PrivSyn: Differentially Private Data Synthesis

Zhikun Zhang[1,2] Tianhao Wang[3] Ninghui Li[3] Jean Honorio[3] Michael Backes[2]
Shibo He[1,4] Jiming Chen[1,4] Yang Zhang[2]

[1]*Zhejiang University* [2]*CISPA Helmholtz Center for Information Security* [3]*Purdue University*
[4]*Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies*

Abstract

In differential privacy (DP), a challenging problem is to generate synthetic datasets that efficiently capture the useful information in the private data. The synthetic dataset enables any task to be done without privacy concern and modification to existing algorithms. In this paper, we present PrivSyn, the first automatic synthetic data generation method that can handle general tabular datasets (with 100 attributes and domain size $> 2^{500}$). PrivSyn is composed of a new method to automatically and privately identify correlations in the data, and a novel method to generate sample data from a dense graphic model. We extensively evaluate different methods on multiple datasets to demonstrate the performance of our method.

1 Introduction

Differential privacy (DP) [21] has been accepted as the *de facto* notion for protecting privacy. Companies and government agencies use DP for privacy-preserving data analysis. Uber implements Flex [30] that answers data SQL queries with DP. LinkedIn builds Pinot [45], a DP platform that enables analysts to gain insights about its members' content engagements. Within the government, the US census bureau plans to publish the 2020 census statistics with DP [5].

Previous work on DP mostly focuses on designing tailored algorithms for specific data analysis tasks. This paradigm is time consuming, requires a lot of expertise knowledge, and is error-prone. For example, many algorithms have been proposed for mining frequent itemset [34, 38, 50]. Some of them incorrectly use the Sparse Vector Technique (SVT) and results in non-private algorithm being incorrectly proven to satisfy DP, see, e.g., [40] for an analysis of incorrect usage of SVT. To answer SQL queries under the constraint of DP, the SQL engine needs to be patched [30]. For another example, to train a differentially private deep neural network, the stochastic gradient descent step is modified [3]. Moreover, this paradigm does not scale: more tasks lead to worse privacy guarantee as each task reveals more information about the private data.

One promising solution to address this problem is generating a synthetic dataset that is similar to the private dataset while satisfying differential privacy. As additional data analysis tasks performed on the published dataset are post-processing, they can be performed without additional privacy cost. Furthermore, existing algorithms for performing data analysis do not need to be modified.

The most promising existing method for private generation of synthetic datasets uses probabilistic graphical models. PrivBayes [53] uses a Bayesian network. It first privately determines the network structure, then obtains noisy marginals for the Conditional Probability Distribution of each node. More recently, PGM, which uses Markov Random Fields, was proposed in [41]. In 2018, NIST hosted a Differential Privacy Synthetic Data Challenge [43], PGM achieves the best result. Approaches that do not use probabilistic graphical models, such as [4, 11, 13, 27–29, 46, 49, 54], either are computationally inefficient or have poor empirical performance.

PrivBayes and PGM have two limitations. First, as a graphical model aims to provide a compact representation of joint probability distributions, it is sparse by design. Once a structure is fixed, it imposes conditional independence assumptions that may not exist in the dataset. Second, since each model is sparse, the structure is data dependent and finding the right structure is critically important for the utility. Bayesian Networks are typically constructed by iterative selection using mutual information metrics. However, mutual information has high sensitivity, and cannot be estimated accurately under DP. PrivBayes introduces a low-sensitivity proxy for mutual information, but it is slow (quadratic to the number of users in the dataset) to compute. In [41], no method for automatically determining the graph structure is provided. In the NIST challenge, manually constructed graph networks are used for PGM.

Our Contributions. In this paper, we propose PrivSyn, for differentially private synthetic data generation. The first novel contribution is that, instead of using graphical models as the summarization/representation of a dataset, we propose to **use a set of large number of low-degree marginals to repre-**

sent a dataset. For example, in the experiments, given around 100 attributes, our method uses all one-way marginals and around 500 two-way marginals. A two-way marginal (specified by two attributes) is a frequency distribution table, showing the number of records with each possible combination of values for the two attributes. At a high level, graphical models can be viewed as a parametric approach to data summarization, and our approach can be viewed as a non-parametric one. The advantage of our approach is that it makes weak assumptions about the conditional independence among attributes, and simply tries to capture correlation relationships that are in the dataset.

This method is especially attractive under DP for several reasons. First, since counting the number of records has a low sensitivity of 1, counting queries can be answered accurately. Second, since a marginal issues many counting queries (one for each cell) with the same privacy cost of one counting query, it is arguably the most efficient way to extract information from a dataset under DP. Third, using either advanced composition theorem [19] or zero-Concentrated DP [14], the variance of noises added to each marginal grows only linearly with the number of marginals under the same privacy budget. Furthermore, when one attribute is included in multiple marginal, one can use averaging to reduce the variance. As a result, one can afford to get a large number of marginals with reasonable accuracy.

There are two main challenges for using a set of marginals for private data synthesis. The first challenge is how to select which marginals to use. Using too many marginals (such as all 2-way marginals) results in higher noises, and slow down data synthesis. The second challenge is how to synthesize the dataset given noisy marginals.

The second contribution is that we propose **a new method to automatically and privately select the marginals.** We first propose a metric InDif (stands for Independent Difference) that measures the correlation between pairwise attributes. InDif is easy to compute and has low global sensitivity. Given InDif scores, we then propose a greedy algorithm that selects the pairs to form marginals.

The third contribution is that we develop a method that **iteratively update a synthetic dataset to make it match the target set of marginals**. When the number of attribute is small enough so that the full contingency table can be stored and manipulated directly, one can use methods such as multiplicative update [8] to do this. However, with tens or even over one hundred attributes, it is infeasible to represent the full contingency table.

The key idea underlying our approach is to view the dataset being synthesized as a proxy of the joint distribution to be estimated, and directly manipulate this dataset. In particular, given a set of noisy marginals, we start from a randomly generated dataset where each attribute matches one-way marginal information in the set, and then gradually "massage" the synthetic dataset so that its distribution is closer and closer to each pairwise marginal. We model this problem as a network flow problem and propose Graduate Update Method (short for GUM), a method to "massage" the dataset to be consistent with all the noisy marginals. We believe that GUM can be of independent interest outside the privacy community. Essentially, it can be utilized more broadly as a standalone algorithm and it allows us to generate synthetic dataset from dense graphical models.

To summarize, the main contributions of this paper are:

- A simple yet efficient method to capture correlations within the dataset.
- A new method to automatically and privately select marginals that capture sufficient correlations.
- A data synthesis algorithm GUM that can also be used standalone to handle dense graphical models.
- An extensive evaluation which demonstrates the performance improvement of the proposed method on real-world dataset and helps us understand the intuition of different techniques.

Roadmap. In Section 2, we present background knowledge of DP and composition theorem, and formally define the data synthesis problem. We then introduce a general framework of private data synthesis in Section 3. We present our proposed marginal selection method and data synthesis method in Section 4 and Section 5, respectively. Experimental results are presented in Section 6. We discuss related work in Section 7 and limitations in Section 8. Finally, we provide concluding remarks in Section 9.

2 Preliminaries

2.1 Differential Privacy

Differential privacy [22] is designed for the setting where there is a **trusted data curator**, which gathers data from individual users, processes the data in a way that satisfies DP, and then publishes the results. Intuitively, the DP notion requires that any single element in a dataset has only a limited impact on the output.

Definition 1 ((ε,δ)-Differential Privacy)**.** *An algorithm $\mathcal{A}$ satisfies (ε,δ)-differential privacy ((ε,δ)-DP), where $\varepsilon > 0, \delta \geq 0$, if and only if for any two neighboring datasets D and D', we have*

$$\forall T \subseteq Range(\mathcal{A}) : \Pr[\mathcal{A}(D) \in T] \leq e^{\varepsilon} \Pr\left[\mathcal{A}(D') \in T\right] + \delta,$$

where $Range(\mathcal{A})$ denotes the set of all possible outputs of the algorithm $\mathcal{A}$.

In this paper we consider two datasets D and D' to be *neighbors*, denoted as $D \simeq D'$, if and only if either $D = D' + r$ or $D' = D + r$, where $D + r$ denotes the dataset resulted from adding the record r to the dataset D.

2.2 Gaussian Mechanism

There are several approaches for designing mechanisms that satisfy (ε,δ)-differential privacy. In this paper, we use the Gaussian mechanism. The approach computes a function f on the dataset D in a differentially privately way, by adding to $f(D)$ a random noise. The magnitude of the noise depends on Δ_f, the *global sensitivity* or the ℓ_2 sensitivity of f. Such a mechanism $\mathcal{A}$ is given below:

$$\begin{aligned} \mathcal{A}(D) &= f(D) + \mathcal{N}\left(0, \Delta_f^2\sigma^2\mathbf{I}\right) \\ \text{where} \quad \Delta_f &= \max_{(D,D'):D\simeq D'} \|f(D)-f(D')\|_2. \end{aligned}$$

In the above, $\mathcal{N}(0,\Delta_f^2\sigma^2\mathbf{I})$ denotes a multi-dimensional random variable sampled from the normal distribution with mean 0 and standard deviation $\Delta_f\sigma$, and $\sigma = \sqrt{2\ln\frac{1.25}{\delta}}/\varepsilon$.

2.3 Composition via Zero Concentrated DP

For a sequential of k mechanisms $\mathcal{A}_1,\ldots,\mathcal{A}_k$ satisfying (ε_i,δ_i)-DP for $i=1,\ldots,k$ respectively, the basic composition result [25] shows that the privacy composes linearly, i.e., the sequential composition satisfies $(\sum_i^k \varepsilon_i, \sum_i^k \delta_i)$-DP. When $\varepsilon_i=\varepsilon$ and $\delta_i=\delta$, the advanced composition bound from [19] states that the composition satisfies $(\varepsilon\sqrt{2k\log(1/\delta')}+k\varepsilon(e^{\varepsilon}-1), k\delta+\delta')$-DP.

To enable more complex algorithms and data analysis task via the composition of multiple differentially private building blocks, zero Concentrated Differential Privacy (zCDP for short) offers elegant composition properties. The general idea is to connect (ε,δ)-DP to Rényi divergence, and use the useful property of Rényi divergence to achieve tighter composition property. In another word, for fixed privacy budget ε and δ, zCDP can provide smaller standard deviation for each task compared to other composition techniques. Formally, zCDP is defined as follows:

Definition 2 (Zero-Concentrated Differential Privacy (zCDP) [14])**.** *A randomized mechanism $\mathcal{A}$ is ρ-zero concentrated differentially private (i.e., ρ-zCDP) if for any two neighboring databases D and D' and all $\alpha \in (1,\infty)$,*

$$\mathcal{D}_\alpha(\mathcal{A}(D)||\mathcal{A}(D')) \triangleq \frac{1}{\alpha-1}\log\left(\mathbb{E}\left[e^{(\alpha-1)L^{(o)}}\right]\right) \leq \rho\alpha$$

Where $\mathcal{D}_\alpha(\mathcal{A}(D)||\mathcal{A}(D'))$ is called α-Rényi divergence between the distributions of $\mathcal{A}(D)$ and $\mathcal{A}(D')$. L^o is the privacy loss random variable with probability density function $f(x)=\log\frac{\Pr[\mathcal{A}(D)=x]}{\Pr[\mathcal{A}(D')=x]}$.

zCDP has a simple linear composition property [14]:

Theorem 1. *Two randomized mechanisms $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy ρ_1-zCDP and ρ_2-zCDP respectively, their sequential composition $\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)$ satisfies $(\rho_1+\rho_2)$-zCDP.*

The following two theorems restate the results from [14], which are useful for composing Gaussian mechanisms in differential privacy.

