Generating Private Synthetic Data with Genetic Algorithms

Terrance Liu [*1] **Jingwu Tang** [*2] **Giuseppe Vietri** [*3] **Zhiwei Steven Wu** [1]

Abstract

We study the problem of efficiently generating differentially private synthetic data that approximate the statistical properties of an underlying sensitive dataset. In recent years, there has been a growing line of work that approaches this problem using first-order optimization techniques. However, such techniques are restricted to optimizing differentiable objectives only, severely limiting the types of analyses that can be conducted. For example, first-order mechanisms have been primarily successful in approximating statistical queries only in the form of marginals for discrete data domains. In some cases, one can circumvent such issues by relaxing the task's objective to maintain differentiability. However, even when possible, these approaches impose a fundamental limitation in which modifications to the minimization problem become additional sources of error. Therefore, we propose PRIVATE-GSD, a private genetic algorithm based on *zeroth*-order optimization heuristics that do not require modifying the original objective; thus, it avoids the aforementioned limitations of first-order optimization. We demonstrate empirically that on data with both discrete and real-valued attributes, PRIVATE-GSD outperforms the state-of-the-art methods on non-differential queries while matching accuracy in approximating differentiable ones.

1. Introduction

Access to high-quality data is critical for decision-making and data analysis. However, the reliance on sensitive data can reveal private information about the individuals in the data, such as medical conditions or financial status. Therefore, privacy concerns impose legal and ethical constraints on what one can access for data analysis. *Differential privacy* (Dwork et al., 2006) has become an increasingly popular framework for protecting sensitive information by providing a formal privacy guarantee that allows one to calibrate the trade-off between privacy and accuracy. Moreover, differentially private synthetic data has become especially attractive, given that such data can be accessed repeatedly without added privacy costs. Consequently, we focus on the problem of generating synthetic data that approximates various properties of the underlying sensitive dataset while providing privacy guarantees.

A standard approach to this problem is to find a synthetic dataset that matches a large family of statistics derived from the underlying dataset. In this work, we focus on statistics in the form of *statistical queries*, which count the fraction of examples that satisfy some specific properties. This approach is often framed as synthetic data for *private query release*, where the goal is to release answers to an extensive collection of statistical queries by outputting a synthetic dataset from which answers are derived from. There exists a long line of research studying synthetic data for private query release (Blum et al., 2008; Hardt et al., 2012; Gaboardi et al., 2014b; McKenna et al., 2018; Vietri et al., 2020; McKenna et al., 2019; Aydöre et al., 2021; Liu et al., 2021a;b). Moreover, when the set of statistical properties is sufficiently rich, this approach has been shown to outperform differentially private deep generative models (e.g., GANs) in both capturing various statistical properties and enabling downstream machine learning tasks (Tao et al., 2021; Vietri et al., 2022).

Even though the problem of generating synthetic data for private query release is shown to be computationally intractable in the worst case (Ullman & Vadhan, 2011; Ullman, 2013), there has been a line of work on practical algorithms that can perform well on real datasets (Gaboardi et al., 2014b; Vietri et al., 2020; McKenna et al., 2019; Liu et al., 2021a). Before our work, the algorithms that achieve state-of-the-art performance all leverage gradient-based optimization (Liu et al., 2021b; Vietri et al., 2022) to minimize the error for certain classes of statistical queries. However, these methods require differentiability, severely limiting the types of statistical properties one can optimize. For example, existing algorithms such as RAP (Aydöre et al., 2021) and GEM (Liu et al., 2021b) focus specifically on using first-

[*]Equal contribution [1]Carnegie Mellon University [2]Peking University [3]University of Minnesota. Correspondence to: Terrance Liu <terrancl@cs.cmu.edu>, Jingwu Tang <tangjingwu@pku.edu.cn>, Giuseppe Vietri <vietr002@umn.edu>, Zhiwei Steven Wu <zstevenwu@cmu.edu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

order mechanisms for approximating marginal queries that apply to discrete data only, requiring real-valued attributes be discretized first.

Many natural classes of queries for data containing real-valued attributes are non-differentiable, such as prefixes (equivalently the CDF of an attribute), and their higher-dimensional extension, halfspaces. Moreover, these queries cannot be optimized directly by methods that operate over discrete data, even with the discretization of real-valued features. Vietri et al. (2022) circumvent discretization by approximating non-differentiable queries with differentiable surrogate functions. Thus, their approach, RAP++, can directly optimize over a large class of statistical queries. However, their method induces additional error due to the relaxation of the original minimization problem, which may not produce an optimal solution for the original one. [1] [2]

In light of these challenges, we propose a new synthetic data generation algorithm, PRIVATE-GSD, that is capable of outputting *mixed*-type data (i.e., containing both discrete and real-valued attributes) without requiring differentiability in its optimization objective or discretization of real-valued attributes. PRIVATE-GSD is a zeroth-order mechanism based on a genetic algorithm that avoids the limitations of first-order methods such as RAP++, which uses surrogate loss functions to ensure differentiability. Instead, PRIVATE-GSD can optimize the error objective directly. Inspired by the genetic algorithm SIMPLEGA (Such et al., 2017), our algorithm PRIVATE-GSD evolves a population of datasets over G generations—starting from a population of E randomly chosen datasets, where on each generation, the algorithm maintains a set of elite synthetic datasets. Then, the algorithm uses this elite set to produce the population for the next round by crossing samples (i.e., combining parameters) and applying specific mutations (i.e., perturbing parameters). For the purpose of private synthetic data, we devise new strategies for crossing and mutating samples. To illustrate the limitations of the first-order approach RAP++ and how PRIVATE-GSD overcomes them, we provide a simple example in Figure 1, where both algorithms try to approximate a target distribution over real values.

Moreover, given that our method only requires zeroth-order access to any objective function, we demonstrate that PRIVATE-GSD can directly optimize over a wide range of statistical queries that capture various statistical properties summarizing both discrete and real-valued data attributes. Specifically, we first conduct experiments in mixed-type data domains, evaluating on random (1) halfspace and (2) prefix queries. We then explore categorical-only data, in which we use (3) k-way categorical marginal queries. Lastly, we again evaluate PRIVATE-GSD on mixed-type data using (4) k-way *binary-tree marginal* queries (defined in Section 5). Note that (3) and (4) can be represented as differentiable functions, while (1) and (2) cannot.

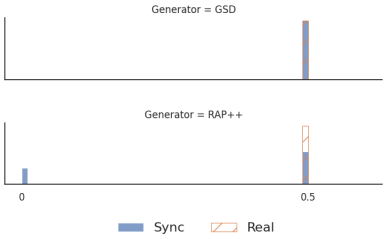

Figure 1. The histogram visually demonstrates a specific scenario where the RAP++ mechanism fails to generate synthetic data that approximates a collection of statistical prefix queries. In this particular scenario, both PRIVATE-GSD and RAP++ take as input three prefix queries with thresholds set at $0.49, 0.50$, and 0.51 and the input dataset REAL, represented by the clear bars, that has all values equal to 0.5. The output of both PRIVATE-GSD or RAP++ is represented by the blue histograms. The graph on the top shows that PRIVATE-GSD successfully produces an optimal synthetic data whose histogram exactly overlaps with the histogram of the REAL data. On the other hand, the bottom graph shows how RAP++ fails to optimize this case by generating a significant portion of its synthetic points near 0.

We summarize our contributions[3] as the following:

- We present PRIVATE-GSD, a versatile genetic algorithm that can generate synthetic data to approximate a wide range of statistical queries, regardless of their differentiability. To achieve this, we develop novel crossover and mutation functions that are crucial to the efficacy of PRIVATE-GSD.
- We conduct an empirical evaluation on data derived from the American Community Survey. We note that our method is highly parallelizable, scaling to the high-dimensional data settings explored in our experiments.
- In the mixed-type data domain, we show that PRIVATE-GSD outperforms the state-of-the-art method, RAP++, on non-differentiable queries (i.e., halfspace and prefix queries). Furthermore, we show that *even* for differentiable query classes (i.e., k-way categorical and binary-tree marginals), PRIVATE-GSD matches first-order optimization methods.[4]

[1] In Appendix B.3, we show a simple example in which the continuous relaxation leads RAP++ to a bad local minimum very far from the optimal solution

[2] Figure 1 gives a simple example where RAP++ fails to optimize a set of 3 prefix queries on 1-dimensional real-valued data.

[3] While not the primary focus of our work, we provide ML evaluation experiments in Appendix C

[4] Our work is also the first to modify our baseline methods, RAP and GEM, to optimize over k-way binary-tree marginals for mixed type data, with the limitation being that real-valued attributes are discretized first.

Additional related work. In addition to algorithms RAP, GEM, and RAP++ (Aydöre et al., 2021; Liu et al., 2021b; Vietri et al., 2022), another benchmark algorithm that uses first-order optimization is the PrivatePGM (PGM) algorithm (McKenna et al., 2019) and its variations (McKenna et al., 2022). Similar to RAP and GEM, PGM does not directly does not support non-differentiable queries such as thresholds and half-spaces on real-valued data. While one can first discretize numerical features, Vietri et al. (2022) show that RAP++ still performs better than PGM on real-valued data for approximating prefix queries and downstream machine learning. Finally, there is a line of work (Xie et al., 2018; Jordon et al., 2018a;b; Torkzadehmahani et al., 2019; Takagi et al., 2020; Harder et al., 2021) that develops differentially private algorithms for training deep generative models (e.g., GANs, VAEs, etc.), but they generally have not shown promising results for enforcing consistency with simple classes of statistics when applied to tabular data (Tao et al., 2021; Vietri et al., 2022).

2. Preliminaries

2.1. Datasets and Queries

Let's consider $\mathcal{X}$ as a data domain of dimensionality d. Here, each element $x \in \mathcal{X}$ is an observation within this data domain. For any given observation $x \in \mathcal{X}$, we refer to its i-th feature as $x_i \in \mathcal{X}_i$. The feature domain associated with this i-th feature is denoted by $\mathcal{X}_i$. We define a dataset D as a multiset of observations derived from the data domain $\mathcal{X}$, where $D \subseteq \mathcal{X}^*$. Often, to ensure a convenient representation of categorical features, we employ the one-hot encoding approach. In this context, one-hot encoding portrays a categorical value as a binary vector where all elements, barring the one corresponding to the category, are set to zero. We introduce $h_{oh} : \mathcal{X} \to [0,1]^{d'}$ to denote the function that maps elements of $\mathcal{X}$ to their one-hot encoding, with d' representing the dimensionality of the one-hot encoded space. Finally, for any discrete set S, we define $U(S)$ as the uniform distribution over this set S. A random sample from $U(S)$ can be represented as $s \sim U(S)$. For any integer i, we use $[i] = 1, \ldots, i$ to represent the set of all integers from 1 to i.

