PriView: Practical Differentially Private Release of Marginal Contingency Tables

Wahbeh Qardaji
Dept. of Computer Science and CERIAS
Purdue University
wahbeh.qardaji@gmail.com

Weining Yang Ninghui Li
Dept. of Computer Science and CERIAS
Purdue University
{yang469,ninghui}@cs.purdue.edu

ABSTRACT

We consider the problem of publishing a differentially private synopsis of a d-dimensional dataset so that one can reconstruct any k-way marginal contingency tables from the synopsis. Marginal tables are the workhorses of categorical data analysis. Thus, the private release of such tables has attracted a lot of attention from the research community. However, for situations where d is moderate to large and k is beyond 3, no accurate and practical method exists. We introduce PriView, which computes marginal tables for a number of strategically chosen sets of attributes that we call views, and then use these view marginal tables to reconstruct any desired k-way marginal. In PriView, we apply maximum entropy optimization to reconstruct k-way marginals from views. We also develop a novel method to efficiently making all view marginals consistent while correcting negative entries to improve accuracy. For view selection, we borrow the concept of covering design from combinatorics theory. We conduct extensive experiments on real and synthetic datasets, and show that PriView reduces the error over existing approaches by 2 to 3 orders of magnitude.

Categories and Subject Descriptors

K.4.1 [**COMPUTERS AND SOCIETY**]: Privacy

Keywords

Differential Privacy; Marginal Table; Contingency Table

1. INTRODUCTION

In recent years, privacy-preserving data analysis has received a lot of attention in the research community, with the notion of differential privacy gradually becoming the de facto standard notion of privacy for this research area [10]. In this paper we consider the problem of computing a differentially private synopsis of a high-dimensional dataset so that one can reconstruct any low-dimensional marginal tables of the dataset. Marginal tables are the workhorses of categorical data analysis, e.g., data analysis in the medical, social and behavior sciences. In addition to being easy to interpret, they are sufficient statistics for popular classes of probabilistic models [5]. Because of this, they are the methods of choice for government agencies when releasing statistical summaries of categorical data. Marginal tables are essentially equivalent to OLAP (online analytical processing) cubes used in data management and analysis.

More specifically, we consider the following problem: Given a d-dimension binary dataset D, construct, in a differentially private way, a synopsis of D, so that one can relatively accurately compute the marginal table for any set of k attributes, referred to as k-way marginals. We assume that d, the number of dimensions, is large so that 2^d complexity is unfeasible. Given a k-way marginal obtained from the private synopsis, we measure its L_2 distance and Jensen-Shannon divergence from the true marginal, and aim at minimizing these error measures.

Marginal tables can be constructed directly by counting queries, which has been the center of focus for differentially private data analysis. Seminal work in differential privacy [6, 11] shows how to answer a set Q of counting queries when Q has at most on the order of $\sqrt{|D|}$ queries. Applying this approach, which we refer to as the Direct method, to our problem is to add independently generated noise to every k-way marginal table using $\epsilon/\binom{d}{k}$ as the privacy parameter. The L_2 error distance of the Direct method, however, is expected to be $O(d^k)$, which is large when d is not small (e.g., when $d \geq 32$) and k is beyond 3. A method of adding noise to Fourier coefficients is proposed in [4], which reduces the error slightly over the Direct method. The problem of generating marginals have been studied in a series papers in recent years [7, 15, 17, 16, 18, 28, 8, 22]; however, these methods do not scale when 2^d is large. In summary the current state of the art is that when 2^d-complexity is unfeasible, then the Fourier method in [4] is the best approach.

In this paper we improve this state of art by introducing PriView. There are two key insights behind PriView. First, rather than directly generating all $\binom{d}{k}$ marginal tables, one could do much better by generating marginal tables for a number of strategically chosen sets of attributes that we call *views*, and then use these views as a synopsis from which one can reconstruct any desired k-way marginals. Second, most datasets in practice define distributions that can be approximately factorized into the product of factors each of which contains only a few dimensions, which is the reason why graphical models are powerful tools to represent probability distributions encountered in practice. Thus it is unnecessary

SIGMOD'14, June 22–27, 2014, Snowbird, UT, USA.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2376-5/14/06 ...$15.00.
http://dx.doi.org/10.1145/2588555.2588575.

to cover all k-way marginals when k is over 3. So long as the chosen views cover all, e.g., 2-way or 3-way marginals, we can reconstruct most k-way margainals with high accuracy for datasets in practice.

More specifically, our PriView approach has the following steps. One first chooses a set of views based on d, ϵ, and N (the number of data points). One then obtains noisy marginal tables for the views by adding Laplacian noise to these marginals. These views, however, may be overlapping. Thus the same information may be provided in multiple noisy views, each with independent noises added. This presents both a challenge and an opportunity. The challenge lies in how to reconcile potentially inconsistent information from different noisy views. The opportunity lies in the fact that by averaging results from different noisy views, one could obtain more accurate results. We thus post-process the noisy tables to make them mutually consistent. These consistent views form the synopsis from which one can compute any k-way marginal.

Our main contribution lies in developing these high-level ideas into an efficient and practical approach. More specifically, we propose to use the concept of covering design to generate sets of views, and introduce a method to select among them. We also introduce an efficient procedure to make noisy views mutually consistent. This step improves the accuracy of these views, and enables accurate reconstruction of k-way marginals. Furthermore, we introduce a Ripple Non-negativity approach to correct negative entries in noise views to improve accuracy. We also propose to apply the principle of maximum entropy for reconstructing k-way marginals. Finally, we experimentally evaluate our approach and show that PriView reduces the error distance by 2 to 3 orders of magnitude when compared with the state of the art. The PriView method also has the advantage that once the synopsis is generated, one can reconstruct k-way marginals for any k; thus one does not need to commit to a specific k value when constructing the noisy views.

The rest of the paper is organized as follows. We formalize our problem definition in Section 2. We then discuss existing approaches in Section 3. We introduce the PriView method in Section 4. We finally present related work in Section 6 and conclude in Section 7.

2. PROBLEM DEFINITION

We consider the following problem: **Given a d-dimensional binary dataset D, and a positive integer $k < d$, we want to differential privately construct a synopsis of D, so that any k-way *marginal contingency table* (marginal table for short) can be computed with reasonable accuracy.**

We now elaborate on this problem definition. D is a dataset that has d binary attributes; thus the set of all attributes can be defined as $\mathbf{A} = \{1, 2, \cdots, d\}$. Each tuple t in D is a binary string in $\{0, 1\}^d$.

Information in D can be equivalently represented using the full contingency table, which has 2^d entries, one for each string $\{0, 1\}^d$. Each entry gives the number of times the corresponding string occurs in D. When one wants to analyze the relationships among some of the attributes in $\mathbf{A}$, marginal tables can be constructed, by summing corresponding entries in the contingency table. More specifically, for any subset $A \subseteq \mathbf{A}$ of $k = |A|$ attributes, the marginal table for A has 2^k entries, one corresponding to each possible assignment of the k attributes in A. We call a marginal table for a set of k attributes a k-way marginal table.

Privacy Goal. The privacy requirement is such that the mechanism should satisfy differential privacy.

Definition 1 (ϵ-Differential Privacy [9, 11]). *A randomized mechanism $\mathcal{A}$ gives ϵ-differential privacy if for any pair of neighboring datasets D and D', and any $S \in Range(\mathcal{A})$,*

$$\Pr[\mathcal{A}(D) = S] \leq e^\epsilon \cdot \Pr[\mathcal{A}(D') = S].$$

In this paper we consider two datasets D and D' to be neighbors if one dataset equals the other dataset with one additional tuple added. We use $D \simeq D'$ to denote this.

One approach for computing a function g on dataset D while satisfying ϵ-differential privacy is to add to the output of $g(D)$ random noise following the Laplace distribution. The magnitude of the noise depends on the privacy budget ϵ, and GS_g, the *global sensitivity* or the L_1 sensitivity of g. This mechanism, denoted by $\mathcal{A}_g$, is given below:

$$\begin{array}{rrl} & \mathcal{A}_g(D) & = g(D) + \mathsf{Lap}\left(\frac{\mathsf{GS}_g}{\epsilon}\right) \\ \text{where} & \mathsf{GS}_g & = \max_{D \simeq D'} |g(D) - g(D')|, \\ \text{and} & \Pr[\mathsf{Lap}(\beta) = x] & = \frac{1}{2\beta} e^{-|x|/\beta} \end{array}$$

In the above, $\mathsf{Lap}(\beta)$ represents a random variable sampled from the Laplace distribution with scale parameter β.

Utility Goal. The utility goal is to generate k-way marginals that are close to the true values. Given a generated k-way marginal table $\tilde{\mathsf{T}}_A$ for a set A of attributes, we consider two error measures and aim at minimizing them.

