Differentially Private Clustering in High-Dimensional Euclidean Spaces

Maria-Florina Balcan [1] **Travis Dick** [1] **Yingyu Liang** [2] **Wenlong Mou** [3] **Hongyang Zhang** [1]

Abstract

We study the problem of clustering sensitive data while preserving the privacy of individuals represented in the dataset, which has broad applications in practical machine learning and data analysis tasks. Although the problem has been widely studied in the context of low-dimensional, discrete spaces, much remains unknown concerning private clustering in high-dimensional Euclidean spaces $\mathbb{R}^d$. In this work, we give differentially private and efficient algorithms achieving strong guarantees for k-means and k-median clustering when $d = \Omega(\mathsf{polylog}(n))$. Our algorithm achieves clustering loss at most $\log^3(n)\mathsf{OPT} + \mathsf{poly}(\log n, d, k)$, advancing the state-of-the-art result of $\sqrt{d}\mathsf{OPT} + \mathsf{poly}(\log n, d^d, k^d)$. We also study the case where the data points are s-sparse and show that the clustering loss can scale logarithmically with d, i.e., $\log^3(n)\mathsf{OPT} + \mathsf{poly}(\log n, \log d, k, s)$. Experiments on both synthetic and real datasets verify the effectiveness of the proposed method.

1. Introduction

In this work, we consider the problem of clustering sensitive data while preserving the privacy of individuals represented in the dataset. In particular, we consider k-means and k-median clustering under the constraint of differential privacy, which is a popular information-theoretic notion of privacy (Dwork et al., 2014; 2006) that roughly requires the output of algorithm to be insensitive to changes in an individual's data. Clustering is an important building block for many data processing tasks with applications in recommendation systems (McSherry & Mironov, 2009), database systems (Ester et al., 1996), image processing (Zhang et al., 2014; 2015; Pappas, 1992), and data mining (Berkhin, 2006). Improved privacy-preserving clustering algorithms have the potential to significantly improve the quality of many different areas of private data analysis.

Formally, we consider the following problem: Given a set of points $x_1, \ldots, x_n$ in $\mathbb{R}^d$, privately find a set of k centers $z_1, \ldots, z_k$ in $\mathbb{R}^d$ that approximately minimize one of the following clustering objectives:

$$\underbrace{\sum_{i=1}^{n} \min_j \|x_i - z_j\|^2}_{k\text{-means objective}} \quad \text{or} \quad \underbrace{\sum_{i=1}^{n} \min_j \|x_i - z_j\|}_{k\text{-median objective}}. \tag{1}$$

Minimizing these objectives exactly is NP-hard (Dasgupta, 2008), so we instead seek to find approximate solutions whose clustering objective is at most $\alpha \times \mathsf{OPT} + \beta$, where OPT denotes the optimal objective. Unlike the non-private setting, any differentially private clustering algorithm must have $\beta > 0$ (Bassily et al., 2014).

Despite a large amount of work on private clustering, many fundamental problems remain open. One of the long-standing challenges is to design *polynomial-time* private k-means and k-median clustering algorithms with small clustering loss for the high-dimensional, big-data setting. There is significant evidence to indicate that even without the requirement on privacy, exact optimization of these objective function in Euclidean spaces might be computationally infeasible (Dasgupta, 2008; Aloise et al., 2009): The best-known result in this line of research is $(9+\epsilon) \times \mathsf{OPT}$ by the local search algorithm (Kanungo et al., 2002). When the space is discrete, there exists a private algorithm that provides slightly better loss guarantee $6 \times \mathsf{OPT}$ (Gupta et al., 2010). However, this problem becomes notoriously hard in the context of differential privacy in Euclidean spaces, as one typically needs to preserve privacy for each point in the much larger Euclidean space. While there exist algorithms in this setting with clustering loss $\mathsf{polylog}(k) \times \mathsf{OPT} + \mathsf{O}(n/\log^2 n)$ (Nock et al., 2016), or $\sqrt{d} \times \mathsf{OPT} + \mathsf{poly}(d^d, k^d, \log n)$ when d is a small constant (Feldman et al., 2009) (See Table 1), the problem is left unresolved in the big-data (large n), high-dimensional ($d = \Omega(\mathsf{polylog} n)$) scenarios if we require both α and β to be as small as $\mathsf{polylog}(n)$. Note that running the algorithm of Feldman et al. (2009) after projecting to $\mathsf{O}(\log n)$ dimensional space gives $\mathsf{O}(\log(n)^{\log(n)})$ error, and the algorithm of Nock et al. (2016) has $\tilde{\mathsf{O}}(n)$ additive loss. Moreover, brute-force discretization in $\mathsf{O}(\log n)$ dimensional space

[1]Carnegie Mellon University, Pittsburgh, PA, USA [2]Princeton University, Princeton, NJ, USA [3]Peking University, Beijing, China. Correspondence to: Wenlong Mou <mouwenlong@pku.edu.cn>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017.

Table 1. Comparisons of our clustering loss with the best prior efficient algorithms. We mark algorithms with extra assumption requirements on datasets with *.

Reference	Loss	Space	Private	Guarantee
Kanungo et al. (2002)	k-means	Euclidean	N	$\alpha = 9+\epsilon$, $\beta = 0$
Nissim et al. (2007)*	k-means	Euclidean	Y	$\alpha = 1$, $\beta = \frac{n\phi^2\sqrt{k}}{\epsilon}$
Gupta et al. (2010)	k-median	Discrete	Y	$\alpha = 6$, $\beta = \mathsf{poly}(k, \log\lvert V\rvert)$, $\lvert V\rvert$ is the size of space
Nock et al. (2016)*	k-means	Euclidean	Y	$\alpha = \mathsf{polylog}(k)$, $\beta = \frac{n}{\log^2 n}$
Feldman et al. (2009)	k-median	Euclidean	Y	$\alpha = \sqrt{d}$, $\beta = \mathsf{poly}(d^d, k^d, \log n)$
Ours	Both	Euclidean	Y	$\alpha = \mathsf{polylog}(n)$, general: $\beta = \mathsf{poly}(d, k, \log n)$ $\alpha = \mathsf{polylog}(n)$, s-sparse: $\beta = \mathsf{poly}(\log d, k, \log n, s)$

does not give a polynomial time algorithm.

Given the difficulty of the general form of private clustering, many positive results in this line of research have focused on the assumptions that the data points are well-separated. A set of data points are called *well-separated* if all near-optimal clusterings of data induce similar data partitions. We can exploit such a structural assumption in many aspects. For example, these well-separated datasets are amenable to slightly perturbed initialization (Ostrovsky et al., 2012). This is the main insight behind prior work for private k-means clustering (Nissim et al., 2007; Wang et al., 2015). However, these observations and techniques break down in the general case.

