DPCube: Differentially Private Histogram Release through Multidimensional Partitioning

Article *in* Transactions on Data Privacy · February 2012

Source: arXiv

Yonghui Xiao
Emory University
13 PUBLICATIONS **153** CITATIONS
SEE PROFILE

Li Xiong
Emory University
144 PUBLICATIONS **3,581** CITATIONS
SEE PROFILE

Slawek Goryczka
Google Inc.
10 PUBLICATIONS **128** CITATIONS
SEE PROFILE

DPCube: Differentially Private Histogram Release through Multidimensional Partitioning

Yonghui Xiao, Li Xiong, Liyue Fan and Slawomir Goryczka

Abstract—Differential privacy is a strong notion for protecting individual privacy in privacy preserving data analysis or publishing. In this paper, we study the problem of differentially private histogram release for random workloads. We study two multidimensional partitioning strategies including: 1) a baseline cell-based partitioning strategy for releasing an equi-width cell histogram, and 2) an innovative 2-phase kd-tree based partitioning strategy for releasing a v-optimal histogram. We formally analyze the utility of the released histograms and quantify the errors for answering linear queries such as counting queries. We formally characterize the property of the input data that will guarantee the optimality of the algorithm. Finally, we implement and experimentally evaluate several applications using the released histograms, including counting queries, classification, and blocking for record linkage and show the benefit of our approach.

Index Terms—Differential privacy, non-interactive data release, histogram, classification, record linkage.

1 Introduction

As information technology enables the collection, storage, and usage of massive amounts of information about individuals and organizations, privacy becomes an increasingly important issue. Governments and organizations recognize the critical value in sharing such information while preserving the privacy of individuals. Privacy preserving data analysis and data publishing [3], [4], [5] has received considerable attention in recent years. There are two models for privacy protection [3]: the interactive model and the non-interactive model. In the interactive model, a trusted *curator* (e.g. hospital) collects data from *record owners* (e.g. patients) and provides an access mechanism for *data users* (e.g. public health researchers) for querying or analysis purposes. The result returned from the access mechanism is perturbed by the mechanism to protect privacy. In the non-interactive model, the curator publishes a "sanitized" version of the data, simultaneously providing utility for data users and privacy protection for the individuals represented in the data.

Differential privacy [6], [7], [3], [5], [8] is widely accepted as one of the strongest known privacy guarantees with the advantage that it makes few assumptions on the attacker's background knowledge. It requires the outcome of computations to be formally indistinguishable when run with or without any particular record in the dataset, as if it makes little difference whether an individual is being opted in or out of the database. Many meaningful results have been obtained for the interactive model with differential privacy [6], [7], [3], [5]. Non-interactive data release with differential privacy has been recently studied with hardness results obtained and it remains an open problem to find efficient algorithms for many domains [9], [10].

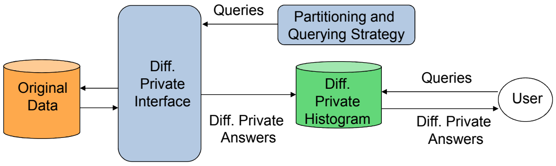


Fig. 1. Differentially private histogram release

In this paper, we study the problem of differentially private histogram release based on a differential privacy interface, as shown in Figure 6. A *histogram* is a disjoint partitioning of the database points with the number of points which fall into each partition. A differential privacy interface, such as the Privacy INtegrated Queries platform (PINQ) [11], provides a differentially private access to the raw database. An algorithm implementing the partitioning strategy submits a sequence of queries to the interface and generates a differentially private histogram of the raw database. The histogram can then serve as a sanitized synopsis of the raw database and, together with an optional synthesized dataset based on the histogram, can be used to support count queries and other types of OLAP queries and learning tasks.

An immediate question one might wonder is what

- *A preliminary version of the manuscript appeared in [1] and a demonstration description will appear in [2]*
- *Y. Xiao, L. Xiong, L. Fan. and S. Goryczka are with the Department of Mathematics & Computer Science, Emory University, Atlanta, GA, 30322.*
  E-mail: {yonghui.xiao, lxiong, lfan3, sgorycz}@emory.edu
- *This research was supported in part by NSF grant CNS-1117763, a Cisco Research Award, and an Emory URC grant.*

is the advantage of the non-interactive release compared to using the interactive mechanism to answer the queries directly. A common mechanism providing differentially private answers is to add carefully calibrated noise to each query determined by the privacy parameter and the sensitivity of the query. The composability of differential privacy [11] ensures privacy guarantees for a sequence of differentially-private computations with additive privacy depletions in the worst case. Given an overall privacy requirement or budget, expressed as a privacy parameter, it can be allocated to subroutines or each query in the query sequence to ensure the overall privacy. When the number of queries grow, each query gets a lower privacy budget which requires a larger noise to be added. When there are multiple users, they have to share a common privacy budget which degrades the utility rapidly. The non-interactive approach essentially exploits the data distribution and the query workload and uses a carefully designed algorithm or query strategy such that the overall noise is minimized for a particular class of queries. As a result, the partitioning strategy and the algorithm implementing the strategy for generating the query sequence to the interface are crucial to the utility of the resulting histogram or synthetic dataset.

Contributions. We study differentially private histogram release for random query workload in this paper and propose partitioning and estimation algorithms with formal utility analysis and experimental evaluations. We summarize our contributions below.

- We study two multidimensional partitioning strategies for differentially private histogram release: 1) a baseline cell-based partitioning strategy for releasing an equi-width cell histogram, and 2) an innovative 2-phase kd-tree (k-dimensional tree) based space partitioning strategy for releasing a v-optimal histogram. There are several innovative features in our 2-phase strategy. First, we incorporate a uniformity measure in the partitioning process which seeks to produce partitions that are close to uniform so that approximation errors within partitions are minimized, essentially resulting in a differentially private v-optimal histogram. Second, we implement the strategy using a two-phase algorithm that generates the kd-tree partitions based on the cell histogram so that the access to the differentially private interface is minimized.
- We formally analyze the utility of the released histograms and quantify the errors for answering linear distribute queries such as counting queries. We show that the cell histogram provides bounded query error for any input data. We also show that the v-optimal histogram combined with a simple query estimation scheme achieves bounded query error and superior utility than existing approaches for “smoothly” distributed data. We formally characterize the “smoothness” property of the input data that guarantees the optimality of the algorithm.
- We implement and experimentally evaluate several applications using the released histograms, including counting queries, classification, and blocking for record linkage. We compare our approach with other existing privacy-preserving algorithms and show the benefit of our approach.

2 Related works

Privacy preserving data analysis and publishing has received considerable attention in recent years. We refer readers to [3], [4], [5] for several up-to-date surveys. We briefly review here the most relevant work to our paper and discuss how our work differs from existing work.

There has been a series of studies on interactive privacy preserving data analysis based on the notion of differential privacy [6], [7], [3], [5]. A primary approach proposed for achieving differential privacy is to add Laplace noise [6], [3], [7] to the original results. McSherry and Talwar [12] give an alternative method to implement differential privacy based on the probability of a returned result, called the exponential mechanism. Roth and Roughgarden [13] proposes a median mechanism which improves upon the Laplace mechanism. McSherry implemented the interactive data access mechanism into PINQ [11], a platform providing a programming interface through a SQL-like language.

A few works started addressing non-interactive data release that achieves differential privacy. Blum et al. [9] proved the possibility of non-interactive data release satisfying differential privacy for queries with polynomial VC-dimension, such as predicate queries. It also proposed an inefficient algorithm based on the exponential mechanism. The result largely remains theoretical and the general algorithm is inefficient for the complexity and required data size. [10] further proposed more efficient algorithms with hardness results obtained and it remains a key open problem to find efficient algorithms for non-interactive data release with differential privacy for many domains. [14] pointed out that a natural approach to side-stepping the hardness is relaxing the utility requirement, and not requiring accuracy for *every* input database.

Several recent work studied differentially private mechanisms for particular kinds of data such as search logs [15], [16] or set-valued data [17]. Others proposed algorithms for specific applications or optimization goals such as recommender systems [18], record linkage [19], data mining [20], or differentially private data cubes with minimized overall cuboid error [21]. It is important to note that [19] uses several tree strategies including kd-tree in its partitioning step and our results show that our 2-phase uniformity-driven kd-tree strategy achieves better utility for random count queries.

A few works considered releasing data for predictive count queries and are closely related to ours. X. Xiao et al. [22] developed an algorithm using wavelet transforms. [23] generates differentially private histograms for single dimensional range queries through a hierarchical partitioning approach and a consistency check technique. [24] proposes a query matrix mechanism that generates an optimal query strategy based on the query workload of linear count queries and further mapped the work in [22] and [23] as special query strategies that can be represented by a query matrix. It is worth noting that the cell-based partitioning in our approach is essentially the identity query matrix referred in [24]. While we will leverage the query matrix framework to formally analyze our approach, it is important to note that the above mentioned query strategies are data-oblivious in that they are determined by the query workload, a static wavelet matrix, or hierarchical matrix without taking into consideration the underlying data. On the other hand, our 2-phase kd-tree based partitioning is designed to explicitly exploit the smoothness of the underlying data indirectly observed by the differentially private interface and the final query matrix corresponding to the released histogram is dependent on the approximate data distribution. We are also aware of a forthcoming work [25] and will compare with it as the proceedings becomes available.

