Differentially Private K-Means Clustering

Dong Su
Purdue University
su17@cs.purdue.edu

Jianneng Cao
Institute for Infocomm Research
caojn@i2r.a-star.edu.sg

Ninghui Li
Purdue University
ninghui@cs.purdue.edu

Elisa Bertino
Purdue University
bertino@cs.purdue.edu

Hongxia Jin
Samsung Research America
hongxia.jin@samsung.com

ABSTRACT

There are two broad approaches for differentially private data analysis. The *interactive* approach aims at developing customized differentially private algorithms for various data mining tasks. The *non-interactive approach* aims at developing differentially private algorithms that can output a synopsis of the input dataset, which can then be used to support various data mining tasks. In this paper we study the effectiveness of the two approaches on differentially private k-means clustering. We develop techniques to analyze the empirical error behaviors of the existing interactive and non-interactive approaches. Based on the analysis, we propose an improvement of DPLloyd which is a differentially private version of the Lloyd algorithm. We also propose a non-interactive approach EUGkM which publishes a differentially private synopsis for k-means clustering. Results from extensive and systematic experiments support our analysis and demonstrate the effectiveness of our improvement on DPLloyd and the proposed EUGkM algorithm.

Keywords

Differential privacy; k-means clustering; Private data publishing

1. INTRODUCTION

In recent years, differential privacy [10] has been increasingly adopted as the privacy notion of choice of data analysis while preserving individual privacy. Several broad classes of approaches exist for developing differentially private techniques for data analysis. In this paper we study differentially private k-means clustering. Clustering analysis plays an essential role in data management tasks. Clustering under differential privacy has also been studied in [3, 9, 19, 22, 24, 25, 34].

Our study has two goals. The first is to improve the techniques for performing k-means clustering differentially privately. The second is to use k-means clustering as a case study to compare several classes of methods for private data analysis, and to identify the strengths and weaknesses of these methods.

There are three state-of-the-art differentially private algorithms on k-means clustering. All of them are interactive approaches. The first method, which we call DPLloyd, makes the iterative Lloyd algorithm [3, 22] differentially private by adding noises to each step. The second method, which we call PGkM, uses PrivGene [34], a framework for differentially private model fitting based on genetic algorithms. We call them *iterative interactive* algorithms. The third algorithm uses the sample and aggregation framework [25] and is implemented in the GUPT system [24], which we call GkM.

An alternative to the interactive setting is the *non-interactive* setting, in which the data curator releases the data in one shot, while still preserving privacy. To the best of our knowledge, performing k-means clustering using the non-interactive approach has not been explicitly proposed in the literature. In this paper, we propose to combine the following non-interactive differentially private synopsis algorithms with k-means clustering. The dataset is viewed as a set of points over a d-dimensional domain, which is divided into M equal-size cells, and a noisy count is obtained from each cell. A key decision is to choose the parameter M. A larger M value means lower average counts for each cell, and therefore noisy counts are more likely to be dominated by noises. A smaller M value means larger cells, and therefore one has less accurate information of where the points are. We propose a method that sets $M = \left(\frac{N\epsilon}{10}\right)^{\frac{2d}{2+d}}$, which is derived based on extending the analysis in [27], which aims to minimize errors when answering rectangular range queries for 2-dimensional data, to higher dimensional case. We call the resulting k-means algorithm EUGkM, where EUG is for Extended Uniform Grid.

We conducted extensive experimental evaluations for these algorithms on 6 external datasets and 81 datasets that we synthesized by varying the dimension d from 2 to 10 and the number of clusters from 2 to 10. Experimental results are quite interesting. GkM was introduced after DPLloyd and was claimed to have accuracy advantage over DPLloyd, and PGkM was introduced after and compared GkM. However, we found that DPLloyd is the best method among these three interactive methods. In the comparison of DPLloyd and GkM in [24], DPLloyd was run using much larger number of iterations than necessary, and thus perform poorly. In [34], PGkM was compared only with GkM, and not with DPLloyd. More specifically, we found that GkM is by far the worst among all methods. However, DPLloyd, the earliest method is clearly the best performing algorithm among the three interactive algorithms. Through analysis, we found that why DPLloyd outperforms PGkM. The genetic programming style PGkM needs more iterations to converge. When making these algorithms differentially private, the privacy budget is divided among all iterations, thus having more iterations means more noise is added to each iteration. Therefore, the more direct DPLloyd outperforms PGkM.

CODASPY'16, March 09-11, 2016, New Orleans, LA, USA

DOI: http://dx.doi.org/10.1145/2857705.2857708

The most intriguing results are those comparing DPLloyd with EUGkM. For most datasets, EUGkM performs much better than DPLloyd. For a few, they perform similarly, and for the rest datasets DPLloyd outperforms EUGkM. Through further theoretical and empirical analysis, we found that while the performance of both algorithms are greatly affected by the two key parameters, the number of dimensions d and the number of clusters k, both of them are affected differently by these two parameters. DPLloyd scales worse when k increases, while EUGkM scales worse when d increases. Again we use analysis to demonstrate why this is the case.

In this paper we advance the state-of-the-art on differentially private data mining in several ways. First, we have introduced a new non-interactive method, EUGkM, for differentially private k-means clustering, which are highly effective and often outperform the state-of-the-art interactive methods and non-interactive methods. Second, we have developed techniques to analyze the error resulted from DPLloyd. Such kind of error analysis is missed in most differentially private data analysis papers. Third, based on the error analysis of DPLloyd, we proposed an improved version of DPLloyd which significantly reduces the clustering error. Fourth, we have extensively evaluated three interactive methods, and three non-interactive methods, and analyzed their strengths and weaknesses.

The rest of the paper is organized as follows. In Section 2, we give preliminary information about differential privacy and k-means clustering. In Section 3, we describe the existing three interactive approaches, DPLloyd, GkM, PGkM, two non-interactive approaches UGkM and MkM, improve DPLloyd and derive a new non-interactive approach EUGkM. In Section 4, we first show the experimental results on the performance comparison among the interactive and non-interactive approaches, and analyze their strengths and weaknesses. In Section 5, we discuss related works. We conclude in Section 6.

2. BACKGROUND

2.1 Differential Privacy

Informally, differential privacy requires that the output of a data analysis mechanism should be approximately the same, even if any single tuple in the input database is arbitrarily added or removed.

Definition 1 (ϵ-Differential Privacy [8, 10])**.** *A randomized mechanism $\mathcal{A}$ gives ϵ-differential privacy if for any pair of neighboring datasets D and D', and any $S \subseteq Range(\mathcal{A})$,*

$$\mathsf{Pr}\left[\mathcal{A}(D) \in S\right] \leq e^{\epsilon} \cdot \mathsf{Pr}\left[\mathcal{A}(D') \in S\right].$$

In this paper we consider two datasets D and D' to be neighbors if and only if either $D = D' + t$ or $D' = D + t$, where $D + t$ denotes the dataset resulted from adding the tuple t to the dataset D. We use $D \simeq D'$ to denote this. This protects the privacy of any single tuple, because adding or removing any single tuple results in e^{ϵ}-multiplicative-bounded changes in the probability distribution of the output.

Differential privacy is composable in the sense that combining multiple mechanisms that satisfy differential privacy for $\epsilon_1, \cdots, \epsilon_m$ results in a mechanism that satisfies ϵ-differential privacy for $\epsilon = \sum_i \epsilon_i$. Because of this, we refer to ϵ as the *privacy budget* of a privacy-preserving data analysis task. When a task involves multiple steps, each step uses a portion of ϵ so that the sum of these portions is no more than ϵ.

There are several approaches for designing mechanisms that satisfy ϵ-differential privacy, including Laplace mechanism [10] and Exponential mechanism [21]. The Laplace mechanism computes a function g on the dataset D by adding to $g(D)$ a random noise, the magnitude of which depends on GS_g, the *global sensitivity* or the L_1 sensitivity of g. Such a mechanism $\mathcal{A}_g$ is given below:

$$\begin{aligned} \mathcal{A}_g(D) &= g(D) + \mathsf{Lap}\left(\tfrac{\mathsf{GS}_g}{\epsilon}\right) \\ \text{where} \quad \mathsf{GS}_g &= \max_{(D,D'):D\simeq D'} |g(D) - g(D')|, \\ \text{and} \quad \mathsf{Pr}\left[\mathsf{Lap}\left(\beta\right) = x\right] &= \tfrac{1}{2\beta} e^{-|x|/\beta}. \end{aligned}$$

In the above, $\mathsf{Lap}\left(\beta\right)$ denotes a random variable sampled from the Laplace distribution with scale parameter β.

2.2 k-means Clustering Algorithms

The k-means clustering problem is as follows: given a d-dimensional dataset $D = \{x^1, x^2, \ldots, x^N\}$, partition data points in D into k sets $\mathbf{O} = \{O^1, O^2, \cdots, O^k\}$ so that the Normalized Intra-Cluster Variance (NICV) is minimized

$$\frac{1}{N}\sum_{j=1}^{k}\sum_{x^\ell \in O^j} ||x^\ell - o^j||^2. \tag{1}$$

The standard k-means algorithm is the Lloyd's algorithm [18]. The algorithm starts by selecting k points as the initial choices for the centroid. The algorithm then tries to improve these centroid choices iteratively until no improvement can be made. In each iteration, one first partitions the data points into k clusters, with each point assigned to be in the same cluster as the nearest centroid. Then, one updates each centroid to be the center of the data points in the cluster.

$$\forall i \in [1..d]\ o_i^j \leftarrow \frac{\sum_{x^\ell \in O^j} x_i^\ell}{|O^j|}, \tag{2}$$

where $j = 1, 2, \ldots, k$, x_i^ℓ and o_i^j are the i-th dimensions of x^ℓ and o^j, respectively. The algorithm continues by alternating between data partition and centroid update, until it converges.

