Differentially Private Marginals

Sivakanth Gopi[2], Sepideh Mahabadi[2], and Sergey Yekhanin[2]

[2]Microsoft Research, {sigopi,smahabadi,yekhanin}@microsoft.com

Abstract

We give a differentially private algorithm to output approximate k-way marginals of tabular data. Marginals are the counts of various tuples in the data, such as the number of user records x such that $(x_i, x_j) = (a, b)$ (which is a 2-way marginal). We are interested in computing approximate k-way marginals for $k = 1, 2, \ldots, T$. The algorithm is inspired by differentially private n-gram extraction [KGKY21].

1 An Algorithm for DP Marginals

In this section we describe our algorithm for DP Marginals. The pseudocode is presented in Algorithm 1. Given private tabular data, we will assume that the rows are indexed by users and the columns denote various attributes. A row is also called a record. A generic record is denoted by X, where the i^{th} column of the record is X_i. A *k-tuple* is defined by a set of k columns along with possible values for each of them, i.e., a k-tuple looks like $(X_{i_1} = a_1, X_{i_2} = a_2, \ldots, X_{i_k} = a_k)$. A record can have many empty columns, so we will denote the empty value with $\perp$. We denote by $\mathrm{wt}(X)$ to be the number of non-empty columns in X. The set of all non-empty[1] k-tuples contained in a record X is denoted by $M_k(X)$. Therefore $|M_k(X)| = \binom{\mathrm{wt}(X)}{k}$. For example if $X = (a, b, \perp, a)$, then $\mathrm{wt}(X) = 3$ and $M_2(X) = \{(X_1 = a, X_2 = b), (X_1 = a, X_4 = a), (X_2 = b, X_4 = a)\}$.

The algorithm iteratively extracts k-tuples for $k = 1, 2, \ldots, T$, i.e., the algorithm uses the already extracted $(k-1)$-tuples to extract k-tuples. Let S_k denote the extracted set of k-tuples.

1.1 Controlling spurious n-tuples using ρ_k:

In our privacy analysis (Section 1.2), we will show that the privacy of the DP Marginals algorithm depends only on ρ_1 and $\sigma_1, \sigma_2, \ldots, \sigma_T$. In particular, $\rho_2, \ldots, \rho_T$ do not affect privacy. Instead, they are used to control the number of *spurious* k-tuples that we extract, i.e., k-tuples which are not actually used by any user but output by the algorithm.

Proposition 1.1. For $k \geq 2$, the expected number of spurious k-tuples output by Algorithm 1 is at most $|V_k|(1 - \Phi(\rho_k/(\sigma_k \Delta_k^{1/2})))$ where Φ is the Gaussian CDF. And the algorithm will not output any spurious 1-tuples.

Proof. A spurious k-tuple will have zero weight in the histogram H_k that the algorithm builds. So after adding $N(0, \Delta_k \sigma_k^2)$ noise, the probability that it will cross the threshold ρ_k is exactly $1 - \Phi(\rho_k/(\sigma_k \Delta_k^{1/2}))$.

□

Larger we set ρ_k, smaller the number of spurious k-tuples. But setting large ρ_k will reduce the number of non-spurious k-tuples extracted by the algorithm. So $\rho_2, \ldots, \rho_T$ should be set delicately

[1]None of the k values should be empty.

Algorithm 1: Algorithm for differentially private marginals

Input: A set of N users where each user u has a record X^u
T: maximum length of marginals to be extracted
Q: Percentile parameter to estimate maximum contributions (can set it to 99% for example)
η: fraction of spurious k-tuples in the output
$\sigma_1, \sigma_2, \ldots, \sigma_T$: Noise parameters
ε_Q: ε for DP Q^{th}-percentile
δ : final δ value in the (ε, δ)-DP guarantee
Output: $S_1, S_2, \ldots, S_T$ where S_k is a set of k-tuples extracted and $H_1, H_2, \ldots, H_T$ which are noisy marginals of $S_1, S_2, \ldots, S_T$

```
// Learn 1-tuples and 1-way marginals
for u ∈ [N] do
    W_u ← M_1(X^u) ;                                  // non-empty 1-tuples with user u
Δ_1 ← Compute Q^th-percentile of {|W_1|, |W_2|, ..., |W_N|} using ε_Q-DP mechanism;
// Limit user contributions
if |W_u| > Δ_1 then
    W_u ← Randomly choose Δ_1 items from W_u;
V_1 ← W_1 ∪ W_2 ∪ ··· ∪ W_N
ρ_1 ← 1 + σ_1 Δ_1^{1/2} Φ^{-1}((1 − δ/2)^{1/Δ_1});
S_1, H_1 ← NoisyThresholding((W_u : u ∈ [N]), Ω = V_1, Δ = Δ_1, ρ = ρ_1, σ = σ_1 Δ_1^{1/2});
// Iteratively learn k-tuples and k-way marginals
for k = 2 to T do
    // Calculate valid k-tuples
    V_k ← All possible k-tuples whose (k − 1)-subtuples belong to S_{k−1};
    // Prune away invalid k-tuples
    for u ∈ [N] do
        W_u ← M_k(X^u) ∩ V_k ; // non-empty k-tuples with user u which are also valid
    Δ_k ← Compute Q^th-percentile of {|W_1|, |W_2|, ..., |W_N|} using ε_Q-DP mechanism;
    ρ_k ← σ_k Δ_k^{1/2} Φ^{-1}(1 − η min{1, |S_{k−1}|/|V_k|});
    S_k, H_k ← NoisyThresholding((W_u : u ∈ [N]), Ω = V_k, Δ = Δ_k, ρ = ρ_k, σ = σ_k Δ_k^{1/2});
Output S_1, S_2, ..., S_T and H_1, H_2, ..., H_T;
```

