Locally Private k-Means in One Round

Alisa Chang [1] Badih Ghazi [1] Ravi Kumar [1] Pasin Manurangsi [1]

Abstract

We provide an approximation algorithm for k-means clustering in the *one-round* (aka *non-interactive*) local model of differential privacy (DP). This algorithm achieves an approximation ratio arbitrarily close to the best *non private* approximation algorithm, improving upon previously known algorithms that only guarantee large (constant) approximation ratios. Furthermore, this is the first constant-factor approximation algorithm for k-means that requires only *one* round of communication in the local DP model, positively resolving an open question of Stemmer (2020). Our algorithmic framework is quite flexible; we demonstrate this by showing that it also yields a similar near-optimal approximation algorithm in the (one-round) shuffle DP model.

1. Introduction

The vast amounts of data collected for use by modern machine learning algorithms, along with increased public awareness for privacy risks, have stimulated intense research into privacy-preserving methods. Recently, differential privacy (DP) (Dwork et al., 2006b;a) has emerged as a popular definition, due to its mathematical rigor and strong guarantees. This has translated into several practical deployments within government agencies (Abowd, 2018) and the industry (Erlingsson et al., 2014; Shankland, 2014; Greenberg, 2016; Apple Differential Privacy Team, 2017; Ding et al., 2017), and integration into widely-used libraries such as TensorFlow (Radebaugh and Erlingsson, 2019) and PyTorch (Testuggine and Mironov, 2020).

Among unsupervised machine learning tasks, clustering in general and k-means clustering in particular are one of the most basic and well-studied from both theoretical and practical points of view. The research literature on k-means is extensive, with numerous applications not only in machine learning but also well beyond computer science. Computationally, finding optimal k-means cluster centers is NP-hard (e.g., Aloise et al., 2009) and the best known polynomial-time algorithm achieves an approximation ratio of 6.358 (Ahmadian et al., 2020).

A natural question is to study clustering and k-means with DP guarantees. There is a growing literature on this topic (Blum et al., 2005; Nissim et al., 2007; Feldman et al., 2009; Gupta et al., 2010; Mohan et al., 2012; Wang et al., 2015; Nissim et al., 2016; Nock et al., 2016; Su et al., 2016; Feldman et al., 2017; Balcan et al., 2017; Nissim and Stemmer, 2018; Huang and Liu, 2018; Nock et al., 2016; Nissim and Stemmer, 2018; Stemmer and Kaplan, 2018; Stemmer, 2020; Ghazi et al., 2020b; Jones et al., 2021; Chaturvedi et al., 2021). In contrast to non-private k-means, additive errors are necessary in this case, and a recent line of work has obtained increasingly tighter approximation ratios, culminating with the work of Ghazi et al. (2020b), which comes arbitrarily close to the smallest possible non-private approximation ratio (currently equal to 6.358 as mentioned above). However, this last result only holds in the *central* model of DP where a curator is entrusted with the raw data and is required to output private cluster centers.

The lack of trust in a central curator has led to extensive study of *distributed* models of DP, the most prominent of which is the *local* setting where the sequence of messages sent by each user is required to be DP. The state-of-the-art local DP k-means algorithm is due to Stemmer (2020); it achieves a constant (that is at least 100 by our estimate) approximation ratio in a constant (at least five) number of rounds of interaction. In fact, all (non-trivial) known local DP k-means algorithms are *interactive*, meaning that the analyzer proceeds in multiple stages, each of which using the (private) outcomes from the previous stages. This begs the question, left open by Stemmer (2020): is there a non-interactive local DP approximation algorithm for k-means and can it attain the non-private approximation ratio?

1.1. Our Contributions

We resolve both open questions above by showing that there is a DP algorithm in the non-interactive local model

[*]Equal contribution [1]Google Research, Mountain View, CA. Correspondence to: Alisa Chang <alisac@google.com>, Badih Ghazi <badihghazi@gmail.com>, Ravi Kumar <ravi.k53@gmail.com>, Pasin Manurangsi <pasin@google.com>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

with an approximation ratio that is arbitrarily close to that of any non-private algorithm, as stated below[1].

Theorem 1. *Suppose that there is a (not necessarily private) polynomial-time approximation algorithm for k-means with approximation ratio κ. For any $\alpha > 0$, there is a one-round ϵ-DP $(\kappa(1+\alpha), k^{O_\alpha(1)} \cdot \sqrt{nd} \cdot \operatorname{polylog}(n)/\epsilon)$-approximation algorithm for k-means in the local DP model. The algorithm runs in time* $\operatorname{poly}(n, d, k)$.

Here and throughout, we assume that the users and the analyzer have access to shared randomness. We further discuss the question of obtaining similar results using only private randomness in Section 6. Furthermore, we assume throughout that $0 < \epsilon \leq O(1)$ and will not state this assumption explicitly in the formal statements.

The dependency on n in the additive error above is also essentially tight, as Stemmer (2020) proved a lower bound of $\Omega(\sqrt{n})$ on the additive error. Our bound also improves upon his protocol, where the additive error grows with $n^{1/2+a}$ for an arbitrary small positive constant a.

In addition to the local model, our approach can be easily adapted to other distributed privacy models. To demonstrate this, we give an algorithm in the shuffle model (Bittau et al., 2017; Erlingsson et al., 2019; Cheu et al., 2019)[2] with a similar approximation ratio but with a smaller additive error of $\tilde{O}(k^{O_\alpha(1)} \cdot \sqrt{d}/\epsilon)$.

Theorem 2. *Suppose that there is a (not necessarily private) polynomial-time approximation algorithm for k-means with approximation ratio κ. For any $\alpha > 0$, there is a one-round (ϵ, δ)-DP $(\kappa(1+\alpha), k^{O_\alpha(1)} \cdot \sqrt{d} \cdot \operatorname{polylog}(nd/\delta)/\epsilon)$-approximation algorithm for k-means in the shuffle DP model. The algorithm runs in time* $\operatorname{poly}(n, d, k, \log(1/\delta))$.

To the best of our knowledge, this is the first k-means algorithm in the one-round shuffle model with (non-trivial) theoretical guarantees. Furthermore, we remark that the above guarantees essentially match those of the central DP algorithm of Ghazi et al. (2020b), while our algorithm works even in the more restrictive shuffle DP model.

The main new ingredient in our framework is the use of a hierarchical data structure called a *net tree* (Har-Peled and Mendel, 2006) to construct a private *coreset*[3] of the input points. The decoder can then simply run any (non-private) approximation algorithm on this private coreset.

The net tree (Har-Peled and Mendel, 2006) is built as follows. Roughly speaking, each input point can be assigned to its "representative" leaf of a net tree; the deeper the tree is, the closer this representative is to the input point. To achieve privacy, noise needs to be added to the number of nodes assigned to each leaf. This creates a natural tension when building such a tree: if the tree has few leaves, the representatives are far from the actual input points whereas if the tree has too many leaves, it suffers a larger error introduced by the noise. Our main technical contribution is an algorithm for constructing a net tree that balances between these two errors, ensuring that neither affects the k-means objective much. We stress that this algorithm works in the non-interactive local model, as each user can encode all possible representatives of their input point without requiring any interaction.

Organization. In Section 2, we provide background on k-means and a description of the tools we need for our algorithm. In Section 3, we describe and analyze net trees, which are sufficient to solve private k-means in low dimensions. In Section 4, we use dimensionality reduction to extend our results to the high-dimensional case. In Section 5, we present some empirical evaluation of our algorithm. Section 6 contains the concluding remarks.

All the missing proofs and the results for the shuffle DP model are in the Supplementary Material.

2. Preliminaries

We use $\|x\|$ to denote the Euclidean norm of a vector x. We use $\mathbb{B}^d(x, r)$ to denote the closed radius-r ball around x, i.e., $\mathbb{B}^d(x, r) = \{y \in \mathbb{R}^d \mid \|x - y\| \leq r\}$. Let $\mathbb{B}^d = \mathbb{B}^d(0, 1)$, the unit ball. For every pair $T, T' \subseteq \mathbb{R}^d$ of sets, we use $d(T, T')$ as a shorthand for $\min_{x \in T, x' \in T'} \|x - x'\|$.

For every $m \in \mathbb{N}$, we use $[m]$ as a shorthand for $\{1, \dots, m\}$. For real numbers $a_1, \dots, a_t$, we use $\operatorname{bottom}_m(a_1, \dots, a_t)$ to denote the sum of the smallest m numbers among $a_1, \dots, a_t$, where $t \geq m$.

We will be working with a generalization of multisets, where each element can have a weight associated with it.

