PrivBayes: Private Data Release via Bayesian Networks

Jun Zhang[1] Graham Cormode[2] Cecilia M. Procopiuc[3] Divesh Srivastava[3] Xiaokui Xiao[1]

[1]Nanyang Technological University
{jzhang027, xkxiao}@ntu.edu.sg

[2]University of Warwick
g.cormode@warwick.ac.uk

[3]AT&T Labs – Research
{magda, divesh}@research.att.com

ABSTRACT

Privacy-preserving data publishing is an important problem that has been the focus of extensive study. The state-of-the-art goal for this problem is differential privacy, which offers a strong degree of privacy protection without making restrictive assumptions about the adversary. Existing techniques using differential privacy, however, cannot effectively handle the publication of high-dimensional data. In particular, when the input dataset contains a large number of attributes, existing methods require injecting a prohibitive amount of noise compared to the signal in the data, which renders the published data next to useless.

To address the deficiency of the existing methods, this paper presents PRIVBAYES, a differentially private method for releasing high-dimensional data. Given a dataset D, PRIVBAYES first constructs a Bayesian network $\mathcal{N}$, which (i) provides a succinct model of the correlations among the attributes in D and (ii) allows us to approximate the distribution of data in D using a set $\mathcal{P}$ of low-dimensional marginals of D. After that, PRIVBAYES injects noise into each marginal in $\mathcal{P}$ to ensure differential privacy, and then uses the noisy marginals and the Bayesian network to construct an approximation of the data distribution in D. Finally, PRIVBAYES samples tuples from the approximate distribution to construct a synthetic dataset, and then releases the synthetic data. Intuitively, PRIVBAYES circumvents the curse of dimensionality, as it injects noise into the low-dimensional marginals in $\mathcal{P}$ instead of the high-dimensional dataset D. Private construction of Bayesian networks turns out to be significantly challenging, and we introduce a novel approach that uses a surrogate function for mutual information to build the model more accurately. We experimentally evaluate PRIVBAYES on real data, and demonstrate that it significantly outperforms existing solutions in terms of accuracy.

Categories and Subject Descriptors

H.2.7 [**Database Administration**]: Security, integrity & protection

Keywords

Differential privacy; synthetic data generation; Bayesian network

SIGMOD'14, June 22–27, 2014, Snowbird, UT, USA.
Copyright 2014 ACM 978-1-4503-2376-5/14/06 ...$15.00.

1. INTRODUCTION

The problem of privacy-preserving data publishing (PPDP) has become increasingly important in recent years. Often, we encounter situations where a data owner wishes to make available a data set without revealing private, sensitive information. For example, this arises in the case of revealing detailed demographic information about citizens (census), patients (health data), investors (financial data), and so on. In each example, there are many potential uses and users for the data: for social analysis, for medical research, for freedom-of-information and other legal disclosure reasons. The canonical case in PPDP is when the database can be modeled as a table, where each row may contain information about an individual (say, details of their medical status, or employment information). Then, the aim is to release some manipulated version of this information so that this can still be used for the intended purpose, but the privacy of individuals in the database is preserved.

Following many attempts to formally define the requirements of privacy, the current state-of-the-art solution is to seek the differential privacy guarantee [16]. Informally, this model requires that what can be learned from the released data is (approximately) the same, whether or not any particular individual was included in the input database. This model offers strong privacy protection, and does not make any limiting assumptions about the power of the notional adversary: it remains a strong model even in the face of an adversary with much background knowledge and reasoning power.

Putting differential privacy into practice remains a challenging problem. Since its proposal, there have been many efforts to develop mechanisms and processes for data release for different kinds of input database, and for different objectives for the end use. However, it seems that all existing techniques encounter difficulties when trying to release even moderately high-dimensional data – that is, an input table with half a dozen columns or more. The reasons for these problems are two-fold:

- *Output Scalability:* Most algorithms (see, e.g., [37]) either explicitly or implicitly represent the database as a vector x of size equal to the *domain size*, that is, the product of cardinalities of the attributes. For many natural data sets, the domain size m is orders of magnitude larger than the data size n [13]. Hence, these algorithms become inapplicable for any realistic dataset with a moderate-to-high number of attributes. For example, a million row table with ten attributes, each of which has 20 possible values, results in a domain size (and hence an output size) of $m = 20^{10} \approx 10TB$, which is very unwieldy and slow to use compared to the input which can be measured in megabytes.

- *Signal-to-noise ratio:* When the high dimensional database is represented as a vector x, the average count in each entry, given by n/m, is typically very small. Once noise is added to x (or some transformation of it) to obtain another vector x^*, the noise completely dominates the original signal, making the published vector x^* next to useless. For example, if the table above has size $n = 1M$, the average entry count is $n/m = 10^{-7}$. By contrast, the average noise injected to achieve, e.g., differential privacy with parameter $\varepsilon = 0.1$ has expected magnitude around 10. Even if the data is skewed in the domain space, i.e., there are some entries $x[i]$ with high counts, such peaks are infrequent and so the vast majority of published values $x^*[i]$ are useless.

1.1 Related Work

A full survey of methods to realize differential privacy is beyond the scope of this work. Here, we identify the most related efforts, and discuss why they cannot fully solve the problems above.

Initial efforts released projections of the data on subsets of dimensions, via Fourier decompositions [5]. This reduces the impact of higher dimensionality, but requires the data owner to determine (somehow) which set of dimensions are of interest, and for the data to be mapped into a space where a Fourier decomposition can be computed. Subsequent work followed this template, for example by searching for meaningful "subcubes" of the datacube representation to release privately [15]. These can be aggregated to answer lower-dimensional cube queries, where the signal-to-noise ratio in each cell is significantly higher than in the original domain. A major limitation is that the running time is exponential in the dimensionality of the domain, making it inapplicable for all but small sets. Accuracy is improved by additional post-processing of the output to restore "consistency" of the counts mandated by the structure (e.g. using the fact that counts in a hierarchy should sum to the count of an ancestor) [5, 15, 21]; however, this improvement does not overcome the inherent error incurred in high dimensions.

Another approach is to use data reduction techniques to avoid dimensionality issues. For example, [13] proposes various sampling mechanisms to reduce the size of (and time to produce) the output x^*, while approximately preserving the accuracy of subset-sum queries. However, the accuracy guarantee is with respect to the accuracy of using the entire vector x^*, rather than the original vector x, which degrades rapidly with data dimensionality. The approach in [12] tries to keep the domain size of the output small, and the density of data within the new domain high, by adaptively grouping some of the attribute values. In the example above, suppose that on each of the ten attributes, we grouped the 20 values into two groups. Thus, we reduce the output size from 20^{10} to $2^{10} \approx 1MB$. This coarser grid representation of the data loses precision, and in this example still leads to small average counts of the order of 1. Cormode et al. [12] address the latter problem by using spatial decompositions to define irregular grids over the domain of x, such that the count $x[i]$ in each grid cell is sufficiently large. This makes sense only for range queries, and requires all attributes to have an ordering.

Other approaches find other ways to recast the input data and release it, so that the noise from the privacy transformation has less impact on certain statistics. In particular, Xiao et al. [37] make use of the wavelet transformation of data. This addresses range queries, and means that the noise incurred by a range query scales proportionately to the logarithm of its length, rather than to its length directly. The value of this is confined primarily to low-dimensional databases where range queries are anticipated. More generally, the matrix mechanism of Li and Miklau [25, 26] and subsequent related approaches [20, 38, 39] take in a query workload (expressed in terms of a weighted set of inner-product queries), and seek to release a version of the database that minimizes the noise for these queries. The cost of this optimization can be high, and critically assumes the foreknowledge of the query distribution.

The above discussion focuses on methods that produce an output in a general form that can be used flexibly for a variety of subsequent analyses. There is also a large class of results that instead generate a description of the output of a specific algorithm computed under privacy, for example the result of building a classifier for the data. Prominent examples include the work of McSherry and Mironov [30] to mask the preferences of individual raters in recommendation systems; Rastogi and Nath [34] to release time-series data based on leading Fourier coefficients; and McSherry and Mahajan [29] for addressing common queries over network trace data. Differential privacy has also been applied to other sophisticated data analysis tasks, e.g., coresets for summarizing geometric data [18], building classifiers in the form of decision trees [19], and support vector machines [35], and mining the occurrence of frequent patterns [6, 27].

Some frameworks have been proposed to solve a class of optimization problems. For example, Chaudhuri et al. [8, 9] and Kifer et al. [22] consider differentially private convex empirical risk minimization, which can be applied to a wide range of optimization problems (e.g., logistic regression and support vector machines). Zhang et al. [40] propose the *PrivGene* framework, which is a combination of genetic algorithms and an enhanced version of the *exponential mechanism* for differentially private model fitting. Smith et al. [33, 36] and Mohan et al. [32] respectively present and implement the *sample and aggregate* framework that can be used for any analysis task whose results are not affected by the number of records in the database. While this class of methods obtains generally good results for the target problem, it requires fixing this objective at the time of data release, and limits the applicability of the output for other uses.

1.2 Our Contributions

In this paper, we propose a powerful solution to the problem of publishing differentially private high-dimensional data. Unlike the bulk of prior work, which focuses on optimizing the output for specific workloads (e.g., range queries, cube queries), we aim to approximate the high-dimensional distribution of the original data with a data-dependent set of well-chosen low-dimensional distributions, in the belief that, for a sufficiently accurate approximation, the resulting data will maintain high accuracy for almost any type of (linear or non-linear) query. Since our approach is query-independent, many different queries can be evaluated (accurately) on the same set of released data. Query-independence means that our approach may be weaker than approaches that target a particular query set; however, we show empirically that the gap is small or non-existent in many natural cases. By working in low-dimensional spaces, we avoid the signal-to-noise problem. Although we compare to the full-dimensional distribution for evaluation purposes, our approach never needs to compute this explicitly, thus avoiding the scalability problem.

To achieve this goal, we start from the well-known *Bayesian network* model, which is widely studied in the statistical and machine learning communities [23]. Bayesian networks combine low-dimensional distributions to approximate the full-dimensional distribution of a data set, and are a simple but powerful example of a graphical model. Our algorithm, dubbed PrivBayes, consists of the following steps:

1. (*Network learning*) We describe how to compute a differentially private Bayesian network that approximates the full-dimensional distribution via the Exponential Mechanism (EM). This step requires new theoretical insights, which are described in Section 3. We improve on this basic approach by defining a new quality function for use in EM, which results in significantly better networks

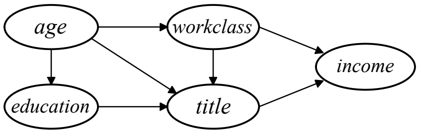


Figure 1: A Bayesian network $\mathcal{N}_1$ over five attributes

being found. The definition and computation of this function are one of our main technical contributions; see Section 4.

