CALM: Consistent Adaptive Local Marginal for Marginal Release under Local Differential Privacy

Zhikun Zhang[1,2], Tianhao Wang[2], Ninghui Li[2], Shibo He[3], Jiming Chen[1]
[1]State Key Laboratory of Industrial Control Technology & Cyber Security Research Center, Zhejiang University
{zhangzhk,cjm}@zju.edu.cn
[2]Department of Computer Science, Purdue University
{zhan3072,tianhaowang,ninghui}@purdue.edu
[3]State Key Laboratory of Industrial Control Technology, Zhejiang University
s18he@zju.edu.cn

ABSTRACT

Marginal tables are the workhorse of capturing the correlations among a set of attributes. We consider the problem of constructing marginal tables given a set of user's multi-dimensional data while satisfying Local Differential Privacy (LDP), a privacy notion that protects individual user's privacy without relying on a trusted third party. Existing works on this problem perform poorly in the high-dimensional setting; even worse, some incur very expensive computational overhead. In this paper, we propose CALM, Consistent Adaptive Local Marginal, that takes advantage of the careful challenge analysis and performs consistently better than existing methods. More importantly, CALM can scale well with large data dimensions and marginal sizes. We conduct extensive experiments on several real world datasets. Experimental results demonstrate the effectiveness and efficiency of CALM over existing methods.

ACM Reference Format:
Zhikun Zhang, Tianhao Wang, Ninghui Li, Shibo He, Jiming Chen. 2018. CALM: Consistent Adaptive Local Marginal for Marginal Release under Local Differential Privacy. In 2018 ACM SIGSAC Conf. on Computer and Communications Security (CCS'18), October 15–19, 2018, Toronto, ON, Canada. ACM, New York, NY, USA, 18 pages. https://doi.org/10.1145/3243734.3243742

1 INTRODUCTION

In recent years, differential privacy [13, 14] has been increasingly accepted as the *de facto* standard for data privacy in the research community [3, 15, 21, 24, 27]. Most early work on DP are in the centralized setting, where a trusted *data curator* obtains data from all individuals, and processes the data in a way that protects privacy of individual users. For example, the data curator could publish a private synopsis of the data, enabling analysis on the data, while hiding individual information.

Zhikun Zhang's work on this paper was done while working as a visiting student at Purdue University. The first two authors are co-first authors.

CCS '18, October 15–19, 2018, Toronto, ON, Canada

ACM ISBN 978-1-4503-5693-0/18/10…$15.00
https://doi.org/10.1145/3243734.3243742

Recently, techniques for satisfying differential privacy (DP) in the local setting, which we call LDP, have been studied and deployed. In the local setting for DP, there are many *users* and one *aggregator*. Unlike the centralized setting, the aggregator does not see the actual private data of each individual. Instead, each user sends randomized information to the aggregator, who attempts to infer the data distribution based on that. LDP techniques enable the gathering of statistics while preserving privacy of every user, without relying on trust in a single trusted third party. LDP techniques have been deployed by companies like Google [16, 17], Apple [1], Microsoft [12] and Samsung [9]. Exemplary use cases include collecting users' default browser homepage and search engine, in order to understand the unwanted or malicious hijacking of user settings; or frequently typed emoji's and words, to help with keyboard typing predictions.

Previous works on LDP focus on estimating the frequencies of frequent values the user possesses [6, 7, 16, 30, 37–39]. The natural and more general setting is when each user has multiple attributes, and the aggregator is interested in the joint distribution of some of these attributes. That is, the aggregator is interested in marginal tables over some subsets of attributes. Marginal tables are the workhorse of capturing the correlations among a set of attributes. Many analysis tasks require the availability of marginal statistics on multidimensional datasets. For example, finding correlations or fitting sophisticated prediction models.

Two recent papers [11, 31] considered the problem of publishing marginals under LDP. Kulkarni [11] et al. proposed to apply the Fourier Transformation method, which was used in publishing marginals under the centralized DP setting [5]. Ren et al. [31] proposed to apply the Expectation Maximization methods, originally developed by Fanti et al. [17] to infer marginals. These methods, however, performs poorly when the number of attributes is more than a few.

We propose a new method CALM, Consistent Adaptive Local Marginal, for computing any k-way marginals under the LDP setting. Our approach is inspired by PriView [29], which was designed for computing arbitrary k-way marginals under centralized DP for binary datasets (i.e., each attribute is binary). PriView privately publishes a synopsis of the dataset, which takes the form of m marginals each of the size ℓ. Using the synopsis, it can reconstruct any k-way marginal.

Similar to PriView, in CALM a number of marginal tables are generated. But there are several challenges when changing from centralized setting to the local setting. We need to integrate FO

protocols to construct marginals, and to extend the methods in PriView to deal with non-binary attributes. Furthermore, since the privacy parameter ϵ affects noises differently in the local setting, the error analysis, which is essential for guiding the choice of key algorithmic parameters, changes as well. A common challenge for many works on differential privacy is that there are often critical algorithmic parameters, the choice of which greatly impact the performance of the algorithm. However, such parameters are often chosen in ad hoc ways, or based on performance on the experimental datasets. Both PriView and CALM have two critical parameters. In the local setting, an additional source of errors is introduced and needed to be considered. We carefully analyze how different errors are affected by different parameters, deriving formulas for estimating them whenever possible. We then develop an approach for choosing these parameters in a principled way. Our approach takes as input one target error threshold, and use an algorithm to find parameter values, using the formulas for estimating errors.

We have implemented CALM and conducted extensive experimental evaluation to compare CALM with other state of the art methods. Experimental results show that CALM's expected value of the Sum of Squared Errors is often one to two orders of magnitude lower than the best current method in [11]. In addition, CALM scales to settings where existing methods do not. To demonstrate the importance of the marginal information in practice, we also evaluate the prediction performance of CALM versus other methods by training an SVM model on some fixed marginal. In most cases, we can see CALM can achieve near optimal results, while other methods are beaten by the naive method that always output the majority label.

To summarize, the main contributions of this paper are three folds:

- We introduce CALM for the marginal release problem under local differential privacy, which also work when there are non-binary attributes.
- We have conducted careful analysis on errors from three different sources, and developed an algorithm for choosing key algorithmic parameters for CALM.
- The performance of the proposed method is extensively evaluated on real-world datasets and demonstrated to greatly outperform state-of-the art approaches.

Roadmap. In Section 2, we present background knowledge of LDP and FO. We then go over the problem definition and existing solutions in Section 3. We present our proposed method in Section 4. Experimental results are presented in 5. Finally we discuss related work in Section 6 and provide concluding remarks in Section 7.

2 BACKGROUND

In the local setting for DP, there are many *users* and one *aggregator*. Each user possesses a value v from domain D, and the aggregator wants to learn the distribution of values among all users, in a way that protects the privacy of individual users.

2.1 Differential Privacy in the Local Setting

To protect privacy, each user perturbs the input value v using an algorithm Ψ and sends $\Psi(v)$ to the aggregator. The formal privacy requirement is that the algorithm $\Psi(\cdot)$ satisfies the following property:

Definition 1 (ϵ Local Differential Privacy). *An algorithm $\Psi(\cdot)$ satisfies ϵ-local differential privacy (ϵ-LDP), where $\epsilon \geq 0$, if and only if for any input $v_1, v_2 \in D$, we have*

$$\forall T \subseteq Range(\Psi): \ \Pr[\Psi(v_1) \in T] \leq e^{\epsilon} \Pr[\Psi(v_2) \in T],$$

where $Range(\Psi)$ denotes the set of all possible outputs of Ψ.

Since a user never reveals v to the aggregator and reports only $\Psi(v)$, the user's privacy is still protected even if the aggregator is malicious.

2.2 Frequency Oracles

A *frequency oracle (*FO*)* protocol enables the estimation of the frequency of any value $x \in D$ under LDP, which serves as the building block of other LDP tasks. It is specified by a pair of algorithms: Ψ is used by each user to perturb her input value, and Φ is used by the aggregator.

*2.2.1 Generalized Randomized Response (*GRR*).* This FO protocol generalizes the *randomized response* technique [40]. Here each user with private value $v \in D$ sends the true value v with probability p, and with probability $1-p$ sends a randomly chosen $v' \in D$ s.t. $v' \neq v$. More formally, the perturbation function is defined as

$$\forall_{y \in D} \ \Pr\left[\Psi_{\mathrm{GRR}(\epsilon)}(v) = y\right] = \begin{cases} p = \frac{e^{\epsilon}}{e^{\epsilon}+d-1}, & \text{if } y = v \\ q = \frac{1}{e^{\epsilon}+d-1}, & \text{if } y \neq v \end{cases}$$

This satisfies ϵ-LDP since $\frac{p}{q} = e^{\epsilon}$. To estimate the frequency of $v \in D$ (i.e., the ratio of the users who have v as private value to the total number of users), one counts how many times v is reported, and denote the count as $C(v)$, and then computes

$$\Phi_{\mathrm{GRR}(\epsilon)}(v) := \frac{C(v)/n - q}{p - q}$$

where n is the total number of users. For example, if 20% of users have value v, the expected number of v in all randomized reports is $0.2 * n * p + 0.8 * n * q$. If the aggregator sees exactly this number of reports, the estimated value is

$$\frac{(0.2np + 0.8nq)/n - q}{p - q} = \frac{0.2p + 0.8q - q}{p - q} = \frac{0.2p - 0.2q}{p - q} = 0.2$$

In [37], it is shown that this is an unbiased estimation of the true count, and the variance for this estimation is

$$\mathrm{Var}[\Phi_{\mathrm{GRR}(\epsilon)}(x)] = \frac{|D| - 2 + e^{\epsilon}}{(e^{\epsilon} - 1)^2 \cdot n} \tag{1}$$

The accuracy of this protocol deteriorates fast when the domain size $|D|$ increases. This is reflected in that the variance given in (1) is linear to $|D|$.

*2.2.2 Optimized Unary Encoding (*OUE*).* The Optimized Unary Encoding (OUE) [37] avoids having a variance that depends on $|D|$ by encoding the value into the unary representation. Wlog, let $D = [0..d-1]$; each value $v \in [0..d-1]$ is encoded into a binary string of length d such that the v-th bit is 1 and all othe bits are 0. The unary encodings of any two different values differ in exactly two bits. OUE applies GRR to each bit, but transmits 1's and 0's differently. The bit 1 is transmitted as a coin toss, i.e., it is perturbed

to 0 with probability 0.5; this can be viewed as applying GRR with $\epsilon = 0$. Doing this enables us to transmit each of the many ($|D| - 1$, to be precise) 0 bits with the maximum allowed privacy budget ϵ, so that the number of 1's resulting from perturbing the 0's is as small as possible. Doing this minimizes the estimation variance when $|D|$ is large [37].

Given reports y^j from all users $j \in [n]$, to estimate the frequency of v, the aggregator counts the number of reports with the bit corresponding to v set to 1, i.e., $C(x) = |\{j \mid y_x^j = 1\}|$. One then transforms $C(v)$ to its unbiased estimation

$$\Phi_{\mathsf{OUE}(\epsilon)}(x) \coloneqq \frac{C(x)/n - q}{\frac{1}{2} - q}$$

It is proved in [37] that the $\Psi_{\mathsf{OUE}}(\cdot)$ satisfies LDP, and estimated frequency is unbiased and has variance

$$\mathrm{Var}[\Phi_{\mathsf{OUE}(\epsilon)}(x)] = \frac{4e^\epsilon}{(e^\epsilon - 1)^2 \cdot n} \tag{2}$$

2.2.3 Adaptive FO. Comparing (1) with (2), the factor $|D| - 2 + e^\epsilon$ is replaced by $4e^\epsilon$. This suggests that for smaller $|D|$ (such that $|D| - 2 < 3e^\epsilon$), one is better off with GRR; but for large $|D|$, OUE is better and has a variance that does not depend on $|D|$.

For simplicity, we use FO to denote the adaptively chosen protocol, i.e., when domain size is less than $3e^\epsilon + 2$, GRR is used as FO; otherwise, OUE is used. It has variance

$$\mathrm{Var}[\Phi_{\mathsf{FO}(\epsilon)}(x)] = \min\left(\frac{4e^\epsilon}{(e^\epsilon - 1)^2}, \frac{|D| - 2 + e^\epsilon}{(e^\epsilon - 1)^2}\right) \cdot \frac{1}{n} \tag{3}$$

3 PROBLEM DEFINITION AND EXISTING SOLUTIONS

We consider the setting where each user has multiple attributes, and the aggregator is interested in the joint distribution of some attributes. Such multi-dimension settings occur frequently in the situation where LDP is applied. In [11, 31], researchers studied the problem of constructing marginals in the LDP setting.