In summary, our work complements and advances the above works in that we focus on differentially private histogram release for random query workload using a multidimensional partitioning approach that is "data-aware". Sharing the insights from [14], [17], our primary viewpoint is that it is possible and desirable in practice to design adaptive or data-dependent heuristic mechanisms for differentially private data release for useful families or subclasses of databases and applications. Our approach provides formal utility guarantees for a class of queries and also supports a variety of applications including general OLAP, classification and record linkage.

3 Preliminaries and Definitions

In this section, we formally introduce the definitions of differential privacy, the data model and the queries we consider, as well as a formal utility notion called (ϵ, δ)-usefulness. Matrices and vectors are indicated with bold letters (e.g $\mathbf{H}$, $\mathbf{x}$) and their elements are indicated as H_{ij} or x_i. While we will introduce mathematical notations in this section and subsequent sections, Table 3 lists the frequently-used symbols for references.

TABLE 1
Frequently used symbols

Symbol	Description
n	number of records in the dataset
m	number of cells in the data cube
x_i	original count of cell $i (1 \leq i \leq m)$
y_i	released count of cell i in cell histogram
y_p	released count of partition p in subcube histogram
n_p	size of partition p
s	size of query range
α, α_1, α_2	differential privacy parameters
γ	smoothness parameter

3.1 Differential Privacy

Definition 3.1 (α-Differential privacy [7]): In the interactive model, an access mechanism $\mathcal{A}$ satisfies α-differential privacy if for any neighboring databases[1] D_1 and D_2, for any query function Q, $r \subseteq Range(Q)$, $\mathcal{A}_Q(D)$ is the mechanism to return an answer to query $Q(D)$,

$$Pr[\mathcal{A}_Q(D_1) = r] \leq e^{\alpha} Pr[\mathcal{A}_Q(D_2) = r]$$

In the non-interactive model, a data release mechanism $\mathcal{A}$ satisfies α-differential privacy if for all neighboring database D_1 and D_2, and released output $\hat{D}$,

$$Pr[\mathcal{A}(D_1) = \hat{D}] \leq e^{\alpha} Pr[\mathcal{A}(D_2) = \hat{D}]$$

Laplace Mechanism. To achieve differential privacy, we use the Laplace mechanism [6] that adds random noise of Laplace distribution to the true answer of a query Q, $\mathcal{A}_Q(D) = Q(D) + \tilde{N}$, where $\tilde{N}$ is the Laplace noise. The magnitude of the noise depends on the privacy level and the query's sensitivity.

Definition 3.2 (Sensitivity): For arbitrary neighboring databases D_1 and D_2, the sensitivity of a query Q, denoted by S_Q, is the maximum difference between the query results of D_1 and D_2,

$$S_Q = max|Q(D_1) - Q(D_2)| \tag{1}$$

To achieve α-differential privacy for a given query Q on dataset D, it is sufficient to return $Q(D) + \tilde{N}$ in place of the original result $Q(D)$ where we draw $\tilde{N}$ from $Lap(S_Q/\alpha)$ [6].

Composition. The composability of differential privacy [11] ensures privacy guarantees for a sequence of differentially-private computations. For a general series of analysis, the privacy parameter values add up, i.e. the privacy guarantees degrade as we expose more information. In a special case that the analyses operate on disjoint subsets of the data, the ultimate privacy guarantee depends only on the worst of the guarantees of each analysis, not the sum.

[1] We use the definition of unbounded neighboring databases [8] consistent with [11] which treats the databases as multisets of records and requires their symmetric difference to be 1.

Theorem 3.1 (Sequential Composition [11]): Let M_i each provide α_i-differential privacy. The sequence of M_i provides $(\sum_i \alpha_i)$-differential privacy.

Theorem 3.2 (Parallel Composition [11]): If D_i are disjoint subsets of the original database and M_i provides α-differential privacy for each D_i, then the sequence of M_i provides α-differential privacy.

Differential Privacy Interface. A privacy interface such as PINQ [11] can be used to provide a differentially private interface to a database. It provides operators for database aggregate queries such as count (**NoisyCount**) and sum (**NoisySum**) which uses Laplace noise and the exponential mechanism to enforce differential privacy. It also provides a **Partition** operator that can partition the dataset based on the provided set of candidate keys. The **Partition** operator takes advantage of parallel composition and thus the privacy costs do not add up.

3.2 Data and Query Model

Data Model. Consider a dataset with N nominal or discretized attributes, we use an N-dimensional data cube, also called a base cuboid in the data warehousing literature [26], [21], to represent the aggregate information of the data set. The records are the points in the N-dimensional data space. Each cell of a data cube represents an aggregated measure, in our case, the count of the data points corresponding to the multidimensional coordinates of the cell. We denote the number of cells by m and $m = |dom(A_1)| * \cdots * |dom(A_N)|$ where $|dom(A_i)|$ is the domain size of attribute A_i. We use the term "partition" to refer to any sub-cube in the data cube.

ID	Age	Income
1	20~30	10~20K
2	30~40	30K
…	…	…
n	30~40	60K

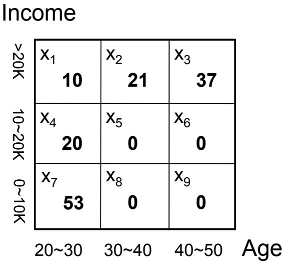


Fig. 2. Running example: original data represented in a relational table (left) and a 2-dimensional count cube (right)

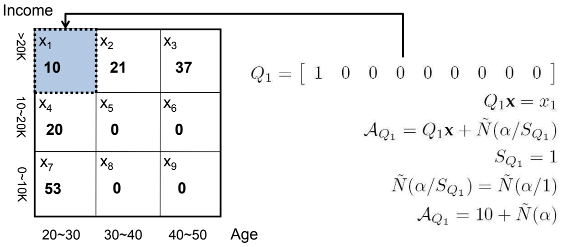
Figure 2 shows an example relational dataset with attribute age and income (left) and a two-dimensional count data cube or histogram (right). The domain values of age are 20$\sim$30, 30$\sim$40 and 40$\sim$50; the domain values of income are 0$\sim$10K, 10$K$$\sim$20$K$ and $> 20K$. Each cell in the data cube represents the population count corresponding to the age and income values.

Query Model. We consider linear counting queries that compute a linear combination of the count values in the data cube based on a query predicate. We can represent the original data cube, e.g. the counts of all cells, by an m-dimensional column vector $\mathbf{x}$ shown below.

$$\mathbf{x} = \begin{bmatrix} 10 & 21 & 37 & 20 & 0 & 0 & 53 & 0 & 0 \end{bmatrix}^T$$

Definition 3.3 (Linear query [24]): A linear query Q can be represented as an m-dimensional boolean vector $\mathbf{Q} = [q_1 \dots q_n]$ with each $q_i \in \mathbb{R}$. The answer to a linear query $\mathbf{Q}$ on data vector $\mathbf{x}$ is the vector product $\mathbf{Qx} = q_1x_1 + \cdots + q_nx_n$.

In this paper, we consider counting queries with boolean predicates so that each q_i is a boolean variable with value 0 or 1. The sensitivity of the counting queries, based on equation (1), is $S_Q = 1$. We denote s as the query range size or query size, which is the number of cells contained in the query predicate, and we have $s = |\mathbf{Q}|$. For example, a query Q_1 asking the population count with age $= [20, 30]$ and income $> 30k$, corresponding to x_1, is shown as a query vector in Figure 3. The size of this query is 1. It also shows the original answer of Q_1 and a perturbed answer with Laplace noise that achieves α-differential privacy. We note that the techniques and proofs are generalizable to real number query vectors.



$$Q_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$Q_1\mathbf{x} = x_1$$
$$\mathcal{A}_{Q_1} = Q_1\mathbf{x} + \tilde{N}(\alpha/S_{Q_1})$$
$$S_{Q_1} = 1$$
$$\tilde{N}(\alpha/S_{Q_1}) = \tilde{N}(\alpha/1)$$
$$\mathcal{A}_{Q_1} = 10 + \tilde{N}(\alpha)$$

Fig. 3. Example: a linear counting query

Definition 3.4 (Query matrix [24]): A query matrix is a collection of linear queries, arranged by rows to form an $p \times m$ matrix.

Given a $p \times m$ query matrix $\mathbf{H}$, the query answer for $\mathbf{H}$ is a length-p column vector of query results, which can be computed as the matrix product $\mathbf{Hx}$. For example, an $m \times m$ identity query matrix $\mathbf{I}_m$ will result in a length-m column vector consisting of all the cell counts in the original data vector $\mathbf{x}$.

A data release algorithm, consisting of a sequence of designed queries using the differential privacy interface, can be represented as a query matrix. We will use this query matrix representation in the analysis of our algorithms.

3.3 Utility Metrics

We formally analyze the utility of the released data by the notion of (ϵ, δ)-usefulness [9].

Definition 3.5 ((ϵ, δ)-usefulness [9]): A database mechanism $\mathcal{A}$ is (ϵ, δ)-useful for queries in class C if with probability $1 - \delta$, for every $Q \in C$, and every database D, $\mathcal{A}(D) = \hat{D}$, $|Q(\hat{D}) - Q(D)| \leq \epsilon$.