Definition 3 (Weighted Point Set)**.** *A* weighted point set $\mathbf{S}$ *on a domain D is a finite set of tuples $(x, w_\mathbf{S}(x))$ where $x \in D$ and $w_\mathbf{S}(x) \in \mathbb{R}_{\geq 0}$ denotes its weight; each x should not be repeated in $\mathbf{S}$. It is useful to think of $\mathbf{S}$ as the function $w_\mathbf{S} : D \to \mathbb{R}_{\geq 0}$; this extends naturally (as a measure) to any subset $T \subseteq D$ for which we define $w_\mathbf{S}(T) = \sum_{x \in T} w_\mathbf{S}(x)$. We say that $w_\mathbf{S}(D)$ is the* total weight *of* $\mathbf{S}$ *and we abbreviate it as $|\mathbf{S}|$. The* average *of* $\mathbf{S}$ *is defined as $\mu(\mathbf{S}) := \sum_{x \in \mathbb{B}^d} w_\mathbf{S}(x) \cdot x / |\mathbf{S}|$.*

Throughout the paper, we alternatively view a multiset $\mathbf{X}$ as a weighted point set, where $w_\mathbf{X}(x)$ denotes the (possibly fractional) number of times x appears in $\mathbf{X}$.

[1]An algorithm achieves a (κ, t)-*approximation* if its output cost is no more than κ times the optimal cost plus t, i.e., κ is the approximation ratio and t is the additive error (see Section 2.1).

[2]The shuffle model of DP is a middle ground between the central and local models. Please see Appendix C for more details.

[3]See Section 2.1 for the definition of coresets.

For an algorithm $\mathcal{A}$, we use $\mathrm{t}(\mathcal{A})$ to denote its running time.

2.1. k-means and coresets

Let $\mathbf{X} \subseteq \mathbb{B}^d$ be an input multiset and let $n = |\mathbf{X}|$.

Definition 4 (k-means)**.** *The* cost *of a set* $\mathbf{C}$ *of centers with respect to an input multiset* $\mathbf{X} \subseteq \mathbb{B}^d$ *is defined as* $\mathrm{cost}_{\mathbf{X}}(\mathbf{C}) := \sum_{x \in \mathbf{X}} (\min_{c \in \mathbf{C}} \|x - c\|)^2$*. The* k-means problem *asks, given* $\mathbf{X}$ *and* $k \in \mathbb{N}$*, to find* $\mathbf{C}$ *of size* k *that minimizes* $\mathrm{cost}_{\mathbf{X}}(\mathbf{C})$*; the minimum cost is denoted* $\mathrm{OPT}^k_{\mathbf{X}}$*.*

For a weighted point set $\mathbf{S}$ *on* $\mathbb{B}^d$*, we define* $\mathrm{cost}_{\mathbf{S}}(C) := \sum_{x \in \mathbb{B}^d} w_{\mathbf{S}}(x) \cdot (\min_{c \in \mathbf{C}} \|x - c\|)^2$*, and let* $\mathrm{OPT}^k_{\mathbf{S}} := \min_{|\mathbf{C}|=k} \mathrm{cost}_{\mathbf{S}}(\mathbf{C})$*.*

A set $\mathbf{C}$ *of size* k *is a* (κ, t)-approximate solution *for input* $\mathbf{S}$ *if* $\mathrm{cost}_{\mathbf{S}}(\mathbf{C}) \le \kappa \cdot \mathrm{OPT}^k_{\mathbf{S}} + t$*.*

It is also useful to define cost with respect to partitions, i.e., the mapping of points to the centers.

Definition 5 (Partition Cost)**.** *For a given partition* $\phi : \mathbf{S} \to [k]$ *where* $\mathbf{S}$ *is a weighted point set and an ordered set* $\mathbf{C} = (c_1, \dots, c_k)$ *of* k *candidate centers, we define the cost of* ϕ *with respect to* $\mathbf{C}$ *as* $\mathrm{cost}_{\mathbf{S}}(\phi, \mathbf{C}) := \sum_{x \in \mathbb{B}^d} w_{\mathbf{S}}(x) \cdot \|x - c_{\phi(x)}\|^2$*. Furthermore, we define the cost of* ϕ *as* $\mathrm{cost}_{\mathbf{S}}(\phi) := \min_{\mathbf{C} \in (\mathbb{R}^d)^k} \mathrm{cost}_{\mathbf{S}}(\phi, \mathbf{C})$*.*

Note that a minimizer for the term $\min_{\mathbf{C} \in (\mathbb{R}^d)^k} \mathrm{cost}_{\mathbf{S}}(\phi, \mathbf{C})$ above is $c_i = \mu(\mathbf{S}_i)$ where $\mathbf{S}_i$ denotes the weighted point set corresponding to partition i, i.e., $w_{\mathbf{S}_i}(x) = \begin{cases} w_{\mathbf{S}}(x) & \text{if } \phi(x) = i, \\ 0 & \text{otherwise.} \end{cases}$

Coresets. In our algorithm, we will produce a weighted set that is a good "approximation" to the original input set. To quantify how good is the approximation, it is convenient to use the well-studied notion of coresets (see, e.g., Har-Peled and Mazumdar, 2004), which we recall below.

Definition 6 (Coreset)**.** *A weighted point set* $\mathbf{S}'$ *is a* (k, γ, t)-coreset *of a weighted point set* $\mathbf{S}$ *if for every set* $\mathbf{C} \subseteq \mathbb{B}^d$ *of* k *centers, it holds that* $(1 - \gamma) \cdot \mathrm{cost}_{\mathbf{S}}(\mathbf{C}) - t \le \mathrm{cost}_{\mathbf{S}'}(\mathbf{C}) \le (1 + \gamma) \cdot \mathrm{cost}_{\mathbf{S}}(\mathbf{C}) + t$*. When* k *is clear from context, we refer to such an* $\mathbf{S}'$ *as just a* (γ, t)*-coreset of* $\mathbf{X}$*.*

2.2. (Generalized) Monge's Optimal Transport

Another tool that will facilitate our proof is (a generalization of) optimal transport (Monge, 1781). Roughly speaking, in Monge's Optimal Transport problem, we are given two measures on a metric space and we would like to find a map that "transports" the first measure to the second measure, while minimizing the cost, which is often defined in terms of the total mass moved times some function of the distance. This problem can be ill-posed as such a mapping may not exist, e.g., when the total masses are different. (We will often encounter this issue as we will be adding noise for differential privacy, resulting in unequal masses.) To deal with this, we use a slightly generalized version of optimal transport where the unmatched mass is allowed but penalized by the L_1 difference,[4] defined next.

Definition 7 (Generalized L_2^2 Transport Cost)**.** *Let* $\mathbf{S}, \mathbf{S}'$ *be weighted point sets on* $\mathbb{B}^d$*. Their* generalized (L_2^2) Monge's transport cost *of a mapping* $\Psi : \mathbb{B}^d \to \mathbb{B}^d$ *is defined as*

$$\mathrm{mt}(\Psi, \mathbf{S}, \mathbf{S}') := \sum_{y \in \mathbb{B}^d} w_{\mathbf{S}}(y) \cdot \|\Psi(y) - y\|^2 + \sum_{x \in \mathbb{B}^d} |w_{\mathbf{S}}(\Psi^{-1}(x)) - w_{\mathbf{S}'}(x)|. \quad (1)$$

Finally, we define the optimal generalized (L_2^2) Monge's transport cost *from* $\mathbf{S}$ *to* $\mathbf{S}'$ *as*

$$\mathrm{mt}(\mathbf{S}, \mathbf{S}') = \min_{\Psi : \mathbb{B}^d \to \mathbb{B}^d} \mathrm{mt}(\Psi, \mathbf{S}, \mathbf{S}'). \quad (2)$$

We remark that the minimizer Ψ always exists because our weighted sets $\mathbf{S}, \mathbf{S}'$ have finite supports. A crucial property of optimal transport we will use is that if the optimal transport cost between $\mathbf{S}, \mathbf{S}'$ is small relative to the optimal k-means objective, then $\mathbf{S}'$ is a good coreset for $\mathbf{S}$. This is encapsulated in the following lemma.

Lemma 8. *For any* $\xi \in (0, 1)$*, any weighted point sets* $\mathbf{S}, \mathbf{S}'$ *over* $\mathbb{B}^d$*, if* $\mathrm{mt}(\mathbf{S}, \mathbf{S}') \le \frac{\xi}{8(1+2/\xi)} \cdot \mathrm{OPT}^k_{\mathbf{S}} + t$ *for some* $t \ge 0$*, then* $\mathbf{S}'$ *is a* $(\xi, 4(1 + 2/\xi)t)$*-coreset of* $\mathbf{S}$*.*

We also use a slightly more refined bound stated below. It is worth to note that the first inequality below is very explicit as it also gives the mapping for $\mathbf{S}$; we will indeed need such an additional property in our analysis when applying dimensionality reduction.