3.1 Problem Definition: Centralized Setting

We assume that there are d attributes $\mathbb{A} = \{a_1, a_2, \ldots, a_d\}$. Each attribute a_i has c_i possible values. Wlog, we assume that the values for a_i are $[c_i] \coloneqq \{0, 1, \cdots, c_i - 1\}$. Each user has one value for each attribute. Thus user j's value is a d-dimensional vector, denoted by $v^j = \langle v_1^j, v_2^j, \ldots, v_d^j \rangle$ such that $v_i^j \in [c_i]$ for each i. The full domain for the users' values is given by $D = [c_1] \times [c_2] \times \cdots \times [c_d]$, in which $\times$ denotes cartesian product. The domain D has size $|D| = \prod_{i=1}^d c_i$.

Let us first consider the setting of answering marginal queries in the *centralized setting*, where the server has all users' data. For a population of n users, the *full contingency table* gives, for each value $v \in D$, the fraction of users having the value v. We use F to denote the full contingency table, and call the fraction for each value $v \in D$ a cell in the full contingency table.

The full contingency table gives the joint distribution of all attributes in $\mathbb{A}$. However, when the domain size is very large, e.g., when there are many attributes, computing the full contingency table can be prohibitively expensive. Oftentimes, one is interested in the joint distribution of some subsets of the attributes. Given a set of attributes $A \subseteq \mathbb{A}$, we use $V_A = \{\langle v_1, v_2, \ldots, v_d \rangle : v_i \in [c_i]$ if $a_i \in A$, otherwise $v_i = *\}$ to denote the set of all possible values specified by A.

	Gender	Age
v^1	male	teenager
v^2	female	teenager
v^3	female	adult
v^4	female	adult
...	...	...
v^n	male	elderly

(a) Dataset.

v	F(v)
⟨male, teenager⟩	0.20
⟨male, adult⟩	0.15
⟨male, elderly⟩	0.20
⟨female, teenager⟩	0.15
⟨female, adult⟩	0.20
⟨female, elderly⟩	0.10

(b) Full contingency table.

v	$\mathsf{M}_{\{\text{gender}\}}(v)$
⟨male,∗⟩	0.55
⟨female,∗⟩	0.45

(c) Marginal table for gender.

v	$\mathsf{M}_{\{\text{age}\}}(v)$
⟨∗,teenager⟩	0.35
⟨∗,adult ⟩	0.35
⟨∗,elderly⟩	0.30

(d) Marginal table for age.

Figure 1: Example of the dataset, the full contingency table, and the marginal tables.

When given a set A of k attributes, the *k-way marginal over A*, denoted by M_A, gives the fraction of users having each value in V_A. We call the fraction for each value $v \in V_A$ a cell of the marginal table. M_A contains fewer cells than the full contingency table F. Each cell in M_A can be computed from summing over the values in the cells in F that have the same values on the attributes in A.

Figure 1 gives an example where each user has two attributes gender and age. In the centralized setting, the server has access to the raw dataset Figure 1(a), from which, it can compute the full contingency table (Figure 1(b)). The two marginal tables (Figure 1(c,d)) can be computed from the contingency table.

3.2 Problem Definition: Local Setting

In the local setting, the aggregator does not have access to the raw dataset, such as the one shown in Figure 1(a). Instead, each user possesses one row of the dataset and sends a randomly perturbed value based on it. Our goal is to have the aggregator to use the perturbed reports to compute with reasonable accuracy any k-way marginal. Some methods (such as those proposed in [11]) require a specification of the maximum k ahead of time. Our proposed method can support queries of arbitrary k values.

To measure the utility empirically, we use *sum of squared error (SSE)*, i.e., the square of the L_2 distance between the true marginal M_A and the reconstructed T_A. When we query many k-way marginals, we calculate the SSE for each marginal, and use the average SSE as the indicator of a method's accuracy.

The reconstructed T_A can be viewed as a random variable since random noises are added in the process to satisfy LDP. When a method is able to produce an unbiased estimation, the expected value of T_A is the true marginal M_A, and the expected value of SSE is the variance of the random variable T_A.

Figure 1 gives an example where each user has two attributes gender and age. The goal is to construct all the marginal tables. Each user's private value corresponds to a row in Figure 1(a). No one has

Symbol	Description
n	The total number of users
v^j	Value of user j
d	Number of attributes
$\mathbb{A}$	The set of all attributes
a_i	Attribute i
c_i	Number of possible values for attribute a_i
F	The full contingency table
A	Some set of attributes
M_A	The marginal table of attribute set A
m	Number of marginal tables output by our method
ℓ	Size of each marginal table in our method

Table 1: List of Notations

the full view of the whole dataset. To construct the marginal tables Figure 1(c,d), one can let each user report the two values (using an FO as described earlier), aggregate the users' reports to construct the full contingency table (with some noise), and build the marginal tables. This method is more formally described in the following.

See Table 1 for the list of notations.

3.3 Full Contingency Table Method (FC)

To estimate M, one straightforward approach is to estimate the full contingency table F first, and then construct M from F. We call this approach the Full Contingency (FC) table method. In this method, each user reports her value $v \in D$ using an FO protocol. The aggregator estimates the frequency of each value in the full domain. Once having the full contingency table, the aggregator can compute any k-way marginal.

The main shortcoming of FC is that, since one has to query the frequency of each value in the full domain of all attributes, the time complexity and space complexity grows exponentially with the number of attribute d and can be prohibitively expensive.

Furthermore, even when it is feasible to construct the full contingency table, computing marginals from a noisy full contingency table can have high variance. For example, suppose we have 32 binary attributes, the domain size is thus 2^{32}. When constructing a 4-way marginal, each value in the 4-way marginal is the result of summing up 2^{28} noisy entries in the full contingency table. Let Var_0 be the variance of estimating each single cell in the full contingency table, the variance of each cell in the reconstructed marginal is then $2^{28} \times \mathsf{Var}_0$, and the expected SSE is $2^4 \times 2^{28} \times \mathsf{Var}_0 = 2^{32} \times \mathsf{Var}_0$. In general, the variance of computing k-way marginals from the noisy full contingency table is

$$\mathsf{Var}_{\mathsf{FC}} = 2^d \cdot \mathsf{Var}_0 \tag{4}$$

3.4 All Marginal Method (AM)

To mitigate the exponential dependency on d, one can construct all the k-way marginals directly. There are two alternatives, one is to divide the privacy budget ϵ into $\binom{d}{k}$ pieces, and have each user reports $\binom{d}{k}$ times, once for each k-way marginal. The second is to divide the user population into $\binom{d}{k}$ disjoint groups, and have users in each group report one k-way marginal. Under the LDP setting, it is generally better to divide the population than dividing the privacy budget, because reporting under low privacy budget is very noisy [26, 37, 38].

Under LDP, estimating fraction frequencies is less accurate with a smaller group than with a larger group, because the noises have larger impact when the true counts are small. The variance is inversely proportional to the group size. Thus dividing the population into $\binom{d}{k}$ groups will add a $\binom{d}{k}$ factor to the variance. This factor results in the following variance.

$$\mathsf{Var}_{\mathsf{AM}} = 2^k \cdot \binom{d}{k} \cdot \mathsf{Var}_0 \tag{5}$$

When k is relatively small (and hence $\binom{d}{k}$ is small), AM performs better than FC; when k is large, AM could perform worse than FC. Another limitation of this method is that one has to specify the value k ahead of time. After the protocol is executed, there is no way to answer any t-way marginal queries for $t > k$.

3.5 Fourier Transformation Method (FT)

Fourier Transformation (FT) was used for publishing k-way marginals in the centralized setting [5]. Kulkarni et al. [11] applied the technique to the local setting. Effectively, it is an optimization of the AM method. The motivation underlying FT is that, the calculation of a k-way marginal requires only a few coefficients in the Fourier domain. Thus, users can submit noisy Fourier coefficients that are needed to compute the desired k-way marginals, instead of values in those marginals.

This method results in slightly lower variance than AM. However, in order to reconstruct all k-way marginals, a large number of coefficients need to be estimated; thus this method would still perform poorly when k is large. Furthermore, the method is designed to deal with the binary attributes. Therefore, the non-binary attributes must be pre-processed to binary attributes, resulting in more dimensions. For example, an attribute with c values has to be transformed into $\lceil \log_2 c \rceil$ binary attributes.

The details of FT are presented in Appendix A.1. Here, we briefly analyze its variance. Specifically, there are $\sum_{s=0}^{k} \binom{d}{s}$ coefficients to be estimated. Estimating $\mathsf{T}_A(v)$ requires information for a selected set of 2^k coefficients, each multiplied by 2^{-k}. Therefore, this method has variance

$$\mathsf{Var}_{\mathsf{FT}} = \sum_{s=0}^{k} \binom{d}{s} \cdot \mathsf{Var}_0 \tag{6}$$

3.6 Expectation Maximization Method (EM)

This method allows each user to upload the value for each attribute separately with split privacy budget. The aggregator then conducts Expectation Maximization (EM) algorithm to reconstruct the marginal tables. This approach is first introduced by Fanti et al. [17] for estimating joint distribution for two attributes, and then generalized by Ren et al. [31] to handle multiple attributes.

Specifically, denote $y^j = \langle y_1^j, y_2^j, \ldots, y_d^j \rangle$ as the report from user j. The algorithm attempts to guess the private value distribution T_A, for any A, by maximizing the probability y^j are reported from user j.

The original EM algorithm runs slowly. Therefore, we use the algorithm proposed in the appendix of [38] to help compute T_A. In most cases, if the initial values are set using the result returned by this algorithm, the EM algorithm finishes quickly. Specifically, this algorithm first estimates the value distribution for any single attribute, and then uses that estimation to estimate distribution for any pair of attributes, and so on. The method is proven to produce unbiased estimation.

The detailed protocol of EM is given in Appendix A.2. Overall, the EM method has the advantage of being able to compute t-way marginals for any t. But since ϵ is split into each attribute, this method has large variance.

4 CALM: CONSISTENT ADAPTIVE LOCAL MARGINAL

In this section, we describe our proposed method CALM (Consistent Adaptive Local Marginal) for publishing k-way marginal via LDP. Our method is inspired by the PriView method for publishing marginal under the centralized DP setting [29], so we describe PriView first.

4.1 An Overview of PriView

The PriView method was designed for privately computing arbitrary k-way marginals for a dataset with d binary attributes in the centralized setting. PriView privately publishes a synopsis of the dataset. Using the synopsis, it can reconstruct any k-way marginal. The synopsis takes the form of m size-ℓ marginals that are called *views*. Below we give an overview of the PriView method, using an example where there are $d = 8$ attributes $\{a_1, a_2, \cdots, a_8\}$, and we aim to answer all 3-way marginals. PriView has the following four steps. (See [29] for complete specification of PriView.)

Choose the Set of Views. The first step is to choose which marginals to include in the private synopsis as views. That is, one needs to choose m sets of attributes. PriView chooses these sets so that each size-2 (or size-3) marginal is covered by some view. For example, if aiming to cover all 2-way marginals, then one could choose the following $m = 6$ sets of attributes to construct views:

$$\{a_1, a_2, a_3, a_4\} \quad \{a_1, a_5, a_6, a_7\} \quad \{a_2, a_3, a_5, a_8\}$$
$$\{a_4, a_6, a_7, a_8\} \quad \{a_2, a_3, a_6, a_7\} \quad \{a_1, a_4, a_5, a_8\}$$

Observe that any pair of two attributes are included in at least one set.

Generate Noisy Views. In this step, for each of the m attribute sets, PriView constructs a noisy marginal over the attributes in the set, by adding Laplace noise $\mathsf{Lap}\left(\frac{m}{\epsilon}\right)$ to each cell in the marginal table. This is the only step that needs direct access to the dataset. After this step, the dataset is no longer accessed.

Consistency Step. Given these noisy marginals/views, some 3-way marginals can be directly computed. For example, to obtain the 3-way marginal for $\{a_1, a_2, a_3\}$, we can start from the view for $\{a_1, a_2, a_3, a_4\}$ and marginalizes out a_4. However, many 3-way marginal are not covered by any of the 6 views. For example, if we want to compute the marginal for $\{a_1, a_3, a_5\}$, we have to rely on partial information provided by the 6 views. We can compute the marginals for $\{a_1, a_3\}$, $\{a_1, a_5\}$, and $\{a_3, a_5\}$, and then combine them to construct an estimation for $\{a_1, a_3, a_5\}$.

Observe that $\{a_1, a_5\}$ can be computed both by using the view for $\{a_1, a_5, a_6, a_7\}$ and by using the view for $\{a_1, a_4, a_5, a_8\}$. Since independent noises are added to the two marginals, the two different ways to compute marginal for $\{a_1, a_5\}$ most likely have different results. In addition, the noisy marginals may contain negative values. PriView performs constrained inference on the noisy marginals to ensure that the marginals in the synopsis are all non-negative and mutually consistent. (For self-containment, we included the description of the consistency step in Appendix A.3.)

