PE-means: Improved Differentially Private k-means Clustering through Private Evolution

Thomas Humphries
University of Waterloo
thomas.humphries@uwaterloo.ca

Zinan Lin
Microsoft Research
zinanlin@microsoft.com

Sergey Yekhanin
Microsoft Research
yekhanin@microsoft.com

arXiv:submit/7655970 [cs.CR] 29 May 2026

Abstract

We study the problem of differentially private (DP) k-means clustering in Euclidean space. Previous solutions rely on summing the private data directly, which induces a sensitivity proportional to the domain. We introduce PE-means, an extension of the private evolution (PE) algorithm (an increasingly popular method for synthetic data generation), to the problem of k-means clustering. The key advantage of PE is that it only computes a private histogram with constant sensitivity to guide the evolution. Our adaptation of PE includes new evolutionary operators for clustering, as well as other algorithmic improvements of independent interest. Overall, PE-means achieves an average improvement of 20% in clustering loss over state-of-the-art baselines.

I. Introduction

The k-means clustering algorithm is a foundational tool in data science with many influential applications such as recommendation systems and healthcare data analysis [25], [16]. The objective is to group the data into k groups with similar features, allowing practitioners to interpret complex datasets. The challenge is that applying standard k-means algorithms to sensitive data risks the privacy of participants in the dataset. In the worst case, the k-means algorithm can publish the exact data record of an individual who is an outlier in the set. To address this, there has been a large body of work that considers the problem of differentially private k-means [31], [5], [4], [3], [1], [30], [25], [12], [15]. Differential privacy (DP) is a popular privacy definition that ensures changes in a single user's input does not have a significant effect on the output of the mechanism [10]. By adding calibrated randomization to the clustering process, existing approaches retain the benefits of k-means clustering while providing a formal privacy guarantee.

The challenge with DP is balancing the inherent trade-off between privacy and utility. Existing state-of-the-art approaches for DP k-means rely on summing the private data directly at some point in the process [4], [31], [1]. The sensitivity of (or largest impact any one user can have on) a sum query is the addition (or removal) of a point on the boundary of the domain. This means that to satisfy DP, noise must be added proportionally to the entire domain, significantly degrading the quality of the clustering. In recent work, an approach called FastLloyd managed to reduce this sensitivity from the whole domain to a smaller radius around each cluster through relative cluster updates [5]. However, the radius is still proportional to the data dimension, and FastLloyd introduces an additional error term due to the relative updates.

Private evolution (PE) is an increasingly popular technique for generating various modalities of synthetic data [22], [32], [21]. PE evolves a population of synthetic data using only inference API access to foundation models that generate and perturb the data in an evolutionary algorithm. A key component in PE's success is that it only uses the private data in a nearest neighbour voting scheme to choose the best samples generated so far. Specifically, each private data point only contributes a single vote to the algorithm in each iteration. Thus, adding or removing any private data point affects the output by a constant sensitivity of 1, regardless of the data domain. This gives PE a significant advantage over gradient-based techniques, which must add noise proportionally to the gradient domain. What makes PE truly impressive is that, despite getting much less signal from the private data, it still effectively traverses large and complex data domains by leveraging the robust optimization characteristics of evolution.

In this work, we introduce PE-means, an extension of the PE framework to the problem of k-means clustering. In this application, the population of PE contains randomly generated centroids that are iteratively evolved towards the centroids of the private data. Our design replaces PE's API calls to foundation models with a lightweight clustering initialization and a Lévy flight-based mutation operator. In addition to adapting PE to clustering, we also make several improvements to the algorithm that are of independent interest. Specifically, we address a critical issue with PE's selection technique where votes can be split between the best candidates, causing no representative from certain areas to be selected. We also reduce the impact of DP noise through better post-processing of the vote histogram and providing a mechanism to adaptively adjust the signal-to-noise ratio at no cost to privacy. Our

experimental evaluation shows PE-means outperforms state-of-the-art baselines over numerous real and synthetic datasets. Furthermore, we propose HDPE-means, a variant of PE-means that uses a similar dimensionality reduction to Balcan et al.'s [1], allowing PE-means to scale to higher dimensions. Overall, we observe up to 91% improvement in clustering loss over the state-of-the-art, with an average improvement of 20% across all the datasets evaluated.

II. Background

A. k-Means Clustering

Given a dataset D of size N and dimension d, the k-means problem aims to partition the dataset into k groups (clusters) $C = \{C_1, \ldots, C_k\}$, with each $C_i \subset D$, such that the following objective is minimized,

$$\arg\min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2 \tag{1}$$

where μ_i is the centroid of cluster i ($\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$). In this work, we assume the size and dimension of the dataset are public information, but we require any other information published from the dataset to satisfy differential privacy.

B. Differential Privacy

Differential privacy (DP) [10] adds randomness to the computation of aggregate statistics to protect the privacy of individuals. DP guarantees that an algorithm's output is approximately the same, regardless of the participation of any one user. More formally, differential privacy can be defined as follows.

Definition II.1 (Differential Privacy)**.** *A randomized algorithm $M : \mathcal{D} \mapsto \mathbb{R}$ is (ϵ, δ)-DP, if for any pair of neighbouring datasets $D, D' \in \mathcal{D}$, and for any $S \subseteq \mathbb{R}$ we have*

$$\Pr[M(D) \in S] \leq e^{\epsilon} \Pr[M(D') \in S] + \delta. \tag{2}$$

The privacy parameter ϵ defines how similar the outputs distributions must be, and δ allows a small chance of failure in the definition. We use the unbounded neighbouring definition, where datasets are neighbours if $|D \backslash D' \cup D' \backslash D| = 1$ (we allow for the addition or removal of a single data point). Arbitrary computations can be carried out on the output of a DP mechanism without affecting privacy due to the post-processing lemma [11]. Finally, DP is composed naturally with multiple runs of a mechanism. If we apply differentially private mechanisms sequentially, the privacy parameter composes through summation or more advanced methods [11].

Definition II.2 (Sensitivity)**.** *Let $f : \mathcal{D} \mapsto \mathbb{R}^k$. If $\mathbb{D}$ is a distance metric between elements of $\mathbb{R}^k$ then the $\mathbb{D}$-sensitivity of f is*

$$\Delta^{(f)} = \max_{(D, D')} \mathbb{D}(f(D), f(D')), \tag{3}$$

where (D, D') are pairs of neighbouring datasets.

We primarily use the ℓ_2 norm as the distance metric $\mathbb{D}$. To satisfy DP, we use Gaussian noise in this work and analyze it with Gaussian Differential Privacy (GDP) [6].

Definition II.3 (GDP [6])**.** *A mechanism M is said to satisfy θ-Gaussian Differential Privacy (θ-GDP) if it is G_θ-DP. That is,*

$$\mathcal{T}\big(M(D), M(D')\big) \geq G_\theta$$

for all neighbouring datasets D and D', where $\mathcal{T}$ is a trade-off function measuring the difficulty for attackers in identifying the presence of an individual data point and $G_\theta = \mathcal{T}\big(\mathcal{N}(0,1), \mathcal{N}(\theta,1)\big)$ (see Dong et al. [6] for specifics of the definition).

Naturally, the Gaussian Mechanism satisfies GDP.