In this research, our focus lies on mixed-valued datasets within a bounded domain, with a particular emphasis on answering statistical queries. A statistical query is defined as a function $q : \mathcal{X}^* \to [0,1]$ that maps a dataset to a value within the bounded interval [0,1]. The formal definition of a statistical query in this context is

Definition 2.1 (Statistical Query (Kearns, 1998))**.** A *statistical query* is defined by a predicate function $\phi : \mathcal{X} \to \{0,1\}$. Given a dataset $D \in \mathcal{X}^N$, the corresponding statistical query q_ϕ is defined as : $q_\phi(D) = \frac{1}{N}\sum_{x \in D} \phi(x)$.

We use $Q = \{q_i, \ldots, q_m\}$ to represent a set of m statistical queries and for a given dataset D. Furthermore, for convenience, we use $Q(D) = [q_1(D), \ldots, q_m(D)] \in [0,1]^m$ to denote the answers of all m statistical queries on D.

In this work, we are interested in answering different types of statistical queries that capture different statistical properties. The first type is the class of categorical marginals, which has been the primary focus of most prior works. The *k-way categorical marginal* queries count the fraction of rows in a dataset that matches a combination of values.

Definition 2.2 (Categorical Marginal Queries)**.** A k-way *categorical marginal* query is defined by a set of categorical features S of cardinality $|S| = k$, together with a particular element $c \in \prod_{i \in S} \mathcal{X}_i$ in the domain of S. Given such a pair (S, c), let $\mathcal{X}(S, c) = \{x \in \mathcal{X} : x_S = c\}$ denote the set of points that match c. The corresponding statistical query $q_{S,c}$ is defined as $q_{S,c}(D) = \sum_{x \in D} \mathbb{1}\{x \in \mathcal{X}(S, c)\}$, where $\mathbb{1}$ is the indicator function.

For real-values features, we define the class of *range* queries or *k-way range-marginals* that capture the k. A 1-way range query is defined by an attribute A and two real-values $a, b \in [0,1]$ and counts the number of rows where real-valued attribute A lies in $[a, b]$.

Definition 2.3 (Range Marginal Queries)**.** A k-way *range marginal* query is defined by a set of categorical features C, an element $y \in \mathcal{X}_C$, a set of numerical features R and a set of intervals $\tau = \{[\tau_{j,0}, \tau_{j,1}]\}_{j \in R}$, with $|C| + |R| = k$ and $|R| = |\tau|$. Let $\mathcal{X}(C, y)$ be as in Definition 2.2 and let $\mathcal{R}(R, \tau) = \{x \in \mathcal{X} : \tau_{j,0} \leq x_j \leq \tau_{j,1} \quad \forall_{j \in R}\}$ denote the set of points where each feature $j \in R$ fall below its corresponding threshold value τ_j. Then the statistical query $q_{C,y,R,\tau}$ is defined as

$$q_{C,y,R,\tau}(D) = \sum_{x \in D} \mathbb{1}\{x \in \mathcal{X}(C, y)\} \cdot \mathbb{1}\{x \in \mathcal{R}(R, \tau)\}.$$

We note that while range marginal queries are useful for capturing statistical properties of real-valued data, they can also be applicable to discretized data. For example, one can discretize a real-valued column into the bins that are a superset of the intervals τ defined in Definition 2.3.[5]

Next, we present two additional queries—halfspace and prefix queries—that are also useful in summarizing real-valued features. We emphasize that such queries are the focus of our work, since they cannot be directly optimized by existing methods using first-order optimization. These queries can in some ways be thought of as higher-dimensional generalizations of CDF functions. Moreover, synthetic data matching statistical properties defined by these queries have

[5] In Section 5 and the appendix, we will discuss how range marginal queries are still applicable to GEM and RAP, which handle real-valued attributes by discretizing them first.

been shown to do well on downstream tasks like machine learning classification (Vietri et al., 2022).

Definition 2.4 (Halfspace Query)**.** A *halfspace* query is defined by a vector $\theta \in \mathbb{R}^{d'}$ and a threshold $\tau \in \mathbb{R}$. The *halfspace* query is given by $q_{\theta,\tau}(D) = \sum_{x \in D} \mathbb{1}\{\langle \theta, h_{oh}(x) \rangle \leq \tau\}$, where $h_{oh} : \mathcal{X} \to [0,1]^{d'}$ is the function that maps elements of $\mathcal{X}$ to their one-hot encoding.

Definition 2.5 (k-way prefix queries)**.** A k-way prefix query is defined by a set of real-valued features $C \subseteq [d]$ ($|C| = k$) and a set of threshold $\tau = \{\tau_i \in [0,1] : i \in C\}$. Let $\mathcal{X}(C,\tau) = \{x \in \mathcal{X} : x_i \leq \tau_i \quad \forall i \in C\}$ be the set of points, that has each feature $i \in C$ fall below the corresponding threshold τ_i. Then the prefix query $q_{C,\tau}$ is given by $q_{C,\tau}(D) = \sum_{x \in D} \mathbb{I}\{x \in \mathcal{X}(C,\tau)\}$.

2.2. Differential Privacy

To protect user privacy, we estimate $Q(D)$ under the constraint of differential privacy (Dwork et al., 2006), which is a widely accepted notion of privacy protection at the user level. For example, the US Census Bureau has begun using differential privacy to protect the privacy of individuals when releasing census data. Differential privacy is used to add noise to the census data in order to obscure the data of any one individual, while still providing accurate statistical information about the population as a whole.

To define differential privacy, we say that two datasets are neighboring if they are different in at most one data point. We give the formal definition here:

Definition 2.6 (Differential Privacy (DP) (Dwork et al., 2006))**.** A randomized algorithm $\mathcal{M} : \mathcal{X}^N \to \mathcal{R}$ satisfies (ε, δ)-differential privacy if for all neighboring datasets D, D' and for all outcomes $S \subseteq \mathcal{R}$ we have

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\varepsilon} \Pr[\mathcal{M}(D') \in S] + \delta$$

A related notion we use is *zero-Concentrated Differential Privacy* (zCDP) (Dwork & Rothblum, 2016; Bun & Steinke, 2016a), which has a simpler privacy composition.

Definition 2.7 (zCDP (Bun & Steinke, 2016b))**.** A randomized algorithm $\mathcal{M} : \mathcal{X}^N \to \mathcal{R}$ satisfies ρ-zero-Concentrated differential privacy (ρ-zCDP) if for all neighboring datasets D, D' and for all $\alpha \in (1, \infty)$ we have: $\mathbb{D}_\alpha(\mathcal{M}(D), \mathcal{M}(D')) \leq \rho\alpha$, where $\mathbb{D}_\alpha(\mathcal{M}(D), \mathcal{M}(D'))$ is α-Renyi divergence between the distributions $\mathcal{M}(D)$ and $\mathcal{M}(D')$.

3. Private Query Release via Synthetic Data

For the problem of estimating statistics $Q(D)$ subject to differential privacy, we follow the framework of the projection mechanism (Nikolov et al., 2013; Dwork et al., 2015), which consists of perturbing the statistics with a differentially private mechanism followed by a *projection step* that consists of finding a synthetic dataset that matches these perturbed statistics.

In this study, we consider two versions of the private projection mechanism. The first is the standard projection mechanism, referred to as the *one-shot* version. The second is an *adaptive* version of the projection mechanism, which proves useful in scenarios where a large number of queries are required. A comprehensive description of both frameworks is presented below.

One-shot projection mechanism: Suppose we are given m statistical queries, represented by Q, and a dataset D. Then the first step consists of independently perturbing every coordinate of $Q(D)$ to obtain a private estimate, denoted by $\hat{a}$. Given a privacy parameter ρ, the noisy answer vector is computed as follows:

$$\hat{a} \leftarrow Q(D) + \mathcal{N}\left(0, \tfrac{\Delta(Q)}{2\rho} \cdot \mathbb{I}\right) \tag{1}$$

Then, the *project* step involves finding a synthetic dataset $\widehat{D}$ that best explains the estimate $\hat{a}$. To represent the projection step, we construct a loss minimization problem, where the loss function is defined by $\hat{a}$ and Q as follows:

$$L_{\hat{a},Q}(\widehat{D}) = \|\hat{a} - Q(\widehat{D})\|_2^2 \tag{2}$$

By the properties of the Gaussian mechanism and the post-processing property (Theorem B.3), the projection mechanism framework satisfies ρ-zCDP regardless of the optimization procedure used to solve (2).

Adaptive selection framework: In practice, when the number of queries m is very large, the objective (2) becomes extremely noisy for any practical optimization routine. Therefore, we follow the Adaptive Selection Framework of Liu et al. (2021b) that, over K rounds, chooses high error queries adaptively using the report noisy max mechanism (alternatively the exponential mechanism, as defined in Definition B.5). Algorithm 2 of Appendix B.2 shows the details of this framework, which we note satisfies the following theorem:

Theorem 3.1. *For any K, $\rho > 0$, dataset D, query set Q and any procedure used to solve objective* (2)*, the adaptive framework strategy satisfies ρ-zCDP. By extension, it also satisfies (ε, δ)-differential privacy for: $\varepsilon = \rho + 2\sqrt{\rho \cdot \log(1/\delta)}$.*

We provide a proof sketch to Theorem 3.1 below in Appendix B.1.