2. (*Distribution learning*) We explain how to compute the necessary differentially private distributions of the data in the subspaces of the Bayesian network, via the Laplace Mechanism.

3. (*Data synthesis*) We show how to generate synthetic data from the differentially private Bayesian network, without explicitly materializing the global distribution.

In Section 5, we provide an extensive experimental evaluation of the accuracy of the synthetic datasets generated above, over workloads of linear and non-linear queries. In each case, we compare to prior methods specifically designed to optimize the accuracy for that type of workload. Our experiments show that PRIVBAYES is often *more* accurate than any prior method, even though it is not optimized for any specific type of query. When PRIVBAYES is less accurate than some prior method, the accuracy loss is small and, we believe, an acceptable tradeoff, since PRIVBAYES offers a generic solution that does not require prior knowledge of the workload and works well on many different types of queries.

2. PRELIMINARIES

This section reviews two concepts closely related to our work, namely, differential privacy and Bayesian networks.

2.1 Differential Privacy

Let D be a sensitive dataset to be published. Differential privacy requires that, prior to D's release, it should be modified using a randomized algorithm G, such that the output of G does not reveal much information about any particular tuple in D. The formal definition of differential privacy is as follows:

DEFINITION 1 (ε-DIFFERENTIAL PRIVACY [17]). *A randomized algorithm G satisfies ε-differential privacy, if for any two datasets D_1 and D_2 that differ only in one tuple, and for any possible output O of G, we have*

$$\Pr\left[G(D_1) = O\right] \leq e^{\varepsilon} \cdot \Pr\left[G(D_2) = O\right], \tag{1}$$

where $\Pr[\cdot]$ *denotes the probability of an event.*

In what follows, we say that two datasets are neighboring if they differ in only one tuple, i.e., the values of one tuple are changed while the rest are identifical. While there are many approaches to achieving differential privacy, we rely on the two best known and most-widely used, namely, the *Laplace mechanism* [17] and the *exponential mechanism* [31].

The Laplace mechanism releases the result of a function F that takes as input a dataset and outputs a set of numeric values. Given F, the Laplace mechanism transforms F into a differentially private algorithm, by adding i.i.d. noise (denoted as η) into each output value of F. The noise η is sampled from a *Laplace distribution* $\mathrm{Lap}(\lambda)$ with the following pdf: $\Pr[\eta = x] = \frac{1}{2\lambda}e^{-|x|/\lambda}$. Dwork et al. [17] prove that the Laplace mechanism ensures ε-differential privacy if $\lambda \geq S(F)/\varepsilon$, where $S(F)$ is the *sensitivity* of F:

Table 1: The attribute-parent pairs in $\mathcal{N}_1$

i	X_i	Π_i
1	*age*	$\emptyset$
2	*education*	{*age*}
3	*workclass*	{*age*}
4	*title*	{*age*, *education*, *workclass*}
5	*income*	{*workclass*, *title*}

DEFINITION 2 (SENSITIVITY [17]). *Let F be a function that maps a dataset into a fixed-size vector of real numbers. The* sensitivity *of F is defined as*

$$S(F) = \max_{D_1, D_2} \|F(D_1) - F(D_2)\|_1, \tag{2}$$

where $\|\cdot\|_1$ denotes the L_1 norm, and D_1 and D_2 are any two neighboring datasets.

Intuitively, $S(F)$ measures the maximum possible change in F's output when we modify one arbitrary tuple in F's input.

When F's output is categorical instead of numeric, the Laplace mechanism does not apply, but the exponential mechanism [31] can be used instead. The exponential mechanism releases a differentially private version of F, by sampling from F's output domain Ω. The sampling probability for each $\omega \in \Omega$ is determined based on a user-specified *score function* f_s, which takes as input any dataset D and any element $\omega \in \Omega$, and outputs a numeric score $f_s(D, \omega)$ that measures the quality of ω: a larger score indicates that ω is a better output with respect to D. More specifically, given a dataset D, the exponential mechanism samples $\omega \in \Omega$ with a probability proportional to $\exp(f_s(D, \omega)/2\Delta)$, where Δ is a scaling factor that controls the degree of privacy protection. McSherry and Talwar [31] show that the exponential mechanism achieves ε-differential privacy if $\Delta \geq S(f_s)/\varepsilon$, where $S(f_s)$ is defined as:

$$S(f_s) = \max_{D_1, D_2, \omega'} \left|f_s(D_1, \omega') - f_s(D_2, \omega')\right|, \tag{3}$$

for D_1 and D_2 any two neighboring datasets, and ω' any element in Ω. For convenience, we also refer to $S(f_s)$ as the *sensitivity* of f_s, as it is similar in form to sensitivity as defined above.

Both mechanisms can be applied quite generally; however, to be effective we seek to ensure that the noise introduced does not outweigh the signal in the data, and that it is computationally efficient to apply the mechanism. This requires a careful design of what functions to use in the mechanisms.

2.2 Bayesian Network

Let $\mathcal{A}$ be the set of attributes on a dataset D, and d be the size of $\mathcal{A}$. A *Bayesian network* on $\mathcal{A}$ is a way to compactly describe the (probability) distribution of the attributes in terms of other attributes. Formally, a Bayesian network is a directed acyclic graph (DAG) that (i) represents each attribute in $\mathcal{A}$ as a node, and (ii) models conditional independence among attributes in $\mathcal{A}$ using directed edges. As an example, Figure 1 shows a Bayesian network over a set $\mathcal{A}$ of five attributes, namely, *age*, *education*, *workclass*, *title*, and *income*. For any two attributes $X, Y \in \mathcal{A}$, there exist three possibilities for the relationship between X and Y:

Case 1: Direct dependence. There is an edge between X and Y, say, from Y to X. This indicates that for any tuple in D, its distribution on X is determined (in part) by its value on Y. We define Y as a *parent* of X, and refer to the set of all parents of X as its *parent set*. For example, in Figure 1, the edge from *workclass* to *income* indicates that the income distribution depends on the type of job (and also on title).

Table 2: Table of notations

Notation	Description
D	A sensitive dataset to be published
n	The number of tuples in D
$\mathcal{A}$	The set of attributes in D
d	The number of attributes in $\mathcal{A}$
$\mathcal{N}$	A Bayesian network over $\mathcal{A}$
$\Pr[\mathcal{A}]$	The distribution of tuples in D
$\Pr_{\mathcal{N}}[\mathcal{A}]$	An approximation of $\Pr[\mathcal{A}]$ defined by $\mathcal{N}$
$\mathrm{dom}(X)$	The domain of random variable X

Case 2: Weak conditional independence. There is a path (but no edge) between Y and X. Assume without loss of generality that the path goes from Y to X. Then, X and Y are *conditionally independent* given X's parent set. For instance, in Figure 1, there is a two-hop path from *age* to *income*, and the parent set of *income* is $\{workclass, title\}$. This indicates that, given workclass and job title of an individual, her income and age are conditionally independent.

Case 3: Strong conditional independence. There is no path between Y and X. Then, X and Y are conditionally independent given any of X's and Y's parent sets.

Formally, a Bayesian network $\mathcal{N}$ over $\mathcal{A}$ is defined as a set of d *attribute-parent (AP) pairs*, $(X_1, \Pi_1), \ldots, (X_d, \Pi_d)$, such that

1. Each X_i is a unique attribute in $\mathcal{A}$;

2. Each Π_i is a subset of the attributes in $\mathcal{A} \setminus \{X_i\}$. We say that Π_i is the parent set of X_i in $\mathcal{N}$;

3. For any $1 \le i < j \le d$, we have $X_j \notin \Pi_i$, i.e., there is no edge from X_j to X_i in $\mathcal{N}$. This ensures that the network is acyclic, namely, it is a DAG.

We define the *degree* of $\mathcal{N}$ as the maximum size of any parent set Π_i in $\mathcal{N}$. For example, Table 1 shows the AP pairs in the Bayesian network $\mathcal{N}_1$ in Figure 1; $\mathcal{N}_1$'s degree equals 3, since the parent set of any attribute in $\mathcal{N}_1$ has a size at most three.

Let $\Pr[\mathcal{A}]$ denote the full distribution of tuples in database D. The d AP pairs in $\mathcal{N}$ essentially define a way to approximate $\Pr[\mathcal{A}]$ with d conditional distributions $\Pr[X_1 \mid \Pi_1], \Pr[X_2 \mid \Pi_2], \ldots, \Pr[X_d \mid \Pi_d]$. In particular, under the assumption that any X_i and any $X_j \notin \Pi_i$ are conditionally independent given Π_i, we have

$$\begin{aligned}\Pr[\mathcal{A}] &= \Pr[X_1, X_2, \ldots, X_d] \\ &= \Pr[X_1] \cdot \Pr[X_2 \mid X_1] \cdot \Pr[X_3 \mid X_1, X_2] \ldots \Pr[X_d \mid X_1, \ldots X_{d-1}] \\ &= \prod_{i=1}^{d} \Pr[X_i \mid \Pi_i]. \end{aligned} \tag{4}$$

Let $\Pr_{\mathcal{N}}[\mathcal{A}] = \prod_{i=1}^{d} \Pr[X_i \mid \Pi_i]$ be the above approximation of $\Pr[\mathcal{A}]$ defined by $\mathcal{N}$. Intuitively, if $\mathcal{N}$ accurately captures the conditional independence among the attributes in $\mathcal{A}$, then $\Pr_{\mathcal{N}}[\mathcal{A}]$ would be a good approximation of $\Pr[\mathcal{A}]$. In addition, if the degree of $\mathcal{N}$ is small, then the computation of $\Pr_{\mathcal{N}}[\mathcal{A}]$ is relatively simple as it requires only d low-dimensional distributions $\Pr[X_1 \mid \Pi_1], \Pr[X_2 \mid \Pi_2], \ldots, \Pr[X_d \mid \Pi_d]$. Low-degree Bayesian networks are the core of our solution to release high-dimensional data. Table 2 shows notation that will be frequently used in this paper.

3. SOLUTION OVERVIEW

This section presents an overview of PRIVBAYES, our solution for releasing a high-dimensional dataset D in an ε-differentially private manner. PRIVBAYES runs in three phases:

Algorithm 1 NoisyConditionals $(D, \mathcal{N}, k)$: returns $\mathcal{P}^*$

```
1: Initialize $\mathcal{P}^* = \emptyset$
2: for $i = k+1$ to $d$ do
3:     Materialize the joint distribution $\Pr[X_i, \Pi_i]$
4:     Generate differentially private $\Pr^*[X_i, \Pi_i]$ by adding Laplace noise $\mathrm{Lap}\left(\frac{4 \cdot (d-k)}{n \cdot \varepsilon}\right)$
5:     Set negative values in $\Pr^*[X_i, \Pi_i]$ to 0 and normalize;
6:     Derive $\Pr^*[X_i \mid \Pi_i]$ from $\Pr^*[X_i, \Pi_i]$; add it to $\mathcal{P}^*$
7: for $i = 1$ to $k$ do
8:     Derive $\Pr^*[X_i \mid \Pi_i]$ from $\Pr^*[X_{k+1}, \Pi_{k+1}]$; add it to $\mathcal{P}^*$
9: return $\mathcal{P}^*$
```

1. Construct a k-degree Bayesian network $\mathcal{N}$ over the attributes in D, using an $(\varepsilon/2)$-differentially private method. (k is a small value that can be chosen automatically by PRIVBAYES.)