Theorem II.1. *(Gaussian Mechanism GDP [6]) Define the Gaussian mechanism that operates on a statistic f as $M(D) = f(D) + \eta$, where $\eta \sim \mathcal{N}(0, (\Delta^{(f)})/\theta)$. Then, M is θ-GDP.*

We consider the composition of GDP over multiple runs of a mechanism.

Theorem II.2 (GDP Composition [6])**.** *The n-fold composition of θ_i-GDP mechanisms is $\sqrt{\theta_1^2 + \cdots + \theta_n^2}$-GDP.*

We also have a way to convert between GDP and DP.

Theorem II.3 (GDP to DP [6], [2]). *A mechanism is θ-GDP if and only if it is $\big(\epsilon, \delta(\epsilon)\big)$-DP for all $\epsilon \geq 0$, where*

$$\delta(\epsilon) = \Phi\Big(-\frac{\varepsilon}{\theta} + \frac{\theta}{2}\Big) - e^{\epsilon}\Phi\Big(-\frac{\varepsilon}{\theta} - \frac{\theta}{2}\Big).$$

In practice, we use the algorithm derived by Balle and Wang to solve this function for θ [2, Algorithm 1].

III. Methodology

Algorithm 1 **PE-means** based on Private Evolution (PE) [22]

Input: Private samples: $D = \{x_i,\}_{i=1}^{N} \subset \mathbb{R}^d$
Number of iterations: T
Number of generated samples: N_{syn} = k
Number of variations: L
Noise multiplier for the DP Nearest Neighbors Histogram: σ
Output: DP cluster centres $S'_T = \{\mu_1, \ldots, \mu_k\} \subset \mathbb{R}^d$

1: $S_0 \leftarrow$ RANDOM_API $(N_{\text{syn}} * L)$
2: **for** $t \leftarrow 1, \ldots, T$ **do**
3: $hist_t \leftarrow$ DP_NN_HISTOGRAM (D, S_{t-1}, σ) ▷ See Algorithm 2
4: ~~$P_t \leftarrow hist_t / \text{sum}(hist_t)$~~ ▷ ~~$P_t$ is a distribution on S_{t-1}~~
5: ~~$S'_t \leftarrow$ **rank samples** by P_t and draw N_{syn} samples from S_{t-1}~~
6: $S'_t \leftarrow$ WEIGHTED_K_MEANS(S_{t-1}, k=N_{syn}, weights=$hist_t$) ▷ Returns centroids
7: $S_t \leftarrow S'_t$ ▷ Save best samples
8: **if** $\text{sum}(hist_t^2)/(N\sigma^2) < 1.0$ **then**
9: $L \leftarrow L/2$
10: **end if**
11: **for** $i \leftarrow 1, \ldots, L$ **do**
12: $S_t \leftarrow S_t \cup$ VARIATION_API (S'_t)
13: **end for**
14: **end for**
15: **return** S'_T

Private Evolution (PE) is an emerging private synthetic data generation technique that evolves a set of synthetic samples using only inference API access to large machine learning models [22], [32], [21]. At a high level, PE starts from randomly generated synthetic data (using the RANDOM_API) and iteratively evolves toward high-quality synthetic data by creating new variations (using the VARIATION_API) and ranking the synthetic data that is closest to the private data. We build upon the text variant of PE called Aug-PE [32]. In Algorithm 1, we outline the baseline version of PE. Algorithm 1 also illustrates our approach, PE-means, highlighting our modifications to both improve PE and adapt it to solve k-means clustering.

A. Random API

The first step in PE is to generate an initial population, independently of the private data, using the RANDOM_API (Line 1). In previous variants of PE, this is done by querying a pre-trained foundation model (using information such as the label, which is assumed to be public). To solve clustering with PE, rather than creating a population of synthetic data samples, we create a population of centroids to evolve. We assume the only context available to generate the initial population is the shape of the data domain. For simplicity, we assume the domain is a hyper-sphere and use the same ℓ_2 bound R on its radius that other approaches need to bound sensitivity [31], [4], [1], [5]. However, in our case, the bound being too tight or loose can at most affect convergence speed and has no effect on the amount of noise added. A naive RANDOM_API for clustering would simply generate uniformly random data within the hyper-sphere. However, random points may group together, which reduces our coverage of the domain. Thus, we instead follow the work of Su et al. [31] and use the sphere packing method. Su et al.'s method starts with a given radius a (typically half the radius of the domain), and iteratively adds random samples that are at least a

away from the boundary and $2a$ from the samples added so far. If after a fixed number of retries, a random sample can not be found to meet these conditions, a is halved and the algorithm continues until k samples are found. We adapt Su et al.'s algorithm to consider an ℓ_2 bounded hyper-sphere rather than an ℓ_∞ bounded hypercube. The rest of the algorithm remains the same.

B. Selecting Candidates

Algorithm 2 **Modified** DP_NN_HISTOGRAM

Input: Private samples: D
Generated samples: $S = \{z_i\}_{i=1}^{n}$
Noise multiplier: σ
Distance function: $d(\cdot, \cdot)$
Output: DP nearest neighbours histogram on S

1: $histogram \leftarrow [0, \ldots, 0]$
2: **for** $x_{\text{priv}} \in D$ **do**
3: $\quad i = \arg\min_{j \in [n]} d(x_{\text{priv}}, z_j)$
4: $\quad histogram[i] \leftarrow histogram[i] + 1$
5: **end for**
6: $histogram \leftarrow histogram + \mathcal{N}(0, \sigma I_n)$
7: $histogram \leftarrow$ sort_descending$(histogram)$
8: $k \leftarrow \min\left\{\kappa \mid \sum_{j=1}^{\kappa} histogram[j] > N\right\}$
9: $histogram[j] \leftarrow 0$ **for all** $j > k$ ▷ Zero out noisy values
10: **return** $histogram$

The next step in PE is to select the best candidates from the population (either from the random initialization or previous iteration). To utilize the private data in this selection, we compute the DP_NN_HISTOGRAM (Line 3) described in Algorithm 2. The idea is that all private samples vote for their nearest neighbour in the population (Line 3 and 4). This creates a histogram of votes that is easy to privatize using Gaussian noise in Line 6. However, depending on the size of the population and progression of the evolution, this histogram can be quite sparse. Previous PE versions post-processed the histogram by zeroing out any noisy votes below a certain threshold. The challenge is how to choose the optimal threshold without spending privacy budget. We propose a modification based on the constraint that, before adding noise, the histogram of votes should be non-negative and sum to the size of the dataset (which we assume to be public). The maximum likelihood estimation (MLE) of the true votes given these constraints is found by projecting the noisy votes onto the set $\{x \in \mathbb{R} | x_i \geq 0, \sum_i x_i = N\}$. Duchi et al. [8] give an efficient algorithm for this projection by computing a subset of bins with the largest noisy counts such that after scaling and thresholding they sum to exactly N. For clustering, we do not need the sum constraint to be exact; a more critical issue is points far from the private data getting assigned a few false votes due to noise and pulling the centroids in the wrong direction. Thus, we give a simple approximation of the MLE algorithm that computes the minimum number of highest count buckets that exceed the sum constraint (Line 8) and zero out the remaining buckets (Line 9), without any additional scaling. Intuitively, we assume histogram buckets with large counts are likely to have dominated the noise.

