Cache Me If You Can: Accuracy-Aware Inference Engine for Differentially Private Data Exploration

Miti Mazmudar
miti.mazmudar@uwaterloo.ca
University of Waterloo

Matthew Rafuse
matthew.rafuse@uwaterloo.ca
University of Waterloo

Thomas Humphries
t3humphries@uwaterloo.ca
University of Waterloo

Xi He
xi.he@uwaterloo.ca
University of Waterloo

ABSTRACT
Differential privacy (DP) allows data analysts to query databases that contain users’ sensitive information while providing a provable privacy guarantee to the users. Recent interactive DP tools such as APEx provide accuracy guarantees over the query responses, but may fail to support a large number of queries with a limited total privacy budget, as they process new incoming queries independently from the historical queries. This paper proposes a DP accuracy-aware inference engine, that maintains a cache of past responses, and utilizes these responses to save privacy budget for future queries. To make use of the cache, we design a modified matrix mechanism and show that it saves privacy budget in our preliminary experiments.

1 INTRODUCTION
Organizations often collect large datasets that contain users’ sensitive data and permit data analysts to query these datasets for aggregate statistics. However, responses to these queries may be used by a curious data analyst to learn an individual’s record. The Differential Privacy (DP) formulation [4, 5] allows organizations to provide a guarantee to their users that the presence or absence of their record in the dataset will only change the distribution of the query response by a small factor, measured in terms of $\epsilon$. Multiple query responses can be released and post-processed without breaking this privacy guarantee. Statistical organizations such as the US Census Bureau [13] and companies like Google and Microsoft have started to adopt differential privacy in their applications.

Existing deployments of differential privacy [1, 2, 10, 12–14] mainly consider a non-interactive setting, where the analyst provides queries in advance. Systems [6, 9, 15, 17] that support interactive settings for differentially private data exploration have been difficult to deploy as data analysts have often been left with choosing an appropriate privacy budget $\epsilon$ per query. Secondly, data analysts desire a level of query accuracy to be maintained, a constraint that traditional differentially private systems do not provide on their outputs. Ge et al.’s APEx system [7] eliminates these two drawbacks; data analysts need only specify native queries and accuracy bounds in the form of an error rate $\alpha$ and a probability of failure $\beta$ without specifying the privacy budget per query.

However, most database systems that support interactive DP [7, 9, 15], may fail to support a large number of queries under a limited total privacy budget. As each query consumes some privacy budget, without proper planning of these queries, the privacy budget will be depleted quickly. Interactive systems such as APEx currently process each set of queries that have the same accuracy requirements, or a workload, independently, without taking into account the differentially private responses to previous workloads. We observe that in data exploration, workloads that arrive at different times can be highly correlated. For example, an analyst may ask a query that spans the entire domain of a given attribute and then ask queries that focus on successively smaller parts of the domain (see Example 1 in Figure 1). Alternatively, their queries may explore the domain of an attribute sequentially (Example 2). Moreover, after an analyst sees a noisy response for a given query, they may request a more accurate response for the same query (Example 3). Our key insight lies in the observation that processing new workloads while using previously cached responses saves privacy budget and thus we can answer more queries, under a given total privacy budget.

In this work, we design CacheDP, an accuracy aware inference engine for differentially private data exploration. This engine consists of a cache algorithm that interacts with the private data and data analysts. We integrate this algorithm to an existing accuracy-aware DP tool, namely APEx, such that the accuracy requirements of the analyst are met and simultaneously, prior responses are used to save the privacy budget on a given workload. Peng et al.’s Pioneer [16] uses historical query answers in order to obtain accurate responses to upcoming queries over the same or related predicates. However, our approach has three distinguishing features. First, we can optimize the privacy budget over multiple predicates, unlike Pioneer which handles a scalar response one by one. To achieve this, we adapt the matrix mechanism [12] from a single workload setting to capture both queries in the cache and
the unanswered queries. In this way, we can spend a much smaller privacy budget than APEx to achieve the accuracy requirement. Second, if the same query is asked under a higher accuracy requirement, we add correlated noise to improve the accuracy [11]. Third, we proactively fetch certain query responses at a marginal cost so that if a user asks for responses to a disjoint query with similar accuracy requirements later, we do not need to spend any additional privacy budget.

Our main contributions are:
- We formulate the problem of interactively using historical query answers to minimize the privacy budget spent.
- We propose a cache-based inference engine and algorithm that satisfies accuracy requirements interactively.
- We integrate our engine into an existing accuracy-aware differential privacy tool, namely APEx.
- We provide a preliminary evaluation of our algorithm across various use-cases to illustrate the savings in the privacy budget compared to APEx.