Algorithm 1 Private Genetic Algorithm for Synthetic Data (PRIVATE-GSD)

Require: The search space $\mathcal{X}^{N'}$ of mixed-type synthetic datasets with N' rows and d features, number of generations G, mutations population size P_{mut}, crossover population size P_{cross} and elite set size E.
Input: A set of m queries $Q : \mathcal{X}^* \to [0,1]^m$, (noisy) query answers $\hat{a} \in [0,1]^m$.
Set the objective function for the synthetic datasets:

$$L_{Q,\hat{a}}(\widehat{D}) = \|\hat{a} - Q(\widehat{D})\|_2$$

Initialize uniformly at random the elite set $\mathcal{E}_1 \leftarrow \{\bar{D}_{1,i} \sim U(\mathcal{X}^{N'}) : i \in [E]\}$ of synthetic datasets with N' rows.
for $g = 1, \ldots, G$ **do**
  Set $\widehat{D}_g^\star \leftarrow \arg\min_{\widehat{D} \in \mathcal{E}_g} L_{Q,\hat{a}}(\widehat{D})$.
  Initialize candidate population set $\mathcal{P}_g \leftarrow \mathcal{E}_g$.
  Mutations: Repeat P_{mut} times: Initialize a synthetic data $\widetilde{D}$ equal to $\widehat{D}_g^\star$. Then, sample a row $i \sim U([N'])$, feature $j \sim U([d])$ and value $v \sim U(\mathcal{X}_j)$. Update $\widetilde{D}$ as follows: $\widetilde{D}(i,j) \leftarrow v$ and add $\widetilde{D}$ to $\mathcal{P}_g$.
  Crossovers: Repeat P_{cross} times: Initialize $\widetilde{D}$ equal to $\widehat{D}_g^\star$. Then, sample two rows $i_1, i_2 \sim U([N'])$, a feature $j \sim U([d])$, and elite member $\bar{D} \sim U(\mathcal{E}_g)$ and update $\widetilde{D}$ as follows: $\widetilde{D}(i_1, j) \leftarrow \bar{D}(i_2, j)$. Then add $\widetilde{D}$ to $\mathcal{P}_g$.
  Update Elites: Evaluate each member of $\mathcal{P}_g$ on $L_{Q,\hat{a}}(\cdot)$. Then, set $\mathcal{E}_{g+1}$ as top E members from population $\mathcal{P}_g$ with respect to *minimizing* objective $L_{Q,\hat{a}}(\cdot)$.
end for
return $\widehat{D}_G^\star$

4. Private Genetic Algorithm (PRIVATE-GSD)

The generation of high-quality synthetic data through the projection mechanism framework, as described in Section 3, presents a challenging task that necessitates the solution of an NP-hard optimization problem (Hardt & Talwar, 2010). Therefore, we propose the PRIVATE-GSD algorithm, which solves the projection step in the projection mechanism using a genetic algorithm (GA).

4.1. Background on genetic algorithms

Genetic Algorithms (GA) are a class of optimization algorithms inspired by natural selection and genetic recombination. Introduced by Holland (1992), GAs provide a heuristic search technique to solve complex optimization problems by evolving a population of candidate solutions towards an optimal or near-optimal solution. Crucially, GAs do not require differentiability in the optimization objective.

4.2. Private-GSD

This section presents a genetic algorithm named Private Genetic Synthetic Data (PRIVATE-GSD), developed specifically for synthetic data generation. PRIVATE-GSD applies two genetic operators known as 'mutations' and 'crossover' in the GA literature ((Such et al., 2017)). In contrast to prior genetic algorithms, the mutation and crossover operator applied by PRIVATE-GSD are specifically designed for searching the space of synthetic datasets and optimizing the projection step described in Section 3. These operators, aimed at generating incremental improvements in synthetic datasets across multiple generations, have been designed for computational efficiency and quick convergence. We begin by unpacking the core elements and structure of PRIVATE-GSD and then dive into the specifics of these genetic operators.

The PRIVATE-GSD algorithm operates on a set of statistical queries, denoted as Q, and their respective target private responses, $\hat{a}$. It formulates an optimization problem via the objective function $L_{Q,\hat{a}} : \mathcal{X}^{N'} \to [0,1]$. A proposed solution to $L_{Q,\hat{a}}$ is a synthetic dataset comprising a user-specified number of rows, N'. Thus, PRIVATE-GSD's ultimate objective is to identify a synthetic dataset within $\mathcal{X}^{N'}$ that minimizes $L_{Q,\hat{a}}$. PRIVATE-GSD also incorporates other parameters such as the maximum number of generations, G, over which it will operate, as well as P_{mut} and P_{cross}, which respectively represent the quantity of proposed solutions generated using mutation and crossover strategies. Lastly, the E parameter determines the number of candidates that will be selected to parent the succeeding generation.

The genetic process unfolds across G generations. For each generation, denoted as $g \in [G]$, PRIVATE-GSD maintains a set of elite E candidate solutions, symbolized as $\mathcal{E}_g$. These represent the optimal E candidate solutions discovered thus far. The initial elite set, $\mathcal{E}_1$, is chosen at random from the set $\mathcal{X}^{N'}$. Within each generation, PRIVATE-GSD generates $(P_{\text{mut}} + P_{\text{cross}})$ new candidate solutions. These are obtained by introducing minor random variations to the best synthetic data identified up to generation g, denoted as $\widehat{D}_g^\star$. To achieve this, PRIVATE-GSD applies a random genetic modification—either mutation or crossover—to $\widehat{D}_g^\star$, yielding a new synthetic data candidate, $\widetilde{D}$. This process allows for inheriting traits from the elite synthetic dataset and adds diversity, paving the way for potentially superior solutions.

The group of candidate synthetic datasets in generation $g \in [G]$, denoted as $\mathcal{P}_g$, comprises the $P_{\text{mut}} + P_{\text{cross}}$ new candidates and the elite candidates $\mathcal{E}_g$. PRIVATE-GSD assigns a score to each candidate in $\mathcal{P}_g$ based on their alignment with the objective function $L_{Q,\hat{a}}$. It then ranks all $P_{\text{mut}} + P_{\text{cross}} + E$ individual candidates according to their scores, and the top E candidates with the highest fitness scores are chosen as the next generation of elite candidates.

This iterative procedure continues until a predetermined termination criterion—such as reaching the maximum number of generations, achieving a satisfactory fitness level, or meeting a predefined condition—is satisfied.

Now, we delve into the specifics of the 'mutation' and 'crossover' genetic operators, which play a significant role in PRIVATE-GSD's performance. These operators are only applied to the optimal dataset $\widehat{D}_g^\star$, with the goal of incremental improvements over many generations. Moreover, both operators only modify a single value of $\widehat{D}_g^\star$. Empirically, we find that having sparse genetic updates is crucial to the success of PRIVATE-GSD, as shown in Table 4 where altering numerous values simultaneously performs noticeably worse.

Mutation. The mutation process is a vital part of a genetic algorithm. It introduces minor random perturbations to the most optimal solution to create novel candidate solutions. Controlled by parameter P_{mut}, the mutation strategy of the PRIVATE-GSD algorithm determines the number of candidate synthetic datasets produced in each generation. During the mutation phase, a row $i \in [N']$ and column $j \in [d]$ are randomly selected. The entry in $\widehat{D}_g^\star$ corresponding to the selected row i and column j is then replaced by a value sampled uniformly from the range of acceptable values for column j. This modified dataset, which differs from $\widehat{D}_g^\star$ by a single entry,[6] is subsequently added to the pool of candidate solutions.

Crossover. Taking inspiration from genetic recombination, the crossover operator merges the parameters of two parent candidates to generate a new offspring. Parameter P_{cross} determines the number of candidate synthetic datasets produced in each PRIVATE-GSD generation via the crossover strategy. To sample a new candidate synthetic dataset, the PRIVATE-GSD's crossover operation selects an elite dataset $\bar{D}$ from the elite set $\mathcal{E}_g$. It then combines the parameters of $\bar{D}$ with those of the optimal dataset $\widehat{D}_g^\star$ to produce a new dataset $\widetilde{D}$. This synthetic dataset, differing from $\widehat{D}_g^\star$ by a single entry, is subsequently added to the candidate solutions population, $\mathcal{P}_g$. The crossover operation combines the parameters of $\widehat{D}_g^\star$ and $\bar{D}$ by selecting two rows $i_1, i_2 \in [N']$, and a feature $j \in [d]$. The new dataset $\widetilde{D}$ then has its (i_1, j) entry set to match the value of $\bar{D}$ at row i_2 and feature j (i.e., $\widetilde{D}(i_1, j) = \bar{D}(i_2, j)$), while all other entries of $\widetilde{D}$ mirror the corresponding values at $\widehat{D}_g^\star$.

The crossover operator plays an integral role in improving the convergence rate of PRIVATE-GSD. Figure 2 shows the convergence of the PRIVATE-GSD algorithm with and without the crossover operator. By encouraging the replication of significant values (with respect to the fitness function) in subsequent generations, PRIVATE-GSD with the crossover step converges exponentially faster compared to an algorithm reliant solely on mutations.

[6] While we could adjust the number of mutated rows at each step, our empirical data shows that in our settings, mutating a single row often yields the best performance (see Table 1).

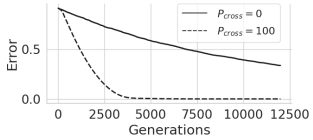

Figure 2. A comparison of the convergence rates of PRIVATE-GSD under two different parameterizations, aiming to understand the impact of the *crossover* genetic operator. The accompanying plot illustrates the error of PRIVATE-GSD after each generation, in relation to matching the histogram distribution (1-way marginals) of the input data. The the solid line represents parameters $P_{\text{mut}} = 200$ and $P_{\text{cross}} = 0$, while the dotted line corresponds to $P_{\text{mut}} = 100$ and $P_{\text{cross}} = 100$. The experiment employs an artificially created dataset, intentionally designed to be sparse and of high cardinality, thus challenging to approximate. The dataset comprises five categorical features, each with a cardinality of 1000, and contains 2000 rows.

4.3. Privacy And Accuracy

Below, we state the formal privacy guarantee of PRIVATE-GSD, which directly follows from Theorem 3.1.

Corollary 4.1. *For any $\rho > 0$, dataset D, and query set Q. Running* PRIVATE-GSD *(Algorithm 1) under the adaptive framework satisfies ρ-zCDP and (ε, δ)-differential privacy for: $\varepsilon = \rho + 2\sqrt{\rho \cdot \log(1/\delta)}$.*

Given that PRIVATE-GSD falls within the 'projection mechanism' framework proposed by Nikolov et al. (2013), an accuracy analysis can be performed under several assumptions: (1) the input data are discrete, (2) the number of queries is finite, and (3) the optimization step invariably finds the optimal solution of the objective. This kind of analysis assumes access to a perfect optimization oracle and is often referred to as 'oracle-efficient' accuracy analysis. It follows a similar approach to previous studies by Gaboardi et al. (2014a); Vietri et al. (2022); Aydöre et al. (2021).

The accuracy statement for PRIVATE-GSD is presented in Theorem A.1, with a detailed proof provided in the Appendix. Our analysis of PRIVATE-GSD's accuracy is similar to the accuracy analysis of the RAP++ mechanism (Aydöre et al., 2021), which also uses the projection mechanism framework and adopts the 'oracle-efficient' assumption. However, unlike RAP++, which implements a relaxation step in the data domain to maintain objective differentiability, PRIVATE-GSD does not perform such a step. Consequently, our accuracy theorem applies to a general class of queries, whereas the one for RAP++ is limited to

k-way marginal queries.

Remark 4.2. Theorem A.1 only applies if the underlying optimization algorithm in Algorithm 1 is successful in minimizing the objective in Equation (2). Therefore, we accompany the accuracy theorem with empirical evidence that validates the performance of PRIVATE-GSD across various datasets (Section 5; Figure 5).