After computing the private histogram, the next evolutionary task is to choose the best candidates from the population to survive to the next iteration based on this histogram. Aug-PE takes the approach of ranking candidates based on the number of votes they received in the histogram (Lines 4 to 5). The drawback to this approach is that choosing the most popular points does not always yield optimal clustering coverage due to what we call vote splitting. Consider a scenario where PE generates a point that is the nearest neighbour of many private dataset points. In the next iteration, PE may generate numerous new samples close to this point, causing the votes to be split between these new samples. Then, since all new points have a low count, an entire area of feature space may not be selected. We illustrate an example of this for clustering in Section IV-B. We argue that instead of just choosing points with the highest votes, we also need to choose one or two solutions from areas with highly split votes. Our key observation is that standard k-means clustering algorithms have a similar objective. Thus, we use the

population as a core set weighted by the histogram and run standard k-means to return the current best (Line 6).[1] We show that this technique is much more robust to vote splitting in Section IV-B.

C. Variation API

After selecting the best candidates, the next step is to continue exploring the space for better centroids by creating variations of the selected candidates. We note that we preserve the best candidates (referred to as elitism in evolutionary algorithms) in Line 7, so that if no variation improves, we maintain the current progress. In previous versions of PE, the VARIATION_API (Line 12) would be another API call to a foundation model. For example, Aug-PE [32] would randomly mask words and ask a large language model to fill them in with a high temperature. In the case of clustering, our VARIATION_API perturbs the real-valued candidate centroids directly. We find that mutation based on Lévy flights works well, as the distribution is heavier-tailed to encourage exploration. We follow Mantegna's algorithm [24] to generate a Lévy stable distribution parameterized by β which we denote by Lévy(β). To sample a value $\zeta \sim$ Lévy(β), we first we compute

$$\sigma_u = \left(\frac{\Gamma(1+\beta)\, \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \cdot \beta \cdot 2^{(\beta-1)/2}} \right)^{1/\beta}, \tag{4}$$

then draw independent samples $u \sim \mathcal{N}(0, \sigma_u I_d)$ and $v \sim \mathcal{N}(0, I_d)$ that we combine to get $\zeta = \frac{u}{|v|^{1/\beta}}$. Our variation API adds i.i.d. noise from this distribution to the existing centroids, scaled by a hyperparameter γ and the radius R, namely

$$\text{VARIATION\_API}(\mu_i) = \mu_i + \gamma R \zeta \tag{5}$$

where $\zeta \sim$ Lévy(β). Finally, we clip any mutated sample with ℓ_2 norm greater than R to the boundary. We experimentally set the hyperparameters for our variation API as $\beta = 1.75$ and $\gamma = 0.01$.

Following Aug-PE, we apply the variation API L times to each sample to increase exploration (Line 12). Tuning the L parameter is critical. If L is too small, the algorithm converges slowly, needing more iterations and privacy budget. Alternatively, if L is too large, the histogram becomes sparse, resulting in a poor signal-to-noise ratio and poor quality selections. While we can tune L as a hyperparameter (we show a heuristic for setting it in Section IV-C), the best setting for L varies greatly depending on the distribution of the private data and the current distribution of the candidates. To help avoid the more damaging case of a large L preventing meaningful selection, we add adaptive tuning of L during evolution. In Line 8, we estimate the noise to signal ratio (without using any privacy budget) by computing the ℓ_2 norm of the noisy histogram vs. the expected ℓ_2 norm of the noise. We argue if this value is less than 1, there is likely more noise than signal, and we reduce L by half (Line 9). We experimented with different thresholds and amounts of reduction but found these settings work well in practice.

D. High Dimensional Case

In high-dimensional domains (especially when k is small), variations become less effective due to the curse of dimensionality. To combat this, in Algorithm 3, we present a modified version of PE-means that is optimized for higher dimensions. We follow the work of Balcan et al. [1] and first use a Johnson-Lindenstrauss transform to project the private data to a low-dimensional space (Line 2). We then run PE-means as normal in this low-dimensional space in Line 3. To project the resulting cluster centres back to the original domain, we use noisy averaging. This can be thought of as a single iteration of DP Lloyd's algorithm [31] in the original domain, initialized with the low-dimensional cluster assignments. Specifically, all private data points are assigned a label using the low-dimensional centroids (Line 4 to 6). Then we compute the sum and count of all private points for a given label to obtain the centroids in high-dimensional space (Line 7 to 11). To ensure privacy, we add Gaussian noise to the sum and count of points in each cluster. We reduce the number of iterations for PE by two in Line 3, so we can use the same noise multiplier to noise the sum and count. We show in Section IV that HDPE-means outperforms PE-means in higher dimensions. The disadvantage of this approach is that, similar to related work [4], [1], we add noise proportional to R (the radius of the data domain) during the projection back to high dimensions. In future work, we will investigate alternate ways to scale the variation API while reducing this dependency.

[1]We could instead use a k-medians algorithm to select candidates strictly from the population satisfying a similar objective. However, we see no reason to enforce that constraint in this work, as the standard k-means algorithm can move us closer to the true centroids while making the selection.

Algorithm 3 High Dimensional PE-means (HDPE-means)

Input: Private samples: $D = \{x_i\}_{i=1}^N \subset \mathbb{R}^d$
Reduced dimension: p
Number of clusters: k
Number of PE iterations: T
Number of PE variations: L
Noise multiplier: σ
Domain radius: R
Output: DP cluster centres $S = \{\mu_1, \dots, \mu_k\} \subset \mathbb{R}^d$

1: Sample random projection matrix $G \sim \mathcal{N}(0,1)^{p \times d}$
2: $\bar{D} \leftarrow \{\bar{x}_i\}_{i=1}^N$ where $\bar{x}_i = \frac{1}{\sqrt{d}} G x_i$ ▷ Project data to $\mathbb{R}^p$
3: $\{\bar{\mu}_1, \dots, \bar{\mu}_k\} \leftarrow \mathsf{PE\text{-}Means}(\bar{D}, T-2, k, L, \sigma)$ ▷ Get centroids in $\mathbb{R}^p$ using Algorithm 1
4: **for** $i \leftarrow 1, \dots, N$ **do**
5: $\quad a_i \leftarrow \arg\min_{j \in \{1,\dots,k\}} \|\bar{x}_i - \bar{\mu}_j\|_2$ ▷ Assign labels in $\mathbb{R}^d$
6: **end for**
7: **for** $j \leftarrow 1, \dots, k$ **do**
8: $\quad z_j \leftarrow \sum_{i:a_i=j} x_i + \mathcal{N}(0, R\sigma I_n)$ ▷ Noisy sum in $\mathbb{R}^d$
9: $\quad n_j \leftarrow \sum_{i:a_i=j} 1 + \mathcal{N}(0, \sigma I_n)$ ▷ Noisy count
10: $\quad \mu_j \leftarrow \frac{z_j}{n_j}$
11: **end for**
12: **return** $S = \{\mu_1, \dots, \mu_k\}$

E. Privacy Analysis

The privacy analysis of PE-means follows from the original PE paper [22] as our changes do not affect noise addition in Algorithm 2 or use private data in any way. We restate the analysis here for completeness.