Another interesting setting is when the data is extremely high dimensional, but each example is s-sparse. In this case, we might hope to improve the additive error β to be as small as $\mathsf{poly}(\log d, k, \log n, s)$. When the entries of data points are dense, the dependence of $\beta = \mathsf{poly}(d)$ is in general inevitable even if the number of centers is 1, according to the lower bound for private empirical risk minimization (Bassily et al., 2014). While some prior works have explored the possibility of *non-private* clustering in the context of sparse data (Barger & Feldman, 2016), much remains unknown in the private setting.

1.1. Our Contributions

Our work tackles the problem of private clustering in high-dimensional Euclidean spaces, specifically in the case when we have plentiful high-dimensional data. We advance the state-of-the-art in several aspects:

- We design and analyze a *computationally efficient* algorithm for *private* k-means clustering with clustering loss at most $\log^3(n) \times \mathsf{OPT} + \mathsf{poly}(\log n, d, k)$ (See Corollary 1). In contrast to Nock et al. (2016) and Nissim et al. (2007), our algorithm achieves small clustering loss without additional assumptions on data. Furthermore, our clustering loss bound is also competitive even under their assumptions.
- We extend our algorithm to the problem of k-median clustering. The clustering loss is at most $\log^{3/2}(n) \times \mathsf{OPT} + \mathsf{poly}(\log n, d, k)$ (See Theorem 12). Our guarantee advances the state-of-the-art results of Feldman et al. (2009) in the high-dimensional space.
- In the case of s-sparse data, we further improve the additive error term β to be at most $\mathsf{poly}(\log n, \log d, s, k)$ for private k-means clustering (See Corollary 2). To the best of knowledge, this is the first result concerning the *computationally efficient*, *differentially private* clustering algorithm for high-dimensional s-sparse data.
- We propose an approach for privately constructing a candidate set of centers with approximation guarantee (See Theorem 5). The candidate set can be potentially applied to other problems and is of independent interest.
- We empirically compare our algorithm with the non-private k-means++ algorithm and four strong private baseilnes. Across all datasets, our algorithm is competitive with k-means++ and significantly outperforms the private baselines, especially for large dimensional data.

1.2. Our Techniques

Our algorithm has two main steps: First, we use the Johnson-Lindenstrauss (JL) transform to project the data into $\mathsf{O}(\log n)$-dimensional space and use a novel technique to privately construct a small set of good candidate centers in the projected space; Then we apply a discrete clustering algorithm to privately find k good centers from the candidate set. Centers in the original space are recovered by noisy averaging.

Private Candidate Set: Our algorithm uses a novel private technique that recursively subdivides low-dimensional Euclidean spaces to construct a set of candidate points containing good centers. The algorithm proceeds in rounds, recursively subdividing the space into multiple cubes until there are few points in each cube. Finally, the algorithm outputs the centers of all cubes as a candidate set of centers.

In order to make the above procedure work well, we need to achieve three goals: (a) The output preserves privacy; (b) The algorithm is computationally efficient, i.e., the size

of candidate set is polynomial in n and k; and (c) The candidate set has high (α, β)-approximation rate, namely, it contains a subset of size k on which the clustering loss is at most $\alpha \times \mathsf{OPT} + \beta$ with α, β small. To achieve goal (a), our algorithm randomizes the decision of whether to subdivide a cube or not. To achieve goal (b), we make the probability of empty cube being further divided negligible, and the size of candidate set is upper bounded by the size of a simple partition tree, which is $\mathsf{poly}(n, k)$. For goal (c), it suffices to ensure each of the cluster centers to be captured within distance at scale of its own radius. As a lot of data points will be gathered around an optimal center, we can put candidate centers at each cube containing many points during partition. Random shift and repetition are used to avoid the worst cases.

From Candidate Set to Private Clustering: Our private clustering algorithm then follows the technique of local swapping on the discrete set of candidate centers, inspired by the work of Gupta et al. (2010) for k-median clustering. Their algorithm maintains a set of k centers and greedily replaces one center with a better one from the candidate set. However, Gupta et al. (2010)'s algorithm only works for the k-median problem where the loss obeys the triangle inequality. To extend the analysis to the k-means problem, we adopt the techniques of Kanungo et al. (2002). In particular, we construct k swap pairs of points, take the average of gains of these swap pairs, and relate it to the optimal loss OPT. Finally, we recover the centers in the original high-dimensional space privately. Given the centers in the projected space, the centers in the original space has low sensitivity. We can thus take the noisy mean of a cluster to obtain a private center for each cluster.

2. Related Work

The problem of private clustering in the Euclidean spaces was investigated by Blum et al. (2005), who proposed a private version of Lloyd iteration, which we refer to as SuLQ k-means. However, the algorithm suffers from the absence of uniform guarantee on the clustering loss. Given the sensitive and non-convex nature of clustering objective, Nissim et al. (2007) and Wang et al. (2015) applied the sample-and-aggregate framework to address the problem of private clustering by making strong assumptions on the input data. When no assumption is made on the structure of data, Feldman et al. (2009) provided an information-theoretic upper bound $\mathsf{OPT} + \mathsf{poly}(k, d, \log n)$ for the clustering loss according to the brute-force discretization of whole space and the exponential mechanism. However, no *computationally efficient* algorithm is available in the high-dimensional Euclidean spaces with clustering loss even close to this bound: Feldman et al. (2009) proposed an efficient algorithm for the bi-criteria approximation in the constant-dimensional spaces, but their additive loss term β actually exponentially depends on d; Gupta et al. (2010) designed an algorithm with a constant-factor approximation ratio and $\mathsf{poly}(k, \log |V|)$ additive loss term for clustering in the finite-data space V, but the algorithm does not work in the Euclidean spaces. Recently, Nock et al. (2016) proposed a private version of the k-means++ algorithm. However, the additive loss term β therein is almost as high as the data size n.

3. Preliminaries

We define some notation and clarify our problem setup.

