**Locally Private** k**-Means Clustering with Constant Multiplicative Approximation**

**and Near-Optimal Additive Error**

**~**

**Anamay Chaturvedi, Matthew Jones, Huy Lê Nguyên**

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 cen-

ters) such that the sum of the l-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 arbi- trarily close to optimal, but suffers high additive error. The additive error has also been seen to be an issue in imple- mentations of differentially private k-means clustering algo- rithms in both the central and local settings. In this work, we introduce a new locally private k-means clustering algo- rithm that achieves near-optimal additive error whilst retain- ing constant multiplicative approximation factors and round complexity. Conc~retely, given any c > √2, our algorithm achieves O(k1+O(1/(c2 -2))√d,n log d, poly log n) additive error with an O(c2 ) multiplicative approximation factor.

**Introduction**

Given a set D, of npoints in a d, -dimensional ball with unit diameter in Euclidean space, the k-means clustering prob-

lem asks for a set of k points S such that the sum of l-

distances from each data point to the closest respective point in S, which we denote fD, (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 pro- cessed and the output of this computation is revealed pub- licly. 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** (Differential privacy (DP), (Dwork et al. 2006))**.** Two datasets D1 , D2 ∈ Xn are *neighboring* if they differ in at most one record. An algorithm A : Xn → Y is said to be ∈*-differentially private* (DP) if for any S C Y and

Copyright © 2022, Association for the Advancement of Artiﬁcial

Inten(..r.til i irdp.roofs and missing

details may be found at (Chaturvedi, Jones, and Nguyen 2021).

any two neighboring datasets D1 , D2 ∈ Xn ,

P(A(D1 ) ∈ S) ≤ exp(∈)P(A(D2 ) ∈ S).

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 sin- gle value from the data universe. In practice, this guarantee is fulﬁlled by adding carefully calibrated noise to quanti- ties 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 typ- ically 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 2** (Local differential privacy (LDP), (Ka- siviswanathan et al. 2011))**.** Consider a protocol which queries some functions of individual records in a distributed private dataset over r rounds of interaction, and let the re- sponse to the queries for the private datum p be A(p) = (A1 (p), . . . , Ar (p)), where Ai (p) is the response in the ith round. We say that this protocol is ∈*-locally differentially private* (LDP) if the algorithm that outputs privatized re- sponses for any agent,i.e. p →} A(p), is ∈-DP.

This constraint forces strong lower bounds for locally pri- vate protocols; a lower bound of Ω((k + √n)/∈) is known for the additive error of any ∈-LDP interactive constant- factor multiplicative approximation algorithm for the k- means clustering problem that operates in a constant num- ber 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 Feld- man 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 so- lution for the private 1-cluster problem and consequently the ﬁrst LDP algorithm for the k-means problem with prov- able guarantees, achieving a multiplic~ative approximation of O(k) and an additive error term of O(n2/3+a · d1/3 · √k). The exponent of n in the additive error holds for arbitrarily small a at the cost of looser multiplicative approximation guarantees, a trade-off which appears in most later work as well. This artifact is the consequence of using *local- ity sensitive hashing* (LSH) families to detect accumulation of data. Stemmer and Kaplan gave the ﬁrst constant fac- tor multiplicative approxim~ation algorithm for this problem within an additive error of O(n2/3+a ·d1/3 ·k2 ). 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. T~he additive error was further brought down by Stemmer to O(n1/2+a ·k ·max{√d, √k}), in which work a lower bound of Ω(√n) was also proved, as mentioned before. Most recently Chang et al. introduced a one-round protocol for LDP k-means in the (∈, 0) setting. They get a multiplicative approximation of η(1 + α) where η is the multiplicative approximation guarantee of any given non-private k-means algorithm and an additive error term of kOα (1) · √nd, · poly log(n)/∈ . We see that the trade-off be- tween the additive and multiplicative approximations in this algorithm has been shifted from n to k. However, the depen- dence of the Oα (1) exponent of k on α is at least Ω(1/α)2 and can make the additive error prohibitively large depend- ing 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. Exper- imental 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 algo- rithm 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 em- pirical evidence, our goal is to reduce the additive error to near-optimal at the cost of looser multiplicative approxima- tion instead of the other way around; we succeed in this goal by driving down the exponent of k to 1 + O(1/(c2 — 2)), where c is an LSH parameter that can be set to any value ≥ √2. This shows for the ﬁrst time that it is possible to have constant factor multiplicative approximation k-means clus- tering algorithms in the LDP setting with additive error that