Theorem III.1. *PE-means as defined in Algorithm 1 satisfies (ϵ, δ)-DP.*

Proof. We break the proof into the same steps as the original paper [22]:

- **Step 1: Bounding the sensitivity (Definition II.2) of the DP Nearest Neighbors Histogram in Algorithm 2.** Each private sample only contributes one vote. If we add or remove one sample, the resulting histogram will change by 1 in the ℓ_2 norm. Therefore, the sensitivity is 1.
- **Step 2: Viewing each PE iteration as a Gaussian mechanism.** In Line 6 of Algorithm 2, we add i.i.d. Gaussian noise with standard deviation σ to each bin. This is the only part of the algorithm that touches the private dataset.
- **Step 3: Viewing the entire PE algorithm as T adaptive compositions of Gaussian mechanisms.** This holds because PE simply applies Algorithm 2 T times sequentially.
- **Step 4: Viewing the entire PE algorithm as one Gaussian mechanism with noise multiplier $\sigma/\sqrt{T}$.** Using the composition theorem (Theorem II.2) from Dong et al. [6, Corollary 2], we get that T applications of a $1/\sigma$-GDP algorithm are $\sqrt{T}/\sigma$-GDP.
- **Step 5: Computing DP parameters ϵ and δ.** The problem is simply computing a σ such that (ϵ, δ)-DP iff $\sqrt{T}/\sigma$-GDP, for which we apply the algorithm of Balle and Wang [2].

□

Our high-dimensional extension of PE includes additional use of the private data in the projection, and thus we state and prove its privacy.

Theorem III.2. *HDPE-means as defined in Algorithm 3 satisfies (ϵ, δ)-DP.*

Proof.

- **Step 1: Composition accounting.** We first assume that PE is called with two fewer iterations than the σ was computed for in Line 3. Applying Theorem III.1 we get that the run of PE is private with two extra applications of a Gaussian mechanism with standard deviation of σ remaining.

- **Step 2: Assignment as post-processing.** After calling PE, the assignment step (Line 4 to 6) uses the published centroids from the previous iteration (post-processing) to divide the dataset into clusters. We can then apply parallel composition over each of the clusters. Thus, we can focus on the privacy cost of a single cluster for the remainder of the proof.
- **Step 3: Privacy of the Gaussian mechanisms.** For each cluster, we apply the Gaussian mechanism twice. The first is to compute a sum which has an ℓ_2 sensitivity of R since adding or removing a point can at most add the largest possible vector in the domain. The second is to compute the count, which has an ℓ_2 sensitivity of 1, as a data point can be counted at most once. Applying Theorem II.1, the result follows.

□

IV. Experiments

A. Experimental Setup

a) Datasets: We use a combination of real and synthetic datasets from various sources. Most of the real datasets as well as the G2 datasets come from the clustering datasets repository [14] or the UCI machine learning repository [7]. For Birch2 [34], we take $25,000$ random samples from the dataset of $100,000$. Gas [7], Letter [7], and MNIST [20] are included for comparability with Chang and Kamath [4]. For MNIST, we train a LeNet5 model and use neural representations following Chang and Kamath [4]. We use some synthetic datasets generated by Diaa et al. [5] that were generated with the `clusterGeneration` R package [29]. Finally, we include synthetic datasets generated with the `make_blobs` function from Sklearn that generate isotropic Gaussian clusters. Table I includes a complete list of datasets and their sizes.

b) Baselines: Our baselines are selected from state-of-the-art DP k-means algorithms with publicly available implementations. The work of Chang and Kamath (which we denote by *Google*) is perhaps the most popular approach [4]. Chang and Kamath use a locality-sensitive hash tree to create a private coreset, upon which the non-private k-means algorithm is applied. Because Google's method is one of the most recent works, we also evaluate the approaches it compares to. The first is IBM's differential privacy library [17] (which we denote by *DP-Lib*) that implements an improved version of Su et al.'s algorithm [31]. Su et al. apply noise to each step of Lloyd's clustering algorithm [23] with various optimizations such as the sphere packing initialization. The second is the work of Balcan et al. (which we denote by *Icml17*) that first projects the data to a low-dimensional space before recursively dividing the space into cubes. They then apply a k-medians-style swapping algorithm to choose the best centres before projecting the result back into the high-dimensional space through noisy averaging. We note that both DP-Lib and Icml17 work in the pure differential privacy model ($\delta = 0$). The final related work is a recent improvement over Su et al.'s algorithm by Diaa et al. [5], which we denote by *FastLloyd*. FastLloyd modifies Su et al.'s work by using Gaussian noise and computing cluster updates relative to the previous iteration, thereby reducing the sensitivity.

c) Implementation Details: We normalize all datasets to have a max ℓ_2 norm of 1, for a clean presentation of metrics over different datasets. We follow Chang and Kamath and centre each dataset first by subtracting the mean [4]. Then, also following Chang and Kamath, we non-privately compute the ℓ_2 and ℓ_∞ sensitivity bounds on each dataset. In practice, this bound should be computed privately or derived from public information about the attributes. However, for the sake of comparison, we give all approaches an equal advantage by providing a tight bound on the domain. Each experiment is repeated 50 times over different random seeds, and we report the average. All shaded areas represent the 95% confidence interval of the mean of the results. We fix the privacy parameter $\delta = 1/N^{1.1}$, following the recommendation that the failure probability should be less than $1/N$ [11]. The primary loss we consider is the normalized k-means loss.

$$\text{Loss} = \frac{1}{N} \sum_{i=1}^{N} \min_{j=1}^{k} \|x_i - \mu_j\|^2 \tag{6}$$

We also compute each method's performance over different privacy budgets using the area under the curve (AUC) of the loss values against $\epsilon \in \{0.25, 0.5, 1.0, 2.0, 4.0\}$, computed via the trapezoidal rule:

$$\text{AUC} = \sum_{i=1}^{n-1} \frac{\text{Loss}_i + \text{Loss}_{i+1}}{2} \cdot (\epsilon_{i+1} - \epsilon_i) \tag{7}$$

B. Vote Splitting Example

In Section III-B, we described the issue of vote splitting in PE's selection. In this section, we illustrate the phenomenon with a toy example. We generate a two-dimensional random dataset with $k = 4$ using the `make_blobs` function and set $\epsilon = \infty$. In Figure 1, we plot four iterations of PE (one per column) and show the difference between the previous top-k voting (top row of plots) and our weighted k-means selection (bottom row of plots). The plots include the private data in grey, the population of PE in blue, and the selected points for each method in red.

In the case of the top-k voting (first row), we observe that the initial iteration chooses points close to the four true clusters in the dataset. However, in the second iteration, there are many good candidates in the lower cluster, splitting the vote and causing all selected points to be in the other three clusters. Then, in the third iteration, the lower cluster dies out and the vote becomes split in the upper cluster. The clusters do not recover in the fourth iteration, as it will now take PE numerous iterations to move back towards those areas. In contrast, the bottom row of plots shows PE maintaining and refining four healthy clusters throughout the iterations. We note that in this toy example, PE has essentially converged in the first iteration, which further highlights the challenge with top-k voting.

