simplifying, for large enough n (specifically, for $n^{a-b} \geq 2$) we get

$$\Pr_{h\in\mathcal{H}}\left[|\{y \in P : h(y) = h(x)\}| \geq \frac{t}{2}\cdot n^{-b}\right] \geq n^{-a}.$$

So far we have established that with probability at least $n^{-a}/2$ over the choice of $h \in \mathcal{H}$ in Step 1 the following events occur:

(E_1) For every $x, y \in S$ s.t. $\|x-y\| \geq cr$ it holds that $h(x) \neq h(y)$; and,

(E_2) There exists a hash value in U, denoted u^*, such that $|\{y \in P : h(y) = u^*\}| \geq \frac{t}{2}\cdot n^{-b}$.

Event (E_1) states that if two points in S are mapped into the same hash value, then these points are close. Event (E_2) states that there is a "heavy" hash value $u^* \in U$, such that "many" of the points in P are mapped into u^*. We proceed with the analysis assuming that these two events occur.

On step 2, we identify a list L containing all such "heavy" hash values $u \in U$. Assuming that $t \geq O\left(\frac{1}{\varepsilon}\cdot n^{0.5+b}\cdot\sqrt{\log(n/\beta)}\right)$, Theorem 2.2 ensures that with probability at least $1-\beta$ we have that $u^* \in L$. We continue with the analysis assuming that this is the case.

On Step 3 we generate a random orthonormal basis Z. By Lemma A.3, with probability at least $(1-\beta)$, for every $x, y \in S$ and for every $z_i \in Z$, we have that the projection of $(x-y)$ onto z_i is of length at most $2\sqrt{\ln(dn/\beta)/d}\cdot\|x-y\|$. In particular, for every hash value $u \in L$ we have that the projection of $S_u \triangleq \{x \in S \ : \ h(x) = u\}$ onto every axis $z_i \in Z$ fits within an interval of length at most $p = 2rc\sqrt{\ln(dn/\beta)/d}$. Recall that we assume that input point come from $\mathcal{B}(0,\Lambda)$. Hence, for every $x, y \in S$ we have $(x-y) \in \mathcal{B}(0, 2\Lambda)$. Now, as $\mathcal{I} = \{I_1, I_2, \dots\}$ is a partition of $[-2\Lambda, 2\Lambda]$ into intervals of length p, for every axis $z_i \in Z$

and for every $u \in U$, we have that the projection of S_u onto z_i is contained within 1 or 2 consecutive intervals from $\mathcal{I}$.

On step 4 we partition S into d subsets $S^i \subseteq S$ of size $\frac{n}{d}$. By the Hoeffding bound, assuming that $t \geq 2 \cdot n^{0.5+b} \cdot \sqrt{2d \ln(\frac{2d}{\beta})}$, with probability at least $1-\beta$, for every $i \in [d]$, we have that $|S^i \cap S_{u^*}| \geq \frac{|S_{u^*}|}{2d} \geq \frac{t \cdot n^{-b}}{4d}$. Recall that the projection of S_{u^*} onto every axis $z_i \in Z$ fits within (at most) 2 consecutive intervals from $\mathcal{I}$. Hence, for every axis $z_i \in Z$, at least 1 interval from $\mathcal{I}$ contains at least half of the points from $S^i \cap S_{u^*}$, i.e., at least $\frac{t \cdot n^{-b}}{8d}$ points. Therefore, for $t \geq O\left(\frac{1}{\varepsilon} \cdot n^{0.5+b} \cdot \sqrt{d \cdot \log(\frac{dn}{\beta})}\right)$, Theorem 2.2 ensures that with probability at least $1-\beta$, for every $z_i \in Z$ we have that $I(i,u^*) = \text{argmax}_{i \in \mathcal{I}}\{a_i(I,u^*)\}$ (defined on step 5) contains at least one point from S_{u^*}.[4] Hence, the interval $\hat{I}(i,u^*)$ obtained by extending $I(i,u^*)$ by p to each direction, contains (the projection of) *all* of the points from S_{u^*} (onto the i^{th} axis). As a result, the box $B(u^*)$, defined on step 7 as the box whose projection onto every axis i is $\hat{I}(i,u^*)$, contains *all* of S_{u^*}. We continue with the analysis assuming that this is the case. Observe that the diameter of $B(u^*)$, as well as the diameter of every other box $B(u)$ defined on step 6, is at most $3p\sqrt{d} = 6cr\sqrt{\ln(dn/\beta)} = \tilde{O}(cr)$.