In this paper, we mainly focus on linear counting queries to formally analyze the released histograms. We will discuss and experimentally show how the released histogram can be used to support other types of OLAP queries such as sum and average and other applications such as classification.

3.4 Laplace Distribution Properties

We include a general lemma on probability distribution and a theorem on the statistical distribution of the summation of multiple Laplace noises, which we will use when analyzing the utility of our algorithms.

Lemma 3.1: If $Z \sim f(z)$, then $aZ \sim \frac{1}{a}f(z/a)$ where a is any constant.

Theorem 3.3: [27] Let $f_n(z,\alpha)$ be the PDF of $\sum_{i=1}^{n} \tilde{N}_i(\alpha)$ where $\tilde{N}_i(\alpha)$ are i.i.d. Laplace noise Lap$(1/\alpha)$,

$$f_n(z,\alpha) = \frac{\alpha^n}{2^n\Gamma^2(n)} exp(-\alpha|z|) \int_0^\infty v^{n-1}(|z| + \frac{v}{2\alpha})^{n-1} e^{-v} dv \quad (2)$$

4 MULTIDIMENSIONAL PARTITIONING

4.1 Motivation and Overview

For differentially private histogram release, a multi-dimensional histogram on a set of attributes is constructed by partitioning the data points into mutually disjoint subsets called *buckets* or *partitions*. The counts or frequencies in each bucket is then released. Any access to the original database is conducted through the differential privacy interface to guarantee differential privacy. The histogram can be then used to answer random counting queries and other types of queries.

The partitioning strategy will largely determine the utility of the released histogram to arbitrary counting queries. Each partition introduces a bounded Laplace noise or *perturbation error* by the differential privacy interface. If a query predicate covers multiple partitions, the perturbation error is aggregated. If a query predicate falls within a partition, the result has to be estimated assuming certain distribution of the data points in the partition. The dominant approach in histogram literature is making the *uniform distribution assumption*, where the frequencies of records in the bucket are assumed to be the same and equal to the average of the actual frequencies [28]. This introduces an *approximation error*.

Example. We illustrate the errors and the impact of different partitioning strategies through an example shown in Figure 4. Consider the data in Figure 2. As a baseline strategy, we could release a noisy count for each of the cells. In a data-aware strategy, as if we know the original data, the 4 cells, x_5, x_6, x_8, x_9, can be grouped into one partition and we release a single noisy count for the partition. Note that the noise are independently generated for each cell or partition. Because the sensitivity of the counting query is 1 and the partitioning only requires parallel composition of differential privacy, the magnitude of the noise in the two approaches are the same. Consider a query, count(x_5), asking the count of data points in the region x_5. For the baseline strategy, the query error is $\tilde{N}$ which only consists of the perturbation error. For the data-aware strategy, the best estimate for the answer based on the uniform distribution assumption is $0 + \tilde{N}/4$. So the query error is $\tilde{N}/4$. In this case, the approximation error is 0 because the cells in the partition are indeed uniform. If not, approximation error will be introduced. In addition, the perturbation error is also amortized among the cells. Clearly, the data-aware strategy is desired in this case.

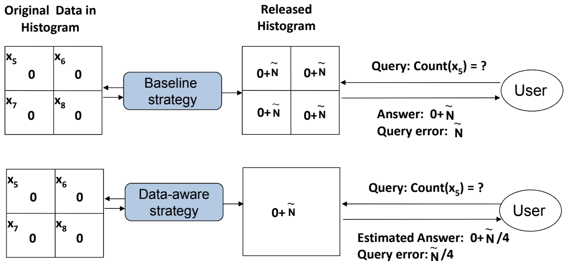


Fig. 4. Baseline strategy vs. data-aware strategy

In general, a finer-grained partitioning will introduce smaller approximation errors but larger aggregated perturbation errors. Finding the right balance to minimize the overall error for a random query workload is a key question. Not surprisingly, finding the optimal multi-dimensional histogram, even without the privacy constraints, is a challenging problem and optimal partitioning even in two dimensions is NP-hard [29]. Motivated by the above example and guided by the composition theorems, we summarize our two design goals: 1) generate uniform or close to uniform partitions so that the approximation error within the partitions is minimized, essentially generating a v-optimal histogram [30]; 2) carefully and efficiently use the privacy budget to minimize the perturbation error. In this paper, we first study the most fine-grained cell-based partitioning as a baseline strategy, which results in a equi-width histogram and does not introduce approximation error but only perturbation error. We then propose a 2-phase kd-tree (k-dimensional tree) based partitioning strategy that results in an v-optimal histogram and seeks to minimize both the perturbation and approximation error.

4.2 A Baseline Cell Partitioning Strategy

A simple strategy is to partition the data based on the domain and then release a noisy count for each cell which results in a equi-width cell histogram. Figure 5 illustrate this baseline strategy. The implementation is quite simple, taking advantage the **Partition** operator followed by **NoisyCount** on each partition, shown in Algorithm 1.

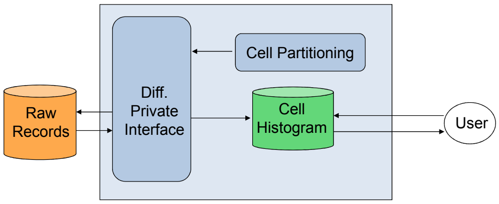


Fig. 5. Baseline cell partitioning

```
Algorithm 1 Baseline cell partitioning algorithm
Require: $\alpha$: differential privacy budget
  1. Partition the data based on all domains.
  2. Release NoisyCount of each partition using privacy parameter $\alpha$
```

Privacy Guarantee. We present the theorem below for the cell partitioning algorithm which can be derived directly from the compobility theorems.

Theorem 4.1: Algorithm 1 achieves α-differential privacy.

Proof: Because every cell is a disjoint subset of the original database, according to theorem 3.2, it's α-differentially private. □

Error Quantification. We present a lemma followed by a theorem that states a formal utility guarantee of cell-based partitioning for linear distributive queries.

Lemma 4.1: If $\tilde{N}_i$ $(i = 1 \ldots m)$ is a set of random variables i.i.d from $Lap(b)$ with mean 0, given $0 < \epsilon < 1$, the following holds:

$$Pr[\sum_{i=1}^{m} |\tilde{N}_i| \le \epsilon] \ge 1 - m \cdot exp(-\frac{\epsilon}{mb}) \qquad (3)$$

Proof: Let $\epsilon_1 = \epsilon/m$, given the Laplace distribution, we have

$$Pr[|\tilde{N}_i| > \epsilon_1] = 2\int_{\epsilon_1}^{\infty} \frac{1}{2b} exp(-\frac{x}{b}) = e^{-\epsilon_1/b}$$

then

$$Pr[|\tilde{N}_i| \le \epsilon_1] = 1 \text{ - } Pr[|\tilde{N}_i| > \epsilon_1] = 1 - e^{-\epsilon_1/b}$$

If each $|\tilde{N}_i| \le \epsilon_1$, we have $\sum_{i=1}^{m} |\tilde{N}_i| \le m \cdot \epsilon_1 = \epsilon$, so we have

$$Pr[\sum_{i=1}^{m} |\tilde{N}_i| \le \epsilon] \ge Pr[|\tilde{N}_i| \le \epsilon_1]^m = (1 - e^{-\epsilon_1/b})^m$$

Let $F(x) = (1 - x)^m + mx - 1$. The derivative of $F(x)$ is $F'(x) = -m(1 - x)^{m-1} + m = m(1 - (1 - x)^{m-1}) \ge 0$ when $0 < x < 1$. Note that $0 < e^{-\epsilon_1/b} < 1$, so $F(e^{-\epsilon_1/b}) \ge F(0) = 0$. We get

$$(1 - e^{-\epsilon_1/b})^m \ge 1 - m \cdot e^{-\epsilon_1/b}$$

Recall $\epsilon_1 = \epsilon/m$, we derive equation (3). □

Theorem 4.2: The released $\hat{D}$ of algorithm 1 maintains (ϵ, δ)-usefulness for linear counting queries, if $\alpha \ge \frac{m \cdot ln(\frac{m}{\delta})}{\epsilon}$, where m is the number of cells in the data cube.

Proof: Given original data D represented as count vector $\mathbf{x}$, using the cell partitioning with Laplace mechanism, the released data $\hat{D}$ can be represented as $\mathbf{y} = \mathbf{x} + \tilde{\mathbf{N}}$, where $\tilde{\mathbf{N}}$ is a length-m column vector of Laplace noises drawn from $Lap(b)$ with $b = 1/\alpha$.

