Dual Query: Practical Private Query Release for High Dimensional Data

Marco Gaboardi Emilio Jesús Gallego Arias
Justin Hsu Aaron Roth Zhiwei Steven Wu

February 10, 2014

Abstract

We present a practical, differentially private algorithm for answering a large number of queries on high dimensional datasets. Like all algorithms for this task, ours necessarily has worst-case complexity exponential in the dimension of the data. However, our algorithm packages the computationally hard step into a concisely defined integer program, which can be solved non-privately using standard solvers. We prove accuracy and privacy theorems for our algorithm, and then demonstrate experimentally that our algorithm performs well in practice. For example, our algorithm can efficiently and accurately answer millions of queries on the Netflix dataset, which has over 17,000 attributes; this is an improvement on the state of the art by multiple orders of magnitude.

1 Introduction

Privacy is becoming a paramount concern for machine learning and data analysis tasks, which often operate on personal data. The Netflix challenge exemplifies the tension between machine learning and data privacy. Netflix released an anonymized dataset of user movie ratings for teams competing to develop an improved recommendation mechanism. The competition was a great success (the winning team improved on the existing recommendation system by more than 10%), but the ad hoc anonymization was not as successful: Narayanan and Shmatikov [27] were later able to re-identify individuals in the dataset. This eventually led to a lawsuit and the cancellation of subsequent competitions.

Differentially private query release is an attempt to solve this problem. Differential privacy is a strong formal privacy guarantee (that, among other things, provably prevents re-identification attacks), and the problem of *query release* is to release accurate answers to a set of statistical queries. As observed early on by Blum et al. [6], performing private query release is sufficient to simulate any learning algorithm in the "statistical query model" of Kearns [22].

Since then, the query release problem has been extensively studied in the differential privacy literature. While simple perturbation can be used to privately answer a small number of queries [15], more sophisticated approaches can accurately answer nearly exponentially many queries in the size of the private database [5, 13, 14, 29, 18, 17, 19]. A natural approach, employed by many of these algorithms, is to answer queries by generating *synthetic data*: a fresh, safe version of the dataset approximates the real dataset on every statistical query of interest.

Unfortunately, even the most efficient approaches have a per-query running time linear in the size of the *data universe*, which is exponential in the dimension of the data [18]. Moreover, this running time is necessary in the worst case [32], especially if the algorithm produces synthetic data [33].

This exponential runtime has hampered practical evaluation of query release algorithms. One notable exception is due to Hardt et al. [19], who perform a thorough experimental evaluation of one such algorithm, which they called MWEM. They find that MWEM has quite good accuracy in practice and scales to higher dimensional data than suggested by a theoretical (worst-case) analysis. Nevertheless, running time remains a problem, and the approach does not seem to scale to high dimensional data (with more than 30 or so attributes for general queries, and more when the queries satisfy special structure[1]). The critical bottleneck is the size of the state maintained by the algorithm: MWEM, like many query release algorithms, needs to manipulate an object that has size linear in the size of the data universe (i.e., exponential in the dimension). This quickly becomes impractical as the record space grows more complex.

We present DualQuery, an alternative algorithm which is *dual* to MWEM in a sense that we will make precise. Rather than manipulate an object of exponential size, DualQuery needs to solve a concisely represented (but NP-hard) optimization problem. Critically, the optimization step does not require a solution that is either private or exact: it can be solved by existing, off-the-shelf optimization packages quickly in practice. Except for this step, all parts of our algorithm are extremely efficient. As a result, DualQuery requires (worst-case) space and (in practice) time only linear in the number of *queries* of interest, which is often significantly smaller than the number of possible records. Like existing algorithms for query release, DualQuery has a provable accuracy guarantee and satisfies the strong differential privacy guarantee.

We evaluate DualQuery on a variety of datasets by releasing *3-way marginals* (also known as *conjunctions* or *contingency tables*), demonstrating that it solves the query release problem accurately and efficiently even when the data includes hundreds of thousands of features. We know of no other algorithm to perform accurate, private query release for rich classes of queries on real data with more than even 100 features.

1.1 Related work

Differentially private learning has been studied since Blum et al. [6] showed how to convert learning algorithms in the SQ model of Kearns [22] into differentially private learning algorithms with similar accuracy guarantees. Since then, private machine learning has become a very active field with both foundational sample complexity results [21, 7, 4, 12] and numerous efficient algorithms for particular learning problems [8, 10, 30, 23, 11, 31].

In parallel, there has been a significant amount of work on privately releasing synthetic data based on a true dataset while preserving the answers to large numbers of statistical queries [5, 13, 29, 14, 18, 17]. These results are extremely strong in an information theoretic sense: they ensure the consistency of the synthetic data with respect to an exponentially large family of statistics. But, all of these algorithms (including the notable multiplicative weights algorithm of Hardt and Rothblum [18], which achieves the theoretically optimal accuracy and runtime) have running time exponential in the dimension of the data. With standard cryptographic assumptions, this is necessary in the worst case for mechanisms that answer many arbitrary statistical queries [32].