Notation: We will use capital letters to represent matrices or datasets and lower-case letter to represent vectors or single data points. For a vector v, $v[i]$ denotes the i^{th} entry of v. We denote by $\mathcal{M}(X)$ the output of an algorithm with input dataset X. We will frequently use d to indicate the dimension of input space, p to indicate the dimension of space after projection, and Λ to indicate the radius of input data. For vector norms, we will denote by $\|\cdot\|$ the ℓ_2 norm, $\|\cdot\|_0$ the number of non-zero entries, and $\|\cdot\|_\infty$ the maximum of absolute value among entries. Define $\mathcal{B}(x, \Lambda) = \{y : \|x - y\| \leq \Lambda\}$. We denote by $\mathcal{U}([-\Lambda, \Lambda]^p)$ the uniform distribution in the p-dimensional cube $[-\Lambda, \Lambda]^p$. For any set V, we denote by $|V|$ the cardinality of the set. We will frequently denote the clustering loss in problem (1) on the centers $z_1, z_2, ..., z_k$ by $\mathcal{L}(z_1, z_2, ..., z_k)$. A sample drawn from one-dimensional Laplace distribution with density $f(x) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right)$ is denoted by $\mathsf{Lap}(b)$.

Problem Setup: We use the following definition of differential privacy:

Definition 1 (ϵ-Differential Privacy)**.** *A randomized algorithm $\mathcal{M}$ with output range $\mathcal{O}$ is ϵ-differentially private if for any given set $S \subseteq \mathcal{O}$ and two datasets $X \in \mathbb{R}^{d \times n}$ and $Y = [X; z]$ for any $z \in \mathbb{R}^d$, we have $e^{-\epsilon} \Pr[\mathcal{M}(X) \in S] \leq \Pr[\mathcal{M}(Y) \in S] \leq e^{\epsilon} \Pr[\mathcal{M}(X) \in S]$.*

In this paper, we study the problem of private clustering in high-dimensional Euclidean spaces without making any assumptions on the data structure. Formally, we define our problem as follows.

Problem 1 (Private Clustering in High-Dimensional Euclidean Spaces)**.** *Suppose $d = \Omega(\mathsf{polylog}(n))$ is the dimension of Euclidean space. Given bounded data points $x_1, x_2, ..., x_n \in \mathbb{R}^d$ as input, how can we efficiently output k centers $z_1, z_2, ..., z_k$ such that the algorithm is ϵ-differentially private and the clustering loss is at most $\mathsf{polylog}(n) \times \mathsf{OPT} + \mathsf{poly}(d, \log n, k, \frac{1}{\epsilon})$? In the case of sparse data where $\|x_i\|_0 \leq s$ for each i, can we improve the clustering loss to $\mathsf{polylog}(n) \times \mathsf{OPT} + \mathsf{poly}(\log d, \log n, k, s, \frac{1}{\epsilon})$?*

Algorithm 1 `private_partition`$(\{x_i\}_{i=1}^n, \epsilon, \delta, Q)$.

input $X = [x_1, x_2, \cdots, x_n] \subseteq \mathcal{B}(0, \Lambda) \subseteq \mathbb{R}^p$, parameters ϵ, δ, initial cube Q s.t. $\{x_i\}_{i=1}^n \subseteq Q$.
output Private grid $C \subseteq \mathbb{R}^p$.
  Initialize depth $a = 0$, active set of cubes $\mathcal{A} = \{Q\}$, and set $C = \varnothing$.
  while $a \le \log n$ **and** $\mathcal{A} \neq \varnothing$ **do**
    $a = a + 1$.
    $C = C \cup \left(\bigcup_{Q_i \in \mathcal{A}} \mathsf{center}(Q_i)\right)$.
    for $Q_i \in \mathcal{A}$ **do**
      Remove Q_i from $\mathcal{A}$.
      Partition Q_i evenly in each dimension and obtain 2^p cubes $\{Q_i^{(l)}\}_{l=1}^{2^p}$.
      for $l \in \{1, 2, \cdots, 2^p\}$ **do**
        Add $Q_i^{(l)}$ to $\mathcal{A}$ with probability $f\left(|Q_i^{(l)} \cap X|\right)$, where

$$f(m) = \begin{cases} \frac{1}{2}\exp(-\epsilon'(\gamma - m)), & m \le \gamma, \\ 1 - \frac{1}{2}\exp(\epsilon'(\gamma - m)), & \text{otherwise}, \end{cases}$$

        $\epsilon' = \frac{\epsilon}{2\log n}$ and $\gamma = \frac{20}{\epsilon'}\log\frac{n}{\delta}$.
      end for
    end for
  end while

4. Private Candidate Set

In this section, we present an efficient algorithm that constructs a polynomial-sized candidate set of centers privately in the low-dimensional space $\mathbb{R}^p$ with dimension $p = 8\log n$. This algorithm will serve as the building block for the private clustering. Our algorithm works by repeatedly applying a recursive discretization of the space with random shifts. It is worth noticing that direct extension of previous methods such as (Matoušek, 2000) lead to arbitrarily bad quality. The random shift is thus essential to our proof, which will be further explained in Appendix.

4.1. Private Discretization Routine

We first describe our subroutine of private discretization, a private recursive division procedure: We start with a cube containing all the data points, privately decide whether to partition the current cubes based on number of data points they contain, and stop when there are few points in each cube. Our algorithm is a variation of the hierarchical partitioning in (Matoušek, 2000), while setting appropriate stopping probabilities preserves privacy for our algorithm (See Algorithm 1).

To make the algorithm computationally efficient, we need to show that the number of candidate centers generated by Algorithm 1 is as small as $\mathsf{poly}(n)$. This is based on the fact that, by design, no empty cube is subdivided by our algorithm with high probability. A cube is called *active* if the algorithm will divide the cube into multiple subcubes in the next round. We have the following theorem on the size of candidate set.

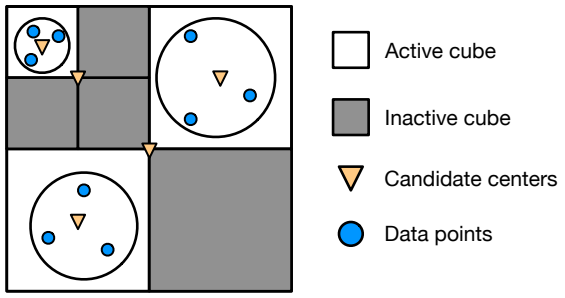


Figure 1. Constructing candidate set of centers by Algorithm 1. Here we recursively divide each active cube into multiple subcubes. Roughly, a cube is called *active* if there are sufficient points therein. The algorithm outputs the centroid of each active cube as the candidate set of centers.

Theorem 1. *The set C generated by Algorithm 1 satisfies $|C| \le n\log n$, with probability $1 - \delta$.*

Though we have generated $n\log n$ candidate centers, they are well-aligned and depend only slightly on the data. This alignment makes it possible for us to perform composition argument by the number of recursion instead of by number of points. We have the following theorem on the privacy.