Given a linear counting query Q represented as $\mathbf{q}$ with query size s $(s \le m)$, we have $Q(D) = \mathbf{qx}$, and

$$Q(\hat{D}) = \mathbf{Qy} = \mathbf{Qx} + \mathbf{Q\tilde{N}} = \mathbf{Qx} + \sum_{i=1}^{s} |\tilde{N}_i|$$

With Lemma 4.1, we have

$$Pr[|Q(D) - Q(\hat{D})| \le \epsilon] = Pr[\sum_{i=1}^{s} |\tilde{N}_i| \le \epsilon] \ge 1 - m \cdot exp(-\frac{\epsilon}{mb})$$

If $m \cdot exp(-\frac{\epsilon}{mb}) \le \delta$, then

$$Pr[|Q(x, D) - Q(x, \hat{D})| \le \epsilon] \ge 1 - \delta$$

In order for $m \cdot exp(-\frac{\epsilon}{mb}) \le \delta$, given $b = 1/\alpha$, we derive the condition $\alpha \ge \frac{m \cdot ln(\frac{m}{\delta})}{\epsilon}$. □

4.3 DPCube: Two-Phase Partitioning

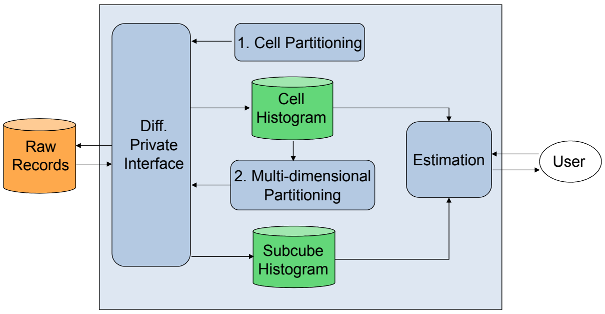


Fig. 6. DPCube: 2-phase partitioning

We now present our DPCube algorithm. DPCube uses an innovative two-phase partitioning strategy as shown in Figure 6. First, a cell based partitioning based on the domains (not the data) is used to generate a fine-grained equi-width cell histogram (as in the baseline strategy), which gives an approximation of the original data distribution. We generate a synthetic database D_c based on the cell histogram. Second, a multi-dimensional kd-tree based partitioning is performed on D_c to obtain uniform or close to uniform partitions. The resulting partitioning keys are used to **Partition** the original database and obtain a **NoisyCount** for each of the partitions, which results in a v-optimal histogram [30]. Finally, given a user-issued query, an estimation component uses either the v-optimal histogram or both histograms to compute an answer. The key innovation of our algorithm is that it is data-aware or adaptive because the multi-dimensional kd-tree based partitioning is based on the cell-histogram from the first phase and hence exploits the underlying data distribution indirectly observed by the perturbed cell-histogram. Essentially, the kd-tree is based on an approximate distribution of the original data. The original database is not queried during the kd-tree construction which saves the privacy budget. The overall privacy budget is efficiently used and divided between the two phases only for querying the NoisyCount for cells and partitions. Algorithm 2 presents a sketch of the algorithm.

The key step in Algorithm 2 is the multi-dimensional partitioning step. It uses a kd-tree based space partitioning strategy that seeks to produce close to uniform partitions in order to minimize the estimation error within a partition. It starts from the root node which covers the entire space. At each step, a splitting dimension and a split value from the range of the current partition on that dimension are chosen heuristically to divide the space into subspaces. The algorithm repeats until a pre-defined requirement are met. In contrast to traditional kd-tree index construction which desires a balanced tree, our main design goal is to generate uniform or close to uniform partitions so that the approximation error within the partitions is minimized. Thus we propose a uniformity based heuristic to make the decision whether to split the current partition and to select the best splitting point. There are several metrics that can be used to measure the uniformity of a partition such as information entropy and variance. In our experiments, we use variance as the metric. Concretely, we do not split a partition if its variance is greater than a threshold, i.e. it is close to uniform, and split it otherwise. In order to select the best splitting point, we choose the dimension with the largest range and the splitting point that minimizes the accumulative weighted variance of resulting partitions. This heuristic is consistent with the goal of a v-optimal histogram which places the histogram bucket boundaries to minimize the cumulative weighted variance of the buckets. Algorithm 3 describes the step 2 of Algorithm 2.

Privacy Guarantee. We present the theorem below for the 2-phase partitioning algorithm which can be derived directly from the composibility theorems.

Theorem 4.3: Algorithm 2 is α-differentially private.

Proof: Step 2 and Step 5 are α_1,α_2-differentially private. So the sequence is α-differentially private because of theorem 3.1 with $\alpha = \alpha_1 + \alpha_2$. □

Algorithm 2 2-phase partitioning algorithm

```
Require: $\beta$: number of cells;
  $\alpha$: the overall privacy budget
  Phase I:
  1. Partition the original database based on all domains.
  2. get NoisyCount of each partition using privacy parameter $\alpha_1$ and generate a synthetic dataset $D_c$.
  Phase II:
  3. Partition $D_c$ by algorithm 3.
  4. Partition the original database based on the partition keys returned from step 3.
  5. release NoisyCount of each partition using privacy parameter $\alpha_2 = \alpha - \alpha_1$
```

Algorithm 3 Kd-tree based v-optimal partitioning

```
Require: $D_t$: input database; $\xi_0$: variance threshold;
  if variance of $D_t > \xi_0$ then
    Find a dimension and splitting point $m$ which minimizes the cumulative weighted variance of the two resulting partitions;
    Split $D_t$ into $D_{t1}$ and $D_{t2}$ by $m$.
    partition $D_{t_1}$ and $D_{t_2}$ by algorithm 3.
  end if
```

Query Matrix Representation. We now illustrate how the proposed algorithm can be represented as a query matrix. We denote $\mathbf{H}$ as the query matrix generating the released data in our algorithm and we have $\mathbf{H} = [\mathbf{H}_{\text{II}}; \mathbf{H}_{\text{I}}]$, where $\mathbf{H}_{\text{I}}$ and $\mathbf{H}_{\text{II}}$ correspond to the query matrix in the cell partitioning and kd-tree partitioning phase respectively. $\mathbf{H}_{\text{I}}$ is an Identity matrix with m rows, each row querying the count of one cell. $\mathbf{H}_{\text{II}}$ contains all the partitions generated by the second phase. We use $\tilde{\mathbf{N}}(\alpha)$ to denote the column noise vector and each noise $\tilde{N}_i$ is determined by a differential privacy parameter (α_1 in the first phase and α_2 in the second phase respectively). The released data is $\mathbf{y} = \mathbf{H}\mathbf{x} + \tilde{\mathbf{N}}$. It consists of the cell histogram $\mathbf{y}_{\text{I}}$ in phase I and the v-optimal histogram $\mathbf{y}_{\text{I}}$ in phase II: $\mathbf{y}_{\text{I}} = (\mathbf{H}_{\text{I}})\mathbf{x} + \tilde{\mathbf{N}}(\alpha_1)$, $\mathbf{y}_{\text{II}} = (\mathbf{H}_{\text{II}})\mathbf{x} + \tilde{\mathbf{N}}(\alpha_2)$.

Using our example data from Figure 2, the query matrix $\mathbf{H}$ consisting of $\mathbf{H}_{\text{I}}$ and $\mathbf{H}_{\text{II}}$ and the released data consisting of the cell histogram $\mathbf{y}_{\text{I}}$ and subcube histogram $\mathbf{y}_{\text{II}}$ are shown in Equation (4, 5). The histograms are also illustrated in Figure 7.

$$\mathbf{H}_{\text{I}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{y}_{\text{I}} = \begin{bmatrix} 10 + \tilde{N}_1(\alpha_1) \\ 21 + \tilde{N}_2(\alpha_1) \\ 37 + \tilde{N}_3(\alpha_1) \\ 20 + \tilde{N}_4(\alpha_1) \\ 0 + \tilde{N}_5(\alpha_1) \\ 0 + \tilde{N}_6(\alpha_1) \\ 53 + \tilde{N}_7(\alpha_1) \\ 0 + \tilde{N}_8(\alpha_1) \\ 0 + \tilde{N}_9(\alpha_1) \end{bmatrix} \tag{4}$$

$$\mathbf{H}_{\text{II}} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} \mathbf{y}_{\text{II}} = \begin{bmatrix} 51 + \tilde{N}_{10}(\alpha_2) \\ 37 + \tilde{N}_{11}(\alpha_2) \\ 53 + \tilde{N}_{12}(\alpha_2) \\ 0 + \tilde{N}_{13}(\alpha_2) \end{bmatrix} \tag{5}$$

4.4 Query Estimation and Error Quantification

Once the histograms are released, given a random user query, the estimation component will compute an answer using the released histograms. We study

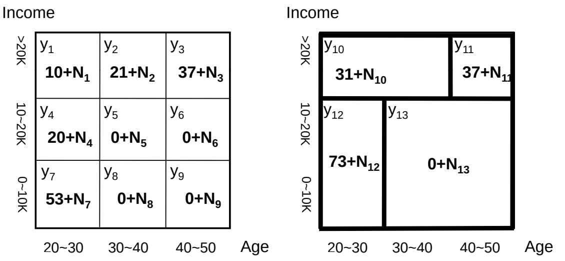


Fig. 7. Example: released cell histogram (left) and subcube histogram (right)

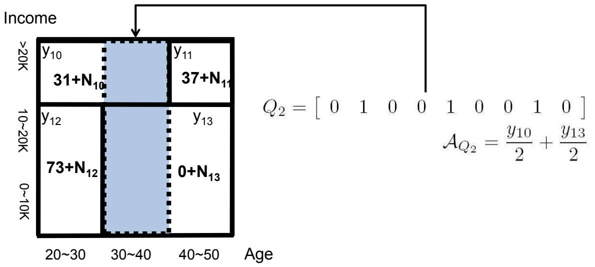


Fig. 8. Example: query estimation with uniform estimation using subcube histogram

two techniques and formally quantify and compare the errors of the query answer they generate. The first one estimates the query answer using only the subcube histogram assuming a uniform distribution within each partition. As an alternative approach, we adopt the least squares (LS) method, also used in [24], [23], to our two-phase strategy. The basic idea is to use both cell histogram and subcube histogram and find an approximate solution of the cell counts to resolve the inconsistency between the two histograms.