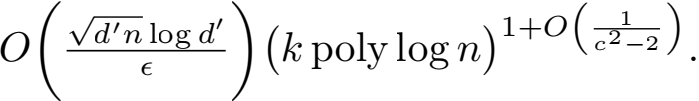
Work Mult. App. Add. Error

|  |  |  |
| --- | --- | --- |
| 2018 | O(k) | (n2/3+a · d,1/3 · √k) |
| 2018 | O(1) | ~ (n2/3+a · d,1/3 · k2 ) |
| 2020 | O(1) | O(n1/2+a · k · max{√d, , √k}) |
| 2021 | (1 + α)η | (n1/2 · d,1/2 · k (1/α2 ) ) |
| LMA | (1 + α)η | (n1/2 · d,1/2 · k (1/α2 ) ) |
| **LAE** | O(c2) |  |

Table 1: Comparison with recent LDP algorithms for k- means. LMA and LAE are introduced in this paper, and LAE is our main co~ntribution. The data domain is a unit diame- ter ball. The O notation suppresses privacy parameters and logarithmic terms, where the additive errors all have a 1/∈ factor. The asummand in the exponents of the ﬁrst three al- gorithms is an arbitrarily small constant, but the constant in the respective Mult. App. terms depends on a. η is the low- est approximation factor that we can achieve for non-private k-means. The value c can be set to any real number greater than √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).

has a truly square-root dependence on the data set size and the ambient dimension and arbitrarily close to linear depen- dence on the number of cluster centers. Formally, our main result is:

**Theorem 3.** *For* ∈ < 1*, Low Additive Error LDP* k*-Means (LAE) is an* ∈*-locally differentially private algorithm for the* k*-means clustering problem that after four rounds of inter- action outputs a set* S, *of size* k *such that with constant prob- ability the clustering cost* fD, (S, ) *equals* O(c2 OPT , ) +



We also introduce a second algorithm that achieves a sim- ilar 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 brieﬁyin this paper and relegate most of the details to the extended version.

**Theorem 4.** *For* ∈ < 1*, Low Multiplicative Approximation*

*LDP* k*-Means (LMA) is an* ∈*-locally differentially private al-*

*gorithm for the* k*-means clustering problem that after one*

*round of interaction outputs a set* S, *of size* k *such that*

*the* ~*clustering cost* fD, (S, ) *equals* (1 + O(α))η OPT , +

kO(1/α2 ) √d,n log d, poly log n. *where* η *is the best ap-*

*proximation factor of anon-private algorithm.*

Although we will sketch the main ideas of and pro- vide 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 extended version 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.

will lead to additive error O(max{√ | S|n, kp √n logq n}) down the line (omitting the dependence on dimension). In order to avoid an exponent of 1/2 + a on n as in the pre- vious works which generate S by detecting data accumula- tion via LSH functions, it is necessary to ﬁnd a bi-criteria solution with O(poly k poly log n) candidate centers with O(poly k√n 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.

**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 l2 dis- tances 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 pre- serving 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) C Rd 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 *bi- criteria 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 the opti- mal k-means cost, 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 that if we move each data point to the closest center in the set S;

the sum of the l distances moved over all points is about the

optimal clustering cost by the construction of S. Therefore, essentially by applications of the triangle inequality, the val- ues of the k-means clustering functions of D and D\* are at most O( 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 algorithm of our choice and generate k centers in the low dimensional space which work well for D. Now we appeal to LDP pri- vate averaging to privately recover the mean of each cluster in the original space, and we are done.

