

Locally Private k -Means Clustering with Constant Multiplicative Approximation and Near-Optimal Additive Error

Anamay Chaturvedi, Matthew Jones, Huy Lê Nguyẽn

Khoury College of Computer Sciences, Northeastern University
 chaturvedi.a@northeastern.edu, jones.m@northeastern.edu, nguyen.hu@northeastern.edu

Abstract

Given a data set of size n in d' -dimensional Euclidean space, the k -means problem asks for a set of k points (called centers) such that the sum of the ℓ_2^2 -distances between the data points and the set of centers is minimized. Previous work on this problem in the local differential privacy setting shows how to achieve multiplicative approximation factors arbitrarily close to optimal, but suffers high additive error. The additive error has also been seen to be an issue in implementations of differentially private k -means clustering algorithms in both the central and local setting (Balcan et. al. (2017), Chaturvedi et. al. (2021), Chang et. al. (2021)). In this work we introduce a new locally private k -means clustering algorithm that achieves near-optimal additive error whilst retaining constant multiplicative approximation factors and round complexity. Concretely, given any $c > \sqrt{2}$, our algorithm achieves $O(k^{1+\tilde{O}(1/(2c^2-1))}\sqrt{d'n} \log d' \text{ poly log } n)$ additive error with an $O(c^2)$ multiplicative approximation factor.

1 Introduction

Given a set D' of n points in a d' -dimensional ball with unit diameter in Euclidean space, the k -means clustering problem asks for a set of k points S such that the sum of ℓ_2^2 -distances from each data point to the closest respective point in S , which we denote $f_{D'}(S)$, is minimized. Although k -means clustering in the non-private setting is well-studied, over the past few years there have been several developments in the differentially private (DP) setting. Differential privacy (Dwork et al. 2006) provides a framework to characterize the loss in privacy which occurs when sensitive data is processed and the output of this computation is revealed publicly. Although there are different ways to define and capture this loss in privacy, broadly speaking these characterizations tend to be either *central* or *local* in nature. The definition of central DP is formalized as follows:

Definition 1.1 (Differential privacy (DP), (Dwork et al. 2006)). Two datasets $D_1, D_2 \in \mathcal{X}^n$ are *neighbouring* if they differ in at most one record. An algorithm $A : \mathcal{X}^n \rightarrow \mathcal{Y}$ is said to be ϵ -*differentially private* (DP) if for any $S \subset \mathcal{Y}$ and any two neighbouring datasets $D_1, D_2 \in \mathcal{X}^n$,

$$P(A(D_1) \in S) \leq \exp(\epsilon) P(A(D_2) \in S).$$

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Informally, differential privacy asks for a guarantee that the likelihood of any possible output does not change too much by adding to or dropping from our data set any single value from the data universe. In practice, this guarantee is fulfilled by adding carefully calibrated noise to quantities that are information-theoretically sensitive to the private data, and the goal is to achieve relatively low error under these privacy constraints. Perfect answers to a problem typically violate privacy; as a consequence, the constraints of privacy usually enforce harsher lower bounds on accuracy or utility than those imposed by the limits of time or sample efficient computation.

In local differential privacy (LDP) the constraints are even more severe; the entity solving the algorithmic problem only gets access to noisy, privatized data. Formally,

Definition 1.2 (Local differential privacy (LDP), (Kasiviswanathan et al. 2011)). Consider a protocol which queries some function of a distributed private data set in some r rounds, and let the response of the queries on the private datum p be $A(p) = (A_1(p), \dots, A_r(p))$, where $A_i(p)$ is the response in the i th round. We say that this protocol is ϵ -*locally differentially private* (LDP) if the algorithm that outputs privatized responses for any agent $p \mapsto A(p)$ is ϵ -DP.

This constraint forces strong lower bounds for locally private protocols; a lower bound of $\Omega((k + \sqrt{n})/\epsilon)$ is known for the additive error of any ϵ -LDP interactive constant-factor multiplicative approximation algorithm for the k -means clustering problem that operates in a constant number of rounds (Stemmer 2020; Nguyen, Chaturvedi, and Xu 2021). Despite the lower bounds known for this and other problems, local differential privacy is often utilized in practice (Erlingsson, Pihur, and Korolova 2014; Thakurta et al. 2017). Making progress for LDP k -means clustering, a fundamental subroutine for many big-data applications is hence of real-world interest as well.

Recent work on LDP k -means It was observed by Feldman et al. that there is a general algorithm that solves the private k -means problem given access to a private solution for the 1-cluster problem. Nissim and Stemmer gave a solution for the private 1-cluster problem and consequently the first LDP algorithm for the k -means problem with provable guarantees, achieving a multiplicative approximation of $O(k)$ and an additive error term of $\tilde{O}(n^{2/3+a} \cdot d^{1/3} \cdot \sqrt{k})$. The ex-

ponent of n in the additive error holds for arbitrarily small α at the cost of looser multiplicative approximation guarantees, a trade-off which appears in most later work as well. This artefact is the consequence of using *locality sensitive hashing* (LSH) families to detect accumulation of data. Stemmer and Kaplan gave the first constant factor multiplicative approximation algorithm for this problem within an additive error of $\tilde{O}(n^{2/3+\alpha} \cdot d^{1/3} \cdot k^2)$. They refine the approach of the previous work by specifically targeting the k -means problem instead of the 1-cluster problem but they also use LSH functions. The additive error was further brought down by Stemmer to $\tilde{O}(n^{1/2+\alpha} \cdot k \cdot \max\{\sqrt{d}, \sqrt{k}\})$, in which work a lower bound of $\Omega(\sqrt{n})$ was also proved, as mentioned before. Most recently Chang et al. introduced a one-round protocol for LDP k -means in the $(\epsilon, 0)$ setting. They get a multiplicative approximation of $\eta(1 + \alpha)$ where η is the multiplicative approximation guarantee of any given non-private k -means algorithm and an additive error term of $k^{O_\alpha(1)} \cdot \sqrt{nd'} \cdot \text{poly log}(n)/\epsilon$. We see that the trade-off between the additive and multiplicative approximations in this algorithm has been shifted from n to k . However, the dependence of the $O_\alpha(1)$ exponent of k on α is at least $\Omega(1/\alpha)^2$ and can make the additive error prohibitively large depending on the regime of interest.