The first is the L$_2$ distance between $\tilde{\mathsf{T}}_A$ and T_A, the k-way marginal table over A computed from the original dataset, when both are viewed as vectors consisting of 2^k elements. We call this the error distance. For a given method, the error distance is a random variable as its value depends on the noise added by the method. The expected value of the square of error distance is also the expected value of the sum of squared errors in each cell, and we call it the **Expected Squared Error (ESE)**. We often use the ESE to understand a method's utility.

The second is the Jensen-Shannon divergence between $\mathsf{norm}(\tilde{\mathsf{T}}_A)$ and $\mathsf{norm}(\mathsf{T}_A)$, where norm normalizes a marginal table by dividing each cell with the sum of all the cells, so that in the normalized table the sum of all cells equals 1. It is natural to use the Kullback-Leibler divergence between $\mathsf{norm}(\mathsf{T}_A)$ and $\mathsf{norm}(\tilde{\mathsf{T}}_A)$, since $D_{\mathsf{KL}}(P||Q)$ measures the information lost when Q is used to approximate P. However, $D_{\mathsf{KL}}(P||Q)$ is undefined when $Q(i) = 0$ and $P(i) \neq 0$ for some i, which can happen in our setting. We thus choose to use Jensen-Shannon divergence [25], which is a symmetrized and smoothed version of the Kullback-Leibler divergence:

$$D_{\mathsf{JS}}(P||Q) = \frac{1}{2} D_{KL}(P||M) + \frac{1}{2} D_{KL}(Q||M) \qquad (1)$$

where $M = \frac{P+Q}{2}$ and $D_{\mathsf{KL}(P||Q)} = \sum_i \ln\left(\frac{P(i)}{Q(i)}\right) P(i)$

Efficiency requirement. Let $N = |D|$ denote the number of tuples in the dataset. We assume that $N \ll 2^d$, and thus want to avoid $\Theta(2^d)$ computational complexity. According to result in [29], it is computationally unfeasible to output a new synthetic dataset that preserves accuracy for k-way marginals. As PriView generates a synopsis instead of a synthetic dataset, this result does not apply here. It only suggests that generating a synthetic dataset that is consistent with our synopsis may be computationally expensive.

Remarks. We assume that k is relatively small compared with d, that is, running time in low-degree polynomial in 2^k is acceptable. Note that a k-way marginal has 2^k entries, thus complexity linear in 2^k is unavoidable. In the experiments, we consider $k = 4, 6, 8$. However, our proposed PriView method easily extends to larger values of k. PriView consists of two stages. The first stage generates a synopsis of the dataset; and does not change its complexity when k increases. The second stage is to reconstruct any desired k-way marginal, which involves solving a convex optimization problem with 2^k variables.

In the experiments, we use three real datasets, one with $d = 9$ to compare our method against other methods that do not scale, the second with $d = 32$, and the third with $d = 45$. We also use Markov chain models to synthesize multiple datasets with $d = 64$.

3. ANALYZING EXISTING APPROACHES

In this section, we discuss existing approaches to the problem and analyze their effectiveness. Later we experimentally compare with all these methods whenever they can scale to the experimental setting.

3.1 The Flat Method

A basic approach, henceforth referred to as the Flat method, is to generate a noisy version of the full contingency table of D. In this method, one adds noise from distribution $\mathsf{Lap}\left(\frac{1}{\epsilon}\right)$ to all counts in the full contingency table of the dataset. This noisy full contingency table can then be used to compute any k-way marginal tables. This approach takes time $O(2^d)$.

To estimate the ESE, we observe that the noise added to each cell of the full contingency table has variance

$$V_u = \frac{2}{\epsilon^2} \tag{2}$$

We treat V_u as the unit of ESE. To reconstruct a k-way marginal table, one sums up the noisy counts for the corresponding cells in the contingency table. For each entry in the k-way marginal table, one sums up 2^{d-k} cells, and the marginal table has 2^k entries, each with independently generated noise. Thus, the ESE of the Flat method is

$$2^k \times 2^{d-k} V_u = 2^d V_u. \tag{3}$$

This approach works very well when d is small. See [21] for an analysis of its accuracy. However, when d is large it suffers from two problems. First, the time complexity and space complexity are both $\Theta(2^d)$, which is too large for larger values of d. Second, it will be highly inaccurate, as the ESE increases exponentially in d.

3.2 The Direct Method

Another method is to add independently generated Laplacian noise to each k-way marginal table [11]. To compute m marginal tables while satisfying ϵ-DP, one adds noise sampled from $\mathsf{Lap}\left(\frac{m}{\epsilon}\right)$ to each entry, resulting in a variance of $m^2 V_u$. As $m = \binom{d}{k}$ and each table has 2^k entries, this results in an ESE of

$$2^k \binom{d}{k}^2 V_u. \tag{4}$$

Comparing Equations (4) and (3), we can see that for smaller values of d, the Flat method is better. The Direct method starts outperforming the Flat method for larger d's. The exact values of d's for which the Direct method overtakes the Flat method depend on k and are given in the table below.

k	2	3	4	5
Condition for Direct better than Flat	$d \geq 16$	$d \geq 26$	$d \geq 36$	$d \geq 46$

Equations (3) and (4) may over-estimate the error, especially when the computed value is greater than 1. This is because large noises may result in negative values, which can be corrected. However, this over-estimation becomes significant only there are large negative values, in which case the error will remain large even after correcting tghem. For example, suppose that one noise entry is close to $-\frac{N}{2}$, then this means that the noise added to each entry is likely large enough to overwhelm the true counts in most cells.

The Direct method has time complexity $\Theta(\max(d^k, N))$ and space complexity of $\Theta(d^k)$; while the Flat method has time and space complexity of $\Theta(\max(2^d, N))$. When k is small, the Direct method scales much better than the Flat method when d increases.

3.3 Adding Noise in the Fourier Domain

Barak et al. [4] proposed the approach to add noises in the Fourier domain. Conceptually, each full contingency table corresponds to a set of 2^d Fourier coefficients, and each k-way marginal can be reconstructed using 2^k of these coefficients. To reconstruct all k-way marginals, one needs $1 + \binom{d}{1} + \binom{d}{2} + \cdots + \binom{d}{k}$ coefficients (corresponding to all binary strings with at most k 1's). One can publish noisy versions of these coefficients, which can be subsequently used to reconstruct any desired k-way marginal. The magnitude of noise added to the coefficients is similar to that in the Direct method. One advantage of this approach over the Direct method is that because there is a one-to-one correspondence between full contingency tables and the vectors of Fourier coefficients, the noisy marginals reconstructed from noisy Fourier coefficients are consistent in the sense that there exists a full contingency table that corresponds to the generated noisy marginals. However, the corresponding marginal table may contain negative entries. To avoid such negative entries, it is proposed to use linear programming to compute a full contingency table that does not contain any negative entries, while minimizing the largest error between the noisy Fourier coefficients and the Fourier coefficients computed from the contingency table.

Using linear programming to compute a full contingency table, which has 2^d entries, is unfeasible when d is large. When the linear programming step is not used, the method becomes adding noise to the Fourier coefficients, and then reconstruct the k-way marginal. Compared with the direct method, this reduces the ESE by a factor of 2^k.

3.4 Data Cubes

Ding et al. [8] tries to solve a very similar problem as PriView, namely how to reconstruct a set of marginals (query marginals) as accurately as possible. Their approach tries to find a set of marginals that together cover all query marginals, and then try to make them consistent. In the case of low-dimensional binary datasets, the principles in [8] will lead it to choose to publish the noisy version of the full contingency table, which is equivalent to the Flat method.

The main limitation of the approach in [8] is that the techniques, both for view selection and for view consistency, require time polynomial in 2^d, and is thus unfeasible in our setting. The techniques in [8] first organize all possible marginals in a lattice under the subset relation. With a d-dimensional binary dataset, there are 2^d marginals in the lattice. The method in [8] iteratively selects which marginals should be released. In each iteration, it traverses all 2^d marginals in the lattice and greedily selects one. The consistency techniques is also polynomial in 2^d because it constructs all marginals in the lattice that are above the chosen ones in order to achieve consistency.

Our approaches solve the same problem as in [8], but use different techniques in order to scale to larger d values. We are able to do this for the following reasons. First, unlike [8] we do not require all query marginals are covered by the views. Second, we use existing covering design, instead of greedily searching the lattice. Third, we develop a much more efficient consistency procedure.

3.5 The Matrix Mechanism

Proposed by Li et al. [22], the matrix mechanism is a method of answering a workload of predicate counting queries. Given some workload, the method finds an alternative set of queries, called a query strategy. Standard Laplace noise is then added to the strategy queries, the answers to which are used to give accurate answers to the original workload. A natural way to adapt the matrix mechanism to our purposes is to assume that the workload consists of all possible k-way marginal count queries.