On step 7 we use algorithm `LDP-AVG` to obtain, for every $u \in L$, an estimation $\hat{y}_u$ for the average of $\{x \in S : h(x) = u \text{ and } x \in B(u)\}$. Let us denote the true average of every such set as y_u. By the properties of `LDP-AVG` (Claim A.2), assuming that $t \geq O\left(\frac{1}{\varepsilon} \cdot \sqrt{dn} \cdot \log\left(\frac{dn}{\beta\delta}\right)\right)$, with probability at least $1-\beta$ we have that $\|y_{u^*} - \hat{y}_{u^*}\|_2 \leq cr$. We continue with the analysis assuming that this is the case.

Observe that y_{u^*} is the average of (some of) the points in S_{u^*}, and that every two points in S_{u^*} are within distance cr from each other. Hence, we get that a ball of radius $2cr$ around y_{u^*} contains all of S_{u^*}. In particular, as S_{u^*} contains at least some of the points from P (the guaranteed cluster radius r with t input points from S), we have that the ball of radius $2cr$ around y_{u^*} contains at least 1 point from P, and that the ball of radius $4cr$ around y_{u^*} contains *all* of P. Therefore, as $\|y_{u^*} - \hat{y}_{u^*}\|_2 \leq cr$ we get that a ball of radius $3cr$ around $\hat{y}_{u^*}$ contains at least one point from P, and that the ball of radius $5cr$ around $\hat{y}_{u^*}$ contains all of P.

Recall that, by Event (E_2), there are at least $\frac{t}{2} \cdot n^{-b}$ input points $x \in P$ such that $h(x) = u^*$. Therefore, in Step 8 we have that $v(u^*) \geq \frac{t}{2} \cdot n^{-b}$. Therefore, with probability at least $1-\beta$ we also have that $\hat{v}(u^*) \geq \frac{t}{2} \cdot n^{-b}$, because when $t \geq O\left(\frac{1}{\varepsilon} n^{0.5+b} \sqrt{\log(\frac{1}{\beta})}\right)$ then the error $|v(u^*) - \hat{v}(u^*)|$ is small compared to $v(u^*)$. This means that u^* is not deleted from the list L in Step 8.

Overall, with probability at least $\frac{n^{-a}}{2} - 7\beta$ we have that the output set Y (from Step 9) contains at least one vector $\hat{y}_{u^*}$ s.t. the ball of radius $3cr$ around $\hat{y}$ contains at least one point from P. ■

4. The constraint on t in Theorem 2.2 depends logarithmically on the number of possible bins. In our case, there are $4\Lambda/p \leq \Lambda\sqrt{d}/r \leq \text{poly}(n\sqrt{d})$ possible bins, where the last inequality is because we assumed that $\Lambda/r \leq \text{poly}(n)$.

A.3 Algorithm GoodCenters

We are now ready to present algorithm GoodCenters, and to prove Theorem 3.1 (restated here as Theorem A.7). As we mentioned, algorithm GoodCenters is obtained by applying algorithm CentersProcedure multiple times to boost its success probability.

Algorithm GoodCenters

Input: Radius r, target number of points t, failure probability β, privacy parameters ε, δ.

Optional input: Parameter t. Otherwise set $t = O\left(\frac{1}{\varepsilon} \cdot n^{0.5+a+b} \cdot \sqrt{d} \cdot \log(\frac{1}{\beta}) \log\left(\frac{dn}{\beta\delta}\right)\right)$.

Setting: Each player $j \in [n]$ holds a value $x_j \in \mathcal{B}(0, \Lambda)$. Define $S = (x_1, \ldots, x_n)$.