In the non-private setting the performance of k -means clustering algorithms is usually not very sensitive to the multiplicative approximation guarantee, unless the data set is chosen in a pathological fashion. On the other hand, the additive error always presents as stated due to the artificial error introduced by the algorithm to protect privacy. Experimental work (Balcan et al. 2017; Nguyen, Chaturvedi, and Xu 2021; Chang et al. 2021) on k -means clustering in the related central model of DP shows that the performance of private clustering algorithms seems to be far more sensitive to the additive error, which is bound to exist owing to the lower bound mentioned before. These observations lead to the following question:

Does there exist an LDP k -means clustering algorithm with constant multiplicative approximation such that the additive error scales well in n and k simultaneously?

Technical contributions: In this work we derive an algorithm for private LDP k -means where we return to a LSH-based approach but go beyond the previous line of work by moving the trade-off in the additive error from the exponent of n to k (as in Chang et al.). However, motivated by the empirical evidence, our goal is to reduce the additive error to near-optimal at the cost of looser multiplicative approximation instead of the other way around; we succeed in this goal by driving down the exponent of k to $1 + O(1/(2c^2 - 1))$, where c is an LSH parameter that can be set to any value $\geq \sqrt{2}$. This shows for the first time that it is possible to have constant factor multiplicative approximation k -means clustering algorithms in the LDP setting with additive error that has a truly square-root dependence on the data set size and the ambient dimension and arbitrarily close to linear dependence on the number of cluster centers. Formally, our main result is:

Theorem 1.3. *For $\epsilon < 1$, Low Additive Error LDP k -Means (LAE) is an ϵ -locally differentially private algo-*

Table 1: Comparison with recent LDP algorithms for k -means

Work	Mult. App.	Add. Error
2018	$O(k)$	$\tilde{O}(n^{2/3+\alpha} \cdot d^{1/3} \cdot \sqrt{k})$
2018	$O(1)$	$\tilde{O}(n^{2/3+\alpha} \cdot d^{1/3} \cdot k^2)$
2020	$O(1)$	$\tilde{O}(n^{1/2+\alpha} \cdot k \cdot \max\{\sqrt{d'}, \sqrt{k}\})$
2021	$(1 + \alpha)\eta$	$\tilde{O}(n^{1/2} \cdot d'^{1/2} \cdot k^{\tilde{O}(1/\alpha^2)})$
LMA	$(1 + \alpha)\eta$	$\tilde{O}(n^{1/2} \cdot d'^{1/2} \cdot k^{\tilde{O}(1/\alpha^2)})$
LAE	$O(c^2)$	$\tilde{O}(n^{1/2} \cdot d'^{1/2} \cdot k^{1+O(1/(2c^2-1))})$

LMA and LAE are introduced in this paper and LAE is our main contribution. The data domain is a unit diameter ball.

The \tilde{O} notation suppresses privacy parameters and logarithmic terms, where the additive errors all have a $1/\epsilon$ factor. η is the best approximation factor of a non-private algorithm. The value c can be set to any real number greater than $\sqrt{2}$. From top to bottom, the previous works appearing in this table are (Nissim and Stemmer 2018; Stemmer and Kaplan 2018; Stemmer 2020; Chang et al. 2021).

rithm for the k -means clustering problem that after four rounds of interaction outputs a set S' of size k such that with constant probability the clustering cost $f_{D'}(S')$ equals $O(c^2 \text{OPT}') + O\left(\frac{\sqrt{d'}n \log d'}{\epsilon}\right)(k \text{poly log } n)^{1+O\left(\frac{1}{2c^2-1}\right)}$.

We also introduce a second algorithm that achieves a similar cost guarantee to that of (Chang et al. 2021). This is of theoretical interest as a natural stopping point of previous methods but being peripheral to our main improvement over previous work we describe it only briefly in this paper and relegate most of the details to the supplementary.

Theorem 1.4. *For $\epsilon < 1$, Low Multiplicative Approximation LDP k -Means (LMA) is an ϵ -locally differentially private algorithm for the k -means clustering problem that after one round of interaction outputs a set S' of size k such that that the clustering cost $f_{D'}(S')$ equals $(1 + O(\alpha))\eta \text{OPT}' + \frac{1}{\epsilon}k^{\tilde{O}(1/\alpha^2)}\sqrt{d'}n \log d' \text{poly log } n$, where η is the best approximation factor of a non-private algorithm.*

Although we will sketch the main ideas of and provide complete pseudocode for LAE, and give a high level overview of the ideas behind LMA, due to space constraints we refer the reader to the supplementary for all complete proofs and any missing details.

Outline of private clustering algorithms: Many works in locally (and centrally) private clustering proceed via the following sequence of steps.