Theorem 2. *Algorithm 1 preserves ϵ-differential privacy.*

The following theorem uses tail bounds for the exponential distribution and the union bound to upper bound the number of points in each cube not subdivided by Algorithm 1.

Theorem 3. *With probability at least $1 - \delta$, in Algorithm 1, when a cube Q_i is removed from $\mathcal{A}$ and its subdivided cubes are not inserted to $\mathcal{A}$, then we have either $|Q_i \cap X| \le O\left(\gamma\log\frac{n}{\delta}\right)$, or the edge length of Q_i is at most $\frac{\Lambda}{n}$.*

4.2. Private Construction of Candidate Set

We now give an algorithm that constructs a polynomial-sized candidate set with ϵ-differential privacy by applying the above procedure of private discretization as a subroutine. A good candidate set should contain k potential centers with small clustering loss relative to OPT. We formalize such a criterion as follows.

Definition 2 ((α, β)-Approximate Candidate Set)**.** *Given a set of points $S = \{x_1, x_2, \cdots, x_n\} \subseteq \mathbb{R}^p$, a set of points $C \subseteq \mathbb{R}^p$ is called an (α, β)-approximate candidate set of centers, if $\exists z_1, z_2, \cdots, z_k \in C$ such that the clustering loss (1) on these points is at most $\alpha \times \mathsf{OPT} + \beta$.*

As an example, the dataset S is itself a $(2, 0)$-approximate candidate set, although is not private. One may also use an

Algorithm 2 `candidate`$(\{x_i\}_{i=1}^n, \epsilon, \delta)$.

input $X = [x_1, x_2, \cdots, x_n] \subseteq \mathcal{B}(0, \Lambda) \subseteq \mathbb{R}^p$, parameters ϵ, δ.
output Candidate center set C.
  Initialize $C = \varnothing$.
  for $t = 1, 2, \cdots, T = 27k \log \frac{n}{\delta}$ **do**
    Sample shift vector $v \sim \mathcal{U}([-\Lambda, \Lambda]^p)$.
    Let $Q_v = [-\Lambda, \Lambda]^d + v$.
    $C = C \cup$ `private_partition`$(\{x_i\}_{i=1}^n, \frac{\epsilon}{T}, \frac{\delta}{T}, Q_v)$.
  end for

$\frac{\Lambda}{n}$-cover of $\mathcal{B}(0, \Lambda)$ to construct an $(1, O(k\Lambda\gamma \log \frac{n}{\delta})\Lambda^2)$-approximate candidate set with privacy. However, this brute-force discretization results in a set of size $n^{\Omega(p)}$ in $\mathbb{R}^p$, which depends on n exponentially even if $p = \Theta(\log n)$. In contrast, our following Algorithm 2 efficiently constructs an $(\mathsf{O}(\log^3 n), \mathsf{O}(k\mathsf{polylog}(n)))$-approximate candidate set of size polynomial in n.

Since Algorithm 2 only sees the private data through repeated application of Algorithm 1, we obtain the following privacy guarantee using standard composition theorems.

Theorem 4. *Algorithm 2 preserves ϵ-differential privacy.*

The remaining key argument is to show the approximation rate of candidate set constructed by Algorithm 2. The randomness and repetition is critical for the algorithm: They make it possible for us to "guess" the position of optimal centers and avoid the worst cases. Figure 1 depicts how each optimal cluster may be captured by a cube of the appropriate scale, provided that the center of the cluster is not near the boundary of a cube. Our proof techniques are partly inspired by random grids for near neighbor reporting (Aiger et al., 2014) and locality sensitive hashing (Andoni & Indyk, 2006). We give a short sketch of our proof in the following, and readers may refer to the Appendix for complete proofs.

Theorem 5. *With probability at least $1 - \delta$, Algorithm 2 outputs an $\left(\mathsf{O}(\log^3 n), \mathsf{O}(k\gamma(\frac{\epsilon}{T}) \log \frac{n}{\delta})\right)$-appriximate candidate set of centers, where $\gamma(c) = \frac{40}{c} \log \frac{n}{\delta} \log n$, and $T = k \log \frac{n}{\delta}$.*

Proof Sketch. There exists a set of fixed but unknown optimal centers $u_1^*, u_2^*, \cdots, u_k^*$, and corresponding optimal clusters $S_1^*, S_2^*, \cdots, S_k^*$. We say u_i^* is captured by C with factor L, if $\mathcal{B}\left(u_j^*, Lr_j^* + \mathsf{O}(\frac{1}{n})\right) \cap C \neq \varnothing$, where $r_j^* = \sqrt{\frac{1}{|S_j^*|} \sum_{i \in S_j^*} \|x_i - u_l\|^2}$ is the average loss. We will show that any optimal center is captured with factor $O(\log^{3/2} n)$, unless the size of corresponding cluster is too small.

For each u_i^*, we can guarantee the number of points around it, using Markov Inequality: $\left|\mathcal{B}(u_j^*, 2r_j^*) \cap S_j^*\right| \geq \frac{1}{2}\left|S_j^*\right|$. Consider the tree induced by hierarchical partition, as shown in Theorem 3, it doesn't stop being divided until either there're only $\gamma(\frac{\epsilon}{T}) \log \frac{n}{\delta}$ data points within it, or it's edge length is less than $\frac{1}{n}$. Since the center of a cube Q_l can capture points within this cube with factor $\sqrt{p}$, we only need to show the partition tree is activated at a level with edge length pr_j^*.

If the ball around u_j^* is completely contained in Q_l, we've already capture this center with $\mathsf{O}(\log^3 n)$ factor. But actually the ball can be divided into several cubes, making it hard to activate this cube. That's why we turn to the random shift. Using geometric arguments we can show that, $\mathcal{B}(u_j^*, 2r_j^*) \subseteq Q_l$ with constant probability. Several repetitions are then used to boost the probability of success, and to make it uniformly hold for k centers.