Generating k-way Marginals. From the m consistent views, one can reconstruct any k-way marginals. When given a set of k attributes, if all k attributes are included in one view, then we can compute the k-way marginal directly. When no view includes all k attributes, PriView uses Maximum Entropy estimation to compute the k-way marginal. For example, when given the marginals for $\{a_1, a_3\}$, $\{a_1, a_5\}$, and $\{a_3, a_5\}$, Maximum Entropy estimation finds among all possibles marginals for $\{a_1, a_3, a_5\}$ that are consistent with the three known marginals, the one with the maximum entropy. Note that while the marginal for $\{a_1, a_3, a_5\}$ have 7 unknowns (the 8 cells must sum up to 1), and each marginal over $\{a_1, a_3\}$, $\{a_1, a_5\}$, and $\{a_3, a_5\}$ gives 3 equations, these equations are not independent. In this case, the three 2-way marginals together give 6 independent linear constraints on the 7 unknowns, leaving one degree of freedom.

Discussions. Using the PriView method, one could answer k-way marginals for arbitrary k values. For a k-way marginal computed by PriView, there are two sources of errors. *Noise Errors* are due to the Laplacian noises added to satisfy DP. *Reconstruction Errors* are due to the fact that one has to estimate a k-way marginal from partial information.

Two important algorithmic parameters affect the magnitude of these two kinds of errors. They are the number m of marginals/views in the synopsis, and the size ℓ (i.e., number of attributes) of these views. With a larger ℓ, the views cover more combinations of attributes, reducing Reconstruction Errors. However, one would be summing over more noisy entries to compute any marginal, increasing the Noise Errors. Similarly, a larger m means more marginals and better coverage of combinations of attributes, which reduces Reconstruction Errors. However, a larger m also means less privacy budget for each marginal and higher Noise Errors. Consider the running example with 8 attributes, by using 14 (instead of 6) size-4 marginals, one can ensure that any set of 3 attributes is covered by at least one of the marginals, eliminating Reconstruction Errors. However, this is done at the cost of adding noises sampled from $\mathsf{Lap}\left(\frac{14}{\epsilon}\right)$ instead of $\mathsf{Lap}\left(\frac{6}{\epsilon}\right)$ to each cell. Note that even if any set of 3 attributes is covered, answering 4-way marginals will still have Reconstruction Errors.

Analysis in [29] shows that the choice of optimal ℓ (size of each marginal) is independent from parameters such as dataset size n, privacy parameter ϵ, and dimensionality d. In particular, setting ℓ to be around 8 works well. The optimal choice of m (number of marginals), however, depends on n, ϵ, d, and the nature of the

dataset. In [29], m is chosen to fully cover either all 2-way marginals or all 3-way marginals, using the concept of covering design [18, 19].

4.2 Overview of the CALM Method

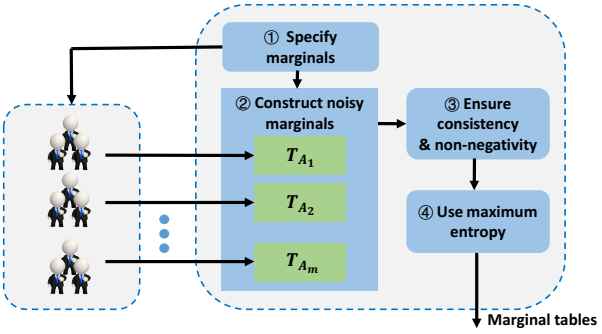


Figure 2: Illustration of CALM. The users to the left are partitioned into groups. The aggregator to the right first specifies the marginals to all the users and aggregate the reports for each marginal table. Then the aggregator process the data to publish the final results.

Ideas in the PriView method inspire the CALM method. In the LDP setting, we cannot compute a noisy marginal by adding Laplace noise to the true marginal, and we need to use FO protocols to do so. In PriView, data from all users are used for computing each of the m views, and the privacy budget is split into m equal portions. But in the local setting, several previous work pointed out that by partitioning users into groups, and having each group use the full privacy budget, the overall error will be smaller [26, 37, 38]. We adopt this design principle and split the user population into groups, with reports from users in each group to estimate one marginal.

Figure 2 illustrates how CALM works. The aggregator first chooses a set of m marginals and the FO protocol to be used (e.g., GRR or OUE). The choice of whether to use GRR or OUE is determined by the number of cells in each marginal, because GRR is more accurate for domains of smaller size and OUE is better for larger domains. CALM adaptively chooses which of GRR and OUE to use, based on ϵ and the domain size for the marginals.

The aggregator then assigns each user to one of the marginals, and informs the user which marginal she should report. How this assignment is done is outside CALM. The aggregator can randomly partition the population into m groups of approximately the same size, and assigns users in one group to each marginal. Alternatively, the aggregator can use *public* information of the users (such as IP addresses) to help ensure that each group is representative of the overall population. It is also possible that the aggregator sends information of all m marginals to each user, having each user randomly select one and report on that marginal.

Each user projects her private value v onto the marginal she is reporting and reports the projected value of v via FO. On receiving users' reports, the server uses the aggregation algorithm of FO to obtain the noisy marginal tables. Then the server processes the data via the consistency and reconstruction steps to obtain the final results, as in the PriView method.

One main challenge is how to choose the set of m marginals, and in particular the parameters ℓ (size of each marginal) and m (number of marginals). The analysis for PriView in [29] is no longer valid for the local setting. We discuss this in Section 4.3. In addition, we want to deal with non-binary attributes, which we discuss in Section 4.4.

4.3 Choosing the Set of Marginals

The most important algorithmic parameters for CALM are the *marginal size* ℓ, i.e., the number of attributes in each marginal, and the marginal number m, i.e., the number of different marginals. For ease of analysis, we assume all marginals are of equal size and receive equal number of users to contribute.

Similar to PriView, there are Noise Errors, which are caused by the addition of noises in the FO protocol, and Reconstruction Errors, which are caused by the fact that a k-way marginal may not be covered by any of the chosen marginal, and has to be estimated using the Maximum Entropy principle.

CALM has one additional source of errors that do not exist in PriView. CALM splits the user population into groups, and uses the marginal of one group as an estimation of the marginal of the whole population. Errors may be caused by the fact that the marginal of one randomly selected group is not representative of that for the whole population. We call these *Sampling Errors*. We analyze these errors below.

Noise Errors. To understand Noise Errors, we analyze the total variance of estimating 1-way marginals when they are included in at least one selected marginal, and how they are affected by the choice of m and ℓ. For each ℓ-way marginal table, there are $\frac{n}{m}$ users reporting it. By Equation (3), the variance for each cell is inversely proportional to the group size used to estimate it. More specifically, we have:

$$\mathrm{Var}_c = \min\left(\frac{4e^\epsilon}{(e^\epsilon-1)^2}, \frac{L-2+e^\epsilon}{(e^\epsilon-1)^2}\right)\cdot\frac{m}{n}$$

Here L is the number of cells in one marginal, and an ℓ-way marginal with binary attributes has $L = 2^\ell$ cells. Note that when each attribute has different number of possible values, L is the expected number of cells in one marginal.

To construct a 1-way marginal from such an ℓ-way marginal, each cell of the 1-way marginal is the summation of some cells from the larger (ℓ-way) marginal. By linearity of variances, the variance for any 1-way marginal is $\mathrm{Var}_1 = \mathrm{Var}_c \cdot L$.

The above shows that increasing m adds a linear factor to the variance. However, increasing m also causes a 1-way marginal to be included more times. When a 1-way marginal is included t times, we can obtain t estimations of the 1-way marginal, one from each size-ℓ marginal that includes it. Averaging these t estimations reduces the variance by a factor of t. More specifically, each size-ℓ marginal includes ℓ attributes. Therefore, in expectation, the information of each attribute will be contributed from $\frac{m\cdot\ell}{d}$ ℓ-way marginals. The

average of these estimates are therefore

$$\begin{aligned}\mathsf{NE}(n,d,\epsilon,\ell) &= \frac{\mathsf{Var}_1}{\frac{m\cdot\ell}{d}}\\ &= \min\left(\frac{4e^\epsilon}{(e^\epsilon-1)^2}, \frac{L-2+e^\epsilon}{(e^\epsilon-1)^2}\right)\cdot\frac{m}{n}\cdot L\cdot\frac{d}{m\cdot\ell}\\ &= \min\left(\frac{4e^\epsilon}{(e^\epsilon-1)^2}, \frac{L-2+e^\epsilon}{(e^\epsilon-1)^2}\right)\cdot\frac{L}{\ell}\cdot\frac{d}{n}\end{aligned} \tag{7}$$

The key observation here is that the magnitude of Noise Errors does not depend on m, which is different from PriView. It does depend on ℓ and ϵ, where ϵ affects the first term, which is the variance of the FO protocol. The parameter ℓ affects both the term $\frac{L}{\ell}$ and the variance for the FO protocol.

Also note that when we estimate k-way marginals based on the estimation of marginals of the k attributes, the estimation is affected by the errors for each of the k attributes, we thus use $k\cdot\mathsf{NE}(n,d,\epsilon,\ell)$ as the Noise Errors when we optimize for a particular k value.

Reconstruction Errors. Reconstruction Errors occur when a k-way marginal is not covered by any of the chosen marginal. The magnitude of Reconstruction Errors depends on to what extent attributes are correlated. If all attributes are mutually independent, then Reconstruction Errors do not exist. When attributes are dependent, the general trend is that larger m and larger ℓ will cover more combination of attributes, reducing reconstruction errors. The reduction effect of Reconstruction Errors diminishes as m increases. For example, if all k-ways marginals are already fully covered, Reconstruction Errors are already 0 and cannot be further decreased. Even if not all k-ways marginals are fully covered, increasing m beyond some reasonably large number will only cause diminishing return. Since Reconstruction Errors are dataset dependent, there is no formula for estimating them.

Sampling Errors. Sampling Errors occur when a marginal in a group of users deviates from the marginal in the whole population. The parameter ℓ has no impact on Sampling Errors. However, increasing m would cause each group size $\frac{n}{m}$ to be smaller, raising Sampling Errors. When computing a marginal from a group of $s = n/m$ users, each cell in the marginal can be viewed as the sum of s independent Bernoulli random variables, divided by s. In other words, each cell is a binomial random variable divided by s. Thus each cell has variance $\frac{\mathsf{M}_A(v)(1-\mathsf{M}_A(v))}{s}$, where $\mathsf{M}_A(v)$ is the fraction of users with value v in the whole population. The Sampling Errors for an ℓ-way marginal A are thus

$$\sum_{v\in V_A}\frac{\mathsf{M}_A(v)(1-\mathsf{M}_A(v))}{s} = \frac{m\times\sum_{c\in V_A}\mathsf{M}_A(v)(1-\mathsf{M}_A(v))}{n}$$

Since $\sum_{v\in V_A}\mathsf{M}_A(v) = 1$, we have $\sum_{v\in V_A}\mathsf{M}_A(v)(1-\mathsf{M}_A(v)) < \sum_{v\in V_A}\mathsf{M}_A(v)\cdot 1 = 1$. Thus the Sampling Errors are simply bounded by

$$\mathsf{SE}(n,m) = \frac{m}{n} \tag{8}$$

Choosing m and ℓ. Both m and ℓ affect Reconstruction Errors. In addition, m affects Sampling Errors, and ℓ affects Noise Errors. Intuitively, we want to choose m and ℓ to minimize the maximum of the three kinds of Errors, since the maximum would dominate the overall errors. However, we do not have a formula to estimate Reconstruction Errors, which is dataset dependent.

We propose to choose a *target error threshold* θ, which serves as a rough estimation of Reconstruction Errors when they are not zero, and choose m and ℓ as follows:

- Compute the largest marginal size ℓ_u, such that $k\cdot\mathsf{NE} < \theta$.
- When $\ell_u < k$, one chooses ℓ_u and the largest m such that $\mathsf{SE} < \theta$.
- Otherwise, one chooses m and $\ell_t \in [k, \ell_u]$ such that the maximum of NE and SE is minimized.

While θ intends to be a rough estimation of Reconstruction Errors, it does not need to be chosen based on one particular dataset. One can run experiments with a public dataset of similar nature under different parameters, the best level of SSE that can be achieved is usually a good indicator of the magnitude of Reconstruction Errors. When a public dataset is unavailable, one can generate a synthetic dataset under some correlation assumption and run experiments. In experiments conducted for this paper, we choose $\theta = 0.001$, and use it for all datasets and settings.