5. Experiments

We conduct experiments to compare the performance of PRIVATE-GSD against baseline mechanisms in various settings. In particular, we evaluate the performance of various synthetic data mechanisms on mixed-typed data, which contain both categorical and numerical attributes. Our results demonstrate that in this setting, PRIVATE-GSD is strong mechanism in approximating various types of statistical queries, outperforming RAP++, the state-of-the-art method for generating synthetic data with real-valued columns. In addition, we demonstrate that PRIVATE-GSD achieves performance comparable to that of mechanisms specifically designed for—and restricted to—handling differentiable query functions for discrete data.

5.1. Data

For our empirical evaluation, we use datasets derived from the Folktables package (Ding et al., 2021), which defines datasets using samples from the American Community Survey (ACS). The ACS is collected annually by the U.S. Census Bureau to capture the demographic and economic characteristics of individuals and households in the country. In particular, Folktables provides an array of classification tasks, which are defined by a target column, a collection of mixed-type features, and some filter that specifies what samples to include.[7] The experiments in this section use samples from California and the five ACS tasks listed in Table 1. Finally, each dataset is normalized by linearly scaling real-valued columns to lie in $[0, 1]$.

Dataset	N	# Real	# Cat.
ACSMOB-CA	64263	4	17
ACSEMP-CA	303054	1	16
ACSINC-CA	156532	2	9
ACSCOV-CA	110843	2	17
ACSTRA-CA	138006	2	14

Table 1. For each dataset, we list the number of rows N, as well as the number of real and categorical features.

[7]For example, a task can be defined to include only rows with individuals over the age of 18.

5.2. Statistical queries

Statistical queries are essential for data analysis and decision-making, as they allow us to extract meaningful insights from large and complex datasets. In this section, we describe how we define the four sets of statistical queries that we use in our experiments. We summarize these queries in Table 2 and describe them in more detail below:

Random Prefixes: The first query class we consider is a random set of $50,000$ prefix queries, as in Definition 2.5. To generate a random prefix query, we randomly sample a categorical column c and two numeric columns a, b. Then, uniformly sample a target value v for column c and the two thresholds $\tau_a, \tau_b \in [0, 1]$ for the numeric columns a and b, respectively. The corresponding prefix query q_{prefix} is defined as follows:

$$q_{\text{prefix}}(x) = \mathbb{I}\{x_c = v \text{ and } x_a < \tau_a \text{ and } x_b < \tau_b\} \quad (3)$$

Random Halfspaces Next, we sample $m = 200,000$ random halfspace queries as in Definition 2.4. Each sample is generated as follows: Let d' be the dimension of the one-hot encoding of $\mathcal{X}$. First, sample a random vector $\theta \in \mathbb{R}^{d'}$, where the value of each coordinate $i \in [d']$ is an i.i.d sample drawn from the normal distribution with variance $\frac{1}{d}$ (i.e., $\theta_i \sim \mathcal{N}(0, \frac{1}{d})$). Then, sample a threshold value τ from the standard normal distribution, i.e., $\tau \sim \mathcal{N}(0, 1)$. Finally, add the query $q_{\theta,\tau}$ to the set Q_{HS}.

k-way Categorical Marginals: Using categorical columns only, we select all 2-way marginals in the one-shot setting and all 3-way marginal queries in the adaptive setting. Since the query answers in a workload sum up to 1, each workload has ℓ_2-sensitivity $\Delta_2 = \sqrt{2}$.

k-way Binary-Tree Marginals: In this work we introduce a class of queries, which we coined *Binary-Tree Marginals*. The Binary-Tree marginals consists of a collection of range queries, as in Definition 2.3) defined over various levels. We construct range queries using one categorical attribute and $k - 1$ real-valued attribute. Recall that real-valued features lie in the domain $[0, 1]$. Thus, we define a workload of range queries by the set of intervals of width $\frac{1}{2^j}$ for $j \in \{1, 2, \ldots, 5\}$. Now consider the set

$$I = \{\left(\tfrac{i}{2^j}, \tfrac{i+1}{2^j}\right) | 0 \leq i < 2^j, j \in \{1, 2, \ldots, 5\}\}$$

Using the same notation from definition 2.3, for real-valued feature c, the corresponding range workload $Q_{c,I} = \{q_{c,(a,b)}\}_{(a,b)\in I}$ is the set of range queries defined by intervals in I. This construction simulates the binary mechanism of Chan et al. (2011) for releasing sum prefixes privately.

Note that a single data point only appears in one interval of size $1/2^j$ for each $0 < j \leq 5$. Therefore, adding a new

data point only affects the counts of 5 range queries in the workload $Q_{c,I}$. By the same logic, replacing a data point affects 10 range queries in $Q_{c,I}$. Therefore, we say that the workload $Q_{c,I}$ has ℓ_2-sensitivity $\sqrt{10}$.

Query class	Query set size	Δ_2
2-way Categorical	$\binom{d}{2}$	$\sqrt{2}$
3-way Categorical	$\binom{d}{3}$	$\sqrt{2}$
2-way Binary-Tree	$d_{cat} \cdot d_{num}$	$\sqrt{10}$
Random Prefix	200000	1
Random Halfspaces	200000	1

Table 2. The total number of queries depends on the input data total number of features, which we denote by d and the number of categorical and numerical features, d_{cat} and d_{num} respectively. In addition, we list the ℓ_2-sensitivity Δ_2 of each query class (the sensitivity for categorical and binary-tree marginals is written with respect to an entire workload of queries). In our experiments, the number of categorical attributes binary-tree marginals is set to 1.

5.3. Baselines

RAP++. We compare our mechanism PRIVATE-GSD against the mechanism RAP++ by Vietri et al. (2022). The RAP++ algorithm uses a first-order optimization algorithm called *Sigmoid Temperature Annealing* (pseudocode can be found in Appendix B.3). Recall that RAP++ replaces the prefix queries with a differentiable surrogate query by using Sigmoid functions. To control the degree by which the Sigmoid functions approximate the prefix functions, it uses the *inverse Sigmoid temperature parameter* σ^{it}. The algorithm starts with a small inverse temperature σ^{it} and runs a gradient-based minimizer on the induced optimization objective with learning rate LR, and then repeats the process for a different choice of inverse temperature.

RP / GN. We also compare our method to RAP[8] (Liu et al., 2021b; Aydöre et al., 2021) and GEM (Liu et al., 2021b), two gradient-based optimization algorithms for generating private synthetic data that belong to discrete data domains. In both algorithms, datasets are modeled as mixtures of product distributions P over the attributes, with the main difference being the way in which such distributions are parameterized—in GEM, P is parameterized by a neural network that given Gaussian noise, outputs a set of product distributions, while in RAP, P is parameterized directly by the probabilities defining the distribution for each attribute. Because both methods are limited to discrete data, real-valued columns must be discretized first. Consequently, RAP and GEM can only optimize over k-way categorical and binary-tree marginals. In Appendix B.4, we show how we modify these methods to optimize over binary-tree queries and also provide additional details about their optimization procedures. Going forward, we denote the one-shot version of RAP and GEM by RP (RelaxedProjection) and GN (GenerativeNetwork) respectively.

[8]We use the softmax variant proposed by Liu et al. (2021b), which has been shown to perform much better that the versions originally proposed in Aydöre et al. (2021). We note in Vietri et al. (2022), references to RAP also refer to this softmax version.

PGM+EM. Lastly, we compare to a graphical model approach, PGM (McKenna et al., 2019), which like RAP and GEM, is also limited to discrete data domains. However, PGM instead models data distributions using probabilistic graphical models. We run an adaptive version of PGM that also uses the exponential mechanism. We refer to this variant as PGM+EM and run it on k-way categorical marginals.

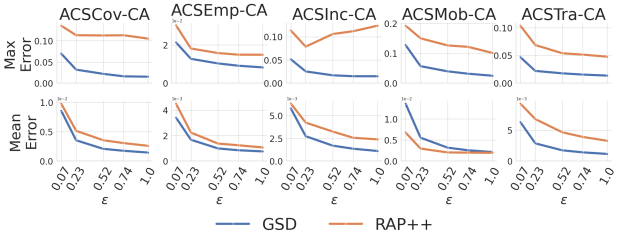

Figure 3. **Prefix Queries:** Max and mean error evaluated on all ACS tasks with non-differentiable queries— 200K random prefixes. Using adaptive version of PRIVATE-GSD and RAP++.

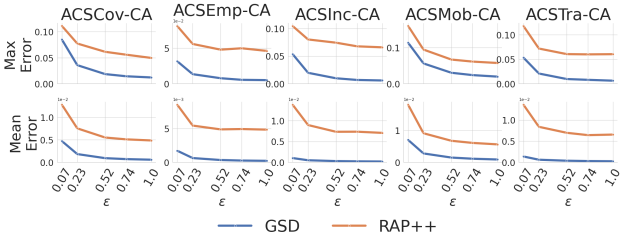

Figure 4. **Halfspace Queries:** Max and average error evaluated on all ACS tasks with non-differentiable queries— 200K random halfspaces. Using adaptive version of PRIVATE-GSD and RAP++.

5.4. Early Stop

In all experiments, PRIVATE-GSD is set to operate over a maximum of G generations but uses an early-stop rule if convergence in optimizing the objective is achieved. This procedure saves computational time and the details are found in Appendix D.1.

5.5. Results

We now present empirical results comparing PRIVATE-GSD to various baseline methods. For our experiments, we

fix $\delta = \frac{1}{N^2}$ and $\varepsilon \in \{0.07, 0.23, 0.52, 0.74, 1.0\}$ to examine the trade-off between privacy and accuracy. For adaptive algorithms, we run using adaptive epochs $T \in \{25, 50, 75, 100\}$ and report the best error (averaged over 3 runs) for each ε value.

First, we provide empirical evidence that PRIVATE-GSD is a better algorithm for generating real-valued synthetic data (i.e., when the statistical queries are non-differentiable). As originally shown in Vietri et al. (2022), Figures 3 and 4 demonstrate that RAP++, the previous state-of-the-art method, can optimize over prefix and halfspace queries using a surrogate differentiable approximation in combination with temperature annealing. However, by directly optimizing over such objectives using zeroth order optimization, PRIVATE-GSD significantly outperforms RAP++ with respect to max and average error.

Next, we show that on discretized data (both categorical and mixed-type), PRIVATE-GSD mechanism performs equally as well as prior work RP, GN and PGM+EM. In Figures 5 and 7, we evaluate on categorical marginals, which are one of the primary focuses of RP, GN and PGM.[9] In addition, we evaluate on binary-tree queries in Figure 6. We note that for RP and GN, we discretize the data in such a way that the resulting bins align with the intervals comprising our binary-tree marginal queries, allowing us to also write differentiable forms of such queries for RP and GN. Nevertheless, we observe that PRIVATE-GSD is able to synthesize both categorical-only (for categorical marginals) and mixed-type (for binary-tree marginals) data that matches the performance of the three baseline algorithms. Moreover, unlike the baseline methods, PRIVATE-GSD does not require that the numerical features be discretized first when optimizing over binary-tree marginal queries.