Therefore, we can guarantee that each optimal cluster with size at least $\Omega\left(k\gamma \log \frac{n}{\delta}\right)$ will be captured. Smaller clusters can be ignored safely, as its contribution to the total clustering loss goes to the $\sigma = \mathsf{O}\left(\frac{k}{\epsilon} \log^3 \frac{n}{\delta}\right)$ term. □

Algorithm 3 `localswap`$(\{x_i\}_{i=1}^n, C, \epsilon, \delta)$.

input Private dataset $\{x_i\}_{i=1}^n \subseteq \mathbb{R}^p$ with $\|x_i\| \leq \Lambda$, parameters ϵ, δ, candidate set C.
output Clustering centers $Z = [z_1, z_2, \cdots, z_k] \subseteq C$.
  Uniformly sample k centers $i.i.d.$ from C and form $Z^{(0)}$.
  $T \leftarrow 100k \log \frac{n}{\delta}$.
  for $t = 1, 2, \cdots, T$ **do**
    Choose $\left(x \in Z^{(t-1)}, y \in C \setminus Z^{(t-1)}\right)$ with probability in proposition to $\exp\left(-\epsilon \frac{\mathcal{L}(Z') - \mathcal{L}(Z^{(t-1)})}{8\Lambda^2(T+1)}\right)$, where $Z' = Z^{(t-1)} - \{x\} + \{y\}$.
    $Z^{(t)} \leftarrow Z^{(t-1)} - \{x\} + \{y\}$.
  end for
  Choose $t \in \{1, 2, \cdots, T\}$ with probability in proportion to $\exp\left(-\frac{\epsilon \mathcal{L}(Z^{(t)})}{8(T+1)\Lambda^2}\right)$.
  Output $Z^{(t)}$.

5. From Candidate Set to Private Clustering

In this section, we develop an efficient algorithm of private clustering based on the candidate set of centers that we construct in the low-dimensional spaces. Technically, our approach is a two-step procedure of *private discrete clustering* in the low-dimensional space and *private recovery* in the original high-dimensional space. In particular, the step of *private discrete clustering* extends the work of Gupta et al. (2010) to the k-means problem on the candidate set of centers, and the step of *private recovery* outputs k centers in the input space.

5.1. Private Discrete Clustering

In this section, we propose a differentially private k-means algorithm in the discrete spaces. Inspired from the previous work on k-median problem (Gupta et al., 2010), our algorithm builds upon the local swap heuristics for k-means clustering (Kanungo et al., 2002): In each round, the algorithm maintains a set greedily by replacing one point therein with a better one outside (See Algorithm 3). We first prove that such an algorithm is differentially private.

Theorem 6. *Algorithm 3 preserves ϵ-differential privacy.*

Proof. The privacy guarantee is straightforward using the basic composition theorem over T rounds of the algorithm, and an additional exponential mechanism that selects the best one. It is easy to verify the sensitivity of loss increments $\mathcal{L}(Z-\{x\}+\{y\})-\mathcal{L}(Z)$ is $8\Lambda^2$, the privacy guarantee of exponential mechanism in each round follows. □

The analysis of clustering loss of Algorithm 3 is based on a lower bound on the total gains of k swap pairs (Gupta et al., 2010). However, for the k-means problem, the triangle inequality does not hold for the quadratic ℓ_2 loss. To resolve this issue, we apply the inequality relaxation techniques for swap pairs developed by Kanungo et al. (2002). We have the following theorem on the clustering loss.

Theorem 7. *With probability at least $1-\delta$, the output of Algorithm 3 obeys $\mathcal{L}(Z) \le 30\mathsf{OPT} + \mathsf{O}\left(\frac{k^2\Lambda^2}{\epsilon}\log^2\frac{n|C|}{\delta}\right)$.*

5.2. Private Recovery of Centers in Original Space

We now propose Algorithm 4 for approximately recovering k centers in the original high-dimensional space. This algorithm is basically built on Algorithms 2 and 3 as subroutines: Algorithm 2 receives a set of points in the low-dimensional projected space as input, and outputs a small set of points that contains k centers with good clustering loss; Algorithm 3 privately outputs a set of clustering centers from a given candidate set.

The following parallel composition lemma (McSherry, 2009) guarantees that if we have an ϵ-differentially private algorithm for recovering the center of one cluster, then we can use it to output the centers of all k centers while still preserving ϵ differential privacy. This result follows from the fact that the clusters are disjoint.

Lemma 1 (McSherry (2009))**.** *Let $C_1, \ldots, C_k$ be any partition of the points $x_1, \ldots, x_n$ in $\mathbb{R}^d$ and suppose that $\mathcal{A}(S)$ is an ϵ differentially private algorithm that operates on sets of points in $\mathbb{R}^d$. Outputting $(\mathcal{A}(C_1), \ldots, \mathcal{A}(C_k))$ also preserves ϵ differential privacy.*

Now we are ready to prove the privacy of Algorithm 4.

Theorem 8. *Assume that* $\texttt{candidate}\left(\{x_i\}_{i=1}^n, \epsilon, \delta\right)$ *(Algorithm 2) preserves ϵ-differential privacy for $\{x_i\}_{i=1}^n$, and that given any candidate set of centers C,* $\texttt{localswap}\left(\{x_i\}_{i=1}^n, C, \epsilon, \delta\right)$ *(Algorithm 3) preserves ϵ-differential privacy for $\{x_i\}_{i=1}^n$. Then Algorithm 4 preserves ϵ-differential privacy.*

Algorithm 4 Private Clustering.

input $x_1, x_2, \cdots, x_n \in \mathcal{B}(0,\Lambda)$, parameters k, ϵ, δ.
output Clustering centres $z_1, z_2, \cdots, z_k \in \mathbb{R}^d$.
  Set dimension $p = 8\log n$, number of trials $T = 2\log\frac{1}{\delta}$.
  for $t = 1, 2, \cdots T$ **do**
    Sample $G \sim \mathcal{N}(0,1)^{p\times d}$.
    $[y_1, y_2, \cdots, y_n] = \frac{1}{\sqrt{d}} G[x_1, x_2, \cdots, x_n]$.
    $C = \texttt{candidate}\left(\{y_i\}_{i=1}^n, \frac{\epsilon}{6T}, \delta\right)$.
    $\{u_1, u_2, \cdots, u_k\} = \texttt{localswap}\left(\{y_i\}_{i=1}^n, C, \frac{\epsilon}{6T}, \delta\right)$.
    $S_j = \{i : j = \operatorname{argmin}_l \|y_i - u_l\|\}$, $j = 1, 2, \cdots, k$.
    $s_j = \max\left\{|S_j| + \mathsf{Lap}\left(\frac{24T}{\epsilon}\right), 1\right\}$.
    $z_j^{(t)} = \frac{1}{|s_j|}\sum_{x_i\in S_j} x_i + \mathsf{Lap}\left(\frac{24T\Lambda}{\epsilon s_j}\right)^d, \forall j$.
  end for
  Choose Z from $Z^{(1)}, Z^{(2)}, \cdots, Z^{(T)}$ with probability in proportion to $\exp\left(-\frac{\epsilon\mathcal{L}(Z^{(t)})}{24\Lambda^2}\right)$.