3. THE INTERACTIVE AND NON-INTERACTIVE APPROACHES

In this section, we describe and analyze three interactive approaches and three non-interactive approaches to differentially private k-means clustering.

3.1 DPLloyd and Proposed Improvements

3.1.1 DPLloyd

A differentially private version of the Lloyd's algorithm was first proposed by Blum et al. [3] and was later implemented in the PINQ system [22], a platform for interactive privacy preserving data analysis. We call this the DPLloyd approach. DPLloyd differs from the standard Lloyd algorithm in the following ways. First, Laplacian noise is added to the iterative update step in the Lloyd algorithm. Second, the number of iterations needs to be fixed in order to decide how much noise needs to be added in each iteration.

Each iteration requires computing the total number of points in a cluster and, for each dimension, the sum of the coordinates of the data points in a cluster. Let t be the number of iterations, and d be the number of dimensions. Then, each tuple is involved in answering dt sum queries and t count queries. To bound the sensitivity of the sum query to a small number r, each dimension is normalized to $[-r, r]$. Therefore, the global sensitivity of DPLloyd is $(dr+1)t$, and each query is answered by adding Laplacian noise $\mathsf{Lap}\left(\frac{(dr+1)t}{\epsilon}\right)$.

3.1.2 Optimization Issues

The overall structure of DPLloyd is to first select initial values, and then iteratively improve them. This same algorithmic structure also applies to many other data analysis tasks, such as linear regression, SVM, etc. When making such an interactive and iterative algorithm differentially private, there are two important decisions one has to make.

The first decision is how to select the initial values? In the standard, non-private setting, a purely random choice may suffice, since one could repeat the algorithm multiple times and choose the best result among them. With privacy constraints, however, running the interactive algorithm multiple times results in each run can use only a fraction of the total privacy budget, and make the results being even less accurate.

The second decision is how many iterations one runs. A large number of iterations causes too much noises being added. A small number of iterations may be insufficient for the algorithm to converge. Existing approaches fix a number. However, intuitively the number of rounds would depend on the available privacy budget ϵ. With a smaller privacy budget, one should run fewer number of rounds, to avoid the results being overwhelmed by too much noise.

How to choose these parameters has not been carefully considered in the literature. In the implementation of DPLloyd in PINQ [19], it is proposed to run 5 iterations, with equal privacy budget allocation for each round. In [24], comparison of GkM with DPLloyd was done by running DPLloyd with 20, 80, and 200 iterations, resulting in incorrect claim that GkM outperforms DPLloyd. Dwork [9] considered the possibility of running k-means clustering without knowing the number of rounds in advance, and proposed to use exponentially decreasing allocation of privacy budgets, i.e., $\frac{\epsilon}{2}$ in the first round, $\frac{\epsilon}{4}$ in the second round, and so on. This mostly likely results in deteriorating performance when the number of rounds increases. The main reason is that in later rounds, when one gets closer to the optimal value, it is desirable to have a larger privacy budget.

Below, we propose an approach to improve the selection of initial centroids for k-means clustering, design a general framework for deciding the number of iterations and apply it to improve DPLloyd. The improved version of DPLloyd is called DPLloyd-Impr.

3.1.3 Selecting Initial Centroids

The quality of initial centroids greatly affects the accuracy of DPLloyd. A poor choice of initial centroids can result in converging to a local optimum that is far from global optimum, or not converging after the given number of iterations. While many methods for choosing the initial points have been developed [26], these methods were developed without the privacy concern and need access to the dataset. In [22], k points at uniform random from the domain are chosen as the initial centroids. We have observed empirically that this can perform poorly in some settings, since some randomly chosen initial centroids are close together. We thus introduce an improved method for choosing initial centroids that is similar to the concept of sphere packing. Given a radius a, we randomly generate k centroids one by one such that each new centroid is of distance at least a away from every corner of the domain $[-r, r]^d$ and each new centroid is of distance at least $2a$ away from any existing centroid. When a randomly chosen point does not satisfy this condition, we generate another point. When we have failed repeatedly, we conclude that the radius a is too large, and try a smaller radius. We use a binary search to find the maximal value for a such that it is the process of choosing k centroids succeed. This process depends only on the shape of the overall domain and not where the data points are, and thus does not affect privacy.

3.1.4 Optimizing Rounds and Budget Allocation

We introduce the following general approach of determining the number of rounds and privacy budget allocation. Our approach depends on the ability to analyze the amount of noise introduced in each round, manifested as the mean squared error (MSE). Given this, one also specifies a threshold for the maximum MSE. The basic idea is to choose the number of iterations so that we try to ensure that each iteration's MSE is no larger than the threshold, and use smaller number of rounds if necessary. Below we show how to apply this idea to DPLloyd.

3.1.5 Error Study of DPLloyd

DPLloyd adds noises to each iteration of updating centroids. We now analyze the mean squared error (MSE) between noisy centroids and true centroids in one iteration.

Consider one centroid and its update in one iteration. The true centroid's i'th dimension should be $o_i = \frac{S_i}{C}$, where C is the number of data points in the cluster and S_i is the sum of i'th dimension coordinates of data points in the cluster. Consider the noisy centroid $\widehat{o}$; its i'th dimension is $\widehat{o_i} = \frac{S_i + \Delta S_i}{C + \Delta C}$, where ΔC is the noise added to the count and ΔS_i is the noise added to the S_i. The MSE is thus:

$$\mathsf{MSE}\left(\widehat{o}\right) = \mathsf{E}\left[\sum_{i=1}^{d}\left(\frac{S_i + \Delta S_i}{C + \Delta C} - \frac{S_i}{C}\right)^2\right] \tag{3}$$

Derivation based on the above formula gives the following proposition.

Proposition 1. *In one round of DPLloyd, the MSE is*

$$\Theta\left(\frac{(kt)^2 d^3}{(N\epsilon)^2}\right).$$

Proof. Let us first consider the MSE on the i-th dimension.

$$\begin{aligned}
&\mathsf{MSE}\left(\widehat{o_i}\right) = \mathsf{E}\left[\left(\tfrac{S_i+\Delta S_i}{C+\Delta C} - \tfrac{S_i}{C}\right)^2\right] \\
\approx &\mathsf{E}\left[\left(\tfrac{C\Delta S_i - S_i \Delta C}{C^2}\right)^2\right] \\
= &\tfrac{\mathsf{E}[(\Delta S_i)^2]}{C^2} + \tfrac{\mathsf{E}[S_i^2(\Delta C)^2]}{C^4} + \tfrac{2CS_i\mathsf{E}[\Delta S_i \Delta C]}{C^4} \\
= &\quad \tfrac{\mathsf{Var}(\Delta S_i)}{C^2} + \tfrac{S_i^2 \mathsf{Var}(\Delta C)}{C^4}
\end{aligned}$$

The last step holds, because ΔS_i and ΔC are independent zero-mean Laplacian noises and the following formulas hold:

$$\begin{cases}
\mathsf{E}[\Delta S_i \Delta C] = 0 \\
\mathsf{E}[(\Delta S_i)^2] = \mathsf{E}[(\Delta S_i)^2] - (\mathsf{E}[\Delta S_i])^2 = \mathsf{Var}\left(\Delta S_i\right) \\
\mathsf{E}[(\Delta C)^2] = \mathsf{E}[(\Delta C)^2] - (\mathsf{E}[\Delta C])^2 = \mathsf{Var}\left(\Delta C\right),
\end{cases}$$

where $\mathsf{Var}\left(\Delta S_i\right)$ and $\mathsf{Var}\left(\Delta C\right)$ are the variances of ΔS_i and ΔC, respectively.

Suppose that on average $\frac{|S_i|}{2r \cdot C} = \rho$, where $[-r, r]$ is the range of the i'th dimension. That is, ρ is the normalized coordinate of i-th dimension of the cluster's centroid. Furthermore, suppose that each cluster is about the same size, i.e., $C \approx \frac{N}{k}$. Then, $\mathsf{MSE}\left(\widehat{o_i}\right)$ can be approximated as follows:

$$\mathsf{MSE}\left(\widehat{o_i}\right) \quad \approx \quad \frac{k^2}{N^2}\left(\mathsf{Var}\left(\Delta S_i\right) + \left(2\rho r\right)^2 \cdot \mathsf{Var}\left(\Delta C\right)\right) \tag{4}$$

DPLloyd adds to each sum/count function Laplace noise $\mathsf{Lap}\left(\frac{(dr+1)t}{\epsilon}\right)$. Therefore, both $\mathsf{Var}\left(\Delta S_i\right)$ and $\mathsf{Var}\left(\Delta C\right)$ are equal to $\frac{2((dr+1)t)^2}{\epsilon^2}$.