Step 1: We reduce dimensions of the d' -dimensional data set D' via the Johnson Lindenstrauss (JL) transform to d dimensions and get a lower-dimensional proxy data set D . It is a well-known fact that with as few as $d = O(\log n)$ dimensions we can preserve the pair-wise squared ℓ_2 distances between the points of the data set and consequently show that the k -means clustering cost function of D (as well as the optimal cost OPT) is close to that of D' (and OPT' , respectively). Reducing dimensions and approximately preserving the diameter of the domain reduces the sensitivity of

the queries made to compute the cluster sets and allows us to add less noise in the main course of the computation. We now work entirely with D in the dimension reduced space $B(0, 1) \subset \mathbb{R}^d$ until we have privately identified cluster sets and are ready to estimate the cluster centers in the original space.

Step 2: A large number of possible candidate k -means centers S (more than the k permitted in the final solution) are privately generated. We permit the clustering cost of D with respect to S be a multiplicative approximation to the optimal clustering cost; such a candidate set is called a *bicriteria solution* for the k -means problem as we have relaxed two constraints of the original problem.

Step 3: We exploit the idea that if the clustering cost with respect to this set of candidate centers S is close to optimal, then we can construct a proxy data set D^* using S whose k -means clustering function is close to that of the original data set D . At a high level the idea is to simply move each data point to the closest center in the set S ; the sum of the ℓ_2^2 distances moved over all points is about the optimal clustering cost by the choice of S and so essentially by applications of the triangle inequality the values of the k -means clustering functions of D and D^* are at most $O(\text{OPT})$ apart when evaluated on any candidate k -means solution. Since D^* is privately generated, we can directly apply the non-private k -means clustering function of our choice and generate k centers in the low dimensional space which work well for D . Now we appeal to LDP private averaging to privately recover the mean of each cluster in the original space and we are done.

It was observed by Stemmer that one of the main road-blocks in computing solutions with low additive error is figuring out how to generate a relatively small bi-criteria S solution to the k -means problem in the second step. It was essentially shown in (Stemmer 2020) that a set S of candidate centers with respect to which the clustering cost is at most $\tilde{O}(k^p \sqrt{n} \log^q n)$ will lead to additive error $\tilde{O}(\max\{\sqrt{|S|n}, k^p \sqrt{n} \log^q n\})$ down the line (omitting the dependence on dimension). In order to avoid an exponent of $1/2 + a$ on n as in the previous works which generate S by detecting data accumulation via LSH functions, it is necessary to find a bi-criteria solution with $O(\text{poly } k \text{ poly log } n)$ candidate centers with $O(\text{poly } k \sqrt{n} \text{ poly log } n)$ additive error. Both our algorithms achieve their improvements by generating such a small size bi-criteria solution for the k -means problem.

LDP k -means with arbitrarily tight multiplicative approximation: We briefly describe how we get near optimal multiplicative approximation in LMA as a warm-up to the discussion for LAE. In the outline for private clustering described above, one natural way to generate candidate centers set S privately in the lower dimension space is to start with a d -dimensional grid-based discretization of the unit ball and compute private scores for each grid point based on how many points lie close to it; we then pick some of the most highly ranked points to construct S . We appeal to recent advances in dimension reduction for k -means clustering (Makarychev, Makarychev, and Razenshteyn 2019) which

show that Johnson-Lindenstrauss (JL) style dimension reduction to $\tilde{O}(\log k/\alpha^2)$ many dimensions preserves the k -means cost of every clustering of a data set within a multiplicative approximation of $(1 \pm \alpha)$; moving from $O(\log n)$ to $O(\log k)$ dimensions allows us to move the trade-off from the exponent of n to the exponent of k in the additive error.

At a high level, this approach can be analyzed as follows. Suppose we fix some optimal k means solution S_{OPT} and decompose the domain in concentric shells depending on the distance from some fixed k cluster centers. By setting geometric thresholds of $1, 1/2, 1/4$, and so on, the l th shell $B(S_{\text{OPT}}, 1/2^{l-1}) - B(S_{\text{OPT}}, 1/2^l)$ has the property that every data point in it has an optimal clustering cost of $O(1/(2^l)^2)$ units. To cluster the data points that occur in the l th shell, if we consider a grid with side-length $\alpha/(2^l \sqrt{d})$ then we have the guarantee that the closest grid point for every data point is at a distance α times its optimal clustering distance. In principle the entire dataset D could lie in the l th shell, but we are able to show that picking $k^{\tilde{O}(1/\alpha^2)}$ poly log n centers from this grid suffices to ensure that most points in the l th shell are within an $O(\alpha/(2^l)^2)$ distance of some candidate center. As the grid scales vary geometrically it turns out only $\log n$ grids are needed in all and the size of the set of candidate centers generated is $k^{\tilde{O}(1/\alpha^2)}$ poly log n . In sum, this allows us to show that the net movement of points from the data set to the bi-criteria solution is in fact $O(\alpha \text{OPT})$, allowing us to achieve a $1+O(\alpha)$ inflation in the multiplicative approximation as opposed to an $O(1)$ inflation.

Challenges in reducing the additive error: Many previous works with tight bounds on the exponents of n and k in the additive error proceed by using LSH functions to discretize the response. We recall that LSH functions with parameters (r, cr, p, q) with $c > 1$ and $p > q$ are hash functions with the property that for any pair of points in the input domain within a distance of r units, the probability of colliding is at least p , and for any pair of points that are at least a distance of cr units apart the probability of colliding is at most q . At a high level the idea behind using LSH functions for clustering is to allocate candidate cluster centers by computing point averages of heavy hash buckets for LSH functions at geometrically varying scales (similar to how we use geometrically varying grid unit lengths for tight multiplicative approximation. Just as allocating points from the l th grid worked well to cluster the l th shell, using LSH functions with scale $r = 1/2^l$ allows us to capture points which lie in the l th shell with respect to any fixed k -means solution.