Putting everything together, we have the following theorem on the clustering loss of Algorithm 4. The key technique in our proof is to convert the argument of preservation of pairwise distance in the JL Lemma to the bound on the clustering loss. This is due to a simple observation that the optimal loss in any cluster only depends on the pairwise distances among its data points.

Theorem 9. *Assume that* $\texttt{candidate}\left(\{x_i\}_{i=1}^n, \epsilon, \delta\right)$ *(Algorithm 2) outputs an $(\alpha, \sigma_1(\epsilon))$-approximate candidate set with probability at least $\frac{2}{3}$, and that with probability at least $\frac{2}{3}$,* $\texttt{localswap}\left(\{x_i\}_{i=1}^n, C, \epsilon, \delta\right)$ *(Algorithm 3) achieves clustering loss at most $c\mathsf{OPT}_C + \sigma_2(\epsilon)$, where OPT_C is the optimal clustering centers in the candidate set of centers C. Then with probability at least $1-\delta$, the output of Algorithm 4 has k-means clustering loss at most $3c\alpha\mathsf{OPT} + 3c\sigma_1' + 3\sigma_2' + \mathsf{O}\left(\frac{d\Lambda^2\log^3\frac{1}{\delta}}{\epsilon^2}\right)$, where $\sigma_i' = \sigma_i\left(\frac{\epsilon}{2\log 1/\delta}\right)$ for $i = 1, 2$.*

Theorem 9, together with Theorems 5 and 7, leads to the following guarantees on the clustering loss of Algorithm 4.

Corollary 1. *There is an ϵ-differential private algorithm that runs in $\mathsf{poly}(k, d, n)$ time, and releases a set of centers $\tilde{z}_1, \tilde{z}_2, \cdots, \tilde{z}_k$ such that with probability at least $1-\delta$, $\mathcal{L}\left(\{\tilde{z}_j\}_{j=1}^k\right) \le \mathsf{O}(\log^3 n)\mathsf{OPT} + \mathsf{O}\left(\frac{k^2\epsilon+d}{\epsilon^2}\Lambda^2\log^5\frac{n}{\delta}\right)$.*

6. Extensions

In this section, we present two extensions of our algorithms: a) private k-means clustering with high-dimensional sparse data; b) private k-median clustering.

6.1. High-Dimensional Sparse Data

Algorithm 5 Privately Recover Centers for Sparse Dataset.

input Private data set $\{x_i\}_{i=1}^n \subseteq \mathbb{R}^d$ with $\|x_i\|_\infty \leq \Lambda$, $\|x_i\|_0 \leq s$, parameters ϵ, δ, accuracy η.
output $v \in \mathbb{R}^d$.
Compute $\mu = \frac{1}{n}\sum_{i=1}^n x_i \in \mathbb{R}^d$.
Initialize $I = \{1, 2, \cdots, d\}$, $v = 0 \in \mathbb{R}^d$.
for $j \in \{1, 2, \cdots, \lceil \frac{2s}{\eta} \rceil\}$ **do**
  Sample $r \in I$ with probability in proportion to $\exp\left\{\frac{\epsilon\eta n}{4\Lambda s}|\mu[r]|\right\}$.
  $I = I \setminus \{r\}$, $v[r] = v[r] + \mu[r] + \mathsf{Lap}\left(\frac{4\Lambda s}{\epsilon\eta n}\right)$.
end for

For the case of high-dimensional sparse data where $\|x_i\|_0 \leq s$, our goal is to improve the additive loss term β to be as small as $\mathsf{poly}(k, s, \log d, \log n)$ by small modifications of Algorithm 4. The steps of discretization routine, construction of candidate set, and clustering in the discrete space all remain the same as in the general case. The only difference is the step of private recovery of centers in the high-dimensional original space. In the non-sparse setting, we simply take a noisy mean of points that belong to cluster i and output the center for cluster i, resulting in $\Omega(d)$ additive loss. However, such a procedure does not exploit the sparse nature of data points. The challenge is that a high-dimensional vector has too many entries to hide for differential privacy, usually resulting in large error. To improve the clustering loss, we force the output vector to be sparse, by choosing coordinates with large absolute values, while zeroing out others (See Algorithm 5). Both the choice of non-zero coordinates and the estimation of their values need to preserve privacy, for which we use both the exponential and Laplacian mechanisms. By a composition argument, we have the privacy of Algorithm 5.

Theorem 10. *Algorithm 5 preserves ϵ-differential privacy.*

The following theorem guarantees that for high-dimensional sparse data, the clustering loss has logarithmic dependence on the dimension.

Theorem 11. *With probability at least $1-\delta$, the output of Algorithm 5 obeys $\sum_{i=1}^n \|x_i - v\|^2 \leq \frac{1}{1-\eta}\mathsf{OPT} + \mathsf{O}\left(\frac{\Lambda^2 s^2 \log\frac{ds}{\eta\delta}}{\eta^2\epsilon}\right)$.*

The intuition of Theorem 11 is based on the following observation: If the mean is approximately sparse, we can truncate it safely with small additional loss; If not, the mean must spread across a large set of entries, so the support of data vectors must be very different from each other, making the variance large. In both cases, the loss of truncation can be bounded by the variance of data points, and we can put such a loss of truncation to the multiplicative factor.

By the privacy argument in Lemma 1, as well as Theorems 5 and 7, we have the following result.

Corollary 2. *For $x_1, x_2, \cdots, x_n \in \mathbb{R}^d$ with $\|x_i\|_0 \leq s$ and $\|x_i\|_\infty \leq C$, there is an ϵ-differentially private algorithm that runs in $poly(k, d, n)$ time, and releases a set of centers $\tilde{z}_1, \tilde{z}_2, \cdots, \tilde{z}_k$ such that with probability at least $1-\delta$, $\mathcal{L}\left(\{\tilde{z}_j\}_{j=1}^k\right) \leq \mathsf{O}(\log^3 n)\mathsf{OPT} + \mathsf{O}\left(\frac{sk^2+s^2\log\frac{d}{\delta}}{\epsilon}\log^2\frac{n}{\delta}\right)$.*

An important implication of Corollary 2 is that private clustering for high-dimensional sparse data is as easy as private clustering in $\mathsf{O}(\log d)$ dimensions. The approximation factor we can achieve in the high-dimensional sparse case is roughly the same as low-dimensional case.