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

Theorem 3. *The Gaussian mechanism which answers $f(D)$ with noise $\mathcal{N}(0,\Delta_f^2\sigma^2\mathbf{I})$ satisfies $(\frac{1}{2\sigma^2})$-zCDP.*

Given ε and δ, we can calculate the amount of noise for each task using Theorem 1 to Theorem 3. In particular, we first use Theorem 2 to compute the total ρ allowed. Then we use Theorem 1 to allocate ρ_i for each task i. Finally, we use Theorem 3 to calculate σ for each task. Compared with (ε,δ)-DP, zCDP provides a tighter bound on the cumulative privacy loss under composition, making it more suitable for algorithms consist of a large number of tasks.

2.4 Problem Definition

In this paper, we consider the following problem: *Given a dataset D_o, we want to generate a synthetic dataset D_s that is statistically similar to D_o.* Generating synthetic dataset D_s allows data analyst to handle arbitrary kinds of data analysis tasks on the same set of released data, which is more general than prior work focusing on optimizing the output for specific tasks (*e.g.*, [3, 36, 44, 52]).

More formally, a dataset D is composed of n records each having d attributes. The synthetic dataset D_s is said to be similar to D_o if $f(D_s)$ is close to $f(D_o)$ for any function f. In this paper, we consider three statistical measures: marginal queries, range queries, and classification models. In particular, a marginal query captures the joint distribution of a subset of attributes. A range query counts the number of records whose corresponding values are within the given ranges. Finally, we can also use the synthetic dataset to train classification models and measure the classification accuracy.

3 A Framework of Private Data Synthesis

In this section, we first propose a general framework for generating differentially private synthetic datasets, and then review some existing studies in this framework. PrivSyn follows this framework and proposes novel techniques for each of the component in the framework.

To generate the synthetic dataset in a differentially private way, one needs to first transform the task to estimate a function f with low sensitivity Δ_f. One straightforward approach is to obtain the noisy full distribution, *i.e.*, the joint distribution of all attributes. Given the detailed information about the distribution, one can then generate a synthetic dataset by sampling from the distribution. However, when there are many attributes in the dataset, computing or even storing the full distribution requires exponentially large space. To overcome this issue, one promising approach is to estimate many

Step / Method	Marginal Selection	Noise Addition	Post Processing	Data Synthesis
PriView [44]	Covering design	Equal budget + Laplace	Max-entropy Estimation	-
PrivBayes [53]	Bayesian network + Info Gain (EM)	Equal budget + Laplace	-	Sampling
PGM [41]	- (not dense)	Equal budget + Gaussian	Markov Random Field	Sampling
PrivSyn	Optimization + Greedy	Weighted budget + Gaussian	Consistency	GUM

Table 1: Summary of existing methods on different steps. The four steps are all new. Our marginal selection method enables private auto selection of marginals. GUM enables usage of dense graphical model.

low-degree joint distributions, also called marginals, which are distributions of only a subset of attributes. More specifically, to generate a synthetic dataset, there are four steps: (1) marginal selection, (2) noise addition, (3) post-processing, and (4) data synthesis.

The current best-performing approaches on private data synthesis all follow this approach. Table 1 summarizes these four steps of existing work and our proposed method. In what follows, we review these steps in the reverse order.

3.1 Data Synthesis

To synthesize a dataset, existing work uses graphical models to model the generation of the data. In particular, PrivBayes [53] uses a differentially private Bayesian network. It is a generative model that can be represented by a directed graph. In the graph, each node v represents an attribute, and each edge from u to v corresponds to $\Pr[v|u]$, the probability of u causing v. As each attribute can take multiple values, all possible $\Pr[v=y|u=x]$ are needed. When a node v has more than one nodes $U=\{u_1,\dots,u_k\}$ connected to it, $\Pr[v|U]$ is needed to sample v. Because the causality is a single-direction relationship, the graph cannot contain cycles. To sample a record, we start from the node with in-degree 0. We then traverse the graph to obtain the remaining attributes following the generation order specified by the Bayesian network.

More recently, [41] proposed to sample from differentially private Markov Random Field (MRF). Different from Bayesian network, MRF is represented by undirected graphs, and each edge u,v contains the joint distribution $\Pr[v,u]$. Moreover, cycles or even cliques are allowed in this model. The more complex structures enable capturing higher dimensional correlations, but will make the sampling more challenging. In particular, one first merge cliques into nodes and form a tree structure, which is called junction tree. The data records can then be sampled from it. The main shortcoming of PGM is that, when the graph is dense, the domain of cliques in the junction tree could be too large to handle.

3.2 Marginal Selection

To build a graphical model, joint distributions in the form of $\Pr[v,u]$ are needed (note that conditional distributions $\Pr[v|u]$ can be calculated from joint distributions). The goal is to capture all the joint distributions. However, by the composition property of DP, having more marginals leads to more noise in each of them. We do not want to select too many marginals which leads to excessive noise on each of them.

PrivBayes chooses the marginals by constructing the Bayesian network. In particular, it first randomly assigns an attribute as the first node, and then selects other attributes one by one using Exponential Mechanism (EM). The original Bayesian network uses mutual information as the metric to select the most correlated marginals. In the setting of DP, the sensitivity for mutual information is high. To reduce sensitivity, the authors of [53] proposed a function that is close to mutual information.

Another method PriView [44] uses a data independent method to select the marginals. In particular, a minimal set of marginals are selected so that all pairs or triples of attributes are contained in some marginal. When some attributes are independent, capturing the relationship among them actually increases the amount of noise. This approach cannot scale with the number of attributes d.

Noise Addition. Given the marginals, the next step is to add noise to satisfy DP. The classic approach is to split the privacy budget equally into those marginals and add Laplace noise.

Post Processing. The DP noise introduces inconsistencies, including (1) some estimated probabilities being negative, (2) the estimated probabilities do not sum up to 1, and (3) two marginals that contain common attributes exist inconsistency.

In PrivBayes, negative probabilities are converted to zeros. In PGM, consistencies are implicitly handled by the estimation procedure of the Markov Random Field.

4 Differentially Private Marginal Selection

In the phase of obtaining marginals, there are two sources of errors. One is information loss when some marginals are missed; the other is noise error incurred by DP. PrivBayes chooses few marginals; as a result, useful correlation information from other marginals is missed. On the other hand, PriView is data-independent and tries to cover all the potential correlations; and when there are more than a few dozen attributes, the DP noise becomes too high.

v	$\mathsf{M}_{\text{gender}}(v)$
$\langle$male,$*\rangle$	0.40
$\langle$female,$*\rangle$	0.60

(a) 1-way marginal for gender.

v	$\mathsf{M}_{\text{age}}(v)$
$\langle *$,teenager$\rangle$	0.20
$\langle *$,adult $\rangle$	0.30
$\langle *$,elderly$\rangle$	0.50

(b) 1-way marginal for age.

v	
$\langle$male, teenager$\rangle$	0.08
$\langle$male, adult$\rangle$	0.12
$\langle$male, elderly$\rangle$	0.20
$\langle$female, teenager$\rangle$	0.12
$\langle$female, adult$\rangle$	0.18
$\langle$female, elderly$\rangle$	0.30

(c) 2-way marginal assume indepent

v	
$\langle$male, teenager$\rangle$	0.10
$\langle$male, adult$\rangle$	0.10
$\langle$male, elderly$\rangle$	0.20
$\langle$female, teenager$\rangle$	0.10
$\langle$female, adult$\rangle$	0.20
$\langle$female, elderly$\rangle$	0.30

(d) Actual 2-way marginal

Figure 1: Example of the calculation of InDif.

To balance between the two kinds of information loss, we propose an effective algorithm DenseMarg that is able to choose marginals that capture more useful correlations even under very low privacy budget.

4.1 Dependency Measurement

To select marginals that capture most of the correlation information, one needs a metric to measure the correlation level. In Bayesian network, mutual information is used to capture pair-wise correlation. As the sensitivity for mutual information is high, the authors of [53] proposed a function that can approximate the mutual information. However, the function is slow (quadratic to the number of users in the dataset) to compute.

To compute correlation in a simple and efficient way, in this subsection, we propose a metric which we call Independent Difference (InDif for short). For any two attributes a,b, InDif calculates the ℓ_1 distance between the 2-way marginal $\mathsf{M}_{a,b}$ and 2-way marginal generated assuming independence $\mathsf{M}_a \times \mathsf{M}_b$, where a marginal M_A specified by a set of attributes A is a frequency distribution table, showing the frequency with each possible combination of values for the attributes, and $\times$ denote the outer product, *i.e.*, $\text{InDif}_{a,b} = |\mathsf{M}_{a,b} - \mathsf{M}_a \times \mathsf{M}_b|_1$.

Figure 1 gives an example to illustrate the calculation of InDif. The 2-way marginal in Figure 1c is directly calculated by the 1-way marginal of gender and age, without analyzing the dataset; and Figure 1d gives the actual 2-way marginal. In this example, $\text{InDif} = 0.08 \cdot n$, where n is the number of records. The advantage of using InDif is that it is easy to compute, and it has low sensitivity in terms of its range, $[0,2n]$:

Lemma 4. *The sensitivity of InDif metric is 4: $\Delta_{InDif} = 4$.*

The proof is deferred to Appendix A. Given d attributes, we use the Gaussian mechanism to privately obtain all InDif scores. To evaluate the impact of noise, one should consider both sensitivity and range of the metrics. We theoretically and empirically analyze the noise-range ratio of entropy-based metrics and InDif in Appendix B, and show that InDif has smaller noise-range ratio than entropy-based metrics. More specifically, given the overall privacy parameters (ε,δ), we first compute the parameter ρ using Theorem 2. We then use $\rho' < \rho$ for publishing all the InDif scores for all $m = \binom{d}{2}$ pairs of attributes. In particular, with the composition theory of zCDP, we can show that publishing all InDif scores with Gaussian noise $\mathcal{N}(0, 8m/\rho'\mathbf{I})$ satisfies ρ'-zCDP (its proof is also deferred to Appendix A).

Theorem 5. *Given d attributes, publishing all $m = d(d-1)/2$ InDif scores with Gaussian noise $\mathcal{N}(0, 8m/\rho'\mathbf{I})$ satisfies ρ'-zCDP.*

4.2 Marginal Selection

Given the dependency scores InDif, the next step is to choose the pairs with high correlation, and use the Gaussian mechanism to publish marginals on those pairs. In this process, there are two error sources. One is the noise error introduced by the Gaussian noise; the other is the dependency error when some of the marginals are not selected. If we choose to publish all 2-way marginals, the noise error will be high and there is no dependency error; when we skip some marginals, the error for those marginals will be dominated by the dependency error.

Problem Formulation. Given m pairs of attributes, each pair i is associated with an indicator variable x_i that equals 1 if pair i is selected, and 0 otherwise. Define ψ_i as the noise error introduced by the Gaussian noise and ϕ_i as its dependency error. The marginal selection problem is formulated as the following optimization problem:

$$\begin{aligned} &\text{minimize} \sum_{i=1}^{m} [\psi_i x_i + \phi_i (1 - x_i)] \\ &\text{subject to } x_i \in \{0,1\} \end{aligned}$$

Notice that the dependency error ϕ_i has positive correlation with InDif_i, *i.e.*, larger InDif_i incurs larger ϕ_i. Thus, we approximate ϕ_i as $\text{InDif}_i + \mathcal{N}(0, m^2\rho'^2\mathbf{I})$, and it is fixed in the optimization problem.