2. Use an $(\varepsilon/2)$-differentially private algorithm to generate a set of *conditional distributions* of D, such that for each AP pair (X_i, Π_i) in $\mathcal{N}$, we have a noisy version of the conditional distribution $\Pr[X_i \mid \Pi_i]$. (We denote this noisy distribution as $\Pr^*[X_i \mid \Pi_i]$.)

3. Use the Bayesian network $\mathcal{N}$ (constructed in the first phase) and the d noisy conditional distributions (constructed in the second phase) to derive an approximate distribution of the tuples in D, and then sample tuples from the approximate distribution to generate a synthetic dataset D^*.

In short, PRIVBAYES utilizes a low-degree Bayesian network $\mathcal{N}$ to generate a synthetic dataset D^* that approximates the high dimensional input data D. The construction of $\mathcal{N}$ is highly nontrivial, as it requires carefully selecting AP pairs and the value of k to derive a close approximation of D without violating differential privacy. By contrast, the second and third phases of PRIVBAYES are relatively straightforward. In the following, we will clarify the details of these phases, and prove the privacy guarantee of PRIVBAYES; the algorithm for PRIVBAYES's first phase will be elaborated in Section 4.

Generation of Noisy Conditional Distributions. Suppose that we are given a k-degree Bayesian network $\mathcal{N}$. To construct the approximate distribution $\Pr_{\mathcal{N}}[\mathcal{A}]$, we need d conditional distributions $\Pr[X_i \mid \Pi_i]$ ($i \in [1, d]$), as shown in Equation (4). Algorithm 1 illustrates how the distributions specified by our algorithm can be derived in a differentially private manner. In particular, for any $i \in [k+1, d]$, the algorithm first materializes the joint distribution $\Pr[X_i, \Pi_i]$ (Line 3), and then injects Laplace noise into $\Pr[X_i, \Pi_i]$ to obtain a noisy distribution $\Pr^*[X_i, \Pi_i]$ (Line 4-5). To enforce the fact that these are probability distributions, all negative numbers in $\Pr^*[X_i, \Pi_i]$ are set to zero, then all values are normalized to maintain a total probability mass of 1 (Line 5).[1] After that, based on $\Pr^*[X_i, \Pi_i]$, the algorithm derives a noisy version of the conditional distribution $\Pr[X_i \mid \Pi_i]$, denoted as $\Pr^*[X_i \mid \Pi_i]$ (Line 6). The scale of the Laplace noise added to $\Pr[X_i, \Pi_i]$ is set to $4(d-k)/n\varepsilon$, which ensures that the generation of $\Pr^*[X_i, \Pi_i]$ satisfies $(\varepsilon/2(d-k))$-differential privacy, since $\Pr[X_i, \Pi_i]$ has sensitivity $2/n$. Meanwhile, the derivation of $\Pr^*[X_i \mid \Pi_i]$ from $\Pr^*[X_i, \Pi_i]$ does not incur any privacy cost, as it only relies on $\Pr^*[X_i, \Pi_i]$ instead of the input data D.

Overall, Lines 2-7 of Algorithm 1 construct $(d-k)$ noisy conditional distributions $\Pr^*[X_i \mid \Pi_i]$ ($i \in [k+1, d]$), and they satisfy $(\varepsilon/2)$-differential privacy, since each $\Pr^*[X_i \mid \Pi_i]$ is $(\varepsilon/2(d-k))$-differentially private. This is due to the composability property of

[1] More generally, we could apply additional post-processing of distributions, in the spirit of [5, 15, 21], to reflect the fact that lower degree distributions should be consistent. For simplicity and brevity, we omit such optimizations from this presentation.

differential privacy [16]. In particular, composability indicates that when a set of m algorithms satisfy differential privacy with parameters $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m$, respectively, the set of algorithms as a whole satisfies $(\sum_i \varepsilon_i)$-differential privacy.

After $\Pr^*[X_{k+1} \mid \Pi_{k+1}], \ldots, \Pr^*[X_d \mid \Pi_d]$ are constructed, Algorithm 1 proceeds to generate $\Pr^*[X_i \mid \Pi_i]$ $(i \in [1,k])$. This generation, however, does not require any additional information from the input data D. Instead, we derive $\Pr^*[X_i \mid \Pi_i]$ $(i \in [1,k])$ directly from $\Pr^*[X_{k+1}, \Pi_{k+1}]$, which has been computed in Lines 2-7 of Algorithm 1. Such derivation is feasible, since our algorithm for constructing the Bayesian network $\mathcal{N}$ (to be clarified in Section 4) ensures that $X_i \in \Pi_{k+1}$ and $\Pi_i \subset \Pi_{k+1}$ for any $i \in [1,k]$. Since each $\Pr^*[X_i \mid \Pi_i]$ $(i \in [1,k])$ is derived from $\Pr^*[X_{k+1}, \Pi_{k+1}]$ without inspecting D, the construction of $\Pr^*[X_i \mid \Pi_i]$ does not incur any privacy overhead. Therefore, Algorithm 1 as a whole is $(\varepsilon/2)$-differentially private. Example 1 illustrates Algorithm 1.

EXAMPLE 1. *Suppose that we are given a* 2*-degree Bayesian network* $\mathcal{N}$ *over a set of four attributes* $\{A,B,C,D\}$*, with* 4 *AP pairs:* $(A,\emptyset), (B,\{A\}), (C,\{A,B\})$*, and* $(D,\{A,C\})$*. Given* $\mathcal{N}$*, Algorithm 1 constructs two noisy joint distributions* $\Pr^*[A,B,C]$ *and* $\Pr^*[A,C,D]$*. Based on* $\Pr^*[A,C,D]$*, Algorithm 1 derives a noisy conditional distribution* $\Pr^*[D \mid A,C]$*. In addition, the algorithm uses* $\Pr^*[A,B,C]$ *to derive three other conditional distributions* $\Pr^*[A]$*,* $\Pr^*[B \mid A]$*, and* $\Pr^*[C \mid A,B]$*. Given these four conditional distributions, the input tuple distribution is approximated as*

$$\Pr^*_{\mathcal{N}}[A,B,C,D] = \Pr^*[A] \cdot \Pr^*[B \mid A] \cdot \Pr^*[C \mid A,B] \cdot \Pr^*[D \mid A,C].$$

Generation of Synthetic Data. Even with the simple closed-form expression in Equation 4, it is still time and space consuming to directly sample from $\Pr^*_{\mathcal{N}}[\mathcal{A}]$ by computing the probability for each element in the domain of $\mathcal{A}$. Fortunately, the Bayesian network $\mathcal{N}$ provides a means to perform sampling efficiently without materializing $\Pr^*_{\mathcal{N}}[\mathcal{A}]$. As shown in Equation 4, we can sample each X_i from the conditional distribution $\Pr^*[X_i \mid \Pi_i]$ independently, without considering any attribute not in $\Pi_i \cup \{X_i\}$. Furthermore, the properties of $\mathcal{N}$ (discussed in Section 2.2) ensure that $X_j \notin \Pi_i$ for any $j > i$. Therefore, if we sample X_i $(i \in [1,d])$ in increasing order of i, then by the time X_j $(j \in [2,d])$ is to be sampled, we must have sampled all attributes in Π_j, i.e., we will be able to sample X_j from $\Pr^*[X_j \mid \Pi_j]$ given the previously sampled attributes. That is to say, the sampling of X_j does not require the full distribution $\Pr^*_{\mathcal{N}}[\mathcal{A}]$.

With the above sampling approach, we can generate an arbitrary number of tuples from $\Pr^*_{\mathcal{N}}[\mathcal{A}]$ to construct a synthetic database D^*. In this paper, we consider the size of D^* is set to n, i.e., the same as the number of tuples in the input data D.

Privacy Guarantee. The correctness of PRIVBAYES directly follows the composability property of differential privacy [16]. In particular, the first and second phases of PRIVBAYES require direct access to the input database, and each of them consumes $\varepsilon/2$ privacy budget. No access to the original database is invoked during the third (sampling) phase. The results of first two steps, i.e., the Bayesian network $\mathcal{N}$ and the set of noisy conditional distributions, are sufficient to generate the synthetic database D^*. Therefore, we have the following theorem.

THEOREM 1. PRIVBAYES *satisfies* ε*-differential privacy.*

4. PRIVATE BAYESIAN NETWORKS

This section presents our solution for constructing differentially private Bayesian networks. We will first introduce a non-private algorithm for Bayesian network construction (in Section 4.1), and then explain how the algorithm can be converted into a differentially private solution (in Sections 4.2 and 4.3).