It was observed by Stemmer that the one of the main road-blocks in computing solutions with low additive error is generating 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 re-

spect to which the clustering ˜cost is at most O(kp √n logq n)

**LDP** k**-means with arbitrarily tight multiplicative ap- proximation:** We brieﬁy describe how we get near opti- mal 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 cen- ters 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 re- cent advances in dimension reduction fork-means clustering (Makarychev, Makarychev, and Razenshteyn 2019) which

duction to O(log k/α2 ) many dimensions preserves the k- means cost of every clustering of a data set within a multi- plicative approximation of (1 ± α); 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 kin the additive error.

show that J˜ohnson-Lindenstrauss (JL) style dimension re-

At a high level, this approach can be analyzed as fol- lows. 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 set- ting geometric thresholds of 1, 1/2, 1/4, and so on, the lth shell B(SOPT, 1/2l—1) —B(SOPT, 1/2l ) has the prop- erty that every data point in it has an optimal clustering cost of O(1/(2l )2 ) units. To cluster the data points that oc- cur in the lth shell, if we consider a grid with side-length α/(2l √d) then we have the guarantee that the closest grid point for every data point is at a distance α times its op- timal clustering distance. In principle, the entire dataset D co~uld lie in the lth shell, but we are able to show that picking kO(1/α2 ) poly log n centers from this gridsufﬁces to ensure that most points in the lth shell are within an O(α/(2l )2 ) distance of some candidate center. As the grid scales vary geometrically, it turns out only log n grids are needed in all a~nd the size of the set of candidate centers generated is kO(1/α2 ) 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(α OPT ), allowing us to achieve a 1 + O(α) inﬁation in the multiplicative approximation as opposed to an O(1) inﬁation.

**Challenges in reducing the additive error:** Many previ- ous works with tight bounds on the exponents of n and k in the additive error proceed by using LSH functions to dis- cretize the response. We recall that LSH functions with pa- rameters (r, cr, p, q) with c > 1 and p > q are hash func- tions with the property that for any pair of points in the input domain within a distance of r units, the probability of col- liding 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 func- tions for clustering is to allocate candidate cluster centers by computing point averages of heavy hash buckets forLSH functions at geometrically varying scales (similar to how we use geometrically varying grid unit lengths for tight multi- plicative approximation. Just as allocating points from the lth grid worked well to cluster the lth shell, using LSH func- tions with scale r = 1/2l allows us to capture points which lie in the lth shell with respect to any ﬁxedk-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 thresh- olds, and the ratio between the collision probabilities p and q which determineshow 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.

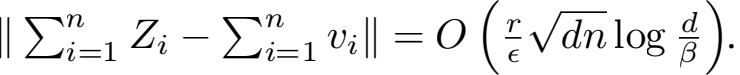
**Preliminaries**

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

**Theorem 5** ((Duchi, Jordan, and Wainwright 2013))**.** *There is an* ∈*-LDP mechanism for* ∈ ≤ 1 *to privately release a vector* v *such that if* Z *is the value returned then* E[Z] = v*. Further,* ⅡZⅡ2 ≤ B0 r√d/∈ *for some universal constant* B0 < ∞ *.*

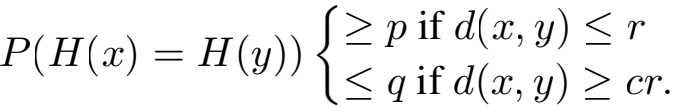
Since the output of the algorithm given by Theorem 5 is a vector of bounded length, we can apply standard concentra- tion bounds to get an ∈-LDP mechanism for locally privately computing the mean of n independent private d-dimensional vectors.

**Corollary 6** (LDP mean estimation)**.** *For private vec- tors* v1 , . . . , vn ∈ Rd *and their respective privatized re- leases* Z1, . . . , Zn*, we have that with probability* 1 — β*,*



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

**Deﬁnition 7** (Locality sensitive hashing (LSH))**.** We say that a family of hash functions H : Rd → B for a ﬁnite set of buckets B is *locality-sensitive* with parameters (p,q, r, cr) if for every x, y ∈ Rd for some 1 ≥ p > q ≥ 0, r > 0 and c > 1



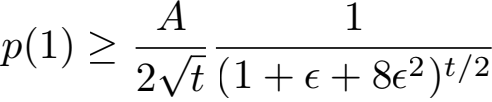
In this work we use a near-optimal LSH family construc- tion from (Andoni and Indyk 2006).