The noise error ψ_i is dependent on the privacy budget ρ_i allocated to the pair i. In particular, we first show that given the true marginal M_i, we add Gaussian noise with scale $1/\rho_i$ to achieve ρ_i-zCDP.

Theorem 6. *(1) The marginal M has sensitivity $\Delta_{\mathsf{M}} = 1$; (2) Publishing marginal M with noise $\mathcal{N}(0, 1/2\rho\mathbf{I})$ satisfies ρ-zCDP.*

The proof of Theorem 6 is deferred to Appendix A. To make ψ_i and ϕ_i comparable, we use the expected ℓ_1 error of the Gaussian noise on marginal i. That is, if the marginal size is c_i, after adding Gaussian noise with scale σ_i, we expect to see the ℓ_1 error of $c_i\sqrt{\frac{2}{\pi}}\sigma_i$. Thus, with privacy budget ρ_i, $\psi_i = c_i\sqrt{\frac{1}{\pi\rho_i}}$. The optimization problem is transformed to:

$$\begin{aligned}\text{minimize } & \sum_{i=1}^{m}\left[c_i\sqrt{\frac{1}{\pi\rho_i}}x_i + \phi_i(1-x_i)\right]\\ \text{subject to } & x_i \in \{0,1\}\\ & \sum x_i\rho_i = \rho\end{aligned}$$

Optimal Privacy Budget Allocation. We first assume the pairs are selected (*i.e.*, variables of x_i are determined), and we want to allocate different privacy budget to different marginals to minimize the overall noise error. In this case, the optimization problem can be rewritten as:

$$\begin{aligned}\text{minimize } & \sum_{i:x_i=1} c_i\sqrt{\frac{1}{\rho_i}}\\ \text{subject to } & \sum_{i:x_i=1}\rho_i = \rho\end{aligned}$$

For this problem, we can construct the Lagrangian function $\mathcal{L} = \sum_i \frac{c_i}{\sqrt{\rho_i}} + \mu\cdot(\sum_i\rho_i - \rho)$. By taking partial derivative of $\mathcal{L}$ for each of ρ_i, we have $\rho_i = \left(\frac{2\mu}{c_i}\right)^{-2/3}$. The value of μ can be solved by equation $\sum_i\rho_i = \rho$. As a result, $\mu = \frac{1}{2}\cdot\left(\frac{\rho}{\sum_i c_i^{2/3}}\right)^{-3/2}$, and we have

$$\rho_i = \frac{c_i^{2/3}}{\sum_j c_j^{2/3}}\cdot\rho \tag{1}$$

That is, allocating privacy budget proportional to the $\frac{2}{3}$ power of the number of cells achieves the minimum overall noise error.

A Greedy Algorithm to Select Pairs. We propose a greedy algorithm to select pairs of attributes, as shown in Algorithm 1. Given the InDif scores of all pairs of attributes $\langle\phi_i\rangle$, size of all marginals $\langle c_i\rangle$, and the total privacy budget ρ, the goal is to determine x_i for each $i \in \{1,\ldots,m\}$, or equivalently, output a set of pairs $X = \{i : x_i = 1\}$ that minimize the overall error. We handle this problem by iteratively including marginals that give the maximal utility improvement. In particular, in each iteration t, we select one marginal that brings the maximum improvement to the overall error. More specifically, we consider each marginal i that is not yet included in X (*i.e.*, $i \in \bar{X}$, where $\bar{X} = \{1,\ldots,m\}\setminus X$): In Line 4, we allocate the optimal privacy budget ρ_i according to Equation 1. We then calculate the error in Line 5, and select one with maximum utility improvement (in Line 6). After the marginal is selected, we then include it in X. The algorithm terminates when the overall error no longer improves. The algorithm is guaranteed to terminate since the noise error would gradually increase when more marginals are selected. When the noise error is larger than any of the remaining dependency error, the algorithm terminates.

Combine Marginals. Till now, we assume two-way

Algorithm 1: Marginal Selection Algorithm

```
Input: Number of pairs $m$, privacy budget $\rho$, dependency error $\langle\phi_i\rangle$,
       marginal size $\langle c_i\rangle$;
Output: Selected marginal set $X$;
1  $X \leftarrow \varnothing$; $t \leftarrow 0$; $E_0 \leftarrow \sum_{i\in\bar{X}}\phi_i$;
2  while True do
3      foreach marginal $i \in \bar{X}$ do
4          Allocate $\rho$ to marginals $j \in X\cup\{i\}$;
5          $E_t(i) = \sum_{j\in X\cup\{i\}} c_j\sqrt{\frac{1}{\pi\rho_j}} + \sum_{j\in\bar{X}\setminus\{i\}}\phi_j$;
6      $\ell \leftarrow \arg\min_{i\in\bar{X}} E_t(i)$;
7      $E_t \leftarrow E_t(\ell)$;
8      if $E_t \geq E_{t-1}$ then
9          Break
10     $X \leftarrow X\cup\{l\}$;
11     $t \leftarrow t+1$;
```

Algorithm 2: Marginal Combine Algorithm

```
Input: Selected pairwise marginals $X$, threshold $\gamma$
Output: Combined marginals $\mathcal{X}$
1  Convert $X$ to a set of pairs of attributes;
2  Construct graph $\mathcal{G}$ from the pairs;
3  $S \leftarrow \varnothing$; $\mathcal{X} \leftarrow \varnothing$
4  foreach clique size $s$ from $m$ to 3 do
5      $C_s \leftarrow$ cliques of size $s$ in $\mathcal{G}$
6      foreach clique $c \in C_s$ do
7          if $|c\cap S| \leq 2$ and domain size of $c \leq \gamma$ then
8              Append $c$ to $\mathcal{X}$
9              Append the attributes of $c$ to $S$
```

marginals are used. When some marginals contain only a small number of possibilities (*e.g.*, when some attributes are binary), extending to multi-way marginals can help capture more information. In particular, given X, which contains indices of the marginals selected from Algorithm 1, we first convert each index to its corresponding pair of attributes; we then build a graph $\mathcal{G}$ where each node represents an attribute and each edge corresponds to a pair. We then find all the cliques of size greater than 2 in the graph. If a clique is not very big (smaller than a threshold $\gamma = 5000$), and does not overlap much with existing selected attributes (with more than 2 attributes in common), we merge the 2-way marginals contained in the clique into a multi-way marginal.

Algorithm 2 gives the pseudocode of our proposed marginal combining technique. We first identify all possible cliques in graph $\mathcal{G}$ and sort them in decending order by their attribute size. Then, we examine each clique c to determine whether to combine it. If the clique has a small domain size (smaller than a threshold γ) and does not contain more than 2 attributes that is already in the selected attributes set S, we include this clique and remove all 2-way marginals within it.

4.3 Post Processing

The purpose of post processing is to ensure the noisy marginals are consistent. By handling such inconsistencies, we avoid impossible cases and ensure there exists a solution (*i.e.*, a synthetic dataset) that satisfies all the noisy marginals. For the case when multiple marginals contain the same set of attributes, and their estimations on the shared attributes do not agree, we use the weighted average method [16, 44]. Note that [16, 44] both assume the privacy budget is evenly distributed. We extend it to the uneven case.

Consistency under Uneven Privacy Budget Allocation. When different marginals have some attributes in common, those attributes are actually estimated multiple times. Utility will increase if these estimates are utilized together. For example, when some marginals are estimated twice, the mean of the estimates is actually more accurate than each of them. More formally, assume a set of attributes A is shared by s marginals $\mathsf{M}_1, \mathsf{M}_2, \ldots, \mathsf{M}_s$, where $A = \mathsf{M}_1 \cap \ldots \cap \mathsf{M}_s$. We can obtain s estimates of A by summing from cells in each of the marginals.

In [44], the authors proposed an optimal method to determine the distribution of the weights when privacy budget is evenly distributed among marginals. The main idea is to take the weighted average of estimates from all marginals in order to minimize the variance of marginals on A. We adopt the weighted average technique, and extend it to hand the case where privacy budget is unevenly allocated. In particular, we allocate a weight w_i for each marginal i. The variance of the weighted average can be represented by $\sum_i w_i^2 \cdot \frac{g_i}{\rho_i}$, where ρ_i is the privacy budget and g_i is the number of cells that contribute to one cell of the marginal on A. Here the Gaussian variance is $1/\rho_i$. By summing up g_i cells, and multiplying the result by w_i, we have the overall variance $w_i^2 \frac{g_i}{\rho_i}$. The weights should add up to 1. More formally, we have the following optimization problem:

$$\begin{aligned} \text{minimize} \sum_i w_i^2 \cdot \frac{g_i}{\rho_i} \\ \text{subject to} \sum_i w_i = 1 \end{aligned}$$

By constructing the Lagrangian function and following the same derivative procedure as we did for obtaining optimal ρ_i (Equation (1)), we have $w_i = \frac{\rho_i/g_i}{\sum_i \rho_i/g_i}$ is the optimal strategy.

Overall Consistency. In addition to the inconsistency among marginals, some noisy marginals may contain invalid distributions (*i.e.*, some probability estimations are negative, and the sum does not equal to 1). Given the invalid distribution, it is known that projecting it to a valid one with minimal ℓ_2 distance achieves the maximal likelihood estimation. This is discovered in different settings (*e.g.*, [10, 35, 51]); and there exists efficient algorithm for this projection.

The challenge emerges when we need to handle the two inconsistencies simultaneously, one operation invalidate the consistency established in another one. We iterate the two operations multiple times to ensure both consistency constraints are satisfied.

5 Synthetic Data Generation

Given a set of noisy marginals, the data synthesis step generates a new dataset D_s so that its distribution is consistent with the noisy marginals. Existing methods [41, 53] put these marginals into a graphical model, and use the sampling algorithm to generate the synthetic dataset. As each record is sampled using the marginals, the synthetic dataset distribution is naturally consistent with the distribution.

The drawback of this approach is that when the graph is dense, existing algorithms do not work. To overcome this issue, we use an alternative approach. Instead of sampling the dataset using the marginals, we initialize a random dataset and update its records to make it consistent with the marginals.

5.1 Strawman Method: Min-Cost Flow (MCF)

Given the randomly initiated dataset D_s, for each noisy marginal, we update D_s to make it consistent with the marginal. A marginal specified by a set of attributes is a frequency distribution table for each possible combination of values for the attributes. The update procedure can be modeled as a graph flow problem. In particular, given a marginal, a bipartite graph is constructed. Its left side represents the current distribution on D_s; and the right side is for the target distribution specified by the marginal. Each node corresponds to one cell in the marginal and is associated with a number. Figure 2 demonstrates an example of this flow graph. Now in order to change D_s to make it consistent with the marginal, we change records in D_s.

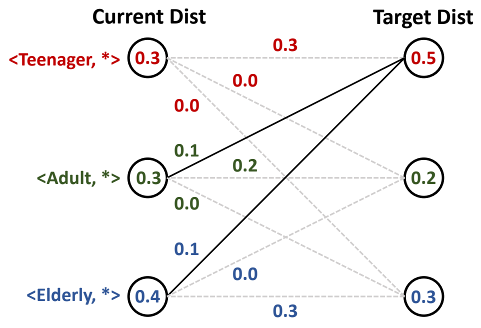


Figure 2: Running example of MCF. The left nodes represent current distribution from D_s; and the right nodes give the target distribution specified by the noisy marginal. The min-cost flow is to move 0.1 from adult to teenager, and 0.1 from elderly to teenager. To change the distribution, we find matching records from D_s and change their corresponding attributes.