Nevertheless, there have been some experimental evaluations of these approaches on real datasets. Most related to our work is the evaluation of the MWEM mechanism by Hardt et al. [19], which is based on the private multiplicative weights mechanism [18]. This algorithm is inefficient (it manipulates a probability distribution over a set exponentially large in the dimension of the data space) but with some heuristic optimizations, Hardt et al. [19] were able to implement the multiplicative weights algorithm on several real datasets with up to 77 attributes (and even more when the queries are restricted to take positive values only

[1] Hardt et al. [19] are able to scale up to 1000 features on synthetic data when the features are partitioned into a number of small buckets, and the queries are chosen to never depend on features in more than one bucket.

on a small number of disjoint groups of features). However, it seems difficult to scale this approach to higher dimensional data.

Another family of query release algorithms are based on the Matrix Mechanism [25, 24]. The runtime guarantees of the matrix mechanism are similar to the approaches based on multiplicative weights—the algorithm manipulates a "matrix" of queries with dimension exponential in the number of features. Yaroslavtsev et al. [34] evaluate an approach based on this family of algorithms on low dimensional datasets, but scaling to high dimensional data also seems challenging.

Our algorithm is inspired by the view of the synthetic data generation problem as a zero-sum game, first proposed by Hsu et al. [20]. In this interpretation, Hardt et al. [19] solves the game by having a *data player* use a no-regret learning algorithm, while the *query player* repeatedly best responds by optimizing over queries. In contrast, our algorithm swaps the roles of the two players: the query player now uses the no-regret learning algorithm, whereas the data player now finds best responses by solving an optimization problem. This is reminiscent of "Boosting for queries," proposed by Dwork et al. [14]; the main difference is that our optimization problem is over single records rather than sets of records. As a result, our optimization can be handled non-privately.

2 Differential privacy background

Differential privacy has become a standard algorithmic notion for protecting the privacy of individual records in a statistical database. It formalizes the requirement that the addition or removal of a data record does not change the probability of any outcome of the mechanism by much.

We consider databases which are multisets of elements from an abstract domain $\mathcal{X}$, representing the set of all possible data records. We often consider elements in $\mathcal{X}$ as bit-strings of length d. Two databases $D, D' \subset \mathcal{X}$ are *neighboring* if they differ in a single data element: i.e., their symmetric difference $\|D \triangle D'\|$ is at most 1.

Definition 2.1 (Dwork et al. [15])**.** A mechanism $M \colon \mathcal{X}^n \to R$ satisfies (ε, δ)-differential privacy if for every $S \subseteq R$ and for all neighboring databases $D, D' \in \mathcal{X}^n$, the following holds:

$$\Pr[M(D) \in S] \leq e^{\varepsilon} \Pr[M(D') \in S] + \delta$$

If $\delta = 0$ we say M satisfies ε-differential privacy.

Definition 2.2. The *(global) sensitivity* of a query Q is its maximum difference when evaluated on two neighboring databases:

$$GS_f = \max_{D, D' \in \mathcal{X}^n : \|D \triangle D'\| = 1} |f(D) - f(D')|.$$

In this paper, we consider the private release of information for the classes of linear queries, which have sensitivity $1/n$.

Definition 2.3. For any predicate $\varphi \colon \mathcal{X} \to \{0, 1\}$, the *linear query* $Q_\varphi \colon \mathcal{X}^n \to [0, 1]$ is defined by

$$Q_\varphi(D) = \frac{\sum_{x \in D} \varphi(x)}{|D|}.$$

The following composition theorem is a fundamental tool for private algorithm analysis: we can bound the privacy cost of an algorithm as a function of the privacy costs of its subcomponents.

Lemma 2.4 (Dwork et al. [14]). *Let $M_1, \ldots, M_k$ be such that each M_i is $(\varepsilon_i, 0)$-private with $\varepsilon_i \leq \varepsilon'$. Then $M(D) = (M_1(D), \ldots, M_k(D))$ is $(\varepsilon, 0)$-private for $\varepsilon = \sum_{i=1}^k \varepsilon_i$, and (ε, δ)-private for*

$$\varepsilon = \sqrt{2 \log(1/\delta) k} \varepsilon' + k\varepsilon'(e^{\varepsilon'} - 1)$$

for any $\delta \in (0, 1)$.

3 The query release game

The analysis of our algorithm relies on the interpretation of query release as a two player, zero-sum game [20]. In the present section, we review this idea and related tools.

3.1 Game definition

Suppose we want to answer all queries in $\mathcal{Q}$. For each query $q \in \mathcal{Q}$, we can form the *negated query* $\overline{q}$, which takes values $\overline{q}(D) = 1 - q(D)$ for every database D. Equivalently, for a linear query defined by a predicate φ, the negated query is defined by the negation $\neg\varphi$ of the predicate. For the remainder, we will assume that $\mathcal{Q}$ is closed under negation; if not, we may add negated copies of each query to $\mathcal{Q}$.

Let there be two players, whom we call the *data* player and *query* player. The data player has action set equal to the data universe $\mathcal{X}$, while the query player has action set equal to the query class $\mathcal{Q}$. Given a play $x \in \mathcal{X}$ and $q \in \mathcal{Q}$, we let the payoff be

$$A(x, q) := q(D) - q(x), \tag{1}$$

where D is the true database. As a zero sum game, the data player will try to minimize the payoff, while the query player will try to maximize the payoff.