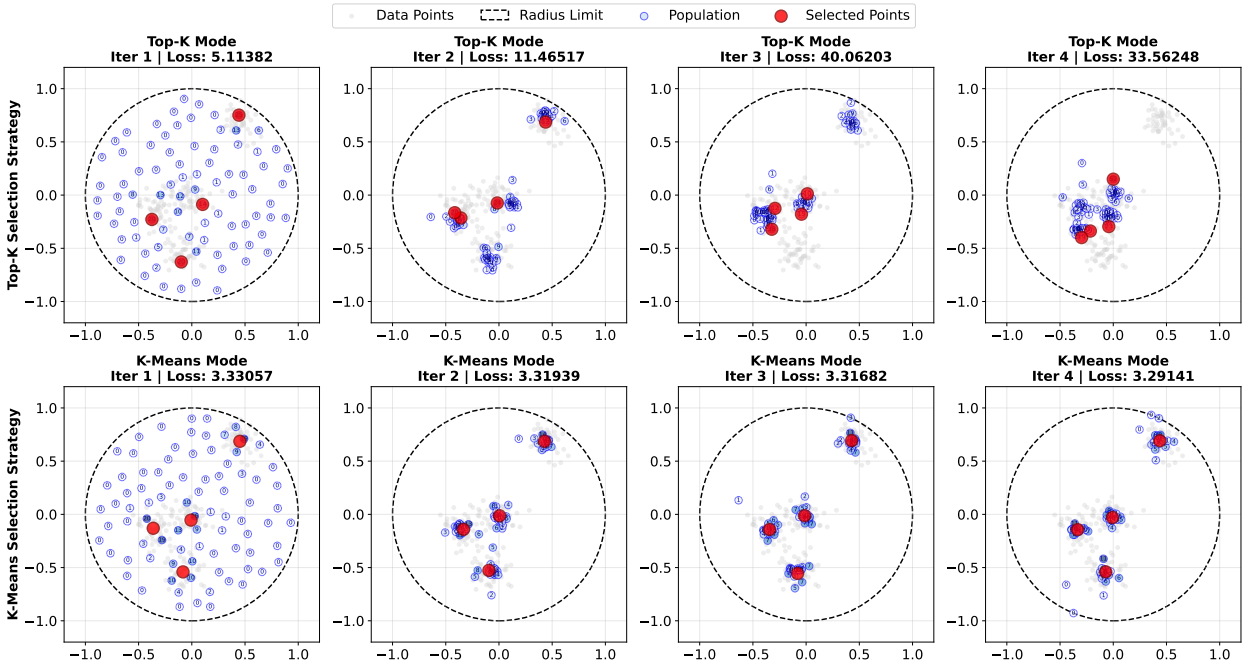

Fig. 1: A visualization of the samples generated and selected by PE in each iteration, comparing the previous top-k selection approach to our proposed k-means approach with $\epsilon = \infty$. The k-means approach effectively mitigates the vote splitting problem and keeps selected points close to the true clusters.

C. Hyperparameter Tuning

Two of the most influential hyperparameters for PE-means are the number of iterations T and the number of variations L. For a fixed privacy budget, if the number of iterations is too high, the amount of noise used in each iteration grows too large, preventing meaningful selections. If the algorithm executes too few iterations, it will not converge. Similarly, if L is too large, the histogram can become sparse and the votes will not dominate even the smallest amount of noise. If L is too small, then the chance of PE-means finding points closer to the private data is reduced, slowing convergence. We recall that our adaptive reduction of L (Line 9 of Algorithm 1) protects against the case when L is causing the noise to dominate the signal, but it is still beneficial to set a good starting point.

The only public parameters we can use to estimate the difficulty of a dataset in the private setting are its dimensions. In our initial testing, we observed that the number of iterations is influenced by the dimension of the dataset. Namely, in higher-dimensional space, PE needs more steps due to the curse of dimensionality. We also observed that the sparsity of the histogram for a given L is highly influenced by the number of values in a dataset

that can vote. To investigate this more rigorously, we conduct an experiment where we vary L and T and plot the values against the dataset size and dimension. We plot the results in Figure 2, where $\epsilon = 1$. We scatter the values of the top-3 best performing parameters for each dataset and draw a line of best fit using the LOWESS method. While the top-left plot reveals no useful relationship between the number of variations L and the number of dimensions d, the top-right plot shows a clear relationship between the dataset size N and L. We approximate this relationship with the heuristic $L = max(N/5, 4)$. Similarly, we see a clear relationship between the dimension d and number of iterations T in the bottom-left plot, which we approximate with the heuristic $T = max(4\sqrt{d}, 1)$. Finally, we see no clear relationship between the N and T in the bottom-right plot. In our experiments, we never encounter the constant cases in the max terms, but include them to ensure a robust implementation.

To extend these heuristics to also account for the privacy budget ϵ, we experimented with scaling them by factors such as $\epsilon, 0.5\epsilon, 2\epsilon, \sqrt{\epsilon}$ (since both parameters should decrease with smaller epsilon). We find that for values of $\epsilon < 1$, no modification improves over the baseline, as our adaptive strategy increases the signal-to-noise ratio for any dataset that needs it (and not all do). However, when $\epsilon > 1$, we find it is best (taking the median performance over all datasets) to scale the number of iterations by ϵ ($T = max(4\epsilon\sqrt{d}, 1)$).