The MCF method enforces a min-cost flow in the graph and updates D_s by changing the values of the records on the flow. For example, in Figure 2, there are two changes to D_s. First, one third of the adults needs to be changed to teenagers.

	Income	Gender	Age
v_1	high	male	teenager
v_2	high	male	adult
v_3	high	male	adult
v_4	high	male	teenager
v_5	high	female	elderly

(a) Dataset before updating.

v	$\mathsf{S}_{\{\text{I,G}\}}(v)$	$\mathsf{T}_{\{\text{I,G}\}}(v)$
⟨low, male,∗⟩	0.0	0.0
⟨low, female,∗⟩	0.0	0.0
⟨high, male,∗⟩	0.8	0.2
⟨high, female,∗⟩	0.2	0.8

(b) Marginal table for {Income, Gender}, where red and blue stands for over-counted and under-counted cells, respectively.

	Income	Gender	Age
v_1	high	male	teenager
v_2	high	male	adult
v_3	high	female	elderly
v_4	high	female	teenager
v_5	high	female	elderly

(c) Dataset after updating.

Figure 3: Example of the synthesized dataset before and after updating procedure. In (a), blue stands for the records to be added, and brown stands for the records to be changed. In (c), v_4 only changes income and gender attributes, while v_3 changes the whole record which is duplicated from v_5. Notice that in this example, we have $\alpha = 2.0, \beta = 0.5$ and the marginal distribution in (c) do not completely match $\mathsf{T}_{\{\text{I,G}\}}(v)$ of $[0.0, 0.0, 0.2, 0.8]$; instead, it becomes $[0.0, 0.0, 0.4, 0.6]$.

Note that we change only the related attribute and keep the other attributes the same. Second, one fourth of the elderly are changed to teenager. We iterate over all the noisy marginals and repeat the process multiple times until the amount of changes is small. The intuition of using min-cost flow is that, the update operations make the minimal changes to D_s, and by changing the dataset in this minimal way, the consistency already established in D_s (with previous marginals) can be maintained. The min-cost flow can be solved by the off-the-shelf linear programming solver, e.g., [7].

When all marginals are examined, we randomly shuffle the whole dataset D_s. Since the modifying procedure would invalidate the consistency established from previous marginals, MCF needs to iterate multiple times to ensure that D_s is almost consistent with all marginals.

5.2 Gradually Update Method (GUM)

Empirically, we find that the convergence performance of MCF is not good (we will demonstrate it via experiment in Section 6). We believe that this is because MCF always changes D_s to make it completely consistent with the current marginal in each step. Doing this reduces the error of the target marginal close to zero, but increases the errors for other marginals to a large value.

To handle this issue, we borrow the idea of multiplicative update [8] and propose a new approach that Gradually Update D_s based on the Marginals; and we call it GUM. GUM also adopts the flow graph introduced by MCF, but differs from MCF in two ways: First, GUM does not make D_s fully consistent with the given marginal in each step. Instead, it changes D_s in a multiplicative way, so that if the original frequency in a cell is large, then the change to it will be more. In particular, we set a parameter α, so that for cells that have values are lower than expected (according to the target marginal), we add at most α times of records, i.e., $\min\{n^t - n^s, \alpha n^s\}$ [1], where n^t is the number in the marginal and n^s is the number from D_s. On the other hand, for cells with values higher than expected, we will reduce $\min\{n^s - n^t, \beta n^s\}$ records that satisfy it. As the total number of record is fixed, given α, β can be calculated.

Figure 3 gives a running example. Before updating, we have 4 out of 5 records have the combination $\langle high, male\rangle$, and 1 record has $\langle high, female\rangle$. To get closer to the target marginal of 0.2 and 0.8 for these two cells, we want to change 2 of the $\langle high, male\rangle$ records to be $\langle high, female\rangle$. In this example, we have $\alpha = 2.0, \beta = 0.5$ [2] and do not completely match the target marginal of 0.2 and 0.8. To this end, one approach is to simply change the Gender attribute value from male to female in these two records as in MCF. We call this a **Replace** operation. Replacing will affect the joint distribution of other marginals, such as $\{Gender, Age\}$. An alternative is to discard an existing $\langle high, male\rangle$ record, and **Duplicate** an existing $\langle high, female\rangle$ record (such as v_5 in the example). Duplicating an existing record help preserve joint distributions between the changed attributes and attributes not in the marginal. However, Duplication will not introduce new records that can better reflect the overall joint distribution. In particular, if there is no record that currently has the combination $\langle high, female, elderly\rangle$, duplication cannot be used.

Therefore, we need to use a combination of Replacement and Duplication (which is the case in Figure 3). Furthermore, once the synthesized dataset is getting close to the distribution, we would prefer Duplication to Replacement, since at that time there should be enough records to reflect the distribution and Replacement disrupts the joint distribution between attributes in a marginal and those not in it. We empirically compare different record updating strategies and validate that introducing the Duplication operation can effectively improve the convergence performance. Due to space limitation, we refer the readers to Appendix I in out technical report [55] for the experimental results.

[1] Notice that α could be greater than 1 since $n^s < n^t$. In the experiments, we always set α to be less than 1 to achieve better convergence performance.

[2] We have $\alpha = \frac{n^t - n^s}{n^s}$ for under-counted cells and $\beta = \frac{n^s - n^t}{n^s}$ for over-counted cells. The number of records for under-counted cell ⟨high, female,∗⟩ increase from 1 to 3; thus $\alpha = \frac{3-1}{1} = 2$. The number of records for over-counted cell ⟨high, male,∗⟩ decrease from 4 to 2; thus $\beta = \frac{4-2}{4} = 0.5$.

5.3 Improving the Convergence

Given the general data synthesize method, we have several optimizations to improve its utility and performance. First, to bootstrap the synthesizing procedure, we require each attribute of D_s follows the 1-way noisy marginals when we initialize a random dataset D_s.

Gradually Decreasing α. The update rate α should be smaller with the iterations to make the result converge. From the machine learning perspective, gradually decreasing α can effectively improve the convergence performance. There are some common practices [1] of setting α.

- Step decay: $\alpha = \alpha_0 \cdot k^{\lfloor \frac{t}{s} \rfloor}$, where α_0 is the initial value, t is the iteration number, k is the decay rate, and s is the step size (decrease α every s iterations). The main idea is to reduce α by some factor every few iterations.
- Exponential decay: $\alpha = \alpha_0 \cdot e^{-kt}$, where k is a hyperparameter. This exponentially decrease α in each iteration.
- Linear decay: $\alpha = \frac{\alpha_0}{1+kt}$.
- Square root decay: $\alpha = \frac{\alpha_0}{\sqrt{1+kt}}$.

We empirically evaluate the performance of different decay algorithms (refer to Appendix J in our technical report [55]) and find that step decay is preferable in all settings. The step decay algorithm is also widely used to update the step size in the training of deep neural networks [33].

Attribute Appending. The selected marginals $\mathcal{X}$ output by Algorithm 2 can be represented by a graph $\mathcal{G}$. We notice that some nodes have degree 1, which means the corresponding attributes are included in exactly one marginal. For these attributes, it is not necessary to involve them in the updating procedure. Instead, we could append them to the synthetic dataset D_s after other attributes are synthesized. In particular, we identify nodes from $\mathcal{G}$ with degree 1. We then remove marginals associated with these nodes from $\mathcal{X}$. The rest of the noisy marginals are feed into GUM to generate the synthetic data but with some attributes missing. For each of these missed attributes, we sample a smaller dataset D_s' with only one attribute, and we concatenate D_s' to D_s using the marginal associated with this attribute if there is such a marginal; otherwise, we can just shuffle D_s' and concatenate it to D_s. Note that this is a one time operation after GUM is done. No synthesizing operation is needed after this step.

Separate and Join. We observe that, when the privacy budget is low, the number of selected marginals is relatively small, and the dependency graph is in the form of several disjoint subgraphs. In this case, we can apply GUM to each subgraph and then join the corresponding attributes. The benefit of Separate and Join technique is that, the convergence performance of marginals in one subgraph would not be affected by marginals in other subgraph, which would improve the overall convergence performance.

Filter and Combine Low-count Values. If some attributes have many possible values while most of them have low counts or do not appear in the dataset. Directly using these attributes to obtain pairwise marginals may introduce too much noise. To address this issue, we propose to filter and combine the low-count values. The idea is to spend a portion of privacy budget to obtain the noisy one-way marginals. After that, we keep the values that have count above a threshold θ. For the values that are below θ, we add them up, if the total is below θ, we assign 0 to all these values. If their total is above θ, then we create a new value to represent all values that have low counts. After synthesizing the dataset, this new value is replaced by the values it represents using the noisy one-way marginal. The threshold is set as $\theta = 3\sigma$, where σ is the standard deviation for Gaussian noises added to the one-way marginals.

5.4 Putting Things Together: PrivSyn

Algorithm 3 illustrates the overall workflow of PrivSyn. We split the total privacy budget into three parts. The first part is used for publishing all 1-way marginals, intending to filter and combine the values with low count or do not exist. The second part is used to differentially privately select marginals. The marginal selection method DenseMarg consists of two components, *i.e.*, 2-way marginal selection (Algorithm 1) and marginal combine (Algorithm 2). The third part is used to obtain the noisy combined marginals. After obtaining the noisy combined marginals, we can use them to construct synthetic dataset D_s without consuming privacy budget, since this is a post processing procedure.

Algorithm 3: PrivSyn

```
  Input: Private dataset $D_o$, privacy budget $\rho$;
  Output: Synthetic dataset $D_s$;
1 Publish 1-way marginals using GM with $\rho_1 = 0.1\rho$;
2 Filter values with estimates smaller than $3\sigma$;
3 Select 2-way marginals with Algorithm 1 and $\rho_2 = 0.1\rho$;
4 Combine marginals using Algorithm 2;
5 Publish combined marginals using GM with $\rho_3 = 0.8\rho$;
6 Make noisy marginals consistent;
7 Construct $D_s$ using GUM;
```

6 Evaluation

In this section, we first conduct a high-level end-to-end experiment to illustrate the effectiveness of PrivSyn. Then, we evaluate the effectiveness of each step of PrivSyn by fixing other steps. As a highlight, our method consistently achieves better performance than the state-of-the-art in all steps.

6.1 Experimental Setup

Datasets. We run experiments on the following four datasets.

- **UCI Adult [9].** This is a widely used dataset for classification from the UCI machine learning repository.
- **US Accident [42].** This is a countrywide traffic accident dataset, which covers 49 states of the United States.

- **Loan [31].** This dataset contains loan data in lending club issued from 2007 to 2015.
- **Colorado [43].** This is the census dataset of Colorado State in 1940. This dataset is used in the final round of the NIST challenge [43].

The detailed information about the datasets are listed in Table 2, where the label column stands for the label used in the classification task.

Dataset	Records	Attributes	Domain	Label
Adult	48,000	15	$6 \cdot 10^{17}$	salary
US Accident	600,000	30	$3 \cdot 10^{39}$	Severity
Loan	600,000	81	$4 \cdot 10^{136}$	home_ownership
Colorado	662,000	97	$5 \cdot 10^{162}$	INCNONWG

Table 2: Summary of datasets used in our experiments.

Tasks and Metrics. We evaluate the statistical performance of the synthesized datasets on three data analysis tasks. For each data analysis task, we adopt its commonly used metric to measure the performance.