6.2. k-Median Clustering

We can also easily modify our algorithms to adapt to k-median problem. Note that Theorem 5 is independent of form of loss function, since it is based on capturing the optimal centers. Therefore, the candidate set constructed in Algorithm 2 guarantees an $\left(\mathsf{O}(\log^{\frac{3}{2}} n), \mathsf{O}(k\Lambda\gamma\log\frac{n}{\delta})\right)$-approximation rate. Since the discrete clustering algorithm proposed by Gupta et al. (2010) is designed for k-median, it only remains to develop a private recovery procedure in the original space $\mathbb{R}^d$. According to Lemma 1, a private 1-median algorithm suffices to recover the centers. We can achieve good approximation rate via log-concave sampling (Bassily et al., 2014).

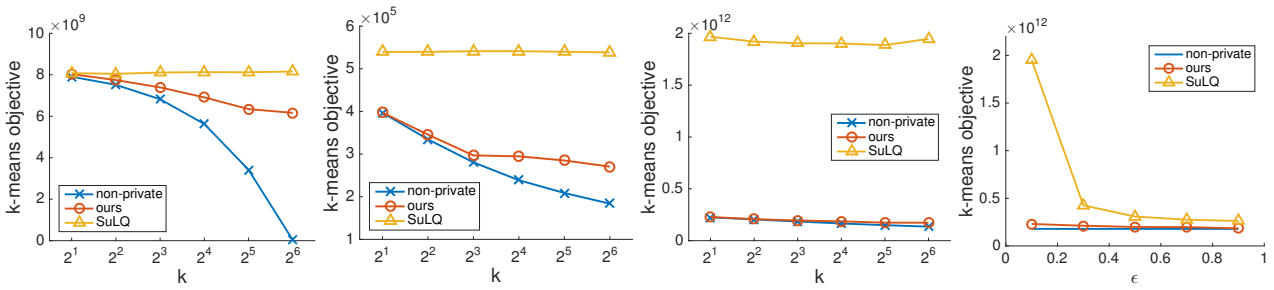
Lemma 2 (Error Bound for Exponential Mechanism (Bassily et al., 2014))**.** *For $x_1, x_2, \cdots, x_n \in \mathbb{R}^d$, there is a polynomial-time algorithm that releases a center z and preserves ϵ-differential privacy, such that with probability at least $1-\delta$, we have $\sum_{i=1}^n \|x_i - z\| - \min_p \|x_i - p\| \leq O(\frac{d\Lambda}{\epsilon}\log^2\frac{1}{\delta})$.*

Incorporating log-concave sampling into the step of private recovery in Algorithm 4, we derive a private k-median algorithm. The privacy guarantee follows directly from Lemma 1 and the composition argument. As for the k-median objective, the optimal clustering loss is no longer a function of pairwise distances. Fortunately, observe that the original dataset is $(2, 0)$-approximate candidate set for k-median loss. By this, we have the following guarantee.

Theorem 12. *For k-median problem, there is an ϵ-differentially private algorithms that runs in $poly(k, d, n)$ time, and releases a set of centers $\tilde{z}_1, \tilde{z}_2, \cdots, \tilde{z}_k$ such that with probability at least $1-\delta$, $\mathcal{L}\left(\{\tilde{z}_j\}_{j=1}^k\right) \leq \mathsf{O}(\log^{3/2} n)\mathsf{OPT} + \mathsf{O}\left(\frac{(k^2+d)\Lambda}{\epsilon}\log^3\frac{n}{\delta}\right)$.*

7. Experiments

In this section, we present an empirical evaluation of our proposed clustering algorithm and several strong baselines on real-world image and synthetic datasets. We compare against non-private k-means++ of Arthur & Vassilvitskii (2007), SuLQ k-means of Blum et al. (2005), the sample and aggregate clustering algorithm of Nissim et al. (2007),



(a) Synthetic (d=100) (b) CIFAR10 in3c (d=160) (c) MNIST (d=784) (d) Effect of ϵ (MNIST)

Figure 2. Figures (a-c) show the effect of k on the clustering objective and Figure (d) shows the effect of ϵ on the clustering objective.

the k-variates++ algorithm of Nock et al. (2016), and the griding algorithm of Su et al. (2016). The k-variates++ algorithm can only run when k is small, and the griding algorithm has time and space complexity exponential in the dimension, so we are only able to compare against these two baselines with small k or small d. We postpone detailed comparisons against these two algorithms to supplementary material. For all other datasets with higher dimensions and all values of k, our algorithm is competitive with non-private k-means++ and is always better than SuLQ and sample and aggregate. Moreover, in agreement with our theory, the gap between the performance of our algorithm and the other private baselines grows drastically as the dimension of the dataset increases.

The implementation of our algorithm projects to a space of dimension $p = \log(n)/2$, rather than $8\log(n)$ and repeats the candidate set construction routine only k times. Finally, we perform 8 iterations of the SuLQ k-means algorithm to further improve the quality of the resulting centers. These modifications do not affect the privacy guarantee of the algorithm, but gave improved empirical performance. Our implementation of the SuLQ k-means algorithm runs for 20 iterations and uses the Gaussian mechanism to approximate the sum of points in each cluster, since this allowed us to add less noise. The SuLQ algorithm initializes its centers to be k randomly chosen points from the bounding box of the data. Unless otherwise stated, we set $\epsilon = 1.0$.

Results: We first compared our algorithm and all baselines on a small synthetic dataset in 3 dimensions with $k = 3$. Su *et al.*'s griding algorithm achieves the best objective, while sample and aggregate, SuLQ, and our method all perform comparably, and k-variates++ is an order of magnitude worse. Details of the comparison are given in the supplementary material. The griding algorithm and k-variates++ were not able to run in the rest of our experiments.

Next, we ran the non-private k-means++, SuLQ k-means, and sample and aggregate algorithms on the following datasets for each value of k in $\{2, 4, 8, 16, 32, 64\}$. A more detailed description is given in the supplementary material.

MNIST: The raw pixels of MNIST (LeCun et al., 1998). It has 70k examples and 784 features.

CIFAR-10: 100k randomly sampled examples from the CIFAR10 dataset (Krizhevsky, 2009) with 160 features extracted from layer `in3c` of a Google Inception (Szegedy et al., 2015) network.

Synthetic: A synthetic dataset of 100k samples drawn from a mixture of 64 Gaussians in $\mathbb{R}^{100}$.

Figure 2 (a-c) shows the objective values obtained by each algorithm averaged over 5 independent runs. The sample and aggregate algorithm's results have been omitted, since its objective values are orders of magnitude worse than the other algorithms. Across all values of k and all datasets, our algorithm is competitive with non-private k-means++ and always outperforms SuLQ k-means. As the dimensionality of the datasets increases, our algorithm remains competitive with k-means++, while SuLQ becomes less competitive for large dimensions.