Algorithm 2 GreedyBayes (D, k): returns $\mathcal{N}$

```
1: Initialize $\mathcal{N} = \emptyset$ and $V = \emptyset$
2: Randomly select an attribute $X_1$ from $\mathcal{A}$; add $(X_1, \emptyset)$ to $\mathcal{N}$; add $X_1$ to $V$
3: for $i = 2$ to $d$ do
4:     Initialize $\Omega = \emptyset$
5:     For each $X \in \mathcal{A} \backslash V$ and each $\Pi \in \binom{V}{k}$, add $(X, \Pi)$ to $\Omega$
6:     Select a pair $(X_i, \Pi_i)$ from $\Omega$ with the maximal mutual information $I(X_i, \Pi_i)$
7:     Add $(X_i, \Pi_i)$ to $\mathcal{N}$; add $X_i$ to $V$
8: return $\mathcal{N}$
```

4.1 Non-Private Methods

Suppose that we aim to construct a k-degree Bayesian network $\mathcal{N}$ on a dataset D containing a set $\mathcal{A}$ of attributes. Ideally, $\mathcal{N}$ should provide an accurate approximation of the tuple distribution in D, i.e., $\Pr_{\mathcal{N}}[\mathcal{A}]$ should be close to $\Pr[\mathcal{A}]$. A natural question is under what condition will $\Pr_{\mathcal{N}}[\mathcal{A}]$ closely approximate $\Pr[\mathcal{A}]$? We make use of standard notions from information theory to measure this. The *entropy* of a random variable X over its domain $\mathrm{dom}(X)$ is denoted by $H(X) = -\sum_{x \in \mathrm{dom}(X)} \Pr[X = x] \log \Pr[X = x]$,[2] and $I(\cdot,\cdot)$ denotes the *mutual information* between two variables as: $I(X, \Pi) =$

$$\sum_{x \in \mathrm{dom}(X)} \sum_{\pi \in \mathrm{dom}(\Pi)} \Pr[X = x, \Pi = \pi] \log \frac{\Pr[X = x, \Pi = \pi]}{\Pr[X = x] \Pr[\Pi = \pi]}, \quad (5)$$

where $\Pr[X, \Pi]$ is the joint distribution of X and Π, and $\Pr[X]$ and $\Pr[\Pi]$ are the marginal distributions of X and Π respectively. The *KL-divergence* [14] of $\Pr_{\mathcal{N}}[\mathcal{A}]$ from $\Pr[\mathcal{A}]$ measures the difference between the two probability distributions, and is defined by:

$$KL(\Pr[\mathcal{A}], \Pr_{\mathcal{N}}[\mathcal{A}]) = -\sum_{i=1}^{d} I(X_i, \Pi_i) + \sum_{i=1}^{d} H(X_i) - H(\mathcal{A}). \quad (6)$$

We seek a Bayesian network representation so that the KL-divergence between the original and the approximate distribution is small. In (6), the term $\sum_{i=1}^{d} H(X_i) - H(\mathcal{A})$ is solely decided by $\Pr[\mathcal{A}]$, which is fixed once the input database D is given. Hence, the KL-divergence of $\Pr_{\mathcal{N}}[\mathcal{A}]$ from $\Pr[\mathcal{A}]$ is small (in which case they closely approximate each other), if and only if $\sum_{i=1}^{d} I(X_i, \Pi_i)$ is maximized. Therefore, the construction of $\mathcal{N}$ can be modeled as an optimization problem, where we aim to choose a parent set Π_i for each attribute X_i in D to maximize $\sum_{i=1}^{d} I(X_i, \Pi_i)$.

For the case when $k = 1$, Chow and Liu show that greedily picking the next edge based on the maximum mutual information is optimal, leading to the celebrated notion of Chow-Liu trees [11]. However, as shown in [10], this optimization problem is NP-hard when $k > 1$. For this reason, heuristic algorithms (e.g., hill-climbing, genetic algorithms, and simulated annealing) are often employed in practice [28]. In the context of differential privacy, however, a different calculus applies: these methods incur a high cost in terms of sensitivity and so incur a large amount of noise. That is, these algorithms make many queries to the data, so that making them differentially private entails large perturbations which lead to poor overall accuracy. Therefore, we seek a new method that will imply less noise, and so give a better overall approximation when the noise is added. Thus we propose a greedy algorithm that makes fewer probes to the data, by extending the Chow-Liu approach to higher degrees, described in Algorithm 2.

In the beginning of the algorithm (Line 1), we initialize the Bayesian network $\mathcal{N}$ to an empty list of AP pairs. Let V be a set

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

that contains all attributes whose parent sets have been fixed in the partial construction of $\mathcal{N}$. As a next step, the algorithm randomly selects an attribute (denoted as X_1) from $\mathcal{A}$, and sets its parent set Π_1 to $\emptyset$ (Line 2). The rest of the algorithm consists of $d-1$ iterations (Lines 3-7), in each of which we greedily add into $\mathcal{N}$ an AP pair with a large mutual information. Specifically, the AP pair in each iteration is selected from a candidate set Ω that contains every AP pair (X, Π) satisfying two requirements:

1. $|\Pi| \leq k$, which ensures that $\mathcal{N}$ is a k-degree Bayesian network. This is ensured by choosing Π only from $\binom{V}{k}$, where $\binom{V}{k}$ denotes the set of all subsets of V with size $\min(k, |V|)$ (Lines 5-6).

2. $\mathcal{N}$ contains no edge from X_i to X_j for any $j < i$, which guarantees that $\mathcal{N}$ is a DAG. We ensure this condition by requiring that in the beginning of any iteration, V only contains the attributes whose parent sets have been decided in the previous iterations (Line 7). In other words, the parent set of X_i can only be a subset of $\{X_1, X_2, \ldots, X_{i-1}\}$, as a consequence of which $\mathcal{N}$ cannot contain any edge from X_i to X_j for any $j < i$.

Once the parent set of each attribute is decided, the algorithm terminates and returns the Bayesian network $\mathcal{N}$ (Line 9). The number of pairs considered in iteration i is $(d-i)\binom{i}{k}$, so summing over all iterations the cost is bounded by $d\sum_{i=1}^{d}\binom{i}{k} = d\binom{d+1}{k+1}$. This determines the asymptotic cost of the procedure. Note that when $k = 1$, the above algorithm is equivalent to Chow and Liu's method [11] for constructing optimal 1-degree Bayesian networks.

4.2 A First-Cut Solution

Observe that in Algorithm 2, there is only one place where we interact directly with the input dataset D, namely, the greedy selection of an AP pair (X_i, Π_i) in each iteration of the algorithm (Line 6). Therefore, if we are to make Algorithm 2 differentially private, we only need to replace Line 6 of Algorithm 2 with a procedure that selects (X_i, Π_i) from Ω in a private manner. Such a procedure can be implemented with the exponential mechanism outlined in Section 2.1, using the mutual information function I as the score function. Specifically, we first inspect each AP pair $(X, \Pi) \in \Omega$, and calculate the mutual information $I(X, \Pi)$ between X and Π; After that, we sample an AP pair from Ω, such that the sampling probability of any pair (X, Π) is proportional to $\exp(I(X, \Pi)/2\Delta)$, where Δ is a scaling factor.

The value of Δ is set as follows. As mentioned in Section 3, PrivBayes requires that the construction of the Bayesian network $\mathcal{N}$ should satisfy $(\varepsilon/2)$-differential privacy. Accordingly, we set $\Delta = 2(d-1)S(I)/\varepsilon$, where $S(I)$ denotes the sensitivity of the mutual information function I (see Equation 3). This ensures that each invocation of the exponential mechanism satisfies $(\varepsilon/2(d-1))$-differential privacy. Given the composability property of differential privacy [16] and the fact that we only invoke the exponential mechanism $d-1$ times during the construction of $\mathcal{N}$, it can be verified that the overall process of constructing $\mathcal{N}$ is $(\varepsilon/2)$-differentially private.

Last, we calculate the sensitivity, $S(I)$.

Lemma 1.

$$S(I(X,\Pi)) = \begin{cases} \frac{1}{n}\log n + \frac{n-1}{n}\log\frac{n}{n-1}, & \textit{if } X \textit{ or } \Pi \textit{ is binary;} \\ \frac{2}{n}\log\frac{n+1}{2} + \frac{n-1}{n}\log\frac{n+1}{n-1}, & \textit{otherwise,} \end{cases}$$

where n is the number of tuples in D.

This can be shown by considering the maximum change in mutual information based on its definition (5), as the various probabilities are changed by the alteration of one tuple. We omit the full proof for brevity, but the maximum difference in mutual information between binary variables is achieved by this example:

$X\backslash\Pi$	0	1
0	0	0
1	0	1

$X\backslash\Pi$	0	1
0	$\frac{1}{n}$	0
1	0	$\frac{n-1}{n}$

The mutual information of the left distribution is 0, and that of the right one is $\frac{1}{n}\log n + \frac{n-1}{n}\log\frac{n}{n-1}$.

4.3 An Improved Solution

The method in Section 4.2 is simple and intuitive, but may not achieve the best results: Observe that $S(I) > \log n/n$; this can be large compared to the range of I. For example, $\text{range}(I) = 1$ for binary distributions. As a consequence, the scaling factor $\Delta = 2(d-1)S(I)/\varepsilon$ tends to be large, and so the exponential mechanism is still quite likely to sample (from Ω) an AP pair with a small mutual information. In that case, the Bayesian network $\mathcal{N}$ constructed using the exponential mechanism will offer a weak approximation of $\Pr[\mathcal{A}]$, resulting in a low-quality output from PrivBayes. To improve over this solution, we propose to avoid using I as the score function in the exponential mechanism. Instead, we define a novel function F that maps each AP pair $(X, \Pi) \in \Omega$ to a score, such that

1. F's sensitivity is small (with respect to the range of F).
2. If $F(X, \Pi)$ is large, then $I(X, \Pi)$ tends to be large.

The rationale is that since $S(F)$ is small with respect to $\text{range}(F)$, the scaling factor $\Delta = 2(d-1)S(F)/\varepsilon$ will also be small, and hence, the exponential mechanism has a high probability to select an AP pair (X, Π) with a large $F(X, \Pi)$. In turn, such an AP pair tends to have a large mutual information between X and Π, which helps improve the quality of the Bayesian network $\mathcal{N}$.

In what follows, we will clarify our construction of F. To achieve property 2 above, we set F to its maximum value (i.e., 0) when I is greatest. To achieve property 1, we make $F(X, \Pi)$ decrease linearly in proportion to the L_1 distance from $\Pr[X, \Pi]$ to a distribution that maximizes F, since linear functions ensure that the sensitivity is controlled: the function does not change sharply anywhere in its domain. We first introduce the concept of *maximum joint distribution*, which will be used to define the peaks of F, and then characterize such distributions:

Definition 3 (Maximum Joint Distribution). *Given an AP pair (X, Π), a maximum joint distribution $\Pr^{\diamond}[X, \Pi]$ for X and Π is one that maximizes the mutual information between X and Π.*

Lemma 2. *Assume that $|\text{dom}(X)| \leq |\text{dom}(\Pi)|$. A distribution $\Pr^{\diamond}[X, \Pi]$ is a maximum joint distribution if and only if*

1. $\Pr^{\diamond}[X = x] = 1/|\text{dom}(X)|$, *for any $x \in \text{dom}(X)$;*

2. *For any $\pi \in \text{dom}(\Pi)$, there is at most one $x \in \text{dom}(X)$ with $\Pr^{\diamond}[X = x, \Pi = \pi] > 0$.*

Proofs in this section are deferred to the appendix. We illustrate Definition 3 and Lemma 2 with an example:

Example 2. *Consider a binary variable X with $\text{dom}(X) = \{0, 1\}$ and a variable Π with $\text{dom}(\Pi) = \{a, b, c\}$. Consider two joint distributions between X and Π as follows:*

$X\backslash\Pi$	a	b	c
0	.5	0	0
1	0	.5	0

$X\backslash\Pi$	a	b	c
0	0	.2	.3
1	.5	0	0

By Lemma 2, both of the above distributions are maximum joint distributions, with $I(X,\Pi) = 1$.