The reason for the trade-off between the multiplicative approximation and the additive error in these algorithms is that the (near-optimal) constructions of LSH functions have a trade-off between c , the ratio of the near and far thresholds, and the ratio between the collision probabilities p and q which determines how many false positives we have to deal with when estimating bucket averages. We will see how to surpass these challenges by using LSH functions in a novel combination with a dyadic tree-based approach following Braverman et al..

2 Preliminaries

We recall here some of the main technical tools which we appeal to in the construction of our main algorithm.

Theorem 2.1 ((Duchi, Jordan, and Wainwright 2013)). *There is an ϵ -LDP mechanism for $\epsilon \leq 1$ to privately release a vector v such that if Z is the value returned then $\mathbb{E}[Z] = v$. Further, $\|Z\|_2 \leq cr\sqrt{d}/\epsilon$ for some universal constant $c < \infty$.*

Since the output of the algorithm given by Theorem 2.1 is a vector of bounded length, we can apply standard concentration bounds to get an ϵ -LDP mechanism for locally privately computing the mean of n independent private d -dimensional vectors.

Corollary 2.2 (LDP mean estimation). *For private vectors $v_1, \dots, v_n \in \mathbb{R}^d$ and their respective privatized releases Z_1, \dots, Z_n , we have that with probability $1 - \beta$, $\|\sum_{i=1}^n Z_i - \sum_{i=1}^n v_i\| = O\left(\frac{r}{\epsilon}\sqrt{dn} \log \frac{d}{\beta}\right)$.*

For more details about privately releasing and averaging vectors, please see the supplementary material. We now formally describe LSH functions which are the core technical tool of our main algorithm:

Definition 2.3 (Locality sensitive hashing (LSH)). We say that a family of hash functions $H : \mathbb{R}^d \rightarrow B$ for a finite set of buckets B is *locality-sensitive* with parameters (p, q, r, cr) if for every $x, y \in \mathbb{R}^d$ for some $1 \geq p > q \geq 0$, $r > 0$ and $c > 1$

$$P(H(x) = H(y)) \begin{cases} \geq p \text{ if } d(x, y) \leq r \\ \leq q \text{ if } d(x, y) \geq cr. \end{cases}$$

In this work we use a near-optimal LSH family construction from (Andoni and Indyk 2006).

Theorem 2.4. *For every sufficiently large d and n there exists a family \mathcal{H} of hash functions defined on \mathbb{R}^d such that for a dataset of size n ,*

1. *A function from this family can be sampled, stored and computed in time $t^{O(t)} \log n + O(dt)$, where t is a free positive parameter of our choosing.*
2. *The collision probability for two points $u, v \in \mathbb{R}^d$ depends only on the ℓ_2 distance between them, which we henceforth denote by $p(\|u - v\|)$.*
3. *The following inequalities hold:*

$$p(1) \geq \frac{A}{2\sqrt{t}} \frac{1}{(1 + \epsilon + 8\epsilon^2)^{t/2}}$$

$$\forall c > 1, p(c) \leq \frac{2}{(1 + c^2\epsilon)^{t/2}}$$

where A is an absolute constant < 1 , and $\epsilon = \Theta(t^{-1/2})$. One can choose $\epsilon = \frac{1}{4\sqrt{t}}$.

4. *The number of buckets N_B an LSH function with parameter t uses is $t^{O(t)} \log n$.*

In the locally private setting, to ϵ -privately release a point directly requires adding a noise vector with length proportional to $1/\epsilon$ to their private data, making it impossible to privately derive fine-grained information. To deal with this,

we will estimate the data set distribution indirectly. One way of accomplishing this is to *discretize* the agents' response. Although the privatized individual responses are highly noisy, the finite range of values on a discretized response allows the slight bias towards values which are *heavy-hitters* to cause their counts to accumulate and be distinguishable from the counts of false positives. We will appeal to prior work on locally private succinct histogram recovery to recover such heavy-hitting values with minimal loss in privacy.

Lemma 2.5 (Algorithm Bitstogram, (Bassily et al. 2020)). *Let V be a finite domain of values, let $f : D' \rightarrow V$, and let $n(v)$ denote the frequency with which v occurs in $f(D')$. Let $\epsilon \leq 1$. Algorithm Bitstogram(f, ϵ, β) interacts with the set of n users in 1 round and satisfies ϵ -LDP. Further, it returns a list $L = ((v_i, a_i))_i$ of value-frequency pairs (i.e. elements of $V \times \mathbb{R}$) with length $\tilde{O}(\sqrt{n})$ such that with probability $1 - \beta$ the following statements hold:*

1. *For every $(v, a) \in L$, $\|a - f(v)\| \leq E$ where $E = O\left(\frac{1}{\epsilon}\sqrt{n \log(n/\beta)}\right)$.*
2. *For every $v \in V$ such that $f(v) \geq M$, $v \in L$, where $M = O\left(\frac{1}{\epsilon}\sqrt{n \log |V|/\beta \log(1/\beta)}\right)$.*

We overload notation to treat the list returned by Bitstogram returns as either a set of (heavy-hitter, frequency) pairs or a function which may be queried on a value to return either the corresponding frequency if it is a heavy hitter or a value of 0 otherwise. A subscript of M denotes the upper bound on the maximum frequency omitted. We see that whenever $|V| = \Omega(n)$, we have that $M = \Omega(E)$ and Bitstogram promises a uniform error bound of M when estimating the frequency of any element in the co-domain for an appropriate choice of constants.