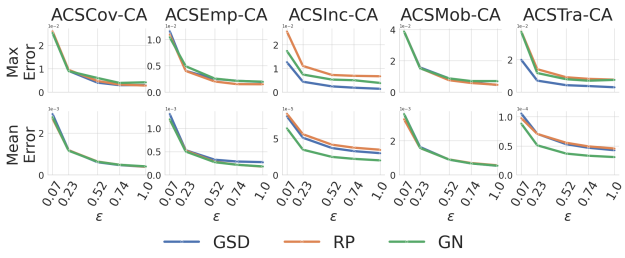

Figure 5. **2-way Categorical Marginals:** For this experiment, we ran the one-shot version of PRIVATE-GSD and RAP. The plot shows the max and average error evaluated on all ACS tasks with differentiable queries— 2-way categorical marginal queries.

[9] We note that PGM cannot be run with the one-shot mechanism since the size of graphical model becomes to large.

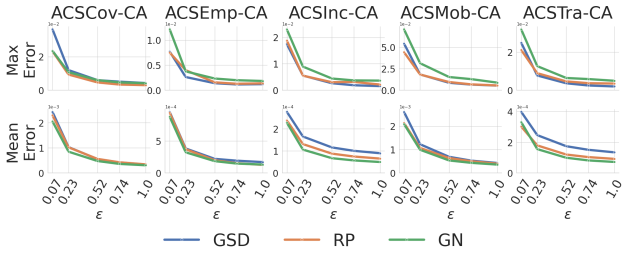

Figure 6. **2-way Binary Tree Marginals:** For this experiment, we ran the one-shot version of PRIVATE-GSD and RAP. The plot shows the max and average error evaluated on all ACS tasks with 2-way *range* marginal queries.

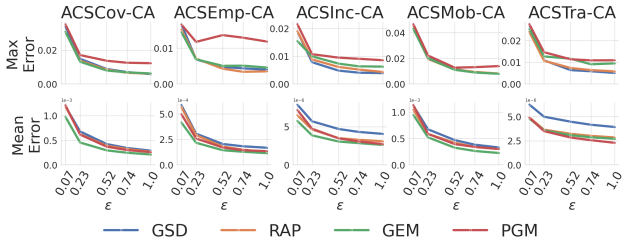

Figure 7. **Adaptive 3-way Categorical Marginals:** For this experiment, we ran the *adaptive* version of PRIVATE-GSD, RAP, GEM and PGM. The plot shows the max error evaluated on all ACS tasks with 3-way *categorical* marginal queries.

6. Conclusion

We present PRIVATE-GSD, a versatile genetic algorithm that can generate synthetic data to approximate a wide range of statistical queries, regardless of their differentiability. Our empirical evaluation of the PRIVATE-GSD mechanism on mixed-type data demonstrates that PRIVATE-GSD outperforms the state-of-the-art method, RAP++, on non-differentiable queries and matches first-order optimization methods for differentiable ones. Overall, genetic algorithms present a promising solution for generating mixed-type synthetic data, and we hope our work will inspire further exploration of such approaches in privacy research.The PRIVATE-GSD source code is publicly available at `https://github.com/giusevtr/private_gsd`.

Acknowledgements

We thank Gokul Swamy for his valuable input during our preliminary conversations, particularly his recommendation to explore genetic algorithms, proved instrumental in our research. Additionally, we thank Shuai Tang, whose assistance facilitated the successful execution of our experiments. TL and ZSW were supported in part by the NSF Award #2120667 and a Cisco Research Grant.

References

Aydöre, S., Brown, W., Kearns, M., Kenthapadi, K., Melis, L., Roth, A., and Siva, A. A. Differentially private query release through adaptive projection. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 457–467. PMLR, 2021. URL http://proceedings.mlr.press/v139/aydore21a.html.

Blum, A., Ligett, K., and Roth, A. A learning theory approach to non-interactive database privacy. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pp. 609–618, 2008.

Bun, M. and Steinke, T. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Proceedings of the 14th Conference on Theory of Cryptography*, TCC '16-B, pp. 635–658, Berlin, Heidelberg, 2016a. Springer.

Bun, M. and Steinke, T. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography: 14th International Conference, TCC 2016-B, Beijing, China, October 31-November 3, 2016, Proceedings, Part I*, pp. 635–658. Springer, 2016b.

Chan, T.-H. H., Shi, E., and Song, D. Private and continual release of statistics. *ACM Transactions on Information and System Security (TISSEC)*, 14(3):1–24, 2011.

Ding, F., Hardt, M., Miller, J., and Schmidt, L. Retiring adult: New datasets for fair machine learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Durfee, D. and Rogers, R. M. Practical differentially private top-k selection with pay-what-you-get composition. *Advances in Neural Information Processing Systems*, 32, 2019.

Dwork, C. and Rothblum, G. N. Concentrated differential privacy, 2016.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pp. 265–284. Springer, 2006.

Dwork, C., Nikolov, A., and Talwar, K. Efficient algorithms for privately releasing marginals via convex relaxations. *Discret. Comput. Geom.*, 53(3):650–673, 2015. doi: 10.1007/s00454-015-9678-x. URL https://doi.org/10.1007/s00454-015-9678-x.

Gaboardi, M., Arias, E. J. G., Hsu, J., Roth, A., and Wu, Z. S. Dual query: Practical private query release for high dimensional data. In *International Conference on Machine Learning*, pp. 1170–1178. PMLR, 2014a.

Gaboardi, M., Gallego-Arias, E. J., Hsu, J., Roth, A., and Wu, Z. S. Dual query: Practical private query release for high dimensional data. In *International Conference on Machine Learning*, pp. 1170–1178, 2014b.

Harder, F., Adamczewski, K., and Park, M. Dp-merf: Differentially private mean embeddings with randomfeatures for practical privacy-preserving data generation. In *International Conference on Artificial Intelligence and Statistics*, pp. 1819–1827. PMLR, 2021.

Hardt, M. and Talwar, K. On the geometry of differential privacy. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pp. 705–714, 2010.

Hardt, M., Ligett, K., and McSherry, F. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems*, pp. 2339–2347, 2012.

Holland, J. H. Genetic algorithms. *Scientific american*, 267 (1):66–73, 1992.

Jordon, J., Yoon, J., and van der Schaar, M. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, ICLR '18, 2018a.

Jordon, J., Yoon, J., and van der Schaar, M. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, 2018b.

Kearns, M. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998.

Liu, T., Vietri, G., Steinke, T., Ullman, J., and Wu, Z. S. Leveraging public data for practical private query release. *arXiv preprint arXiv:2102.08598*, 2021a.

Liu, T., Vietri, G., and Wu, Z. S. Iterative methods for private synthetic data: Unifying framework and new methods. *arXiv preprint arXiv:2106.07153*, 2021b.

McKenna, R., Miklau, G., Hay, M., and Machanavajjhala, A. Optimizing error of high-dimensional statistical queries under differential privacy. *Proceedings of the VLDB Endowment*, 11(10):1206–1219, 2018.

McKenna, R., Sheldon, D., and Miklau, G. Graphical-model based estimation and inference for differential privacy. In *International Conference on Machine Learning*, pp. 4435–4444. PMLR, 2019.

McKenna, R., Mullins, B., Sheldon, D., and Miklau, G. Aim: An adaptive and iterative mechanism for differentially private synthetic data. *arXiv preprint arXiv:2201.12677*, 2022.

McSherry, F. and Talwar, K. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pp. 94–103. IEEE, 2007.

Nikolov, A., Talwar, K., and Zhang, L. The geometry of differential privacy: The sparse and approximate cases. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pp. 351–360, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450320290. doi: 10.1145/2488608.2488652. URL https://doi.org/10.1145/2488608.2488652.

Such, F. P., Madhavan, V., Conti, E., Lehman, J., Stanley, K. O., and Clune, J. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

Takagi, S., Takahashi, T., Cao, Y., and Yoshikawa, M. P3GM: Private high-dimensional data release via privacy preserving phased generative model. *arXiv preprint arXiv:2006.12101*, 2020.

Tao, Y., McKenna, R., Hay, M., Machanavajjhala, A., and Miklau, G. Benchmarking differentially private synthetic data generation algorithms. *arXiv preprint arXiv:2112.09238*, 2021.

Torkzadehmahani, R., Kairouz, P., and Paten, B. DP-CGAN: Differentially private synthetic data and label generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

Ullman, J. Answering n $\{2+$ o $(1)\}$ counting queries with differential privacy is hard. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 361–370, 2013.

Ullman, J. and Vadhan, S. PCPs and the hardness of generating private synthetic data. In *Theory of Cryptography Conference*, pp. 400–416. Springer, 2011.

Vietri, G., Tian, G., Bun, M., Steinke, T., and Wu, Z. S. New oracle-efficient algorithms for private synthetic data release. *CoRR*, abs/2007.05453, 2020. URL https://arxiv.org/abs/2007.05453.

Vietri, G., Archambeau, C., Aydore, S., Brown, W., Kearns, M., Roth, A., Siva, A., Tang, S., and Wu, Z. S. Private synthetic data for multitask learning and marginal queries. *arXiv preprint arXiv:2209.07400*, 2022.

Xie, L., Lin, K., Wang, S., Wang, F., and Zhou, J. Differentially private generative adversarial network. *arXiv preprint arXiv:1802.06739*, 2018.

A. Accuracy Analysis of PRIVATE-GSD

Our mechanism inherits similar oracle accuracy guarantees of the (non-adaptive) RAP mechanism, which were proven in Aydöre et al. (2021).

Theorem A.1. *For a discrete data domain $\mathcal{X}$ and any dataset $D \in \mathcal{X}^N$ with N rows. Fix privacy parameters $\epsilon, \delta > 0$, the synthetic dataset size N', and any set of m queries $Q : \mathcal{X}^* \to [0,1]^m$. If the one-shot* PRIVATE-GSD *mechanism solves the minimization in the projection step exactly, then the one-shot* PRIVATE-GSD *mechanism outputs a synthetic data $\widehat{D}$ with average error bounded as*

$$\sqrt{\tfrac{1}{m}\|Q(D) - Q(\widehat{D})\|_2^2} \leq O\left(\frac{((\log(|\mathcal{X}|/\beta)\ln(1/\delta))^{1/4}}{\sqrt{\epsilon \cdot N}} + \frac{\sqrt{\log(m)}}{\sqrt{N'}}\right)$$

Proof. We begin with the projection mechanism, which has the following guarantee proven by Nikolov et al. (2013): Given the discrete data domain $\mathcal{X}$, let $\mathcal{X}^*$ be the set of datasets of arbitrary size with rows from the domain $\mathcal{X}$.