Algorithm 1 Pseudocode to determine m and ℓ

Require: Dataset parameters n, d, ϵ, k, error threshold θ.
Ensure: m and ℓ.

procedure INFERENCE(n, d, ϵ, θ)
  Assign $m_u \leftarrow \theta\cdot n$, $\ell_u \leftarrow 2$
  while $k\cdot\mathsf{NE}(n,d,\epsilon,\ell_u+1) \le \theta$ **do**
    Increment $\ell_u \leftarrow \ell_u + 1$
  if $\ell_u < k$ **then**
    return $\min(m_u, \binom{d}{\ell_u}), \ell_u$
  Assign $\ell_b \leftarrow \ell_u$
  while $\ell_b > k$ and CoverDesign$(d,k,\ell_b-1) \le m_u$ **do**
    Decrement $\ell_b \leftarrow \ell_b - 1$
  if $\ell_b == \ell_u$ **then**
    return $\min(m_u, \binom{d}{\ell_u}), \ell_u$
  Assign $\mathsf{E} \leftarrow 1, m \leftarrow m_u, \ell \leftarrow \ell_u$
  for ℓ_t in $[\ell_b, \ell_u]$ **do**
    Assign $m_t \leftarrow$ CoverDesign(d,k,ℓ_t)
    if $\max(\mathsf{SE}(n,m_t), k\cdot\mathsf{NE}(n,d,\epsilon,\ell_t)) < \mathsf{E}$ **then**
      Update $\mathsf{E} \leftarrow \max(\mathsf{SE}(n,m_t), k\cdot\mathsf{NE}(n,d,\epsilon,\ell_t))$
      Update $m \leftarrow m_t, \ell \leftarrow \ell_t$
  return m, ℓ

Algorithm 1 gives the pesudocode for determining m and ℓ. The algorithm uses the formula to calculate Noise Errors NE from (7), and Sampling Errors SE as in (8). CoverDesign is an external procedure to calculate the number of ℓ-way marginals that can fully include all k-way marginals. Note that NE is for a single attribute; one can multiply NE by k to approximate the Noise Errors for the k-way marginals.

For example, Figure 3 gives the Noise Errors times k (i.e., $k\cdot\mathsf{NE}$) for $n = 2^{16}, d = 8$, and $k = 3$ when ϵ ranges from 0.2 to 2.0. If we fix $\theta = 10^{-3}$, we can read from the figure that when $\epsilon \le 1.4$, only $\ell = 2$ can be used. Because larger ℓ will make NE even larger; and we choose to allow some RE to exist. When ϵ is larger, e.g.,

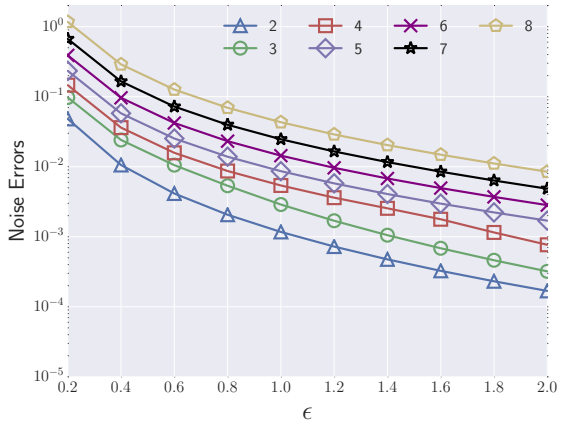


Figure 3: Noise Errors times k when $n = 2^{16}, d = 8, k = 3$.

$\epsilon = 2.0$, NE is already very small that we can tolerate more NE to eliminate RE. In this case, both $\ell = 3$ and $\ell = 4$ will give an $k \cdot \text{NE} < \theta$, so the goal is to choose an ℓ so that the maximum of $k \cdot \text{NE}$ and SE is minimized. Specifically, when $\ell = 3$, CoverDesign gives $m = \binom{8}{3} = 56$ to cover all the 3-way marginals, rendering max(NE = 0.00032, SE = 0.00085) = 0.00085; when $\ell = 4$, CoverDesign gives $m = 14$ (meaning that 14 4-way marginals suffice to cover all 3-way marginals when $d = 8$), thus giving max(NE = 0.00076, SE = 0.00021) = 0.00076. Thus $\ell = 4$ and $m = 14$ is used.

4.4 Consistency between Noisy Marginals

When different marginals have some attributes in common, those attributes are actually estimated multiple times. Utility will increase if these estimates are utilized together. Specifically, assume a set of attributes A is shared by s marginals, $A_1, A_2, \ldots, A_s$. That is, $A = A_1 \cap \ldots \cap A_s$. Now we can obtain s copies of T_A by summing from cells in each of the T_A's, i.e., $\mathsf{T}_{A_i}(v) = \sum_{v' \in V_{A_i}, v'_A = v_A} \mathsf{T}_{A_i}(v')$.

To obtain a better estimation of T_A, we use the weighted average of T_{A_i} for all marginal A_i. That is,

$$\mathsf{T}_A(v) = \sum_i w_i \cdot \mathsf{T}_{A_i}(v).$$

Since each T_{A_i} is unbiased, their average $\mathsf{T}_A(v)$ is also unbiased. To determine the distribution of the weights, the intuition is to put more weights to the more accurate estimations. Specifically, we minimize the variance of $\mathsf{T}_A(v)$, i.e., $\text{Var}[\mathsf{T}_A(v)] = \sum_i w_i^2 \cdot \text{Var}\left[\mathsf{T}_{A_i}(v)\right] = \sum_i w_i^2 \cdot C_i \cdot \text{Var}_0$, where C_i is the number of cells from A_i that contribute to A, i.e., $C_i = |\{v' : v' \in V_{A_i}, v'_A = v_A\}|$, and Var_0 is the basic variance for estimating a single cell (we assume each marginal has a similar amount of users, but the analysis can be easily changed to different number of users). Formally, we have the following problem:

$$\begin{array}{ll} \text{minimize} & \sum_i w_i^2 \cdot C_i \\ \text{subject to} & \sum_i w_i = 1 \end{array}$$

According to KKT condition [22, 23], we can derive the solution: Define $L = \sum_i w_i^2 \cdot C_i + \mu \cdot (\sum_i w_i - 1)$, by taking the partial derivative of L for each of w_i, we have $w_i = -\frac{\mu}{2C_i}$. The value of μ can be solved by the equation $\sum_i w_i = 1$. As a result, $\mu = -\frac{2}{\sum_i \frac{1}{C_i}}$, and $w_i = \frac{\frac{1}{C_i}}{\sum_i \frac{1}{C_i}}$. Therefore, the optimal weighted average is

$$\mathsf{T}_A(v) = \frac{\sum_i \frac{1}{C_i} \cdot \mathsf{T}_{A_i}(v)}{\sum_i \frac{1}{C_i}}$$

Once the accurate T_A is obtained, all T_{A_i}'s can be updated. For any marginal A_i, we update all $v' \in V_{A_i}$ using the result of v where $v \in VA$ and $v'_A = v_A$. Specifically,

$$\mathsf{T}_{A_i}(v') \leftarrow \mathsf{T}_{A_i}(v') + \frac{1}{C_i}\left(\mathsf{T}_A(v) - \mathsf{T}_{A_i}(v)\right)$$

The remaining reconstruction operations are borrowed from PriView and described in Appendix A.3. After that, one can obtain the k-way marginals.

4.5 Discussion

We claim that CALM satisfies ϵ-LDP because all the information from each user to the server goes through an FO with ϵ as privacy budget, and no other information is leaked.

Although CALM is inspired from PriView, there are several differences between the two. Among the differences, many are because the two methods work under different privacy requirements. That is, PriView works in the centralized setting of differential privacy, while CALM works in the local setting. We summarize the differences as follows.

- In PriView, all the information are accessible to the server. The server operates on the dataset, adds noise, and then derive the answers. On the other hand, in CALM, each user sends noisy information to the server, who aggregates the reports, and then calculate the answers.
- CALM can handle non-binary datasets, while PriView is designed to handle only binary attributes.
- In PriView, each view is estimated through the information of all users, with split privacy budget. While in the local setting, it is known that it is better to partition users into groups. Therefore, in CALM, each marginal is estimated by only a group of users.
- Because of the above, CALM faces Sampling Errors, in addition to Noise Errors and Reconstruction Errors.
- In PriView, the number of marginals is critical and is dependent on the dataset. On the other hand, CALM is much less sensitive to the number of views (marginals).
- In PriView, the optimal view size does not depends on ϵ, and is around 8. However, in CALM, view size affects which FO protocol to be used and depends on ϵ.

5 EVALUATION

We use experiments to empirically evaluate the effectiveness of our proposed method CALM, and to verify our analysis.

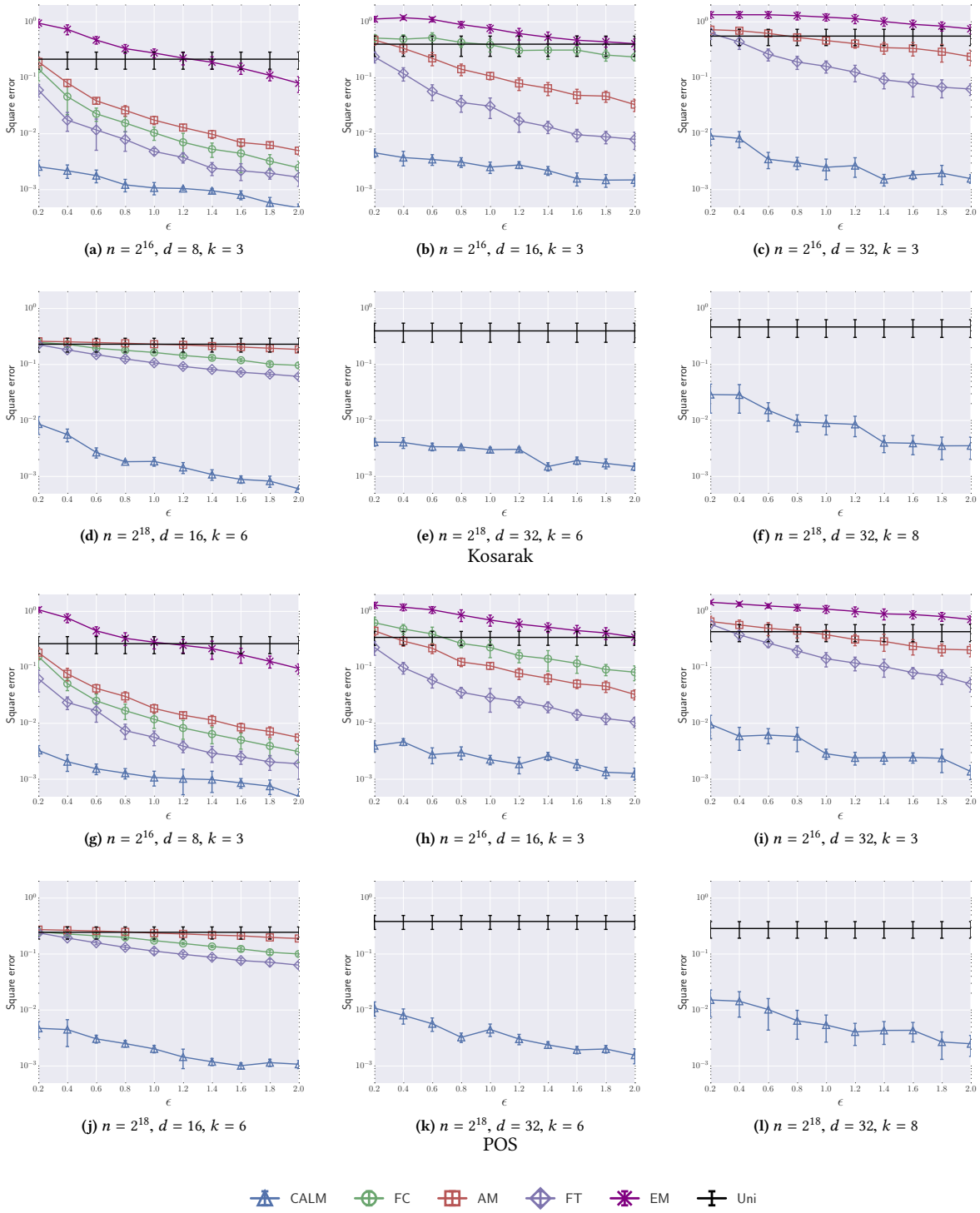


Figure 4: Comparison of different methods on binary datasets. We only plot the methods that are scalable in each setting, Uni **method is a baseline method. Results are shown in log scale.**

5.1 Experimental Setup

Our experimental setup is largely influenced by that in [11], which introduced the Fourier Transformation method and ran extensive comparisons of several methods for this problem.

Environment. All algorithms are implemented in Python 3.5 and all the experiments are conducted on a PC with Intel Core i7-4790 3.60GHz and 16GB memory.

Datasets. We run experiments on the following four datasets.