3.2 Equilibrium of the game

Let $\Delta(\mathcal{X})$ and $\Delta(\mathcal{Q})$ be the set of probability distributions over $\mathcal{X}$ and $\mathcal{Q}$. We consider how well each player can do if they randomize over their actions, i.e., if they play from a probability distribution over their actions. By von Neumann's minimax theorem,

$$\min_{u \in \Delta(\mathcal{X})} \max_{w \in \Delta(\mathcal{Q})} A(u, w) = \max_{w \in \Delta(\mathcal{Q})} \min_{u \in \Delta(\mathcal{Q})} A(u, w),$$

for any two player zero-sum game, where

$$A(u, w) := \mathbb{E}_{x \sim u, q \sim w} A(x, q)$$

is the expected payoff. The common value is called the *value of the game*, which we denote by v_A. Intuitively, von Neumann's theorem states that there is no advantage in a player going first: the minimizing player can always force payoff at most v_A, while the maximizing player can always force payoff at least v_A.

This suggests that each player can play an optimal strategy, assuming best play from the opponent—this is the notion of equilibrium strategies, which we now define. We will soon interpret these strategies as solutions to the query release problem.

Definition 3.1. Let $\alpha > 0$. Let A be the payoffs for a two player, zero-sum game with action sets $\mathcal{X}, \mathcal{Q}$. Then, a pair of strategies $u^* \in \Delta(\mathcal{X})$ and $w^* \in \Delta(\mathcal{Q})$ form an *α-approximate mixed Nash equilibrium* if

$$A(u^*, w) \leq v_A + \alpha \qquad \text{and} \qquad A(u, w^*) \geq v_A - \alpha$$

for every strategy $u \in \Delta(\mathcal{X}), w \in \Delta(\mathcal{Q})$.

If the true database D is normalized to be a distribution $\widehat{D}$ in $\Delta(\mathcal{X})$, then $\widehat{D}$ always has zero payoff:

$$A(\widehat{D}, w) = \mathbb{E}_{x \sim \widehat{D}, q \sim w}[q(x) - q(D)] = 0.$$

Hence, the value of the game v_A is at most 0. Also, for any data strategy u, the payoff of query q is the negated payoff of the negated query $\overline{q}$:

$$A(u, \overline{q}) = \mathbb{E}_{x \sim u}[\overline{q}(x) - \overline{q}(D)] = \mathbb{E}_{x \sim u}[q(D) - q(x)],$$

which is $A(u, \overline{q})$. Thus, any query strategy that places equal weight on q and $\overline{q}$ has expected payoff zero, so v_A is at least 0. Hence, $v_A = 0$.

Now, let (u^*, w^*) be an α-approximate equilibrium. Suppose that the data player plays u^*, while the query player always plays query q. By the equilibrium guarantee, we then have $A(u^*, q) \leq \alpha$, but the expected payoff on the left is simply $q(D) - q(u^*)$. Likewise, if the query player plays the negated query $\overline{q}$, then

$$-q(D) + q(u^*) = A(u^*, \overline{q}) \leq \alpha,$$

so $q(D) - q(u^*) \geq -\alpha$. Hence for every query $q \in \mathcal{Q}$, we know $|q(u^*) - q(D)| \leq \alpha$. This is precisely what we need for query release: we just need to privately calculate an approximate equilibrium.

3.3 Solving the game

To construct the approximate equilibrium, we will use the multiplicative weights update algorithm (MW).[2] This algorithm maintains a distribution over actions (initially uniform) over a series of steps. At each step, the MW algorithm receives a (possibly adversarial) loss for each action. Then, MW reweights the distribution to favor actions with less loss. The algorithm is presented in Algorithm 1.

Algorithm 1 The Multiplicative Weights Algorithm

```
Let $\eta > 0$ be given, let $\mathcal{A}$ be the action space
Initialize $\widetilde{A}^1$ uniform distribution on $\mathcal{A}$
For $t = 1, 2, \ldots, T$:
  Receive loss vector $\ell^t$
  **For each** $a \in \mathcal{A}$**:**
    Update $A_a^{t+1} = e^{-\eta \ell_a^t} \widetilde{A}_a^t$ for every $a \in \mathcal{A}$
  Normalize $\widetilde{A}^{t+1} = \frac{A^{t+1}}{\sum_i A_i^{t+1}}$
```

For our purposes, the most important application of MW is to solving zero-sum games. Freund and Schapire [16] showed that if one player maintains a distribution over actions using MW, while the other player selects a *best-response* action versus the current MW distribution (i.e., an action that maximizes his expected payoff), the average MW distribution and empirical best-response distributions will converge to an approximate equilibrium rapidly.

[2] The MW algorithm has wide applications; it has been rediscovered in various guises several times. More details can be found in the comprehensive survey by Arora et al. [1].

Theorem 3.2. *Let $\alpha > 0$, and let $A(i,j) \in [-1,1]^{m \times n}$ be the payoff matrix for a zero-sum game. Suppose the first player uses multiplicative weights over their actions to play distributions $p^1, \dots, p^T$, while the second player plays $(\alpha/2)$-approximate best responses $x^1, \dots, x^T$, i.e.,*

$$A(p^t, x^t) \geq \max_x A(p^t, x) - \alpha/2.$$

Setting $T = 16 \log n/\alpha^2$ and $\eta = \alpha/4$ in the MW algorithm, the empirical distributions

$$\frac{1}{T}\sum_{i=1}^{T} p^i \quad \text{and} \quad \frac{1}{T}\sum_{i=1}^{T} x^i$$

form an α-approximate mixed Nash equilibrium.