Let (X,Π) be an AP pair, and $\mathcal{P}^\diamond[X,\Pi]$ be the set of all maximum joint distributions for X and Π. Our score function F (for evaluating the quality of (X,Π)) is defined as

$$F(X,\Pi) = -\frac{1}{2}\min_{\Pr^\diamond\in\mathcal{P}^\diamond}\left\|\Pr[X,\Pi]-\Pr^\diamond[X,\Pi]\right\|_1. \qquad (7)$$

If $F(X,\Pi)$ is large, then $\Pr[X,\Pi]$ must have a small L_1 distance to one of the maximum joint distributions in $\mathcal{P}^\diamond[X,\Pi]$, and vice-versa. In turn, if $\Pr[X,\Pi]$ is close to a maximum joint distribution in $\mathcal{P}^\diamond[X,\Pi]$, then intuitively, $\Pr[X,\Pi]$ is likely to give a large mutual information between X and Π. In other words, the value of $F(X,\Pi)$ tends to be positively correlated with $I(X,\Pi)$. This explains why F could be a good score function to replace I. In addition, F has a much smaller sensitivity than I, as shown in the following theorem:

THEOREM 2. *$S(F) = 1/n$.*

This follows immediately from considering the L_1 distance between neighboring distributions. Observe that $S(F) < S(I)/\log n$, where n is the number of tuples in the input data. Meanwhile, the ranges of F and I are comparable; for example, range$(I) = 1$ and range$(F) = 0.5$ for binary domains. Therefore, when n is large (as is often the case), the sensitivity-to-range ratio of F is significantly smaller than that of I, which makes F a favorable score function over I for selecting AP pairs in the Bayesian network $\mathcal{N}$.

4.4 Computation of F

While (7) defines the function F, it still remains unclear how we can calculate $F(X,\Pi)$ given $\Pr[X,\Pi]$. In this subsection, we use dynamic programming to solve the problem for the case when all attributes in $\Pi\cup\{X\}$ have binary domains; we address the case of non-binary domains in Section 4.5.

Let (X,Π) be an AP pair where $|\Pi| = k$. Then, the joint distribution $\Pr[X,\Pi]$ can be represented by a 2×2^k matrix where the sum of all elements is 1. For example, Table 3(a) illustrates a joint distribution $\Pr[X,\Pi]$ with $|\Pi| = 2$. To compute $F(X,\Pi)$, we need to identify the minimum L_1 distance between $\Pr[X,\Pi]$ and a maximum joint distribution $\Pr^\diamond[X,\Pi]\in\mathcal{P}^\diamond[X,\Pi]$. Table 3(b) illustrates one such maximum joint distribution, whose L_1 distance to the distribution in Table 3(a) equals 0.4. To derive the minimum L_1 distance, a naive approach is to enumerate all maximum joint distributions in $\mathcal{P}^\diamond[X,\Pi]$; nevertheless, as $\mathcal{P}^\diamond[X,\Pi]$ may contain an infinite number of maximum joint distributions, a brute-force enumeration of $\mathcal{P}^\diamond$ is infeasible. To address this issue, we will first introduce an exponential-time algorithm for computing $F(X,\Pi)$, which will serve as the basis of our dynamic programming solution.

The basic idea of our exponential-time algorithm is to (i) partition the distributions in $\mathcal{P}^\diamond$ into a finite number of equivalence classes, and then (ii) compute $F(X,\Pi)$ by processing each equivalence class individually. By Lemma 2, any maximum joint distribution $\Pr^\diamond[X,\Pi]$ has the following property: for any $\pi\in\text{dom}(\Pi)$, either $\Pr^\diamond[X=0,\Pi=\pi]=0$ or $\Pr^\diamond[X=1,\Pi=\pi]=0$. In other words, for each column in the matrix representation of $\Pr^\diamond[X,\Pi]$ (where a column corresponds to a value in dom(Π)), there should be at most one non-zero entry. For example, the gray cells in Table 3(b) indicate the positions of non-zeros in the given maximum joint distribution.

For any two distributions in $\mathcal{P}^\diamond$, we say that they are *equivalent* if (i) their matrix representations have the same number of non-zero entries, and (ii) the positions of the non-zero entries are the same in the two matrices. Suppose that we divide the distributions in $\mathcal{P}^\diamond$ into equivalence classes, each of which contains a maximal subset of equivalent distributions. Then, totally there are $O(3^{2^k})$ equivalent classes. It turns out that we can easily calculate the minimum L_1 distance from $\Pr[X,\Pi]$ to each equivalence class.

Table 3: An example of joint distributions

$X\backslash\Pi$	00	01	10	11
0	.6	0	0	0
1	.1	.1	.1	.1

(a) input joint distribution

$X\backslash\Pi$	00	01	10	11
0	.5	0	0	0
1	0	.3	.1	.1

(b) maximum joint distribution

To explain, consider a particular equivalence class E. Let Z^- be the set of pairs (x,π), such that $\Pr^\diamond[X=x,\Pi=\pi]=0$ for any $\Pr^\diamond[X,\Pi]\in E$. That is, Z^- captures the positions of all zero entries in the matrix representation of $\Pr^\diamond[X=x,\Pi=\pi]$. Similarly, we define the sets of non-zero entries in row $X=0$ and $X=1$ as

$$\begin{aligned} Z_0^+ &= \{(0,\pi)\mid \Pr^\diamond[X=0,\Pi=\pi]>0\}\,\text{, and} \\ Z_1^+ &= \{(1,\pi)\mid \Pr^\diamond[X=1,\Pi=\pi]>0\}\,. \end{aligned}$$

For convenience, we also abuse notation and define

$$\begin{aligned} \Pr[Z^-] &= \textstyle\sum_{(x,\pi)\in Z^-}\Pr[X=x,\Pi=\pi], \\ \Pr[Z_0^+] &= \textstyle\sum_{(x,\pi)\in Z_0^+}\Pr[X=x,\Pi=\pi], \\ \Pr[Z_1^+] &= \textstyle\sum_{(x,\pi)\in Z_1^+}\Pr[X=x,\Pi=\pi]. \end{aligned}$$

By Lemma 2, we have $\Pr^\diamond[Z^-]=0$, $\Pr^\diamond[Z_0^+]=1/2$, and $\Pr^\diamond[Z_1^+]=1/2$ for any $\Pr^\diamond[X,\Pi]\in E$. Then, for any $\Pr[X,\Pi]$, its L_1 distance to a distribution $\Pr^\diamond[X,\Pi]\in E$ is bounded by:

$$\left\|\Pr[X,\Pi]-\Pr^\diamond[X,\Pi]\right\|_1 \geq \Pr[Z^-]+\left|\Pr[Z_0^+]-\frac{1}{2}\right|+\left|\Pr[Z_1^+]-\frac{1}{2}\right|.$$

Let $(x)_+$ denote $\max(0,x)$. Given that $\Pr[Z^-]+\Pr[Z_0^+]+\Pr[Z_1^+]=1$, the above inequality can be simplified to

$$\left\|\Pr[X,\Pi]-\Pr^\diamond[X,\Pi]\right\|_1 \geq 2\cdot\left[\left(\tfrac{1}{2}-\Pr[Z_0^+]\right)_+ + \left(\tfrac{1}{2}-\Pr[Z_1^+]\right)_+\right]. \qquad (8)$$

Furthermore, there always exists a $\Pr^\diamond[X,\Pi]\in E$ that makes the equality hold. In other words, once the positions of the non-zero entries in $\Pr^\diamond[X,\Pi]$ are fixed, we can use (8) to derive the minimum L_1 distance from any $\Pr[X,\Pi]$ to E, with a linear scan of the entries in the matrix representation of $\Pr[X,\Pi]$. By enumerating all $O(3^{2^k})$ equivalence classes of $\mathcal{P}^\diamond$, we can then derive $F(X,\Pi)$.

The above procedure for calculating F is impractical when $k\geq 4$, as the exhaustive search over all possible equivalence classes of $\mathcal{P}^\diamond$ is prohibitive. To tackle this problem, we propose a dynamic-programming-based optimization that reduces computation costs by taking advantage of the fact that the distributions are induced by n items.

Based on (8), our target is to find a combination of Z_0^+ and Z_1^+ (which therefore determine Z^-) that minimizes

$$\left(\frac{1}{2}-\Pr[Z_0^+]\right)_+ + \left(\frac{1}{2}-\Pr[Z_1^+]\right)_+.$$

We define the probability mass associated with Z_0^+ and Z_1^+ as K_0 and K_1 respectively. Initially, $K_0=K_1=0$. For each $\pi\in\text{dom}(\Pi)$, we can either increase K_0 by $\Pr[X=0,\Pi=\pi]$ (by assigning $(0,\pi)$ to Z_0^+) or increase K_1 by $\Pr[X=1,\Pi=\pi]$ (by assigning $(1,\pi)$ to Z_1^+). We index $\pi\in\text{dom}(\Pi)$ as $\pi_1,\pi_2,\ldots,\pi_{2^k}$. We use $C(i,a,b)$ to indicate if $K_0=a/n$ and $K_1=b/n$ is reachable by using the first i π's, i.e., $\pi_1,\pi_2,\ldots,\pi_i$. It can be verified that (i) $C(i,a,b)=$ true if $i=a=b=0$, (ii) $C(i,a,b)=$ false if $i<0$ or $a<0$ or $b<0$, and

(iii) otherwise,

$$\begin{aligned} C(i,a,b) \quad &= \quad C(i-1, a-n\Pr[X=0,\Pi=\pi_i], b) \\ &\quad \vee\ C(i-1, a, b-n\Pr[X=1,\Pi=\pi_i]). \end{aligned}$$

Given an input dataset D with n tuple, each cell in $\Pr[X,\Pi]$ must be a multiple of $1/n$. Thus, we only consider the case when a and b are integers in the range $[0,n]$. Thus, the total number of states $C(i,a,b)$ is $n^2 2^k$. A direct traversal of all states takes $O(n^2 2^k)$ time. To reduce this time complexity, we introduce the following concept:

Definition 4 (Dominated State). *A state $C(i,a_1,b_1)$ is* dominated *by $C(i,a_2,b_2)$ if and only if $a_1 \leq a_2$ and $b_1 \leq b_2$.*

Note that a dominated state is always inferior to some other states, and hence, it can be ignored without affecting the correctness of final result. Consequently, we maintain the set of at most n non-dominated reachable states for each $i \in [1, 2^k]$. The function F can be calculated by

$$F(X,\Pi) = -\min_{C(2^k,a,b)=\text{true}} \left(\frac{1}{2}-\frac{a}{n}\right)_+ + \left(\frac{1}{2}-\frac{b}{n}\right)_+ .$$

As such, the total number of states that need to be traversed is $n2^k$, and thus the complexity of the algorithm is reduced to $O(n2^k)$. Note that k is small in practice, since we only need to consider low-degree Bayesian networks.