- POS [43]: A dataset containing merchant transactions of half a million users.
- Kosarak [2]: A dataset of click streams on a Hungarian website that contains around one million users.
- Adult [4]: A dataset from the UCI machine learning repository. After removing missing values, the dataset contains around 50 thousands records. The numerical attributes are bucketized into categorical attributes.
- US [32]: A dataset from the *Integrated Public Use Microdata Series* (IPUMS). It has around 40k records of the United States census in 2010.

The first two are transactional datasets where each record contains some items. We treat each item as a binary attribute. Thus these two datasets are binary. When running experiments with k binary attributes, we pre-process a dataset to include only the top d most frequent items. The later two are non-binary datasets, i.e., each attribute contains more than two categories.

Evaluation Methodology. To evaluate the performance of different methods, the *Sum of Squared Error* (SSE) of the marginals is reported. That is, we compute the ground truth and calculate the sum of squared difference in each cell. For each dataset and each method, we choose 50 random k-way marginal queries and measure their SSE. This procedure is repeated 20 times, with result mean and standard deviation reported.

Competitors. The FC, AM, and EM methods can be directly applied. For a fair comparison, the FO used in those methods are also chosen adaptively.

The FT method is unable to deal with the non-binary attributes. Therefore, we implement the non-binary version of FT by encoding each non-binary attribute into several binary attributes.

As a baseline comparison, we also plot the SSE of the Uniform method (Uni in the figures), which always returns a uniform distribution for any marginal tables. Clearly, if the performance of one method is worse than the Uniform method, the marginal constructed from that method is meaningless.

Experimental Settings. Different methods scale differently with respect to d, the number of attributes, and k, the size of marginals. Also, the error depends on n, the size of the dataset. We use three values of d: 8, 16, and 32. We consider $k = 3$ for all three settings of d. We consider $k = 6$ only for $d \in \{16, 32\}$, and $k = 8$ only for $d = 32$. This is because a larger k value makes more sense with a larger d value.

We consider two dataset sizes $n = 2^{16}$ and $n = 2^{18}$, which were used in [11]. Since all methods benefit similarly when n increases, the comparison results remain valid for other n sizes.

The settings for m and ℓ are given in Table 2 in the appendix.

5.2 SSE on Binary Datasets

Figure 4 illustrates the results for comparing CALM against existing methods we discussed in Section 3 on two binary datasets Kosarak and POS.

In all settings, CALM significantly outperforms all existing algorithms, and the advantage of CALM increases for larger d and larger k values, and for smaller ϵ values. For most settings, the difference between CALM and FT, the closest competitor, is between one and two orders of magnitude. When ϵ is small, e.g., when $\epsilon = 0.2$, all existing algorithms perform close to the Uniform baseline, meaning they can provide very little information when the privacy budget is small. Whereas CALM can still provide enough information even for very small ϵ. Furthermore, many methods simply do not scale to the case of $d = 32$.

EM performs poorly, in fact it is often worse than the Uniform baseline. This is because EM requires each user to report information on all d attributes, in order to perform inference. This means dividing the privacy budget by d, which results in large perturbation. The other methods can split the population into groups, instead of splitting privacy budget, thus performing better. Also, when k is larger than 5, the computation time for EM method is too long to run efficiently (about 20 minutes each query). We thus do not plot EM for the $k = 6, 8$ cases.

Among the competitors, FT performs the best. When $d = 8, k = 3$, we can compute the variance for FC, AM and FT using Formulas (4), (5), and (6). The results are $256 \cdot \mathrm{Var}_0$ for FC, $448 \cdot \mathrm{Var}_0$ for AM, and $93 \cdot \mathrm{Var}_0$ for FT. From Figures 4a and 4g, we can see that the experimental results match the analytical comparison.

For $d = 16$, CALM's performance is similar to the case of $d = 8$. Other methods, however, have significantly larger error. For example, in Figure 4b, when $\epsilon = 0.2$, the squared error of CALM is 0.0055, which is 41 times better than the state-of-the-art method, i.e., FT with squared error of 0.2266.

The performance of FC does not depends on k, since it constructs a full contingency table.

When $d = 32$, most of the existing methods are unable to scale, especially when $k = 8$. For the AM method, the number of possible marginals are $\binom{32}{8} = 10518300$. As a result, the average number of users that contribute information to each marginal is less than one when we choose $n = 2^{16}$ and 2^{18}. Similarly, the number of Fourier coefficients required to reconstruct 8-way marginals are $\sum_{s=1}^{8} \binom{32}{s} = 15033173$, resulting less than one user contributes to each coefficient.

5.3 SSE on Non-binary Datasets

The experimental results for non-binary datasets, i.e., Adult and US, are shown in Figure 5. To reduce computational complexity, we pre-process all attributes to contain at most 3 categories.

The experimental results show the superiority of CALM, which achieves around 1 to 2 orders magnitude of improvement over existing methods.

By comparing the $d = 8$ and $k = 3$ setting in Figure 4 with Figure 5, we observe that FT performs better than FC and AM in the binary datasets, whereas performs worse in the non-binary datasets. The bad performance in the non-binary datasets is due to the binary encoding process, which dramatically increases the

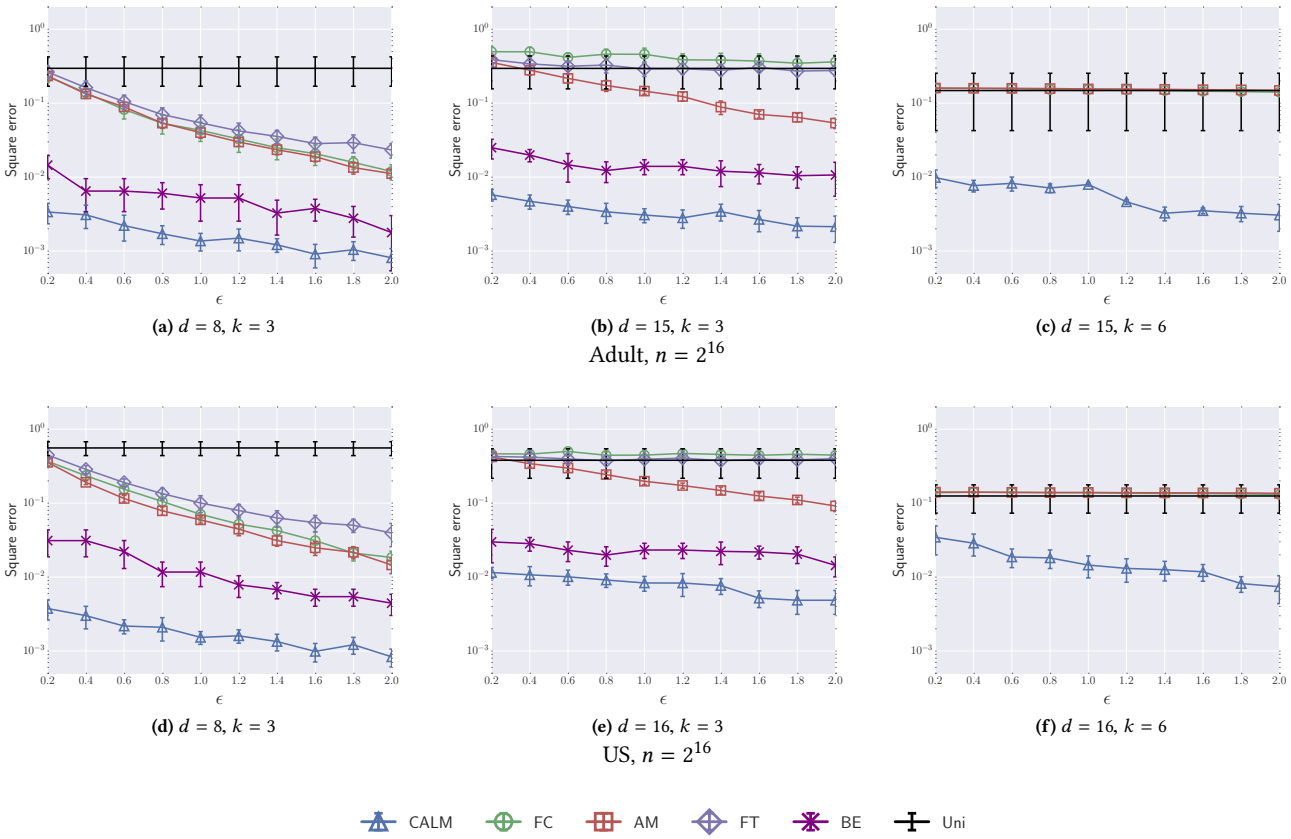


Figure 5: Comparison of different methods in two non-binary datasets. We only plot the methods that are scalable in each setting, Uni **method is a baseline method,** BE **method is the binary encoding version of** CALM**. Results are shown in log scale.**

number of Fourier coefficients required to reconstruct marginals. Considering the case where all the 8 attributes have 3 categories, d and k becomes 16 and 6 after the binary encoding. By variance analysis, the variance becomes $14893 \cdot \text{Var}_0$, which is much larger than $93 \cdot \text{Var}_0$ in the binary datasets.

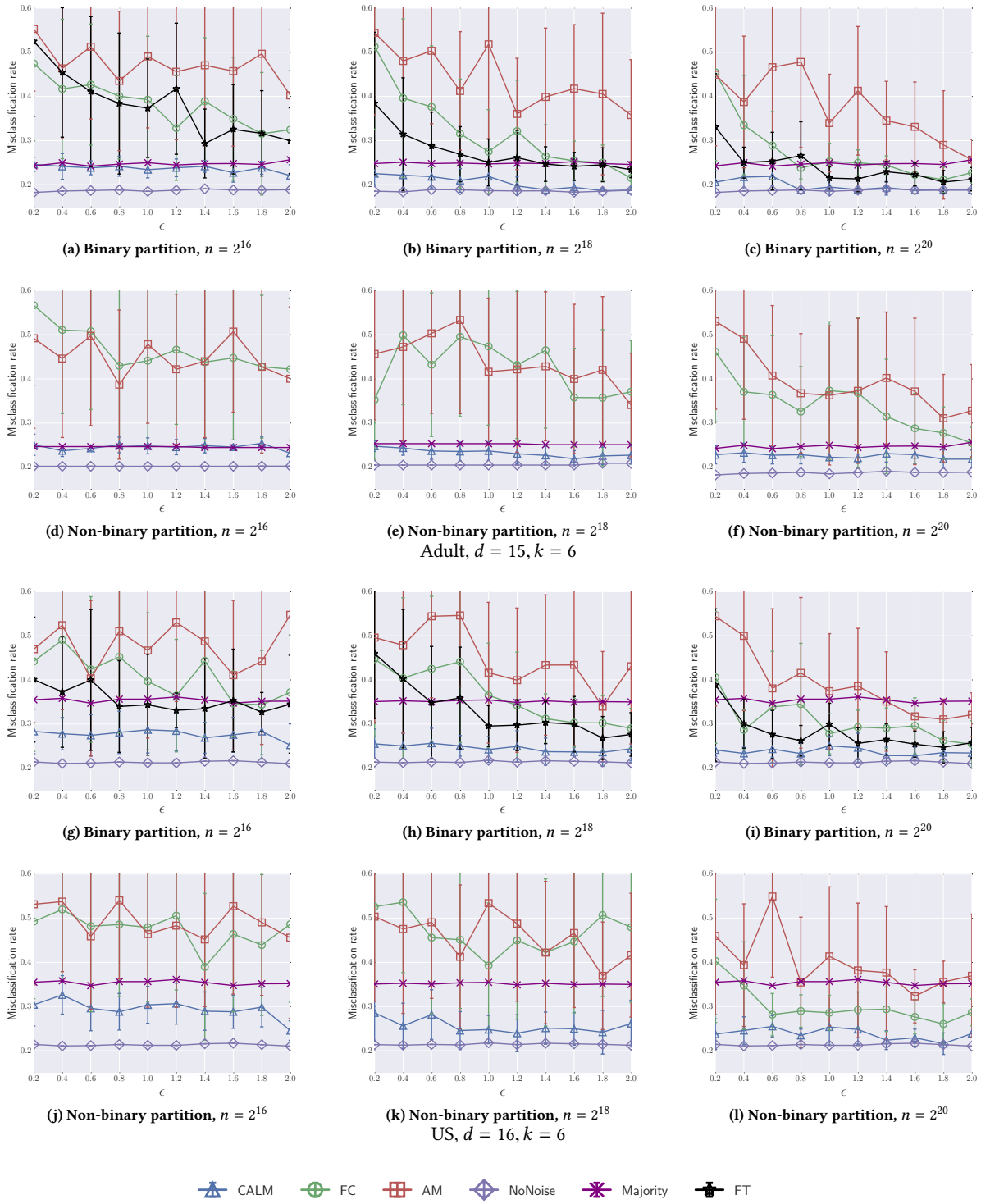
To demonstrate the impact of handling non-binary attributes in CALM, we also utilize the idea of binary encoding to implement CALM, to which the consistent step of PriView can be directly applied. We call the binary encoding version of CALM the BE method. We observe that the CALM performs better than BE. The reason is that d and k becomes very large after binary encoding, which increase the variance. When $d = 16, k = 6$, the BE takes too much time in the Maximum Entropy estimation step. Thus, we do not plot BE in this case.

