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 to demonstrate the performance of our method.

1 INTRODUCTION

Differential privacy (DP) (Dwork, 2006) has been accepted as the *de facto* notion for protecting privacy. A number of companies and government agencies has used DP for privacy-preserving data analysis. Previous work on DP mostly focuses on designing tailored algorithms for specific data analysis tasks. However, this paradigm is time consuming, requires a lot of expertise knowledge, and is error-prone. 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, and do not consume additional privacy budget. 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 (Zhang et al., 2017) 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 (Mckenna et al., 2019). In 2018, NIST hosted a Differential Privacy Synthetic Data Challenge (NIST), PGM achieves the best result. Approaches that do not use probabilistic graphical models either are computationally inefficient or have poor empirical performance. However, 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.

Our Contributions. In this paper, we propose PrivSyn, for differentially private synthetic data generation. 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 represent 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.

Concretely, 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 (Arora et al., 2012) 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.

2 PRELIMINARIES

2.1 DIFFERENTIAL PRIVACY

The notion of differential privacy (Dwork et al., 2006) guarantees 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 $\epsilon > 0, \delta \geq 0$, if and only if for any two neighboring datasets D and D', we have*

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

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

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 by $\mathcal{A}(D) = f(D) + \mathcal{N}\left(0, \Delta_f^2 \sigma^2 \mathbf{I}\right)$, where $\Delta_f = \max_{(D,D'):D \simeq D'} ||f(D) - f(D')||_2$, $\mathcal{N}(0, \Delta_f^2 \sigma^2 \mathbf{I})$ denotes a multi-dimensional random variable sampled from the normal distribution with mean 0 and standard deviation $\Delta_f \sigma$, and $\sigma = \sqrt{2 \ln \frac{1.25}{\delta}}/\epsilon$.

2.2 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.*, (Qardaji et al., 2014; Xiao et al., 2010; Abadi et al., 2016; Li et al., 2010)). 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 OUR PROPOSAL

3.1 METHOD OVERVIEW

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 an exponentially large space. To overcome this issue, one promising approach is to estimate many 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. Both PrivBayes and PGM follow this framework. We propose new techniques for all four of these steps in PrivSyn.

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. 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. Due to space limitation, we defer the details of our marginal selection method, noise addition method and post-processing method to Appendix B.

3.2 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 (Zhang et al., 2017; Mckenna et al., 2019) 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.

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. 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. In order to change D_s to make it consistent with the marginal, we change records in D_s.

	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	$\mathsf{S}_{\{\mathrm{I,G}\}}(v)$	$\mathsf{T}_{\{\mathrm{I,G}\}}(v)$
⟨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}.

	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 1: Example of the synthesized dataset before and after updating procedure.

Gradually Update Method (GUM). We empirically find that the convergence performance of MCF is not good. 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 (Arora et al., 2012) 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., $\min\{n^t - n^s, \alpha n^s\}$, 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.

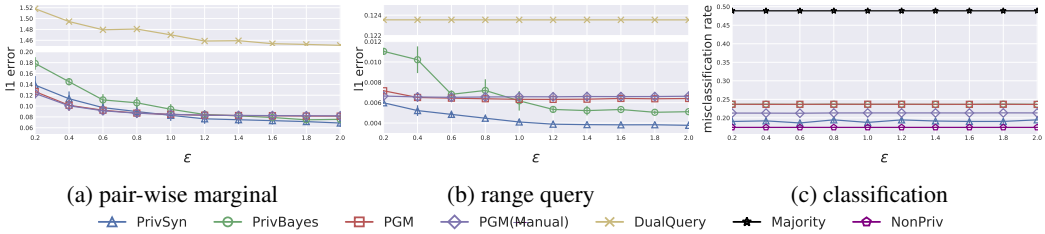
Figure 1 gives a running example. Before updating, we have 4 out of 5 records have the combination $\langle high, male \rangle$, and 1 record has $\langle high, female \rangle$. To get closer to the target marginal of 0.2 and 0.8 for these two cells, we want to change 2 of the $\langle high, male \rangle$ records to be $\langle high, female \rangle$. In this example, we have $\alpha = 2.0, \beta = 0.5$ 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 1). 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. To further improve the convergence performance, we propose several tricks in Appendix C.

4 EVALUATION

Datasets. We use the Colorado (NIST) dataset in our experiment, which is the census dataset of Colorado State in 1940. This dataset is used in the final round of the NIST challenge. It contains $662,000$ records and 97 attributes with a domain of $5 \cdot 10^{162}$.

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. (1) Marginal release: We compute all the 2-way marginals and use the average ℓ_1 error to measure the performance. (2) Range query: We randomly sample 1000 range queries, each contains 3 attributes. We use the average ℓ_1 error to measure the performance. (3) Classification: We use the synthesized dataset to train an SVM classification model, and use misclassification rate to measure the performance.

Competitors. We compare PrivSyn with PrivBayes, PGM, and a game-based method DualQuery. For the classification task, we have another two competitors, *i.e.*, Majority (blindly predicts the label by the majority label) and NonPriv (without enforcing differential privacy). Note that we have two versions of synthesis methods for PrivSyn, *i.e.*, MCF and GUM.



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

Figure 2: End-to-end Comparison of different dataset generation methods.