Finding the best possible strategy queries is a semi-definite optimization problem. This requires a running time $O\left(2^{3d}(2^k\binom{d}{k})^3\right)$. An approximation presented in [23] runs in $O\left(2^{3d}\right)$, and another method in [31]. For all practical purposes, running the original optimization problem is unfeasible. Any approximation must be polynomial in 2^d, since there are 2^d basic queries one can use to form any strategy. We are able to run both approximation method for $d = 9$. We explore the accuracy of the approximations for small datasets in our experiments.

3.6 Multiplicative Weights Mechanism

Several approaches based on multiplicative updates were proposed in [17, 15, 16]. In these approaches, one maintains and improves a distribution approximating a given dataset for answering a set of counting queries. We consider the non-interactive version of the approach by Hartz et al. [16], which we use MWEM to denote. To release k-way marginals, the approach works as follows. The set of queries is the set of all k-way marginals. One starts with an initial full contingency table that is uniform, and then selects, using the exponential mechanism [26], a k-way marginal that is most incorrectly answered by the current distribution. One then obtains a noisy answer to the selected marginal, and updates the distribution to match the current state of knowledge. This process is repeated T times. Each round is allocated a privacy budget ϵ/T. Half of the privacy budget at each round is used to select a marginal that is most incorrectly answered, and the other half is used to get a noisy answer. The basic version of MWEM performs a single multiplicative update step in each round using the new query, and outputs the average from the distribution after each round. A theorem provides a utility bound for this method.

Experiments in [16] use the following two improvements over the basic version. In each round, one iterates over all existing query results 100 times to attempt to maximally benefit from the query results. Furthermore, queries are answered using the final distribution instead of using the average. While these improvements greatly improved the actual performance in practice, using them means that the theoretically proven utility bound no longer holds.

Maintaining a full contingency table requires time and space complexity $\Omega(2^d)$, and is unfeasible for larger d values. The largest d value used in [16] is $d = 16$. A technique is introduced in [16] to avoid explicitly storing the full contingency table. This technique, however, is limited to the case when the attributes can be partitioned into disjoint parts so that no query involves attributes from more than one part, which is not applicable to our problem.

3.7 Learning Based Approaches

The class of learning based approaches, first described in [15], with the current state of the art approach in this line of work, which is also the most practical, described in [28]. These approaches view the database as a function on queries, and aim to learn a good approximation of this function while satisfying differential privacy.

This approach in [28] runs in time $|D| \cdot d^{C\sqrt{k}\log(1/\gamma)}$, where γ is an accuracy parameter and C is a constant. While this is asymptotically appealing, it is impractical for the ranges of parameters that are of practical relevance. The complexity $d^{C\sqrt{k}\log(1/\gamma)}$ is asymptotically better than the $\binom{d}{k}2^k$ complexity of the Direct method; however, for smaller k's, this complexity is actually much worse. The utility bound suggests that when k is between 60 and 100, this method can start outperforming the Direct method; however, the method (and in fact all methods) will become computationally unfeasible long before k reaches that range, e.g., a k-way marginal requires $\Theta(2^k)$ storage.

4. THE PRIVIEW APPROACH

In this section, we describe our PriView approach for publishing k-way marginal tables from a d-dimensional dataset.

4.1 A Fruitful Middle Ground

Both the Direct method and the Flat method are unsuitable when d is large. They represent two extremes. The Flat method generates one huge contingency table, and the Direct method generates many tiny marginal tables. We observe that there is a fruitful middle ground between the two extremes: one could generate a small number of "mid-size" marginal tables that can be used to reconstruct k-way marginal tables with better accuracy.

To see the benefit of publishing these "mid-size" marginals, consider, for example, the case where $d = 16$ and $k = 2$. The Flat methods publishes 1 table with 2^{16} entries, resulting in ESE of $2^{16}\,V_u = 65536\,V_u$. The Direct method publishes $\binom{16}{2} = 120$ tables, resulting in ESE of $2^2\binom{16}{2}^2\,V_u = 57600\,V_u$. However, one can publish six 8-way marginal tables, while ensuring that any 2-way marginal table can be reconstructed from one of the published 8-way marginal tables. This results in an ESE of $2^2 \cdot 6^2 \cdot 2^6 = 9126\, V_u$, a number significantly smaller than either extreme.

We call these "mid-size" marginals "views" of the dataset D. When we consider larger values of k, e.g., $k = 6$ or $k = 8$, trying to come up with views that cover all sets of k attributes may result in too many views, and too much noise to be added. Our method does not attempt to cover all such k-way marginals, and instead develop techniques to compute marginal tables when the k attributes are not fully contained in any single view.

Notation. We use T_A to denote the noisy marginal table over $A \subseteq \mathbf{A}$. Let Q_A denote the set of all possible counting queries by assigning each attribute in A a value of either 0 or 1. Then T_A can be viewed as a function $\mathsf{T}_A : Q_A \to \mathcal{R}$, where $\mathcal{R}$ is the set of real numbers. We use $\mathsf{T}_A \equiv \mathsf{T}'_A$ to denote that the two marginal tables over A are the same for every entry. And $\mathsf{T}_A(\cdot) \geq x$ to mean that every entry in the table is greater than x. Given T_A, and $A' \subseteq A$, we use $\mathsf{T}_A[A']$ to denote the marginal over A' constructed from T_A by summing the corresponding entries. In particular, $\mathsf{T}_A[\emptyset]$ gives the sum of all cells in T_A.

Definition 2 (View Consistency). *We say that two marginal tables T_{V_i} and T_{V_j} are **consistent** if and only if the marginal table over attributes in $V_i \cap V_j$ reconstructed from T_{V_i} is exactly the same as that reconstructed from T_{V_j}, that is, $\mathsf{T}_{V_i}[V_i \cap V_j] \equiv \mathsf{T}_{V_j}[V_i \cap V_j]$.*

4.2 Summary of The PriView Approach

We now give a summary of our PriView approach, which has the following steps.

1. **Choose the Set of Views.** The first step is to choose which marginal tables to generate. More specifically, we need to choose the set of views:
$$\mathbf{V} = \{V_1, V_2, \ldots, V_w\},$$
where each view $V_i \subseteq \mathbf{A}$ for $1 \leq i \leq w$ and $\bigcup_{i=1\ldots w} V_i = \mathbf{A}$. We choose these views so that each size-j marginal is covered by some V_i for some small values of j (often $j = 2$ or $j = 3$), and the resulting noise variance is relatively small. We describe the details for choosing $\mathbf{V}$ in Section 4.5.

2. **Generate Noisy Views.** In this step, for each view V_i we construct a differentially private marginal table, T_{V_i}, by adding Laplace noise $\mathsf{Lap}\left(\frac{w}{\epsilon}\right)$ to the counts specified by the table. This is the only step in the PriView approach that needs direct access to the dataset D. After this step, the dataset D is no longer accessed.

3. **Consistency Step.** In this step, we perform constrained inference on the marginal tables T_{V_i}'s. This step serves two purposes. First, by exploiting existing redundancy among views, it improves the accuracy of T_{V_i}'s. Second, it ensures that for any $i \neq j$, T_{V_i} and T_{V_j} are consistent. (See Definition 2 above.) This enables more accurate computation of marginals not fully covered by any view. We describe how this is performed in Section 4.4.

4. **Generating k-way Marginals.** Finally, we propose to use the maximum entropy method to generate an arbitrary k-way marginal table based on the T_{V_i}'s. In Section 4.3, we also discuss a few alternatives.

In the rest of this section, we present the steps of our PriView approach in the reverse order.

4.3 Computing k-way Marginals

Given a set of marginals produced from views in $\mathbf{V}$ that are mutually consistent, and a set A of attributes, where $|A| = k$, we want to compute the k-way marginal T_A. When all the attributes in A are "covered" by at least one view V_i, i.e. $A \subseteq V_i$, this step is trivial. We can construct T_A by summing over corresponding entries in T_{V_i}. However, when $A \not\subseteq V_i$ for each $V_i \in \mathbf{V}$, we need a method to return the marginal which best represents what we know about it.

Given A, for any view V_i, if $A \cap V_i = \emptyset$, then T_{V_i} provides no information. When $A \cap V_i$ contains j attributes, then T_{V_i} provides 2^j linear constraints on cells for T_A. Constraints obtained from different views, however, may not be linearly independent. Thus, we can extract all linearly independent constraints from all views. Unless A is fully contained in some view, the number of linearly independent constraints will always be less than 2^k. The combination of all constraints from all intersecting views results in an *under-specified* system of equations.

We now describe three approaches we have considered for the reconstruction step. The last approach is what we propose to use. We will experimentally compare the effectiveness of these four approaches.

Linear programming. One strategy is to adopt the linear programming approach proposed in [4], that is, to compute a marginal so that all the entries are non-negative, and the constraints are satisfied as much as possible.

$$\begin{array}{ll} \mathsf{minimize} & \tau \\ \mathsf{subject\ to} & \forall_{a \in Q_A} \mathsf{T}_A(a) \geq 0 \\ & \forall_{V_i \in \mathbf{V}} \forall_{a' \in Q_{V_i \cap A}} |\mathsf{T}_{V_i}(a') - \mathsf{T}_A(a')| \leq \tau \end{array}$$

In the above, τ is the maximum error for all constraints. This approach does not require the marginals for the views to be consistent.This approach is unsatisfying in that it considers consistency with regards to the constraints as the only factor in choosing a solution. Recall that we have under-specified constraints for the marginal table, and when they are consistent there are more than one solutions. The linear programming method has *no preference* among the set of consistent solutions. We thus consider other methods below. These methods all assume that all views are consistent.