From Equation (4) we obtain

$$\mathsf{MSE}\left(\widehat{o}_i\right) \approx \frac{k^2}{N^2}\left(\mathsf{Var}\left(\Delta S_i\right) + \left(2\rho r\right)^2 \cdot \mathsf{Var}\left(\Delta C\right)\right) \quad (5)$$

$$= 2(1+(2\rho r)^2)\left(\frac{kt(dr+1)}{N\epsilon}\right)^2. \quad (6)$$

As the noise added to each dimension is independent, from Equation 3 we know that the MSE is

$$\mathsf{MSE}\left(\widehat{o}\right) = \sum_{i=1}^{d}\mathsf{MSE}\left(\widehat{o}_i\right) \approx 2d(1+(2\rho r)^2)\left(\frac{kt(dr+1)}{N\epsilon}\right)^2 \quad (7)$$

When r is a small constant, this becomes $\Theta\left(\frac{(kt)^2 d^3}{(N\epsilon)^2}\right)$. □

Proposition 1 shows that the distortion to the centroid proportional to $t^2k^2d^3$, while inversely proportional to $(N\epsilon)^2$.

3.1.6 Optimizing Privacy Budget Allocation Within Each Round

An issue specific to DPLloyd and may not be shared by all iterative algorithms is that within each round of DPLloyd, the privacy budget needs to be divided among the count and the d sum queries. Suppose ϵ_0 is allocated to the count query, and ϵ_i is allocated to the sum query for the i-th dimension, for each $i = 1, 2, \ldots, d$. While all dimensions should be treated equally, i.e., $\epsilon_1 = \epsilon_2 = \ldots = \epsilon_d$, an interesting question is what should be the right value of $\frac{\epsilon_i}{\epsilon_0}$? The DPLloyd approach allocates the privacy budget according to the sensitivities of different queries; thus $\frac{\epsilon_i}{\epsilon_0} = r$, assuming that each dimension is normalized to $[-r, r]$. Different r values will result in different allocations of privacy budget.

We observe that the analysis in Section 3.1.5 calls for a fixed allocation of $\frac{\epsilon_i}{\epsilon_0}$, independent from how the data ranges are normalized. Plugging $\mathsf{Var}\left(\Delta S_i\right) = \frac{2r^2}{\epsilon_i^2}$ and $\mathsf{Var}\left(\Delta C\right) = \frac{2}{\epsilon_0^2}$ into Equation (5), one obtains

$$\sum_{i=1}^{d}\mathsf{MSE}\left(\widehat{o}_i\right) \approx \frac{k^2}{N^2}\sum_{i=1}^{d}\left(\mathsf{Var}\left(\Delta S_i\right) + \left(2\rho r\right)^2 \cdot \mathsf{Var}\left(\Delta C\right)\right)$$

$$= \frac{2r^2k^2}{N^2}\left(\sum_{i=1}^{d}\frac{1}{\epsilon_i^2} + \frac{4d\rho^2}{\epsilon_0^2}\right) \quad (8)$$

Minimization of the above subject to $\sum_{i=1}^{d}\epsilon_i + \epsilon_0 = z$ can be solved using the method of *Lagrange multipliers*, where z is the privacy budget allocated to the current round. The optimal proportion is

$$\epsilon_1 : \epsilon_2 : \cdots : \epsilon_d : \epsilon_0 = 1 : 1 : \cdots : 1 : \sqrt[3]{4d\rho^2} \quad (9)$$

To compute $\sqrt[3]{4d\rho^2}$, we need an estimation of ρ, the normalized coordinate of i-th dimension of the cluster's centroid. We note that $0 \leq \rho \leq 0.5$. If a cluster includes points perfectly balanced between the negative side and the positive side, then $\rho = 0$. If all points have r $(-r)$ as its i-th coordinate, then $\rho = 0.5$. We empirically compute ρ from 81 synthetic datasets that are not used for purpose of evaluation. We use $\rho = 0.225$ in this paper, and conjecture that it provides a good enough approximation for most scenarios.

We note that in the DPLloyd approach, if one chooses $r = 1$, i.e., normalizes each dimension to the range of $[-1, 1]$, one would allocate the privacy budget with a ratio of $\epsilon_i : \epsilon_0 = 1 : 1$, which is suboptimal in most cases.

3.1.7 Determining the Number of Rounds

Based on our analysis in Section 3.1.5, we make several observations. First, it makes no sense to run DPLloyd with a large number of rounds. From Proposition 1, the distortion on the centroid is on the order of $\Theta\left(\frac{t^2}{(N\epsilon)^2}\right)$. Thus, all one gets from running DPLloyd with too many rounds results large distortion on the cluster centroids. Second, one should dynamically determine the number of rounds based on parameters such as N and ϵ, since the distortion on the centroid is inversely proportional to $(N\epsilon)^2$.

By exploiting these observations, we propose a way to determine the number of iterations. We first determine a minimum privacy budget ϵ^m that needs to be allocated to each iteration (see below). Then, the privacy budget allocation across the iterations is decided by the following two cases. *Case 1:* $\epsilon \leq 2\epsilon^m$. In this case, the total privacy budget is inadequate. If we distribute it to more than 2 iterations, then as stated before the added noise in each round would easily dominate the centroid improvement. Therefore, we decide that DPLloyd runs for two iterations only, each with privacy budget of $\frac{\epsilon}{2}$. *Case 2:* $\epsilon > 2\epsilon^m$. In this case, the total privacy budget is able to meet the requirement of assigning minimum budget to each iteration. We require that the total number of iterations is at most 7. Thus, the total number of iterations $t^- = \min\{\frac{\epsilon}{\epsilon^m}, 7\}$, and the privacy budget allocated to each of them is $\frac{\epsilon}{t^-}$.

We now come to the calculation of ϵ^m. The intuition is that if the centroid improvement of one iteration is effective, then the MSE value should not be too big. We use the heuristic that the MSE of all the centroids improvement should be no more than $0.004 \cdot r^d$. It follows from Equation 8 that

$$\frac{2r^2k^3}{N^2}\left(\sum_{i=1}^{d}\frac{1}{\epsilon_i^2} + \frac{4d\rho^2}{\epsilon_0^2}\right) \leq 0.004r^d, \quad (10)$$

where $\sum_{i=0}^{d}\epsilon_i = \epsilon^m$. According to the optimized ratio in Equation 9, the privacy budget ϵ^m is distributed between ϵ_i's as follows:

$$\begin{cases} \epsilon_0 = \frac{\sqrt[3]{4d\rho^2}}{d+\sqrt[3]{4d\rho^2}}\epsilon^m \\ \epsilon_i = \frac{1}{d+\sqrt[3]{4d\rho^2}}\epsilon^m, \text{for } i = 1, 2, \ldots, d. \end{cases}$$

Plugging the above into Inequality 10 we can find the minimal ϵ^m value,

$$\epsilon^m = \left(\frac{500k^3}{N^2}\left(d + \sqrt[3]{4d\rho^2}\right)^3\right)^{1/2}. \quad (11)$$

For the Gowalla dataset, $\epsilon^m \approx 0.011$; for the Adult-num dataset, it is approximately equal to 0.096.

3.2 PGkM

PrivGene [34] is a general-purpose differentially private model fitting framework based on genetic algorithms. Given a dataset D and a fitting-score function $f(D, \theta)$ that measures how well the parameter θ fits the dataset D, the PrivGene algorithm initializes a candidate set of possible parameters θ and iteratively refines them by mimicking the process of natural evolution. Specifically, in each iteration, PrivGene uses the exponential mechanism [21] to privately select from the candidate set m' parameters that have the best fitting scores, and generates a new candidate set from the m' selected parameters by crossover and mutation. Crossover regards each parameter as an h-dimensional vector. Given two parameter vectors, it randomly selects a number $\bar{h}$ such that $0 < \bar{h} < h$ and splits each vector into the first $\bar{h}$ dimensions in the vector and the remaining $h - \bar{h}$ dimensions (the lower half). Then, it swaps the lower halves of the two vectors to generate two child vectors. These

vectors are then mutated by adding a random noise to one randomly chosen dimension.

In [34], PrivGene is applied to logistic regression, SVM, and k-means clustering. In the case of k-means clustering, the NICV formula in Equation 1, more precisely its non-normalized version, is used as the fitting function f, and the set of k cluster centroids is defined as parameter θ. Each parameter is a vector of $h = k \cdot d$ dimensions. Initially, the candidate set is populated with 200 sets of cluster centroids randomly sampled from the data space, each set containing exactly k centroids. Then, the algorithm runs iteratively for $\max\{8, (xN\epsilon)/m'\}$ rounds, where x and m' are empirically set to 1.25×10^{-3} and 10, respectively, and N is the dataset size.

We call the approach of applying PrivGene to k-means clustering PGkM, which is similarly to DPLloyd in that it tries to iteratively improve the centroids. However, rather than maintaining and improving a single set of k centroids, PGkM maintains a pool of candidates, uses selection to improve their quality, and crossover and mutation to broaden the pool.

By selecting multiple sets of centroids in each round and applying mutation, PGkM reduces the chance that the iterative process is stuck in a suboptimal solution. At the same time, doing this invariably slows down the converging process. At the same time, if one increases the number of iterations, each iteration becomes highly inaccurate. Thus whether PGkM is a suitable approach for a problem depends on whether the benefit of PGkM can compensate for the slow converging speed. Our experimental results in Section 4 show that for k-means clustering, this is not the case and PGkM performs poorly.

3.3 GkM

The k-means clustering problem was also used to motivate the *sample and aggregate* framework (SAF) for satisfying differential privacy, which was developed in [25, 31], and implemented in the GUPT system [24].

Given a dataset D and a function f, SAF first partitions D into ℓ blocks, then it evaluates f on each of the block, and finally it privately aggregates results from all blocks into a single one. Since any single tuple in D falls in one and only one block, adding one tuple can affect at most one block's result, limiting the sensitivity of the aggregation step. Thus one can add less noise in the final step to satisfy differential privacy.