to balance this tension. One convenient choice of ρ_k for $k \geq 2$ is to set,

$$\rho_k = \Delta_k^{1/2} \sigma_k \Phi^{-1}\left(1 - \eta \min\left\{1, \frac{|S_{k-1}|}{|V_k|}\right\}\right)$$

for some $\eta \in (0, 1)$. This implies that the expected number of spurious k-tuples output is at most $\eta \min\{|S_{k-1}|, |V_k|\}$ by Proposition 1.1. And the total number of spurious k-tuples output is at most $\eta(|S_1| + |S_2| + \cdots + |S_{T-1}|)$. Therefore spurious k-tuples output by the algorithm are at most an η-fraction of all the k-tuples output.

Remark 1.1. One can also completely eliminate spurious all spurious k-tuples if we set

$$\rho_k = 1 + \sigma_k \Delta_k^{1/2} \Phi^{-1}\left(\left(1 - \frac{\delta}{2T}\right)^{1/\Delta_k}\right)$$

for $k = 1, 2, \ldots, T$, and while extracting k-tuples, we set

$$\Omega = W_1 \cup W_2 \cup \cdots \cup W_N$$

where we prune each W_i to have size at most Δ_k. But typically, this will increase the thresholds beyond what we set in Algorithm 1 which will reduce the number of k-tuples we learn.

1.2 Privacy Analysis

We are now ready to prove the privacy of our DP Marginals algorithm.

Algorithm 2: NoisyThresholding $((W_u : u \in [N]), \Omega, \Delta, \rho, \sigma)$

Input: A set of N users where each user u has some subset $W_u \subset \Omega$
Δ: maximum contribution parameter
ρ: Threshold parameter
σ: Noise parameter.
Output: S: Set of items extracted and H: a histogram of noisy counts of items in S

```
// Initialize histogram to 0 for all items in the universe Ω
for ω in Ω do
    H[ω] ← 0;
// Build histogram by limiting user contributions to Δ
for u = 1 to N do
    // Limit user contributions
    if |W_u| > Δ then
        W_u ← Randomly choose Δ items from W_u;
    for ω in W_u do
        H[ω] ← H[ω] + 1;
// Add noise to H
for ω ∈ Ω do
    H[ω] ← H[ω] + σN(0, 1)
// Output items which cross the threshold ρ
S = {} (empty set);
for ω ∈ Ω do
    if H[ω] > ρ then
        S ← S ∪ {ω};
H ← H|_S ;                      // Remove items not in S from H
Output S, H
```

Theorem 1.1. Let $\varepsilon > 0, 0 < \delta < 1$. Then Algorithm 1 satisfies (ε, δ)-DP if

$$\frac{T\varepsilon_Q^2}{2} + \frac{1}{2}\sum_{i=1}^{T}\frac{1}{\sigma_i^2} \leq \sqrt{\varepsilon + \log(2/\delta)} - \sqrt{\log(2/\delta)}.$$

Proof. The privacy of learning 1-tuples in Algorithm 1 is given by the composition of a Gaussian mechanism with ℓ_2-sensitivity 1 and noise σ_1 composed with $(0, \delta/2)$-algorithm as shown in [GGK+20] if we set ρ_1 as shown. The construction of k-tuples is a Gaussian mechanism with ℓ_2-sensitivity 1 and noise σ_k. We now combine these privacy guarantees using zCDP framework from [BS16] to get $(\varepsilon, \delta/2)$-DP. □

Remark 1.2. Let $\rho = \sqrt{\varepsilon + \log(2/\delta)} - \sqrt{\log(2/\delta)}$. To begin with, we suggest using only 10% of the privacy budget for finding the Q^{th} percentiles, i.e., we set ε_Q and $\sigma_1 = \cdots = \sigma_T$ such that $T\varepsilon_Q^2/2 = \rho/10$ and $\frac{1}{2}\sum_{i=1}^{T} 1/\sigma_i^2 = 9\rho/10$.

Remark 1.3. One can improve the privacy bound using numerical composition from [GLW21]. We think this will further reduce ε by an additive 1 or 2 for a fixed δ.

Remark 1.4. Note that the choice of $\rho_2, \ldots, \rho_k$ doesn't affect the privacy parameters. It only affects the fraction of spurious k-tuples in the output. We are using η parameter to control the fraction of spurious k-tuples, and $\rho_2, \ldots, \rho_k$ are set automatically based on η.

References

[BS16] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[GGK$^+$20] Sivakanth Gopi, Pankaj Gulhane, Janardhan Kulkarni, Judy Hanwen Shen, Milad Shokouhi, and Sergey Yekhanin. Differentially private set union. In *International Conference on Machine Learning*, pages 3627–3636. PMLR, 2020.

[GLW21] Sivakanth Gopi, Yin Tat Lee, and Lukas Wutschitz. Numerical composition of differential privacy. *Advances in Neural Information Processing Systems*, 34, 2021.

[KGKY21] Kunho Kim, Sivakanth Gopi, Janardhan Kulkarni, and Sergey Yekhanin. Differentially private n-gram extraction. *Advances in Neural Information Processing Systems*, 34, 2021.