Lemma 9. *For any* $\xi \in (0, 1]$*, weighted point sets* $\mathbf{S}, \mathbf{S}'$ *over* $\mathbb{B}^d$*,* $\mathbf{C} \in (\mathbb{B}^d)^k$*,* $\phi : \mathbb{B}^d \to [k]$ *and* $\Psi : \mathbb{B}^d \to \mathbb{B}^d$*,*

$$\mathrm{cost}_{\mathbf{S}}(\phi \circ \Psi, \mathbf{C}) \le (1 + \xi) \cdot \mathrm{cost}_{\mathbf{S}'}(\phi, \mathbf{C}) + 4(1 + 1/\xi) \cdot \mathrm{mt}(\Psi, \mathbf{S}, \mathbf{S}'), \text{ and}$$

$$\mathrm{cost}_{\mathbf{S}'}(\mathbf{C}) \le (1 + \xi) \cdot \mathrm{cost}_{\mathbf{S}}(\mathbf{C}) + 4(1 + 1/\xi) \cdot \mathrm{mt}(\Psi, \mathbf{S}, \mathbf{S}').$$

The proofs of both lemmas are deferred to Appendix A.1

2.3. Efficiently Decodable Nets

Let $\mathcal{L} \subseteq \mathbb{B}^d$ be a finite set. Its *covering radius*, denoted $\rho(\mathcal{L})$, is defined as $\max_{x \in \mathbb{B}^d} \min_{y \in \mathcal{L}} \|x - y\|$. Its *packing radius*, denoted $\gamma(\mathcal{L})$, is defined as the largest γ such that the open balls around each point of $\mathcal{L}$ of radius γ are disjoint. We say $\mathcal{L}$ is a (ρ, γ)*-net* if $\rho(\mathcal{L}) \le \rho$ and $\gamma(\mathcal{L}) \ge \gamma$.

Using known results on lattices (Rogers, 1959; Micciancio, 2004; Micciancio and Voulgaris, 2013), Ghazi et al.

[4]Several works have defined similar but slightly different notions (see, e.g., Benamou, 2003; Piccoli and Rossi, 2014).

(2020b) show that there exist nets with $\gamma = \Omega(\rho)$ that are "efficiently decodable", i.e., given any point we can find points in $\mathcal{L}$ close to it in $\exp(O(d))$ time.

Lemma 10 ((Micciancio, 2004; Ghazi et al., 2020b))**.** *For any given $\rho > 0$, there exists a $(\rho, \rho/3)$-net $\mathcal{L}$ such that, for any given point $x \in \mathbb{B}^d$ and any $r \geq \rho$, we can find all points in $\mathbb{B}^d(x, r) \cap \mathcal{L}$ in time $(1 + r/\rho)^{O(d)}$.*

2.4. (Generalized) Histogram and Vector Summation

Finally, we need a few fundamental protocols as building blocks. We first introduce a general aggregation problem.

Definition 11 (Generalized Bucketized Vector Summation)**.** *In the* generalized bucketized vector summation problem*, each user holds a set $Y_i \subseteq Y$ of T buckets and a vector $v_i \in \mathbb{B}^d$. The goal is to determine, for a given $y \in Y$, the* vector sum *of bucket y, which is $v_y := \sum_{\substack{i\in[n]\\ y\in Y_i}} v_i$. An* approximate generalized vector sum oracle *$\tilde{v}$ is said to be η-*accurate *at y if we have $\|v_y - \tilde{v}_y\| < \eta$.*

The setting in this problem—each user holds a d-dimensional vector and contributes to T buckets—generalizes many well-studied problems including (i) the *histogram* problem where $T = d = 1$ and the (scalar) sum of a bucket is the *frequency*, (ii) the *generalized histogram* problem where $d = 1$ but T can be more than one, and (iii) the *bucketized vector summation* problem where $T = 1$ but d can be more than one.

2.5. Privacy Models

We first recall the formal definition of differential privacy (DP). For $\epsilon > 0$ and $\delta \in [0, 1]$, a randomized algorithm $\mathcal{A}$ is (ϵ, δ)-*DP* if for every pair X, X' of inputs that differ on one point and for every subset S of the algorithm's possible outputs, it holds that $\Pr[\mathcal{A}(X) \in S] \leq e^\epsilon \cdot \Pr[\mathcal{A}(X') \in S] + \delta$. When $\delta = 0$, we simply say the algorithm is ϵ-DP.

Let n be the number of users, let $\mathbf{X} = \{x_1, \ldots x_n\}$ and let the input x_i be held by the ith user. An algorithm in the *local DP model* consists of (i) an encoder whose input is the data held by one user and whose output is a sequence of messages and (ii) a decoder, whose input is the concatenation of the messages from all the encoders and whose output is the output of the algorithm. A pair (Enc, Dec) is (ϵ, δ)-DP in the local model if for any input $\mathbf{X} = (x_1, \ldots, x_n)$, the algorithm $\mathcal{A}(\mathbf{X}) := (\mathrm{Enc}(x_1), \ldots, \mathrm{Enc}(x_n))$ is (ϵ, δ)-DP.

For any generalized bucketized vector summation problem Π, we say that the pair (Enc, Dec) is an (η, β)-*accurate* (ϵ, δ)-*DP* algorithm (in a privacy model) if:

- $\mathrm{Enc}^{\Pi}_{(\epsilon,\delta)}$ is an (ϵ, δ)-DP algorithm that takes in the input and produces a randomized output,
- $\mathrm{Dec}^{\Pi}_{(\epsilon,\delta)}$ takes in the randomized output, a target bucket y and produces an estimate vector sum for y,
- For each $y \in Y$, the above oracle is η-accurate at y with probability $1 - \beta$.

For the local DP model, the following generalized histogram guarantee is a simple consequence of the result of Bassily et al. (2020):

Theorem 12 ((Bassily et al., 2020))**.** *There is an $(O(\sqrt{nT^3 \log(|Y|/\beta)}/\epsilon), \beta)$-accurate ϵ-DP algorithm for generalized histogram in the local model. The encoder and the decoder run in time* $\mathrm{poly}(nT/\epsilon, \log |Y|)$.

Using the technique of Bassily et al. (2020) together with that of Duchi et al. (2013), we can obtain the following guarantee for generalized bucketized vector summation:

Lemma 13. *There is an $(O(\sqrt{ndT^3 \log(|Y|/\beta)}/\epsilon), \beta)$-accurate ϵ-DP algorithm for generalized bucketized vector summation in the local model. The encoder and the decoder run in time* $\mathrm{poly}(ndT/\epsilon, \log |Y|)$.

The derivations of the above two bounds are explained in more detail in Appendix B.

3. Net Trees

In this section, we describe *net trees* (Har-Peled and Mendel, 2006), which are data structures that allow us to easily construct coresets of the inputs when the dimension is small. We remark that, although the main structure of net trees we use is similar to that of (Har-Peled and Mendel, 2006), there are several differences, the main one being the construction algorithm which in our case has to be done via (noisy) oracles, leading to considerable challenges.

3.1. Description and Notation

Let $\mathcal{L}_1, \ldots, \mathcal{L}_T \subseteq \mathbb{B}^d$ be a family of efficiently decodable nets, where $\mathcal{L}_i$ has covering radius[5] $\rho_i := 1/2^i$ and packing radius $\gamma_i := \gamma/2^i$. Furthermore, let $\mathcal{L}_0 = \{\mathbf{0}\}, \rho_0 = \gamma_0 = 1$. For convenience, we assume that $\mathcal{L}_0, \mathcal{L}_1, \ldots, \mathcal{L}_T$ are disjoint; this is w.l.o.g. as we can always "shift" each net slightly so that their elements are distinct.

For $i \in \{0, \ldots, T\}$, let $\Psi_i : \mathbb{B}^d \to \mathcal{L}_i$ denote the map from any point to its closest point in $\mathcal{L}_i$ (ties broken arbitrarily).

Complete Net Tree. Given a family of nets $\mathcal{L}_1, \ldots, \mathcal{L}_T$, the *complete net tree* is defined as a tree with $(T+1)$ levels. For $i \in \{0, \ldots, T\}$, the nodes in level i are exactly the elements of $\mathcal{L}_i$. Furthermore, for $i \in [T]$, the parent of node $z \in \mathcal{L}_i$ is $\Psi_{i-1}(z) \in \mathcal{L}_{i-1}$. We use $\mathrm{children}(z)$ to denote the set of all children of z in the complete net tree.

Net Tree. A *net tree* $\mathcal{T}$ is a subtree of the complete net

[5]The 2^i term here can be replaced by λ^i for any $\lambda \in (0, 1)$; we use 2^i to avoid introducing yet another parameter.

tree rooted at $\mathbf{0}$, where each node z is either a leaf or all of $\mathrm{children}(z)$ must be present in the tree $\mathcal{T}$.