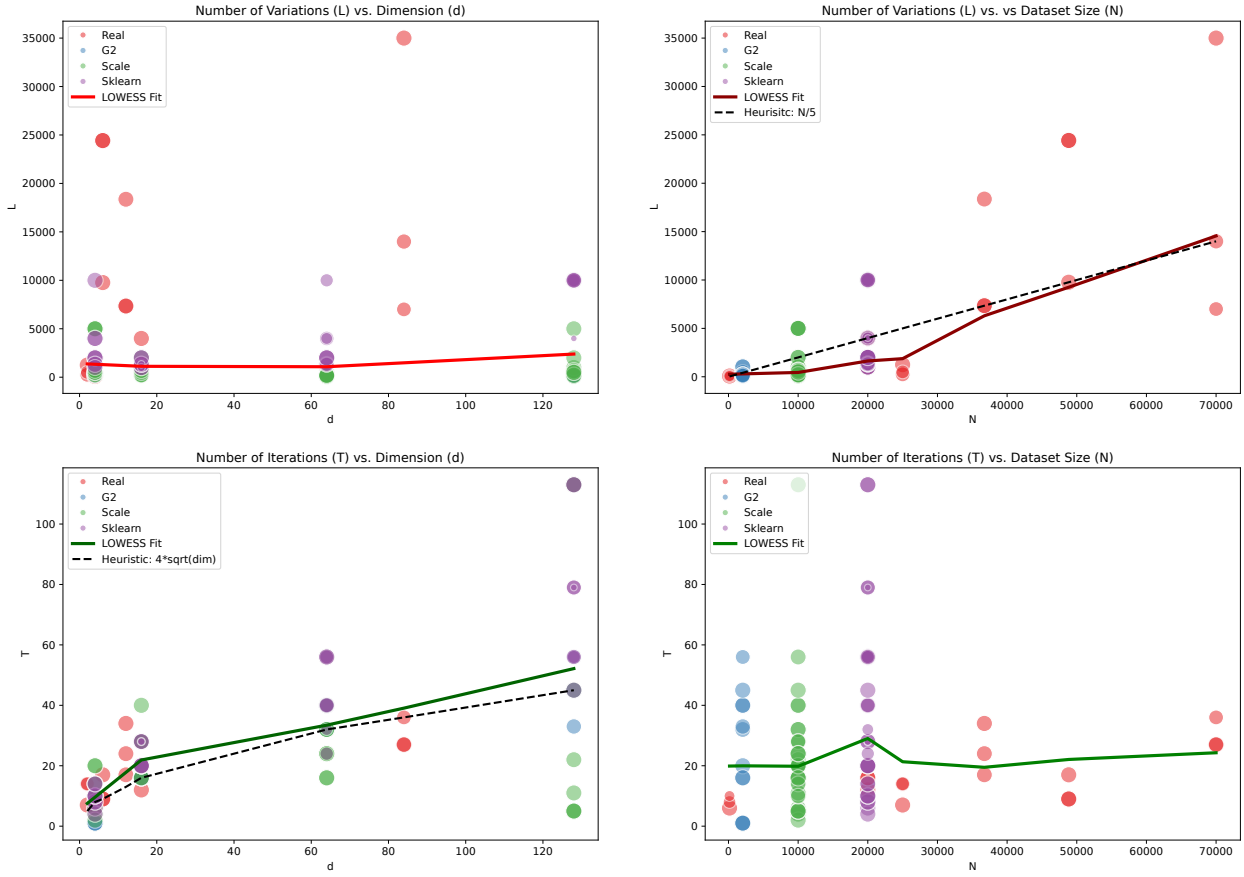

Fig. 2: A scatter plot of top-3 parameter configurations of L and T for each dataset, showing a clear relationship between L and N and between T and d that are captured by our heuristics.

D. Utility Benchmark

We compare PE-means and HDPE-means to the baselines over all datasets and summarize the results in Table I. The metric we use is the AUC of the k-means loss (a summary of performance over a collection of epsilons, with smaller values being best). We highlight the approach with the lowest loss AUC in green. HDPE-means is only included when the dimension of the dataset is greater than 16, as that is the dimensionality we project down to. Overall, PE-means and HDPE-means perform the best over most datasets, with FastLloyd doing well on the g2_4 dataset, and Icml17 doing well in the higher-dimensional scale datasets. We give the percentage of improvement

TABLE I: A summary of results over all datasets (real datasets highlighted in bold) showing the AUC of the clustering loss across ϵ values with the best approach highlighted for each dataset. The final column shows the relative improvements of PE-based techniques over the state-of-the art baseline for that dataset, with an average improvement of 20% over all datasets for PE-based approaches.

Dataset	N	d	k	Non-Priv	PE-means	HDPE-means	Google	FastLloyd	DP-Lib	Icml17	PE Improv.
birch2	25000	2	100	0.0000	**0.0003**	-	0.0658	0.0031	0.0124	0.0031	+90.92%
iris	150	4	3	0.0363	**0.2894**	-	0.6892	0.3979	1.3048	0.5507	+27.28%
adult	48842	6	3	0.0015	**0.0056**	-	0.0076	0.0155	0.0213	0.0066	+14.57%
mnist	70000	84	10	0.0522	0.2920	**0.2169**	0.2173	0.5822	0.5315	0.2375	+0.16%
letter	20000	16	26	0.0662	**0.2852**	-	0.3187	0.3963	0.4341	0.3247	+10.50%
gas	36733	12	6	0.0274	**0.1081**	-	0.1121	0.1561	0.1761	0.1168	+3.60%
g2_4	2048	4	2	0.1240	0.4761	-	0.5465	**0.4666**	0.5468	0.4823	-2.05%
g2_16	2048	16	2	0.2047	**0.7800**	-	0.8391	0.8687	1.1816	0.8116	+3.89%
g2_64	2048	64	2	0.2922	1.1538	**1.1295**	1.2478	1.9647	2.6608	1.1759	+2.11%
g2_128	2048	128	2	0.3609	1.7237	**1.4055**	1.5906	2.6425	4.4632	1.5057	+6.65%
scale_4_4	10000	4	4	0.0219	**0.0814**	-	0.1190	0.1272	0.1381	0.1148	+29.08%
scale_4_16	10000	16	4	0.0803	**0.3147**	-	0.3392	0.3721	0.4114	0.3224	+2.41%
scale_4_64	10000	64	4	0.1324	0.5326	0.5340	0.5430	0.5863	0.6622	**0.5113**	-4.18%
scale_4_128	9999	128	4	0.1528	0.6179	0.5940	0.5987	0.6282	0.7481	**0.5802**	-2.38%
scale_16_4	10000	4	16	0.0080	**0.0335**	-	0.0555	0.0864	0.0976	0.0624	+39.63%
scale_16_16	10000	16	16	0.0563	**0.2452**	-	0.2880	0.3624	0.3778	0.2715	+9.68%
scale_16_64	10000	64	16	0.1286	0.5884	0.5757	0.5823	0.6371	0.6949	**0.5404**	-6.53%
scale_16_128	10001	128	16	0.1443	0.6131	0.6023	0.5992	0.6294	0.7612	**0.5842**	-3.09%
scale_64_4	10002	4	64	0.0008	**0.0051**	-	0.0214	0.0306	0.0381	0.0156	+67.27%
scale_64_16	10001	16	64	0.0300	**0.1607**	-	0.1902	0.2393	0.2771	0.1981	+15.54%
scale_64_64	10001	64	64	0.1197	0.5518	0.5567	0.5453	0.5827	0.6913	**0.5452**	-1.22%
scale_64_128	10016	128	64	0.1388	0.6874	0.6232	0.6299	0.6421	1.0248	**0.6120**	-1.82%
sklearn_4_4	20000	4	4	0.0207	**0.0778**	-	0.1305	0.1777	0.2355	0.0875	+11.00%
sklearn_16_4	20000	4	16	0.0122	**0.0445**	-	0.0979	0.0990	0.1409	0.1205	+54.59%
sklearn_64_4	20000	4	64	0.0091	**0.0379**	-	0.1289	0.0818	0.1191	0.1051	+53.68%
sklearn_4_16	20000	16	4	0.0253	**0.0955**	-	0.1395	0.7110	0.5049	0.1833	+31.50%
sklearn_16_16	20000	16	16	0.0162	**0.0668**	-	0.1596	0.7160	0.5809	0.6901	+58.16%
sklearn_64_16	20000	16	64	0.0171	**0.1183**	-	0.3872	0.7631	0.8720	1.0101	+69.43%
sklearn_4_64	20000	64	4	0.0254	0.0993	**0.0974**	0.1090	1.4623	0.8614	0.3276	+1.96%
sklearn_16_64	20000	64	16	0.0226	0.1192	**0.1189**	0.2258	1.7475	1.3716	1.3519	+0.26%
sklearn_64_64	20000	64	64	0.0220	**0.3070**	0.4678	0.7878	2.2002	2.2456	2.0639	+34.37%
sklearn_4_128	20000	128	4	0.0281	0.1333	**0.1097**	0.1186	1.7604	1.1632	0.3499	+7.49%
sklearn_16_128	20000	128	16	0.0245	0.1942	**0.1562**	0.2838	2.3038	1.9742	1.7085	+19.55%
sklearn_64_128	20000	128	64	0.0249	**0.5740**	0.7190	0.9675	2.7204	2.9608	2.4354	+20.16%

(or decline) of the best PE-based approach compared to the best non-PE private approach in the last column. We see that when PE-based approaches are outperformed, it is by at most 7%, but on average, they improve by 20%, with improvements of as much as 91%. Finally, we note that HDPE-means typically outperforms PE-means on datasets with dimension larger than 16. In high dimensional cases where PE-means outperforms HDPE-means, it is not by a significant amount compared to the performance of related work, and thus we recommend always using HDPE-means in $d > 16$ datasets.