We introduce an extension of the Bitstogram algorithm called HeavySumsOracle that allows us to query the sums of some vector valued function over the set of elements that map to a given heavy-hitter value. For a given value-mapping function $f : \mathcal{X} \rightarrow \mathcal{V}$ and a vector-valued function $g : \mathcal{X} \rightarrow \mathbb{R}^d$ the sum estimation oracle privately returns for every heavy hitter $v \in \mathcal{V}$ the sum of all agents that map to v , i.e. $\sum_{p: f(p)=v} p$. We recall that Bitstogram is a modular algorithm with two subroutines; a frequency oracle that privately estimates the frequency of any value in the data universe, and a succinct histogram construction that constructs the heavy hitters in a bit-wise manner by making relatively few calls to the frequency oracle. The construction of HeavySumsOracle essentially mimics the frequency oracle construction called Hashtogram from (Bassily et al. 2020) and can be run in parallel with Bitstogram, allowing us to reduce the round complexity of our protocols. The pseudo-code and proof of lemma 2.6 may be found in the supplementary.

Lemma 2.6 (HeavySumsOracle). *Let $f : \mathcal{X} \rightarrow \mathcal{V}$, $g : \mathcal{X} \rightarrow B(0, \Delta/2) \subset \mathbb{R}^d$ and $\Delta > 0$ be some publicly known functions and parameters, and let $D' \subset \mathcal{X}$ be a distributed dataset over n users. Let the private parameter $\epsilon \leq 1$. With probability at least $1 - \beta$, for every $v \in \mathcal{V}$ that occurs in $f(D')$, if $S(v)$ is the value returned by the HeavySumsOracle*

then

$$\left\| S(v) - \sum_{f(y)=v} g(y) \right\| \leq O\left(\frac{\Delta}{\epsilon} \sqrt{d'n} \log \frac{d'}{\beta}\right)$$

Further, HeavySumsOracle is ϵ -LDP.

3 Low Additive Error LDP k -Means

As described in the outline of private clustering algorithms and the following description, the main challenge that we resolve is in the second step of the private clustering outline, where we identify a bicriteria relaxation that has $O(\text{OPT})$ clustering cost with small additive error and only $O(k^{1+O(1/(2c^2-1))})$ many candidate centers.

Instead of applying LSH functions directly on the whole domain, we derive a more accurate measure of the data distribution by first appealing to the following construction from (Braverman et al. 2017). In the domain $[0, 1]^d$, a dyadic 2^d -ary tree of cells is constructed, where each rectangular cell is recursively sub-divided into 2^d child cells by bisecting the cell along each axis. The cell at the top of the hierarchy, with side-length one unit is the whole domain, and the side length of each level l cell is $t_l = 2^{-l}$ units. $L = \log n$ levels of the grid suffice to discretize the domain to a sufficiently fine degree. This construction follows the dimension reduction step ensuring $d = O(\log n)$. The authors observe that if we *randomly shift* this hierarchy of cells, then in expectation there are $O(1)$ many cells with side-length t_l within an ℓ_2 distance of t_l/d units of any fixed point; we use this observation multiple times in the sequel.

Guessing the optimal cost: Suppose we knew the optimal cost OPT , and let S_{OPT} be some arbitrary k -means solution. Data points in cells further than t_l/d units away from S_{OPT} must have a clustering cost of at least t_l^2/d^2 , but the sum of their costs cannot exceed the total cost OPT . There are hence at most $\text{OPT} d^2/t_l^2$ many such points. Tracing a similar argument with cells, we derive a threshold $T_l \simeq \text{OPT}/(t_l^2 k \log n)$ (dropping some terms) such that there cannot be more than $O(k \log n)$ many cells that have more than T_l points further than t_l/d units from S_{OPT} (across all L levels). By the random shift observation there are unconditionally at most $O(1) \cdot |S_{\text{OPT}}| \cdot L = O(k \log n)$ many cells closer than t_l/d to S_{OPT} so we get that regardless of where they lie in the domain there are at most $O(k \log n)$ *heavy* cells in any level, i.e. cells that beat the threshold T_l for their level. Any cell which is not heavy is called *light*.

Assuming OPT is known, this allows us to identify regions of the domain at different scales where data accumulates beyond these thresholds. We guess values for OPT varying in factors of 2 from $k\sqrt{n}$ (the targeted additive error) to n (trivially achieved clustering cost) and allocate candidate centers for all guesses. To allocate candidate centers for any level we use LSH functions as discussed in the introduction, but we only invite responses from points in *medium* cells, the light children of heavy cells.

Allocating candidate centers: LSH functions with parameters (p, q, r, cr) promise that any two points in the domain

within a distance of r units collide with probability at least p and points further than cr units with collide with probability at most q where $q < p$. The key idea is that any optimal cluster with radius r will end up populating one of the LSH functions buckets, and that the average of all points hashing to this bucket (contaminated with a few points from outside the cluster) should lie within a distance of at most $O(cr)$ units from the cluster. However, doing this naively would lead to a similar bound with a $n^{1/2+a}$ term as previous work; we now discuss some of the ideas behind our improvement.