We will also use the following additional notation. For each $i \in \{0, \dots, T\}$, let $\mathcal{T}_i$ be the set of all nodes at level i of tree $\mathcal{T}$. Moreover, we use $\mathrm{leaves}(\mathcal{T})$ to denote the set of all leaves of $\mathcal{T}$ and $\mathrm{leaves}(\mathcal{T}_i)$ to denote the set of all leaves at level i. For a node $z \in \mathcal{T}$, we use $\mathrm{level}(z) \in \{0, \dots, T\}$ to denote its level and $\mathrm{children}(z)$ to denote its children.

Representatives. Given a point $x \in \mathbb{B}^d$, its *potential representatives* are the $T + 1$ nodes $\Psi_T(x), \Psi_{T-1}(\Psi_T(x)), \dots, \Psi_0(\cdots(\Psi_T(x))\cdots)$ in the complete net tree. The *representative* of x in a net tree $\mathcal{T}$, denoted by $\Psi_{\mathcal{T}}(x)$, is the unique leaf of $\mathcal{T}$ that is a potential representative of x. Note that $\Psi_{\mathcal{T}} : \mathbb{B}^d \to \mathcal{T}$ induces a partition of points in $\mathbb{B}^d$ based on the leaves representing them.

For a weighted point set $\mathbf{S}$, its frequency at a leaf z of $\mathcal{T}$, denoted by f_z, is defined as the total weights of points in $\mathbf{S}$ whose representative is z, i.e., $f_z = w_{\mathbf{S}}(\Psi_{\mathcal{T}}^{-1}(z))$.

Representative Point Set. Let $\tilde{f}$ be a frequency oracle on domain $\mathcal{L}_1 \cup \cdots \cup \mathcal{L}_t$. The above partitioning scheme yields a simple way to construct a weighted point set from a net tree $\mathcal{T}$. Specifically, the *representative point set* of a tree $\mathcal{T}$ (and frequency oracle $\tilde{f}$), denoted by $\mathbf{S}_{\mathcal{T}}$, is the weighted point set where each leaf $z \in \mathrm{leaves}(\mathcal{T})$ receives a weight of $\tilde{f}_z$. We stress that $\mathbf{S}_{\mathcal{T}}$ depends on the frequency oracle; however we discard it in the notation for readability.

3.2. Basic Properties of Net Trees

Before we proceed, we list a few important properties of net trees that both illustrate their usefulness and will guide our algorithms. The first property is that the potential representation of point x at level i cannot be too far from x:

Lemma 14 (Distance Property)**.** *For any $x \in \mathbb{B}^d$ and $i \in \{0, \dots, T\}$, we have $\|x - \Psi_i(\cdots(\Psi_T(x))\cdots)\| \le 2^{1-i}$.*

Proof. Using the triangle inequality, we can bound $\|x - \Psi_i(\cdots(\Psi_T(x))\cdots)\|$ above by $\|\Psi_T(x) - x\| + \sum_{j=i}^{T-1} \|\Psi_j(\cdots(\Psi_T(x))\cdots) - \Psi_{j+1}(\cdots(\Psi_T(x))\cdots)\|$ Since the covering radius of $\mathcal{L}_j$ is 2^{-j}, this latter term is at most $2^{-T} + \sum_{j=i}^{T-1} 2^{-j} \le 2^{1-i}$ as desired. □

Second, we show that the number of children is small.

Lemma 15 (Branching Factor)**.** *For any $z \in \mathcal{L}_0 \cup \cdots \cup \mathcal{L}_{T-1}$, we have $|\mathrm{children}(z)| \le B := (1 + 2/\gamma)^d$.*

Proof. Let $i = \mathrm{level}(z)$. Since each $z' \in \mathrm{children}(z)$ has z as its closest point in $\mathcal{L}_i$, we have $\|z' - z\| \le 2^{-i}$. Furthermore, since $\mathrm{children}(z) \subseteq \mathcal{L}_{i+1}$, $\mathrm{children}(z)$ form a $(\gamma \cdot 2^{-i-1})$-packing. As a result, a standard volume argument implies that

$$|\mathrm{children}(z)| \le \left(1 + \frac{2^{-i}}{\gamma \cdot 2^{-i-1}}\right)^d = (1 + 2/\gamma)^d. \quad \square$$

We note that when applying dimensionality reduction in the next section, we will have[6] $d = O(\log k)$, meaning $B = k^{O(1)}$.

The last property is an upper bound on the optimal transport cost from a given weighted point set to a representative point set created via a net tree $\mathcal{T}$. Recall from Lemma 8 that this implies a certain coreset guarantee for the constructed representative point set, a fact we will repeatedly use in the subsequent steps of the proof. The bound on the transport cost is stated in its general form below.

Lemma 16. *For a weighted point set $\mathbf{S}$ and a net tree $\mathcal{T}$, let f_z denote the frequency of $\mathbf{S}$ on a leaf $z \in \mathrm{leaves}(\mathcal{T})$, i.e., $f_z = w_{\mathbf{S}}(\Psi_{\mathcal{T}}^{-1}(z))$. Let $S_{\mathcal{T}}$ denote the representative point set constructed from $\mathcal{T}$ and frequency oracle $\tilde{f}$. Then,*

$$\mathrm{mt}(\Psi_{\mathcal{T}}, \mathbf{S}, \mathbf{S}_{\mathcal{T}}) \le \sum_{z \in \mathrm{leaves}(\mathcal{T})} \left(f_z \cdot (4\rho^2_{\mathrm{level}(z)}) + |f_z - \tilde{f}_z|\right).$$

Proof. Recall by definition that $\mathrm{mt}(\Psi_{\mathcal{T}}, \mathbf{S}, \mathbf{S}_{\mathcal{T}})$ is equal to $\sum_{z \in \mathrm{leaves}(\mathcal{T})} |w_{\mathbf{S}}(\Psi_{\mathcal{T}}^{-1}(z)) - w_{\mathbf{S}_{\mathcal{T}}}(z)| + \sum_{y \in \mathbb{B}^d} w_{\mathbf{S}}(y) \cdot \|\Psi_{\mathcal{T}}(y) - y\|^2$. By construction of $\mathbf{S}_{\mathcal{T}}$, the first term is equal to $\sum_{z \in \mathrm{leaves}(\mathcal{T})} |f_z - \tilde{f}_z|$. Using Lemma 14, the second term is bounded above by

$$\sum_{z \in \mathrm{leaves}(\mathcal{T})} \sum_{y \in \Psi_{\mathcal{T}}^{-1}(z)} w_{\mathbf{S}}(y) \cdot (2\rho_{\mathrm{level}(z)})^2$$
$$= \sum_{z \in \mathrm{leaves}(\mathcal{T})} f_z \cdot (4\rho^2_{\mathrm{level}(z)}).$$

Combining the two bounds completes the proof. □

3.3. Building the Net Tree

Although we have defined net trees and shown several of their main properties, we have not addressed how a net tree should be constructed from a given (approximate) frequency oracle (on $\mathcal{L}_0 \cup \cdots \cup \mathcal{L}_T$). Lemma 16 captures the tension arising when constructing the tree: if we decide to include too many nodes in the tree, then all of the nodes contribute to the additive error, i.e., the second term in the bound. On the other hand, if we include too few nodes, then many leaves will be at a small level, resulting in a large first term. In this section, we give a tree building algorithm that balances these two errors, and prove its guarantees.

We assume that each approximate frequency $\tilde{f}_z$ is non-negative.[7] The tree construction (Algorithm 1) itself is

[6]Since we will pick $\gamma > 0$ to be a constant, we hide its dependency in asymptotic notations throughout this section.

[7]This is w.l.o.g. as we can always take $\max(\tilde{f}_z, 0)$ instead.

simple; we build the tree in a top down manner, starting with small levels and moving on to higher levels. At each level, we compute a *threshold* τ on the number of nodes to expand. We then only expand τ nodes with maximum approximate frequencies. The algorithm for computing this threshold τ (COMPUTETHRESHOLD) and its properties will be stated next.

Algorithm 1 Building the Net Tree.