5.4 Classification Performance

To demonstrate the practical utility of the proposed method, we train the SVM classifiers using the Adult and US datasets. The goal of the classifier is to predict whether a user's annual income is above $50k$.

To train the model, we pick five attributes (features) and have each method output the 6-way marginals (five features plus the annual income label). The features for the Adult dataset are age, workclass, education, education-num, and occupation, as features; and the features for the US dataset are WRKRECAL (informed of work recall), GRADEATT (grade level attending), SCHLTYPE (public or private school), SCHOOL (school attendance), DIFFPHYS (ambulatory difficulty). Note that we pick features by their semantic relationships to the label. After the noisy marginal is obtained, a synthetic dataset is generated based on this marginal. The synthetic dataset is then used to train the SVM classifier. There are also two baselines: NoNoise represents the method without enforcing ϵ-LDP, it is the best case to aim for (using the same set of attributes). Majority represents the naive method that blindly predict the label by the majority label. All the methods are evaluated following the typical process, where 80% of the records are sampled as training set, and the other 20% are used as the testing set. And we evaluate its utility by the *misclassification rate* on the testing set, i.e., the fraction of records in the testing set that are incorrectly classified.

Figure 6 illustrates the misclassification rate of SVM classifier trained by different methods. It is shown that in most cases, the average misclassification rate of CALM is close to NoNoise. When ϵ is small, the classification model trained by FC and AM is not



(a) Binary partition, $n = 2^{16}$ (b) Binary partition, $n = 2^{18}$ (c) Binary partition, $n = 2^{20}$

(d) Non-binary partition, $n = 2^{16}$ (e) Non-binary partition, $n = 2^{18}$ (f) Non-binary partition, $n = 2^{20}$

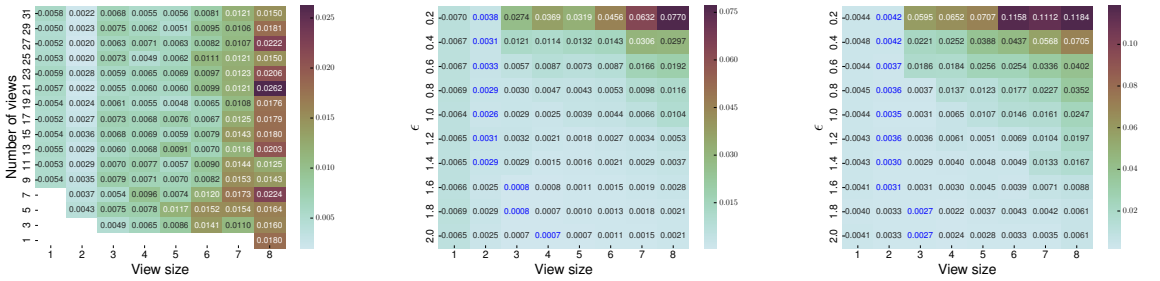
Adult, $d = 15, k = 6$

(g) Binary partition, $n = 2^{16}$ (h) Binary partition, $n = 2^{18}$ (i) Binary partition, $n = 2^{20}$

(j) Non-binary partition, $n = 2^{16}$ (k) Non-binary partition, $n = 2^{18}$ (l) Non-binary partition, $n = 2^{20}$

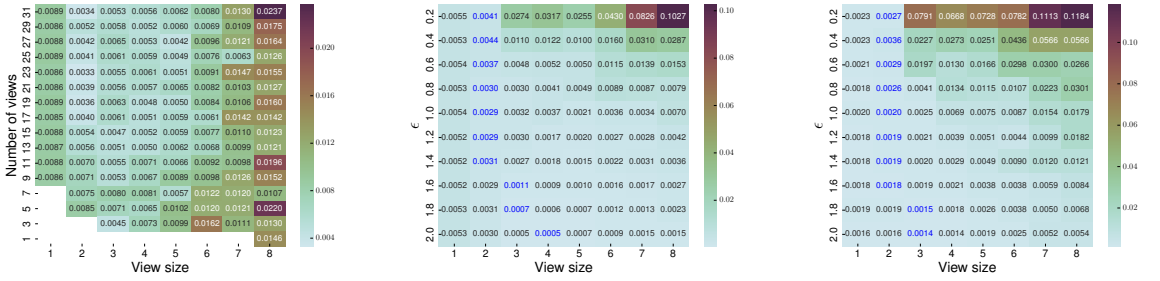
US, $d = 16, k = 6$

Figure 6: Comparison on classification performance. We only plot the methods that are scalable in each setting. NoNoise **is the baseline where no noise is added;** Majority **is the naive method to always answer the majority label.**



(a) ℓ vs m, $d = 8$, $k = 3$, $\epsilon = 0.6$ (b) ℓ vs ϵ, $n = 2^{16}$, $d = 8$, $k = 3$, $m = 16$ (c) ℓ vs ϵ, $n = 2^{16}$, $d = 16$, $k = 3$, $m = 16$

POS, $n = 2^{16}$



(d) ℓ vs m, $d = 8$, $k = 3$, $\epsilon = 0.6$ (e) ℓ vs ϵ, $d = 8$, $k = 3$, $m = 16$ (f) ℓ vs ϵ, $d = 16$, $k = 3$, $m = 16$

Kosarak, $n = 2^{16}$

Figure 7: Mutual effects of marginal size ℓ, number of marginals m and the privacy budget ϵ.

better than Majority (some times even worse than 50%, which is even worse than random guess, and thus useless).

In the right two columns of Figure 6, we duplicate the datasets 4 times and 16 times to boost the accuracy. It can be seen that more users will help with accuracy. For example, for the binary case of the Adult dataset, the accuracy of CALM is almost optimal when $\epsilon = 1.4$ when the dataset is duplicated 4 times; while when the dataset is duplicated 16 times, the accuracy of CALM is almost optimal when $\epsilon = 0.8$. We also observe that when the dataset is non-binary (in the even rows of Figure 6), the performance is slightly worse than if the dataset is binary. This is because in the non-binary setting, there are more possible values in each attribute, thus making the result worse.

5.5 Verifying Marginal Parameters

In Figure 7, we use heatmaps to illustrate the impact of marginal size ℓ, number of marginals m and the privacy budget ϵ on the squared error.

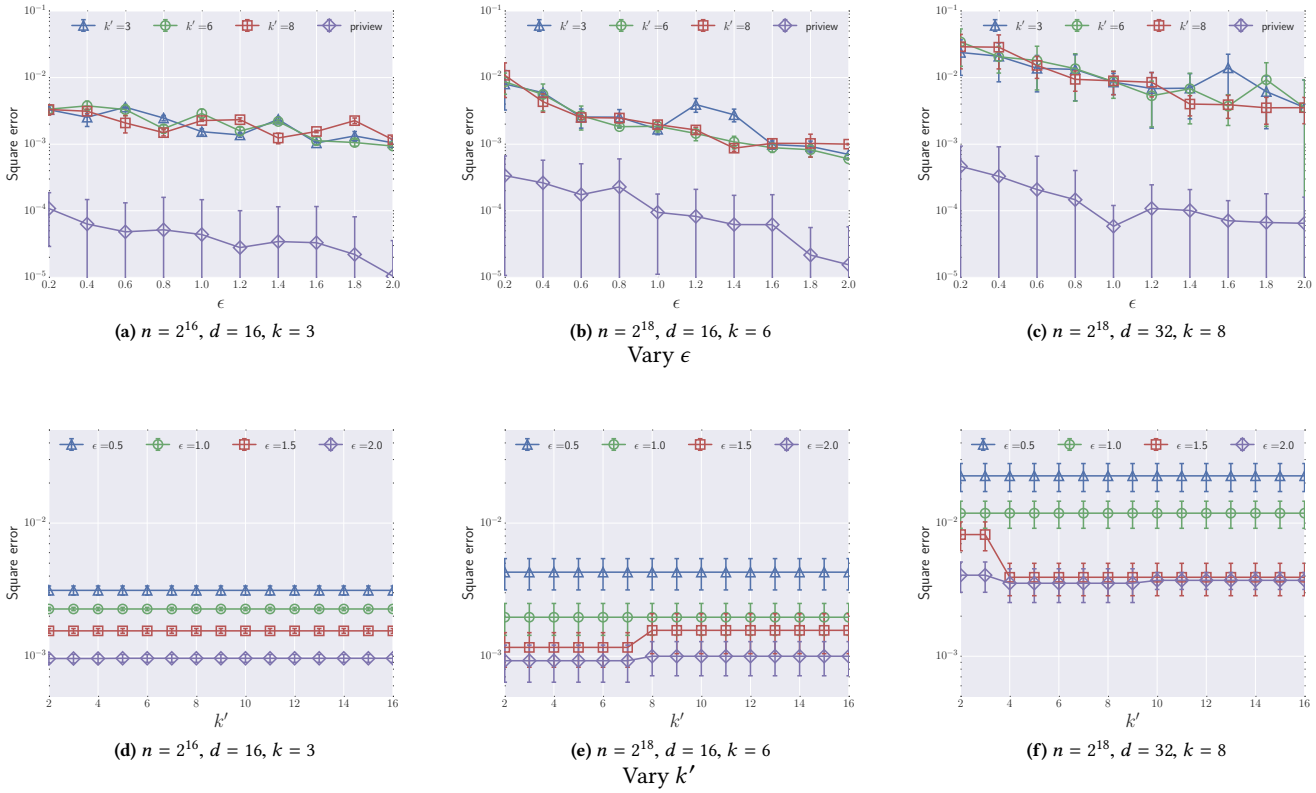
Figure 7a and 7d shows the mutual effect of ℓ and m on POS and Kosarak, respectively. This is for the setting of $d = 8, k = 3, \epsilon = 0.6$. The two heatmaps show that when ℓ is fixed to a value other than 1 and 8, increasing m will gradually decrease the error, which is in accordance with the analysis in Section 4.3, as increasing m leads to covering more marginals, thus reducing Reconstruction Errors. While increasing m increases Sampling Errors, the level of Sample Errors even when $m = 32$ is around $2^{-11} = 0.0005$. Note that when $\ell = 1$, each marginal includes a single attribute, increasing m does not reduce Reconstruction Errors. Similarly, when $\ell = 8$ all 8 attributes are already covered in any marginal; increasing m thus doesn't change the error.

Figure 7b, 7c, 7e, and 7f show the mutual effect of ℓ and ϵ when m is fixed. We observe that when ϵ is small, it is better to choose smaller ℓ. The reason is that in this case the noise dominates the error; thus we should choose smaller ℓ to reduce noise. When ϵ is large, lager ℓ is preferred since the effect of Reconstruction Errors is dominant. The blue numbers show the squared errors under the optimal setting through analysis, which is approximately approach to the experiment results.

5.6 Impact of k and the Local Setting

Figure 8 serves two purposes. One is to study the accuracy of CALM when one chooses parameters m and ℓ that are optimized for k', but the query is for k-way marginals, where $k \neq k'$. This is interesting to know because one may want to support k-way marginals for different k values. The other is to compare the accuracy of CALM with the centralized setting of PriView, to understand how much utility one is giving up for the enhanced privacy of the local setting.

The first row of Figure 8 plots the effect of answering k-way marginals when optimized for $k' \in \{3, 6, 8\}$ and when using centralized PriView. When $k = 3$ (the left sub-figure), different settings of k' perform similarly. When $k = 6$ (the middle sub-figure), optimizing for $k' = 3$ clearly is worse when $\epsilon = 1.2$ and 1.4. This



(a) $n = 2^{16}$, $d = 16$, $k = 3$ (b) $n = 2^{18}$, $d = 16$, $k = 6$ (c) $n = 2^{18}$, $d = 32$, $k = 8$

Vary ϵ

(d) $n = 2^{16}$, $d = 16$, $k = 3$ (e) $n = 2^{18}$, $d = 16$, $k = 6$ (f) $n = 2^{18}$, $d = 32$, $k = 8$

Vary k'

Figure 8: Kosarak dataset. Using m and ℓ optimized for different k'.

is because when optimizing for $k' = 3$, increasing ϵ from 1 to 1.2 causes ℓ to increase from 2 to 3, because it is estimated that for 3-way marginals, the noise error when going to $\ell = 3$ is sufficiently low at $\epsilon = 1.2$. See Table 2 for the parameters chosen under different configurations. However, when optimizing for $k' = 6$, the change of ℓ from 2 to 3 occurs when $\epsilon = 1.6$. This also happens when $k = 8$ (the right sub-figure), where the setting of $k' = 3$ performs bad when $\epsilon = 1.6$ and 1.8. Overall, we generally see the best result when $k = k'$. Also, it appears that if one is unsure about k, the size of query marginals, one should optimize for a slightly larger k' value.