1. Denote $M = 4n^a \ln(\frac{1}{\beta})$ and randomly partition $[n]$ into M subsets $I_1, \ldots, I_m \subseteq [n]$. For $m \in [M]$ define $S_m = \{x_i \in S : i \in I_m\}$.
2. For $m \in [M]$ apply algorithm CentersProcedure on S_m with the following parameters: radius r, failure probability $\hat{\beta} = \frac{\beta}{7M}$, privacy parameters ε, δ, and target number of points $\hat{t} = \frac{t}{2M}$. For every $m \in [M]$ denote the outcomes as Y_m, L_m, and h_m.
3. Return the sets of centers $Y_1, \ldots, Y_M$, the lists $L_1, \ldots, L_M$, the hash functions $h_1, \ldots, h_M$, and the partition $I_1, \ldots, I_M$.

Theorem A.7 *Let $\beta, \varepsilon, \delta, n, d, \Lambda, r$ be such that $t \geq O\left(\frac{n^{0.5+a+b} \cdot \sqrt{d}}{\varepsilon} \log(\frac{1}{\beta}) \log\left(\frac{dn}{\beta\delta}\right)\right)$ and such that $\Lambda/r \leq \text{poly}(n)$. Let $S = (x_1, \ldots, x_n)$ be a distributed database where every x_i is a point in the d-dimensional ball $\mathcal{B}(0, \Lambda)$, and let* GoodCenters *be executed on S with parameters $r, t, \beta, \varepsilon, \delta$. Denote $M = 4n^a \ln(\frac{1}{\beta})$. The algorithm outputs a partition $I_1, \ldots, I_M \subseteq [n]$, lists $L_1, \ldots, L_M$, hash functions $h_1, \ldots, h_M$, and sets of centers $Y_1, \ldots, Y_M$, where for every $m \in [M]$ and $u \in L_m$, the set Y_m contains a center $\hat{y}_{m,u}$. The following holds.*

1. *With probability at least $1 - \beta$, for every $m \in [M]$ and $u \in L_m$ we have $|\{i \in I_m : h_m(x_i) = u \text{ and } \|x_i - \hat{y}_{m,u}\| \leq 5cr\}| \geq \frac{t}{16M} \cdot n^{-b}$.*
2. *Denote $Y = \bigcup_{i \in [M]} Y_m$. Then $|Y| \leq \frac{512 \cdot n^{1+a+b}}{t} \ln(\frac{1}{\beta})$.*
3. *Let $P \subseteq S$ be a set of t points which can be enclosed in a ball of radius r. With probability at least $1 - \beta$ there exists $\hat{y} \in Y$ such that the ball of radius $5cr$ around $\hat{y}$ contains all of P.*

Proof Items 1 and 2 of the lemma follow directly from the properties of algorithm CentersProcedure. We now proceed with the analysis of item 3. Let $P \subseteq S$ be s.t. $|P| = t$ and $\text{diam}(P) \leq r$, and consider the following good event, which happens with probability at least $1 - \beta$ by the Chernoff bound (assuming that $t \geq 24M \ln(\frac{eM}{\beta})$).

Event E_1 (over partitioning S into $S_1, \ldots, S_M$):

(a) For every $m \in [M]$ we have $|P \cap S_m| \geq \frac{t}{2M}$.

(b) For every $m \in [M]$ we have $\frac{n}{2M} \leq |I_m| \leq \frac{2n}{M}$.

We proceed with the analysis assuming that Event E_1 occurred. Let us say that the mth execution of `CentersProcedure` *succeeds* if $\exists y_m \in Y_m$ such that the ball of radius $3cr$ around y_m contains at least one point from $P \cap S_m$. Recall that we assume that $t \geq O\left(\frac{n^{0.5+a+b}\cdot\sqrt{d}}{\varepsilon}\log(\frac{1}{\beta})\log(\frac{dn}{\beta\delta})\right)$. Hence, by the properties of algorithm `CentersProcedure`, every single execution succeeds with probability at least $n^{-a}/4$. As the different executions of `CentersProcedure` are independent, when $M \geq 4n^a \ln(\frac{1}{\beta})$, the probability that at least one execution succeeds is at least $1-\beta$. In this case, there is a point $y \in Y = Y_1 \cup \cdots \cup Y_m$ such that the ball of radius $3cr$ around it contains at least one point from P, and hence, the ball of radius $5cr$ around y contains *all* of P. ■

References

Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. In *FOCS*, 2017.