Oracle Access: Frequency oracle $\tilde{f}$ on $\mathcal{L}_0 \cup \cdots \cup \mathcal{L}_T$
Parameters: $k, a, \Gamma \in \mathbb{N}$

```
1: procedure BUILDTREE$^{\tilde{f}}$
2:     $\mathcal{T} \leftarrow$ root node $z = \mathbf{0}$ at level 0
3:     for $i = 0, \dots, T-1$
4:         $z_i^1, \dots, z_i^{m_i} \leftarrow$ level-$i$ nodes sorted in non-decreasing order of $\tilde{f}_z$
5:         $\tau_i \leftarrow$ COMPUTETHRESHOLD$^{\tilde{f}}_{k,a,\Gamma}(z_i^1, \dots, z_i^{m_i})$
6:         for $j = 0, \dots, \tau_i - 1$
7:             Add children$(z_i^{m_i - j})$ to $\mathcal{T}$
8:     return $\mathcal{T}$
```

Computing the Threshold. Before we describe the threshold computation algorithm, we provide some intuition. Recall Lemma 16, which gives an upper bound on the optimal transport cost and Lemma 8, which relates this quantity to the quality of the coreset. At a high level, these two lemmas allow us to stop branching as soon as the bound in Lemma 16 becomes much smaller than $\mathrm{OPT}^k_{\mathbf{S}}$. Of course, the glaring issue in doing so is that we do *not* know $\mathrm{OPT}^k_{\mathbf{S}}$! It turns out however that we can give a *lower bound* on $\mathrm{OPT}^k_{\mathbf{S}}$ based on the tree constructed so far. To state this lower bound formally, we first state a general lower bound on the k-means cost, without any relation to the tree.

Lemma 17. *Let $a, b, k \in \mathbb{N}$ and $r \in \mathbb{R}_{\geq 0}$. Let $\mathbf{S}$ be a weighted point set, and $T_1, \dots, T_{ka+b} \subseteq \mathbb{R}^d$ be any $ka+b$ disjoint sets such that for any point $c \in \mathbb{R}^d$ it holds that $|\{i \in [ka+b] \mid \mathbb{B}^d(c, r) \cap T_i \neq \emptyset\}| \leq a$. Then, $\mathrm{OPT}^k_{\mathbf{S}} \geq r^2 \cdot \mathrm{bottom}_b(w_{\mathbf{S}}(T_1), \dots, w_{\mathbf{S}}(T_{ka+b}))$.*

Proof. Consider any set $\mathbf{C}$ of k candidate centers. From the assumption, there must be at least b subsets T_i's such that $d(\mathbf{C}, T_i) \geq r$; for such a subset, its contribution to the k-means objective is at least $r^2 \cdot w_{\mathbf{S}}(T_i)$. As a result, the total k-means objective is at least $r^2 \cdot \mathrm{bottom}_b(w_{\mathbf{S}}(T_1), \dots, w_{\mathbf{S}}(T_{ka+b}))$. □

This lets us prove a lower bound on the k-means objective for net trees:

Corollary 18. *For any $\theta > 0$, let $r = \theta \cdot 2^{-i}$ and $a = \lceil (1 + (2+\theta)/\gamma)^d \rceil$. Let $b \in \mathbb{N}$. Suppose that there exist $(ka+b)$ level-i nodes $\tilde{z}^1, \dots, \tilde{z}^{ka+b}$ in a net tree $\mathcal{T}$. Furthermore, let $\mathbf{S}$ be any multiset and f the frequency of $\mathbf{S}$. Then, we have $\mathrm{OPT}^k_{\mathbf{S} \cap \Psi^{-1}_{\mathcal{T}}(\{\tilde{z}^1, \dots, \tilde{z}^{ka+b}\})} \geq r^2 \cdot \mathrm{bottom}_b(f_{\tilde{z}^1}, \dots, f_{\tilde{z}^{ka+b}})$.*

Proof. Consider any center $c \in \mathbb{R}^d$. Recall from Lemma 14 that $\Psi^{-1}_{\mathcal{T}}(\tilde{z}^j) \subseteq \mathbb{B}^d(\tilde{z}^j, 2^{1-i})$. In other words, if $\mathbb{B}^d(c, r) \cap (S \cap \Psi^{-1}_{\mathcal{T}}(\tilde{z}^j)) \neq \emptyset$, we must have

$$\|c - \tilde{z}^j\| \leq r + 2^{1-i} = (2+\theta)2^{-i}, \quad (3)$$

implying that $\mathbb{B}^d(\tilde{z}^j, \gamma \cdot 2^{-i}) \subseteq \mathbb{B}^d(c, (2+\theta)2^{-i} + \gamma \cdot 2^{-i})$.

Furthermore, since $\tilde{z}^1, \dots, \tilde{z}^{ka+b} \subseteq \mathcal{L}_i$ form a $(\gamma \cdot 2^{-i})$-packing, the balls $\mathbb{B}^d(\tilde{z}^j, \gamma \cdot 2^{-i})$ are disjoint. This means that any point c satisfies (3) for at most

$$\left(1 + \frac{(2+\theta) \cdot 2^{-i}}{\gamma \cdot 2^{-i}}\right)^d \leq a.$$

many j's. Applying Lemma 17 completes the proof. □

Thus, we can use $r^2 \cdot \mathrm{bottom}_b(f_{\tilde{z}^1}, \dots, f_{\tilde{z}^{ka+b}})$ as a "running lower bound" on $\mathrm{OPT}^k_{\mathbf{S}}$. Still, we have to be careful as the additive error introduced will add up over all the levels of the tree; this can be an issue since we will select the number of levels to be as large as $\Theta(\log n)$. To overcome this, we make sure that the additive error introduced at each level is only "charged" to the optimum of the weighted point set corresponding to *leaves* in that level. This ensures that there is no double counting in the error.

Below we formalize our cutoff threshold computation algorithm and prove its main property, which will be used later to provide the guarantees on the tree construction.

Algorithm 2 Computing the Threshold.

Oracle Access: Frequency oracle $\tilde{f}$ on $\mathcal{L}_0 \cup \cdots \cup \mathcal{L}_T$
Parameters: $k, a, \Gamma \in \mathbb{N}$
Inputs: Nodes $z^1, \dots, z^m$ from the same level of a net tree

```
1: procedure COMPUTETHRESHOLD$^{\tilde{f}}_{k,a,\Gamma}$
2:     for $j \in [\min\{\Gamma, \lfloor m/ka \rfloor\}]$
3:         if $\sum_{i=1}^{m-(j-1)ka} f_{z^i} \leq 2 \cdot \sum_{i=1}^{m-jka} f_{z^i}$
4:             return $(j-1)ka$
5:     return $\min\{m, \Gamma ka\}$
```

Lemma 19. *Suppose that $\tilde{f}$ is η-accurate at each of $z^1, \dots, z^m$ and suppose $\tilde{f}_{z^1} \leq \cdots \leq \tilde{f}_{z^m}$. Then,* COMPUTETHRESHOLD$^{\tilde{f}}_{a,\Gamma}(z^1, \dots, z^m)$ *outputs τ satisfying*

$$\sum_{i=1}^{m-\tau} \tilde{f}_{z^i} \leq 2 \cdot \left(\sum_{i=1}^{m-\tau-ka} \tilde{f}_{z^i} \right) + \frac{n + m\eta}{2^\Gamma}.$$

Proof. If the algorithm returns on line 4, then the inequality trivially holds due to the check on line 3 before. Furthermore, the inequality also trivially holds if $\tau = m$, as the left hand side is simply zero. As a result, we may assume that the algorithm returns $\tau = \Gamma ka < m$. This means the condition on line 3 does *not* hold for all $j \in [\Gamma]$. From this, we can conclude that

$$\tilde{f}_{z^1} + \cdots + \tilde{f}_{z^m} > 2\left(\tilde{f}_{z^1} + \cdots + \tilde{f}_{z^{m-ka}}\right) > \cdots > 2^{\Gamma}\left(\tilde{f}_{z^1} + \cdots + \tilde{f}_{z^{m-\Gamma ka}}\right).$$

Finally, since the frequency oracle is η-accurate at each of $z^1, \ldots, z^m$, we have

$$\tilde{f}_{z^1} + \cdots + \tilde{f}_{z^m} \leq m\eta + f_{z^1} + \cdots + f_{z^m} \leq m\eta + n,$$

where the latter inequality follows since each input point is mapped to at most one node at level i. Combining the above two inequalities yields the desired bound. □

We remark that $\tilde{f}_{z^1} + \cdots + \tilde{f}_{z^{m-\tau}}$ is the approximate frequency of points that will be mapped to the leaves at this level, which in turn gives the upper bound on the transport cost (in Lemma 16); whereas $\tilde{f}_{z^1} + \cdots + \tilde{f}_{z^{m-\tau-ka}}$ indeed governs the lower bound on the optimum k-means objective in Corollary 18. Intuitively, Lemma 19 thus allows us to "charge" the transport cost at each level to the lower bound on the optimum of the leaves at that level as desired. These arguments will be formalized in the next subsection.

Putting it Together. The main property of a net tree $\mathcal{T}$ output by our tree construction algorithm is that its representative point set is a good coreset of the underlying input. This, plus some additional properties, are stated below.