Interpreted this way, the private multiplicative weights algorithm of Hardt and Rothblum [18] (and the MWEM algorithm of Hardt et al. [19]) uses MW for the data player, while the query player plays best responses. For privacy, their algorithm selects the query best-responses privately via the exponential mechanism of McSherry and Talwar [26]. Our algorithm plays the same game, but with the roles reversed.

4 Dual query release

At a high level, our algorithm solves the query release game using an approach dual to the MWEM algorithm of Hardt et al. [19]. While MWEM uses a no-regret algorithm to maintain the data player's distribution, we will instead use a no-regret algorithm for the query player's distribution. Likewise, instead of finding a maximum payoff query at each round, our algorithm selects a minimum payoff record at each turn. The full algorithm can be found in Algorithm 2.

Our privacy argument differs slightly from the analysis for MWEM. There, the data distribution is public, and finding a query with high error requires access to the private data. Our algorithm instead maintains a distribution Q over queries which depends directly on the private data, so we cannot use Q directly. Instead, we argue that *queries sampled from* Q are privacy preserving. Then, we can use a non-private optimization method to find a minimal error record versus queries sampled from Q. We then trade off privacy (which degrades as we take more samples) with accuracy (which improves as we take more samples, since the distribution of sampled queries converges to Q).

Given known hardness results for the query release problem [32], our algorithm must have worst-case runtime polynomial in the universe size $|\mathcal{X}|$, so is not theoretically more efficient than prior approaches. In fact, even compared to prior work on query release (e.g., Hardt and Rothblum [18]), our algorithm has a worse accuracy guarantee.

However, our approach has important practical benefits, the most important being that that hard step can be handled with standard, non-private solvers. The rest of our algorithm runs in time linear in the number of queries, independent of the dimension of the data.

The iterative structure of our algorithm, combined with our use of constraint solvers, allows for several heuristics improvements. For instance, we may run for fewer iterations than predicted by theory. Or, if the optimization problem turns out to be hard (even in practice), we can stop the solver early at a suboptimal (but often still good) solution. These heuristic tweaks can improve accuracy beyond what is guaranteed by our accuracy theorem, while always maintaining a strong *provable* privacy guarantee.

Algorithm 2 DualQuery

```
Input: Database $D \in \mathbb{R}^{|\mathcal{X}|}$ (normalized) and linear queries $q_1, \ldots, q_k \in \{0,1\}^{|\mathcal{X}|}$.
Initialize: Let $\mathcal{Q} = \bigcup_{j=1}^{k} q_j \cup \overline{q_j}$, $Q^1$ uniform distribution on $\mathcal{Q}$,

    $$T = \frac{16 \log |\mathcal{Q}|}{\alpha^2}, \qquad \eta = \frac{\alpha}{4}, \qquad s = \frac{48 \log\left(\frac{2|\mathcal{X}|T}{\beta}\right)}{\alpha^2}.$$

For $t = 1, \ldots, T$:
    Sample $s$ queries $\{q_i\}$ from $\mathcal{Q}$ according to $Q^t$.
    Let $\overline{q} := \frac{1}{s}\sum_i q_i$.
    Find $x^t$ with $\langle \overline{q}, x^t \rangle \geq \max_x \langle \overline{q}, x \rangle - \alpha/4$.
    Update: For each $q \in \mathcal{Q}$:
        $Q_q^{t+1} := \exp(-\eta \langle q, x^t - D \rangle) \cdot Q_q^t$.
    Normalize $Q^{t+1}$.
Output synthetic database $\widehat{D} := \bigcup_{t=1}^{T} x^t$.
```

4.1 Privacy

The privacy proofs are largely routine, based on the composition theorems. Rather than fixing ε and solving for the other parameters, we present the privacy cost ε as function of parameters T, s, η. Later, we will tune these parameters for our experimental evaluation. We will use the privacy of the following mechanism (due to McSherry and Talwar [26]) as an ingredient in our privacy proof.

Definition 4.1 (McSherry and Talwar [26])**.** Given some arbitrary output range R, the *exponential mechanism* with score function S selects and outputs an element $r \in R$ with probability proportional to

$$\exp\left(\frac{\varepsilon S(D, r)}{2\Delta}\right),$$

where Δ is the *sensitivity* of S, defined as

$$\Delta = \max_{D, D' : |D \triangle D'| = 1, r \in R} |S(D, r) - S(D', r)|.$$

The exponential mechanism is ε-differentially private.

We first prove pure ε-differential privacy.

Theorem 4.2. DualQuery *is ε-differentially private for*

$$\varepsilon = \frac{\eta T (T-1) s}{n}.$$

Proof. We will argue that sampling from Q^t is equivalent to running the exponential mechanism with some quality score. At round t, let $\{x^i\}$ for $i \in [t-1]$ be the best responses for the previous rounds. Let $r(q, d)$ be defined by

$$r(q, D) = \sum_{i=1}^{t-1} (q(x^i) - q(D)),$$

where $q \in \mathcal{Q}$ is a query and D is the true database. This function is evidently $((t-1)/n)$-sensitive in D: changing D changes each $q(D)$ by at most $1/n$. Now, note that sampling from Q^t is simply sampling from the exponential mechanism, with quality score $r(q, D)$. Thus, the privacy cost of each sample in round t is $\varepsilon_t' = 2\eta(t-1)/n$ (Definition 4.1).