Uniform Estimation using Subcube Histogram. Based on our design goal to obtain uniform partitions for the subcube histogram, we make the *uniform distribution assumption* in the subcube histogram, where the counts in each cell within a partition are assumed to be the same. Given a partition p, we denote n_p as the size of the partition in number of cells. Hence the count of each cell within this partition is y_p/n_p where y_p is the noisy count of the partition. We denote $\hat{\mathbf{x}}_H$ as the estimated cell counts of the original data. If a query predicate falls within one single partition, the estimated answer for that query is $\frac{s}{n_p}y_p$ where s is the query range size. Given a random linear query that spans multiple partitions, we can add up the estimated counts of all the partitions within the query predicate. In the rest of the error analysis, we only consider queries within one partition as the result can be easily extended for the general case.

Figure 8 shows an example. Given a query Q_2 on population count with age $= [30, 40]$, the original answer is $x_2 + x_5 + x_8$. Using the subcube histogram, the query overlaps with partition y_{10} and partition y_{13}. Using the uniform estimation, the estimated answer is $\frac{y_{10}}{2} + \frac{y_{13}}{2}$.

Error Quantification for Uniform Estimation. We derive the (ϵ, δ)-usefulness and the expected error for the uniform estimation method. To formally understand the distribution properties of input database that guarantees the optimality of the method, we first define the smoothness of the distribution within a partition or database similar to [31]. The difference is that [31] defines only the upper bound, while our definition below defines the bound of differences of all cell counts. Intuitively, the smaller the difference, the smoother the distribution.

Definition 4.1 (γ-smoothness): Denote $\mathbf{x}$ the original counts of cells in one partition, $\forall x_i, x_j \in \mathbf{x}$, if $|x_j - x_i| \le \gamma$, then the distribution in the partition satisfies γ-smoothness.

We now present a theorem that formally analyzes the utility of the released data if the input database satisfies γ-smoothness, followed by a theorem for the general case.

Theorem 4.4: For γ-smooth data $\mathbf{x}$, given a query q with size s, n_p the size of the partition, the uniform estimation method is (ϵ, δ)-useful if equation (6) holds; the upper bound of the expected absolute error $\mathbb{E}(\epsilon_H)$ is shown in equation (7).

$$\gamma \le (\epsilon + \frac{s \cdot log\delta}{\alpha_2 n_p})/min(s, n_p - s) \tag{6}$$

$$\mathbb{E}(\epsilon_H) \le \gamma[min(s, n_p - s)] + \frac{s}{\alpha_2 n_p} \tag{7}$$

Proof: Given a query vector $\mathbf{Q}$, the answer using the original data is $\mathbf{Qx}$, the estimated answer using the released partition is $\frac{s}{n_p}y_p$ where y_p is the released partition count, n_p is the partition size, and s is the query size. The released count $y_p = \sum_{i=1}^{n_p} x_i + \tilde{N}(\alpha_2)$. So we have the absolute error as

$$\epsilon_H = |\frac{s}{n_p}y_p - \mathbf{Qx}| = |(\frac{s}{n_p}\sum_{i=1}^{n_p} x_i - \mathbf{Qx}) + \frac{s}{n_p}\tilde{N}(\alpha_2)|$$

According to the definition of γ-smoothness, $\forall i, j, |x_j - x_i| \le \gamma$, then $|\frac{s}{n_p}\sum_{i=1}^{n_p} x_i - \mathbf{Qx}| \le min(s, n_p - s)\gamma$. Because of the symmetry of the PDF of $\tilde{N}(\alpha_2)$, we have

$$\begin{aligned}\epsilon_H &\le |min(s, n_p - s)\gamma + \frac{s}{n_p}\tilde{N}(\alpha_2)| \\ &\le min(s, n_p - s)\gamma + |\frac{s}{n_p}\tilde{N}(\alpha_2)|\end{aligned}$$

By Lemma 3.1, we know the PDF of $\frac{s}{n_p}\tilde{N}(\alpha_2)$. To satisfy (ϵ, δ)-usefulness, we require $Pr(\epsilon_H \le \epsilon) \ge 1 - \delta$, then we derive the condition as in equation (6).

The expected absolute error is

$$\begin{aligned}\mathbb{E}(\epsilon_H) &= \frac{s}{n_p}\sum_{i=1}^{n_p} x_i - \mathbf{Qx} + \frac{s}{n_p}\mathbb{E}|\tilde{N}(\alpha_2)| \\ &\le min(s, n_p - s)\gamma + \frac{s}{n_p}\mathbb{E}|\tilde{N}(\alpha_2)|\end{aligned}$$

Based on the PDF of $\frac{s}{n_p}\tilde{N}(\alpha_2)$, we have

$$\mathbb{E}|\tilde{N}(\alpha_2)| = 1/\alpha_2$$

and hence derive equation (7). □

From theorem 4.4, we can conclude that if the input data is smoothly distributed or very sparse, γ would be small, the error would be small. In this case, our algorithm achieves the best result. In the general case, if we do not know the distribution properties of the input data, we present Theorem 4.5 to quantify the error.

Theorem 4.5: Given a linear counting query Q, the expected absolute error for the uniform estimation method, $\mathbb{E}(\epsilon_H)$, is a function of (α_1, s, η) shown in equation (8):

$$\mathbb{E}(\epsilon_H) = \int f_s(z, \alpha_1)|\eta + z|dz \tag{8}$$

where $\eta = \frac{s}{n_p}y_p - Q\mathbf{y}_{\text{I}}$, y_p is the released count of the partition in the subcube histogram, n_p the size of the partition, s is the size of the query, and $\mathbf{y}_{\text{I}}$ is the released counts of the cell histogram.

Proof: Given a query Q, the answer using the original data is $Q\mathbf{x}$, the estimated answer using the released partition is $\frac{s}{n_p}y_p$. The released partition count is $y_p = \sum_{i=1}^{n_p} x_i + \tilde{N}(\alpha_2)$. The released cell counts for the cell histogram in the first phase is $\mathbf{y}_{\text{I}} = \mathbf{x} + \tilde{\mathbf{N}}(\alpha_1)$. So we have

$$\begin{aligned}\epsilon_H = |\frac{s}{n_p}y_p - Q\mathbf{x}| &= |\frac{s}{n_p}y_p - Q(\mathbf{y}_{\text{I}} - \tilde{\mathbf{N}}(\alpha_1))| \\ &= |(\frac{s}{n_p}y_p - Q\mathbf{y}_{\text{I}}) + \sum_{i=1}^{s}\tilde{N}_i(\alpha_1)|\end{aligned}$$

Denote $\eta = \frac{s}{n_p}y_p - Q\mathbf{y}_{\text{I}}$, which is the difference (inconsistency) between the estimated answers using the cell histogram and the subcube histogram, then $\epsilon_H = |\eta + \sum_{i=1}^{s}\tilde{N}_i(\alpha_1)|$. By equation (2) , we have $\mathbb{E}(\epsilon_H)$ in equation (8). □

Least Square Estimation using Cell Histogram and Subcube Histogram. The least square (LS) method, used in [24], [23], finds an approximate (least square) solution of the cell counts that aims to resolve the inconsistency between multiple differentially private views. In our case, the cell histogram and the subcube histogram provide two views. We derive the theoretical error of the least square method and compare it with the uniform estimation method. Note that the error quantification in [24] is only applicable to the case when α_1 and α_2 are equal. We derive new result for the general case in which α_1 and α_2 may have different values.

Theorem 4.6: Given a query Q, a least square estimation $\hat{\mathbf{x}}_{LS}$ based on the cell histogram and subcube histogram, the expected absolute error of the query answer, $\mathbb{E}(\epsilon_{LS})$, is a function of $(\alpha_1, \alpha_2, n_p, s)$ in equation (9), where s is the size of Q, and n_p is the size of the partition.

$$\begin{aligned}\mathbb{E}(\epsilon_{LS}) = &\frac{(n_p+1)^3}{s^2(n_p+1-s)} \\ &\cdot \int |\epsilon| \int f_{n_p-s}(-\frac{(\epsilon - z)(n_p+1)}{s}, \alpha_1) \\ \cdot \int f_s(&\frac{(z-y)(n_p+1)}{n_p+1-s}, \alpha_1) f_1(\frac{y(n_p+1)}{s}, \alpha_2)dydzd\epsilon\end{aligned} \tag{9}$$

Proof: Given our two phase query strategy, the query matrix for the partition is $\mathbf{H} = [ones(1, n_p); I_{n_p}]$, where n_p is the partition size. Using the least square method in [24], we solve $\mathbf{H}^+ = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$, we have

$$\mathbf{H}^+ = \begin{bmatrix} \frac{1}{n_p+1} & \frac{n_p}{n_p+1} & -\frac{1}{n_p+1} & -\frac{1}{n_p+1} & \cdots & -\frac{1}{n_p+1} \\ \frac{1}{n_p+1} & -\frac{1}{n_p+1} & \frac{n_p}{n_p+1} & -\frac{1}{n_p+1} & \cdots & -\frac{1}{n_p+1} \\ \frac{1}{n_p+1} & -\frac{1}{n_p+1} & -\frac{1}{n_p+1} & \frac{n_p}{n_p+1} & \cdots & -\frac{1}{n_p+1} \\ \frac{1}{n_p+1} & -\frac{1}{n_p+1} & -\frac{1}{n_p+1} & -\frac{1}{n_p+1} & \ddots & -\frac{1}{n_p+1} \\ \frac{1}{n_p+1} & -\frac{1}{n_p+1} & -\frac{1}{n_p+1} & -\frac{1}{n_p+1} & \cdots & \frac{n_p}{n_p+1} \end{bmatrix}$$