The $n^{1/2+a}$ barrier: Fixing a point x in an optimal cluster C and a (p, q, r, cr) -LSH function, one can show that with probability $p/4$ the distance between x and the average over all points colliding with x is at most $cr \cdot |\{\text{points from } C \text{ colliding with } x\}| + \Delta \cdot |\{\text{points further than } cr \text{ units from } x\}|$, where Δ is the diameter of the domain of an LSH function. Rearranging terms one can show that in order for this to be at most $O(cr)$ (the desired distance for candidate center allocation), one needs the ratio of collision probabilities p/q to beat the product of Δ/r and $|D_l|/|C|$, where D_l is all points lying in the domain of the LSH function applied on level l medium cells. Tuning p/q to beat a term B causes p and consequently the success probability to scale with $B^{-1/\Theta(c)}$. In previous work the term that p/q had to beat scaled as a polynomial in n ; driving up the success probability by running $1/p$ many independent copies of this scheme is what lead prior work to incur a small n^a factor in the number of candidate centers allocated. The decomposition of the data set across different levels with decreasing side-lengths and increasing thresholds allows us to bound both Δ/r and $|D_l|/|C|$ by $k \text{ poly log } n$, which is how we manage to avoid tuning the collision probabilities too aggressively and avoid the n^a factor as before, substituting it instead with a $k^{1/O(2c^2-1)} \text{ poly log } n$ factor.

Reducing the exponent of k : If we apply LSH functions on heavy cells in a cell-wise manner, we must account for the fact that optimal clusters can be partitioned arbitrarily by the cells in each level leading to many *cluster sections*. One can have all k clusters intersecting with the $O(k \log n)$ heavy cells in any one level, which would require an allocation of $\tilde{O}(k^2)$ many centers to serve the $\tilde{O}(k^2)$ many cluster sections. In order to reduce the exponent of k in the number of cluster centers allocated, we make three algorithmic choices.

Firstly, we allocate a candidate center at the center of every heavy cell (i.e. at most $O(k \log n) \cdot L = O(k \log^2 n)$ more candidate centers). This guarantees that every point in the heavy cells in level l has a candidate center at a distance of $t_l \sqrt{d}$. Secondly, we go up $1.5 \lg d$ levels and apply LSH functions to the ancestors of the level $l+1$ medium cells which have side-length $d^{3/2} t_l$. The consequence of these two modifications is that we only need to allocate cluster centers within a distance of $2^{-l} \sqrt{d}$ units of any point of D_l , and that since there are only $O(1)$ many cells with side-length $d^{3/2} 2^{-l}$ within a distance of $2^{-l} \sqrt{d}$ units of an optimal center (again by the random shift observation), there are only $O(k)$ many cluster sections we must account for.

Thirdly, in order to avoid dealing with the worst case $O(k)$

```

/* Step 1 - Initialization and
   first interaction */
```

 $\gamma \leftarrow$ uniformly random vector in $[-1/2, 1/2]^d$
 $T : \mathbb{R}^{d'} \rightarrow \mathbb{R}^d$ dimension reduction for
 $d = O(\log(k/\alpha\beta)/\alpha^2)$, $S : \mathbb{R}^d \rightarrow \mathbb{R}^d$ scaling by a
factor $\frac{1}{2(1+\alpha)}$, $P : \mathbb{R}^d \rightarrow B(0, 1/2)$ projection to
 $B(0, 1/2)$ followed by translation by γ
 $Q = P \circ S \circ T : \mathbb{R}^{d'} \rightarrow B(0, 1) \subset \mathbb{R}^d$
 $L = \lg n$
Do in parallel:
 $\text{CH}^l \leftarrow \text{Bitstogram}(\mathcal{C}_l \circ Q(\cdot), \epsilon_{\text{CH}}, \beta/L)$;
 /* Cell-wise Histogram of points */
end
 $F = \log_2 \frac{n}{\sqrt{nk}}$; /* Exponent of 2 in
guess for OPT */
for $f \in [F]$ **do**
 $\{\mathcal{H}_i^f, \mathcal{L}_i^f, \mathcal{M}_i^f : i \in [L]\} \leftarrow$
 HeavyCellMarker($\{\text{CH}^l : l \in [L]\}$, guess for OPT = $k\sqrt{n} \cdot 2^f$)
end

Algorithm 1: Step 1

many cluster sections for every heavy cell when calling the LSH subroutine on heavy cells separately, we construct a synthetic space out of the union of all heavy cells in a level and apply the LSH functions on this entire space. We will be able to extend the ℓ_2 metric in a natural way to work across cells, ensure that the cells are far enough apart in this distance measure so that bucket averages that land up “between” cells end up in the correct cell after projection, and that the diameter of this synthetic space is still small enough to keep the improvements we have derived so far.

We now give the pseudo-code for this algorithm in four pieces and walk through a sample run. Note that division of the algorithm in the steps coincides with the outline as given in the introduction; step 1 deals largely with the dimension reduction (but also tags cells as heavy, medium or light), steps 2a and 2b deal with the candidate center allocation, and step 3 runs the non-private k -means on the proxy data set and returns candidate centers in the original space.

Step 1 - Initialization and first interaction: We use public randomness to generate the dimension reduction map T and compose it with scaling and projection; this mapping is passed in L calls to Bitstogram in parallel to receive estimates of how many points lie in each cell in the dimension reduced space via histograms CH^l . We have geometrically varying guesses for OPT $k\sqrt{n}2^f$ for $f \in [F]$ where $F = \log_2(n/\sqrt{nk})$. Each guess generates a marking of cells $\{\mathcal{H}_i^f, \mathcal{L}_i^f, \mathcal{M}_i^f\}$ as being, heavy, light, or medium (the HeavyCellMarker algorithm which generates this marking is described in the supplementary).