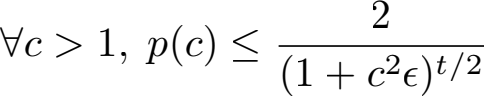
**Theorem 8.** *For every sufﬁcientlylarge* d *and* n *there exists a family* H *of hash functions deﬁned on* Rd *such that for a dataset of size* n*,*

*1. A function from this family can be sampled, stored and computed in time* tO(t) log n + O(dt)*, where* t *isa free positive parameter of our choosing.*

*2. The collision probability for two points* u, v ∈ Rd *de- pends only on the* l2 *distance between them, which we henceforth denote by* p(Ⅱu — vⅡ)*.*

*3. The following inequalities hold:*





*where* A *is an absolute constant* < 1*, and* ∈ = Θ(t-1/2)*. One can choose* ∈ =  *.*

*4. The number of buckets* NB *anLSH function with param-*

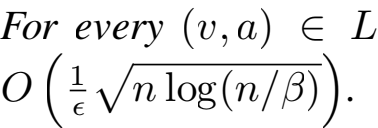
*eter* t *uses is* tO(t) log n*.*

In the locally private setting, to ∈-privately release a point directly requires adding a noise vector with length propor- tionalto 1/∈ to their private data, making it impossible to pri- vately 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. Al- though 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 of privacy.

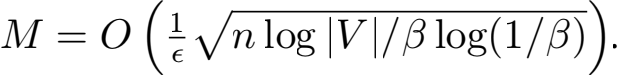
**Lemma 9** (Algorithm Bitstogram, (Bassily et al. 2020))**.** *Let* V *be a ﬁnite domain of values, let* f : D/ → V*, and let* n(v) *denote the frequency with which* v *occurs in* f (D/ )*. Let* ∈ ≤ 1*. Algorithm* Bitstogram(f, ∈, β) *interacts with the set of* n *users in 1 round and satisﬁes* ∈*-LDP. Further, it returns*

*alist* L = ((vi , ai ))i *of*˜ *value-frequency pairs (i.e. elements*

*of* V ×R*) with length* O(√n) *such that with probability* 1—β *the following statements hold:*

*1.* *,* Ⅱa — f Ⅱ ≤ E *where* E =

*2. For every* v ∈ V *such that* f (v) ≥ M*,* v ∈ L*, where*



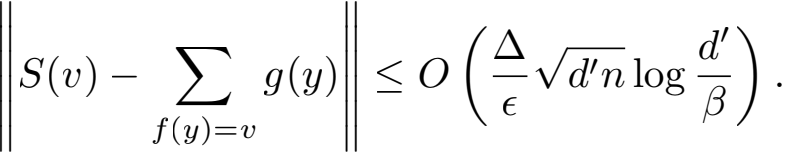
*We overload notation to treat the list returned by* Bitstogram *returns as either a set of (heavy-hitter, frequency) pairs or a function which maybe queried on a value to return either the corresponding frequency if it is a heavy hitter or a value of* 0 *otherwise. A subscript of* M *denotes the upper bound on the maximum frequency omitted. We see that whenever* |V | = Ω(n)*, we have that* M = Ω(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 : X → V and a vector-valued func- tion g : X → Rd , the sum estimation oracle privately re- turns for every heavy hitter v ∈ V the sum of all agents that map to x, i.e. Σp:f (p)=xp. We recall that Bitstogram is a modular algorithm with two subroutines; a frequency or- acle 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 construc- tion of HeavySumsOracle essentially mimics the frequency oracle ExplicitHist from (Bassily et al. 2020) and can be run in parallel with Bitstogram, allowing us to reduce the

round complexity of our protocols. It is similar to the Bucke- tized Vector Summation Oracle construction of (Chang et al. 2021). The pseudocode of HeavySumsOracle and the proof of Lemma 10 may be found in the extended version.

**Lemma 10** (HeavySumsOracle)**.** *Let* f : X → V*,* g : X → B(0, △/2) C Rd, *and* △ > 0 *be some publicly known functions and parameters, and let* D, C X *be a distributed dataset over* n *users. Let the private parameter* ∈ ≤ 1*. With probability at least* 1 — β*, for every* v ∈ V *that occurs in* f (D, )*, if* S(v) *is the value returned by the* HeavySumsOracle *then*



*Further,* HeavySumsOracle *is* ∈*-LDP.*

**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 out- line, where we privately generate a bicriteria solution that has O( OPT ) clustering cost with small additive error and only O(k1+O(1/(c2 -2)) 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 2d-ary tree of cells is constructed, where each rectangular cell is recursively subdivided into 2d 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 cellistl = 2-l units. L = log n levels of the gridsufﬁce to discretize the domain to a sufﬁciently ﬁne degree. This construction follows the dimension reduc- tion step ensuring d = O(log n). The authors observe that if we *randomly shift* this hierarchy of cells, thenin expec- tation there are O(1) many cells with side-length tl within an l2 distance of tl /d units of any ﬁxed point; we use this observation multiple times in the sequel.