- **Marginal Release.** We compute all the 2-way marginals and use the average ℓ_1 error to measure the performance.
- **Range Query.** We randomly sample 1000 range queries, each contains 3 attributes. We use the average ℓ_1 error to measure the performance. In particular, we calculate $\frac{1}{|Q|}\sum_{q_i \in Q}|c_i - \hat{c}_i|$, where Q is the set of randomly sampled queries, c_i and $\hat{c}_i$ are the ratio of records that fall in the range of query q_i in the original dataset and synthesized dataset, respectively.
- **Classification.** We use the synthesized dataset to train an SVM classification model, and use misclassification rate to measure the performance.

Competitors. We compare each component of PrivSyn with a series of other methods, respectively.

- **Marginal Selection Methods.** We compare our proposed DenseMarg method (Algorithm 1) with PrivBayes. The computational complexity of dependency in original PrivBayes method is too high. Thus, we replace the dependency calculation part of PrivBayes by our proposed InDif metric, which we call PrivBayes(InDif). For Colorado dataset, the PGM team open sourced a set of manually selected marginals in the NIST challenge [43], which serves as an alternative competitor.
- **Noise Addition Methods.** We compare our proposed Weighted Gaussian method with Equal Laplace and Equal Gaussian methods. Both Gaussian methods use zCDP to compose, and the Laplace mechanism use the naive composition, *i.e.*, evenly allocate ε for each marginal.
- **Data Synthesis Methods.** We compare PrivSyn with PrivBayes and PGM, which use the selected marginals to estimate a graphical model, and sample synthetic records from it. Note that we have two versions of synthesis methods for PrivSyn, *i.e.*, MCF and GUM.

We also compare with a few other algorithms that do not follow the framework in Section 3.

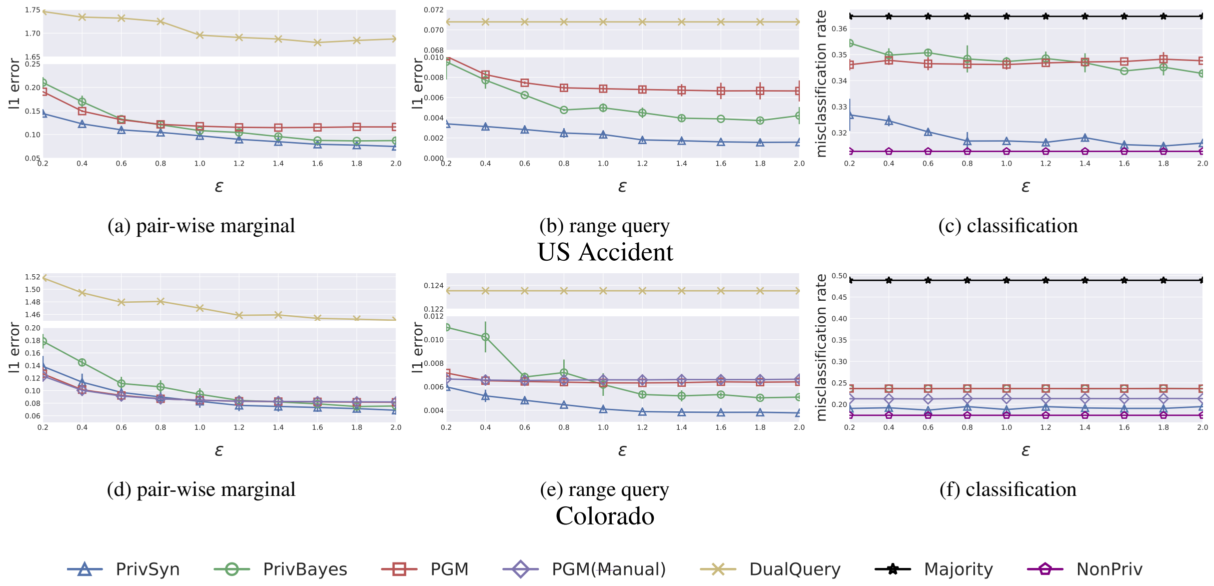
- **DualQuery.** It generates records in a game theoretical manner. The main idea is to maintain a distribution over a workload of queries. One first samples a set of queries from the workload each time, and then generates a record that minimize the error of these queries. We refer the readers to Section 7 for detailed discussion.
- For the classification task, we have another two competitors, *i.e.*, Majority and NonPriv. Majority represents the naive method that blindly predicts the label by the majority label. Methods that perform worse than Majority means that the published dataset doesn't help the classification task, since the majority label can be outputted correctly even under very low privacy budget. NonPriv represents the method without enforcing differential privacy, it is the best case to aim for. For NonPriv, we split the original dataset into two disjoint parts, one for training and another for testing.

Experimental Setting. For PrivBayes, PGM and PrivSyn methods, we set the number of synthesized records the same as that of the original dataset. Notice that we adopt unbounded differential privacy [32] in this paper, we cannot directly access the actual number of records in the original dataset. Thus, we instead use the total count of marginals to approximate it. For DualQuery method, the number of synthesized records is inherently determined by the privacy budget, the step size and the sample size [28]. We use the same hyper-parameter settings as [28], *i.e.*, the step size is 2.0 and sample size is 1000. We illustrate the impact of the number of synthesized records on PrivSyn in Appendix K of our technical report [55]. By default, we set $\delta = \frac{1}{n^2}$ for all methods, where n is the number of records in original dataset.

All algorithms are implemented in Python 3.7 and all the experiments are conducted on a server with Intel Xeon E7-8867 v3 @ 2.50GHz and 1.5TB memory. We repeat each experiment 5 times and report the mean and standard deviation. Due to space limitation, we put the experimental results of US Accident and Colorado in the main context, and refer the readers to the results of Adult and Loan datasets to Appendix L in our technical report [55].

6.2 End-to-end Comparison

Setup. For fair comparison, we use the optimal components and hyper-parameters for all methods. Concretely, we use PrivBayes(InDif) to select marginals for PrivBayes and PGM, since they can only handle sparse marginals. Both PrivSyn and DualQuery can handle dense marginals; thus we use DenseMarg to select marginals for them. For noise addition, we use Weighted Gaussian for PrivBayes, PGM and PrivSyn. DualQuery uses a game theoretical manner to generate synthetic datasets; thus it does not need the noise addition



(a) pair-wise marginal (b) range query (c) classification

US Accident

(d) pair-wise marginal (e) range query (f) classification

Colorado

Figure 4: End-to-end Comparison of different dataset generation methods. PrivSyn is our proposed method.

step. For PrivBayes, PGM and DualQuery, we use the open-sourced code [2] by the author of PGM to run the experiments.

Results. Figure 4 illustrates the performance of different methods. We do not show the classification performance of DualQuery since the misclassification rate is larger than Majority and the variance is large. The experimental results show that PrivSyn consistently outperforms other methods for all datasets and all data analysis tasks.

For the pair-wise marginal task, the performance of PGM and PrivBayes is quite close to PrivSyn, meaning these two methods can effectively capture low-dimensional correlation. However, the performance of range query task and classification task are much worse than PrivSyn, since range query and classification tasks require higher dimensional correlation. PrivSyn can effectively preserve both low-dimensional and high-dimensional correlation.

The performance of DualQuery is significantly worse than other methods. The reason is that generating each record consumes a portion of privacy budget, which limits the number of records generated by DualQuery. In our experiments, the number of generated records by DualQuery is less than 300 in all settings. When the privacy budget is low, *e.g.*, $\varepsilon = 0.2$, the number of generated records is less than 50. Insufficient number of records would lead to bad performance for all three data analysis tasks.

6.3 Comparison of Marginal Selection Methods

Setup. We use Weighted Gaussian method for noise addition, and use GUM for data synthesis. For each marginal selection method, we compare their performance in both private and non-private settings. In the non-private setting, the marginal selection step do not consume privacy budget. This can serve as a baseline to illustrate the robustness of different marginal selection methods.

Results. Figure 5 illustrates the performance of different marginal selection methods. For all datasets and all data analysis tasks, our proposed DenseMarg method consistently outperforms PrivBayes(InDif). In the range query task, DenseMarg reduces the ℓ_1 error by about 50%, which is much significant than that in pair-wise marginal release task. This is because our range queries contain 3 attributes, which requires higher dimensional correlation information than pair-wise marginal (contain 2 attributes). DenseMarg preserves more higher dimensional correlation by selecting more marginals than PrivBayes(InDif).

In all settings, the performance of DenseMarg in private setting and non-private setting are very close. The reason is that DenseMarg tends to select the set of marginals with high InDif, and adding moderate level of noise is unlikely to significantly change this set of marginals. In our experiments, the overlapping ratio of the selected marginals between private setting and non-private setting is larger than 85% in most cases. This indicates that DenseMarg is very robust to noise.

6.4 Comparison of Noise Addition Methods

Setup. We compare our proposed Weighted Gaussian method with Equal Laplace and Equal Gaussian methods. Both Gaussian methods use zCDP for composition. The Laplace mechanism uses the naive composition, *i.e.*, evenly allocate ε for all marginals. All methods use DenseMarg for marginal selection and GUM for data synthesis.

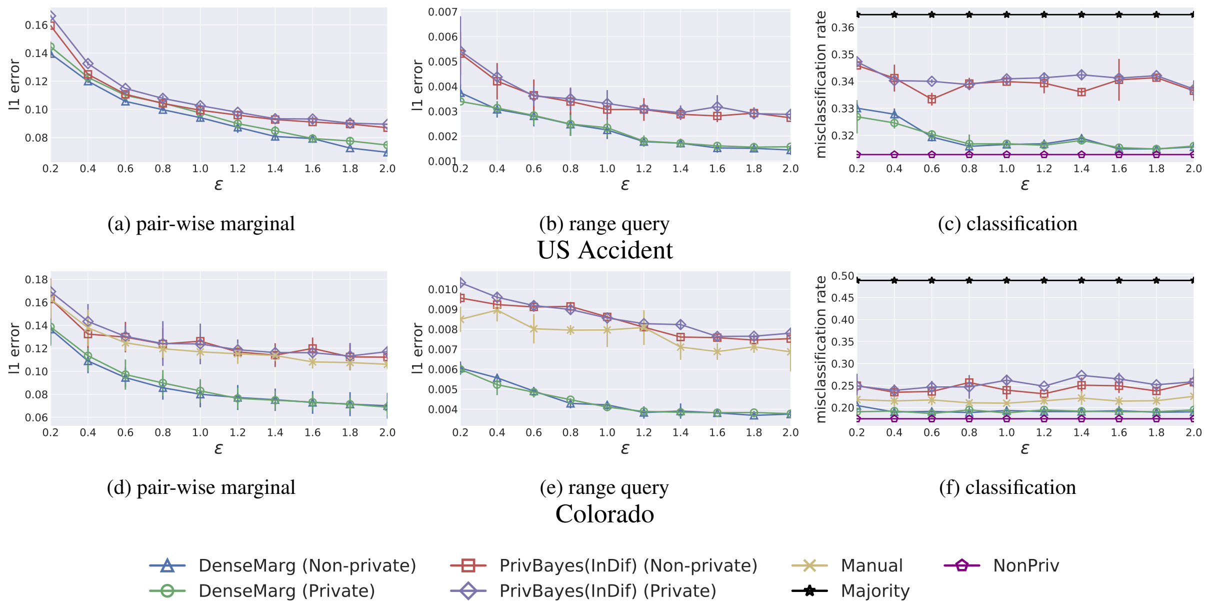


Figure 5: Comparison of different marginal selection methods. DenseMarg is our proposed method. Non-private in the parenthese indicates that the marginal selection step do not consume privacy budget.