Finally, figure 2 (d) shows the effect of the privacy parameter ϵ on the objective values for each algorithm on MNIST with $k = 10$. Our algorithm is competitive with the non-private k-means algorithm even for small ϵ, while the SuLQ algorithm objective deteriorates quickly.

8. Conclusions

In this paper, we propose efficient algorithms for ϵ-private k-means and k-median clustering in $\mathbb{R}^d$ that achieves clustering loss at most $\mathsf{O}\left(\log^3 n\right)\mathsf{OPT} + \mathsf{poly}\left(k, d, \log n, \frac{1}{\epsilon}\right)$ and $\mathsf{O}\left(\log^{\frac{3}{2}} n\right)\mathsf{OPT} + \mathsf{poly}\left(k, d, \log n, \frac{1}{\epsilon}\right)$, respectively. We also study the scenario where the data points are s-sparse and show that the k-means clustering loss can be even smaller, namely, $\mathsf{O}\left(\log^3 n\right)\mathsf{OPT} + \mathsf{poly}\left(k, s, \log d, \log n, \frac{1}{\epsilon}\right)$. Results of this type advance the state-of-the-art approaches in the high-dimensional Euclidean spaces. Our method of constructing candidate set can be potentially applied to other problems, which might be of independent interest more broadly.

Acknowledgments

The authors would like to thank Liwei Wang and Colin White for helpful discussions. Parts of this work was done when W.M. was visiting CMU. This work was done when Y.L. was visiting Simons Institute. This work was supported in part by grants NSF IIS-1618714, NSF CCF-1535967, NSF CCF-1422910, NSF CCF-1451177, a Sloan Fellowship, a Microsoft Research Fellowship, NSF grants CCF-1527371, DMS-1317308, Simons Investigator Award, Simons Collaboration Grant, ONRN00014-16-1-2329, and the Chinese MOE Training Plan for Top-Notch Students in Basic Discipline.

References

Aiger, Dror, Kaplan, Haim, and Sharir, Micha. Reporting neighbors in high-dimensional euclidean space. *SIAM Journal on Computing*, 43(4):1363–1395, 2014.

Aloise, Daniel, Deshpande, Amit, Hansen, Pierre, and Popat, Preyas. Np-hardness of euclidean sum-of-squares clustering. *Machine learning*, 75(2):245–248, 2009.

Andoni, Alexandr and Indyk, Piotr. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *IEEE Symposium on Foundations of Computer Science*, pp. 459–468, 2006.

Arthur, David and Vassilvitskii, Sergei. k-means++: The advantages of careful seeding. In *ACM-SIAM Symposium on Discrete Algorithms*, pp. 1027–1035, 2007.

Barger, Artem and Feldman, Dan. k-means for streaming and distributed big sparse data. In *SIAM International Conference on Data Mining*, pp. 342–350, 2016.

Bassily, Raef, Smith, Adam, and Thakurta, Abhradeep. Differentially private empirical risk minimization: Efficient algorithms and tight error bounds. *arXiv preprint arXiv:1405.7085*, 2014.

Berkhin, Pavel. A survey of clustering data mining techniques. In *Grouping Multidimensional Data*, pp. 25–71. 2006.

Blum, Avrim, Dwork, Cynthia, McSherry, Frank, and Nissim, Kobbi. Practical privacy: the SuLQ framework. In *ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pp. 128–138, 2005.

Dasgupta, Sanjoy. *The hardness of k-means clustering*. Department of Computer Science and Engineering, University of California, San Diego, 2008.

Dwork, Cynthia, McSherry, Frank, Nissim, Kobbi, and Smith, Adam. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, 2006.

Dwork, Cynthia, Roth, Aaron, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Ester, Martin, Kriegel, Hans-Peter, Sander, Jörg, Xu, Xiaowei, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, volume 96, pp. 226–231, 1996.

Feldman, Dan, Fiat, Amos, Kaplan, Haim, and Nissim, Kobbi. Private coresets. In *ACM Symposium on Theory of Computing*, pp. 361–370, 2009.

Gupta, Anupam, Ligett, Katrina, McSherry, Frank, Roth, Aaron, and Talwar, Kunal. Differentially private combinatorial optimization. In *ACM-SIAM symposium on Discrete Algorithms*, pp. 1106–1125, 2010.

Kanungo, Tapas, Mount, David M, Netanyahu, Nathan S, Piatko, Christine D, Silverman, Ruth, and Wu, Angela Y. A local search approximation algorithm for k-means clustering. In *Annual Symposium on Computational Geometry*, pp. 10–18, 2002.

Krizhevsky, Alex. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 1998.

Matoušek, Jırı. On approximate geometric k-clustering. *Discrete & Computational Geometry*, 24(1):61–84, 2000.

McSherry, Frank and Mironov, Ilya. Differentially private recommender systems: building privacy into the net. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 627–636, 2009.

McSherry, Frank D. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *ACM SIGMOD International Conference on Management of Data*, pp. 19–30, 2009.

Nissim, Kobbi, Raskhodnikova, Sofya, and Smith, Adam. Smooth sensitivity and sampling in private data analysis. In *ACM Symposium on Theory of Computing*, pp. 75–84, 2007.

Nock, Richard, Canyasse, Raphaël, Boreli, Roksana, and Nielsen, Frank. k-variates++: more pluses in the k-means++. *arXiv preprint arXiv:1602.01198*, 2016.

Ostrovsky, Rafail, Rabani, Yuval, Schulman, Leonard J, and Swamy, Chaitanya. The effectiveness of Lloyd-type methods for the k-means problem. *Journal of the ACM*, 59(6):28, 2012.

Pappas, Thrasyvoulos N. An adaptive clustering algorithm for image segmentation. *IEEE Transactions on Signal Processing*, 40(4):901–914, 1992.

Su, Dong, Cao, Jianneng, Li, Ninghui, Bertino, Elisa, and Jin, Hongxia. Differentially private k-means clustering. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, CODASPY '16, 2016.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Wang, Yining, Wang, Yu-Xiang, and Singh, Aarti. Differentially private subspace clustering. In *Advances in Neural Information Processing Systems*, pp. 1000–1008, 2015.

Zhang, Hongyang, Lin, Zhouchen, Zhang, Chao, and Gao, Junbin. Robust latent low rank representation for subspace clustering. *Neurocomputing*, 145:369–373, 2014.

Zhang, Hongyang, Lin, Zhouchen, Zhang, Chao, and Gao, Junbin. Relations among some low-rank subspace recovery models. *Neural computation*, 2015.