As far as we know, GUPT [24] is the only implementation of SAF. Authors of [24] implemented k-means clustering and used it to illustrate the effectiveness of GUPT. We call this algorithm GkM. Given a dataset D, it first partitions D into ℓ blocks $D_1, D_2, \ldots, D_\ell$. Then, for each block D_b ($1 \le b \le \ell$), it calculates its k centroids $o^{b,1}, o^{b,2}, \ldots, o^{b,k}$. Finally, it averages the centroids calculated from all blocks and adds noise. Specifically, the i'th dimension of the j'th aggregated centroid is

$$o_i^j = \frac{1}{\ell}\sum_{b=1}^{\ell} o_i^{b,j} + \mathsf{Lap}\left(\frac{2(max_i - min_i)\cdot k \cdot d}{\ell \cdot \epsilon}\right), \qquad (12)$$

where $o_i^{b,j}$ is the i'th dimension of $o^{b,j}$, $[min_i, max_i]$ is the estimated output range of i'th dimension. One half of the total privacy budget is used to estimate this output range, and the other half is used for adding Laplace noise.

We have found that the implementation downloaded from [23], which uses Equation (12), performed poorly. Analyzing the data closely, we found that min_i and max_i often fall outside of the data range, especially for small ϵ. We slightly modified the code to bound min_i and max_i to be within the data domain. This does not affect the privacy, was able to greatly improve the accuracy. In this paper we use this fixed version.

Here a key parameter is the choice of ℓ. Intuitively, a larger ℓ will result in each block being very small and unable to preserve the cluster information in the blocks, and a smaller ℓ, on the other hand, results in large noise added. (Note the inverse dependency on ℓ in Equation (12). Analysis in [24] suggests to set $\ell = N^{0.4}$. Our experimental results, however, show that the performance is quite poor. We can analytically show why that is the case.

There are two sources of errors in GkM. The first is that the aggregation from the cluster centers obtained from different subsamples may not be accurate. The second is due to the added noise. The MSE due to the added noise is on the order of $\frac{k^2 d^2}{\ell^2 \epsilon^2}$. Compared with the MSE analysis of DPLloyd, they are comparable when $\ell \approx \frac{N}{t\sqrt{d}}$, that is, when each block contains only a small number of data points. It is unlikely that one could learn k centroids from such small subsamples. At the same time, if one chooses $\ell = N^{0.4}$, then MSE will be linear in $\frac{k^2 d^2}{N^{0.8}\epsilon^2}$, which is much larger than that of the DPLloyd method.

3.4 Non-interactive Approaches

Interactive approaches such as DPLloyd and GkM suffer from two limitations. First, often times the purpose of conducting k-means clustering is to visualize how the data points are partitioned into clusters. The interactive approaches, however, output only the centroids. In the case of DPLloyd, one could also obtain the number of data points in each cluster; however, it cannot provide more detailed information on what shapes data points in the clusters take. The value of interactive private k-means clustering is thus limited. Second, as the privacy budget is consumed by the interactive method, one cannot perform any other analysis on the dataset; doing so will violate differential privacy.

Non-interactive approaches, which first generate a synopsis of a dataset using a differentially private algorithm, and then apply k-means clustering algorithm on the synopsis, avoid these two limitations. In this paper, we consider the following synopsis method. Given a d-dimensional dataset, one partitions the domain into M equal-width grid cells, and then releases the noisy count in each cell, by adding Laplacian noise to each cell count.

The synopsis released is a set of cells, each of which has a rectangular bounding box and a (noisy) count of how many data points are in the bounding box. The synopsis tells only how many points are in a cell, but not the exact locations of these points. For the purpose of clustering, We treat all points as if they are at the center of the bounding box. In addition, these noisy counts might be negative, non-integer, or both. A straightforward solution is to round the noisy count of a cell to be a non-negative nearest integer and replicate the cell center as many as the rounded count. This approach, however, may introduce a significant systematic bias in the clustering result, when many cells in the UG synopsis are empty or close to empty and these cells are not distributed uniformly. In this case, simply turning negative counts to zero can produce a large number of points in those empty areas, which can pull the centroid away from its true position. We take the approach of keeping the noisy count unchanged and adapting the centroid update procedure in k-means to use the cell as a whole. Specifically, given a cell with center c and noisy count $\tilde{n}$, its contribution to the centroid is $c \times \tilde{n}$. Using this approach, in one cluster, cells who have negative noisy count can "cancel out" the effect of other cells with positive noise. Therefore, we can have better clustering performance.

For this method, the key parameter is M, the number of cells. When M is large, the average count per cell is low, and the noise

will have more impact. When M is small, each cell covers a large area, and treating all points as at the center may be inaccurate when the points are not uniformly distributed. We now describe two existing methods of choosing M and extend one of them.

3.4.1 MkM

Lei [16] proposed a scheme to release differentially private synopses tailored for the M-estimator. Given a d-dimensional dataset with N tuples, statistical analysis in [16] suggests that

$$M = \left(\frac{N}{\sqrt{\log(N)}}\right)^{\frac{2d}{2+d}} \tag{13}$$

We name the approach of applying the k-means clustering on this synopsis MkM.

3.4.2 UGkM

UG is a simple algorithm proposed in [27] for producing synopsis of 2-dimensional datasets that can be used to answer rectangular range queries (i.e., how many data points there are in a rectangular range) with high accuracy. The algorithm partitions the space into $M = m \times m$ equal-width grid cells, and then releases the noisy count in each cell. It is observed that for counting queries, a larger M value results in higher errors because more noises are added, and a smaller M value results in higher errors due to the fact that points within cells may be distributed nonuniformly, and queries including a portion of these cells may be answered inaccurately. To balance these two kinds of errors, it is suggested to set

$$m = \sqrt{\frac{N\epsilon}{10}}, \text{ or equivalently, } M = \frac{N\epsilon}{10} \tag{14}$$

It has been shown that UG performs quite well for answering rectangular range queries [27]. UG can be easily extended to d-dimensional dataset by setting $m = \sqrt[d]{M}$. We use UGkM to represent the UG-based k-means clustering scheme.

3.4.3 EUGkM

We now analyze the choice of M for higher-dimensional case. We use *mean squared error* (MSE) to measure the accuracy of est with respect to act. That is,

$$\mathsf{MSE}\,(est) = \mathsf{E}\left[(est - act)^2\right] = \mathsf{Var}\,(est) + (\mathsf{Bias}\,(est))^2,$$

where $\mathsf{Var}\,(est)$ is the variance of est and $\mathsf{Bias}\,(est)$ is its bias.

There are two error sources when computing est. First, Laplace noises are added to cell counts to satisfy differential privacy. This results in the variance of est. Since counting a cell size has the sensitivity of 1, Laplace noise $\mathsf{Lap}\left(\frac{1}{\epsilon}\right)$ is added. Thus, the noisy count has the variance of $\frac{2}{\epsilon^2}$. Suppose that the given counting query covers α portion of the total M cells in the data space. Then, $\mathsf{Var}\,(est) = \alpha\frac{2M}{\epsilon^2}$. Second, the given counting query may not fully contain the cells that fall on the border of the query rectangle. To estimate the number of points in the intersection between the query rectangle and the border cells, it assumes that data are uniformly distributed. This results in the bias of est, which depends on the number of tuples in the border cells. The border of the given query consists of $2d$ hyper rectangles, each being $(d-1)$-dimensional. The number of cells falling on a hyper rectangle is in the order of $M^{\frac{d-1}{d}}$. On average the number of tuples in these cells is in the order of $M^{\frac{d-1}{d}} \cdot \frac{N}{M} = \frac{N}{M^{\frac{1}{d}}}$. Therefore, we estimate the bias of est with respect to one hyper rectangle to be $\beta\frac{N}{M^{\frac{1}{d}}}$, where $\beta \geq 0$ is a parameter. We thus estimate $(\mathsf{Bias}\,(est))^2$ to be $2d\left(\beta\frac{N}{M^{\frac{1}{d}}}\right)^2$.

Summing the variance and the squared bias, it follows that

$$\mathsf{MSE}\,(est) = \alpha\frac{2M}{\epsilon^2} + \beta^2\frac{2dN^2}{M^{\frac{2}{d}}}.$$

To minimize the MSE, we set the derivative of the above equation with respect to M to 0. This gives

$$M = \left(\frac{N\epsilon}{\theta}\right)^{\frac{2d}{2+d}}, \tag{15}$$

where $\theta = \sqrt{\frac{\alpha}{2\beta^2}}$. We name the above extended approach as EUG (extended uniform griding approach). We use EUGkM to represent the EUG-based k-means clustering scheme.

4. PERFORMANCE AND ANALYSIS

Table 1: Descriptions of the Datasets.

Dataset	# tuples	# dims	# clusters
S1	5,000	2	15
Gowalla	107,091	2	5
TIGER	16,281	2	2
Image	34,112	3	3
Adult-num	48,841	6	5
Lifesci	26,733	10	3
Synthe	10,000 + O	[2, 10]	[2, 10]
Synthe-PT	10,000	[2, 10]	[2, 10]

O is # outliers and is uniformly sampled from $[0, 100]$.

In this section, we compare and analyze the performance of the six methods described in Section 3.