4.5 Extension to General Domains

The dynamic programming approach in Section 4.4 assumes that all attributes in the input data D are binary. In this section, we extend our solution to the case when D contains non-binary attributes.

Following the common practice in the literature [38], our first solution converts each non-binary attribute in the dataset into a set of binary attributes. In particular, for each categorical attribute X whose domain size equals ℓ, we first encode each value in X's domain into a binary representation with $\lceil \log \ell \rceil$ bits; after that, we convert X into $\lceil \log(l) \rceil$ binary attributes $X_1, X_2, \ldots, X_{\lceil \log \ell \rceil}$, such that X_i corresponds to the i-th bit in the binary representation. Meanwhile, for each continuous attribute Y, we first discretize the domain of Y into a fixed number b of equi-width bins (we use $b = 16$), and then convert Y into $\lceil \log b \rceil$ binary attributes, using a similar approach to the transformation of X. After the transformation, D can be encoded to form a new database D_b in the binary domain. After that, we apply PrivBayes on D_b to generate a synthetic dataset D_b^*, then decode it to get D^* in the original domain.

A second solution tries to preserve the semantics of (discrete) attributes more directly, via a more complex search. Following the outline for the binary case, we can model the computation of $F(X,\Pi)$ over non-binary attribute X and parent set Π as an optimization problem. Now the structure of X is more complex, the approach of dynamic programming does not apply. Instead, we can find a different combinatorial characterization of maximum distributions, and encode this with a set of (linear) constraints. The search for the minimum cost maximum distribution is the an optimization problem over these constraints, which can be solved by an integer program. We postpone details of this approach to the full version of this paper; in our experiments, we show results using the binary encoding, which is effective enough for our purposes.

4.6 Choice of k and θ-usefulness.

We have discussed how to build a k-degree Bayesian network under differential privacy, where k is considered as a given input to the algorithm. However, k is usually unknown in real applications and should be chosen carefully. The choice of k is non-trivial for PrivBayes. Intuitively, a Bayesian network with a larger k keeps more information from the full dimensional distribution $\Pr[\mathcal{A}]$, e.g., a $(d-1)$-degree Bayesian network approximates $\Pr[\mathcal{A}]$ perfectly without having any information loss. On the other hand, the downside of using large k is that it forces PrivBayes to anonymize a set of high-dimensional marginal distributions in the second phase, which are very vulnerable to noise due to their domains of large size. These noisy distributions are less useful after anonymization especially when the privacy budget ε is small, leading to a synthetic database full of random perturbation. With very small values of ε, the best choice may be to pick $k = 0$, i.e. to model all attributes as independent. Hence, the choice of k should balance the informativeness of a Bayesian network and the robustness of marginal distributions. This balancing act is affected by three parameters: the total privacy budget ε, the total number of tuples in database n, and usefulness of each noisy marginal distribution in the second phase θ. We quantify this in the following definition.

Definition 5 (θ-usefulness). *A noisy distribution is θ-useful if the ratio of average scale of information to average scale of noise is no less than θ.*

Lemma 3. *The noisy distributions in Algorithm 1 are $\left(\dfrac{n\cdot\varepsilon}{(d-k)\cdot 2^{k+3}}\right)$-useful.*

Proof. In Algorithm 1, each marginal distribution is $(k+1)$-dimensional with a domain size 2^{k+1}. Therefore, the average scale of information in each cell is $1/2^{k+1}$.

For the scale of noise, we have $d-k$ marginal distributions to be anonymized and each of them consumes privacy budget $\varepsilon/2(d-k)$. The sensitivity of each marginal distribution is $2/n$. According to the Laplace mechanism, the Laplace noise N injected to each cell is drawn from distribution $\text{Lap}\left(4(d-k)/n\varepsilon\right)$ where the average scale of noise is $\text{E}(|\eta|) = 4(d-k)/n\varepsilon$. □

The notion of θ-usefulness provides a more intuitive way to choose k automatically without closely studying the specific instance of the input database. Generally speaking, we believe a 0.5-useful noisy distribution is not good because the scale of noise is twice as that of information, while a 5-useful one is more reliable due to its large information to noise ratio. In practice, we set up a threshold θ, then choose the largest positive integer k that guarantees θ-usefulness in parameter learning (note, this is independent of the data, as it depends only on the non-private values ε, θ, n and d). If such a k does not exist, k is set to the minimum value, 0. In the experimental section, we will show that there is a wide range of θ to choose to train a PrivBayes model for good performance.

5. EXPERIMENTS

5.1 Experimental Settings

Datasets. We make use of four real databsets in our experiments: (i) Adult [4], which includes the information of 45,222 individuals extracted from the 1994 US Census, (ii) NLTCS [2], which contains records of 21,574 individuals participated in the National Long Term Care Survey, (iii) TPC-E [3], 40,000 tuples obtained by joining four tables in the TPC-E benchmark: “Trade”, “Security”, “Security status” and “Trade type”, and (iv) BR2000 [1], which consists of 38,000 census records collected from Brazil in year 2000. Each of the four datasets contains both continuous and categorical attributes. Table 4 illustrates the properties of the datasets.

Tasks. We evaluate the performance of PrivBayes on two different tasks. The first task is to build all α-way marginals of a

Table 4: Dataset characteristics.

Dataset	Cardinality	Dimensionality	Domain size
Adult	45,222	15	$\approx 2^{52}$
NLTCS	21,574	16	$\approx 2^{16}$
TPC-E	40,000	24	$\approx 2^{77}$
BR2000	38,000	14	$\approx 2^{32}$

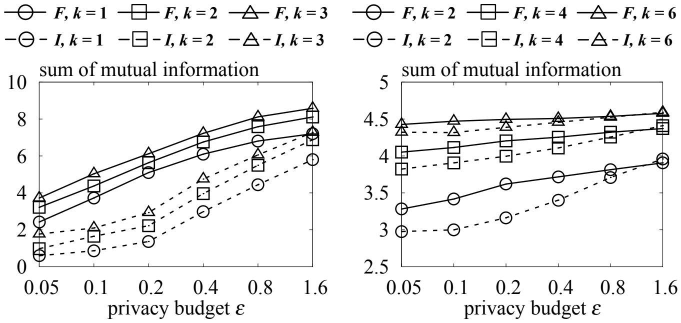
dataset [5]. For convenience, we use Q_α to denote the set of all α-way marginals. We evaluate Q_3 and Q_4 on NLTCS, but examine Q_2 and Q_3 instead on the remaining three datasets, since each of those datasets leads to a prohibitively large number of queries in Q_4. We measure the accuracy of each noisy marginal by the *total variation distance* [14] between itself and the noise-free marginal (i.e., half of the L_1 distance between the two marginals, when both of them are treated as probability distributions). We use the average accuracy over all marginals as the final error metric for Q_α.

The second task that we consider is to *simultaneously* train multiple SVM classifiers on a dataset, where each classifier predicts one attribute in the data based on all other attributes. Specifically, on Adult, we train four classifiers to predict whether an individual (i) is a female, (ii) holds a post-secondary degree, (iii) makes over $50K$ a year, and (iv) has never married, respectively. Meanwhile, on NLTCS, we construct four classifiers to predict whether a person (i) is unable to get outside, (ii) is unable to manage money, (iii) is unable to bathe, and (iv) is unable to travel, respectively. We omit the experiments on BR2000 and TCP-E for space reasons. For each classification task, we use 80% of the tuples in the data as the training set, and the other 20% as the testing set. We apply PRIVBAYES on the training data to generate a synthetic dataset, and then use the synthetic data to construct SVM classifiers. The quality of each classifier is measured by its *misclassification rate* on the testing set, i.e., the fraction of tuples in the testing data that are incorrectly classified.

For each of the aforementioned tasks, we repeat each experiment on each method 50 times, and we report the average measurements in our experimental results.

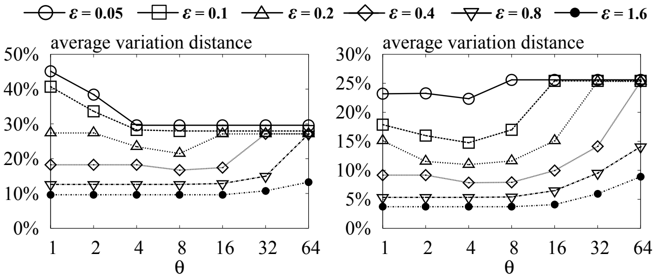
Baselines. For the task of answering count queries in Q_α, we compare PRIVBAYES with three approaches: (i) *Laplace* [17], which generates all α-way marginals of a dataset and then injects Laplace noise directly into each cell of the marginals, (ii) *Fourier* [5], which transforms the input data D into the Fourier domain, adds Laplace noise to a subset of Fourier coefficients and uses the noisy coefficients to construct α-way marginals, and (iii) *Contingency*, which first builds the noisy contingency table, and then projects it onto attribute subsets to compute marginals. However, *Contingency* is only applicable to NLTCS since its computational cost is proportional to the domain size of input data. We also considered several other existing approaches [15, 20, 24, 25, 39] for answering count queries under differential privacy, but find them inapplicable due to our datasets' large domain size. For fair comparison, we adopt two consistency techniques to boost the accuracies of baselines: (i) nonnegativity, which rounds all negative counts in a noisy marginal to 0, and (ii) normalization, which linearly rescales the counts in a noisy marginal to make them sum to n.

For the task of training multiple SVM classifiers, we compare PRIVBAYES against four methods: *PrivateERM* [9], *PrivGene* [40], *NoPrivacy*, and *Majority*. In particular, *PrivateERM* and *PrivGene* are two state-of-the-art methods for SVM classification under differential privacy. *NoPrivacy* constructs classifiers directly on the input data without any privacy protection. *Majority* is a naive classification method under differential privacy that works as follows. Let $Y = \{0,1\}$ be the attribute to be classified,



(a) Adult (b) NLTCS

Figure 2: Comparison between F and I



(a) Adult, Q_2 (b) NLTCS, Q_3

Figure 3: Choice of θ

and n be the number of tuples in the training data. *Majority* first counts the number of tuples in the training data with $Y = 1$, and then adds Laplace noise (with scale $1/\varepsilon$) into the count to ensure ε-differential privacy. If the noisy count is larger than $n/2$, then *Majority* predicts that all tuples in the testing data should have $Y = 1$; otherwise, *Majority* predicts $Y = 0$ for all tuples. For PRIVBAYES, *PrivGene*, and *NoPrivacy*, we adopt the standard hinge-loss C-SVM model [7] with $C = 1$; for *PrivateERM*, we adopt a slightly different SVM model with Huber loss [9], as it does not support the hinge-loss model.