The projection mechanism by Nikolov et al. (2013), perturbs the answers of Q on dataset D using the Gaussian mechanism to obtain a vector $\hat{a}$, and then finds a synthetic dataset $\widehat{D}^* \in \mathcal{X}^*$ by solving the following objective

$$\widehat{D}^* \leftarrow \arg\min_{\widehat{D} \in \mathcal{X}^*} \|\hat{a} - Q(\widehat{D})\|$$

with the following accuracy guarantee:

$$\sqrt{\tfrac{1}{m}\|Q(D) - Q(\widehat{D}^*)\|_2^2} \leq \alpha = O\left(\frac{(\ln(|\mathcal{X}|/\beta)\ln(1/\delta))^{1/4}}{\sqrt{\epsilon \cdot N}}\right)$$

Next, let m be the number of queries in Q. By a simple sampling argument, proven in Lemma 3.7 of Blum et al. (2008), there exists a dataset $\widehat{D}$ of size N' that satisfies the following:

$$\|Q(\widehat{D}) - Q(\widehat{D}^*)\|_\infty \leq \sqrt{\frac{\log(m)}{N'}}$$

Finally, suppose that the output of PRIVATE-GSD is $\widehat{D} \in \mathcal{X}^{N'}$, which is generated by solving an optimization problem exactly as follows:

$$\widehat{D} \leftarrow \arg\min_{\widehat{D} \in \mathcal{X}^{N'}} \|\hat{a} - Q(\widehat{D})\|$$

Then, by triangle inequality:

$$\begin{aligned}
\sqrt{\tfrac{1}{m}\|Q(D) - Q(\widehat{D})\|_2^2} &= \sqrt{\tfrac{1}{m}\|Q(D) - Q(\widehat{D}^*) + Q(\widehat{D}^*) - Q(\widehat{D})\|_2^2} \\
&\leq \sqrt{\tfrac{1}{m}\|Q(D) - Q(\widehat{D}^*)\|_2^2} + \sqrt{\tfrac{1}{m}\|Q(\widehat{D}^*) - Q(\widehat{D})\|_2^2} \qquad \text{(Triangle inequality)} \\
&\leq \sqrt{\tfrac{1}{m}\|Q(D) - Q(\widehat{D}^*)\|_2^2} + \|Q(\widehat{D}) - Q(\widehat{D}^*)\|_\infty \\
&\leq O\left(\frac{(\ln(|\mathcal{X}|/\beta)\ln(1/\delta))^{1/4}}{\sqrt{\epsilon \cdot N}}\right) + \sqrt{\frac{\log(m)}{N'}}
\end{aligned}$$

□

Despite the bounds in Theorem A.1 matching from the bounds in Aydöre et al. (2021), there is no guarantee about the performance PRIVATE-GSD's optimization. Therefore, we empirically show that PRIVATE-GSD has comparable performance to RAP on matching marginal statistics over many different datasets. In Figure 5, we provide results on categorical marginals and with different ACS tasks as defined in Table 5. This brief experiment also shows that, like in the adaptive setting, PRIVATE-GSD achieves similar performance to RAP on differentiable queries in the one-shot setting across many datasets. We use the same hyperparameters used in our adaptive experiments (Table 7), excluding our choices for T, which are not applicable in the one-shot setting.

B. Additional algorithm details

B.1. Privacy Theorem details

We first provide additional preliminaries.

The composition property of zCDP differential privacy is given by the following result:

Lemma B.1 (zCDP Composition, (Bun & Steinke, 2016b)). *Let $\mathcal{A}_1 : \mathcal{X}^N \to R_1$ be ρ_1-zCDP. Let $\mathcal{A}_2 : \mathcal{X}^n \times R_1 \to R_2$ be such that $\mathcal{A}_2(\cdot, r)$ is ρ_2-zCDP for every $r \in R_1$. Then the algorithm $\mathcal{A}(D)$ that computes $r_1 = \mathcal{A}_1(D), r_2 = \mathcal{A}_2(D, r_1)$ and outputs (r_1, r_2) satisfies $(\rho_1 + \rho_2)$-zCDP.*

Any mechanism that satisfies zCDP also satisfies DP.

Theorem B.2 (zCDP to DP). *If M provides ρ-zCDP, then M is $(\rho + 2\sqrt{\rho \log(1/\delta)}, \delta)$-differentially private for any $\delta > 0$.*

We will use the post-procession properties of zCDP in our analysis. Essentially, there is no loss of privacy when doing an operation on the output of a zCDP mechanism.

Lemma B.3 (Post-processing). *Let $M : \mathcal{X}^N \to \mathcal{Y}$ and $f : \mathcal{Y} \to \mathcal{Z}$ be randomized algorithms. Suppose M satisfies ρ-zCDP. Define $M' : \mathcal{X}^N \to \mathcal{Z}$ by $M'(X) = f(M(X))$. Then M' satisfies ρ-zCDP.*

We use the Gaussian mechanism for privately estimating a subset of statistical queries, which perturbs the statistics with Gaussian noise scaled by the queries' sensitivity. A query of the data has low sensitivity if a small change in the data produces a small change in the output. Formally, let $\Delta(Q)$ denote the ℓ_2-sensitivity of vector-valued query Q, where $\Delta(Q) = \max_{\text{neighboring } D,D'} \|Q(D) - Q(D')\|_2$.

Definition B.4 (Gaussian Mechanism). Let $Q : \mathcal{X}^N \to \mathbb{R}^m$ be a function of the dataset. The Gaussian mechanism on input $D \in \mathcal{X}^N$, Q, and $\rho > 0$, outputs $Q(D) + \mathcal{N}\left(0, \frac{\Delta(Q)^2}{2\rho} \cdot \mathbb{I}\right)$. And the Gaussian mechanism satisfies ρ-zCDP (Bun & Steinke, 2016b).

We will also use the private selection algorithm *report noisy max* (Durfee & Rogers, 2019) with Gumbel perturbation noise, whose output distribution is identical to that of the exponential mechanism (McSherry & Talwar, 2007).

Definition B.5 (Exponential Mechanism). Given some dataset $D \in \mathcal{X}^*$, arbitrary range $\mathcal{R}$, and score function $S : \mathcal{X}^* \times \mathcal{R} \to \mathbb{R}$, the exponential mechanism $\mathcal{M}_E(D, S, \mathcal{R}, \varepsilon)$ selects and outputs an element $r \in \mathcal{R}$ with probability proportional to $\exp\left(\frac{\varepsilon S(D,r)}{2\Delta_S}\right)$, where Δ_S is the sensitivity of S.

Definition B.6 (Report Noisy Max With Gumbel Noise). The Report Noisy Max mechanism $RNM(D, q, a, \rho)$ takes as input a dataset $D \in \mathcal{X}^N$, a vector of m statistical queries q, a vector of m query answers a, and a zCDP parameter ρ. It outputs the index of the query with highest noisy error estimate, $i^* = \arg\max_{i \in [m]}(|q_i(D) - a_i| + Z_i)$ where each $Z_i \sim \text{Gumbel}(1/\sqrt{2\rho}N)$. $RNM(\cdot, q, a, \rho)$ satisfies ρ-zCDP.

We now provide a proof sketch to Theorem 3.1.

Proof. In the adaptive selection framework, the report noisy max (RNM, Definition B.6) and Gaussian mechanism (GM, Definition B.4) are both called $K \cdot S$ times, each with a privacy budget of $\rho' = \rho/(2 \cdot K \cdot S)$. As shown in Definition B.4 and the work of Durfee & Rogers (2019), each individual call to RNM or GM satisfies (ρ')-zCDP. Thus, by the composition properties outlined in Bun & Steinke (2016b)(see Lemma B.1), the combination of $2 \cdot K \cdot S$ ρ'-zCDP mechanisms satisfies ρ-zCDP. □

B.2. Adaptive Selection

We present the Adaptive Selection framework in Algorithm 2, and also provide its privacy proof below.

B.3. RAP++ and its limitations

We present the details of RAP++ 's first-order optimization routine in Algorithm 3. As mentioned before, this technique relaxes the objective to gain differentiability. However, some issues arise. In order to showcase the issue with the relaxation step, we construct an example, where RAP++ fails to optimize even a single prefix query on a dataset with one real-valued

Algorithm 2 Adaptive Framework for Synthetic Data

Input: A private dataset $D \in \mathcal{X}^N$ with N rows and d columns, A set of queries Q, number of adaptive epochs T, number of samples per epoch S, privacy parameter ρ.
Split privacy budget $\rho' = \rho/(2 \cdot T \cdot S)$.
Randomly initialize a synthetic data $\widehat{D}_1$.
for $t \in [T]$ **do**
  for $i \in [S]$ **do**
    Select a query $q_{t,i}$ from Q using a ρ'-zCDP select mechanism and score function

$$\text{score}(q) = \|q(D) - q(\widehat{D}_t)\|_\infty$$

    Measure statistic with Gaussian mechanism:

$$\hat{a}_{t,i} \leftarrow Q(D) + \mathcal{N}\left(0, \tfrac{\Delta(q)}{2 \cdot \rho'}\right)$$

  end for
  Let $\hat{a}_t = (a_{1,1}, \ldots, a_{t,S})$ and $Q_t(\widehat{D}) = (q_{1,1}(\widehat{D}), \ldots, q_{t,S}(\widehat{D}))$.
  Project $\widehat{D}_{t+1} \leftarrow \arg\min_{\widehat{D}} \|\hat{a}_t - Q_t(\widehat{D})\|_2$
end for
return $\widehat{D}_{T+1}$