We compute the least square estimation based on the released data $\mathbf{y}$ as $\hat{\mathbf{x}}_{LS} = \mathbf{H}^+\mathbf{y}$. So the query answer using the estimation is

$$Q\hat{\mathbf{x}}_{LS} = Q\mathbf{H}^+\mathbf{y} = Q\mathbf{H}^+(\mathbf{H}\mathbf{x} + \tilde{\mathbf{N}}(\alpha)) = Q\mathbf{x} + Q\mathbf{H}^+\tilde{\mathbf{N}}(\alpha)$$

The absolute error is

$$\begin{aligned}Q\hat{\mathbf{x}}_{LS} - Q\mathbf{x} = \frac{s}{n_p+1}\tilde{N}(\alpha_2) + \frac{n_p+1-s}{n_p+1}\sum_{i=1}^{s}\tilde{N}(\alpha_1) \\ - \frac{s}{n_p+1}\sum_{i=1}^{n_p-s}\tilde{N}(\alpha_1)\end{aligned}$$

By Lemma 3.1 and equation (2), we know the PDF of $\frac{s}{n_p+1}\tilde{N}(\alpha_2)$, $\frac{n_p+1-s}{n_p+1}\sum_{i=1}^{s}\tilde{N}(\alpha_1)$ and $\frac{s}{n_p+1}\sum_{i=1}^{n_p-s}\tilde{N}(\alpha_1)$, then by convolution formula, we have equation (9). □

We will plot the above theoretical results for both uniform estimation and least square estimation with varying parameters in Section 6 and demonstrate the benefit of the uniform estimation method, esp. when data is smoothly distributed.

5 Applications

Having presented the multidimensional partitioning approach for differentially private histogram release, we now briefly discuss the applications that the released histogram can support.

OLAP. On-line analytical processing (OLAP) is a key technology for business-intelligence applications. The computation of multidimensional aggregates, such as count, sum, max, average, is the essence of on-line analytical processing. The released histograms with the estimation can be used to answer most common OLAP aggregate queries.

Classification. Classification is a common data analysis task. Several recent works studied classification with differential privacy by designing classifier-dependent algorithms (such as decision tree) [32], [33]. The released histograms proposed in this paper can be also used as training data for training a classifier, offering a classifier-independent approach for classification with differential privacy. To compare the approach with existing solutions [32], [33], we have chosen an ID3 tree classifier algorithm. However, the histograms can be used as training data for any other classifier.

Blocking for Record Linkage. Private record linkage between datasets owned by distinct parties is another common and challenging problem. In many situations, uniquely identifying information may not be available and linkage is performed based on matching of other information, such as age, occupation, etc. Privacy preserving record linkage allows two parties to identify the records that represent the same real world entities without disclosing additional information other than the matching result. While Secure Multi-party Computation (SMC) protocols can be used to provide strong security and perfect accuracy, it incurs prohibitive computational and communication cost in practice. Typical SMC protocols require $O(n * m)$ cryptographic operations where n and m correspond to numbers of records in two datasets. In order to improve the efficiency of record linkage, *blocking* is generally in use according to [34]. The purpose of blocking is to divide a dataset into mutually exclusive blocks assuming no matches occur across different blocks. It reduces the number of record pairs to compare for the SMC protocols but meanwhile we need to devise a blocking scheme that provides strong privacy guarantee. [19] has proposed a hybrid approach with a differentially private blocking step followed by an SMC step. The differentially private blocking step adopts tree-structured space partitioning techniques and uses Laplace noise at each partitioning step to preserve differential privacy. The matching blocks are then sent to SMC protocols for further matching of the record pairs within the matching blocks which significantly reduces the total number of pairs that need to be matched by SMC.

The released histograms in this paper can be also used to perform blocking which enjoys differential privacy. Our experiments show that we can achieve approximately optimal reduction ratio of the pairs that need to be matched by SMC with appropriate setting for the threshold ξ_0. We will examine some empirical results in Section 6.

6 Experiment

We first simulate and compare the theoretical query error results from Section 4.4 for counting queries that falls within one partition to show the properties and benefits of our approach (Section 6.1). We then present a set of experimental evaluations of the quality of the released histogram in terms of weighted variance (Section 6.2), followed by evaluations of query error against random linear counting queries and a comparison with existing solutions (Section 6.3). Finally, we also implemented and experimentally evaluate the two additional applications using the released histograms, classification and blocking for record linkage, and show the benefit of our approach (Section 6.4).

6.1 Simulation Plots of Theoretical Results

As some of the theoretical results in Section 4.4 consists of equations that are difficult to compute for given parameters, we use the simulation approach to show the results and the impact of different parameters in this section. Detailed and sophisticated techniques about the simulation approach can be found in [35] and [36]. Table 6.1 shows the parameters used in the simulation experiment and their default values.

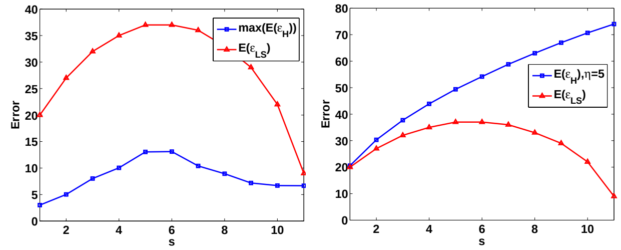
TABLE 2
Simulation parameters

parameter	description	default value
n_p	partition size	$n_p = 11$
s	query size	$s \leq n_p$
α_1, α_2	diff. privacy parameters	$\alpha_1 = 0.05$, $\alpha_2 = 0.15$
γ	smoothness parameter	$\gamma = 5$
η	inconsistency between cell and subcube histogram	$\eta = 5$

Metric. We evaluate the error of absolute count queries. Recall that $\mathbb{E}(\epsilon_{LS})$ is the expected error of the least square estimation method; $max(\mathbb{E}(\epsilon_H))$ is the upper bound of expected error of the uniform estimation method when the data is γ-smooth; $\mathbb{E}(\epsilon_H)$ is the expected error of the uniform estimation method in the general case. Note that $\mathbb{E}(\epsilon_{LS})$, $max(\mathbb{E}(\epsilon_H))$, and $\mathbb{E}(\epsilon_H)$ are derived from equation (9), (7), (8) respectively.

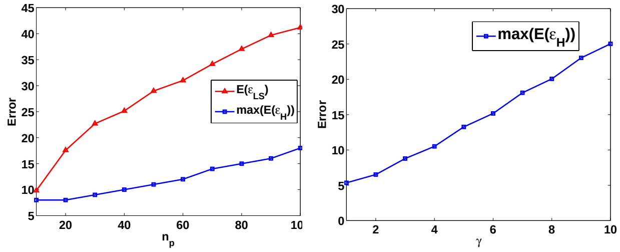
Impact of Query Size. We first study the impact of the query size. Figure 9 shows the error of the uniform and least square solutions with respect to varying query size for γ-smooth data and general case (any distribution) respectively. We can see that the highest error appears when the query size s is half of the partition size n_p. When γ is small, i.e. data is smoothly distributed, the uniform estimation method outperforms the least square method. In the general case when we do not have any domain knowledge about the data distribution, it is beneficial to use both cell histogram and subcube histogram to resolve their inconsistencies.

Impact of Privacy Budget Allocation. We now study the impact of the allocation of the overall differential privacy budget α between the two phases. The



(a) γ-smooth distribution (b) any distribution

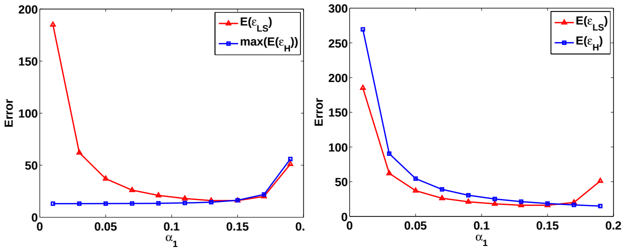
Fig. 9. Query error vs. query size s



(a) vs. partition size n_p (b) vs. smoothness γ

Fig. 11. Query error for γ-smooth distribution

overall budget is fixed with $\alpha = 0.2$. Figure 10 shows the error of the uniform and least square solutions with respect to varying query size for γ-smooth data and general case (any distribution) respectively. For the LS method, equally dividing α between the two phases or slightly more for cell-based partitioning works better than other cases. The results for the uniform estimation method present interesting trends. For smoothly distributed data, a smaller privacy budget for the partitioning phase yields better result. Intuitively, since data is smoothly distributed, it is beneficial to save the privacy budget for the second phase to get a more accurate overall partition count. On the contrary, for the random case, it is beneficial to spend the privacy budget in the first phase to get a more accurate cell counts, and hence more accurate partitioning.