4.1 Evaluation Methodology

We experimented with six external datasets and two sets of syntheticly generated datasets. The first external dataset is a 2D synthetic dataset S1 [12], which is a benchmark to study the performance of clustering schemes. S1 contains 5,000 tuples and 15 Gaussian clusters. The Gowalla dataset contains the user checkin locations from the Gowalla location-based social network whose users share their checking-in time and locations (longitude and latitude). We sample one locaiton of each user ID and obtain a 2D dataset of 107,091 tuples. We set $k = 5$ for this dataset. The third dataset is a 1-percentage sample of road dataset which was drawn from the 2006 TIGER (Topologically Integrated Geographic Encoding and Referencing) dataset [4]. It contains the GPS coordinates of road intersections in the states of Washington and New Mexico. The fourth is Image [12], a 3D dataset with 34,112 RGB vectors. We set $k = 3$ for it. We also use the well known Adult dataset [1]. We use its six numerical attributes, and set $k = 5$. The last dataset is Lifesci. It contains 26,733 records and each of them consists of the top 10 principal components for a chemistry or biology experiment. As previous approaches [24, 34], we set $k = 3$. Table 1 summarizes the datasets. For all the datasets, we normalize the domain of each attribute to [-1.0, 1.0].

We generate two sets of synthetic datasets. The first set of synthetic datasets, which we call Synthe, is generated by using the clusterGeneration [28] R package. It is designed for generating cluster datasets with specified degree of separation which is a quantitative measure of closeness between any cluster and its nearest neighboring cluster. Besides, the clusterGeneration package can generate clusters with arbitrary diameters, shapes and orientations.

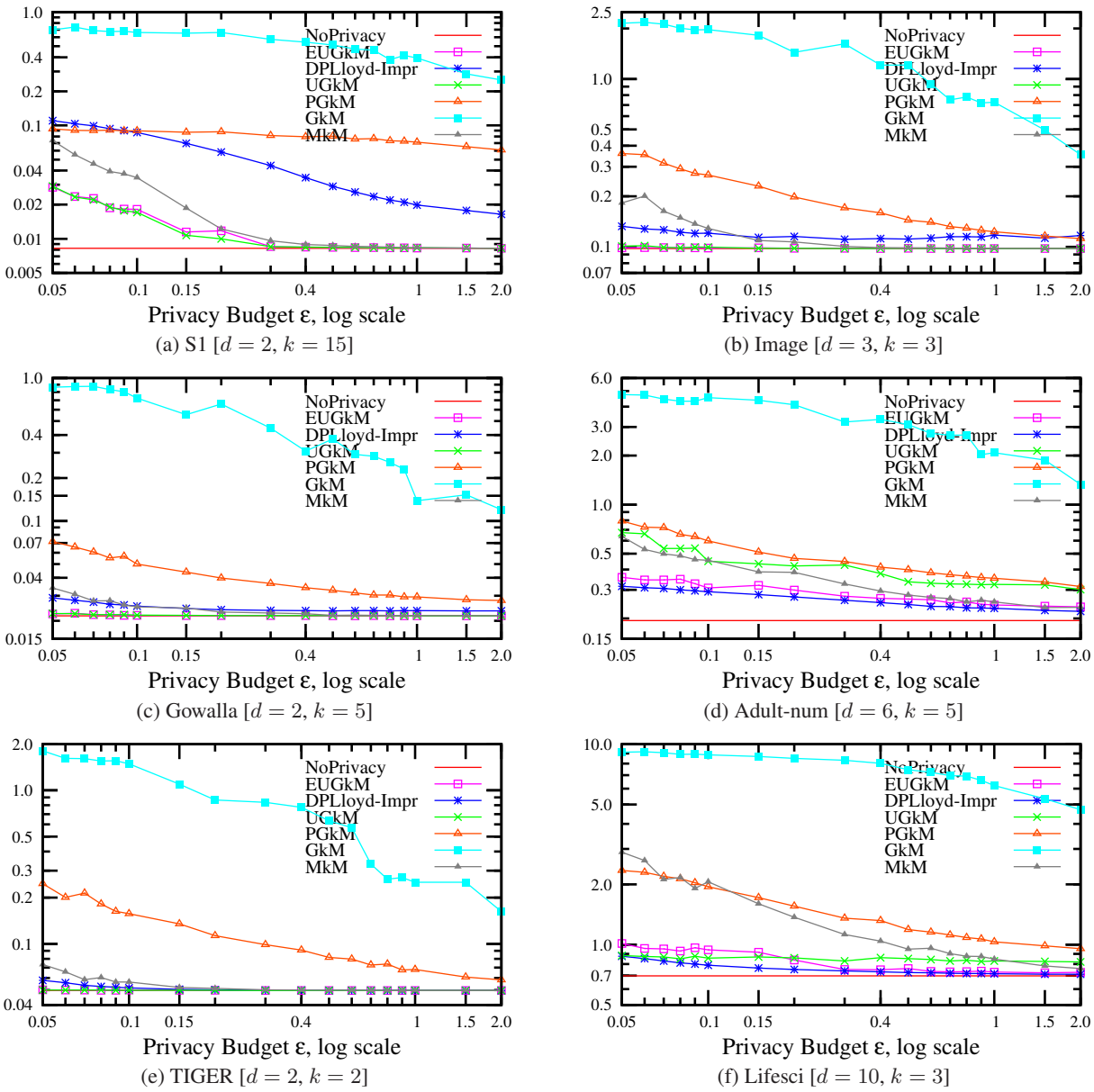


Figure 1: The comparison of DPLloyd-Impr, PGkM, GkM, EUGkM, UGkM and MkM by varying the privacy budget ϵ. x-axis: privacy budget ϵ in log-scale. y-axis: NICV in log-scale.

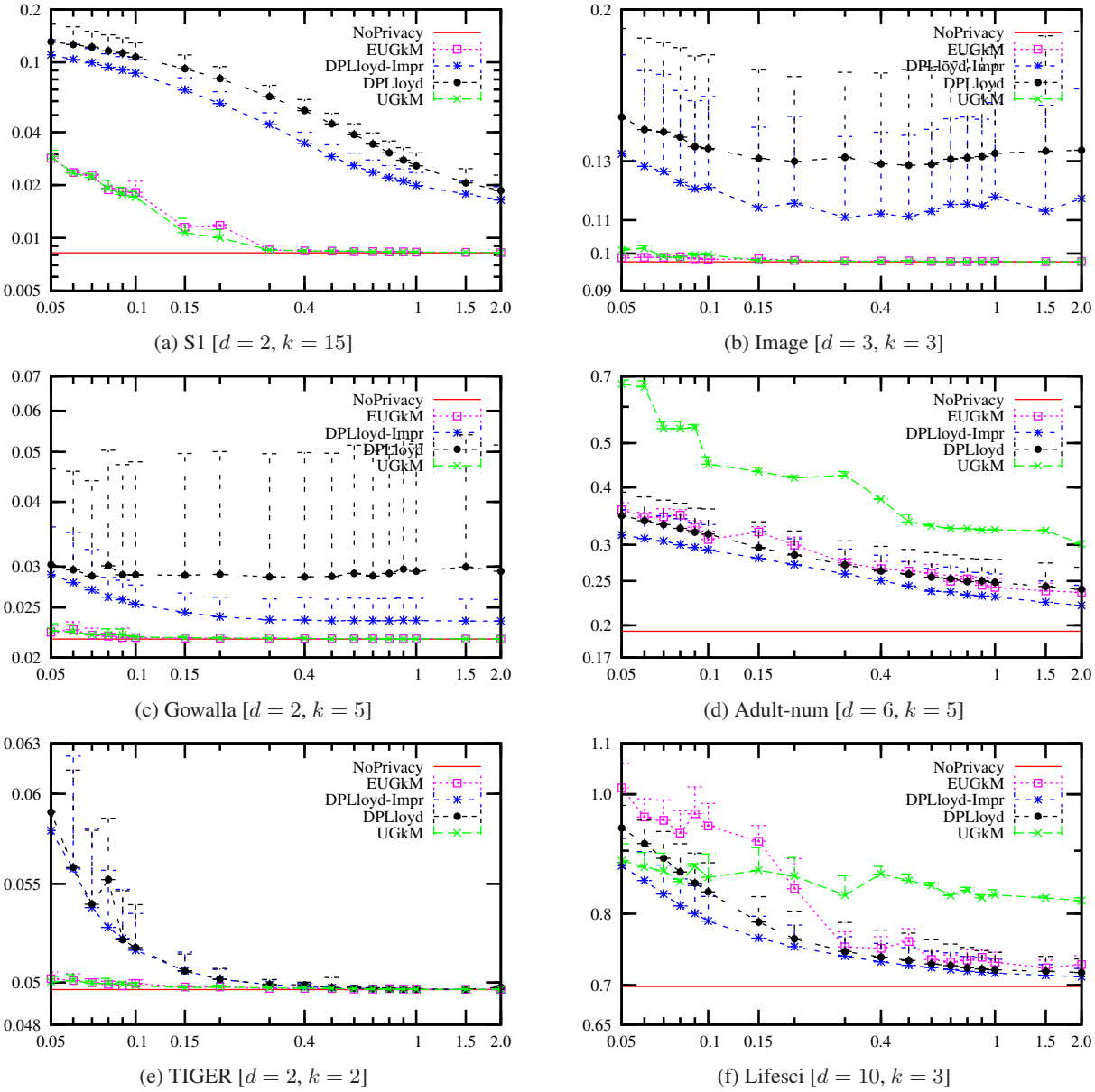
In this paper, we generate 81 dataset by varying k and d from 2 to 10. We fix the dataset size to 10,000 and distribute them into k clusters with size proportional to the ratio $1 : 2 : \ldots : k$. We also inject few outliers whose number is uniformly sampled from $[0, 100]$. For each dataset, we randomly sample its degree of separation from $[0.16, 0.26]$, which means from clusters with small overlapping to separated-but-close clusters.