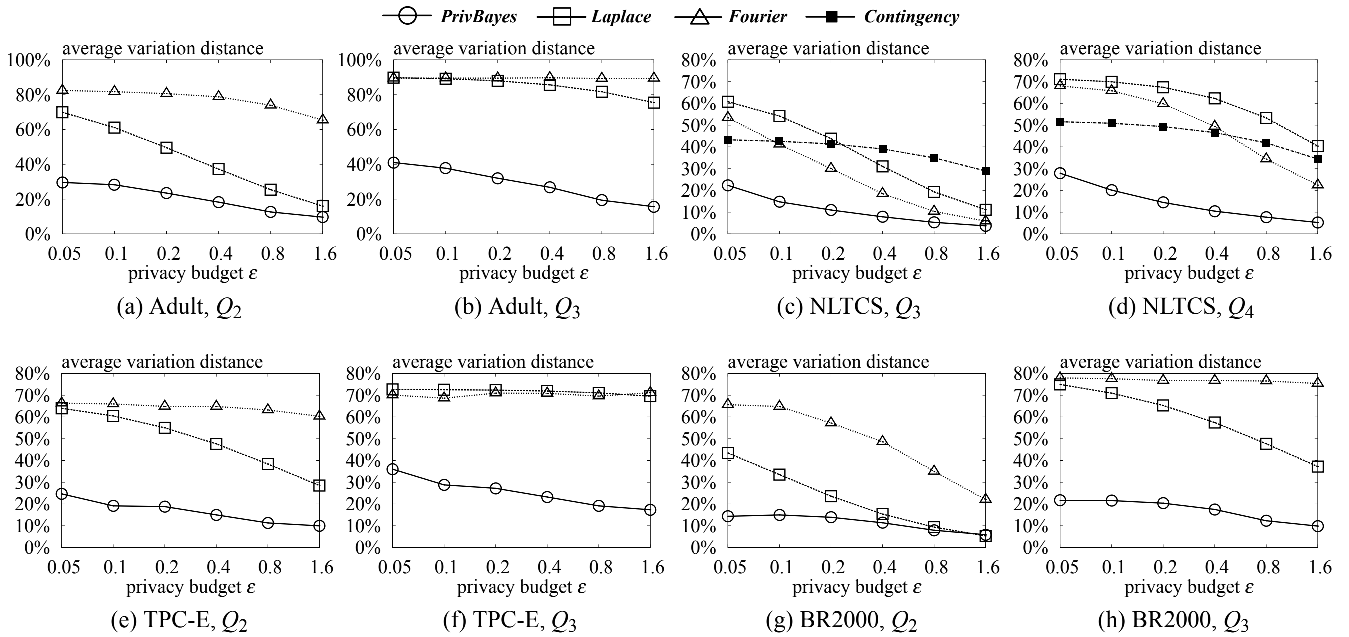
5.2 Effect of Quality Function F

In the first set of experiments, we evaluate the effectiveness of score function F (in Section 4.2) against the mutual information function I. Figure 2 illustrates the performance of PRIVBAYES when combined with F and I, respectively, using Adult and NLTCS. The performance of each combination evaluated by the sum of the mutual information of every AP pair in the Bayesian network $\mathcal{N}$, i.e., $\sum_{i=1}^{d} I(X_i, \Pi_i)$. Observe that F significantly outperforms I in almost all cases. This is consistent with our analysis in Section 5.2 that F helps improve the quality of the Bayesian network constructed by PRIVBAYES.

For small k values (i.e., $k = 0, 1, 2$), the time taken to construct a Bayesian network with F is less than 1 minute and is negligible. For larger values of k, the time taken is typically higher (a few hours in the case of $k = 5$). Note that this is not a major concern, since data release is not considered a real-time problem, and the computation of F for different combinations of attributes can be easily parallelized.

5.3 Choice of θ

Recall that PRIVBAYES has only one internal parameter: the degree of Bayesian network k. As discussed in Section 4.6, we adopt the θ-usefulness criterion to automatically select an appropriate



(a) Adult, Q_2 (b) Adult, Q_3 (c) NLTCS, Q_3 (d) NLTCS, Q_4

(e) TPC-E, Q_2 (f) TPC-E, Q_3 (g) BR2000, Q_2 (h) BR2000, Q_3

Figure 4: α-way marginals on four datasets

value for k, based on the number of tuples in the input data D as well as the domains of the attributes in D. To evaluate the effect of θ on PrivBayes, we use the Q_2 query set on Adult and the Q_3 query set on NLTCS, and examine the performance of PrivBayes in answering the queries, with varying θ and ε. Figures 3 illustrates the results. Observe that the average variation distance of the tested query set tends to be higher when θ is very small or very large. This is consistent with our analysis in Section 4.6 that (i) small θ leads to very noisy marginal distributions in the second phase of PrivBayes, which makes the synthetic dataset drastically different from the input data, and (ii) large θ makes it difficult for PrivBayes to construct a high quality Bayesian network in its first phase, which also leads to inferior synthetic data. Based on Figure 3, we infer that an appropriate value for θ should be in the range of $[2,16]$. For all subsequent experiments, we set $\theta = 4$.

Note that our tuning of θ is only based on the Q_2 query set on Adult and the Q_3 query set on NLTCS, without inspecting other datasets or other tasks on Adult and NLTCS. Therefore, our choice of θ does not reveal any private information on TPC-E and BR2000, and it does not unfairly favor PrivBayes on the tasks of SVM classification on Adult and NLTCS.

5.4 α-way Marginals

This section compares PrivBayes with the *Laplace* and *Fourier* approaches on eight sets of marginals over four datasets, and *Contingency* on two sets over NLTCS. Figure 4 shows the average variation distance of each method for each query set Q_α, varying the privacy budget ε. PrivBayes clearly outperforms the other three methods in all cases. The relative superiority of PrivBayes is more pronounced when (i) ε decreases or (ii) the value of α increases. To explain, observe that when ε is small, PrivBayes chooses to construct a very low-degree Bayesian network (down to $k = 0$), due to the θ-usefulness criterion. As a consequence, the marginal distributions in the second phase of PrivBayes will be more robust against noise injection, which ensures the quality of the synthetic data will not degrade too significantly. In contrast, the performance of *Laplace* and *Fourier* is highly sensitive to ε, owing to which they incur considerable errors when ε decreases. *Contingency* also generates inaccurate marginals with low privacy budget, with performance improving slowly as ε increases.

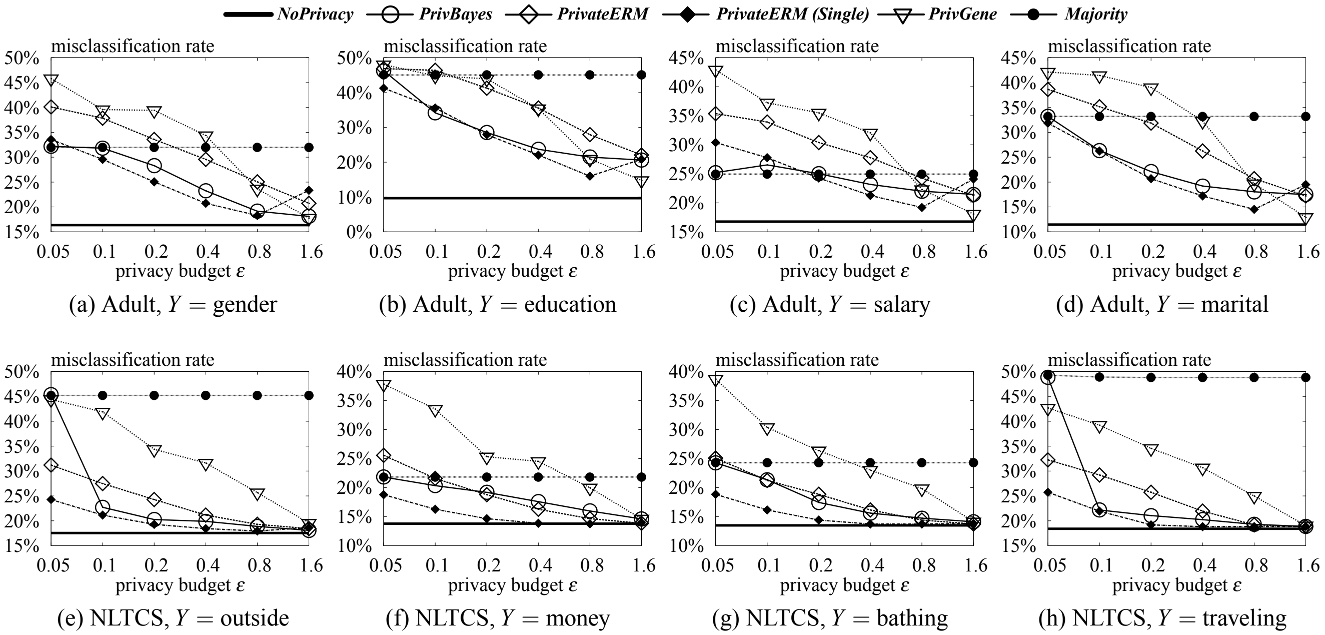
Meanwhile, when α increases, the query set Q_α corresponds to a larger set of marginals, in which case the queries Q_α have a higher sensitivity. Therefore, *Laplace* needs to inject a larger amount of noise into Q_α for privacy protection, leading to higher query errors. *Fourier* also suffers from a similar issue. On the other hand, the error of PrivBayes is not sensitive to α, as the Bayesian network constructed (once) by PrivBayes enables it to nicely capture the correlations among attributes. Consequently, it can closely approximate the marginals pertinent to Q_α even when α increases.

5.5 Multiple SVM Classifiers

In the last set of experiments, we evaluate different methods for SVM classification. As explained in Section 5.1, on each of Adult and NLTCS, we train four SVM classifiers simultaneously. For PrivBayes, we apply it to generate only *one* synthetic dataset D^* from each training set, and then use D^* to train all four classifiers required. The other differentially private methods (i.e., *PrivateERM*, *PrivGene*, and *Majority*) can only produce one classifier at a time. Therefore, for each of those method, we evenly divide the privacy budget ε evenly into four parts, and use $\varepsilon/4$ budget to train each classifier. To illustrate the performance of *PrivateERM* when building a single classifier, we include an additional baseline referred to as *"PrivateERM (Single)"*. This baseline is identical to *PrivateERM*, expect that it uses a privacy budget of ε (instead of $\varepsilon/4$) in training each classifier.

Figure 5 shows the misclassification rate of each method as a function of the overall ε. The error of NoPrivacy remains unchanged for all ε, since it does not enforce ε-differential privacy—it represents the best case to aim for. The accuracy of Majority is insensitive to ε, since (i) it performs classification only by checking whether there exists more than 50% tuples in the training set with a certain label, and (ii) this check is quite robust against noise injection when the number of tuples in the training set is large (as is the case in our experiments).