By the standard composition theorem (Lemma 2.4), the total privacy cost is

$$\varepsilon = \sum_{t=1}^{T} s\varepsilon_t' = \frac{2\eta s}{n} \cdot \sum_{t=1}^{T}(t-1) = \frac{\eta T(T-1)s}{n}.$$

□

We next show that DualQuery is (ε, δ)-differentially private, for a much smaller ε.

Theorem 4.3. *Let $0 < \delta < 1$. DualQuery is (ε, δ)-differentially private for*

$$\varepsilon = \frac{2\eta(T-1)}{n} \cdot \left[\sqrt{2s(T-1)\log(1/\delta)} + s(T-1)\left(\exp\left(\frac{2\eta(T-1)}{n}\right) - 1\right)\right].$$

Proof. Let ε be defined by the above equation. By the advanced composition theorem (Lemma 2.4), running a composition of k ε'-private mechanisms is (ε, δ)-private, for

$$\varepsilon = \sqrt{2k\log(1/\delta)}\varepsilon' + k\varepsilon'(\exp(\varepsilon') - 1).$$

Again, note that sampling from Q^t is simply sampling from the exponential mechanism, with a $(T-1)/n$-sensitive quality score. Thus, the privacy cost of each sample is $\varepsilon' = 2\eta(T-1)/n$ (Definition 4.1). We plug in $k = s(T-1)$ samples, as in the first round our samplings are 0-differentially private. □

4.2 Accuracy

The accuracy proof proceeds in two steps. First, we show that "average query" formed from the samples is close to the true weighted distribution Q^t. We will need a standard Chernoff bound.

Lemma 4.4 (Chernoff bound)**.** *Let $X_1, \ldots, X_N$ be IID random variables with mean μ, taking values in $[0,1]$. Let $\overline{X} = \frac{1}{N}\sum_i X_i$ be the empirical mean. Then,*

$$\Pr[|\overline{X} - \mu| > T] < 2\exp(-NT^2/3)$$

for any T.

Lemma 4.5. *Let $\beta \in (0,1)$, and let p be a distribution over queries. Suppose we draw*

$$s = \frac{48\log\left(\frac{2|\mathcal{X}|}{\beta}\right)}{\alpha^2}$$

samples $\{\widehat{q_i}\}$ from p, and let $\overline{q}$ be the aggregate query

$$\overline{q} = \frac{1}{s}\sum_{i=1}^{s}\widehat{q_i}.$$

Define the true weighted answer $Q(x)$ to be

$$Q(x) = \sum_{i=1}^{|\mathcal{Q}|} p_i q_i(x).$$

Then with probability at least $1-\beta$, we have $|\overline{q}(x) - Q(x)| < \alpha/4$ for every $x \in \mathcal{X}$.

Proof. For any fixed x, note that $\overline{q}(x)$ is the average of random variables $\widehat{q_1}(x), \dots, \widehat{q_s}(x)$. Also, note that $\mathbb{E}[\overline{q}(x)] = Q(x)$. Thus, by the Chernoff bound (Lemma 4.4) and our choice of s,

$$\Pr[|\overline{q}(x) - Q(x)| > \alpha/4] < 2\exp(-s\alpha^2/48) = \beta/|\mathcal{X}|.$$

By a union bound over $x \in \mathcal{X}$, this equation holds for all $x \in \mathcal{X}$ with probability at least $1-\beta$. □

Next, we show that we compute an approximate equilibrium of the query release game. In particular, the record best responses form a synthetic database that answer all queries in $\mathcal{Q}$ accurately. Note that our algorithm doesn't require an exact best response for the data player; an approximate best response will do.

Theorem 4.6. *With probability at least $1-\beta$,* DualQuery *finds a synthetic database that answers all queries in $\mathcal{Q}$ within additive error α.*

Proof. As discussed in Section 3, it suffices to show that the distribution of best responses $x^1, \dots, x^T$ forms is an α-approximate equilibrium strategy in the query release game. First, we set the number of samples s according to in Lemma 4.5 with failure probability β/T. By a union bound over T rounds, sampling is successful for every round with probability at least $1-\beta$; condition on this event.

Since we are finding an $\alpha/4$ approximate best response to the sampled aggregate query $\overline{q}$, which differs from the true distribution by at most $\alpha/4$ (by Lemma 4.5), each x^i is an $\alpha/4 + \alpha/4 = \alpha/2$ approximate best response to the true distribution Q^t. Since q takes values in $[0,1]$, the payoffs are all in $[-1,1]$. Hence, Theorem 3.2 applies; setting T and η accordingly gives the result. □

Remark 4.7. *The guarantee in Theorem 4.6 may seem a little unusual, since the convention in the literature is to treat ε, δ as inputs to the algorithm. We can do the same: from Theorem 4.3 and plugging in for T, η, s, we have*

$$\begin{aligned}\varepsilon &= \frac{4\eta T\sqrt{2sT\log(1/\delta)}}{n} \\ &= \frac{256\log^{3/2}|\mathcal{Q}|\sqrt{6\log(1/\delta)\log(2|\mathcal{X}|T/\beta)}}{\alpha^3 n}.\end{aligned}$$

Solving for α, we find

$$\alpha = O\left(\frac{\log^{1/2}|\mathcal{Q}|\log^{1/6}(1/\delta)\log^{1/6}(2|\mathcal{X}|/\gamma)}{n^{1/3}\varepsilon^{1/3}}\right),$$

for $\gamma < \beta/T$.