Least Square Solution. One common approach to solving underdetermined systems of equations is to select the solution that has the least L_2 norm. We can thus express the problem of computing a k-way marginal as the following optimization

$$\begin{array}{ll} \mathsf{minimize} & \sum_{a \in Q_A} \mathsf{T}_A(a)^2 \\ \mathsf{subject\ to} & \forall_{a \in Q_A} \mathsf{T}_A(a) \geq 0 \\ & \forall_{V_i \in \mathbf{V}} \forall_{a' \in Q_{V_i \cap A}} \mathsf{T}_{V_i}(a') = \mathsf{T}_A(a') \end{array}$$

One can use standard quadratic programming approaches to solve the above optimization problem.

Maximum Entropy. The problem we have at hand is the estimation of a probability distribution with partial information. A common approach is to apply the Principle of Maximum Entropy. In Bayesian probability theory, the principle of maximum entropy states that, subject to precisely stated prior data (a set of precise constraints), the probability distribution which best represents the current state of knowledge is the one with largest information-theoretical entropy. We can thus express the problem of computing a k-way marginal as the following optimization

$$\begin{array}{ll} \textsf{maximize} & -\sum_{a\in Q_A} \frac{\mathsf{T}_A(a)}{N_{\mathbf{V}}} \cdot \log\left(\frac{\mathsf{T}_A(a)}{N_{\mathbf{V}}}\right) \\ \textsf{subject to} & \underset{a\in Q_A}{\forall}\ \mathsf{T}_A(a) \geq 0 \\ & \underset{V_i\in\mathbf{V}}{\forall}\ \underset{a'\in Q_{V_i\cap A}}{\forall}\ \mathsf{T}_{V_i}(a') = \mathsf{T}_A(a') \end{array}$$

In the above, $N_{\mathbf{V}}$ denotes the total count obtained from any $V_i \in \mathbf{V}$. Since all views are consistent, the value should be the same for any V_i.

We use an off-the-shelf convex optimization tool to solve the above optimization problem. In experiments, we found that sometimes convex optimization fails to find a solution after a fixed number of iterations. Since the constraints we have are noisy, we gradually relax the constraints by replacing each equality constraint as a pair of tight inequality constraints (using increasing larger toleration threshold) until the solver can find a solution.

4.4 Consistency between Noisy Views

The input to this step is a set of noisy marginal tables T_{V_i}, one for each set of attributes in the view set $\mathbf{V}$. The output is perturbed versions of these tables that are mutually consistent.

Mutual Consistency on a Set of Attributes. We first describe the method of ensuring that a set of view marginals are consistent on their common set of the attributes. Consider, wlog, views $V_1, \cdots, V_j$, let $A = V_1 \cap \ldots \cap V_j$. We want to ensure that $\mathsf{T}_{V_1}[A] \equiv \cdots \equiv \mathsf{T}_{V_j}[A]$. The goal is to make sure that $\mathsf{T}_{V_x}(a) = \mathsf{T}_{V_y}(a)$ for any two views $V_x, V_y \in \{V_1, V_2, \cdots, V_j\}$.

Here we apply the consistency techniques proposed in [19], and achieve consistency in two steps. The first step is to compute the best approximation for the marginal table T_A. Assuming that all $\mathsf{T}(V_i)$'s are constructed using the same privacy budget, and that $|V_1| = \cdots = |V_j|$, then for any $a \in Q_A$, the best approximation of $\mathsf{T}_A(a)$ that minimizes the variance of the noise is the arithmetic mean of the value from all views, i.e.:

$$\mathsf{T}_A(a) = \frac{1}{j}\sum_{i=1}^{j} \mathsf{T}_{V_i}(a)$$

The second step is to update all T_{V_i}'s to be consistent with T_A. One can view T_A as a set of $2^{|A|}$ mutually exclusive constraints on all the entries in T_{V_i}. For each constraint $c \in Q_A$, do

$$\mathsf{T}_{V_i}(c) \leftarrow \mathsf{T}_{V_i}(c) + \frac{\mathsf{T}_A(a) - \mathsf{T}_{V_i}(a)}{2^{|V_i|-|A|}},$$

where a is the query c restricted to attributes in A.

The following example illustrates mutual consistency between $\mathsf{T}_{V_1=\{a_1,a_2\}}$ and $\mathsf{T}_{V_2=\{a_1,a_3\}}$.

$\mathsf{T}_{V_1=\{a_1,a_2\}}$ before consistency	0.3	0.3	0.3	0.1
$\mathsf{T}_{V_2=\{a_1,a_3\}}$ before consistency	0.2	0.3	0.1	0.4
T_{V_1} projected on a_1	0.6		0.4	
T_{V_2} projected on a_1	0.5		0.5	
best estimat of $\mathsf{T}_{\{a_1\}}$	0.55		0.45	
$\mathsf{T}_{V_1=\{a_1,a_2\}}$ after consistency	0.275	0.275	0.325	0.125
$\mathsf{T}_{V_2=\{a_1,a_3\}}$ after consistency	0.225	0.325	0.075	0.375
T_{V_1} projected on a_2 is unchanged	0.6		0.4	
T_{V_2} projected on a_3 is unchanged	0.3		0.7	

Note that after mutual consistency, $\mathsf{T}_{V_1=\{a_1,a_2\}}$ and $\mathsf{T}_{V_2=\{a_1,a_3\}}$ agree on $\{a_1\}$ without changing the marginals of attributes not involved in the consistency.

Overall Consistency. We now describe how to ensure that all marginals are consistent on all subsets of attributes by performing a series of mutual consistency steps. The challenge here is to determine the order with which to perform the mutual consistency procedure. Suppose that we first make T_{V_1} and T_{V_2} consistent on the attribute set $V_1 \cap V_2 = \{a_1, a_2\}$, and then make T_{V_1} and T_{V_3} consistent on the attribute set $V_1 \cap V_3 = \{a_1, a_3\}$. This second step may make T_{V_1} and T_{V_2} inconsistent, because it may change the distribution of the attribute $\{a_1\}$. We note, however, if we have already made V_1, V_2, V_3 consistent on $\{a_1\}$, then the second step will not invalidate the consistency established in step 1. This is formalized in the following lemma.

LEMMA 1. *If T_{V_1} and T_{V_2} are already consistent on $A \subset V_1 \cap V_2$, then enforcing mutual consistency between T_{V_1} and T_{V_2} does not change the projection of T_{V_1} on any subset of $(V_1 \setminus V_2) \cup A$.*

PROOF. When updating T_{V_1} to affect the distribution of attributes in $V_1 \cap V_2$, increase to some cells in T_{V_1} must be balanced by equal amount of subtraction to other cells. When projecting T_{V_1} onto any subset of $(V_1 \setminus V_2) \cup A$, these changes cancel out. □

Utilizing Lemma 1, we propose the following procedure to ensure overall consistency. Take all sets of attributes that are the result of the intersection of some subset of $\mathbf{V}$; these sets form a partial order under the subset relation. One then obtain a topological sort of these sets, starting from the empty set. For each set A, one finds all views that include A, and ensures that these views are consistent on A. Note that consistency on the empty set of attributes ensures that the total counts in all marginals are the same, and this is done first. Following this order, a later consistency step will not invalidate consistency established in previous steps. Furthermore, following any topological order will result in the same result.

Accuracy through Ripple Non-Negativity. Noisy marginals may contain negative numbers. Since all entries in the true marginals are non-negative, one should change these negative numbers to 0. Intuitively this will improve the accuracy. However, experimental results have found that simply changing negative numbers to 0 dramatically increases the error. We believe that this is because the noisy answer for any query that includes cells that have low true counts is positively biased, because negative noises are partially removed, but positive noises remain. At the same time, the noisy answer for a query that does not include cells with low true counts does not have such a bias.

To avoid such bias, we introduced the following "Ripple" non-negativity method, which turns negative counts into 0 while decreasing the counts for its neighbors to maintain overall count unchanged. Given an ℓ-way marginal table, for any entry that has count $c < -\theta$ for some small value of θ, we set the entry to 0 and subtract $|c|/\ell$ from each of its ℓ neighboring cells (i.e., cells obtained by flipping one of the ℓ bits). As doing this may result in other cells to be below $-\theta$, this is iterated until no cell is below $-\theta$. As each iteration distributes a negative count into ℓ neighbors, it is guaranteed to terminate quickly.