(a) γ-smooth distribution (b) any distribution

Fig. 10. Query error vs. privacy budget allocation α_1

Impact of Partition Size. For γ-smooth data, the expected error is dependent on the partition size n_p and the smoothness level γ. Figure 11(a) shows the error of the uniform and least square solutions with respect to varying partition size n_p for γ-smooth data. We observe that the error increases when the partition size increases because of the increasing approximation error within the partition. Therefore, a good partitioning algorithm should avoid large partitions.

Impact of Data Smoothness. For γ-smooth data, we study the impact of the smoothness level γ on the error bound for the uniform estimation method. Figure 11(b) shows the maximum error bound with varying γ. We can see that the more smooth the data, the less error for released data. Note that the query size s is set to be nearly half size of the partition size n_p as default, which magnifies the impact of the smoothness. We observed in other parameter settings that the error increases only slowly for queries with small or big query sizes.

Impact of Inconsistency. For data with any (unknown) distribution, $\mathbb{E}(\epsilon_H)$ is a function of η, the level of inconsistency between the cell histogram and the subcube histogram. In contrast to the the γ-smooth case in which we have a prior knowledge about the smoothness of the data, here we only have this observed level of inconsistency from the released data which reflects the smoothness of the original data. Figure 12 shows the error for the uniform estimation method with varying η. Note that the error increases with increasing η and even when $\eta = 10$, the error is still lager than the error of least square method in Figure 11.

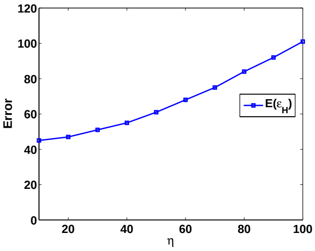
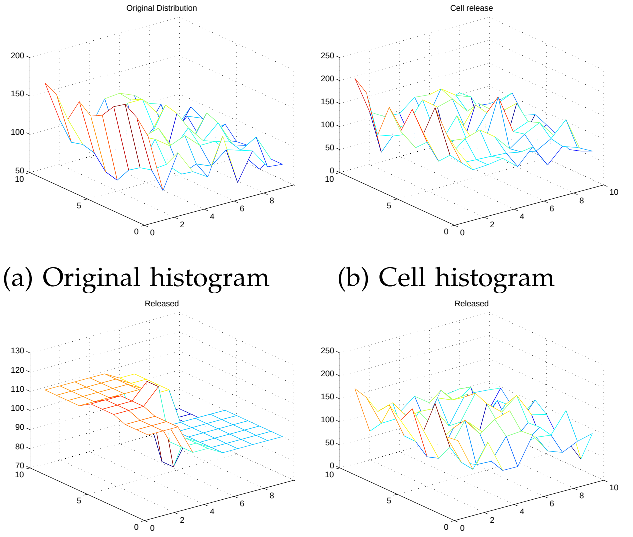


Fig. 12. Query error vs. level of inconsistency η for any distribution

6.2 Histogram Variance

We use the Adult dataset from the Census [37]. All experiments were run on a computer with Intel P8600(2∗2.4 GHz) CPU and 2GB memory. For computation simplicity and smoothness of data distribution, we only use the first 10^4 data records.

Original and Released Histograms. We first present some example histograms generated by our algorithm for the Census dataset. Figure 13(a) shows the original 2-dimensional histogram on Age and Income. Figure 13(b) shows a cell histogram generated in the first phase with $\alpha_1 = 0.05$. Figure 13(c) shows a subcube histogram generated in the second phase with estimated cell counts using uniform estimation, in which each horizontal plane represents one partition. Figure 13(d) shows an estimated cell histogram using the LS estimation using both cell histogram and subcube histogram. We systematically evaluate the utility of the released subcube histogram with uniform estimation below.

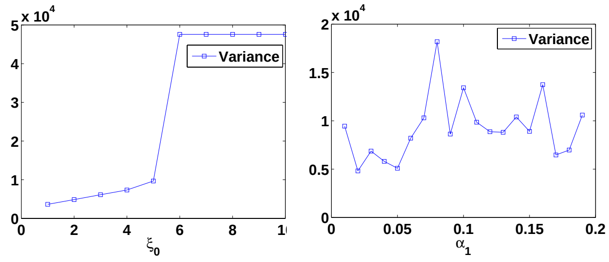


(a) Original histogram (b) Cell histogram

(c) Subcube histogram (d) Estimated cell histogram

Fig. 13. Original and released histograms

Metric. We now evaluate the quality of the released histogram using an application-independent metric, the weighted variance of the released subcube histogram. Ideally, our goal is to obtain a v-optimal histogram which minimizes the weighted variance so as to minimize the estimation error within partitions. Formally, the weighted variance of a histogram is defined as $V = \sum_{i=1}^{p} x_i V_i$, where p is the number of partitions, x_i is the number of data points in the i-th partition, and V_i is the variance in the i-th partition [30].



((a) vs. threshold ξ_0 (b) vs. privacy budget α_1

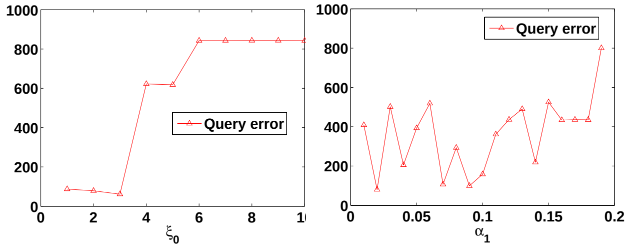
Fig. 14. Histogram variance and impact of parameters

Impact of Variance Threshold. Figure 14(a) shows the weighted variance of the released subcube histogram with respect to varying variance threshold value ξ_0 used in our algorithm. As expected, when ξ_0 becomes large, less partitioning is performed, i.e. more data points are grouped into one bucket, which increases the variance.

Impact of Privacy Budget Allocation. Figure 14(b) shows the weighted variance with respect to varying privacy budget in the first phase α_1 (with fixed ξ_0. Note that the overall privacy budget is fixed. We see that the correlation is not very clear due to the randomness of noises. Also the ξ_0 is fixed which does not reflect the different magnitude of noise introduced by different α_1. Generally speaking, the variance should decrease gradually when α_1 increases because large α_1 will introduce less noise to the first phase, which will lead to better partitioning quality.

6.3 Query Error

We evaluate the quality of the released histogram using random linear counting queries and measure the average absolute query error and compare the results with other algorithms. We use the Age and Income attributes and generated 10^5 random counting queries to calculate the average query error. The random queries consist of two range predicates on the two attributes respectively. We also implemented an alternative kd-tree strategy similar to that used in [19], referred to as hierarchical kd-tree, and another data release algorithm [23], referred to as consistency check, for comparison purposes.



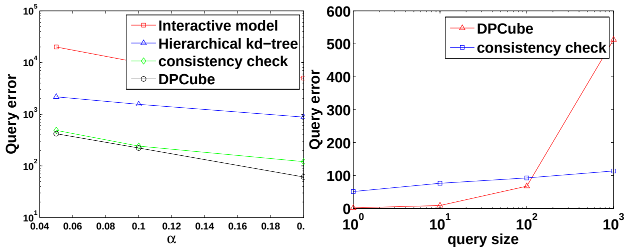
(a) vs. threshold ξ_0 (b) vs. privacy budget α_1

Fig. 15. Query error and impact of parameters

Impact of Variance Threshold. We first evaluate the impact of the threshold value ξ_0 on query error. Figure 15(a) shows the average absolute query error with respect to varying threshold values. Consistent with Figure 14, we observe that the query error increases with an increasing threshold value, due to the increased variance within partitions.

Impact of Privacy Budget Allocation. We next evaluate the impact of how to allocate the overall privacy budget α into α_1, α_2 for the two phases in the algorithm. Figure 15(b) shows the average absolute query error vs. varying α_1. We observe that small α_1 values yield better results, which complies with our theoretical result for γ-smooth data in figure 10. This verifies that the real life Adult dataset has a somewhat smooth distribution. On the other hand, for data with unknown distribution, we expect α_1 cannot be too small as it is more important for generating a more accurate partitioning.

Comparison with Other Works. We compare our approach with two representative approaches in existing works, the hierarchical kd-tree strategy used in [19], and the hierarchical partitioning with consistency check in [23]. Figure 16 shows the average absolute query error of different approaches with respect to varying privacy budget α. We can see that the DPCube algorithm achieves best utility against the random query workload because of its efficient 2-phase use of the privacy budget and the v-optimal histogram.



(a) vs. privacy budget α (b) vs. query size

Fig. 16. Query error for different approaches

We also experimented with query workloads with different query sizes for the DPCube approach and the consistency check approach to closely examine their differences. Figure 16(b) shows the average absolute query error with respect to varying query sizes for the two approaches. Note that the query size is in log scale. As proven in our theoretical result in Figure 11, the query error of our algorithm would increase with increasing query size. We can see that for queries with reasonable sizes, our 2-phase algorithm achieves better results than the consistency check algorithm. On the other hand, the consistency check algorithm is beneficial for large size queries because our algorithm favors smaller partitions with small variances which will result in large aggregated perturbation errors for large queries that spans multiple partitions.

6.4 Additional Applications

Classification. We evaluate the utility of the released histogram for classification and compare it with other differentially private classification algorithms. In this experiment, the dataset is divided into training and test subsets with 30162 and 15060 records respectively. We use the *work class*, *martial status*, *race*, and *sex* attributes as features. The class was represented by *salary* attribute with two possible values indicating if the salary is greater than \$50k or not.