Maria-Florina Balcan, Travis Dick, Yingyu Liang, Wenlong Mou, and Hongyang Zhang. Differentially private clustering in high-dimensional Euclidean spaces. In *ICML*, 2017.

Raef Bassily and Adam D. Smith. Local, private, efficient protocols for succinct histograms. In *STOC*, 2015.

Raef Bassily, Kobbi Nissim, Uri Stemmer, and Abhradeep Thakurta. Practical locally private heavy hitters. In *NIPS*, 2017.

Amos Beimel, Kobbi Nissim, and Eran Omri. Distributed private data analysis: Simultaneously solving how and what. In *CRYPTO*, 2008.

Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: The SuLQ framework. In *PODS*, 2005.

Mark Bun, Jelani Nelson, and Uri Stemmer. Heavy hitters and the structure of local privacy. In *PODS*, 2018.

Alisa Chang, Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. Locally private k-means in one round. *CoRR*, abs/2104.09734, 2021.

Edith Cohen, Haim Kaplan, Yishay Mansour, Uri Stemmer, and Eliad Tsfadia. Differentially-private clustering of easy instances. In *ICML*, 2021.

Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, 2006a.

Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006b.

Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *FOCS*, 2010.

Dan Feldman, Amos Fiat, Haim Kaplan, and Kobbi Nissim. Private coresets. In *STOC*, 2009.

Dan Feldman, Chongyuan Xiang, Ruihao Zhu, and Daniela Rus. Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks. In *IPSN*, 2017.

Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. Differentially private clustering: Tight approximation ratios. In *NeurIPS*, 2020.

Anupam Gupta, Katrina Ligett, Frank McSherry, Aaron Roth, and Kunal Talwar. Differentially private combinatorial optimization. In *SODA*, 2010.

John A Hartigan. *Clustering algorithms*. John Wiley & Sons, Inc., 1975.

Justin Hsu, Sanjeev Khanna, and Aaron Roth. Distributed private heavy hitters. In *ICALP*, 2012.

Zhiyi Huang and Jinyan Liu. Optimal differentially private algorithms for k-means clustering. In *PODS*, 2018.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, 1998.

Matthew Joseph, Jieming Mao, Seth Neel, and Aaron Roth. The role of interactivity in local differential privacy. In *FOCS*, 2019.

Haim Kaplan and Uri Stemmer. Differentially private k-means with constant multiplicative error. In *NeurIPS*, 2018.

Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, 2011.

Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. *Commun. ACM*, 53(9):89–97, 2010.

Prashanth Mohan, Abhradeep Thakurta, Elaine Shi, Dawn Song, and David Culler. Gupt: Privacy preserving data analysis made easy. In *SIGMOD*, 2012.

Kobbi Nissim and Uri Stemmer. Clustering algorithms for the centralized and local models. In *ALT*, 2018.

Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, 2007.

Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Locating a small cluster privately. In *PODS*, 2016.

Richard Nock, Raphaël Canyasse, Roksana Boreli, and Frank Nielsen. k-variates++: more pluses in the k-means++. In *ICML*, 2016.

Moshe Shechner, Or Sheffet, and Uri Stemmer. Private k-means clustering with stability assumptions. In *AISTATS*, 2020.

Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. Differentially private k-means clustering. In *CODASPY*, 2016.

Salil Vadhan. *The Complexity of Differential Privacy*. 2016.

Professor Vazirani and Professor Rao. Lecture notes in combinatorial algorithms and data structures. URL `http://www.cs.berkeley.edu/~satishr/cs270/sp11/rough-notes/measure-concentration.pdf`.

Yining Wang, Yu-Xiang Wang, and Aarti Singh. Differentially private subspace clustering. In *NIPS*, 2015.