Applying the ripple non-negativity step to the views, however, may make them inconsistent. We use Consistency + Non-negativity + Consistency. While the last consistency step may again introduce negative values, they tend to be very small. We tried adding more iterations of "Non-negativity + Consistency" and found that they produce negligible additional improvements.

4.5 Choosing a Set of Views

Finally, we need a method to choose the set **V** of views. Obviously it is desirable to have every attribute appear in at least one view; otherwise, information about that attribute is not present, and the estimation has to assume that it is equally likely to be 1 or 0, which may turn out to be very inaccurate. Going beyond covering every single attribute, it is natural to ensure that every pair of attributes are covered in some view, and then every triple of attributes, and so on. We thus take advantage of the following concept from combinatorics theory.

Definition 3 (Covering Design [14, 13]). *A (w, ℓ, t)-covering design of a set P is a collection of w subsets of P, each of which contains ℓ elements and is called a block, such that any t-element subset of P is contained in at least one block. An* optimal *(w, ℓ, t)-covering design minimizes the number of blocks.*

Clearly, for a fixed P and t, the larger the ℓ, the smaller the w needs to be. We need a method to choose w, ℓ, t.

Setting $\ell = 8$. We recommend choosing $\ell = 8$ (or values close to it, such as between 7 and 11), and we now explain why. There are two sources of errors for any marginal generated by PriView: noise error, which is due to Laplacian noise added to satisfy differential privacy, and coverage error, which occurs when the marginal is not fully covered by any view.

To compute the noise error, we consider the result of reconstructing a pair of attributes. When choosing a set of w views of length ℓ each, reconstructing a pair from a single view that covers it has ESE of, $2^\ell w^2 \frac{2}{\epsilon^2}$. However, on average we expect a pair to be covered by $\frac{w\ell(\ell-1)}{d(d-1)}$ views, since there are $\frac{d(d-1)}{2}$ pairs, and these views include $\frac{w\ell(\ell-1)}{2}$ pairs. Averaging will reduce the ESE; thus the normalized error (after dividing N) is on the order of

$$err = \frac{1}{N}\sqrt{\frac{\frac{2^{\ell+1}w^2}{\epsilon^2}}{\frac{w\ell(\ell-1)}{d(d-1)}}} = \frac{2^{\frac{\ell+1}{2}}}{N\epsilon}\sqrt{\frac{wd(d-1)}{\ell(\ell-1)}} \tag{5}$$

We want to choose ℓ to minimize (5). We are given N, d, ϵ, and can choose w and ℓ. We observe that w and ℓ are interdependent. When we compare covering designs with similar coverage, we can assume that they have roughly the same number of pairs, and thus $w \propto \frac{1}{\ell(\ell-1)}$. Then choosing ℓ to minimize (5) can be achieved by minimizing $\frac{2^{\ell/2}}{\ell(\ell-1)}$. A similar analysis for noise error of reconstructing triples require minimizing $\frac{2^{\ell/2}}{\ell(\ell-1)(\ell-2)}$. Choosing $\ell = 8$ works well, as can be seen from the following table.

ℓ	$\frac{2^{\ell/2}}{\ell(\ell-1)}$	$\frac{2^{\ell/2}}{\ell(\ell-1)(\ell-2)}$	ℓ	$\frac{2^{\ell/2}}{\ell(\ell-1)}$	$\frac{2^{\ell/2}}{\ell(\ell-1)(\ell-2)}$
5	0.283	0.094	9	0.314	0.045
6	0.267	0.067	10	0.356	0.044
7	0.269	0.054	11	0.411	0.046
8	0.286	0.048	12	0.485	0.048

It is interesting to know that the choice of ℓ is independent from N, d, ϵ, thus $\ell = 8$ can be used cross all settings. We experimentally tested other choices of ℓ, and found that values close to 8 give quite close results.

Choosing t. Choosing the value for t does depend on N, d, ϵ. We observe that the overall error is affected by both the noise error and coverage error. Larger t provides better coverage, at the cost of higher noise error. We want to ensure that neither one is much larger than the other. The coverage error, however, cannot be analytically computed, as it depends on the nature of the dataset. It is high when there is high correlation among some attributes, which are not covered together. Empirically, we observe that choosing t so that the noise error falls in the range of 0.001 and 0.003 seems to work well.

Therefore, we find the best covering design with $\ell = 8$ for $t = 2, 3, 4$ and compute the error value from Equation (5). Many covering design solutions exist in online repositories such as [13], from which we look up the covering designs for the desired N, t, ℓ value and find the design as well as the value w. Computing the noise error using (5) also requires N, for which a rough estimate suffices. For example, one could use a very small privacy budget, say, $\epsilon = 0.001$ to obtain a noisy count for it.

As an example, consider the Kosarak dataset. The errors below are computed using the best known covering designs we found from [13], with $d = 32, N \approx 900,000, \epsilon = 1$.

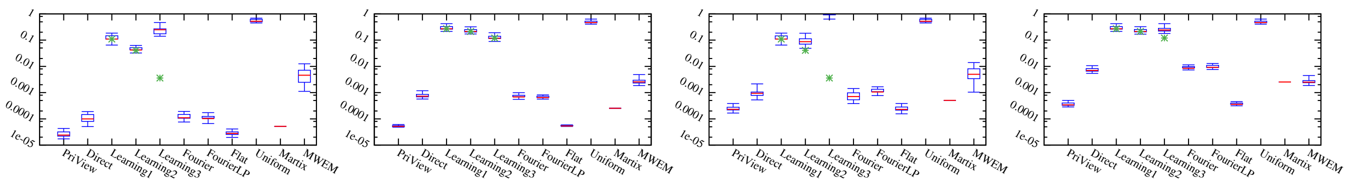
	$t=2$		$t=3$		$t=4$	
ℓ	w	err	w	err	w	err
8	20	0.00047	106	0.0011	620	0.0026

The noise error for $t = 2$ is 0.00045, which is significantly less than 0.001; that for $t = 3$ is 0.0011, and that for $t = 4$ is 0.0026, which is probably not worthwhile of the additional coverage improvement over $t = 3$. We thus would choose $t = 3$. However, for $\epsilon = 0.1$, these errors will be multiplied by a factor of 10, and we would need to choose $t = 2$. The other datasets we use have higher d values, and we will need to use $t = 2$ for them.

4.6 Space and Time Complexity

The space complexity of PriView approach is the set of views, i.e., $w2^\ell$, which is quite small.

The time complexity for publishing the synopsis consists of the following three steps: First, constructing noisy views takes time $O(wN + w2^\ell)$. Second, the time complexity of the consistency step is determined by how many mutual consistency steps are taken, and how many views are involved in



(a) $\epsilon = 1.0, k = 2$ (b) $\epsilon = 1.0, k = 4$ (c) $\epsilon = 0.1, k = 2$ (d) $\epsilon = 0.1, k = 4$

Figure 1: Comparing all approaches on the MSNBC dataset ($d = 9$). Results are L_2 error shown in log scale. PriView uses $C_2(6, 3)$.

each step. The non-negativity step has time complexity $O(w2^\ell)$. In our experiments, the total time for publishing a synopsis varies from several seconds to a few minutes.

The time to reconstruct one k-way marginal is determined by the convex optimization procedure with 2^k variables. Depending on k and the number of constrains, we've observed that it takes time between less a second and half a minute.

The following table shows the actual running time using a Python implementation and a 2.3GHZ processor. In the table, P denotes the time to construct the synopsis, including constructing noisy views, conduct ripple non-negativity on the views, and then the consistency step. The process of reconstructing a single 6-way marginal and a single 8-way marginal is represented by Q_6 and Q_8 correspondingly. The unit of time is second.

	Kosarak : $d = 32$		*AOL* : $d = 45$	
	$C_2(8, 20)$	$C_3(8, 106)$	$C_2(8, 42)$	$C_3(8, 326)$
P	8.78s	90.81s	47.42s	593.27s
Q_6	0.16s	1.78s	2.71s	11.79s
Q_8	2.79s	35.96s	20.76s	77.54s

Note that using covering design for $t = 2$ is quite fast on both datasets, and $C_3(8, 326)$ for AOL is not recommended and included here only for comparison purposes.

4.7 Extensions to Categorical Attributes

We focus on binary dataset, following the tradition in the literature on general methods for computing marginals [4, 15, 7, 18, 28]. This simplifies the presentation of the methods. However, PriView can be extended to datasets with categorical attributes that are not binary. The view consistency method in Section 4.4 and maximum entropy-based reconstruction method in Section 4.3 can be applied directly with non-binary categorical attributes. The only change is in the Ripple Non-negativity step, neighboring cells are obtained by changing only one value (as opposed to flipping one value). The view selection method, however, needs some modification. The same principles still apply. First, we want to limit the size (i.e., number of cells) in each view marginal. Let s denote the number of cells of each view, i.e., $s = 2^\ell$ when attributes are binary. For ease of analysis, assume that each attribute has b values, thus each view has $\log_b s$ attributes. To minimize the error, we want to minimize $\frac{\sqrt{s}}{\log_b s(\log_b s-1)}$ and $\frac{\sqrt{s}}{\log_b s(\log_b s-1)\log_b s(\log_b s-2)}$. As b increases, the optimal s values generally increases. Based on the minimization objective, as a rough guideline, we recommend the following ranges of s for different b values:

b	2	3	4	5
s	$100 - 1000$	$150 - 2000$	$200 - 3200$	$250 - 5000$

We also recommend choosing $t = 2$, since it apparently works well for a wide range of cases. One can then choose views to ensure that every pair of attributes are covered in some view. This step likely requires adapting existing covering design algorithms to the case where the number of attributes in each block depends on how many values these attributes have. Simple greedy algorithms can also be developed. Evaluating the performance of such methods is beyond the scope of this paper.