Results. Figure 6 demonstrates the performance of different noise addition methods. For all datasets and all data analysis tasks, our proposed Weighted Gaussian method consistently outperforms the other two methods. The advantage of Weighted Gaussian increases when ε is larger.

In our experiment, both Weighted Gaussian and Equal Gaussian methods use zCDP to calculate the noise variance to each marginal, the main difference is that Weighted Gaussian allocates privacy budget according to the number of cells, while Equal Gaussian evenly allocate privacy budget to all marginals. The experimental results validate our analysis in Section 4.2 that Weighted Gaussian is the optimal privacy budget allocation strategy.

6.5 Comparison of Synthesis Methods

To better understand the performance of different synthesis methods, we select marginals in a non-private setting and purely compare the performance of different synthesis methods. This is different from the end-to-end evaluation in Section 6.2 that makes all steps private. Other settings are the same as Section 6.2. We do not compare with DualQuery in this experiment since Section 6.2 has illustrated that its performance is much worse than other methods.

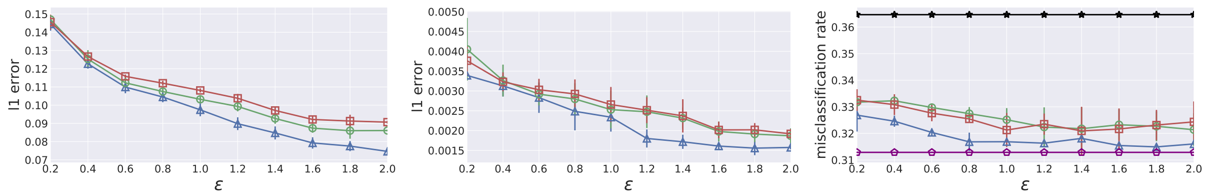
Results. Figure 7 shows the performance of different data synthesis methods. Both MCF and GUM exploit dense marginals selected by DenseMarg, while the performance of MCF is even worse than the PGM method and the PrivBayes method that using spare marginals. The reason is that, in each iteration, MCF enforces the synthetic dataset D_s to fully match the marginal. This would severely destroy the correlation established by other marginals. While GUM preserves the correlation of other marginals by gradually updating marginals in each iteration and using duplication technique.

Comparing Figure 4 and Figure 7, we observe that the experimental results in the private and non-private settings are similar, showing the robustness of PrivSyn. This is consistent with the results in Section 6.3.

7 Related Work

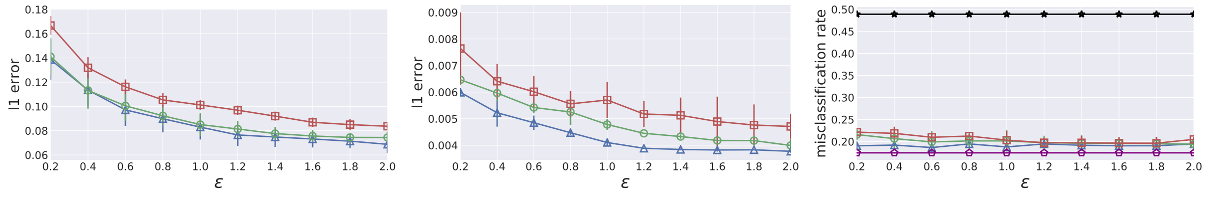
Differential privacy (DP) has been the *de facto* notion for protecting privacy. Many DP algorithms have been proposed (see [25,48] for theoretical treatments and [37] in a more practical perspective). Most of the algorithms are proposed for specific tasks. In this paper, we study the general task of generating a synthetic dataset with DP. Compared to the ad-hoc methods, this approach may not achieve the optimal utility for the specific task. But this approach is general in that given the synthetic dataset, any task can be performed, and there is no need to modify existing non-private algorithms. There are a number of previous studies focus on generating synthetic dataset in a differentially private manner. We classify them into three categoreis: graphical model based methods, game based methods and deep generative model based methods. There are also some theoretical studies that discuss the hardness of differentially private data synthesis.

Graphical Model Based Methods. The main idea is to estimate a graphical model that approximates the distribu-



(a) pair-wise marginal (b) range query (c) classification

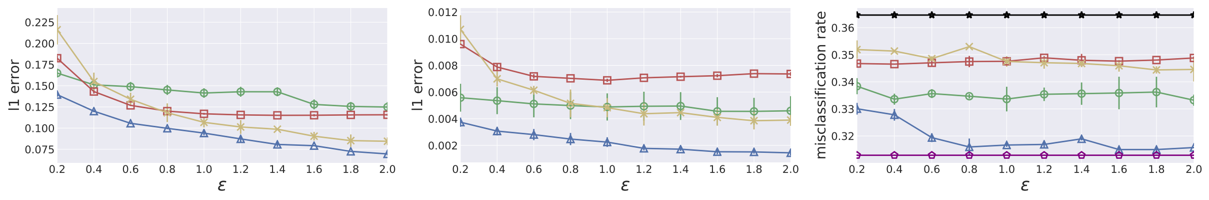
US Accident



(d) pair-wise marginal (e) range query (f) classification

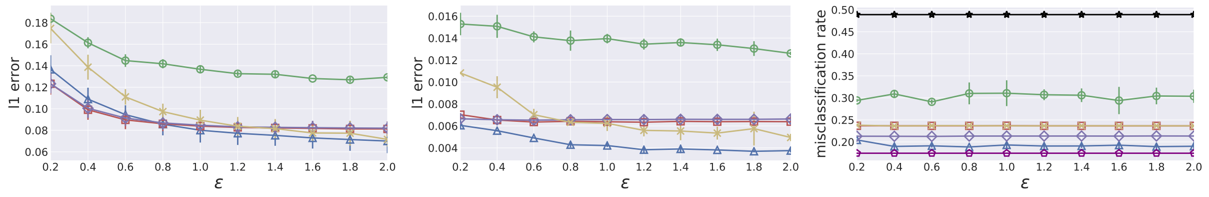
Colorado

Figure 6: Comparison of different noise addition methods. Weighted Gaussian is our proposed method.



(a) pair-wise marginal (b) range query (c) classification

US Accident



(d) pair-wise marginal (e) range query (f) classification

Colorado

Figure 7: Comparison of different synthesis methods. GUM is our proposed method.

tion of the original dataset in a differentially private way. PrivBayes [53] and BSG (the initials of the authors' last names) [12] approximate the data distribution using Bayesian Network. These methods, however, need to call Exponential Mechanism [53] or Laplace Mechanism [12] many times, making the network structure inaccurate when the privacy budget is limited; and the overall utility is sensitive to the quality of the initial selected node.

PGM [41] and JTree [15] utilize Markov Random Field to approximate the data distribution. PGM takes as input a set of predefined low-dimensional marginals, and estimates a Markov Random Field that best matches these marginals. JTree first estimates a dependency graph by setting a threshold to the mutual information of pairwise attributes, and then obtains the Markov Random Field by transforming the dependency graph into a junction tree. PGM do not provide marginal selection method in the paper [41]. JTree proposes to use SVT to select marginals; however, Lyu *et al.* [39] point out that JTree utilizes SVT in a problematic way. The main limitation of graphical model based methods is that they cannot handle dense marginals that capture more correlation information.

Game Based Methods. There are works that formulate the dataset synthesis problem as a zero-sum game [28, 29, 49]. Assume there are two players, data player and query player. MWEM [29] method solves the game by having the data player use a no-regret learning algorithm, and the query player repeatedly best responds. Dual Query [28] switches the role of the two players. Concretely, the data player in MWEM maintains a distribution over the whole data domain, and query player repeatedly use exponential mechanism to select a query that have the worse performance from a workload of queries to update data player's distribution. The main limitation of MWEM is that when the dataset domain is large (from $3 \cdot 10^{39}$ to $5 \cdot 10^{162}$ in our experiments), maintaining the full distribution is infeasible. Thus, we do not compare with MWEM in our experiments.

In contrast, the query player in Dual Query maintains a distribution over all queries. The query player each time samples a set of queries from the workload, and the data player generates a record that minimizes the error of these queries. The shortcoming is that generating each record would consume a portion of privacy budget; thus one cannot generate sufficient records as discussed in Section 6. Moreover, both methods require a workload of queries in advance, and the generated dataset is guaranteed to be similar to the original dataset with respect to the query class. This makes MWEM and Dual Query incapable of handling arbitrary kinds of tasks with satisfied accuracy. The authors of [49] improve both MWEM and DualQuery by replacing their core components; however, this work follows the same framework with MWEM and QualQuery and do not address the main limitation of them.

Deep Generative Model Based Methods. Another approach is to train a deep generative model satisfying differential privacy, and use the deep generative model to generate a synthetic dataset. The most commonly used deep generative model is the Generative Adversarial Network (GAN), and there are multiple studies focus on training GAN in a differentially private way [4, 11, 27, 46, 54]. The main idea is to utilize the DP-SGD framework [3] to add noise in the optimization procedure (i.e., stochastic gradient descent). However, the preliminary application of GAN is to generate images. Thus, the objective of GAN is to generate data records that look authentic, instead of approximating the original distribution, applying the GAN model to the current problem cannot generate a synthetic dataset with enough variations. In the NIST challenge [43], there are two teams adapting the GAN-based method to synthesize high-dimensional data, while their scores are much lower than PGM and PrivBayes. Thus, we do not compare this line of methods in our experiments.

In addition to GAN, there are also studies based on Restricted Boltzmann Machine (RBM) [26] and Variational Auto-Encoder (VAE) [6]. These methods are not as effective as GAN.

Theoretical Results. There are a series of negative theoretical results concerning DP in the non-interactive setting [17, 18, 20, 22–24, 47]. These results have been interpreted "to mean that one cannot answer a linear, in the database size, number of queries with small noise while preserving privacy" and to motivate "an interactive approach to private data analysis where the number of queries is limited to be small – sub-linear in the size n of the dataset" [18].

We point out that, theoretical negative results notwithstanding, non-interactive publishing can serve an important role in private data publishing. The negative results essentially say that when the set of queries is sufficiently broad, one cannot guarantee that all of them are answered accurately. These results are all based on query sets that are broader than the natural set of queries in which one is interested. For example, suppose the dataset is one-dimensional where each value is an integer number in domain $[m] = \{1, 2 \ldots, m\}$. These results say that one cannot answer counting queries for arbitrary subsets of $[m]$ with error less than $\Theta(\sqrt{n})$, where n is the size of the dataset. However, range queries, which are likely to be what one is interested in, can be answered with less error. Moreover, these results are all asymptotic and do not rule out useful algorithms in practice. When one plugs in actual parameters, the numbers that come out often have no bearing on practice.

8 Discussion and Limitations

In this section, we discuss the application scope and limitations of PrivSyn.

Only Applicable to Tabular Data. PrivSyn focuses on the tabular data and cannot handle other types of data such as image or streaming data. Note that other existing methods (PrivBayes, PGM and DualQuery) also have this limitation. We defer the application of PrivSyn to image dataset and sequential dataset to future work.

Miss Some Higher Dimensional Correlation. PrivSyn only considers low-degree marginals that may not capture some high-dimensional correlation information. Notice that other marginal selection methods such as PrivBayes and BSG also use low-degree marginals to approximate the high-dimensional datasets and also have this limitation. To capture higher dimensional correlation, one possibility is to consider all triple-wise marginals or higher-dimensional marginals; however, doing this may introduce too much noise for each of the marginal, resulting in inaccurate selection. In practice, low-dimensional marginals are sufficient to capture enough correlation information, as illustrated on the four real-world datasets used in our experiments.