The second set is mainly for tuning parameters of the EUGkM algorithm. We fix the dataset size to be 10,000, and vary k and d from 2 to 10 respectively. For each dataset, k well separated Gaussian clusters with equal size are generated. We call the second set of synthetic dataset as the Synthe-PT set, where PT stands for parameter tuning.

Implementations for DPLloyd and GkM were downloaded from [19] and [23], respectively. The source code of PGkM [34] was shared by the authors. We implemented EUGkM, UGkM and MkM.

Configuration. Each algorithm outputs k centroids $\mathbf{o} = \{o^1, o^2, \cdots, o^k\}$. The quality of the centroids $\mathbf{o}$ is evaluated by the Normalized Intra-Cluster Variance (NICV) (Eq.1).

We note that since both DPLloyd, EUGkM, UGkM and MkM use Lloyd-style iteration, they are affected by the choice of initial centroids. In addition, all algorithms have random noises added somewhere to satisfy differential privacy. To conduct a fair comparison, we need to carefully average out such randomness effects. GkM and PGkM do not take a set of initial centroids as input. GkM divides the input dataset into multiple blocks, and for each block invokes the standard k-means implementation from the Scipy pack-



(a) S1 [$d = 2$, $k = 15$]

(b) Image [$d = 3$, $k = 3$]

(c) Gowalla [$d = 2$, $k = 5$]

(d) Adult-num [$d = 6$, $k = 5$]

(e) TIGER [$d = 2$, $k = 2$]

(f) Lifesci [$d = 10$, $k = 3$]

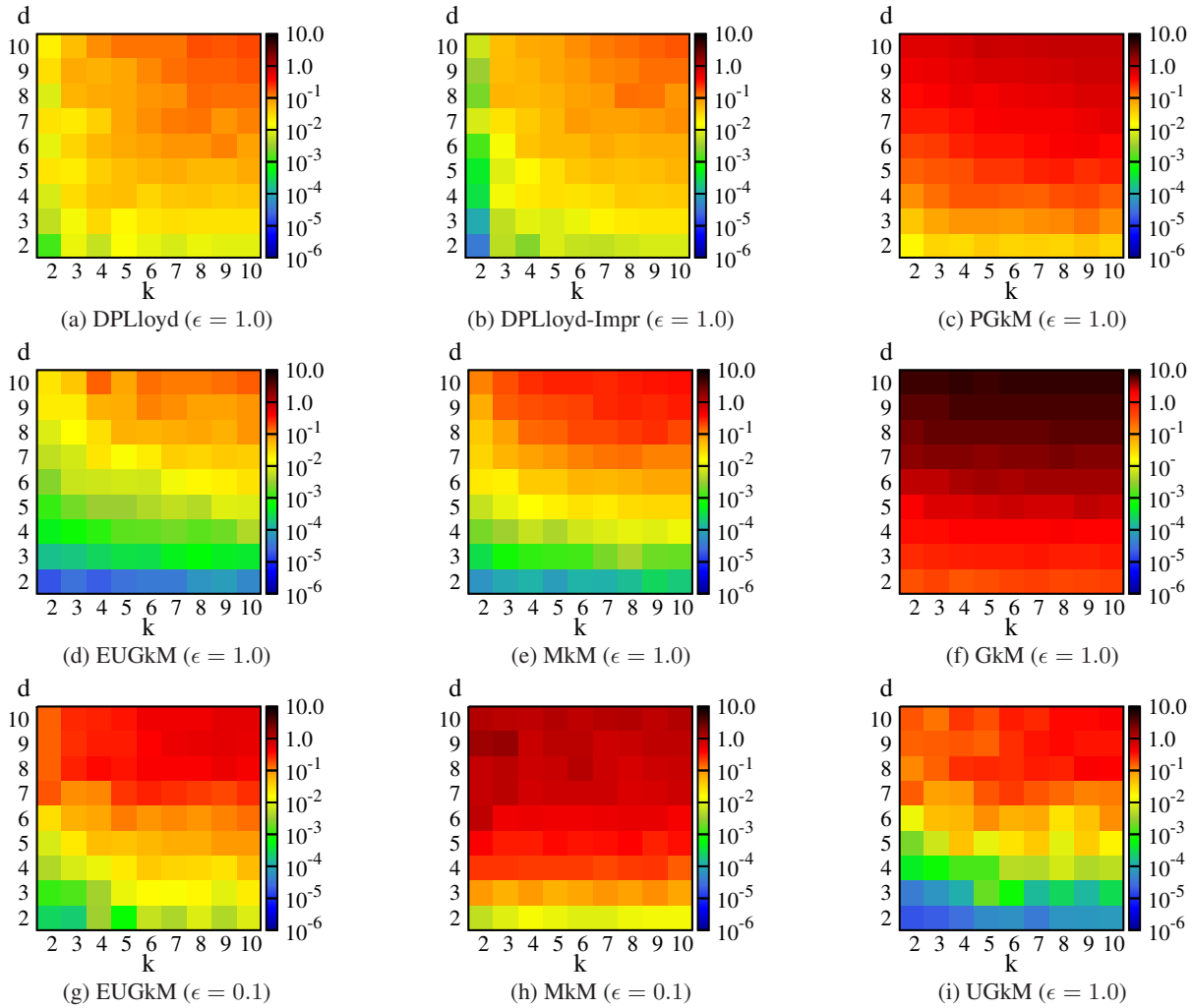
Figure 2: The close-up view of the comparison of DPLloyd-Impr, DPLloyd, EUGkM, and UGkM by varying the privacy budget ϵ. x-axis: privacy budget ϵ in log-scale. y-axis: NICV in log-scale.

age [30] with a different set of initial centroids to get the result, and finally aggregates the outputs for all the blocks. We run GkM and PGkM 100 times and report the average result.

DPLloyd-Impr generates 30 sets of initial centroids by using the proposed sphere packing method in Section 3.1.3. We run DPLloyd-Impr 100 times on each set of initial centroids, and report the average of the 3000 NICV values as the final evaluation of DPLloyd-Impr. For DPLloyd, we randomly generate 30 sets of initial centroids and use the same way to compute the averaged NICV values.

The non-interactive approach (EUGkM) has the advantage that once a synopsis is published, one can run k-means clustering with as many sets of initial centroids as one wants and choose the result that has the best performance relative to the synopsis. In our experiments, given a synopsis, we use the same 30 sets of initial centroids as those generated for the DPLloyd-Impr method. For each set, we run clustering and output a set of k centroids. Among all the 30 sets of output centroids, we select the one that has the lowest NICV relative to the synopsis rather than to the original dataset. This process ensures selecting the set of output centroids satisfies differential privacy. We then compute the NICV of this selected set relative to the original dataset, and take it as the resulting NICV with respect to the synopsis. To deal with the randomness introduced by the process of generating synopsis, we generate 10 different synopses and take the average of the resulting NICV values.

For EUGkM, we set the the parameter $\theta = 10$. We experimentally compare the EUGkM's performance on different θ choices and find that $\theta = 10$ for EUGkM works well in most cases. This parameter tuning for EUGkM is performed on the Synthe-PT dataset. Therefore, the following evaluation of EUGkM on the Synthe dataset



(a) DPLloyd ($\epsilon = 1.0$) (b) DPLloyd-Impr ($\epsilon = 1.0$) (c) PGkM ($\epsilon = 1.0$)

(d) EUGkM ($\epsilon = 1.0$) (e) MkM ($\epsilon = 1.0$) (f) GkM ($\epsilon = 1.0$)

(g) EUGkM ($\epsilon = 0.1$) (h) MkM ($\epsilon = 0.1$) (i) UGkM ($\epsilon = 1.0$)

Figure 3: The heatmap by varying k and d on the Synthe datasets with $\epsilon = 1.0$ and $\epsilon = 0.1$

strictly satisfies differential privacy, since the parameter is determined on an independent set of datasets.

As the baseline, we run standard k-means algorithm [18] over the same 30 sets of initial centroids generated in DPLloyd-Impr and take the minimum NICV among all the 30 runs.

Experimental Results. Figure 1 and Figure 2 report the results for the 6 external datasets. For these, we vary ϵ from 0.05 to 2.0 and plot the NICV curve for the methods mentioned in Section 3. This enables us to see how these algorithms perform under different privacy budgets.

Figure 3 reports the results on the Synthe datasets. For these, we fix $\epsilon = 1.0$ and $\epsilon = 0.1$ and report the difference of NICV between each approach and the baseline. This enables us to see the scalability of these algorithms when k and d increase.

Among interactive approaches, DPLloyd-Impr has the best performance in most cases. It also outperforms DPLloyd in most cases. For non-interactive approaches, both EUGkM and UGkM clearly outperform MkM, especially for small ϵ values. EUGkM and UGkM has close performance on the low dimensional datasets. As the dimensionality increases, the advantage of EUGkM to UGkM becomes obvious. Comparing DPLloyd-Impr and EUGkM (Figure 2), we observe that in the four low dimensional external datasets (S1, Gowalla, TIGER and Image), EUGkM clearly outperforms DPLloyd-Impr at small ϵ value and their gap becomes smaller as ϵ increases. However, in the two high dimensional datasets (Adult-num and Lifesci), DPLloyd-Impr outperforms EUGkM almost in all given privacy budgets. The similar observations can also be found in Figure 3.