Algorithm 3 Relaxed Projection with Sigmoid Temperature Annealing

Input: A set of m sigmoid differentiable queries $\{\tilde{q}_i^{[\cdot]}\}_{i\in[m]}$, a set of m target answers $\hat{a} = \{\hat{a}_i\}_{i\in[m]}$, initial inverse temperature $\sigma_1 \in \mathbb{R}^+$, stopping condition $\gamma > 0$, and initial dataset $\widehat{D}_1$.
for $j = 1$ **to** J **do**
  Set inverse temperature $\sigma_j = \sigma_1 \cdot 2^{j-1}$.
  Define the sigmoid differentiable loss function: $\mathrm{L}_j(\widehat{D}) = \sum_{i\in[m]} \left(\tilde{q}_i^{[\sigma_j]}(\widehat{D}) - \hat{a}_i\right)^2$
  Starting with $\widehat{D} \leftarrow \widehat{D}_j$. Run gradient descent on $L_j(\widehat{D})$ until $\|\nabla \mathrm{L}_j(\widehat{D})\| \le \gamma$. Set $\widehat{D}_{j+1} \leftarrow \widehat{D}$.
end for
Output $\widehat{D}_{J+1}$

feature. Let $f_{0.5}(x) = \mathbb{I}\{x \le 0.5\}$ be a prefix function with threshold 0.5, then for a one-dimensional dataset $D \in [0,1]^N$ with N rows the corresponding statistical query is $q_{0.5}(D) = \frac{1}{N}\sum_{x\in D} f_{0.5}(x)$. And, let $\hat{a} = q_{0.5}(D)$ be the true answer of the prefix query on dataset D. Then, RAP++ wants to find a synthetic data $\widehat{D}$ that minimizes the objective $\|\hat{a} - q_{0.5}(\widehat{D})\|_2$, which is non-differentiable due to the threshold query. Therefore, RAP++ optimizes a differentiable relaxation of the objective defined by $q_{0.5}$. The relaxation stop replaces the prefix function $f_{0.5}$, with a *Sigmoid* function, which has well-behaved gradients with adjustable magnitudes via the inverse temperature parameter. A Sigmoid function with threshold 0.5 and *inverse temperature parameter* σ^{it} is given by $f_{0.5}^{[\sigma^{\text{it}}]}(x) = \frac{1}{(1+\exp(-\sigma^{\text{it}}\cdot(x-0.5)))}$

The corresponding Sigmoid approximation to the prefix query is given by $\tilde{q}_{0.5}^{\sigma^{\text{it}}}(D) = \frac{1}{N}\sum_{x\in D} f_{0.5}^{[\sigma^{\text{it}}]}(x)$. The relaxed objective then becomes $\|\hat{a} - \tilde{q}_{0.5}^{\sigma^{\text{it}}}(\widehat{D})\|_2$.

Now, using a simple example, we show that RAP++ can get stuck in a bad local minimum of the differentiable loss function. Suppose that the input data D consist of $N/2$ entries with value equal to 0 and $N/2$ entries with value equal to 1, that is $D = \{0, \ldots, 0, 1, \ldots, 1\} \in [0,1]^N$. Then the input dataset D, evaluates to $q_{0.5}(D) = 0.5$ on the prefix query (half the data points are bellow the 0.5 threshold). Next, consider the synthetic data point $\widehat{D} = \{0.5, \ldots, 0.5\}$, which evaluates to $\tilde{q}_{0.5}(\widehat{D}) = 0.5$ on the Sigmoid prefix function, then, $\widehat{D}$ is a local minimum of the objective function ($\|\hat{a} - \tilde{q}_{0.5}^{\sigma^{\text{it}}}(\widehat{D})\|_2 = 0$). However, when evaluated on the actual prefix query, the synthetic data point $\widehat{D}$ evaluates to $q_{0.5}(\widehat{D}) = 1$. Therefore, the error on the original loss function is 0.5.

This simple example shows how the first-order optimization procedure by Vietri et al. (2022) can get stuck in a bad local minimum, resulting in a poor approximation of the statistical queries. Furthermore, in the experiment section, we will show

that the RAP++ algorithm fails in practice.

B.4. Modifications to RP / GN

Suppose we have a data domain $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_d$ contains d discrete attributes. Then both RP and GN (both the one-shot and adaptive variants) model such features as a mixture of K product distributions over attributes $\mathcal{X}_i$ (and possible values they can take on). We denote each product distribution as $P_i = \prod_{j=1}^{d} P_{ij}$, where P_{ij} is a distribution over the discrete values of $\mathcal{X}_j$. The difference between RP and GN then is how each distribution P_{ij} is parametrized. In RP, the parameters θ are exactly the values P_{ij}. Meanwhile, GN instead outputs the values P_{ij} using a neural network F_θ, which takes in as input Gaussian noise $\mathbf{z} \sim \mathcal{N}(0, I_K)$.

In either case, query answers for k-way categorical queries can be calculated as a product over various columns in P_i. For example (relaxing notation), suppose $P_{i,\text{COLOR=red}}$ and $P_{i,\text{SHAPE=circle}}$ correspond to the probabilities that attributes COLOR $=$ red and SHAPE $=$ circle. Then the 2-way marginal query counting the proportion of samples that are red circles can be written down as the product query $\frac{1}{K}\sum_{i=1}^{K} P_{i,\text{COLOR=red}} \times P_{i,\text{SHAPE=circle}}$.

Now suppose we have some numerical column that has been discretized into bins. Then a k-way range marginal query can be simple written down as a sum of product queries. For example, suppose we have an attribute QUANTITY that has been discretized into bins of size 10. Then the 2-way range query for proportion of samples that are red and have quantity between 1 and 20 can be written down as $\frac{1}{K}\sum_{i=1}^{K} P_{i,\text{COLOR=red}} \times [P_{i,\text{QTY=1-10}} + P_{i,\text{QTY=11-20}}]$. Therefore, we are able to still optimize over such queries, since they are simply sums over (differentiable) product queries that the algorithms were originally designed for.

Finally, while Liu et al. (2021b) and Aydöre et al. (2021) propose different optimization strategies for their algorithms at each round, we introduce a simpler update procedure that we find performs well across our experiments. We present this procedure in Algorithm 4.

Algorithm 4 RP / GN Update

Input: RP/GN$_\theta$, queries (sampled via the exponential mechanism) $Q = \langle q_1, \ldots, q_t \rangle$, noisy answers (measured via the Gaussian mechanism) $A = \langle a_1, \ldots, a_t \rangle$, max iterations M, batch size B
for $m = 1$ **to** $2M$ **do**
  Let synthetic answers $\hat{A} = \text{RAP/GEM}_\theta(Q)$
  Let errors $P = |A - \hat{A}|$
  if $m \bmod 2 = 0$ **then**
    Sample a set of query indices S of size B uniformly
  else
    Sample a set of query indices S of size B proportional to errors P
  end if
  Update θ via stochastic gradient decent according to $\|P_S\|_1$
end for

B.5. Ablation study on PRIVATE-GSD parameters

This sections analyzes different configurations and parameters of PRIVATE-GSD and their effect on the final approximation error and runtime. As described in Algorithm 1, the relevant parameters are the number of generations G, the number of mutations P_{mut}, the number of crossovers P_{cross} and the elite size E.

Population size In this section, we delve into the investigation of parameters P_{mut} and P_{cross}, which guide the generation of candidate synthetic datasets in each iteration. P_{mut} and P_{cross} represent the count of candidates created via the mutation and crossover strategies respectively. Consequently, these population size parameters correspond to the exploration rate of the PRIVATE-GSD method; larger population sizes correspond to a broader array of potential solution candidates at each generation.

Nonetheless, a trade-off exists: larger population sizes incur higher computational costs since PRIVATE-GSD to computes the objective function for synthetic dataset candidate. If we denote the quantity of queries as m and the population size as P, the computational cost of PRIVATE-GSD for each iteration equals $O(P \cdot m)$. Conversely, a reduced population size

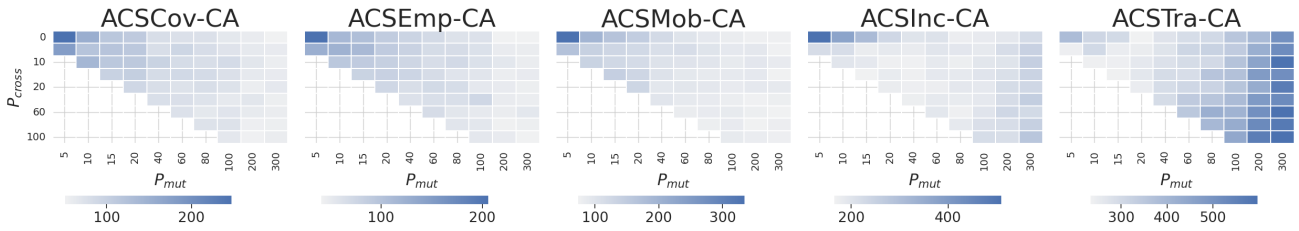

Figure 8. **Ablation Study Runtimes:** Heat maps displaying the total runtime of PRIVATE-GSD optimizing over all 2-way binary-tree marginals in the oneshot setting. This is carried out for varying P_{mut} and P_{cross} parameters across five datasets. The constant parameters are: $G = 300000$, $N' = 1000$, $\epsilon = 1$, and $\delta = 1/N^2$. The PRIVATE-GSD algorithm incorporates the early stopping rule described in Section 4. Darker colors correspond to longer runtimes for each P_{mut}/ P_{cross} pairing. A color-runtime legend is provided under each heat map. For a summary of error, runtime, and query set size in this ablation study, refer to Table 3.

$P = P_{\text{mut}} + P_{\text{cross}}$ may result in slower convergence.

To identify the optimal population size, we conduct an experiment where we adjust the population size and observed its impact on the final error and computation time. We run PRIVATE-GSD that outputs a synthetic dataset matching all 2-way marginal queries on five separate ACS datasets. For each dataset, we initiate PRIVATE-GSD with different P_{mut} and P_{cross} values, maintaining $P_{\text{mut}} \leq P_{\text{cross}}$ and keeping other parameters constant. The privacy parameter ϵ is set to 1, synthetic data rows to $N' = 1000$, and the number of iterations to $G = 300,000$ with early stopping.

We present our experimental findings in figures Table 3 and Figure 8. Table 3 offers insights into the impact of the population size parameters on the final error and runtime. Notably, while the population size parameters only moderately affect the final error, they substantially influence the computational time. Given the stability of the PRIVATE-GSD error across different parameters, we next visualized the effect of these parameters on computational time in Figure 8.

The heatmap in Figure 8 presents total computational time for each parameter combination, where darker hues indicate longer computational times. From this visualization, it is evident that the optimal population size is contingent on the dataset in question. For instance, while the ACSCOV-CA, ACSEMP-CA, and ACSMOB-CA datasets benefited from larger population sizes, the ACSINC-CA and ACSTRA-CA datasets performed better with smaller ones. However, we find that a robust parameter configuration that yields consistently good performance across all tasks was found to be $P_{\text{mut}} = 100$ and $P_{\text{cross}} = 100$.

Data	Queries	Min Average Error	Max Average Error	Min Run Time(s)	Max Run Time(s)
ACSCOV-CA	15894	0.0005921	0.0006259	50	247
ACSEMP-CA	10570	0.0004293	0.0004939	41	206
ACSMOB-CA	38077	0.0005560	0.0005838	73	335
ACSINC-CA	274641	0.0000761	0.0000849	165	510
ACSTRA-CA	438594	0.0000960	0.0001012	234	598

Table 3. **Ablation Study Summary:** A summary of the PRIVATE-GSD ablation study outlined in Appendix B.5. The study entails running PRIVATE-GSD to optimize over all 2-way binary-tree marginals in a oneshot setting, testing various P_{mut} and P_{cross} parameter combinations. The 'Queries' column shows the total size of the query set designated for each dataset task. The remaining columns indicate best and worst case error/runtime for each (P_{mut}, P_{cross}) pairing per dataset. This summary suggests that PRIVATE-GSD error is relatively insensitive to P_{mut} and P_{cross} parameters, but the choice of these parameters significantly affects runtime.