Theorem 20. *Let $\xi \in (0, 1)$. Suppose that the frequency oracle $\tilde{f}$ is η-accurate on every element queried by the algorithm. Let $\mathcal{T}$ be the tree output by Algorithm 1 where $\Gamma = \lceil \log n \rceil, T = \lceil 0.5 \log n \rceil, \theta = 8\sqrt{\frac{1+2/\xi}{\xi}}$, and a be as in Corollary 18. Let $N_T = 2^{O_\xi(d)} \cdot k \cdot (\log^2 n)$. Then,*

- *The number of nodes in $\mathcal{T}$ is N_T. Furthermore, this holds regardless of the frequency oracle accuracy.*
- $\mathrm{mt}(\Psi_{\mathcal{T}}, \mathbf{S}, \mathbf{S}_{\mathcal{T}}) \leq \frac{\xi}{8(1+2/\xi)} \cdot \mathrm{OPT}^k_{\mathbf{S}} + \eta \cdot O(N_T)$.
- *$S_{\mathcal{T}}$ is a $(\xi, \eta \cdot O(N_T))$-coreset of S.*

Moreover, the tree construction algorithm runs in time $\mathrm{poly}(N_T)$ multiplied by the time to query $\tilde{f}$.

4. From Low to High Dimension

The net tree-based algorithm can already be applied to give an approximation algorithm for k-means, albeit with an additive error of $2^{O_\alpha(d)} \cdot k \cdot (\log^2 n) \cdot \eta$. The exponential dependency on d is undesirable. In this section, we show how to eliminate this dependency via random projections, following the approach of Ghazi et al. (2020b) for private clustering in the central DP model. Specifically, the breakthrough work of Makarychev et al. (2019) allows one to randomly project to $d = O(\log k)$ dimensions while maintaining the objective for any given partition.

Theorem 21 ((Makarychev et al., 2019))**.** *For every $0 < \tilde{\beta}, \tilde{\alpha} < 1$ and $k \in \mathbb{N}$, there exists $d' = O_{\tilde{\alpha}}(\log(k/\beta))$ such that the following holds. Let P be a random d'-dimensional subspace of $\mathbb{R}^d$ and Π_P denote the projection from $\mathbb{R}^d$ to P. With probability $1 - \tilde{\beta}$, we have the following for all partitions $\phi : \mathbb{B}^{d'} \to [k]$:*

$$\frac{1}{1+\tilde{\alpha}} \leq \frac{d \cdot \mathrm{cost}_{\Pi_P(\mathbf{S})}(\phi)}{d' \cdot \mathrm{cost}_{\mathbf{S}}(\phi \circ \Pi_P)} \leq 1 + \tilde{\alpha}.$$

Our encoding algorithm first projects x to $\tilde{x}$ in a given subspace P and appropriately scales it to x'. It then computes all potential representatives (corresponding to the complete net tree of the nets $\mathcal{L}_1, \ldots, \mathcal{L}_T$), and then encodes these representatives in the generalized histogram and the generalized bucketized vector summation encoders, the latter with the input vector x. This is presented more formally below in Algorithm 3. (Note that we treat x'_i as a vector in $\mathbb{B}^{d'}$ directly; this is w.l.o.g. as we can rotate to make the basis of P into the first d' standard basis vectors.)

Algorithm 3 Encoding Algorithm for k-means.

Input: Point $x_i \in \mathbb{B}^d$ of user i.
Parameters: Privacy parameters ϵ, δ, nets $\mathcal{L}_1, \ldots, \mathcal{L}_T$, d'-dimensional subspace P, and $\Lambda > 0$.
Subroutines: Encoders $\mathrm{Enc}^{\mathrm{hist}}, \mathrm{Enc}^{\mathrm{vec}}$ for generalized histogram and bucketized vector summation.

```
1:  procedure KMEANSENCODER_{ε,δ,Λ,P,L_1,...,L_T}($x_i$)
2:      $\tilde{x}_i \leftarrow \Pi_P(x_i)$
3:      if $\|\tilde{x}_i\| \leq 1/\Lambda$
4:          $x'_i = \Lambda\tilde{x}$
5:      else
6:          $x'_i = \mathbf{0}$
7:      $y^T_i \leftarrow$ Closest point to $x'_i$ in $\mathcal{L}_T$
8:      for $j = T-1, \ldots, 1$
9:          $y^j_i \leftarrow$ Closest point to $y^{j+1}_i$ in $\mathcal{L}_j$
10:     $e^h_i \leftarrow \mathrm{Enc}^{\mathrm{hist}}_{(\epsilon/2,\delta/2)}(\{y^1_i, \ldots, y^T_i\})$
11:     $e^v_i \leftarrow \mathrm{Enc}^{\mathrm{vec}}_{(\epsilon/2,\delta/2)}(\{y^1_i, \ldots, y^T_i\}, x_i)$
12:     return $(e^h_i, e^v_i)$
```

To decode, we first use the encoded histogram to build a frequency oracle, from which we construct a net tree $\mathcal{T}$ using the algorithm in Section 3. We then run any approximation algorithm $\mathcal{A}$ for k-means on the representative set of $\mathcal{T}$. The output of $\mathcal{A}$ gives a partition of the leaves of $\mathcal{T}$ according to which centers are the closest. We then use the vector summation oracle on these partitions to determine the k centers in the original (high-dimensional) space. A pseudo-code of this algorithm is given below as Algorithm 4. We stress here that the approximation algorithm $\mathcal{A}$ need *not* be private.

A generic guarantee of our algorithm is stated next. As we will explain below, plugging known histogram/vector summation algorithms immediately yields our main results.

Algorithm 4 Decoding Algorithm for k-means.

Input: Encoded inputs $e^h_1, e^v_1, \dots, e^h_n, e^v_n$.
Parameters: Privacy parameters ϵ, δ, approximation algorithm $\mathcal{A}$ for k-means.
Subroutines: Decoders $\mathrm{Dec}^{\mathrm{hist}}, \mathrm{Dec}^{\mathrm{vec}}$ for generalized histogram and bucketized vector summation.

```
1:  procedure $\textsc{kMeansDecoder}_{\epsilon,\delta,\mathcal{A}}(e^h_1, e^v_1, \dots, e^h_n, e^v_n)$
2:      $\tilde{f} \leftarrow$ frequency oracle from $\mathrm{Dec}^{\mathrm{hist}}_{(\epsilon/2,\delta/2)}(e^h_1, \dots, e^h_n)$
3:      $\tilde{v} \leftarrow$ vector sum oracle from $\mathrm{Dec}^{\mathrm{vec}}_{(\epsilon/2,\delta/2)}(e^v_1, \dots, e^v_n)$
4:      $\mathcal{T} \leftarrow \textsc{BuildTree}^{\tilde{f}}$
5:      $\{c'_1, \dots, c'_k\} \leftarrow \mathcal{A}(\mathbf{S}_{\mathcal{T}})$
6:      $\phi_* \leftarrow$ mapping leaves$(\mathcal{T}) \to [k]$ where $\phi_*(z) = j$
          iff $c'_j$ is closest to $z$ (with ties broken arbitrarily)
7:      for $j = 1, \dots, k$
8:          $\tilde{v}^j \leftarrow \mathbf{0}$
9:          $\tilde{n}^j \leftarrow 0$
10:         for $z \in \phi_*^{-1}(j)$
11:             $\tilde{v}^j \leftarrow \tilde{v}^j + \tilde{v}_z$
12:             $\tilde{n}^j \leftarrow \tilde{n}^j + \tilde{f}_z$
13:         $\tilde{c}^j = \tilde{v}^j / \max\{1, \tilde{n}^j\}$
14:         if $\|\tilde{c}^j\| \le 1$
15:             $c_j \leftarrow \tilde{c}_j$
16:         else
17:             $c_j \leftarrow \tilde{c}_j / \|\tilde{c}_j\|$
18:     return $\{c_1, \dots, c_k\}$
```

Theorem 22. $\textsc{kMeansEncoder}_{\epsilon,\delta}$ *is* (ϵ, δ)*-DP. Furthermore, suppose that the following hold:*

- *$\mathcal{A}$ is a κ-approximation algorithm for k-means.*
- *d' is as in Theorem 21 with $\tilde{\beta} = 0.1\beta, \tilde{\alpha} = 0.1\alpha$, and $\Lambda = \sqrt{\frac{0.01}{\log(n/\beta)} \cdot \frac{d}{d'}}$.*
- *P is a random d'-dimensional subspace of $\mathbb{R}^d$.*
- *The parameters of* $\textsc{BuildTree}$ *are as in Theorem 20 with $\xi = 0.1\alpha$, and let $N_T = 2^{O_\alpha(d')} \cdot k \cdot (\log^2 n)$ be an upper bound on the number of nodes of $\mathcal{T}$.*
- $(\mathrm{Enc}^{\mathrm{hist}}_{(\epsilon/2,\delta/2)}, \mathrm{Dec}^{\mathrm{hist}}_{(\epsilon/2,\delta/2)})$ *is $(\eta, 0.1\beta/N_T)$-accurate for generalized histogram.*
- $(\mathrm{Enc}^{\mathrm{vec}}_{(\epsilon/2,\delta/2)}, \mathrm{Dec}^{\mathrm{vec}}_{(\epsilon/2,\delta/2)})$ *is $(\tilde{\eta}, 0.1\beta/N_T)$-accurate for generalized bucketized vector summation.*

Then, with probability $1 - \beta$, $\textsc{kMeansDecoder}$ *outputs a $\left(\kappa(1+\alpha), k^{O_\alpha(1)}(\log^2 n)\left(\log(n/\beta) \cdot \eta + \tilde{\eta}\right)\right)$-approximate solution for k-means. Moreover, the encoder runs in time* $\mathrm{poly}(ndk^{O_\alpha(1)}, \mathrm{t}(\mathrm{Enc}^{\mathrm{hist}}), \mathrm{t}(\mathrm{Enc}^{\mathrm{vec}}))$*, and the decoder runs in time* $\mathrm{poly}(ndk^{O_\alpha(1)}, \mathrm{t}(\mathcal{A}), \mathrm{t}(\mathrm{Dec}^{\mathrm{hist}}), \mathrm{t}(\mathrm{Dec}^{\mathrm{vec}}))$.