Experimental Results. Figure 2 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 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.

REFERENCES

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

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*, pp. 308–318. ACM, 2016.

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.

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

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

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

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

Cynthia Dwork. Differential privacy. In *ICALP*, pp. 1–12, 2006.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014. ISSN 1551-305X. doi: 10.1561/0400000042. URL http://dx.doi.org/10.1561/0400000042.

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

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

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

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*, pp. 123–134, 2010.

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

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

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*, pp. 1435–1446. ACM, 2014.

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

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

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.

A 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 (Dwork & Roth, 2014) shows that the privacy composes linearly, i.e., the sequential composition satisfies $(\sum_i^k \epsilon_i, \sum_i^k \delta_i)$-DP. When $\epsilon_i = \epsilon$ and $\delta_i = \delta$, the advanced composition bound from (Dwork et al., 2010) states that the composition satisfies $(\epsilon\sqrt{2k\log(1/\delta')} + k\epsilon(e^\epsilon - 1), k\delta + \delta')$-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) (Bun & Steinke, 2016))**.** *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 (Bun & Steinke, 2016):

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 (Bun & Steinke, 2016), 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.

B DIFFERENTIALLY PRIVATE MARGINAL SELECTION

B.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 (Zhang et al., 2017) 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.*, $\text{InDif}_{a,b} = |\mathsf{M}_{a,b} - \mathsf{M}_a \times \mathsf{M}_b|_1$.

v	$\mathsf{M}_{\text{gender}}(v)$
⟨male,∗⟩	0.40
⟨female,∗⟩	0.60

(a) 1-way marginal for gender.

v	$\mathsf{M}_{\text{age}}(v)$
⟨∗,teenager⟩	0.20
⟨∗,adult ⟩	0.30
⟨∗,elderly⟩	0.50

(b) 1-way marginal for age.

v	
⟨male, teenager⟩	0.08
⟨male, adult⟩	0.12
⟨male, elderly⟩	0.20
⟨female, teenager⟩	0.12
⟨female, adult⟩	0.18
⟨female, elderly⟩	0.30

(c) 2-way marginal assume independent

v	
⟨male, teenager⟩	0.10
⟨male, adult⟩	0.10
⟨male, elderly⟩	0.20
⟨female, teenager⟩	0.10
⟨female, adult⟩	0.20
⟨female, elderly⟩	0.30

(d) Actual 2-way marginal

Figure 3: Example of the calculation of InDif.

Figure 3 gives an example to illustrate the calculation of InDif. The 2-way marginal in Figure 3c is directly calculated by the 1-way marginal of gender and age, without analyzing the dataset; and Figure 3d gives the actual 2-way marginal. In this example, 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*:* $\Delta_{\mathit{InDif}} = 4$.

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 $\alpha_{11}, \alpha_{12}, \cdots$.

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),

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

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

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

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

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. InDif typically 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.

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.*

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. □

B.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.*

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. □

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:

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

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:

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

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

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

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 3. 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.

Algorithm 1: Marginal Selection Algorithm

Input: Number of pairs m, privacy budget ρ, dependency error $\langle \phi_i \rangle$, marginal size $\langle c_i \rangle$;
Output: Selected marginal set X;

$X \leftarrow \varnothing$; $t \leftarrow 0$; $E_0 \leftarrow \sum_{i \in \bar{X}} \phi_i$;
while *True* **do**
  foreach *marginal* $i \in \bar{X}$ **do**
    Allocate ρ to marginals $j \in X \cup \{i\}$;
    $E_t(i) = \sum_{j \in X \cup \{i\}} c_j \sqrt{\frac{1}{\pi \rho_j}} + \sum_{j \in \bar{X} \setminus \{i\}} \phi_j$;
  $\ell \leftarrow \arg\min_{i \in \bar{X}} E_t(i)$;
  $E_t \leftarrow E_t(\ell)$;
  if $E_t \geq E_{t-1}$ **then**
    Break
  $X \leftarrow X \cup \{l\}$;
  $t \leftarrow t + 1$;

Combine Marginals. Till now, we assume two-way 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.

B.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 (Ding et al., 2011; Qardaji et al., 2014). Note that (Ding et al., 2011; Qardaji et al., 2014) both assume the privacy budget is evenly distributed. We extend it to the uneven case.

Algorithm 2: Marginal Combine Algorithm

Input: Selected pairwise marginals X, threshold γ
Output: Combined marginals $\mathcal{X}$

Convert X to a set of pairs of attributes;
Construct graph $\mathcal{G}$ from the pairs;
$S \leftarrow \varnothing$; $\mathcal{X} \leftarrow \varnothing$
foreach *clique size* s *from* m *to* 3 **do**
  $C_s \leftarrow$ cliques of size s in $\mathcal{G}$
  foreach *clique* $c \in C_s$ **do**
    if $|c \cap S| \leq 2$ *and domain size of* $c \leq \gamma$ **then**
      Append c to $\mathcal{X}$
      Append the attributes of c to S

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 (Qardaji et al., 2014), 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 (3)), 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.*, (Lee et al., 2015; Wang et al., 2020; Bassily, 2019)); 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.

C 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 (dec) 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 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 (Krizhevsky et al., 2012).

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.