Step 2a - Construction of synthetic space: M is the number of LSH scales used in any one level l , and R the num-

```

/* Step 2a - Construction of
synthetic space */
```

 $M = 1 + \log_2 d^{3/2} \sqrt{L} = O(\log \log n)$;
/* Number of LSH scales */
 $r_{l,m} = \frac{2^m t_l}{d \sqrt{L}}$ for $m \in [M]$; /* LSH scales */
 $R = O\left(\frac{\log k L^2 / \beta}{p(1)}\right)$; /* Number of
repetitions for LSH */
 $\lambda_l = (14c + 5)t_l \sqrt{d}$
 $V := \{v_C \in \mathbb{R}^{O(d)} : C \in \text{Anc}^*(\mathcal{H}_l)\}$ where v_C are
all nearly orthogonal unit vectors, i.e.
 $\|v_i - v_j\| \geq 1_{i \neq j}$
 $\Lambda_l^f(\cdot) := p \mapsto \begin{cases} (\lambda_l v_{\text{Anc}^*(C(p))}, p - o(C_l(p))) & \text{if } p \in \cup_{C \in \text{Anc}^*(\mathcal{H}^f)} C \\ 0 & \text{otherwise} \end{cases}$;
/* Mapping to LSH domain */

 $H_{l,r,m,f}(p) = \begin{cases} (p(1), p(c), r_{l,m}, cr_{l,m})\text{-sensitive} & \text{Hash function on the space } \Lambda_l^f \\ \text{if } \mathcal{C}_l(p) \in \mathcal{M}_l & \perp \text{ otherwise} \end{cases}$

Do in parallel for

 $f \in [F], l \in [L], m \in [M], r \in [R]:$
 $\text{BH}_{l,m,r,f} \leftarrow \text{Bitstogram}(H_{l,m,r}^f, \beta, \epsilon_{\text{BH}})$;
/* Bucket-wise Histogram of
points */
 $\text{BSO}_{l,m,r,f} \leftarrow$
HeavySumsOracle($H_{l,m,r,f}, \Lambda_l, \beta, \epsilon_{\text{BSO}}$) ;
/* Bucket Sum Oracle */
end

Algorithm 2: Step 2a

ber of independent repetitions of the hashing subroutine to boost the success probability. We compute a set of nearly orthogonal vectors indexed by the cells in $\text{Anc}^*(\mathcal{H}_l)$; since $|\text{Anc}^*(\mathcal{H}_l)| = O(k \log n)$, we only need $O(\log(k \log n)) = O(\log n)$ many dimensions at most. We define a mapping $\Lambda_l^f : \mathbb{R}^d \rightarrow \mathbb{R}^{O(\log n)} \times \mathbb{R}^d$ where the co-domain is a synthetic space mimicking the union of all heavy cells in level l . Essentially, for points p such that $\text{Anc}^*(\mathcal{C}_l(p))$ is a heavy cell, the image is a 2-tuple of an indicator vector indicating which heavy ancestor cell p lies in, and the p 's position with respect to the center of its ancestor cell. The mapping $H_{l,m,r,f}$ computes the output of the hash function for points in medium cells. The counts of the heavy buckets of these hash functions are recovered through Bitstogram via the bucket histogram $\text{BH}_{l,m,r,f}$ and the sums of all points mapping to heavy buckets are recovered via HeavySumsOracle and stored in the bucket sum oracle $\text{BSO}_{l,m,r,f}$.

Step 2b - Candidate center allocation: For every guess for OPT (parameterized by f) we allocate a candidate center $\Pi_l(\hat{b})$ for every heavy bucket b whose count \hat{n}_b crosses the

```

/* Step 2b - Candidate center
   allocation */  

 $S \leftarrow \emptyset$   

for  $f \in [F]$  do  

   $S_{\mathcal{H},f} \leftarrow \{o(C) : \exists iC \in \mathcal{H}_i\}$   

   $T_{l,f} = \frac{p(1)}{2} \cdot \max \left( \frac{\beta \text{OPT}}{t_i^2 k L^2 d}, \frac{4\text{BH}_M}{p(1)}, O \left( \frac{c_G \sqrt{n \text{poly log } n / \beta}}{\epsilon_{\text{BSO}}} \right) \right);$   

  /* Bucket threshold */  

   $S_{l,f} \leftarrow \emptyset$   

  for  $l \in [L], m \in [M], r \in [R]$  do  

    for  $(b, \hat{n}_b) \in \text{BH}^l$  such that  $\hat{n}_b \geq T_{l,f}$  do  

       $\hat{b} \leftarrow \frac{\text{BSO}^{l,m,r}(b)}{\text{BH}^{l,m,r}(b)}$   

       $\Pi_l(\hat{b}) \leftarrow \text{project } \hat{b} \text{ to } \Lambda_l$   

       $S_{l,f} \leftarrow S_{l,f} \cup \{\hat{b}\}$   

    end  

  end  

   $S_f \leftarrow S_{\mathcal{H},f} \cup \bigcup_{l=1}^L S_{l,f}$   

   $S \leftarrow S \cup S_f$   

end

```

Algorithm 3: Step 2b

threshold $T_{l,f}$. The center is allocated at the bucket point average estimate $\text{BSO}^{l,m,r,f}(b)/\text{BH}^{l,m,r,f}(b) = \hat{n}_b$. This average is projected to the embedding of heavy cells in Λ_l to get the point $\Pi_l(\hat{b})$, which is naturally identified with a point in the original data domain; these projections are collected to form the set of candidate centers $S_{l,f}$. We allocate a candidate center at the center of every heavy cell to get $S_{\mathcal{H},f}$. The centers allocated for the guess for OPT parameterized by f are stored in S_f . The total bi-criteria solution then is simply $S = \bigcup_{f \in [F]} S_f$.