We give the full privacy vs. utility trade-off for the various real datasets (including the three evaluated by Chang and Kamath [4]) in Figure 3. We observe that a PE-based approach is always the best approach, being the closest to the non-private baseline in all plots. Among the baselines, we observe Icml17 performs well in all sets, whereas FastLloyd performs well in the lower dimensions (birch2 and iris) and Google does well in higher dimensions. Finally, in Figure 4, we show how the algorithms scale with the number of clusters and a fixed privacy budget of $\epsilon = 1$. The scale datasets follow the trend of the non-private version until we increase dimensions, at which point the noise has more of an effect at higher numbers of clusters. This is likely because the data size remains fixed, causing the number of samples voting (or being aggregated) per cluster to decline. The scale datasets also highlight the room for improvement in PE-based approaches in higher dimensions. For the Sklearn sets, which tend to be more well-separated Gaussian blobs, we observe PE-based approaches outperforming all other approaches and also scaling better with k.

V. Related Work

DP k-means: In addition to the baselines we compare against in Section IV, there are several previous works that also solve the DP k-means clustering problem, which are either improved upon by our baselines or are more theoretical in nature. One line of work focused on developing a private version of Lloyd's algorithm [3], [25], [9], [31], [5] starting with Blum et al. [3] and concluding with our baselines of Su et al. [31] and FastLloyd [5]. Another

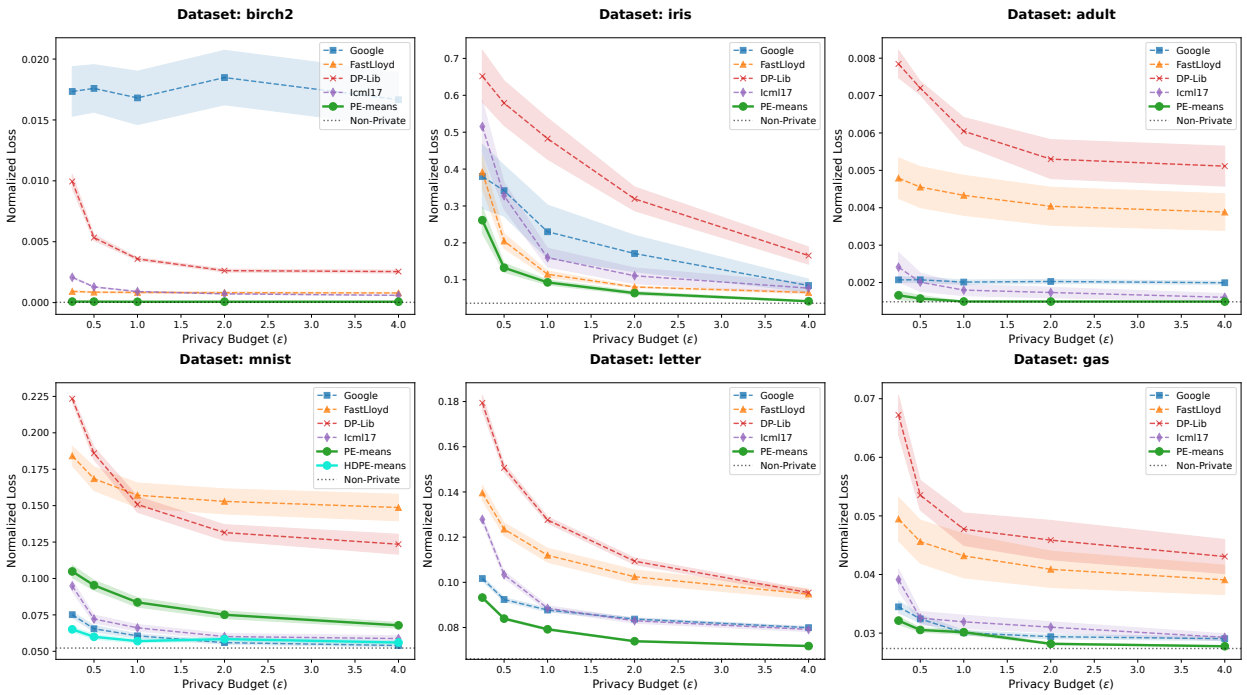

Fig. 3: A plot of the privacy-utility trade-off of all approaches on real datasets, showing a PE-based approach always performs best.

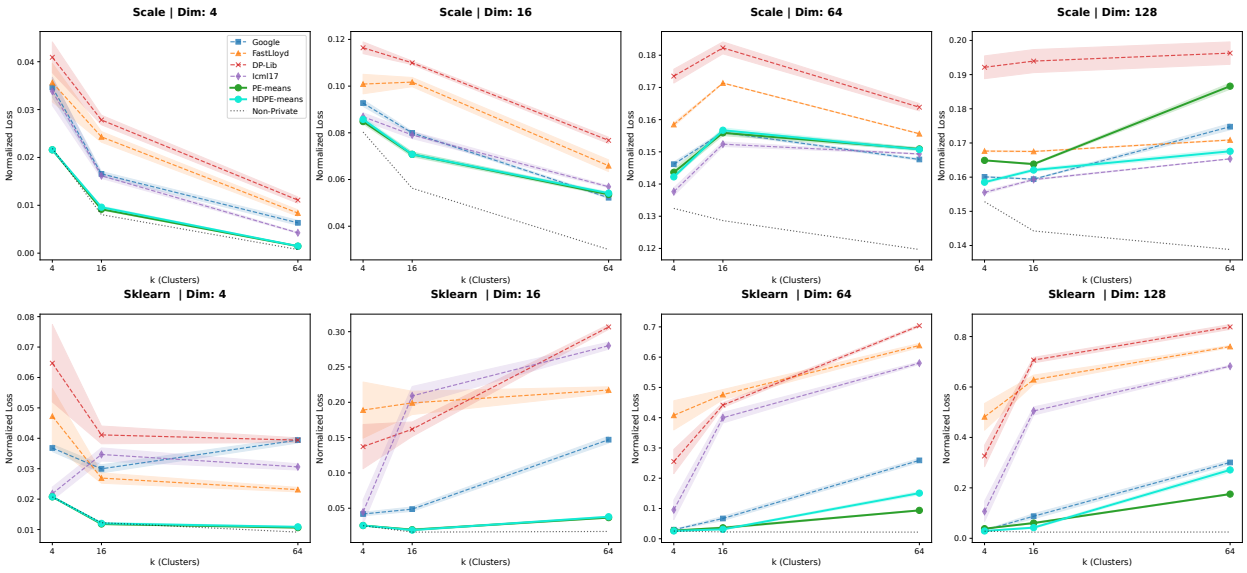

Fig. 4: A comparison illustrating how approaches scale with the number of clusters k over different synthetic datasets at ϵ=1.0.

line of work used the sample and aggregate framework [27], [26] to solve k-means, but was outperformed by Su et al. [31]. A theoretical line of work focused on minimizing the bounds on approximation error, but did not provide experimental evaluation [12], [13], [28], [30], [15], [19].