5 Case study: 3-way marginals

In our algorithm, the computationally difficult step is finding the data player's approximate best response against the query player's distribution. As mentioned above, the form of this problem depends on the particular query class $\mathcal{Q}$. In this section, we first discuss the optimization problem in general, and then specifically for the well-studied class of *marginal* queries.

5.1 The best-response problem

Recall that the query release game has payoff $A(x, q)$ defined by Equation (1); the data player tries to minimize the payoff, while the query player tries to maximize it. If the query player has distribution Q^t over queries, the data player's best response minimizes the expected loss:

$$\operatorname*{argmin}_{x \in \mathcal{X}} \mathop{\mathbb{E}}_{q \leftarrow Q^t} \left[q(D) - q(x) \right].$$

For privacy reasons, the data player actually plays against the distribution of samples $\widehat{q_1}, \dots, \widehat{q_s}$. So, the best-response problem is

$$\operatorname*{argmin}_{x \in \mathcal{X}} \frac{1}{s} \sum_{i=1}^{s} \widehat{q_i}(D) - \widehat{q_i}(x) = \operatorname*{argmax}_{x \in \mathcal{X}} \sum_{i=1}^{s} \widehat{q_i}(x),$$

since D is fixed and $\widehat{q_i}$ are linear queries. By Theorem 4.6 it even suffices to find an approximate maximizer, in order to guarantee accuracy.

5.2 3-way marginal queries

To look at the precise form of the best-response problem, we consider *3-way marginal* queries. We think of records as having d binary attributes, so that the data universe $|\mathcal{X}|$ is all bitstrings of length d. We write x_i for $x \in \mathcal{X}$ to mean the ith bit of record x.

Definition 5.1. Let $\mathcal{X} = \{0, 1\}^d$. A *3-way marginal query* is a linear query specified by 3 integers $a \neq b \neq c \in [d]$, taking values

$$q_{abc}(x) = \begin{cases} 1 & : x_a = x_b = x_c = 1 \\ 0 & : \text{otherwise.} \end{cases}$$

Though everything we do will apply to general k-way marginals, for concreteness we work with 3-way marginals. Recall that the query class $\mathcal{Q}$ includes each query and its negation. So, we also have negated conjunctions:

$$\overline{q_{abc}}(x) = \begin{cases} 0 & : x_a = x_b = x_c = 1 \\ 1 & : \text{otherwise.} \end{cases}$$

Given sampled conjunctions $\{\widehat{u_i}\}$ and negated conjunctions $\{\widehat{v_i}\}$, the best-response problem is

$$\operatorname*{argmax}_{x \in \mathcal{X}} \sum_i \widehat{u_i}(x) + \sum_j \widehat{v_j}(x).$$

In other words, this is a MAXCSP problem—we can associate a clause to each conjunction:

$$q_{abc} \Rightarrow (x_a \wedge x_b \wedge x_c) \quad \text{and} \quad \overline{q_{abc}} \Rightarrow (\overline{x_a} \vee \overline{x_b} \vee \overline{x_c}),$$

and we want to find $x \in \{0, 1\}^d$ satisfying as many clauses as possible.[3]

Since most solvers do not directly handle MAXCSP problems, we convert this optimization problem into a more standard, integer program form. We introduce a variable x_i for each literal x_i, a variable c_i

[3] Note that this is almost a MAX3SAT problem, except there are also "negative" clauses.

for each sampled conjunction $\widehat{u_i}$, a variable d_i for each sampled negated conjunction $\widehat{v_i}$, and we form the following integer program.

$$\begin{aligned} &\max \sum_i c_i + \sum_j d_j \\ &\text{with } \forall \widehat{u_i} = q_{abc} : \; x_a + x_b + x_c \geq 3c_i \\ &\qquad \forall \widehat{v_j} = \overline{q_{abc}} : \; (1 - x_a) + (1 - x_b) + (1 - x_c) \geq d_j \\ &\qquad x_i, c_i, d_i \in \{0, 1\} \end{aligned}$$

Note $x_i, 1 - x_i$ corresponds to the literals $x_i, \overline{x_i}$, $c_i = 1$, $d_i = 1$ exactly when their respective clauses are satisfied. Thus, the objective is the number of satisfied clauses. The resulting integer program can be solved using many existing solvers; we use CPLEX.