Figure 3 also exhibits the effects of the number of clusters and the number of dimensions. The EUGkM's performance is more sensitive to the increase of dimension, while DPLloyd-Impr gets worse quickly as the number of clusters increases.

4.2 The Analysis of the GkM Approach

From Figures 1 and 3, it is clear that GkM is always much worse than others. There are two sources of errors for GkM. One is that GkM is aggregating centroids computed from the subsets of data, and this aggregation may be inaccurate even without adding noise. The other is that the noise added according to Equation (12) may be too large. We find that setting $\ell = N^{0.4}$ in GkM, which corresponds to block size of $N^{0.6}$, is far from optimal, as the error GkM is dominated by that from the noise, and is much higher than the error due to sample and aggregation. Detailed explanations are deferred to Appendix 8.1.

4.3 The Analysis of the PGkM Approach

PGkM is a stochastic k-means method based on genetic algorithms. A stochastic method converges to global optimum [15]. On the contrary, DPLloyd is a gradient descent method derived from the standard Lloyd's algorithm [18], which may reach local optimum. However, PGkM is still inferior to DPLloyd in Figure 1.

There are two possible reasons. First, a stochastic approach typically takes a 'larger' number of iterations to converge [15]. Detailed explanations are deferred to Appendix 8.2. The second reason is that the low privacy budget allocated to select a parameter (i.e., a set of k cluster centroids) from the candidate set. In each iteration PGkM selects 10 parameters, and the total number of iterations is at least 8. Thus, the privacy budget allocated to select a single parameter is at most $\epsilon/80$. Therefore, PGkM has reasonable performance only for big ϵ value.

4.4 The Analysis of the EUGkM, UGkM and MkM Approaches

The difference between of the three non-interactive methods, EUGkM, UGkM and MkM is the choice of grid size M. The EUGkM method sets it to $\left(\frac{N\epsilon}{10}\right)^{\frac{2d}{2+d}}$, the UGkM method sets it to $\left(\frac{N\epsilon}{10}\right)$ and the MkM method sets it to $\left(\frac{N}{\sqrt{\log(N)}}\right)^{\frac{2d}{2+d}}$. Figure 1 and Figure 3 show that the performance of UGkM and EUGkM are superior to that of MkM. An important reason is that MkM does not take ϵ as a factor in M. Thus, it is nonadaptive to the variation of ϵ. This explains why EUGkM and UGkM perform much better than MkM for small ϵ values. On the other hand, although UGkM considers the impact of the privacy budget ϵ, it does not produce large enough grids for the high dimensional data. This explains why EUGkM performs better on high dimensional data than UGkM.

4.5 Estimating the Number of Clusters.

In cluster analysis, an important problem is to estimate the number of clusters, which has a deterministic effect on the clustering results. Such problem becomes more prominent in the differential privacy setting, since the data analyst cannot access the private database as many times as she/he wants.

Our EUGkM approach can address this problem. Several heuristics and statistics [29, 32] have been proposed to determine the number of clusters k automatically. Suppose we have a list of candidate values of k and one statistics ϕ for determining the best k. Once an EUGkM synopsis is published, we evaluate ϕ for each candidate k value on this noisy synopsis. The k value with the best ϕ score will be selected for the following k-means clustering. All the operations are performed on the released EUGkM synopsis. So the estimation process satisfies the differential privacy. This is another advantage of the non-interactive approaches over the interactive approaches on the k-means clustering.

We also experimentally evaluate the above method on the six external datasets and on six privacy budget values. This method gives very accurate estimations on the k values under most of the privacy budget settings. We omit the experimental results for space reasons.

5. RELATED WORK

The notion of differential privacy was developed in a series of papers [7, 11, 3, 10, 8]. Several primitives for answering a single query differentially privately have been proposed. Dwork et al. [10] introduced the method of adding Laplacian noise scaled with the sensitivity. McSherry and Talwar [21] introduced a more general exponential mechanism.

Blum et al. [3] proposed a sublinear query (SuLQ) database model for interactively answering a sublinear number (in the size of the underlying database) of count queries differential privately. The users (e.g. machine learning algorithms) issue queries and get responses which are added laplace noises. They applied the SuLQ framework to the k-means clustering and some other machine learning algorithms. McSherry [22] built the PINQ (Privacy INtegrated Queries) system, a programming platform which provides several differentially-private primitives to enable data analysts to write privacy-preserving applications. These private primitives include noisy count, noisy sum, noisy average, and exponential mechanism. The DPLloyd algorithm, which we compare against in this paper, has been implemented using these primitives.

Nissim et al. [25, 31] propose the sample and aggregate framework (SAF), and use k-means clustering as a motivating application for SAF. This SAF framework has been implemented in the GUPT system [24] and is evaluated by k-means clustering. This is the GkM algorithm that we compared with in the paper. Dwork [9] suggested applying a geometric decreasing privacy budget allocation strategy among the iterations of k-means, whereas we use an increasing sequence. Geometric decreasing sequence will cause later rounds using increasingly less privacy budget, resulting in higher and higher distortion with each new iteration. Zhang et al. [34] proposed a general private model fitting framework based on genetic algorithms. The PGkM approach in this paper is an instantiation of the framework to k-means clustering.

Interactive methods for other data mining tasks have been proposed. McSherry and Mironov [20] adapted algorithms producing recommendations from collective user behavior to satisfy differential privacy. Friedman and Schuster [13] made the ID3 decision tree construction algorithm differentially private. Chaudhuri and Monteleoni [5] proposed a differentially private logistic regression algorithm. Zhang et al. [35] introduced the functional mechanism, which perturbs an optimization objective to satisfy differential privacy, and applied it to linear regression and logistic regression. Differentially private frequent itemset mining has been studied in [2, 17]. The tradeoffs of interactive and non-interactive approaches in these domains are interesting future research topics.

Most non-interactive approaches aim at developing solutions to answer histogram or range queries accurately [10, 33, 14, 6]. Dwork et al. [10] calculate the frequency of values and release their distribution differentially privately. Such method makes the variance of query result increase linearly with the query size. To address this issue, Xiao et al. [33] propose a wavelet-based method, by which the variance is polylogarithmic to the query size. Hay et al. [14] organize the count queries in a hierarchy, and improve the accuracy by enforcing the consistency between the noisy count value of a parent node and those of its children. Cormode et al. [6] adapted standard spatial indexing techniques, such as quadtree and kd-tree, to decompose data space differential-privately. Qardaji et al. [27] proposed the UG and AG method for publishing 2-dimensional datasets.

6. CONCLUSION AND DISCUSSIONS

We have improved the state-of-the-art on differentially private k-means clustering in several ways. We have introduced a non-interactive methods for differentially private k-means clustering and improved one interactive methods based on a systemized error analysis. Concerning the question of non-interactive versus interactive, the insights obtained from k-means clustering are as follows. The non-interactive EUGkM has clear advantage, especially when

the privacy budget ϵ is small. Considering the further advantage that non-interactive methods enable other analysis on the dataset, we would tentatively conclude that non-interactive is the winner in this comparison. We conjecture that this tradeoff will hold for many other data analysis tasks. We plan to investigate whether this holds in other analysis tasks.

7. ACKNOWLEDGMENTS

This paper is based upon work supported by the United States National Science Foundation under Grant CNS-1116991.

8. APPENDIX

8.1 Detailed Explanations for the Analysis of the GkM Approach

This section gives detailed explanations for the two sources of errors in the GkM approach as mentioned in the Section 4.2. We use the Figure 4 to show the effect of varying block size from around $N^{0.1}$ to N on the two sources of errors. In Figure 4, we show error from GkM, error from using the aggregation without noise (SAG), and error from adding noise computed by Equation 12) to the best known centroids (Noise). From the figure, it is clear that setting $\ell = N^{0.4}$, which corresponds to block size of $N^{0.6}$ is far from optimal, as the error GkM is dominated by that from the noise, and is much higher than the error due to sample and aggregation. Indeed, we observed that as the block size decreases the error of GkM keeps decreasing, until when the block size gets close to k. It seems that even though many individual blocks result in poor centroids, aggregating these relatively poor centroids can result in highly accurate centroids. This effect is most pronounced in the Tiger dataset, which consists of two large clusters. The two centroids computed from each small block can be approximately viewed as choosing one random point from each cluster. When averaging these centroids, one gets very close to the true centroids.

8.2 Detailed Explanations for the Analysis of the PGkM Approach

This section gives detailed explanations for the first reason why PGkM is still inferior to DPLloyd as mentioned in the Section 4.3. Generally, a stochastic approach typically takes a 'larger' number of iterations to converge [15]. Figure 5 compares the Lloyd's algorithm with Gene (i.e., the non-private version of PGkM without considering differential privacy). For Lloyd, we reuse the initial centroids generated in Section 4.1. Given a dataset, we run Lloyd on the 30 sets of initial centroids generated for the dataset, and report the average NICV. Generally, Gene overtakes Lloyd as the number of iterations increases and finally converges to the global optimum. However, Lloyd improves its performance much faster than Gene in the first few iterations, and converges to the global optimal (or local optimum) more quickly. For example, in the Image dataset, Lloyd reaches the best baseline after three iterations, while the Gene needs more than 10 iterations to achieve the same.