9 Conclusion

In this paper, we present PrivSyn for publishing a synthetic dataset under differential privacy. We identify the core steps in the process and summarize previous studies for each step. PrivSyn achieves the state-of-the-art by proposing novel methods for all steps. We extensively evaluate different methods on multiple real-world datasets to demonstrate the superiority of PrivSyn.

Acknowledgments

We thank the anonymous reviewers for their constructive feedback. This work is partially funded by NSFC under grant No. 61731004, U1911401, Alibaba-Zhejiang University Joint Research Institute of Frontier Technologies, the Helmholtz Association within the project "Trustworthy Federated Data Analytics" (TFDA) (funding number ZT-I-OO1 4), and United States NSF under grant No. 1931443.

References

[1] http://cs231n.github.io/neural-networks-3/#anneal.

[2] https://github.com/ryan112358/private-pgm.

[3] Martín Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318. ACM, 2016.

[4] Nazmiye Ceren Abay, Yan Zhou, Murat Kantarcioglu, Bhavani Thuraisingham, and Latanya Sweeney. Privacy preserving synthetic data release using deep learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 510–526. Springer, 2018.

[5] John M Abowd. The us census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2867–2867, 2018.

[6] Gergely Acs, Luca Melis, Claude Castelluccia, and Emiliano De Cristofaro. Differentially private mixture of generative neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 31(6):1109–1121, 2018.

[7] Ravindra K Ahuja, Thomas L Magnanti, and James B Orlin. Network flows. 1988.

[8] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

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

[10] Raef Bassily. Linear queries estimation with local differential privacy. In *AISTATS*, 2019.

[11] Brett K Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, Ran Lee, Sanjeev P Bhavnani, James Brian Byrd, and Casey S Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *Circulation: Cardiovascular Quality and Outcomes*, 12(7):e005122, 2019.

[12] Vincent Bindschaedler, Reza Shokri, and Carl A Gunter. Plausible deniability for privacy-preserving data synthesis. *Proceedings of the VLDB Endowment*, 10(5), 2017.

[13] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618, 2008.

[14] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[15] Rui Chen, Qian Xiao, Yu Zhang, and Jianliang Xu. Differentially private high-dimensional data publication via sampling-based inference. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 129–138. ACM, 2015.

[16] Bolin Ding, Marianne Winslett, Jiawei Han, and Zhenhui Li. Differentially private data cubes: optimizing noise sources and consistency. In *SIGMOD Conference*, pages 217–228, 2011.

[17] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.

[18] C Dwork, M Naor, O Reingold, G.N Rothblum, and S Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. *STOC*, pages 381–390, 2009.

[19] C Dwork, G Rothblum, and S Vadhan. Boosting and differential privacy. *Foundations of Computer Science (FOCS), 2010 51st Annual IEEE Symposium on*, pages 51 – 60, 2010.

[20] C Dwork and S Yekhanin. New efficient attacks on statistical disclosure control mechanisms. *Advances in Cryptology–CRYPTO 2008*, pages 469–480, 2008.

[21] Cynthia Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.

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

[23] Cynthia Dwork, Frank McSherry, and Kunal Talwar. The price of privacy and the limits of LP decoding. In *STOC*, pages 85–94, 2007.

[24] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N Rothblum, and Salil Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 381–390, 2009.

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

[26] Asja Fischer and Christian Igel. An introduction to restricted boltzmann machines. In *Iberoamerican congress on pattern recognition*, pages 14–36. Springer, 2012.

[27] Lorenzo Frigerio, Anderson Santana de Oliveira, Laurent Gomez, and Patrick Duverger. Differentially private generative adversarial networks for time series, continuous, and discrete open data. In *IFIP International Conference on ICT Systems Security and Privacy Protection*, pages 151–164. Springer, 2019.

[28] Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Zhiwei Steven Wu. Dual query: Practical private query release for high dimensional data. In *International Conference on Machine Learning*, pages 1170–1178, 2014.

[29] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems*, pages 2339–2347, 2012.

[30] Noah Johnson, Joseph P Near, and Dawn Song. Towards practical differential privacy for sql queries. *Proceedings of the VLDB Endowment*, 11(5):526–539, 2018.

[31] Kaggle. Kaggle lending club loan data. https://www.kaggle.com/wendykan/lending-club-loan-data.

[32] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *SIGMOD*, pages 193–204, 2011.

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[34] Jaewoo Lee and Christopher W Clifton. Top-k frequent itemsets via differentially private fp-trees. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 931–940, 2014.

[35] Jaewoo Lee, Yue Wang, and Daniel Kifer. Maximum likelihood post-processing for differential privacy under consistency constraints. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 635–644, 2015.

[36] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. Optimizing linear counting queries under differential privacy. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 123–134, 2010.

[37] Ninghui Li, Min Lyu, Dong Su, and Weining Yang. *Differential Privacy: From Theory to Practice*. Synthesis Lectures on Information Security, Privacy, and Trust. Morgan Claypool, 2016.

[38] Ninghui Li, Wahbeh Qardaji, Dong Su, and Jianneng Cao. Privbasis: Frequent itemset mining with differential privacy. *Proceedings of the VLDB Endowment*, 5(11):1340–1351, 2012.

[39] Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *arXiv preprint arXiv:1603.01699*, 2016.

[40] Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *PVLDB*, 10(6):637–648, 2017.

[41] Ryan Mckenna, Daniel Sheldon, and Gerome Miklau. Graphical-model based estimation and inference for differential privacy. In *International Conference on Machine Learning*, pages 4435–4444, 2019.

[42] Sobhan Moosavi, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, Radu Teodorescu, and Rajiv Ramnath. Accident risk prediction based on heterogeneous sparse data: New dataset and insights. In *Proceedings of ACM SIGSPATIAL'19*, pages 33–42.

[43] NIST. 2018 differential privacy synthetic data challenge. https://www.nist.gov/ctl/pscr/open-innovation-prize-challenges/past-prize-challenges/2018-differential-privacy-synthetic.

[44] Wahbeh Qardaji, Weining Yang, and Ninghui Li. Priview: practical differentially private release of marginal contingency tables. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1435–1446. ACM, 2014.

[45] Ryan Rogers, Subbu Subramaniam, Sean Peng, David Durfee, Seunghyun Lee, Santosh Kumar Kancha, Shraddha Sahay, and Parvez Ahammad. Linkedin's audience engagements api: A privacy preserving data analytics system at scale. *arXiv preprint arXiv:2002.05839*, 2020.

[46] Uthaipon Tantipongpipat, Chris Waites, Digvijay Boob, Amaresh Ankit Siva, and Rachel Cummings. Differentially private mixed-type data generation for unsupervised learning. *arXiv preprint arXiv:1912.03250*, 2019.

[47] Jonathan Ullman and Salil Vadhan. Pcps and the hardness of generating private synthetic data. In *Theory of Cryptography Conference*, pages 400–416. Springer, 2011.

[48] Salil Vadhan. The complexity of differential privacy. In *Tutorials on the Foundations of Cryptography*, pages 347–450. Springer, 2017.

[49] Giuseppe Vietri, Grace Tian, Mark Bun, Thomas Steinke, and Zhiwei Steven Wu. New oracle-efficient algorithms for private synthetic data release.

[50] Ning Wang, Xiaokui Xiao, Yin Yang, Zhenjie Zhang, Yu Gu, and Ge Yu. Privsuper: A superset-first approach to frequent itemset mining under differential privacy. In *Data Engineering (ICDE), 2017 IEEE 33rd International Conference on*, pages 809–820. IEEE, 2017.

[51] Tianhao Wang, Milan Lopuhaä-Zwakenberg, Zitao Li, Boris Skoric, and Ninghui Li. Locally differentially private frequency estimation with consistency. In *NDSS*, 2020.

[52] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke. Differential privacy via wavelet transforms. *IEEE Transactions on knowledge and data engineering*, 23(8):1200–1214, 2010.

[53] Jun Zhang, Graham Cormode, Cecilia M Procopiuc, Divesh Srivastava, and Xiaokui Xiao. Privbayes: Private data release via bayesian networks. *ACM Transactions on Database Systems (TODS)*, 42(4):25, 2017.

[54] Xinyang Zhang, Shouling Ji, and Ting Wang. Differentially private releasing via deep generative model. *arXiv preprint arXiv:1801.01594*, 2018.

[55] Zhikun Zhang, Tianhao Wang, Ninghui Li, Jean Honorio, Michael Backes, Shibo He, Jiming Chen, and Yang Zhang. Privsyn: Differentially private data synthesis. *arXiv preprint arXiv:2012.15128*, 2020.

A Missing Proofs

Proof of Theorem 5: Publishing m InDif scores with $\mathcal{N}(0, 8m/\rho'\mathbf{I})$ satisfies ρ'-zCDP.

Proof. The proof is trivial given Lemma 4, Theorem 1 and Theorem 3: Because the sensitivity of InDif is 4, publishing it with $\mathcal{N}(0, 8m/\rho'\mathbf{I})$ satisfies ρ'/m-zCDP. For m InDif scores, by composition, publishing all of them satisfies ρ'-zCDP. □

Proof of Theorem 6: (1) The marginal M has sensitivity $\Delta_{\mathsf{M}} = 1$; (2) Publishing M with noise $\mathcal{N}(0, 1/2\rho\mathbf{I})$ satisfies ρ-zCDP.

Proof. We first prove the marginal function has sensitivity 1. A marginal M_A specified by a set of attributes A is a frequency distribution table, showing the number of record with each possible combination of values for the attributes. For two marginals M_A and M'_A, where M'_A is obtained by adding or removing one user to M_A. In general, for any A, it is obviously that $\Delta_{\mathsf{M}} = |\mathsf{M} - \mathsf{M}'|_2 = 1$.

Given this fact, by Theorem 3, it is trivial that adding $\mathcal{N}(0, 1/2\rho\mathbf{I})$ to a marginal satisfies ρ-zCDP. □

Proof of Lemma 4: $\Delta_{\mathbf{InDif}} = 4$.

Proof. Assume D contains n records and consider the two attributes a and b. Denote the number of users for histogram on attribute a as $a_1, a_2, \ldots$, and $b_1, b_2, \ldots$ for b. For the two-way marginal on a, b, denote the number of users for it as $\alpha_{11}, \alpha_{12}, \ldots$.