2 BACKGROUND

We assume a single-table relational schema $R(A_1, \ldots, A_d)$. The domain of an attribute $A_i$ is represented as $\text{dom}(A_i)$. The full domain of $R$ is $\text{dom}(R) = \text{dom}(A_1) \times \cdots \times \text{dom}(A_d)$. A database instance $D$ of relation $R$ is a multiset whose elements are tuples in $\text{dom}(R)$. A predicate counting query takes a predicate $\phi : \text{dom}(R) \rightarrow \{0, 1\}$ and returns the number of tuples in $D$ that satisfy $\phi$, i.e., $\phi(D) = \sum_{t \in D} \phi(t)$. Given a database instance $D$ and $\Phi = \{\phi_1, \ldots, \phi_n\}$, the set of unit length predicates, we can represent $D$ with a data vector $x$, where $x[i] = \phi_i(D)$. Given $x$, we can represent all linear counting queries as a length-$n$ vector $w = [w_1, \ldots, w_n]$ with $w_i \in \{0, 1\}$. We say that two queries $w$ and $v$ are disjoint, if $w \cdot v = 0$. The answer to a linear counting query is thus $w \cdot x$. Hence, we can represent a workload of $\ell$ linear counting queries as a $\ell \times n$ matrix $W$ over the domain of $x$ and the response to this workload is $Wx$. We follow the standard definition of differential privacy (DP).

**Definition 2.1 ($\epsilon$-Differential Privacy [4]).** A randomized mechanism $M : \mathcal{D} \rightarrow \mathcal{O}$ satisfies $\epsilon$-differential privacy if

$$Pr[M(D) \in O] \leq e^\epsilon Pr[M(D') \in O]$$

(1)

for any set of outputs $O \subseteq \mathcal{O}$, and any pair of neighboring databases $D, D'$ such that $|D \setminus D' \cup D' \setminus D| = 1$.

**Definition 2.2.** (Accuracy Bound [7]). Given a DP mechanism $M$ that answers a workload counting query $W$ over $x$, we say that $M$ is $(\alpha, \beta)$-accurate if for any output of $M$, $y$, we have

$$Pr[|Wx - y|_{\infty} \geq \alpha] \leq \beta$$

One way to obtain noisy workload response with $(\alpha, \beta)$-accuracy is to use the Laplace mechanism with a specific choice of epsilon.

**Definition 2.3** (Laplace Mechanism). For a given $\ell \times n$ workload matrix $W$ and accuracy requirement $(\alpha, \beta)$ the Laplace Mechanism $L$ is defined as

$$L(W, x, \alpha, \beta) = Wx + \text{Lap}(|W||1/\epsilon)^{\ell}$$

where $\epsilon = \frac{\|W\| \ln(1/(1-(1-\beta)^{1/\ell}))}{\alpha}$.

The Laplace Mechanism as defined above satisfies $\epsilon$-DP whilst achieving $(\alpha, \beta)$-accuracy as shown in A.1 of the APEx Paper [7].

Another way to obtain $(\alpha, \beta)$-accurate responses is to use the Matrix Mechanism [12] using a Monte Carlo (MC) simulation to empirically bound the accuracy, as in APEx [7].

**Definition 2.4** (Matrix Mechanism [12]). Given an $\ell \times n$ workload matrix $W$ and a strategy matrix $A$, such that $W$ is some linear combination of $A$, the matrix mechanism is defined as

$$M_{K,A}(W, x) = WA^T K(A, x)$$

where $K(A, x) = Ax + \text{Lap}(|A||1/\epsilon)^{|A|}$, and $|A|$ denotes the number of rows in matrix $A$.

The flexibility of the matrix mechanism comes from different choices of the strategy matrix $A$. The Hierarchical Tree inferencer introduced by Hay et al. [8] can be represented as a strategy matrix $H$, as Li et al. point out. McKenna et al. [14] provide a technique to optimize the choice of strategy matrix.

3 APPROACH

Given a database instance $D$ and a total privacy budget set by data owner $B$, our system receives a sequence of workloads with their accuracy requirements $\ldots, (Q, \alpha, \beta), \ldots$. The workload and its accuracy requirement will be translated to a differentially private algorithm that meets the accuracy requirement while minimizing the privacy budget spent. Our system, CacheDP, is different from prior work APEx [7] in that we interface CacheDP with a cache in order to exploit past responses. In this section, we describe how our system caches responses and uses them to answer each workload accurately, while saving the privacy budget for future workloads.

3.1 Overview

Our cache $C$ is indexed by the query predicate in the strategy matrix. For each query predicate, we store a noisy response $\tilde{y}$ that was drawn from the Laplace distribution, as an intermediate step in the matrix mechanism $(K(A, x))$. We also store the noise parameter $\rho$, that was used to obtain $\tilde{y}$. We present our main algorithm in Algorithm 1.
We then obtain a strategy matrix \( W \) that optimally uses the cache to answer the predicates in \( W' \) (line 9). Under our accuracy definition 2.2, when the number of predicates increases, the cost of obtaining noisy responses to these predicates increases slightly. Thus, we ensure that the privacy budget for the proactive \( W' \) is within a multiplicative factor \( T \) of the original budget estimate \( \epsilon \) (lines 10–12).