From first row of Figure 8, we also see that PriView performs one to two magnitudes better than CALM. This is mainly because much less noise is needed in the centralized setting. Theoretically, the amount of noise added in the local setting is $\Theta\left(\frac{1}{\sqrt{n}}\right)$, while in the centralized setting, the amount of noise is $\Theta\left(\frac{1}{n}\right)$.

The second row of Figure 8 plots the effect of answering k-way marginals while optimizing for a broader range of k' values (from 2 to 16). We consider $\epsilon \in \{0.5, 1.0, 1.5, 2.0\}$. When a setting results in choosing the same pair of m, ℓ as another configuration, we reuse the accuracy number in the plot instead of running the experiments again; thus any difference in a line is due to changes in m, ℓ. We observe that when ϵ is small (i.e., $\epsilon \in \{0.5, 1.0\}$) the same parameters ($m = 65, \ell = 2$, to be precise) are chosen no matter which k' one is optimizing for. Also there is little difference in accuracy when computing $k = 3$. When $\epsilon = 1.5$ and $k = 6$, optimizing for $k' \geq 8$ is sub-optimal. In the right sub-figure, we observe that to compute $k = 8$-way marginals, optimizing for $k' \leq 3$ results in worse accuracy (especially for $\epsilon = 1.5$).

Figure 9 shows SSE for different k-way marginals fixing $n = 2^{18}$ and $d = 16$. For $\epsilon \in \{0.5, 1, 1.5, 2\}$, we plot results for two settings: m and ℓ optimized for $k' = 3$; and m and ℓ optimized for $k' = k$. We found that in most cases, the results for the two settings are similar, because the m and ℓ settings are the same. For $\epsilon = 1.5$, when $k > 7$, the difference becomes significant. This is mainly because the ℓ values are different: when $k' = 3$, $\ell = 3$ by Algorithm 1; but when $k' = k \in \{8, 9, 10\}$, $\ell = 2$ by Algorithm 1.

6 RELATED WORK

Differential privacy has been the *de facto* notion for protecting privacy. In the centralized settings, many DP algorithms have been proposed (see [15, 35] for theoretical treatments and [24] in a more practical perspective). Recently, Uber has deployed a system enforcing DP during SQL queries [21], Google also proposed several works that combine DP with machine learning, e.g., [28].

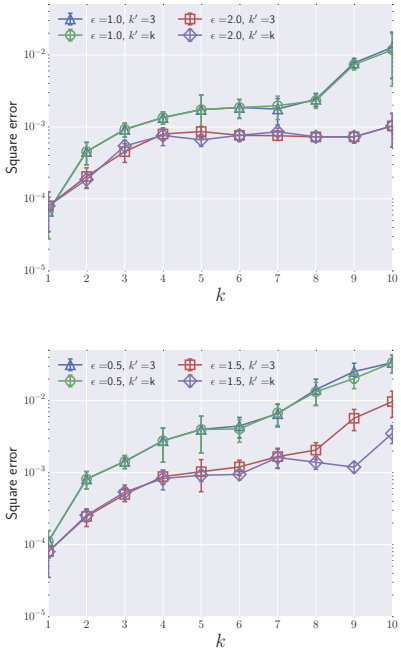


Figure 9: Kosarak dataset, $n = 2^{18}, d = 16$.

In the local setting, we have also seen real world deployment: Google deployed RAPPOR [16] as an extension within Chrome; Apple [1] uses similar methods to help with predictions of spelling and other things; Microsoft also deployed an LDP system for telemetry collection [12].

Of all the problems, one basic mechanism in LDP is to estimate frequencies of values. For this problem, several mechanisms have been proposed [6, 7, 16]. Wang et al. compare different mechanisms using estimation variance [37], and conclude that when the domain size is small, the Generalized Random Response provides best utility, and Optimal Local Hash (OLH)/Optimal Unary Encoding (OUE) [37] perform better when the domain is large.

The problem of marginal release is a classic application of histogram estimation, investigated by [11, 31]. These methods have been examined in Section 3. Note that the problem has also been investigated in the centralized setting [10, 29, 41], but the techniques cannot be directly used in this setting, because there is no centralized aggregator that has the overall view of all users' data, and the noise in the local setting is significantly larger (dependent on $\sqrt{n}$ instead of sensitivity of the function).

Besides the marginal release problem, there are other problems in the LDP setting that rely on mechanisms for frequency estimation. The problem of finding heavy hitters in a very large domain was exhaustively investigated [6–8, 17, 20, 25, 34, 38]. The frequent itemset mining problem is to identify the frequent set of items that appear simultaneously where each user has a set of items. Qin et al. proposed LDPMiner [30] that finds the frequent singletons; Wang et al. [39] improved LDPMiner by proposing SVIM for singleton mining and SWSM for itemset mining. The problem of empirical risk minimization also attracts lots of investigation, both from the theoretically [33, 36, 42] and empirically [26].

7 CONCLUSIONS

In this paper, we propose LDP protocols to construct marginals for high-dimensional attributes. Instead of directly generating all marginal tables, we propose to strategically choose sets of attributes, with which we can reconstruct all k-way marginals. To mitigate the effect of noise, we adaptively choose randomization algorithm based on the marginal size. We then consist all marginals to provide more accurate estimation. Extensive experiments on real-world datasets are conducted to illustrate the superiority over the current state of the art.

ACKNOWLEDGMENT

We would like to thank the sherperd Prof. Ting Yu and the anonymous reviewers for their helpful comments. The work is supported by the NSF 1640374, NSFC 61429301, NSFC U1401253 and Zhejiang Natural Science Foundation under grant No. LR16F020001.

REFERENCES

[1] Apple differential privacy team, learning with privacy at scale. Available at https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf.
[2] Frequent itemset mining dataset repository. Available at http://fimi.ua.ac.be/data/.
[3] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang. Deep learning with differential privacy. In *Proceedings of CCS*, pages 308–318. ACM, 2016.
[4] A. Asuncion and D. Newman. UCI machine learning repository, 2010.
[5] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of PODS*, pages 273–282. ACM, 2007.
[6] R. Bassily, K. Nissim, U. Stemmer, and A. Thakurta. Practical locally private heavy hitters. *arXiv:1707.04982*, 2017.
[7] R. Bassily and A. Smith. Local, private, efficient protocols for succinct histograms. In *Proceedings of Symposium on Theory of Computing*, pages 127–135. ACM, 2015.
[8] M. Bun, J. Nelson, and U. Stemmer. Heavy hitters and the structure of local privacy. *arXiv:1711.04740*, 2017.
[9] R. Chen, H. Li, S. Kasiviswanathan, and H. Jin. Private data aggregation framework for untrusted servers, Oct. 12 2017. US Patent App. 15/282,776.
[10] R. Chen, Q. Xiao, Y. Zhang, and J. Xu. Differentially private high-dimensional data publication via sampling-based inference. In *Proceedings of SIGKDD*, pages 129–138. ACM, 2015.
[11] G. Cormode, T. Kulkarni, and D. Srivastava. Marginal release under local differential privacy. In *Proceedings of SIGMOD*, pages 131–146. ACM, 2018.
[12] B. Ding, J. Kulkarni, and S. Yekhanin. Collecting telemetry data privately. In *Advances in Neural Information Processing Systems*, pages 3574–3583, 2017.
[13] C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.
[14] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284. Springer, 2006.
[15] C. Dwork and A. Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.
[16] Ú. Erlingsson, V. Pihur, and A. Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of CCS*, pages 1054–1067. ACM, 2014.
[17] G. Fanti, V. Pihur, and Ú. Erlingsson. Building a rappor with the unknown: Privacy-preserving learning of associations and data dictionaries. *Proceedings on Privacy Enhancing Technologies*, 2016(3):41–61, 2016.
[18] D. Gordon. La jolla covering repository. Available at https://www.ccrwest.org.
[19] D. M. Gordon, O. Patashnik, and G. Kuperberg. New constructions for covering designs. *Journal of Combinatorial Designs*, 3(4):269–284, 1995.
[20] J. Hsu, S. Khanna, and A. Roth. Distributed private heavy hitters. In *International Colloquium on Automata, Languages, and Programming*, pages 461–472. Springer, 2012.
[21] N. Johnson, J. P. Near, and D. Song. Practical differential privacy for sql queries using elastic sensitivity. *arXiv preprint arXiv:1706.09479*, 2017.
[22] W. Karush. Minima of functions of several variables with inequalities as side constraints. *M. Sc. Dissertation. Dept. of Mathematics, Univ. of Chicago*, 1939.

[23] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Traces and emergence of nonlinear programming*, pages 247–258. Springer, 2014.

[24] N. Li, M. Lyu, D. Su, and W. Yang. *Differential Privacy: From Theory to Practice*. Synthesis Lectures on Information Security, Privacy, and Trust. Morgan Claypool, 2016.

[25] N. Mishra and M. Sandler. Privacy via pseudorandom sketches. In *Proceedings of PODS*, pages 143–152. ACM, 2006.

[26] T. T. Nguyên, X. Xiao, Y. Yang, S. C. Hui, H. Shin, and J. Shin. Collecting and analyzing data from smart device users with local differential privacy. *arXiv:1606.05053*, 2016.

[27] N. Papernot, M. Abadi, Ú. Erlingsson, I. Goodfellow, and K. Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv:1610.05755*, 2016.

[28] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Ú. Erlingsson. Scalable private learning with pate. In *ICLR*, 2018.

[29] W. Qardaji, W. Yang, and N. Li. Priview: practical differentially private release of marginal contingency tables. In *Proceedings of SIGMOD*, pages 1435–1446. ACM, 2014.

[30] Z. Qin, Y. Yang, T. Yu, I. Khalil, X. Xiao, and K. Ren. Heavy hitter estimation over set-valued data with local differential privacy. In *Proceedings of CCS*, pages 192–203. ACM, 2016.

[31] X. Ren, C.-M. Yu, W. Yu, S. Yang, X. Yang, J. A. McCann, and S. Y. Philip. Lopub: High-dimensional crowdsourced data publication with local differential privacy. *IEEE Transactions on Information Forensics and Security*, 13(9):2151–2166, 2018.

[32] S. Ruggles, J. T. Alexander, K. Genadek, R. Goeken, M. B. Schroeder, and M. Sobek. Integrated public use microdata series: Version 5.0 [machine-readable database], 2010.

[33] A. Smith, A. Thakurta, and J. Upadhyay. Is interaction necessary for distributed private learning? In *Proceedings of Symposium on Security and Privacy (SP)*, pages 58–77. IEEE, 2017.

[34] A. G. Thakurta, A. H. Vyrros, U. S. Vaishampayan, G. Kapoor, J. Freudiger, V. R. Sridhar, and D. Davidson. Learning new words, Mar. 14 2017. US Patent 9,594,741.

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

[36] D. Wang, M. Gaboardi, and J. Xu. Efficient empirical risk minimization with smooth loss functions in non-interactive local differential privacy. *arXiv:1802.04085*, 2018.

[37] T. Wang, J. Blocki, N. Li, and S. Jha. Locally differentially private protocols for frequency estimation. In *Proceedings of USENIX*. USENIX Association, 2017.

[38] T. Wang, N. Li, and S. Jha. Locally differentially private heavy hitter identification. *arXiv:1708.06674*, 2017.

[39] T. Wang, N. Li, and S. Jha. Locally differentially private frequent itemset mining. In *Proceedings of the Symposium on Security and Privacy,*, page 578–594. IEEE, 2018.

[40] S. L. Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[41] C. Xu, J. Ren, Y. Zhang, Z. Qin, and K. Ren. Dppro: Differentially private high-dimensional data release via random projection. *IEEE Transactions on Information Forensics and Security*, 12(12):3081–3093, 2017.

[42] K. Zheng, W. Mou, and L. Wang. Collect at once, use effectively: Making non-interactive locally private learning possible. *arXiv:1706.03316*, 2017.

[43] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *Proceedings of SIGKDD*, pages 401–406. ACM, 2001.