Theorem 22 allows us to easily derive approximation algorithms for k-means in different distributed models of DP, by simply plugging in the known generalized histogram/vector summation guarantees.

4.1. Approximation Algorithm for Local DP

Next we consider the local DP model and prove Theorem 1.

Proof of Theorem 1. Let $\beta = 0.1$. From Theorem 12, there is an $(\eta, 0.1\beta/N_T)$-accurate 0.5ϵ-DP algorithm for generalized histogram with

$$\eta = O(\sqrt{nT^3 \log(N_T|\mathcal{L}_1 \cup \cdots \cup \mathcal{L}_T|/\beta)}/\epsilon).$$

Since we set $T = O(\log n)$ (in Theorem 20), $N_T = k^{O_\alpha(1)} \cdot \mathrm{poly}\log n$ by our choice of parameters, and since $|\mathcal{L}_1 \cup \cdots \cup \mathcal{L}_T| \le \exp(O(Td'))$ by a volume argument, we get $\eta = O(\sqrt{n}\mathrm{poly}\log(n)/\epsilon)$.

Similarly, from Lemma 13, there is an $(\tilde{\eta}, 0.1\beta/N_T)$-accurate 0.5ϵ-DP algorithm for generalized histogram with

$$\tilde{\eta} = O(\sqrt{ndT^3 \log(dN_T|\mathcal{L}_1 \cup \cdots \cup \mathcal{L}_T|/\beta)}/\epsilon),$$

which as before yields $\tilde{\eta} = O(\sqrt{nd} \cdot \mathrm{polylog}(n)/\epsilon)$. Plugging this into Theorem 22, we indeed arrive at a one-round ϵ-local DP $(\kappa(1+\alpha), k^{O_\alpha(1)} \cdot \sqrt{nd} \cdot \mathrm{polylog}(n)/\epsilon)$-approximation algorithm for k-means (with failure probability 0.1). It is easy to verify that the encoder and the decoder run in time $\mathrm{poly}(n, d, k^{O_\alpha(1)})$. □

5. Experiments

Implementation Details. We modify our algorithm in several places to make it more practical. First, instead of using nets we use locality-sensitive hashing (LSH). Specifically, given LSH $g_1, \dots, g_T$, the level-i representation of x is now $z_i = (g_1(x), \dots, g_T(x))$. In this sense, our tree bears a strong resemblance to the so-called *LSH forests* (Bawa et al., 2005; Andoni et al., 2017). As for the specific family of hashes, we use SimHash (Charikar, 2002) in which a random vector v_i is picked and $g_i(x)$ is the sign of $\langle v_i, x \rangle$. Since LSH can already be viewed as a dimensionality reduction method, we skip the random projection at the beginning of the algorithm. Consequently, we also directly compute the approximate centers of all the nodes in the tree and then use a non-private algorithm (in particular, k-means++ (Arthur and Vassilvitskii, 2007)) to compute the k centers on this privatized dataset. Details on the choice of parameters can be found in Appendix D.

Dataset Generation. We use mixtures of Gaussians, which are generated as follows. For a separation parameter r and the number k of clusters, first pick k centers uniformly at random from the sphere of radius slightly less than one (i.e., $1 - \Theta(r)$). Then, for each center, we create n/k points by adding to the center a Gaussian-distributed vector whose expected norm is $1/r$. Finally, we project any point that is outside of the unit ball back into it. Note that we run our algorithm on this dataset using the same value of k.

Although these datasets are relatively easy, we would like to stress that we are not aware of any prior experiments or

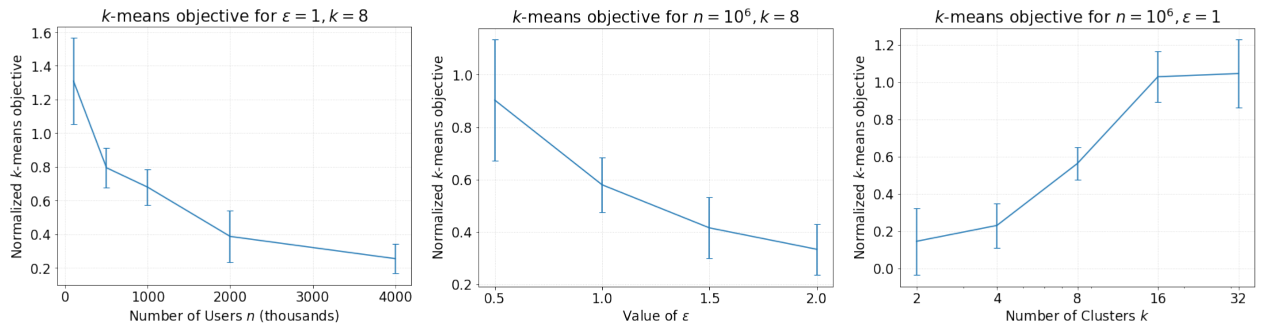

Figure 1. Normalized k-means objective of the output clusters for varying n, ϵ or k for $d = 100, r = 100$. Each set of parameters is run 10 times; the average and the standard deviation of the normalized k-means objectives are included.

implementations of non-interative local DP algorithms.[8][9] Furthermore, we point out that even experiments in the central model of Balcan et al. (2017), which uses almost the same data generation process (including separation parameters) as ours, suggest that the datasets are already challenging for the much more relaxed central DP model.

Results Summary. Figure 1 presents the *normalized* k-means objective (i.e., the k-means objective divided by n) as n, ϵ or k vary. Note that the "trivial" clustering where we simply use the origin as the center has normalized objective roughly equal to one. Our clustering algorithm significantly improves upon these when n is sufficiently large. More importantly, as either n or ϵ increases, the normalized objective decreases, as predicted by theory. Finally, when the number of clusters k becomes larger, our algorithm suffers larger errors, once again agreeing with theory.

6. Conclusions and Open Questions

We give private approximation algorithms for k-means clustering whose ratios are essentially the same as those of non-private algorithms in both the (one-round) local DP and the (one-round) shuffle DP models. An interesting open question is to extend this result to other clustering objectives, such as k-median. While the net trees can also be applied to k-median with little change, the techniques we use to handle high dimensions do not directly carry over. This is due to the fact that, in k-means, it is simple to find a center of a cluster in one round, by finding the average of all the points. However, finding a cluster center in k-median (to the best of our knowledge) requires solving a linear program and it seems challenging to do so non-interactively.

[8]Very recently, (Xia et al., 2020) have reported experimental results for k-means in the local DP model. However, their algorithm, which is based on Lloyd's iteration, requires interaction.

[9]We remark that we also tried some "naive" baseline algorithms, such as noising each point using Gaussian/Laplace noise and running k-means++ on this noisy dataset. However, they do not produce any meaningful clustering even for n as large as 10^6.

Even for the k-means problem itself, there are still several interesting questions. First, as mentioned in the Introduction, our algorithm relies crucially on public randomness in the dimensionality reduction step, where every user has to project onto the same random subspace P. Can k-means be approximated almost optimally via a one-round local DP algorithm that uses only private randomness?

Another direction is to tighten the additive error. In the central model, this has recently been investigated in Chaturvedi et al. (2021); Jones et al. (2021), who achieved essentially tight additive errors in terms of k (and in certain regimes d, n). It is a natural question to determine such a tight dependency in the (non-interactive) local model too.

References

J. M. Abowd. The US Census Bureau adopts differential privacy. In *KDD*, pages 2867–2867, 2018.

A. Aggarwal, A. Deshpande, and R. Kannan. Adaptive sampling for k-means clustering. In *APPROX*, volume 5687, pages 15–28, 2009.