For this experiment, we compare several classifiers. As a baseline, we trained a *ID3* classifier from the original data using Weka [38]. We also adapted the Weka ID3 implementation such that it can use histogram as its input. To test the utility of the differentially private histogram generated from our algorithm, we used it to train an ID3 classifier called *DPCube histogram ID3*. As a comparison, we implemented an interactive differentially private ID3 classifier, *private interactive ID3*, introduced by Friedman et al. [33].

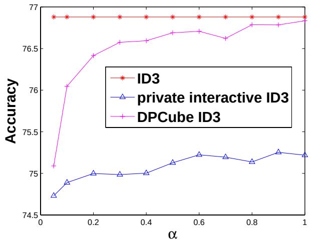


Fig. 17. Classification accuracy vs. privacy budget α

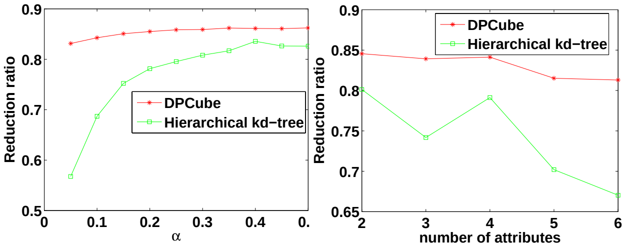
Figure 17 shows the classification accuracy of the different classifiers with respect to varying privacy budget α. The original *ID3* classifier provides a baseline accuracy at 76.9%. The *DPCube ID3* achieves slightly worse but comparable accuracy than the baseline due to the noise. While both the DPCube ID3 and the private interactive ID3 achieve better accuracy with increasing privacy budget as expected, our DPCube ID3 outperforms the private interactive ID3 due to its efficient use of the privacy budget.

Blocking for Record linkage. We also evaluated the utility of the released histogram for record linkage and compared our method against the *hierarchical kd-tree* scheme from [19]. The attributes we considered for this experiment are: age, education, wage, marital status, race and sex. As the histogram is used for the blocking step and all pairs of records in matching blocks will be further linked using an SMC protocol, our main goal is to reduce the total number of pairs of records in matching blocks in order to reduce the SMC cost. We use the reduction ratio used in [19] as our evaluation metric. It is defined as follows:

$$reduction\_ratio = 1 - \frac{\sum_{i=1}^{k} n_i * m_i}{n * m} \qquad (10)$$

where n_i (m_i) corresponds to the number of records in dataset 1 (resp. 2) that fall into the ith block, and k is the total number of blocks.

We compared both methods by running experiments with varying privacy budget (α) values (using the first 2 attributes of each record) and with varying numbers of attributes (with α fixed to 0.1). Figure 18(a) shows the reduction ratio with varying privacy budget. Both methods exhibit an increasing trend in reduction ratio as the privacy budget grows but our 2-phase v-optimal histogram consistently outperforms the hierarchical kd-tree approach and maintains a steady reduction ratio around 85%. Figure 18(b) shows the reduction ratio with varying number of attributes (dimensions). As the number of attributes increases, both methods show a drop in the reduction ration due to the sparsification of data points, thus increasing the relative error for each cell/partition. However, our DPCube approach exhibits desirable robustness when the dimensionality increases compared to the hierarchical kd-tree approach.



(a) vs. privacy budget α (b) vs. number of attributes

Fig. 18. Reduction ratio of blocking for record linkage

7 Conclusions and Future works

We have presented a two-phase multidimensional partitioning algorithm with estimation algorithms for differentially private histogram release. We formally analyzed the utility of the released histograms and quantified the errors for answering linear counting queries. We showed that the released v-optimal histogram combined with a simple query estimation scheme achieves bounded query error and superior utility than existing approaches for "smoothly" distributed data. The experimental results on using the released histogram for random linear counting queries and additional applications including classification and blocking for record linkage showed the benefit of our approach. As future work, we plan to develop algorithms that are both data- and workload-aware to boost the accuracy for specific workloads and investigate the problem of releasing histograms for temporally changing data.

References

[1] Y. Xiao, L. Xiong, and C. Yuan, "Differentially Private Data Release through Multidimensional Partitioning," in *7th VLDB Workshop on Secure Date Management, Lecture Notes in Computer Science,*, vol. 6358, 2010.

[2] Y. Xiao, J. Gardner, and L. Xiong, "Dpcube: Releasing differentially private data cubes for health information (demo paper)," in *ICDE*, 2012.

[3] C. Dwork, "Differential privacy: a survey of results," in *5th international conference on Theory and applications of models of computation, TAMC*, 2008.

[4] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: A survey on recent developments," *ACM Computing Surveys*, vol. 42, no. 4, 2010.

[5] C. Dwork, "A firm foundation for private data analysis," *Commun. ACM*, vol. 54, January 2011.

[6] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *3rd Theory of Cryptography Conference*, 2006.

[7] C. Dwork, "Differential privacy," *Automata, Languages and Programming, Pt 2*, vol. 4052, 2006.

[8] D. Kifer and A. Machanavajjhala, "No free lunch in data privacy," in *SIGMOD*, 2011.

[9] A. Blum, K. Ligett, and A. Roth, "A learning theory approach to non-interactive database privacy," in *STOC*, 2008.

[10] C. Dwork, M. Naor, O. Reingold, G. N. Rothblum, and S. Vadhan, "On the complexity of differentially private data release: efficient algorithms and hardness results," in *STOC*, 2009.

[11] McSherry, "Privacy integrated queries: an extensible platform for privacy-preserving data analysis," in *SIGMOD*, 2009.

[12] F. McSherry and K. Talwar, "Mechanism design via differential privacy," in *FOCS*, 2007.

[13] A. Roth and T. Roughgarden, "Interactive privacy via the median mechanism," in *STOC*, 2010.

[14] M. Hardt and G. Rothblum, "A multiplicative weights mechanism for interactive privacy-preserving data analysis," in *FOCS*, 2010.

[15] A. Korolova, K. Kenthapadi, N. Mishra, and A. Ntoulas, "Releasing search queries and clicks privately," in *WWW*, 2009.

[16] M. Götz, A. Machanavajjhala, G. Wang, X. Xiao, and J. Gehrke, "Privacy in search logs," *CoRR*, vol. abs/0904.0682, 2009.

[17] R. Chen, N. Mohammed, B. C. M. Fung, B. C. Desai, and L. Xiong, "Publishing set-valued data via differential privacy," in *VLDB*, 2011.

[18] F. McSherry and I. Mironov, "Differentially private recommender systems: building privacy into the net," in *KDD*, 2009.

[19] A. Inan, M. Kantarcioglu, G. Ghinita, and E. Bertino, "Private record matching using differential privacy," in *EDBT*, 2010.

[20] N. Mohammed, R. Chen, B. C. Fung, and P. S. Yu, "Differentially private data release for data mining," in *KDD*, 2011.

[21] B. Ding, M. Winslett, J. Han, and Z. Li, "Differentially private data cubes: optimizing noise sources and consistency," in *SIGMOD*, 2011.

[22] X. Xiao, G. Wang, and J. Gehrke, "Differential privacy via wavelet transforms," in *ICDE*, 2010.

[23] M. Hayy, V. Rastogiz, G. Miklauy, and D. Suciu, "Boosting the accuracy of differentially-private histograms through consistency," *VLDB*, 2010.

[24] C. Li, M. Hay, V. Rastogi, G. Miklau, and A. McGregor, "Optimizing linear counting queries under differential privacy," in *PODS*, 2010.

[25] J. Xu, Z. Zhang, X. Xiao, Y. Yang, and G. Yu, "Differentially private histogram publication," in *ICDE*, 2012.

[26] K. M and H. J, W, *Data mining: concepts and techniques, Second Edition*. MorganKaufman, 2006.

[27] U. Kchler and S. Tappe, "On the shapes of bilateral gamma densities," *Statistics & Probability Letters*, vol. 78, no. 15, 2008.

[28] Y. Ioannidis, "The history of histograms (abridged)," in *Proc. of VLDB Conference*, 2003.

[29] S. Muthukrishnan, V. Poosala, and T. Suel, "On rectangular partitionings in two dimensions: Algorithms, complexity, and applications," in *ICDT*, 1999, pp. 236–256.

[30] V. Poosala, P. J. Haas, Y. E. Ioannidis, and E. J. Shekita, "Improved histograms for selectivity estimation of range predicates," *SIGMOD Rec.*, vol. 25, June 1996.

[31] M. Hardt and G. N. Rothblum, "A multiplicative weights mechanism for privacy-preserving data analysis," in *FOCS*, 2010.

[32] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright, "A practical differentially private random decision tree classifier," in *ICDM Workshops*, 2009.

[33] A. Friedman and A. Schuster, "Data mining with differential privacy," in *SIGKDD*, 2010.

[34] A. Elmagarmid, P. Ipeirotis, and V. Verykios, "Duplicate record detection: A survey," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 19, no. 1, pp. 1 –16, jan. 2007.

[35] C. George and R. L.Berger, *Statistical Inference*, 2001.

[36] H. Anton and C. Rorres, *Introduction to Probability Models, Tenth Edition*. John Wiley and Sons, Inc, 2005.

[37] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[38] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, pp. 10–18, November 2009.