**Guessing the optimal cost:** Suppose we knew the opti- mal cost OPT , and let S OPT be some arbitrary k-means solution. Data points in cells further than tl /d units away

from S OPT must have a clustering cost of at least t/d2 ,

but the sum of their costs cannot exceed the total cost OPT .

There are hence at most OPTd2 /t many such points. Trac-

ing a similar argument with cells, we derive a threshold

Tl ≈ OPT /(tk poly log n) (dropping some terms) such

that there cannot be more than O(klog n) many cells that have more than Tl points further than tl /d units from S OPT (across all L levels). By the random shift observation there are unconditionally at most O(1) ·|S OPT |·L = O(klog n) many cells closer than tl /d to SOPT so we get that re- gardless of where they lie in the domain there are at most O(klog n) *heavy* cells in any level, i.e. cells that beat the threshold Tl for their level. Any cell which is not heavy is called *light*.

Assuming OPT is known, this allows us to identify re- gions of the domain at different scales where data accumu- lates beyond these thresholds. We guess values for OPT varying in factors of 2 from k√n (the targeted additive er- ror) to n (trivially achieved clustering cost) and allocate can- didate centers for all guesses. To allocate candidate centers for any level we use LSH functions as discussed in the intro- duction, but we only invite responses from points in *medium* cells, the light children of heavy cells.

**Allocating candidate centers:** LSH functions with pa- rameters (p,q, r, cr) promise that any two points in the do- main 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 n1/2+a term as previ- ous work; we now discuss some ideas behind our improve- ment.

**The** n1/2+a **barrier:** Fixing a point x in an opti- mal 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 · |{points from C colliding with x}| + △

·

|{points further than cr units from x}|, where △ is the di- ameter 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 △/rand |Dl |/|C|, where Dl 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/Θ(c) . In previous work, the term that p/q had to beat scaled as a polynomial inn; driving up the success probability by running 1/p many independent copies of this scheme is what lead prior work to incur a small na factor in the number of candidate centers allocated. The decomposition of the data set across different levels with de- creasing side-lengths and increasing thresholds allows us to bound both △/rand |Dl |/|C| by k poly log n, which is how we manage to avoid tuning the collision probabilities too ag- gressively and avoid thena factor as before, substituting it instead with a k1/O(c2 -2) 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(klog n)

tion of O(k2 ) many centers to serve the O(k2 ) many cluster sections. In order to reduce the exponent of k in the num- ber of cluster centers allocated, we make three algorithmic choices.

heavy ce˜lls in any one level, which would˜ require an alloca-

Algorithm 1: Step 1 - Initialization and ﬁrst interaction

|  |  |  |
| --- | --- | --- |
| 1: √ ← uniformly random vector in [−1/2, 1/2]d | | |
| 2: T : Rd/ → Rd dimension reduction for d = | | |
| O(log(k/αβ)/α2 )  3: S : Rd → Rd |  | |
| 4: P : Rd → B(0, 1/2) projection to B(0, | 1/2) followed | |
| by translation by √  5: Q = P ◦ S ◦ T : Rd/ → B(0, 1) ⊂ Rd |  | |
| 6: L = lg n |  | |
| 7: **for** l ∈ [L] **do in parallel** |  | |
| 8: CHl ← Bitstogram(Cl ◦ Q(·), ∈CH , β Cell-wise Histogram of points | /L) D | |
| 9: **end for** |  | |
| 10: F = log2  D Exponent of 2 in g | uess for OPT | |
| 11: **for** f ∈ [F] **do** |  | |
| 12: {H , L , M : i ∈  HeavyCellMarker({CHl : l ∈ [L]}  OPT = k√n · 2f ) | [L]}  D guess | ← for |
| 13: **end for** |  | |

Firstly, we allocate a candidate center at the center of ev- ery heavy cell (i.e. at most O(klog n) · L = O(k log2 n) more candidate centers). This guarantees that every point in the heavy cells in level lhas a candidate center at a distance of tl √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 d3/2tl. Since tl = 2-l, the conse- quence of these two modiﬁcations is that we only need to allocate cluster centers within a distance of 2-l√dunits of any point of Dl , and that since there are only O(1) many cells with side-length 2-ld3/2 within a distance of 2-l√d units of an optimal center (again by the random shift obser- vation), there are only O(k) many cluster sections we must account for.