S. Ahmadian, A. Norouzi-Fard, O. Svensson, and J. Ward. Better guarantees for k-means and Euclidean k-median by primal-dual algorithms. *SIAM J. Comput.*, 49(4), 2020.

D. Aloise, A. Deshpande, P. Hansen, and P. Popat. NP-hardness of Euclidean sum-of-squares clustering. *Machine Learning*, 75(2):245–249, 2009.

A. Andoni, I. P. Razenshteyn, and N. S. Nosatzki. LSH forest: Practical algorithms made theoretical. In *SODA*, pages 67–78, 2017.

Apple Differential Privacy Team. Learning with privacy at scale. *Apple Machine Learning Journal*, 2017.

D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *SODA*, pages 1027–1035, 2007.

M. Balcan, T. Dick, Y. Liang, W. Mou, and H. Zhang. Differentially private clustering in high-dimensional Euclidean spaces. In *ICML*, pages 322–331, 2017.

V. Balcer and A. Cheu. Separating local & shuffled differential privacy via histograms. In *ITC*, pages 1:1–1:14, 2020.

V. Balcer, A. Cheu, M. Joseph, and J. Mao. Connecting robust shuffle privacy and pan-privacy. In *SODA*, pages 2384–2403, 2021.

B. Balle, J. Bell, A. Gascón, and K. Nissim. The privacy blanket of the shuffle model. In *CRYPTO*, pages 638–667, 2019.

B. Balle, J. Bell, A. Gascón, and K. Nissim. Private summation in the multi-message shuffle model. In *CCS*, pages 657–676, 2020.

R. Bassily, K. Nissim, U. Stemmer, and A. Thakurta. Practical locally private heavy hitters. *JMLR*, 21:16:1–16:42, 2020.

M. Bawa, T. Condie, and P. Ganesan. LSH forest: self-tuning indexes for similarity search. In *WWW*, pages 651–660, 2005.

J.-D. Benamou. Numerical resolution of an "unbalanced" mass transport problem. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 37(5):851–868, 2003.

A. Bittau, Ú. Erlingsson, P. Maniatis, I. Mironov, A. Raghunathan, D. Lie, M. Rudominer, U. Kode, J. Tinnés, and B. Seefeld. Prochlo: Strong privacy for analytics in the crowd. In *SOSP*, pages 441–459, 2017.

A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the sulq framework. In *PODS*, pages 128–138, 2005.

C. Canonne, G. Kamath, and T. Steinke. The discrete Gaussian for differential privacy. In *NeurIPS*, 2020.

M. Charikar. Similarity estimation techniques from rounding algorithms. In *STOC*, pages 380–388, 2002.

A. Chaturvedi, H. Nguyen, and E. Xu. Differentially private k-means clustering via exponential mechanism and max cover. *AAAI*, 2021.

L. Chen, B. Ghazi, R. Kumar, and P. Manurangsi. On distributed differential privacy and counting distinct elements. In *ITCS*, 2021.

A. Cheu, A. D. Smith, J. Ullman, D. Zeber, and M. Zhilyaev. Distributed differential privacy via shuffling. In *EUROCRYPT*, pages 375–403, 2019.

B. Ding, J. Kulkarni, and S. Yekhanin. Collecting telemetry data privately. In *NIPS*, pages 3571–3580, 2017.

J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Local privacy and statistical minimax rates. In *FOCS*, pages 429–438, 2013.

C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006a.

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

Ú. Erlingsson, V. Pihur, and A. Korolova. RAPPOR: Randomized aggregatable privacy-preserving ordinal response. In *CCS*, pages 1054–1067, 2014.

Ú. Erlingsson, V. Feldman, I. Mironov, A. Raghunathan, K. Talwar, and A. Thakurta. Amplification by shuffling: From local to central differential privacy via anonymity. In *SODA*, pages 2468–2479, 2019.

D. Feldman, A. Fiat, H. Kaplan, and K. Nissim. Private coresets. In *STOC*, pages 361–370, 2009.

D. Feldman, C. Xiang, R. Zhu, and D. Rus. Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks. In *IPSN*, pages 3–16, 2017.

B. Ghazi, N. Golowich, R. Kumar, P. Manurangsi, R. Pagh, and A. Velingker. Pure differentially private summation from anonymous messages. In *ITC*, 2020a.

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

B. Ghazi, R. Kumar, P. Manurangsi, and R. Pagh. Private counting from anonymous messages: Near-optimal accuracy with vanishing communication overhead. In *ICML*, pages 3505–3514, 2020c.

B. Ghazi, P. Manurangsi, R. Pagh, and A. Velingker. Private aggregation from fewer anonymous messages. In *EUROCRYPT*, pages 798–827, 2020d.

B. Ghazi, N. Golowich, R. Kumar, R. Pagh, and A. Velingker. On the power of multiple anonymous messages. In *EUROCRYPT*, 2021a.

B. Ghazi, R. Kumar, P. Manurangsi, R. Pagh, and A. Sinha. Differentially private aggregation in the shuffle model: Almost central accuracy in almost a single message. In *ICML*, 2021b.

A. Greenberg. Apple's "differential privacy" is about collecting your data – but not your data. *Wired, June*, 13, 2016.

A. Gupta, K. Ligett, F. McSherry, A. Roth, and K. Talwar. Differentially private combinatorial optimization. In *SODA*, pages 1106–1125, 2010.

S. Har-Peled and S. Mazumdar. On coresets for k-means and k-median clustering. In *STOC*, pages 291–300, 2004.

S. Har-Peled and M. Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SICOMP*, 35(5):1148–1184, 2006.

Z. Huang and J. Liu. Optimal differentially private algorithms for k-means clustering. In *PODS*, pages 395–408, 2018.

Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Cryptography from anonymity. In *FOCS*, pages 239–248, 2006.

C. Jin, P. Netrapalli, R. Ge, S. M. Kakade, and M. I. Jordan. A short note on concentration inequalities for random vectors with subgaussian norm. *CoRR*, abs/1902.03736, 2019.

M. Jones, H. L. Nguyen, and T. Nguyen. Differentially private clustering via maximum coverage. In *AAAI*, 2021.

P. Kairouz, Z. Liu, and T. Steinke. The distributed discrete gaussian mechanism for federated learning with secure aggregation. In *ICML*, 2021.

K. Makarychev, Y. Makarychev, and I. P. Razenshteyn. Performance of Johnson–Lindenstrauss transform for k-means and k-medians clustering. In *STOC*, pages 1027–1038, 2019.

D. Micciancio. Almost perfect lattices, the covering radius problem, and applications to Ajtai's connection factor. *SICOMP*, 34(1):118–169, 2004.

D. Micciancio and P. Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. *SICOMP*, 42(3):1364–1391, 2013.

P. Mohan, A. Thakurta, E. Shi, D. Song, and D. Culler. GUPT: privacy preserving data analysis made easy. In *SIGMOD*, pages 349–360, 2012.

G. Monge. *Mémoire sur la théorie des déblais et des remblais*. De l'Imprimerie Royale, 1781.

K. Nissim and U. Stemmer. Clustering algorithms for the centralized and local models. In *ALT*, pages 619–653, 2018.

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

K. Nissim, U. Stemmer, and S. P. Vadhan. Locating a small cluster privately. In *PODS*, pages 413–427, 2016.

R. Nock, R. Canyasse, R. Boreli, and F. Nielsen. k-variates++: more pluses in the k-means++. In *ICML*, pages 145–154, 2016.

B. Piccoli and F. Rossi. Generalized Wasserstein distance and its application to transport equations with source. *Archive for Rational Mechanics and Analysis*, 211(1): 335–358, 2014.

C. Radebaugh and U. Erlingsson. Introducing TensorFlow Privacy: Learning with Differential Privacy for Training Data, March 2019. `blog.tensorflow.org`.

C. A. Rogers. Lattice coverings of space. *Mathematika*, 6 (1):33–39, 1959.

S. Shankland. How Google tricks itself to protect Chrome user privacy. *CNET, October*, 2014.

U. Stemmer. Locally private k-means clustering. In *SODA*, pages 548–559, 2020.

U. Stemmer and H. Kaplan. Differentially private k-means with constant multiplicative error. In *NeurIPS*, pages 5436–5446, 2018.

D. Su, J. Cao, N. Li, E. Bertino, and H. Jin. Differentially private k-means clustering. In *CODASPY*, pages 26–37, 2016.

D. Testuggine and I. Mironov. PyTorch Differential Privacy Series Part 1: DP-SGD Algorithm Explained, August 2020. `medium.com`.

Y. Wang, Y.-X. Wang, and A. Singh. Differentially private subspace clustering. In *NIPS*, pages 1000–1008, 2015.

C. Xia, J. Hua, W. Tong, and S. Zhong. Distributed k-means clustering guaranteeing local differential privacy. *Computers & Security*, 90:101699, 2020.