As for the other methods, PrivBayes consistently outperforms *PrivateERM* and *PrivGene* on Adult, except for the case of $\varepsilon = 1.6$.



(a) Adult, Y = gender (b) Adult, Y = education (c) Adult, Y = salary (d) Adult, Y = marital

(e) NLTCS, Y = outside (f) NLTCS, Y = money (g) NLTCS, Y = bathing (h) NLTCS, Y = traveling

Figure 5: Multiple SVM classifiers on two datasets

Meanwhile, on NLTCS, PrivBayes and *PrivateERM* are comparable, and they both outperform *PrivGene*. Interestingly, in Figure 5(c), the misclassification rate of PrivBayes increases when ε changes from 0.05 to 0.1. The reason is that we have tuned the parameter θ for PrivBayes based on count queries on Adult and NLTCS, and hence, our choice of θ (and thus, the choise of k) does not always guarantee the best performance for PrivBayes on classification tasks. Overall, PrivBayes is superior to both *PrivateERM* and *PrivGene* on the this classification task.

On the other hand, PrivBayes is outperformed by *PrivateERM (Single)*[3] in most cases. This is reasonable given that *PrivateERM* is designed solely for SVM classification, whereas PrivBayes does not specifically optimize for SVM classification when it generates the synthetic data. In general, the fact that PrivBayes can support multiple analytical tasks (without incurring extra privacy overhead) makes it highly favorable in the common case when the user does not have a specific task in mind and would like to conduct *exploratory* data analysis by experimenting with various tasks.

6. CONCLUDING REMARKS

The model of Bayesian networks has proven a powerful way to represent correlated data approximately. We have seen that it is also highly effective as a model to release data while respecting privacy. We see that data released this way is is very accurate, and indeed offers better accuracy than customized mechanisms for particular objectives, such as classification. A crucial part of our approach is the crafting of a novel quality function F as a surrogate for mutual information, which dramatically improves the quality of the released data. This requires some effort in order to compute efficiently, although since this is part of the release process, we can afford to spend more time on this. Nevertheless, an open problem is to study functions which can substitute for mutual information and which are fast to compute.

The natural next step is to extend this work to databases over multiple tables. The approach of building a graphical model and releasing this privately applies here also. However, care is needed: in the examples we consider, each individual affects a single row of the initial database table. As we consider more complex schemas, the impact of an individual (and hence the scale of noise needed for privacy) may grow very large, and a more careful analysis is needed to ensure that noise does not outweigh the signal.

Acknowledgments. This work was supported by the Nanyang Technological University under SUG Grant M4080094.020, by the Microsoft Research Asia under a gift grant, and by the Ministry of Education (Singapore) under an AcRF Tier-2 grant.

7. REFERENCES

[1] Integrated public use microdata series international. https://international.ipums.org.

[2] Statlib. http://lib.stat.cmu.edu/.

[3] Transaction processing performance council. http://www.tpc.org.

[4] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[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 *PODS*, pages 273–282, 2007.

[6] R. Bhaskar, S. Laxman, A. Smith, and A. Thakurta. Discovering frequent patterns in sensitive data. In *KDD*, pages 503–512, 2010.

[7] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. page 27, 2011.

[8] K. Chaudhuri and C. Monteleoni. Privacy-preserving logistic regression. In *NIPS*, pages 289–296, 2008.

[9] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12:1069–1109, 2011.

[10] D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of bayesian networks is NP-Hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

[11] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

[12] G. Cormode, C. M. Procopiuc, E. Shen, D. Srivastava, and T. Yu. Differentially private spatial decompositions. In *ICDE*, 2012.

[3] The behavior of *PrivateERM (Single)* on Adult with $\varepsilon = 1.6$ is an artifact of the algorithm itself: it computes an internal parameter ε'_p as a function of ε, which yields a sub-optimal choice when $\varepsilon = 1.6$.

[13] G. Cormode, C. M. Procopiuc, D. Srivastava, and T. T. L. Tran. Differentially private publication of sparse data. In *ICDT*, 2012.

[14] A. B. Cybakov. *Introduction to nonparametric estimation*. Springer, 2009.

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

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

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

[18] D. Feldman, A. Fiat, H. Kaplan, and K. Nissim. Private coresets. In *STOC*, pages 361–370, 2009.

[19] A. Friedman and A. Schuster. Data mining with differential privacy. In *KDD*, pages 493–502, 2010.

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

[21] M. Hay, V. Rastogi, G. Miklau, and D. Suciu. Boosting the accuracy of differentially private histograms through consistency. *PVLDB*, 3(1):1021–1032, 2010.

[22] D. Kifer, A. D. Smith, and A. Thakurta. Private convex optimization for empirical risk minimization with applications to high-dimensional regression. *Journal of Machine Learning Research - Proceedings Track*, 23:25.1–25.40, 2012.

[23] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

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

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

[26] C. Li and G. Miklau. Optimal error of query sets under the differentially-private matrix mechanism. In *ICDT*, pages 272–283, 2013.

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

[28] D. Margaritis. *Learning Bayesian Network Model Structure from Data*. PhD thesis, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, May 2003.

[29] F. McSherry and R. Mahajan. Differentially-private network trace analysis. In *SIGCOMM*, pages 123–134, 2010.

[30] F. McSherry and I. Mironov. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *KDD*, pages 627–636, 2009.

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

[32] P. Mohan, A. Thakurta, E. Shi, D. Song, and D. Culler. GUPT: privacy preserving data analysis made easy. In *SIGMOD*, pages 349–360, 2012.

[33] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84, 2007.

[34] V. Rastogi and S. Nath. Differentially private aggregation of distributed time-series with transformation and encryption. In *SIGMOD*, pages 735–746, 2010.

[35] B. I. P. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *Journal of Privacy and Confidentiality*, 4(1):65–100, 2012.

[36] A. Smith. Privacy-preserving statistical estimation with optimal convergence rate. In *STOC*, 2011.

[37] X. Xiao, G. Wang, and J. Gehrke. Differential privacy via wavelet transforms. In *ICDE*, pages 225–236, 2010.

[38] G. Yaroslavtsev, G. Cormode, C. M. Procopiuc, and D. Srivastava. Accurate and efficient private release of datacubes and contingency tables. In *ICDE*, pages 745–756, 2013.

[39] 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, 2012.

[40] J. Zhang, X. Xiao, Y. Yang, Z. Zhang, and M. Winslett. PrivGene: differentially private model fitting using genetic algorithms. In *SIGMOD*, pages 665–676, 2013.

APPENDIX

PROOF OF **Lemma 2**. The maximum mutual information between variables X and Π is

$$\begin{aligned}\max I(X,\Pi) &= \min\{\max H(X), \max H(\Pi)\}\\ &= \min\{\log|\text{dom}(X)|, \log|\text{dom}(\Pi)|\} = \log|\text{dom}(X)|,\end{aligned}$$

given that $|\text{dom}(X)| \le |\text{dom}(\Pi)|$. Therefore, the maximum joint distribution for X and Π should be a joint distribution for X and Π with mutual information $\log|\text{dom}(X)|$.

Suppose that $\Pr^\diamond(X,\Pi)$ is a joint distribution satisfying the two properties in Lemma 2. Given basic results in information theory, the two properties are equivalent to

1. $H(X) = \log|\text{dom}(X)|$;
2. $H(X \mid \Pi) = 0$.

Thus, the mutual information of $\Pr^\diamond(X,\Pi)$ is

$$I(X,\Pi) = H(X) - H(X \mid \Pi) = \log|\text{dom}(X)|.$$

By definition, $\Pr^\diamond(X,\Pi)$ is a maximum joint distribution.

On the other hand, suppose that $\Pr^\diamond(X,\Pi)$ is a maximum joint distribution with mutual information $\log|\text{dom}(X)|$. The mutual information can be expressed as:

$$I(X,\Pi) = \log|\text{dom}(X)| = H(X) - H(X \mid \Pi),$$

where $H(X) \le \log|\text{dom}(X)|$ and $H(X \mid \Pi) \ge 0$ always hold. Thus, we conclude that I is maximized in the (achievable) case that

1. $H(X) = \log|\text{dom}(X)|$, which is achieved only by the uniform distribution over $\text{dom}(X)$;
2. $H(X \mid \Pi) = 0$, which implies that there is an x for each π such that $\Pr[X = x \mid \Pi = \pi] = 1$.

The above two conditions are equivalent to the properties in the statement of Lemma 2. □

PROOF OF **Theorem 2**. Let $F_D(X,\Pi)$ be the F function for variable X and Π given input database D, i.e,

$$F_D(X,\Pi) = -\frac{1}{2}\min_{\Pr^\diamond \in \mathcal{P}^\diamond} \|\Pr_D[X,\Pi] - \Pr^\diamond[X,\Pi]\|_1.$$

Notice that $\mathcal{P}^\diamond[X,\Pi]$ is independent of the input database D. Now consider a pair of neighboring databases D_1 and D_2. We have

$$\|\Pr_{D_1}[X,\Pi] - \Pr_{D_2}[X,\Pi]\|_1 = 2/n. \tag{9}$$

Assume that $\Pr \in \mathcal{P}^\diamond[X,\Pi]$ is the closest maximum joint distribution to $\Pr_{D_1}[X,\Pi]$. We have

$$F_{D_1}(X,\Pi) = -1/2 \cdot \|\Pr_{D_1}[X,\Pi] - \Pr\|_1.$$

Combined with Equation 9, the L_1 distance between $\Pr_{D_2}[X,\Pi]$ and $\Pr$ can be upper bounded using the triangle inequality:

$$\begin{aligned}&\|\Pr_{D_2}[X,\Pi] - \Pr\|_1\\ &\quad\le \|\Pr_{D_1}[X,\Pi] - \Pr\|_1 + \|\Pr_{D_1}[X,\Pi] - \Pr_{D_2}[X,\Pi]\|_1\\ &\quad= -2 \cdot F_{D_1}(X,\Pi) + 2/n.\end{aligned}$$

On the other hand, recall that $\Pr$ is a maximum joint distribution in $\mathcal{P}^\diamond[X,\Pi]$. Therefore,

$$\begin{aligned}F_{D_2}(X,\Pi) &= -\frac{1}{2}\min_{\Pr^\diamond \in \mathcal{P}^\diamond} \|\Pr_{D_2}[X,\Pi] - \Pr^\diamond[X,\Pi]\|_1\\ &\ge -\frac{1}{2}\,\|\Pr_{D_2}[X,\Pi] - \Pr\|_1\\ &\ge -\frac{1}{2}\left(-2 \cdot F_{D_1}(X,\Pi) + \frac{2}{n}\right) = F_{D_1}(X,\Pi) - \frac{1}{n}.\end{aligned}$$

Thus, $F_{D_1}(X,\Pi) - F_{D_2}(X,\Pi) \le 1/n$. □