5. EXPERIMENTAL EVALUATION

In this section, we present the experimental results that demonstrate the efficacy of PriView.

Datasets. We use the following datasets in our experimental comparison

Kosarak [2]: This is a click-stream dataset from a Hungarian on-line news portal. We processed this dataset to include only the top 32 most popular pages. Each record thus has 32 binary attributes corresponding to whether a user visited the webpage. The total number of records in this dataset is 912,627.

AOL [1]: The AOL search log dataset includes the search keywords for 647,377 users. We used the WordNet [12] hypernyms corpus to generate a generalization taxonomy for the search keywords, which resulted in 45 categories. Each keyword was replaced by one of the 45 categories, resulting in 45 binary attributes.

MSNBC [3]: This is a click-stream dataset for msnbc.com, with 989,818 users. Each attribute indicates whether a user visited a particular pages in the msnbc portal. We preprocessed this dataset to include only 9 attributes.

MCHAIN: We synthesize several datasets using the markov chain model. We follow the approach of stationary binary sequence used in [30]. We vary the orders from 1 to 7, to examine the effectiveness of PriView with datasets with various degree of correlation among attributes. For order i, given the previous i bits, the probability of next bit being 1 is $0.5 + \left(1 - \frac{2s}{i}\right)/4$. Here s is number of 1's in the previous i bits. We treat each generated series of 64 bits as a record.

Evaluation Methodology. We use $\epsilon \in \{1.0, 0.1\}$ and $k \in \{4, 6, 8\}$ for our experiments. For each value of k, we randomly generate 200 sets of k attributes. For PriView with a set of views, we compute the average error of each query of five runs. In each run, we add noises to the views, and conduct consistency to compute the synopsis. We then plot the distribution of the 200 average errors. For comparison of PriView with other methods on the two large datasets, we also measure Jensen-Shannon divergence given in Equation 1. As results show that L_2 error distance and Jensen-Shannon divergence are similar, we do not show the Jensen-Shannon divergence for other experiments due to space limitation. When plotting L2 error, we normalize the value by dividing the error with N, so that the numbers are comparable across datasets.

To provide a clearer picture of an algorithm's behavior, we use candlesticks to plot the profile of errors for all query sizes. Each candlestick provides 5 pieces of information: the 25 percentile (the bottom of candlestick), the median (the bottom of the box), the 75 percentile (the top of the box), the 95 percentile (the top of the candlestick), and the arithmetic mean (the red bar). We use the notation $C_t(\ell, w)$ to denote a covering design that covers all t-sized subsets of attributes using w views with ℓ attributes in each view.

As a baseline comparison, we also plot the L_2 error for the Uniform method, which always returns a uniformly distributed marginal. If a method is no better than Uniform in a setting, then applying it in the setting is meaningless.

5.1 Comparisons on MSNBC

Figure 1 shows the results for comparing PriView against the existing methods we have discussed in Section 3 on the MSNBC dataset. We reduce the number of dimensions to 9 in order to be able to run the Learning-based approach [15] and the Matrix Mechanism [23]. These approaches have difficulty to scale to larger dimensions.

For such a small d value, most methods perform well. Our analysis would predict the Flat method to work quite well, and the Direct method to be slightly worse. We observe that PriView performs as well as Flat.

For the Fourier method [4], we present the results before and after the linear programming optimization, indicated by Fourier and FourierLP respectively. The results for both are comparable, and are essentially the same as that of the Direct method. The DataCube method in [8] would choose Flat, which performs very well. This method, however, cannot be used for large d values.

For the Matrix Mechanism, we plot the expected error variance by examining the strategy matrix. The result is better than direct, and worse than flat and PriView. This indicates that the approximation used in the Matrix Mechanism is not closer to optimal than the other methods.

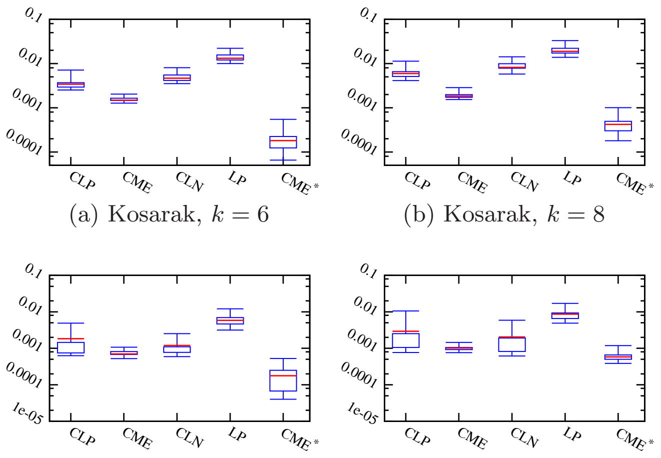
The most interesting results in this figure are those for the learning-based approach. The algorithm requires choosing a γ value. Decreasing γ decreases approximation error, at the cost of increasing noise errors. We choose $\gamma = \{1/2, 1/4, 1/8\}$ which correspond to Learning1, Learning2, and Learning3 respectively in Figure 1. We also plot the result without adding the noise to satisfy differential privacy, indicted by the green stars on the graphs; this shows errors due to the approximation. We observe that the learning-based approach performs very poorly. Even without noises, the results are much worse than other methods. When $\gamma = 1/8$, noise error starts to dominate the approximation error; thus continuing decreasing γ will only make the accuracy worse. These learning-based approaches also do not scale well when d increases.

For the MWEM approach [16], an important parameter is T, the number of rounds. It is suggested in [16] that T should be greater than $4 \log d$. We use $T = \lceil 4 \log d \rceil + 2 = 15$, and note that the results are worse than Flat and Direct. We note that the error range is wide for $k = 2$, but narrower for $k = 4$. This is because obtaining 15 pairs is not quite enough to cover all 45 pairs, and pairs not covered may have large errors. On the other hand, obtaining 15 quadruples provides a better coverage.

5.2 Comparison on Kosarak and AOL.

The results for the Kosarak and AOL datasets are shown in Figure 2. The only methods that scale to such large datasets are those that do not require 2^d complexity; namely the Direct method and the Fourier method. We optimize their results by removing negative values and redistributing the difference evenly to all cells. This can sometimes make the L_2 error to be smaller than predicted by Equation (4). We show both the L_2 error and the Jensen-Shannon divergence for them. Since it is unfeasible to run the Flat method on 32 and 45 dimensions, we calculate and plot the expected normalized L_2 norm for Flat. In many situations, this calculated value is far greater than 1, which are most likely due to reconstructed marginals including large negative values. We cap it at 1 in order to simulate improvements from applying nonnegativity constraint. The results show the superiority of the PriView method, achieving 2 to 3 orders of magnitude of improvement over other approaches.

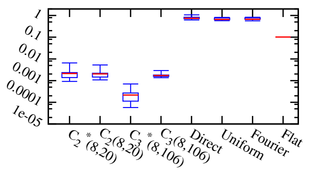
Our results show that the only case Direct outperforms Uniform is for the Kosarak dataset ($d = 32$) when $\epsilon = 1.0$ and $k = 4$. For all other settings, it essentially returns noise. The Flat method fares a little better in that when $d = 32$ and $\epsilon = 1.0$, its accuracy for $k = 6$ and $k = 8$ is an order of magnitude better than Uniform. For all other settings, it also returns noise. The Fourier method performs just slightly better than Direct.



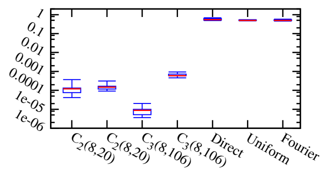
(a) Kosarak, $k = 6$ (b) Kosarak, $k = 8$

(c) AOL, $k = 6$ (d) AOL, $k = 8$

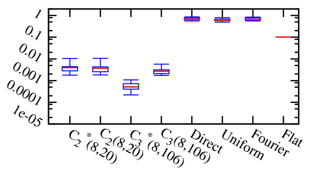
Figure 3: Comparing L_2 error for different reconstruction methods on the Kosarak dataset using $\mathbf{V} = C_3(8, 106)$ and on the AOL dataset using $\mathbf{V} = C_2(8, 42)$, all with $\epsilon = 1.0$. CME is the maximum entropy method, LP is the linear programming method, CLP is the linear programming method with a consistency preprocessing step, CLN is the least square method, and CME* is the maximum entropy method without adding noise.