.

Genetic Operators Rate In this section, we investigate an alternate approach to the PRIVATE-GSD algorithm that formulates new candidates by modifying multiple rows of synthetic data via either mutation or crossover operations. Typically, as detailed in Section 4, the PRIVATE-GSD method generates a new candidate synthetic data by adjusting a single entry of the optimal synthetic dataset, $\widehat{D}_g^\star$. However, by applying genetic operators to multiple rows of $\widehat{D}_g^\star$, we can potentially accelerate the convergence rate. We hence introduce 'mutation rate' and 'crossover rate' to denote the number of rows altered in each operation. For example, if the mutation rate is k, each candidate solution generated using the mutation

strategy will differ from $\widehat{D}_g^\star$ at k locations.

We conduct an experiment to understand the influence of these rates, in which we vary both the mutation rate and the crossover rate. We run PRIVATE-GSD on ACSMOB-CA and with all 2-way marginal statistics. The privacy parameter is set to $\epsilon = 1$, the maximum number of generations capped at $100,000$, the population size constrained to 100, and the size of the synthetic data set at $1,000$.

Table 4 depicts the final average error and the corresponding runtime in seconds for each combination of mutation rate and crossover rate. Our findings indicate that both the mutation rate and the crossover rate have a significant impact on error and run time. Interestingly, a mutation rate and crossover rate of 1 yielded the best average error and run time across all combinations, establishing it as a consistently dependable choice in our experiment. This result emphasizes the importance of selecting these rates carefully for optimal algorithm performance. It further provides evidence that amplifying the level of perturbation in the candidate generation step can destabilize PRIVATE-GSD and lead to suboptimal performance.

		Mutation Rate				
		1	2	5	10	25
Crossover Rate	1	**0.0476 / (83)**	0.0474 / (102)	0.0495 / (129)	0.0553 / (162)	0.0826 / (266)
	2	0.0480 / (117)	0.0487 / (116)	0.0507 / (135)	0.0590 / (168)	0.0860 / (272)
	5	0.0542 / (127)	0.0562 / (135)	0.0617 / (154)	0.0728 / (189)	0.0995 / (293)
	10	0.0825 / (161)	0.0839 / (168)	0.0908 / (189)	0.0987 / (223)	0.1216 / (327)
	25	0.1922 / (264)	0.1942 / (272)	0.1918 / (291)	0.1905 / (326)	0.1945 / (435)

Table 4. Effect of 'mutation rate' and 'crossover rate'. The numbers represent the average error for answering 2-way range queries and, in parenthesis, it is the total runtime in seconds. Results are averaged over 3 runs. For all experiments we fixed the following parameters: Input data is ACSMOB-CA, the max number of generations was $100,000$, the synthetic data size was $1,000$, $P_{\text{mut}} = 50$ and $P_{\text{cross}} = 50$, and the privacy parameter $\epsilon = 1$.

C. ML Evaluation

This section critically assesses the utility of synthetic data in training machine learning models. We explore two main parameters in our investigation: the choice of mechanisms, such as PRIVATE-GSD or RAP++, and the selection of statistics to match, including categorical marginals, Binary Tree, or Halfspace queries.

We direct our attention toward datasets that encapsulate multiple classification tasks concurrently. To this end, we construct a multitask dataset that amalgamates all five predictive tasks outlined in Figure 9 by combining all feature and target columns. The learning challenge presented by a multitask dataset involves developing separate models, each predicting one of the target columns based on the feature columns. We exclude the employment task because its target is highly correlated with feature from another tasks, making the prediction problem trivial. Thus, the multitask dataset incorporates 25 categorical features, nine numerical features, and four binary target labels.

We initiate the experiment by dividing each dataset into a training and test set, using an 80/20 partition. The training set is subsequently processed by a private mechanism to produce a synthetic dataset that we use to train a logistic regression model. Lastly, the performance of the model is evaluated on the test data using the F1 metric. The F1 metric considers both precision and recall, providing a balanced assessment of model performance, particularly when dealing with imbalanced datasets. It is calculated as the harmonic mean of precision and recall, with a potential range from 0 to 1, wherein a higher score denotes superior model performance.

Our results are tabulated in Table ML and showcase the F1 score for predicting four target labels using logistic regression trained on various synthetic data methodologies alongside the F1 score of a logistic regression model trained on the original training data. The findings suggest that the PRIVATE-GSD mechanism coupled with Binary-Tree queries is a favorable choice among the other synthetic data baselines examined in this experiment. However, we note that the F1 scores of all private synthetic data methodologies are inferior to the model trained on the original data, even in a low privacy regime (i.e., $\epsilon = 1$).

The question of why synthetic data does not get better performance in training ML models remains an open area of exploration. One conjecture is that we have yet to identify the right set of data properties (or statistical queries) that can be used to generate high-quality synthetic data for machine learning applications. However, we posit that the PRIVATE-GSD

mechanism is well-equipped to tackle this challenge due to its inherent ability and flexibility in managing diverse queries.

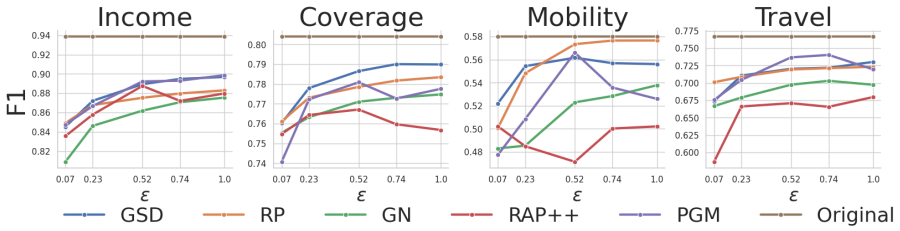

Figure 9. **Machine Learning Evaluation:** F1 test score of logistic regression trained on synthetic data. Each ACS dataset is partitioned into training and testing set, using an 80/20 split. The train set is used to train the synthetic data algorithm, and the F1 score is computed using the test set. Each algorithm is trained either on Binary Tree (BT) marginal queries or Halfspace (HS) queries. We fix $\delta = 1/N^2$ and vary $\varepsilon \in \{0.07, 0.23, 0.52, 0.74, 1.0\}$. Results are averaged over 3 runs.

D. Additional experiment details

In this section we include additional details about our experiments, including information about datasets, queries, and hyperparameters.

Data. In Table 5, we list all attributes for each ACS task that are used in empirical results presented in either the main paper or this appendix.

Table 5. We list the ACS data attributes used in our experiments.

Dataset	Categorical Attributes	Numeric Attributes
ACSMOBILITY	MIL, DREM, CIT, DIS, COW, MAR, SCHL, MIG, NATIVITY ANC, DEAR, DEYE, SEX, ESR, GCL, RAC1P, RELP	JWMNP, PINCP WKHP, AGEP
ACSEMPLOYMENT	COW, OCCP, SCHL, MAR, SEX, PINCP, RAC1P, RELP, POBP	AGEP
ACSCOVERAGE	MIL, FER, DREM, CIT, DIS, MAR, SCHL, PUBCOV MIG, NATIVITY, ANC, DEYE, SEX, ESR, RAC1P, DEAR, ESP	PINCP, AGEP
ACSINCOME	COW, OCCP, SCHL, MAR, SEX, PINCP, RAC1P, RELP, POBP	WKHP, AGEP
ACSTRAVEL	JWMNP, PUMA, CIT, DIS, OCCP, SCHL, JWTR, MAR MIG, SEX, RAC1P, POWPUMA, RELP, ESP	AGEP, POVPIP
ACSMULTITASK	OCCP, MIG, CIT, ESP, ANC, PUMA, FER, DEAR, DEYE ESR, FOCCP, JWTR, WAOB, JWMNP_bin, DREM, GCL MIL, SCHL, RELP, MAR, NATIVITY, POWPUMA RAC1P, DIS, COW, PUBCOV, SEX, PINCP	INTP, WAGP SEMP, POVPIP WKHP, JWRIP AGEP

Table 6. Number of workloads for categorical and range marginal queries used in our experiments.

	mobility	coverage	employment	income	travel
2-way Cat. Marginals	136	136	120	36	91
3-way Cat. Marginals	680	680	560	84	364
2-way BT Marginals	68	34	16	18	28

Hyperparameters. In Table 7, we list the hyperparameters used for our algorithms.

Query Class	Method	Hyperparameter	Values
All Experiments	PRIVATE-GSD	Data Size	2000
		P_{mut}	$500/50$
		P_{cross}	$500/50$
		Elite Size	2
		Max Generations	200000
	RAP++	Data Size (N')	1000
		Queries Sampled (K)	1000
		Learning Rate	0.0005, 0.001, 0.002, 0.005, 0.01
		Inverse Temp. (σ^{it})	2, 4, 8, 16, 32, 64, 128 256, 512, 1024, 2048
	RAP / GEM	# Product Mixtures (K)	1000
		Batch Size (B)	5000
		Max Iterations (M)	1000
		# Samples	50000
	RAP	LR	0.03
	GEM	LR	0.0003
		Hidden Layers	$(128, 256)$
		Noise Dimension	16
	PGM+EM	Max Iterations	250
		# Samples	N
Halfspace / Prefix	PRIVATE-GSD	T	25, 50, 75, 100
	RAP++	T	3, 4, 5, 6, 7, 8, 9
3-way Categorical	All algorithms	T	25, 50, 75, 100

Table 7. Hyperparameters experiments (with adaptivity).

D.1. Early Stop

In all experiments, PRIVATE-GSD operates over a maximum of G generations. However, we employ an early-stop rule if the algorithm has reached a satisfactory level of convergence in optimizing its objective. This not only aids in achieving the desired result more swiftly but also conserves computational resources.

To determine the convergence of the algorithm, we utilize a time-window approach. We define a window size, w (where $w < G$), for assessing the early-stop condition. The algorithm is halted if the percent change in loss between the current generation, g, and the generation $g - w$ is less than a predetermined threshold. In all cases in this paper, the threshold is set to 0.0001. If the change in loss between the two generations within the window falls below this threshold, the algorithm is halted prematurely. This allows us to conclude that the algorithm has sufficiently converged and further iterations would not significantly contribute to the optimization of the objective. Additionally, the size of the time-window, w, is set to match the size of the synthetic dataset, N'. The proportionality of the window size to the dataset size ensures that our early-stop condition is adapted to the scale of the problem at hand.