The metric InDif_{ab} is

$$\text{InDif}_{ab} = \sum_{ij} \left| \frac{a_i b_j}{n} - \alpha_{ij} \right|$$

If we add one user (wlog, whose values for a and b are x and y),

$$\text{InDif}'_{ab} = \sum_{i \neq x, j \neq y} \left| \frac{a_i b_j}{n+1} - \alpha_{ij} \right| + \sum_{i \neq x} \left| \frac{a_i (b_y + 1)}{n+1} - \alpha_{iy} \right| + \sum_{j \neq y} \left| \frac{(a_x + 1) b_j}{n+1} - \alpha_{xj} \right| + \left| \frac{(a_x + 1)(b_y + 1)}{n+1} - (\alpha_{xy} + 1) \right|$$

Since $|s| - |t| \leq |s - t|$, the sensitivity is given by

$$\Delta_{\text{InDif}} = |\text{InDif}_{ab} - \text{InDif}'_{ab}|$$
$$\leq \sum_{i\neq x, j\neq y} \left| \frac{a_i b_j}{n(n+1)} \right|$$
$$+ \sum_{i\neq x} \left| \frac{a_i b_y}{n(n+1)} - \frac{a_i}{n+1} \right|$$
$$+ \sum_{j\neq y} \left| \frac{a_x b_j}{n(n+1)} - \frac{b_j}{n+1} \right|$$
$$+ \left| \frac{(n+1)a_x b_y - n(a_x+1)(b_y+1) + n(n+1)}{n(n+1)} \right|$$
$$= \frac{\sum_{i\neq x, j\neq y} a_i b_j - \sum_{i\neq x}(a_i b_y - n a_i) - \sum_{j\neq y}(a_x b_j - n b_j)}{n(n+1)}$$
$$+ \frac{(n+1)a_x b_y - n(a_x+1)(b_y+1) + n(n+1)}{n(n+1)} \quad (2)$$
$$= \frac{(n-a_x)(n-b_y) - (n-a_x)(b_y-n) - (a_x-n)(n-b_y)}{n(n+1)}$$
$$+ \frac{(n+1)a_x b_y - n(a_x+1)(b_y+1) + n(n+1)}{n(n+1)} \quad (3)$$
$$= \frac{4\left(n^2 - (a_x+b_y)n + a_x b_y\right)}{n(n+1)} = \frac{4(n-a_x)(n-b_y)}{n(n+1)} \leq 4$$

In the above derivation, Equation 2 is due to $\frac{a_i b_j}{n(n+1)} \geq 0$, $\frac{a_i b_y}{n(n+1)} - \frac{a_i}{n+1} \leq 0$, $\frac{a_x b_j}{n(n+1)} - \frac{b_j}{n+1} \leq 0$ and $\frac{(n+1)a_x b_y - n(a_x+1)(b_y+1) + n(n+1)}{n(n+1)} = \frac{(n-a_x)(n-b_y)}{n(n+1)} \geq 0$. Equation 3 is due to $\sum_{i\neq x} a_i = n - a_x$, $\sum_{j\neq y} b_j = n - b_y$ and $\sum_{i\neq x, j\neq y} a_i b_j = (n-a_x)(n-b_y)$.

□

B Comparison of InDif and Entropy-based Metrics

To evaluate the impact of noise on the dependency metrics, one should consider both sensitivity and the range of the metrics. In this section, we compare InDif with two dependency metrics in the literature with respect to sensitivity and range.

Mutual Information (MI) [53]. PrivBayes adopts mutual information to measure the dependency between attributes. For attribute A and B, their mutual information $I(A;B)$ is defined as [3]

$$\sum_{a\in dom(A)} \sum_{b\in dom(B)} \Pr[A=a, B=b] \log \frac{\Pr[A=a,B=b]}{\Pr[A=a]\Pr[B=b]}$$

From [53], we know that the sensitivity of MI is $\frac{2}{n}\log\frac{n+1}{2} + \frac{n-1}{n}\log\frac{n+1}{n-1}$. Besides, the range of MI is $[0, \log c]$, where $c = \max\{c_A, c_B\}$, c_A and c_B are the number of possible values for attribute A and B, respectively. Thus, the noise-range ratio of MI is defined as

$$R_{MI} = \frac{1}{n} \cdot \frac{2\log\frac{n+1}{2} + (n+1)\log\frac{n+1}{n-1}}{\log c}$$

[3] All logarithms used in this section are to the base 2.

c	2	50	100	1000	10000	100000
$n \cdot R_{InDif}$	2.0	2.0	2.0	2.0	2.0	2.0
$n \cdot R_{MI}$	39.3	7.0	5.9	3.9	3.0	2.4
$n \cdot R_{Ent}$	41.8	7.4	6.3	4.2	3.1	2.5

Table 3: Noise-range ratio of different metrics when $n = 600000$ and c is varying.

	Adult	Accident	Loan	Colorado
InDif	0.017	0.028	0.161	0.137
MI	34	314	543	735
SUC	396	2858	4205	6933

Table 4: Relative error of different metrics when $\varepsilon = 2.0$.

Symmetrical Uncertainty Coefficient (SUC) [12]. BSG adopts symmetrical uncertainty coefficient to measure the dependency between attributes, which is defined as

$$corr(A,B) = 2 - 2\frac{H(A,B)}{H(A)+H(B)}$$

where $H(\cdot)$ is the entropy function.

To achieve differential privacy, the authors in [12] propose to add noise to three entropy values in corr(A, B), respectively. The authors prove that the sensitivity of entropy is $\frac{1}{n}\left[2 + \frac{1}{\ln 2} + 2\log n\right]$. Besides, the range of entropy is $[0, \log c]$. Thus, the noise-range ratio of entropy is given by

$$R_{Ent} = \frac{1}{n} \cdot \frac{2 + \frac{1}{\ln 2} + 2\log n}{\log c}$$

Comparison with InDif. Recall that the sensitivity and range of InDif is 4 and $[0, 2n]$, respectively; thus, its noise-range ratio is given by $R_{InDif} = \frac{1}{n} \cdot 2$.

We list the noise-range ratio of three methods in Table 3 when c varies. We set $n = 600000$ which is the case of three datasets in our experiments. We observe that the noise-range ratio of InDif is consistently smaller than the other two methods when $c \leq 100000$. In the three datasets in our experiments, most of the attributes contains less than 100 possible values, and the noise-range ratio of InDif is 3 times smaller than the other two methods.

Comparison of Relative Errors. To further evaluate the impact of noise on real-world datasets, we compare the relative errors between true values and noisy values of different metrics in Table 4 when $\varepsilon = 2.0$. The relative errors are calculated as $\frac{1}{m}\sum_{i=1}^{m}\left|\frac{s_i - \tilde{s}_i}{s_i}\right|$, where m is the total number of pair-wise marginals, s_i and $\tilde{s}_i$ are the true value and noisy value of marginal i, respectively. We run each experiment 1000 times and report the average relative error.

The experimental results show that the relative errors of InDif are significantly smaller than MI and SUC. The reason is that most of the MI values and SUC values are much smaller than their maximal value $\log C$, while most of the InDif values are close to their maximal value $2n$. For example, in the Colorado dataset, 78% of the MI values and 87% of the SUC values are smaller than 0.1 (much smaller than

$\log C$). In another hand, 37% of the InDif values are larger than $0.5n$ (close to $2n$).

C Computational Complexity Analysis

In this section, we first theoretically analyze the computational complexity of different methods, and then empirically evaluate the running time and memory consumption.

Time Complexity. The computational time for all methods consist of two parts, marginal selection and dataset generation.

For PrivBayes, the marginals are selected by constructing a Bayesian network. The general idea is to start with a randomly selected node, then gradually add node to the Bayesian network that maximally increase MI of the selected nodes. To reduce time complexity, PrivBayes only consider at most γ parents nodes in the selected nodes for each newly added node. The number of pairs considered in iteration i is $(d-i)\binom{i}{\gamma}$, where d is the number of attributes; thus summing over all iterations the computational complexity is bounded by $d\sum_{i=1}^{d}\binom{i}{\gamma} = d\binom{d+1}{\gamma+1}$. In the dataset generation step, PrivBayes simply sample records one-by-one using the Bayesian network; thus the time complexity is $O(nd)$, where n is the number of synthetic records.

For PGM, except for marginal selection and dataset generation, it includes another component that learn the parameters of Markov random field. The core idea is to use all the marginals and gradient decent to update the parameters. The gradient decent process would repeat t_{pg} times until convergence. In practice, t_{pg} is always set to be larger than 10000. Thus, the time complexity for learning Markov random field is $O(t_{pg}k_{pg})$, where k_{pg} is the number of marginals. The time complexity for generating synthetic dataset is the same with PrivBayes, *i.e.*, $O(nd)$. Notice that PGM does not provide method to select marginals, we only report the time complexity for parameter learning and dataset generation in Table 5.

For PrivSyn, there are $m = \binom{d}{2} = \frac{d(d-1)}{2}$ possible pairwise marginals in the marginal selection step. In iteration i of Algorithm 1, we need to check $m-i$ pairwise marginals; thus, the time complexity is $\sum_{i=1}^{k_{ps}}(m-i) = k_{ps}m - \frac{k_{ps}(k_{ps}+1)}{2} = O\left(k_{ps}d^2\right)$. In the dataset generation step, we should go through all marginals t_{ps} times to ensure consistency. Thus, the time complexity is $t_{ps}k_{ps}$ and we typically set $t_{ps} = 100$.

Space Complexity. The memory consumption of all methods consist of two parts, marginal tables and synthetic dataset. The memory consumption of synthetic dataset for all methods are the same, *i.e.*, $O(nd)$. The memory consumption for marginal tables differs in the number of marginals $k_\star$ and the average number of cells for each marginal $C_\star$. Specifically, PrivBayes contains $d-1$ marginals where each marginal contains at most $\gamma+1$ attributes. The number of marginals for PGM is unlimited; however, when the number of marginals is large, the Markov random field can be dense, resulting in large clique in the induced junction tree, which can be prohibitively large. PrivSyn uses the 2-way marginal; thus the average number of cells in each marginal is relatively small. The number of marginals is typically in the range of $[100, 700]$ in our experiment.

	Time Complexity	Space Complexity
PrivBayes	$O\left(d\binom{d+1}{\gamma+1}+nd\right)$	$O(C_{pb}d+nd)$
PGM	$O(t_{pg}k_{pg}+nd)$	$O(C_{pg}k_{pg}+nd)$
PrivSyn	$O\left(k_{ps}d^2+t_{ps}k_{ps}\right)$	$O(C_{ps}k_{ps}+nd)$

Table 5: Comparison of computational complexity for different methods. $n, d, k_\star$ stand for the number of records in synthetic dataset, the number of attributes and the number of marginals, respectively; $C_\star$ stands for the average number of cells in each marginal; $t_\star$ stands for the number of required iterations in each method.

Datasets	Adult	Accident	Loan	Colorado
PrivBayes	1 min	2 min	7 min	10 min
PGM	4 min	18 min	40 min	1 h 10 min
PrivSyn	4 min	40 min	2 h 10 min	3 h 30 min

Table 6: Comparison of running time for different methods.

Datasets	Adult	Accident	Loan	Colorado
PrivBayes	0.06	0.13	0.36	0.43
PGM	0.06	0.13	0.36	0.43
PrivSyn	0.06	0.13	0.36	0.43

Table 7: Comparison of memory consumption of different methods. The unit is Gigabytes.

Empirical Evaluation. Table 6 and Table 7 illustrate the running time and memory consumption for all methods on four datasets in our experiment.

The empirical running time in Table 6 shows that PrivBayes performs best in terms of running time, since it requires only $d-1$ marginals and the sampling process is very fast. PGM uses the same set of marginals with PrivBayes, while it needs additional time to learn the parameters of Markov random field, and the gradient decent process should repeat more than 10000 times. PrivSyn is slower than PrivBayes and PGM since it uses much more marginals. For example, when $\varepsilon = 2.0$, the Colorado dataset has about 700 marginals, while PrivBayes and PGM only have 96 marginals. Although PrivSyn costs more time than PrivBayes and PGM, it only takes less than 4 hours to generate large dataset such as Colorado (97 attributes with total domain of $5 \cdot 10^{162}$), which is acceptable in practice considering its superior performance.

The empirical memory consumption in Table 7 shows that the memory consumption for all methods are similar for the same dataset. The reason is that the memory consumption for all methods are dominated by the storage of synthetic datasets, and the storage of marginal tables are less than 10 Megabytes for all datasets.