Step 3 - Proxy data set construction and CenterRecovery: We release the privately derived set of candidate centers, i.e. the bicriteria clustering solution S and get the candidate center histogram CCH that for each $s \in S$ returns a privatized count of the number of points for which s is the closest candidate center. The proxy data set D^* is then simply each point $s \in S$ repeated with multiplicity $\text{CCH}(s)$. We now apply any non-private k -means algorithm to D^* and derive cluster centers $S^* = \{s_1^*, \dots, s_k^*\}$. Note that this implicitly defines a clustering of the original data set D' where a point $p \in D'$ lies in cluster i if $\arg \min_j \|Q(p) - s_j^*\|^2 = i$. To compute the cluster centers in the original space, we invite agents to release their original locations privatized via theorem 2.1 (the response $\hat{v}(p)$), and in the same round of interaction derive SH, the cluster centers histogram, which estimate the number of data points that lie in the i th cluster via a call to Bitstogram. We then divide the sum of noisy vectors by the noisy count for each cluster to compute an estimate for the true cluster centers, which is our final output.

```

/* Step 3 - Proxy data set
   construction and center recovery */
Data: Bicriteria  $k$ -means relaxation  $S$  for  $k$ -means
   clustering under dimension reducing
   transformation  $M$ , the transformation
    $M : \mathbb{R}^{d'} \rightarrow \mathbb{R}^d$ 
Setting: Distributed dataset  $D' \subset \mathbb{R}^{d'}$  over  $n$  agents
 $s(p) := p \mapsto \arg \min_{s \in S} \|p - s\|_2$ 
CCH = Bitstogram( $s(\cdot), \beta, \epsilon_{\text{SH}}$ ); /* Candidate
   center histogram */
 $D^* \leftarrow \{s \in S \text{ with multiplicity } \text{SH}(s)\}$ 
 $S^* = \{s_1^*, \dots, s_k^*\} \leftarrow \text{Standard } k - \text{Means}$ 
 $s^*(p) := p \mapsto \arg \min_{s^* \in S^*} \|M(p) - s^*\|_2$ 
Do in parallel:
  Agents reveal  $\hat{v}(p)$  for  $p \in D'$  via theorem 2.1
  SH = Bitstogram( $s^*(\cdot), \beta, \epsilon_{\text{SH}}$ );
  /* Cluster centers histogram */
end
 $\hat{v} = \sum_{p \in D'} \hat{v}(p)$ 
 $\hat{s}^* = \sum_{p \in D'} \hat{s}^*(p)$ 
for  $j = 1, \dots, k$  do
   $\hat{\mu}_j = \frac{\hat{v}_j}{\text{SH}(s_j^*)}$ 
end
return  $S' = \{\hat{\mu}_1, \dots, \hat{\mu}_k\}$ 

```

Algorithm 4: Step 3

Acknowledgements

All the authors were supported by the National Science Foundation under NSF grants AF 1909314 and CAREER 1750716.

References

- Andoni, A.; and Indyk, P. 2006. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings*, 459–468. IEEE Computer Society.
- Balcan, M.-F.; Dick, T.; Liang, Y.; Mou, W.; and Zhang, H. 2017. Differentially private clustering in high-dimensional euclidean spaces. In *International Conference on Machine Learning*, 322–331. PMLR.
- Bassily, R.; Nissim, K.; Stemmer, U.; and Thakurta, A. 2020. Practical Locally Private Heavy Hitters. *J. Mach. Learn. Res.*, 21: 16:1–16:42.
- Braverman, V.; Frahling, G.; Lang, H.; Sohler, C.; and Yang, L. F. 2017. Clustering High Dimensional Dynamic Data Streams. *CoRR*, abs/1706.03887.
- Chang, A.; Ghazi, B.; Kumar, R.; and Manurangsi, P. 2021. Locally Private k -Means in One Round. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, 1441–1451. PMLR.

Duchi, J. C.; Jordan, M. I.; and Wainwright, M. J. 2013. Local Privacy and Statistical Minimax Rates. *CoRR*, abs/1302.3203.

Dwork, C.; Kenthapadi, K.; McSherry, F.; Mironov, I.; and Naor, M. 2006. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 486–503. Springer.

Erlingsson, Ú.; Pihur, V.; and Korolova, A. 2014. RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. In Ahn, G.; Yung, M.; and Li, N., eds., *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, 1054–1067. ACM.

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

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

Makarychev, K.; Makarychev, Y.; and Razenshteyn, I. P. 2019. Performance of Johnson-Lindenstrauss transform for k -means and k -medians clustering. In Charikar, M.; and Cohen, E., eds., *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, 1027–1038. ACM.

Nguyen, H. L.; Chaturvedi, A.; and Xu, E. Z. 2021. Differentially Private k-Means via Exponential Mechanism and Max Cover. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Event, February 2-9, 2021*, 9101–9108. AAAI Press.

Nissim, K.; and Stemmer, U. 2018. Clustering algorithms for the centralized and local models. In *Algorithmic Learning Theory*, 619–653. PMLR.

Stemmer, U. 2020. Locally private k-means clustering. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 548–559. SIAM.

Stemmer, U.; and Kaplan, H. 2018. Differentially Private k-Means with Constant Multiplicative Error. In Bengio, S.; Wallach, H. M.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, 5436–5446.

Thakurta, A. G.; Vyrros, A. H.; Vaishampayan, U. S.; Kapoor, G.; Freudiger, J.; Sridhar, V. R.; and Davidson, D. 2017. Learning new words. *Granted US Patents*, 9594741.