Thirdly, in order to avoid dealing with the worst case O(k) 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 naturally extend the l2 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 improve- ments we have derived so far.

We now give the pseudocode 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.

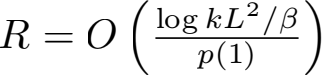
**Step 1 - Initialization and ﬁrst interaction:** We use pub- lic randomness to generate the dimension reduction map T

Algorithm 2: Step 2a - Construction of synthetic space

M = 1 + log2 d3/2√L = O(log log n) D Number of

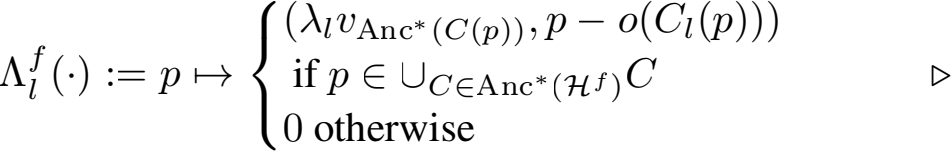
LSH scales

rl ,m =  form ∈  D LSH scales

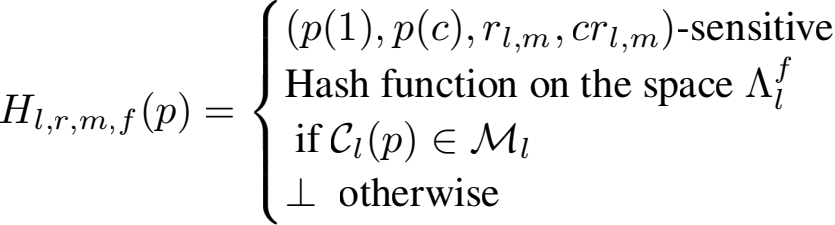
 D Number of repetitions for LSH

λl = (14c + 5)tl √d

V := {vC ∈ RO(d) : C ∈ Anc\* (Hl )} where vC are all nearly orthogonal unit vectors, i.e. Ⅱvi − vj Ⅱ ≥ 1ij



Mapping to LSH domain



**for** f ∈ [F], l ∈ [L], m ∈ [M], r ∈ [R] **do in parallel**

BHl,m,r,f ← Bitstogram(Hlf,m,r , β, ∈BH ) D

Bucket-wise Histogram of points BSOl,m,r,f ←

HeavySumsOracle(Hl,m,r,f, Λl , β, ∈BSO ) D Bucket Sum Oracle

**end for**

and compose it with scaling and projection; this mapping is passed in L calls to Bitstogram in parallel to receive es- timates of how many points lie in each cell in the dimen- sion reduced space via histograms CHl. We have geometri- cally varying guesses for OPT k √n2f for f ∈ [F] where F = log2 (n/√nk). Each guess generates a marking of

cells {H , L , M} as being, heavy, light, or medium (the

HeavyCellMarker algorithm which generates this marking is described in the extended version).

**Step 2a - Construction of synthetic space:** M is the number of LSH scales used in any one level l, and R the number 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 Anc\* (Hl ); since |Anc\* (Hl )| = O(klog n), we only need O(log(klog n)) = O(log n) many dimensions at most. We deﬁne a mapping

Λ : Rd → RO(log n) × Rd where the co-domain is a syn-

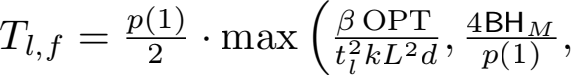
thetic space mimicking the union of all heavy cells in level l. Essentially, for points p such that Anc\* (Cl (p)) is a heavy cell, the image is a 2-tuple of an indicator vector indicat- ing which heavy ancestor cell plies in, and the p’s position with respect to the center of its ancestor cell. The mapping Hl,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 BHl,m,r,f and the sums of all points map- ping to heavy buckets are recovered via HeavySumsOracle

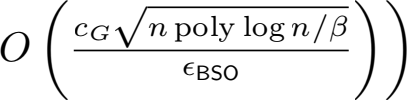
Algorithm 3: Step 2b - Candidate center allocation