Dataset	Size	Attributes	Binary attributes
Adult	30162	14	235
KDD99	494021	41	396
Netflix	480189	17,770	17,770

Figure 1: Test Datasets

6 Experimental evaluation

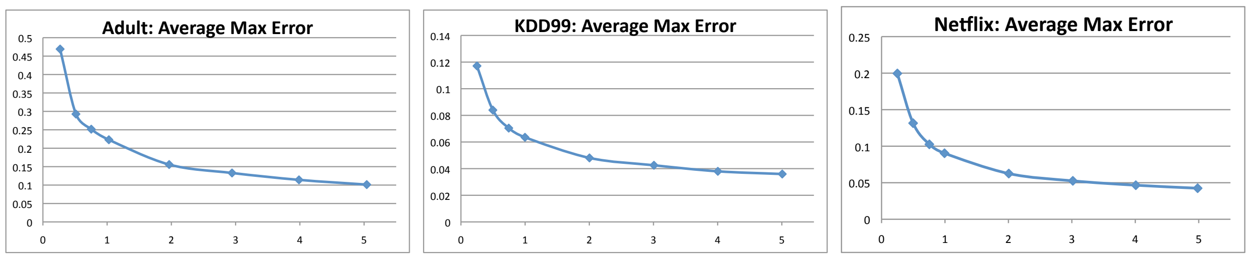

Figure 2: Average max error of $(\varepsilon, 0.001)$-private DualQuery on $500{,}000$ 3-way marginals versus ε.

We evaluate DualQuery on a large collection of 3-way marginal queries on several real datasets (Figure 1) and high dimensional synthetic data. Adult (census data) and KDD99 (network packet data) are from the UCI repository [2], and have a mixture of discrete (but non-binary) and continuous attributes, which we discretize into binary attributes. We also use the (in)famous Netflix [28] movie ratings dataset, with more than 17,000 binary attributes. More precisely, we can consider each attribute (corresponding to a movie) to be 1 if a user has watched that movie, and 0 otherwise.

Rather than set the parameters as in Algorithm 2, we experiment with a range of parameters. For instance, we frequently run for fewer rounds (lower T) and take fewer samples (lower s). As such, the accuracy guarantee (Theorem 4.6) need not hold for our parameters. However, we find that our algorithm gives good error, often much better than predicted. In all cases, our parameters satisfy the privacy guarantee Theorem 4.3.

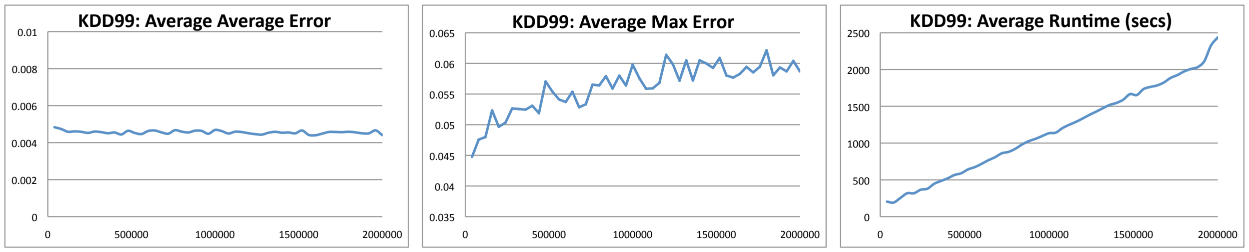

Figure 3: Error and runtime of $(1, 0.001)$-private DualQuery on KDD99 versus number of queries.

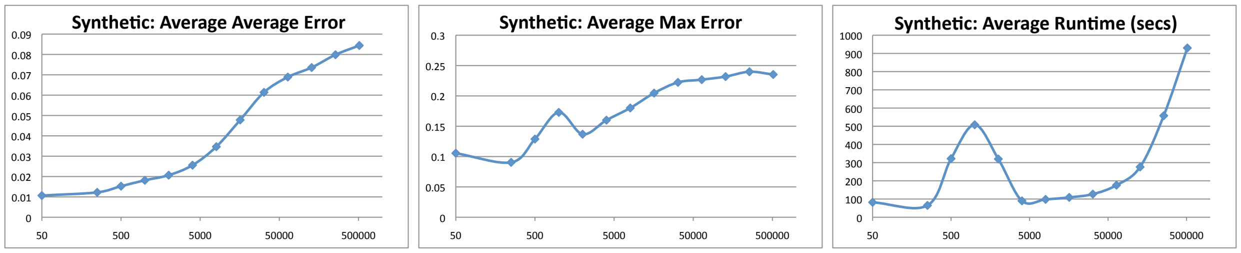

Figure 4: Error and runtime of $(1, 0.001)$-private DualQuery on $100{,}000$ 3-way marginal queries versus number of attributes.

6.1 Accuracy

We evaluate the accuracy of the algorithm on $500{,}000$ 3-way marginals on Adult, KDD99 and Netflix. We report maximum error in Figure 2, averaged over 5 runs. (Marginal queries have range $[0, 1]$, so error 1 is trivial.) The runs are $(\varepsilon, 0.001)$-differentially private, with ε ranging from 0.25 to 5.[4] For the Adult and KDD99 datasets, we set step size $\eta = 2.0$, sample size $s = 1000$ while varying the number of steps T according to the privacy budget ε, using the formula from Theorem 4.3. For the Netflix dataset, we adopt the same heuristic except we set s to be 5000.