DP Evolutionary Algorithms: There has been previous work that have applied evolutionary techniques to the problem of clustering. The first differentially private evolutionary algorithm was developed by Zhang et al. [33] and evaluated the problem of k-means clustering. Follow-up work by Humphries and Kerschbaum [18] showed

that Zhang et al.'s solution suffered from prohibitively poor utility and provided an improved algorithm which was evaluated on the k-medians clustering problem. While our work also applies an evolutionary-based approach to solve clustering, our algorithm uses significantly less privacy budget. The population of PE-means is a set of centroids from which a single solution of k points is chosen in each iteration using a DP histogram with sensitivity 1, following the previous work in PE [22], [32], [21]. Humphries and Kerschbaum instead evolve a population where each candidate is a solution of k points and applies the exponential mechanism on the clustering loss (with sensitivity proportional to the data domain) to select multiple candidates in each iteration, using significantly more privacy budget.

VI. Conclusion

We have shown that the PE algorithm is well-suited to the problem of k-means clustering and have made various improvements to the PE algorithm of independent interest. Our benchmark shows that PE-means and HDPE-means offer state-of-the-art performance on many synthetic and real clustering datasets. In future work, we will study mutation operators that more efficiently search high-dimensional spaces.

Acknowledgements

We would like to thank Sivakanth Gopi for helpful discussions on an earlier version of this work, including the suggestion to use MLE post-processing on the vote histogram.

References

[1] Maria-Florina Balcan, Travis Dick, Yingyu Liang, Wenlong Mou, and Hongyang Zhang. Differentially private clustering in high-dimensional Euclidean spaces. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 322–331. PMLR, 06–11 Aug 2017.

[2] Borja Balle and Yu-Xiang Wang. Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 394–403. PMLR, 10–15 Jul 2018.

[3] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the SuLQ framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '05, pages 128–138, 2005.

[4] Alisa Chang and Pritish Kamath. Practical differentially private clustering, 2024.

[5] Abdulrahman Diaa, Thomas Humphries, and Florian Kerschbaum. {FastLloyd}: Federated, accurate, secure, and tunable {k-Means} clustering with differential privacy. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 2733–2752, 2025.

[6] Jinshuo Dong, Aaron Roth, and Weijie J. Su. Gaussian differential privacy. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 84(1):3–37, 02 2022.

[7] Dheeru Dua and Casey Graff. Uci machine learning repository. http://archive.ics.uci.edu/ml, 2021.

[8] John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 272–279, New York, NY, USA, 2008. Association for Computing Machinery.

[9] Cynthia Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 54(1):86–95, 2011.

[10] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography*, pages 265–284, 2006.

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

[12] Dan Feldman, Amos Fiat, Haim Kaplan, and Kobbi Nissim. Private coresets. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, STOC '09, pages 361–370. Association for Computing Machinery, 2009.

[13] Dan Feldman, Chongyuan Xiang, Ruihao Zhu, and Daniela Rus. Coresets for differentially private k-means clustering and applications to privacy in mobile sensor networks. In *Proceedings of the 16th ACM/IEEE International Conference on Information Processing in Sensor Networks*, pages 3–15. ACM, 2017.

[14] Pasi Fränti and Sami Sieranoja. K-means properties on six clustering benchmark datasets, 2018.

[15] Badih Ghazi, Ravi Kumar, and Pasin Manurangsi. Differentially private clustering: Tight approximation ratios. In *Advances in Neural Information Processing Systems*, volume 33, pages 4040–4054. Curran Associates, Inc., 2020.

[16] Attri Ghosal, Arunima Nandy, Amit Kumar Das, Saptarsi Goswami, and Mrityunjoy Panday. A short review on different clustering techniques and their applications. *Emerging Technology in Modelling and Graphics: Proceedings of IEM Graph 2018*, pages 69–83, 2020.

[17] Naoise Holohan, Stefano Braghin, Pól Mac Aonghusa, and Killian Levacher. Diffprivlib: the IBM differential privacy library. *ArXiv e-prints*, 1907.02444 [cs.CR], July 2019.

[18] Thomas Humphries and Florian Kerschbaum. Differentially private simple genetic algorithms. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*, pages 540–558, 2023.

[19] Matthew Jones, Huy L. Nguyen, and Thy D Nguyen. Differentially private clustering via maximum coverage. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11555–11563, May 2021.

[20] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[21] Zinan Lin, Tadas Baltrusaitis, and Sergey Yekhanin. Differentially private synthetic data via APIs 3: Using simulators instead of foundation model. In *ICLR 2025 Workshop on Navigating and Addressing Data Problems for Foundation Models*, 2025.

[22] Zinan Lin, Sivakanth Gopi, Janardhan Kulkarni, Harsha Nori, and Sergey Yekhanin. Differentially private synthetic data via foundation model APIs 1: Images. In *The Twelfth International Conference on Learning Representations*, 2024.

[23] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, March 1982.

[24] Rosario Nunzio Mantegna. Fast, accurate algorithm for numerical simulation of lévy stable stochastic processes. *Phys. Rev. E*, 49:4677–4683, May 1994.

[25] Frank D. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, page 19–30, New York, NY, USA, 2009. Association for Computing Machinery.

[26] Prashanth Mohan, Abhradeep Thakurta, Elaine Shi, Dawn Song, and David Culler. Gupt: privacy preserving data analysis made easy. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pages 349–360. ACM, 2012.

[27] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*, STOC '07, page 75–84, New York, NY, USA, 2007. Association for Computing Machinery.

[28] Kobbi Nissim and Uri Stemmer. Clustering algorithms for the centralized and local models. In *Proceedings of Algorithmic Learning Theory*, pages 619–653. PMLR, 2018.

[29] Weiliang Qiu and Harry Joe. *Random Cluster Generation (with Specified Degree of Separation)*, 2023. R package version 1.3.8.

[30] Uri Stemmer and Haim Kaplan. Differentially private k-means with constant multiplicative error. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[31] Dong Su, Jianneng Cao, Ninghui Li, Elisa Bertino, and Hongxia Jin. Differentially private k-means clustering. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, CODASPY '16, page 26–37, 2016.

[32] Chulin Xie, Zinan Lin, Arturs Backurs, Sivakanth Gopi, Da Yu, Huseyin A Inan, Harsha Nori, Haotian Jiang, Huishuai Zhang, Yin Tat Lee, Bo Li, and Sergey Yekhanin. Differentially private synthetic data via foundation model APIs 2: Text. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 54531–54560. PMLR, 21–27 Jul 2024.

[33] Jun Zhang, Xiaokui Xiao, Yin Yang, Zhenjie Zhang, and Marianne Winslett. Privgene: Differentially private model fitting using genetic algorithms. New York, NY, USA, 2013. Association for Computing Machinery.

[34] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, 1(2):141–182, 1997.