A SUPPLEMENTARY METHOD DESCRIPTIONS

A.1 Details of FT

Define $b(\cdot)$ as the function that transforms a d-dimensional value v into an integer, where $b(v) = \sum_{i=1}^{d} 2^{d-i} \cdot v_{a_i}$. The Fourier transformation aims to project $\mathbf{e}_{b(v)}$, the standard basis of $b(v)$, onto the Fourier basis. Specifically, denote all the Fourier basis as a $2^d \times 2^d$ matrix $\Omega = \{\omega_{ij}\}$, where $\omega_{ij} = 2^{-d/2}(-1)^{\langle i|j\rangle}$, and $\langle i|j\rangle$ is the inner product of i and j in their binary representations. Each user j reports one bit of the local coefficient $(\Omega \mathbf{e}_{b(v^j)})$ at location i, using *randomized response*. By aggregating all users' reports into the noisy Fourier coefficients θ, the aggregator in fact is estimation $\Omega \cdot \mathsf{F}$, where F is the full contingency table. The marginal tables can then be calculated from θ:

$$\mathsf{T}_A(v) = \sum_{\alpha \in V_{[d]}, \alpha_{[d]\setminus A} = \mathbf{0}} \theta_{b(\alpha)} \cdot \left(\sum_{\eta \in V_{[d]}, \eta_A = v_A} \omega_{b(\alpha), b(\eta)} \right) \quad (9)$$

Note that in the formula above, fixing α, for any η, $\omega_{b(\alpha),b(\eta)}$ is the same. This is because α fixes the bits in positions not contained in A to be all zeros, while η enumerates all of these bits. As a result, to calculate Equation (9), one only needs to enumerate all α's.

The advantages of this method lies in that it only needs to access $\sum_{j=0}^{k} \binom{d}{j}$ Fourier coefficients. For variance, each coefficient is multiplied by $2^{-d/2} \cdot 2^{d/2-k}$. Moreover, each $\mathsf{T}_A(v)$ is the summation of 2^k coefficients.

A.2 Details of EM

For any k-way marginal A_i, it can be estimated via the following formula:

$$\mathsf{T}_{A_i}(v) = \frac{C(v) - \sum_{A \subset A_i} \left[\mathsf{T}_A(v) q^{|A_i|-|A|} (p-q)^{|A|} \right]}{(p-q)^{|A_i|}}$$

where $C(v)$ denotes the fraction of users that has value v, i.e., $C(v) = \frac{j : y^j_{A_i} = v_{A_i}}{n}$. Note that to calculate T_{A_i}, one should get T_A for all $A \subset A_i$ first. The base case is $\mathsf{T}_\emptyset = 1$. The results are then used as the initial values of the EM algorithm, i.e., $\Pr[v]_0 = \mathsf{T}_{A_i}(v)$.

The EM algorithm has two parts, the E step and the M step. In the E step, the likelihood is computed as

$$\Pr\left[v|y^j\right]_t = \frac{\Pr[v]_t \cdot \Pr\left[y^j|v\right]}{\sum_v \Pr[v]_t \cdot \Pr\left[y^j|v\right]},$$

where $\Pr[v]_t$ denote the probability for v in round t, and $\Pr[v]_0$ is initialized to $\frac{1}{|V_A|}$. Then $\Pr[v]_t$ is updated in the M step,

$$\Pr[v]_{t+1} = \frac{1}{n} \sum_{j=1}^{n} \Pr\left[v|y^j\right]_t$$

until $\max_v |\Pr[v]_{t+1} - \Pr[v]_t| \le \delta$ for some $\delta > 0$. This procedure eventually converges to the local maximum of the log-likelihood function.

A.3 Other Details of PriView

We provide the remaining consistency and reconstruction steps for the PriView method. Note that these steps are also used in CALM.

Overall Consistency. We can conduct the following procedure to achieve overall consistency. First, enumerate all the subsets of $\mathbb{A}$. These subsets form a partial order under the subset relation, which can be organized as a topological graph. This topological graph starts from the empty set. Then, for each subset of A in the topological order, we ensure the consistency among the marginals that include this subset. It is shown in [29] that following the topological order, a later consistency step will not invalidate consistency established in previous steps.

Non-Negativity through Ripple. We propose to adopt the following "Ripple" non-negativity method, which turns negative

d, k, n \ ϵ	0.2	0.4	0.6	0.8	1	1.2	1.4	1.6	1.8	2
$8, 3, 2^{16}$	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	3, 56	3, 56	4, 14
$8, 4, 2^{16}$	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	3, 56	3, 56	3, 56
$8, 5, 2^{16}$	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	3, 56	3, 56
$8, 6, 2^{16}$	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	3, 56	3, 56
$8, 7, 2^{16}$	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	3, 56
$8, 8, 2^{16}$	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	3, 56
$16, 3, 2^{16}$	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	3, 65	3, 65
$16, 4, 2^{16}$	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	3, 65
$16, 5, 2^{16}$	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65
$16, 6, 2^{16}$	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65
$16, 7, 2^{16}$	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65
$16, 8, 2^{16}$	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65
$32, 3, 2^{16}$	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65
$32, 4, 2^{16}$	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65
$32, 5, 2^{16}$	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65
$32, 6, 2^{16}$	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65
$32, 7, 2^{16}$	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65
$32, 8, 2^{16}$	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65	2, 65
$8, 3, 2^{18}$	2, 28	2, 28	2, 28	2, 28	3, 56	3, 56	3, 56	3, 56	3, 56	4, 14
$8, 4, 2^{18}$	2, 28	2, 28	2, 28	2, 28	3, 56	3, 56	4, 70	4, 70	4, 70	4, 70
$8, 5, 2^{18}$	2, 28	2, 28	2, 28	2, 28	2, 28	3, 56	3, 56	4, 70	5, 56	5, 56
$8, 6, 2^{18}$	2, 28	2, 28	2, 28	2, 28	2, 28	3, 56	3, 56	4, 70	4, 70	5, 56
$8, 7, 2^{18}$	2, 28	2, 28	2, 28	2, 28	2, 28	3, 56	3, 56	3, 56	4, 70	5, 56
$8, 8, 2^{18}$	2, 28	2, 28	2, 28	2, 28	2, 28	2, 28	3, 56	3, 56	4, 70	4, 70
$16, 3, 2^{18}$	2, 120	2, 120	2, 120	2, 120	2, 120	3, 262	3, 262	4, 140	4, 140	4, 140
$16, 4, 2^{18}$	2, 120	2, 120	2, 120	2, 120	2, 120	2, 120	3, 262	3, 262	4, 262	4, 262
$16, 5, 2^{18}$	2, 120	2, 120	2, 120	2, 120	2, 120	2, 120	3, 262	3, 262	4, 262	4, 262
$16, 6, 2^{18}$	2, 120	2, 120	2, 120	2, 120	2, 120	2, 120	2, 120	3, 262	3, 262	4, 262
$16, 7, 2^{18}$	2, 120	2, 120	2, 120	2, 120	2, 120	2, 120	2, 120	3, 262	3, 262	4, 262
$16, 8, 2^{18}$	2, 120	2, 120	2, 120	2, 120	2, 120	2, 120	2, 120	3, 262	3, 262	3, 262
$32, 3, 2^{18}$	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	3, 262	3, 262	4, 262
$32, 4, 2^{18}$	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	3, 262	3, 262	3, 262
$32, 5, 2^{18}$	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	3, 262	3, 262
$32, 6, 2^{18}$	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	3, 262	3, 262
$32, 7, 2^{18}$	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	3, 262
$32, 8, 2^{18}$	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	2, 262	3, 262

Table 2: The setting of ℓ and m in the runtime, calculated by Algorithm 1. Each cell is of the format (ℓ, m), and each row represents the values for the same d, k, and n setting.

counts into 0 while decreasing the counts for its neighbors to maintain the overall count constant. Specifically, given an k-way marginal table T_A, for any $v \in V_A$ with $\mathsf{T}_A(v) < -\theta$, we set the entry to 0 and subtract $|c|/h$ from each of its h neighboring cells, defined as the cells obtained by changing one of the attributes' category to other categories, and h is determined by the number of categories of each attribute in A. However, this procedure may make the count of other cells to be $c < -\theta$, the procedure iterates until no cell has count $c < -\theta$. As each iteration distributes a negative count into h neighbors, it is guaranteed to terminate quickly.

Applying the ripple non-negativity step to the marginals, however, may make them inconsistent. To resolve this problem, we run the consistency step after the non-negativity step several times.

Reconstructing k-way Marginals After the consisting phase, when all the attributes in A are "covered" by at least one marginal A_i, i.e. $A \subseteq A_i$, this step is trivial. We can construct T_A by summing over corresponding entries in T_{A_i}.

If A is not fully covered by any marginal A_i, we compute the k-way marginal table as the following optimization problem:

$$\begin{array}{ll} \text{maximize} & -\sum_{v \in V_A} \mathsf{T}_A(v) \cdot \log(\mathsf{T}_A(v)) \\ \text{subject to} & \underset{v \in V_A}{\forall} \mathsf{T}_A(v) \geq 0 \\ & \underset{A_i}{\forall} \underset{v \in V_{A_i \cap A}}{\forall} \mathsf{T}_{A_i}(v) = \mathsf{T}_A(v) \end{array}$$

Since all marginals are consistent, the value should be the same for all A_i. The above optimization problem can be solved by an off-the-shelf convex optimization tool.

A.4 Complexity Analysis

We give the time complexity, space complexity, and communication cost of different methods in Table 3. For ease of exposition, we assume all attributes are binary. Note that this can be easily converted to non-binary cases.

	Time	Space	Comm
CALM	$\Theta\left(n \cdot 2^\ell\right)$ or $\Theta\left(n + m \cdot 2^\ell\right)$	$\Theta\left(m \cdot 2^\ell\right)$	$\Theta\left(2^\ell\right)$
FC	$\Theta\left(n \cdot 2^d\right)$ or $\Theta\left(n + 2^d\right)$	$\Theta\left(2^d\right)$	$\Theta\left(2^d\right)$
AM	$\Theta\left(n \cdot 2^k\right)$ or $\Theta\left(n + \binom{d}{k} \cdot 2^k\right)$	$\Theta\left(2^k\right)$	$\Theta\left(2^k\right)$
FT	$\Theta\left(n + \binom{d}{k} \cdot 2^{2k}\right)$	$\Theta\left(\sum_{s=0}^{k} \binom{d}{s}\right)$	$\Theta(d)$
EM	$\Theta\left(n \cdot \sum_{s=0}^{k} \binom{d}{s} 2^s\right)$	$\Theta\left(\sum_{s=0}^{k} \binom{d}{s} 2^s\right)$	$\Theta(d)$

Table 3: Comparison of complexity for different methods. Server-side computation, server-side storage, and client-server communication are listed. All attributes are assumed to be binary.

Time Complexity: CALM is dominated by processing users' reports, which takes $\Theta\left(2^\ell\right)$ for each user, and $\Theta\left(n \cdot 2^\ell\right)$ in total. The similar arguments also hold for FC and AM methods, where user reports are basically vectors of size $\Theta\left(2^d\right)$ and $\Theta\left(2^k\right)$, respectively. Note that when ϵ is small and GRR is used in the above three methods, each user's report is one value (instead of a vector); thus one only needs to aggregate the reports, making the running time to $\Theta\left(n + m \cdot 2^\ell\right)$, $\Theta\left(n + 2^d\right)$, and $\Theta\left(n + \binom{d}{k} \cdot 2^k\right)$ for CALM, FC, and AM, respectively. For FT, since GRR is always used, the processing time is in the order of n. But the calculation of Equation (9) takes $\Theta\left(2^k\right)$ (for enumerating α). Therefore, to build $\binom{n}{k}$ marginal tables, each with 2^k values, the total computation takes $\Theta\left(n + 2^{2k} \cdot \binom{d}{k}\right)$. For EM, the running time is dominated by the counting operation. Specifically, the method first counts the number of users that reports a particular value for any single attribute, and then for any combination of values for any pair of attributes, and so on. There are $\sum_{s=0}^{k} \binom{d}{s} 2^s$ possible values that are needed to be counted; and one should count all of them for each user, making the resulting running time $\Theta\left(\sum_{s=0}^{k} \binom{d}{s} 2^s\right)$.

Space Complexity: We measure the memory needed assuming that all inputs and outputs are discarded. If the user reports and answers are to be stored, the same amount of space should be allocated for each method.

For the memory consumption, the CALM method needs to maintain the m ℓ-way marginal tables. For FC, the full contingency table is maintained for counting reports for each possible value, requiring $\Theta\left(2^d\right)$ storage. Note that one can sacrifice computation for storage by scanning through each user's report multiple times and directly construct the k-way marginal tables. But this will make the computation even more overwhelming. For FT, one can maintain do the similar: either maintain all the $2^k \cdot \binom{d}{k}$ coefficients or calculate the $\sum_{s=0}^{k} \binom{d}{s}$ marginals on time. Finally, for EM, a storage to count all possible intermediate results are needed.

Communication Overhead: The communication from each user to the server is the same as the report size. Note that since both FT and EM uses GRR as FO. The report in FT is only one bit, plus the index, which can be represented by d bits; and the report for EM is one bit for each of the d dimensions. Note that one can use OLH, presented in [37], instead of OUE. The OLH protocol is equivalent to OUE, with the communication overhead reduced to almost constant. But the disadvantage is that the server needs to do extra computation (i.e., evaluate hash functions) to retrieve the full report.