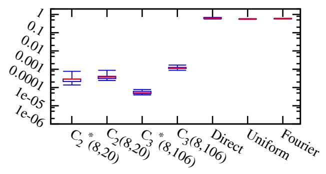
(a) $L_2, \epsilon = 1.0, k = 6$



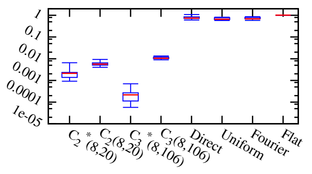
(b) $D_{JS}, \epsilon = 1.0, k = 6$



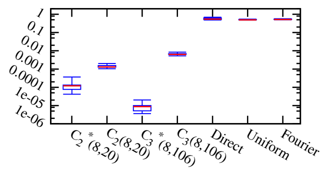
(c) $L_2, \epsilon = 1.0, k = 8$



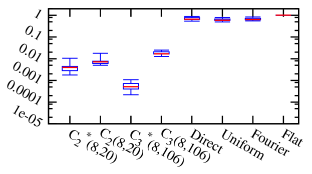
(d) $D_{JS}, \epsilon = 1.0, k = 8$



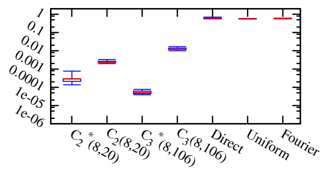
(e) $L_2, \epsilon = 0.1, k = 6$



(f) $D_{JS}, \epsilon = 0.1, k = 6$

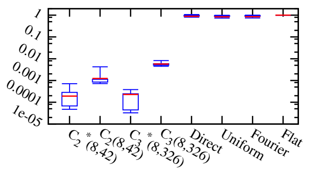


(g) $L_2, \epsilon = 0.1, k = 8$

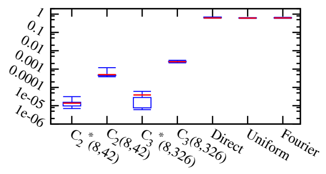


(h) $D_{JS}, \epsilon = 0.1, k = 8$

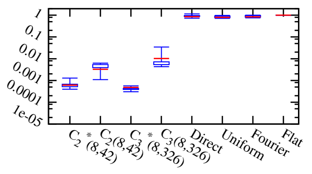
Kosarak, $d = 32$



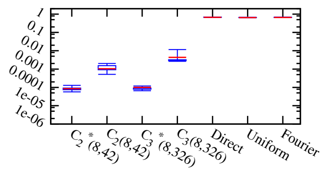
(i) $L_2, \epsilon = 1.0, k = 6$



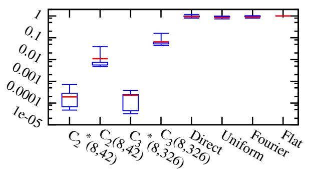
(j) $D_{JS}, \epsilon = 1.0, k = 6$



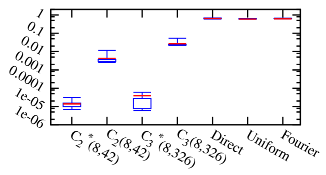
(k) $L_2, \epsilon = 1.0, k = 8$



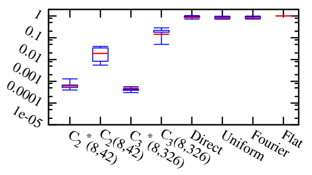
(l) $D_{JS}, \epsilon = 1.0, k = 8$



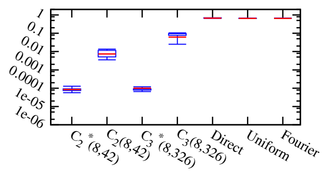
(m) $L_2, \epsilon = 0.1, k = 6$



(n) $D_{JS}, \epsilon = 0.1, k = 6$



(o) $L_2, \epsilon = 0.1, k = 8$



(p) $D_{JS}, \epsilon = 0.1, k = 8$

AOL, $d = 45$

Figure 2: Comparing PriView with Flat and Direct on Kosarak and AOL. For Flat, only the expected average L_2 error is plotted. Results are shown in log scale. $C_t^*(l, w)$ uses covering design $\mathbf{V} = C_t(l, w)$ without adding noise.

5.3 Reconstruction & Non-negativity

Figure 3 shows the results from comparing the various reconstruction methods presented in Section 4.3. It is clear from the results that the maximum entropy method outperforms the alternatives.

The Linear Programming approach performs the worst. Recall that the Linear Programming approach does not require a consistency step. Instead, it tries to find a k-way marginal that is closest to all constraints. We conjecture that adding a consistency preprocessing step before applying the linear program will improve accuracy. To validate this claim, we ran the experiment again with the consistency step. This is shown as 'CLP' in the figures. As expected, the error substantially decreased. The result is most substantial with larger overlapping views for the AOL covering designs.

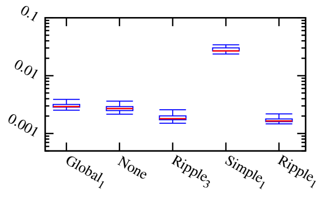
Figure 4 shows the impact of the performing the non-negativity step. Not using non-negativity ('None') has 2 to 4 times more error when compared with applying non-negativity. Simply change negative counts to zero(Simple) performs even worse, as this introduces systematic bias. Applying the naive non-negativity procedure of subtracting the same constant globally (Global) provide some improvement. Ripple method produces the best result. On the other hand, Consistency + Ripple + Consistency (Ripple$_1$) performs as well as repeating (Ripple + Consistency) 3 times after the initial Consistency step (Ripple$_3$).

5.4 Comparing Different Views

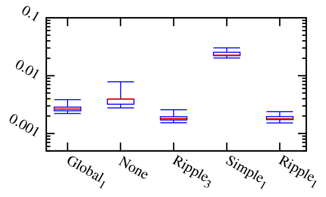
We now compare the accuracy due to choosing different views. The results for Kosarak are shown in Figure 6. Results for AOL are similar and are omitted due to space limitation. We use purple stars to denote the noise error computed from Equation 5. We can see that multiple covering designs with different ℓ values perform similarly. Choosing $\ell = 8$, while not optimal in all cases, perform reasonably well overall. We can also see that covering designs with $t = 3$ tend to have tighter band of errors than those covering only pairs. It also appears that choosing noise error to be around 0.002 would perform quite well.

5.5 Comparison on MCHAIN

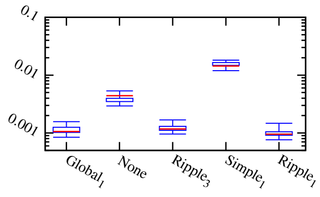
Figure 5 illustrates the L_2 error on datasets generated by different order markov chain. For i'th order markov chain,



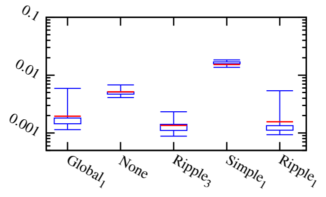
(a) Kosarak, $k = 6$



(b) Kosarak, $k = 8$

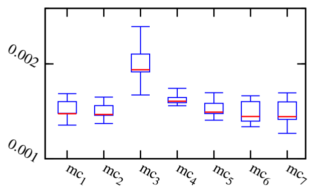


(c) AOL, $k = 6$

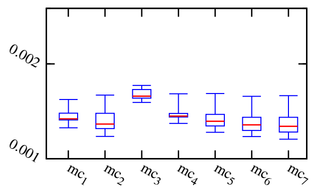


(d) AOL, $k = 8$

Figure 4: Comparing L_2 error for non-negativity methods on Kosarak using $\mathbf{V} = C_3(8, 106)$ and on AOL using $\mathbf{V} = C_2(8, 42)$, with $\epsilon = 1.0$. *Ripple*$_i$ applies the *Ripple* method i times. *None* keeps negative values unchanged. *Simple* changes all negative counts to zero. *Global* changes negative counts to zero, and then also subtracts a value from positive counts to maintain total count unchanged.



(a) MChain, $k = 6$



(b) MChain, $k = 8$

Figure 5: Comparing L_2 error on mc$_i$'s, the i'th order markov chain datasets, using $\mathbf{V} = C_2(8, 72)$, with $\epsilon = 1.0$.

the value of current state is only denpendent on the previous i states. In the experiment, we are only using consecutive queries so that the queries demonstrate all interdependencies. We use $\mathbf{V} = C_2(8, 72)$. The results show that even though only pairs are covered, the results are quite accurate even for larger orders. Interestingly, the 3'rd order markov chain produces largest error. To understand this, observe that i'th order results in $i+1$ attributes being highly correlated. For mc$_1$, all correlations are covered. For mc$_2$, while we do not guarantee that triples are covered, we guarantee that for any triple a_1, a_2, a_3, all three pairs are covered, and this seems to be sufficient to recover the joint distribution accurately. For mc$_3$, we have high correlation among 4 attributes, and covering only pairs results in noticeably larger (though still quite) errors. For even higher orders, the dependency among attributes weakens as each attribute is affected by the joint effect of more attributes, and the errors are quite small.