We check that the privacy budget spent is less than the remaining privacy budget irrespective of whether we proactively cache query responses or not (line 13). We thus provide \( \mathcal{B} \)-differential privacy, just as APEx does. We finally execute our modified matrix mechanism, with the chosen strategy matrix \( A \) under the privacy budget \( \epsilon \), which was estimated to provide the \((\alpha, \beta)\) guarantee. We estimate the response \( \tilde{y}_i \), which answers \( W \), and the actual privacy budget spent \( \epsilon_Q \) (line 14) and update the cumulative privacy budget (line 15). We now proceed to discuss the modified matrix mechanism function, which is presented in Algorithm 2. We remark that the cost estimation function for the matrix mechanism is similar to the original function in APEx. Only the implementation of the Monte Carlo simulation, encapsulated in the call to the \( \text{estimateBeta} \) function, is modified to be cache-aware and is also discussed below.

### 3.2 Cached Matrix Mechanism

Unlike the matrix mechanism in prior work [7, 12], the new mechanism and its cost estimation function, take in the cache \( C \) as input. We partition the strategy matrix \( A \) into two parts based on whether the predicate exists in the cache. That is, if the predicate belongs to the cache, it is in the free matrix \( F \), otherwise it is in the matrix to be fetched from the database \( P \) (line 2). If noisy responses for all of the predicates in the strategy matrix are present in the cache (line 3), then our modified mechanism improves the accuracy of these noisy responses to satisfy the analyst’s accuracy bounds. Otherwise, we use as many cached responses as possible to answer the query and use a lower privacy budget to noise all non-cached predicates, \( P \). For each case, we describe the relevant parts of the \( \text{modifiedMatrixMechanism} \) and \( \text{estimateBeta} \) functions below.

In the first case, we estimate the least privacy cost \( \epsilon_{\text{est}} \) required to answer these queries without the cache. This is achieved by treating all the queries found in the cache \( F \) as new queries \( P \) (lines 21–23 and second case of line 25) in the function \( \text{estimateBeta} \). That is, all cached responses should meet the target epsilon \( \epsilon_{\text{est}} \) in order to be sufficiently accurate. To improve the accuracy of cached responses while minimizing the privacy budget spent, we apply a similar technique as the \( \text{relaxPrivacy} \) function in APEx, which was based on Koufogiannis et al.’s work [11]. The new noisy response \( \tilde{y}_p \) is correlated with the old response in the cache and the new privacy budget spent here, \( \epsilon \), is less than \( \epsilon_{\text{est}} \). However, in contrast with APEx, each of these cached responses may be obtained across different workloads, and hence the privacy budget for each response may be different. For such cases, we leave a careful analysis of the new privacy loss to the full paper. We also update the cache with these more accurate responses (lines 5–6).

On the other hand, if the noisy response for at least one of the predicates in the strategy matrix is absent from the cache \( (F \subset A) \), then our mechanism fetches the noisy responses \( \tilde{y}_f \) for the cached
We integrated our cached-based approach CacheDP into the workload sequence consists of all nodes in the tree as this query. We use the matrix mechanism with the hierarchical strategy matrix \( H_k \), and thereby exploit the cache for the response of \( Q_1 \) by building a \( k \)-ary tree of height 1.

Within our approach, only a very small privacy budget needs to be spent in proactively fetching disjoint predicate responses for the second workload. In return, we get significant savings in the future, which will occur for even a single query over one of the disjoint predicates. These observations favor incorporating a proactive caching method. We note that benefits of the proactive approach are unique to the interactive setting as in the non-interactive one a more optimal workload could have been derived.

5 FUTURE WORK

Our prototype system can be extended and developed in several ways. First, we will integrate a cache within McKenna et al.’s [14] technique of identifying an optimal strategy matrix for a given workload query. Second, we will develop a detailed proof that CacheDP preserves \( \varepsilon \)-differential privacy, with particular attention to the relaxPrivacy algorithm. Third, we will also test our approach for queries over multiple dimensions, and optimize it for such queries. We plan to implement an efficient cache structure and test our algorithm under multiple different query workload sequences with different datasets. Although we simply use constraints over the noisy responses to improve the accuracy of our output, our work opens up the question of consistency constraints for the interactive setting. For instance, an analyst may prefer knowing more accurate versions of noisy answers to historical queries, as new and related queries are answered. Our algorithm could potentially be modified to support this setting.
REFERENCES