9. REFERENCES

[1] A. Asuncion and D. Newman. UCI machine learning repository, 2010.

[2] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *KDD*, pages 503–512, 2010.

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

[4] U. S. Census. Topologically integrated geographic encoding and referencing. http://www.census.gov/geo/maps-data/data/tiger.html.

[5] K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In *NIPS*, pages 289–296, 2008.

[6] G. Cormode, C. M. Procopiuc, D. Srivastava, E. Shen, and T. Yu. Differentially private spatial decompositions. In *ICDE*, pages 20–31, 2012.

[7] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.

[8] C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.

[9] C. Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, Jan. 2011.

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

[11] C. Dwork and K. Nissim. Privacy-preserving data mining on vertically partitioned databases. In *CRYPTO*, pages 528–544, 2004.

[12] P. Fränti. Clustering datasets. http://cs.joensuu.fi/sipu/datasets/.

[13] A. Friedman and A. Schuster. Data mining with differential privacy. In *KDD*, pages 493–502, 2010.

[14] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *Proc. VLDB Endow.*, 3(1-2):1021–1032, Sept. 2010.

[15] K. Kummamuru and M. N. Murty. Genetic k-means algorithm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 29(3):433–439, 1999.

[16] J. Lei. Differentially private m-estimators. In *NIPS*, pages 361–369, 2011.

[17] N. Li, W. Qardaji, D. Su, and J. Cao. Privbasis: Frequent itemset mining with differential privacy. *Proc. VLDB Endow.*, 5(11):1340–1351, July 2012.

[18] S. P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–136, 1982.

[19] F. McSherry. Privacy integrated queries (pinq) infrastructure. http://research.microsoft.com/en-us/downloads/73099525-fd8d-4966-9b93-574e6023147f/.

[20] F. McSherry and I. Mironov. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *KDD*, pages 627–636, 2009.

[21] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007.

[22] F. D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *SIGMOD*, pages 19–30, 2009.

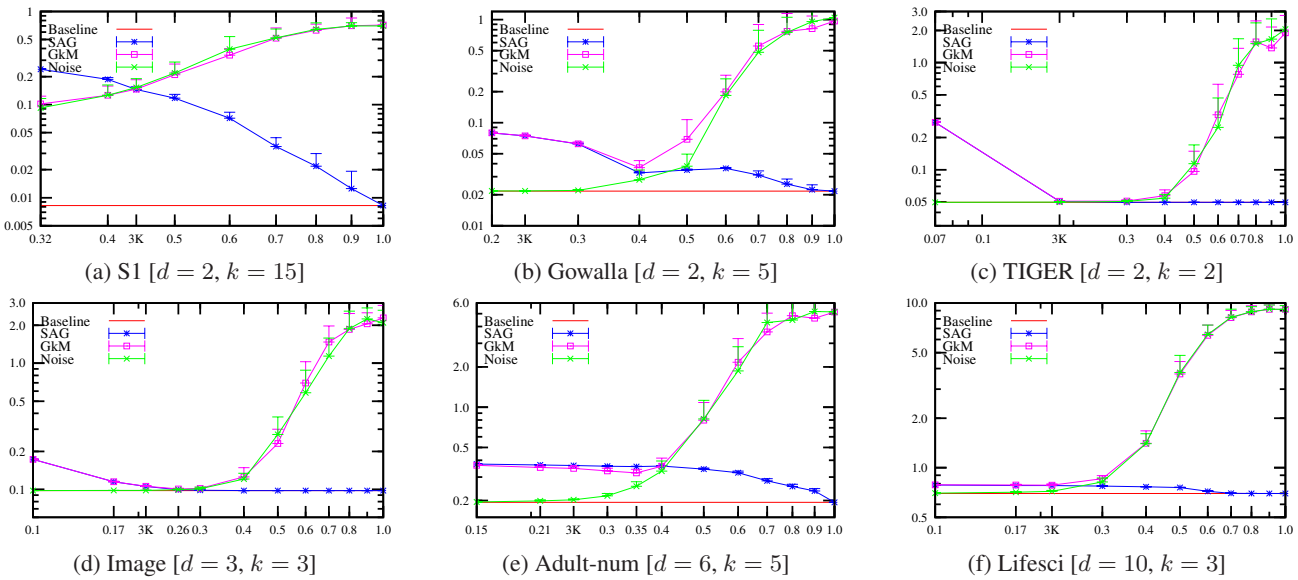
[23] P. Mohan. Gupt: a platform for privacy-preserving data mining. https://github.com/prashmohan/GUPT.

[24] P. Mohan, A. Thakurta, E. Shi, D. Song, and D. Culler. Gupt: Privacy preserving data analysis made easy. In *SIGMOD*, pages 349–360, 2012.

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

[26] J. M. Peña, J. A. Lozano, and P. Larrañaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recogn. Lett.*, 20(10):1027–1040, 1999.

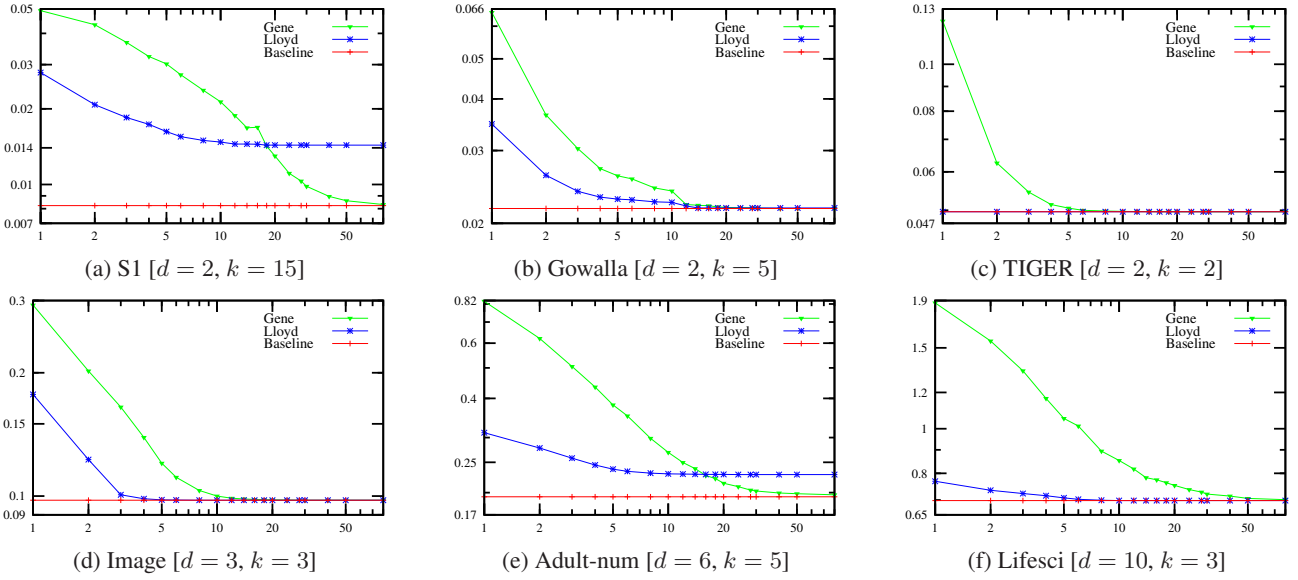
[27] W. H. Qardaji, W. Yang, and N. Li. Differentially private grids for geospatial data. In *ICDE*, pages 757–768, 2013.



(a) S1 [$d = 2$, $k = 15$] (b) Gowalla [$d = 2$, $k = 5$] (c) TIGER [$d = 2$, $k = 2$]

(d) Image [$d = 3$, $k = 3$] (e) Adult-num [$d = 6$, $k = 5$] (f) Lifesci [$d = 10$, $k = 3$]

Figure 4: The analysis of the GkM Approach. x-axis: block size exponent in log-scale, y-axis: NICV in log-scale.



(a) S1 [$d = 2$, $k = 15$] (b) Gowalla [$d = 2$, $k = 5$] (c) TIGER [$d = 2$, $k = 2$]

(d) Image [$d = 3$, $k = 3$] (e) Adult-num [$d = 6$, $k = 5$] (f) Lifesci [$d = 10$, $k = 3$]

Figure 5: The comparison of the convergence rate of the genetic algorithm based k-means and Lloyd algorithm. x-axis: number of iterations in log-scale, y-axis: NICV in log-scale.

[28] W. Qiu. clustergeneration: Random cluster generation (with specified degree of separation). http://cran.r-project.org/web/packages/clusterGeneration/index.html.

[29] S. Ray and R. H. Turi. Determination of number of clusters in k-means clustering and application in colour image segmentation. In *ICAPRDT'99*, pages 137–143, 1999.

[30] Scipy.org. Scientific computing tools for python. http://scipy.org/.

[31] A. Smith. Privacy-preserving statistical estimation with optimal convergence rates. In *STOC*, pages 813–822, 2011.

[32] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[33] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. *IEEE Trans. Knowl. Data Eng.*, 23(8):1200–1214, 2011.

[34] J. Zhang, X. Xiao, Y. Yang, Z. Zhang, and M. Winslett. Privgene: Differentially private model fitting using genetic algorithms. In *SIGMOD*, pages 665–676, 2013.

[35] J. Zhang, Z. Zhang, X. Xiao, Y. Yang, and M. Winslett. Functional mechanism: Regression analysis under differential privacy. *Proc. VLDB Endow.*, 5(11):1364–1375, July 2012.