6. RELATED WORK

The notion of differential privacy was introduced in [6, 11, 9]. The method of adding Laplacian noise scaled with the sensitivity, which we use to generate view marginals, is proposed in [11]. Most closely related are previous approaches of releasing marginal tables with differential privacy. Several of these have been examined in detail in Section 3, including [4, 28, 22, 8, 15, 16].

There are a number of negative results regarding generating marginal tables. In [29], it is shown that it is computationally infeasible to construct a synthetic dataset that approximates 2-way marginals. Because we are constructing a synopsis, without trying to generate synthetic dataset from it, this negative result does not apply here. In [20], it is proven that when releasing all k-way marginals, if distortion is $o(\min\{\sqrt{n}/\log^{k^2+k+1} n, \sqrt{d^{k-1}}/\log^{k^2+3k-1} n\})$ per entry, then an adversary can reconstruct large fraction of the sensitive attribute entries, thus violating privacy. Note that the Direct method adds $O(\sqrt{d^k})$, and is thus "close to optimal", ignoring the inverse poly-logarithmic term. For $n \approx 2^{20}$, $d = 45$, $k = 6$, the above lowerbound becomes $\min\{2^{10}/20^{43}, 45^3/20^{53}\}$, which is about 10^{-64}. Thus the asymptotic lowerbound does not rule out the ability of accurately publishing k-way marginals in practice.

We use the concept of using constrained inference to post-process noisy results to improve accuracy introduced in [19], Qardaji and Li [27] present a method that employs recursive partitioning and summarization for multi-dimensional data publishing. This technique does not scale with larger d. Li et al. [24] use sets of attributes, similar to views, to release frequent itemsets in a differentially private manner. The views in that paper, however, are constructed using frequent items, require no consistency, and do not cover all attributes.

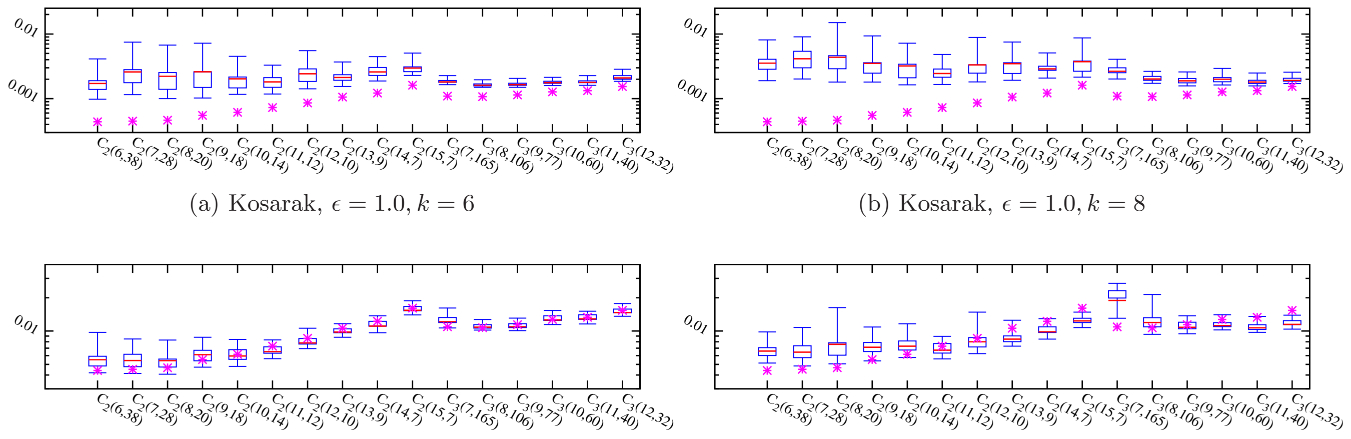
7. CONCLUSION

We have presented PriView: a practical method of computing marginal tables for datasets with more than a few dimensions. Instead of directly generating all k-way marginals, PriView generates marginals for a strategically chosen sets of attributes, which we call views. PriView combines existing ideas from different fields with new techniques developed in the paper. We have conducted extensive experiments on real and synthetic datasets, and shown that PriView advances the current sate of the art.

Acknowledgement. This paper is based upon work supported by the United States National Science Foundation under Grant No. 1116991. We would also like to thank the reviewers for their helpful comments.

8. REFERENCES

[1] AOL search log dataset. `http://www.gregsadetsky.com/aol-data/`.
[2] Frequent itemset mining dataset repository. `http://fimi.ua.ac.be/data/`.
[3] MSNBC.com anonymous web data data set. `http://archive.ics.uci.edu/ml/datasets/MSNBC.com+Anonymous+Web+Data`.
[4] 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 *PODS'07*, pages 273–282, 2007.
[5] Y. M. Bishop, S. E. Fienberg, and P. W. Holl. *Discrete Multivariate Analysis: Theory and Practice*. Springer, 2007.



(a) Kosarak, $\epsilon = 1.0, k = 6$

(b) Kosarak, $\epsilon = 1.0, k = 8$

(c) Kosarak, $\epsilon = 0.1, k = 6$

(d) Kosarak, $\epsilon = 0.1, k = 8$

Figure 6: Comparing L_2 error for different views on the Kosarak dataset

[6] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the SuLQ framework. In *PODS*, pages 128–138, 2005.

[7] M. Cheraghchi, A. Klivans, P. Kothari, and H. K. Lee. Submodular functions are noise stable. In *SODA*, pages 1586–1592, 2012.

[8] B. Ding, M. Winslett, J. Han, and Z. Li. Differentially private data cubes: optimizing noise sources and consistency. In *SIGMOD*, pages 217–228, 2011.

[9] C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.

[10] C. Dwork. A firm foundation for private data analysis. *Commun. ACM*, 54(1):86–95, 2011.

[11] C. Dwork, F. Mcsherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284. Springer, 2006.

[12] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[13] D. Gordon. La jolla covering repository, Nov. 2012. http://www.ccrwest.org/cover.html.

[14] D. M. Gordon, G. Kuperberg, and O. Patashnik. New constructions for covering designs. *J. Combin. Designs*, 3:269–284, 1995.

[15] A. Gupta, M. Hardt, A. Roth, and J. Ullman. Privately releasing conjunctions and the statistical query barrier. In *STOC*, pages 803–812, 2011.

[16] M. Hardt, K. Ligett, and F. McSherry. A simple and practical algorithm for differentially private data release. In *NIPS*, pages 2348–2356, 2012.

[17] M. Hardt and G. N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS*, pages 61–70. IEEE, 2010.

[18] M. Hardt, G. N. Rothblum, and R. A. Servedio. Private data release via learning thresholds. In *SODA*, pages 168–187, 2012.

[19] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3:1021–1032, September 2010.

[20] S. P. Kasiviswanathan, M. Rudelson, A. Smith, and J. Ullman. The price of privately releasing contingency tables and the spectra of random matrices with correlated rows. In *STOC*, pages 775–784, 2010.

[21] J. Lei. Differentially private m-estimators. In *NIPS*, pages 361–369, 2011.

[22] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor. Optimizing linear counting queries under differential privacy. In *PODS*, pages 123–134, New York, NY, USA, 2010. ACM.

[23] C. Li and G. Miklau. An adaptive mechanism for accurate query answering under differential privacy. *PVLDB*, 5(6):514–525, Feb. 2012.

[24] N. Li, W. H. Qardaji, D. Su, and J. Cao. PrivBasis: Frequent itemset mining with differential privacy. *PVLDB*, 5(11):1340–1351, 2012.

[25] J. Lin. Divergence measures based on the shannon entropy. *IEEE Trans. Inf. Theory*, 37(1):145–151, 1991.

[26] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, 2007.

[27] W. Qardaji and N. Li. Recursive partitioning and summarization: A practical framework for differential private data publishing. In *ASIACCS*, 2012.

[28] J. Thaler, J. Ullman, and S. Vadhan. Faster algorithms for privately releasing marginals. In *ICALP*, pages 810–821, 2012.

[29] J. Ullman and S. Vadhan. PCPs and the hardness of generating private synthetic data. In *TCC*, pages 400–416, 2011.

[30] O. V. Usatenko and V. A. Yampol'skii. Binary n-step markov chains and long-range correlated systems. *Phys. Rev. Lett.*, 90:110601, Mar 2003.

[31] G. Yuan, Z. Zhang, M. Winslett, X. Xiao, Y. Yang, and Z. Hao. Low-rank mechanism: Optimizing batch queries under differential privacy. *PVLDB*, 5(11):1352–1363, July 2012.