S ← ∅

**for** f ∈ [F] **do**

SH,f ← {o(C) : ∃iC ∈ Hi }

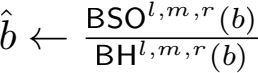


 D Bucket threshold

Sl,f ← ∅

**for** l ∈ [L], m ∈ [M], r ∈ [R] **do**

**for** (b, b ) ∈ BHl such that b ≥ Tl,f **do**



Πl () ← project to Λ

Sl,f ← Sl,f ∪ {}

**end for end for**

Sf ← SH,f ∪ u Sl,f S ← S ∪ Sf

**end for**

and stored in the bucket sum oracle BSOl,m,r,f .

**Step 2b - Candidate center allocation:** For every guess for OPT (parameterized by f) we allocate a candidate cen-

ter Πl () for every heavy bucket b whose count b crosses

the threshold Tl,f . The center is allocated at the bucket point

average estimate BSOl,m,r,f (b)/BHl,m,r,f (b) = b. This

average is projected to the embedding of heavy cells in Λl to

get the point Πl (), which is naturally identiﬁed with a point

in the original data domain; these projections are collected to form the set of candidate centers Sl,f We allocate a can- didate center at the center of every heavy cell to get SH,f . The centers allocated for the guess for OPT parameterized by f are stored in Sf . The total bi-criteria solution then is simply S = ∪f∈[F]Sf .

**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 cen- ter histogram CCH that for each s ∈ S returns a privatized count of the number of points for which s is the closest can- didate center. The proxy data set D\* is then simply each point s ∈ S repeated with multiplicity CCH(s). We now apply any non-private k-means algorithm to D\* and derive

cluster centers S\* = {s . . . . , s}. Note that this implicitly

deﬁnes a clustering of the original dataset D/ where a point

p ∈ D/ lies in cluster iifargminj ⅡQ(p) — s Ⅱ2 = i. To

compute the cluster centers in the original space, we invite agents to release their original locations privatized via Theo-

rem 5 (the response (p)), and in the same round of interac-

tion derive SH, the cluster centers histogram, which estimate the number of data points that lie in the ith 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.

Algorithm 4: Step 3 - Proxy data set construction and center recovery

**Input:** Bicriteria k-means relaxation S fork-means clus- tering under dimension reducing transformation M, the transformation M : Rd/ → Rd

s(p) := p →} arg mins∈S Ⅱp — sⅡ2

CCH = Bitstogram(s(·), β, ∈SH ) D Candidate center

histogram

D\* ← {s ∈ S with multiplicity SH(s)}

S\* = {s , . . . , s} ← Standard k — Means s\* (p) := p →} arg mins\* ∈S \* ⅡM(p) — s\* Ⅱ2

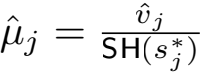
**do in parallel**

Agents reveals (p) for p ∈ D/ via Theorem 5

SH = Bitstogram(s\* (·), β, ∈SH ) D Cluster centers histogram

**end parallel**

= Σp∈D/ (p) \* = Σp∈D/ \* (p)

**for** j = 1, . . . , k **do** 

**end for**

**Output:** S/ = {1 , . . . , k }

**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 Al- gorithms for Approximate Nearest Neighbor in High Di- mensions. In *47th Annual IEEE Symposium on Founda- tions 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.

Chaturvedi, A.; Jones, M.; and Nguyen, H. L. 2021. Locally Private k-Means Clustering with Constant Multiplicative Approximation and Near-Optimal Additive Error. *CoRR*, abs/2105.15007.

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 Re- sponse. In Ahn, G.; Yung, M.; and Li, N., eds., *Proceed- ings 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. Core- sets for differentially private k-means clustering and appli- cations to privacy in mobile sensor networks. In *2017 16th ACM/IEEE International Conference on Information Pro- cessing in Sensor Networks (IPSN)*, 3-16. IEEE.

Kasiviswanathan, S. P.; Lee, H. K.; Nissim, K.; Raskhod- nikova, 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 Co- hen, 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. Dif- ferentially 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 Learn- ing Theory*, 619-653. PMLR.

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

*Proceedings of the Fourteenth Annual ACM-SIAM Sympo- sium 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 Infor- mation 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.