The accuracy improves noticeably when ε increases from 0.25 to 1 across 3 datasets, and the improvement diminishes gradually with larger ε. With larger sizes, both KDD99 and Netflix datasets allow DualQuery to run with more steps and get significantly better error.

6.2 Scaling to More Queries

Next, we evaluate accuracy and runtime when varying the number of queries. We use a set of $40{,}000$ to 2 million randomly generated marginals $\mathcal{Q}$ on the KDD99 dataset and run DualQuery with $(1, 0.001)$-privacy. For all experiments, we use the same set of parameters: $\eta = 1.2, T = 170$ and $s = 1750$. By Theorem 4.3, each run of the experiment satisfies $(1, 0.001)$-differential privacy. These parameters give stable performance as the query class $\mathcal{Q}$ grows. As shown in Figure 3, both average and max error remain mostly stable, demonstrating improved error compared to simpler perturbation approaches. For example,

[4] Since our privacy analysis follows from Lemma 2.4, our algorithm actually satisfies (ε, δ)-privacy for smaller values of δ. For example, our algorithm is also $(\sqrt{2}\varepsilon, \delta')$-private for $\delta' = 10^{-6}$. Similarly, we could choose any arbitrarily small value of δ, and Lemma 2.4 would tell us that our algorithm was (ε', δ)-differentially private for an appropriate value ε', which depends only sub-logarithmically on $1/\delta$.

the Laplace mechanism's error growth rate is $O(\sqrt{|\mathcal{Q}|})$ under (ε, δ)-differential privacy. The runtime grows almost linearly in the number of queries, since we maintain a distribution over all the queries.

6.3 Scaling to Higher Dimensional Data

Finally, we evaluate accuracy and runtime behavior for data dimension ranging from 50 to $512{,}000$. We evaluate DualQuery under $(1, 0.001)$-privacy on 100,000 3-way marginals on synthetically genearted datasets. We report runtime, max, and average error over 3 runs in Figure 4; note the logarithmic scale for attributes axis. We do not include query evaluation in our time measurements—this overhead is common to all approaches that answer a set of queries.

When generating the synthetic data, one possibility is to set each attribute to be 0 or 1 uniformly at random. However, this generates very uniform synthetic data: a record satisfies any 3-way marginal with probability $1/8$, so most marginals will have value near $1/8$. To generate more challenging and realistic data, we pick a separate bias $p_i \in [0, 1]$ uniformly at random for each attribute i. For each data point, we then set attribute i to be 1 independently with probability equal to p_i. As a result, different 3-way marginals have different answers on our synthetic data.

For parameters, we fix step size η to be 0.4, and increase the sample size s with the dimension of the data (from 200 to 50,000) at the expense of running fewer steps. For these parameters, our algorithm is $(1, 0.001)$-differentially private by Theorem 4.3. With this set of parameters, we are able to obtain 8% average error in an average of 10 minutes of runtime, excluding query evaluation.

6.4 Methodology

In this section, we discuss our experimental setup in more detail.

Implementation details. The implementation is written in OCaml, using the CPLEX constraint solver. We ran the experiments on a mid-range desktop machine with a 4-core Intel Xeon processor and 12 Gb of RAM. Heuristically, we set a timeout for each CPLEX call to 20 seconds, accepting the best current solution if we hit the timeout. For the experiments shown, the timeout was rarely reached.

Data discretization. We discretize KDD99 and Adult datasets into binary attributes by mapping each possible value of a discrete attribute to a new binary feature. We bucket continuous attributes, mapping each bucket to a new binary feature. We also ensure that our randomly generated 3-way marginal queries are sensible (i.e., they don't require an original attribute to take two different values).

Setting free attributes. Since the collection of sampled queries may not involve all of the attributes, CPLEX often finds solutions that leave some attributes unspecified. We set these *free* attributes heuristically: for real data, we set the attributes to 0 as these datasets are fairly sparse; [5] for synthetic data, we set attributes to 0 or 1 uniformly at random. [6]

[5] The adult and KDD99 datasets are sparse due to the way we discretize the data; for the Netflix dataset, most users have only viewed a tiny fraction of the 17,000 movies.

[6] For a more principled way to set these free attributes, the sparsity of the dataset could be estimated at a small additional cost to privacy.

Parameter tuning. DualQuery has three parameters that can be set in a wide variety of configurations without altering the privacy guarantee (Theorem 4.3): number of iterations (T), number of samples (s), and learning rate (η), which controls how aggressively to update the distribution. For a fixed level of ε and δ, there are many feasible private parameter settings.

While the accuracy guarantee (Theorem 4.6) may not hold for a particular set of parameters, the empirical accuracy is much better than predicted. Performance depends strongly on the choice of parameters: T has an obvious impact, increasing s increases the number of constraints in the integer program for CPLEX. We have investigated a range of parameters; for the experiments we have used some informal heuristics coming from our observations.

For sparse datasets like the ones in Figure 1, with larger η (in the range of 1.5 to 2.0) and smaller s (in the same order as the number of attributes), we obtain good accuracy when we set the free attributes to be 0. But for denser data like our synthetic data, we get better accuracy when we have smaller η (around 0.4) and set the free attributes randomly. As for runtime, we observe that CPLEX could finish running quickly (in seconds) when the sample size is in about the same range as the number of attributes (factor of 2 or 3 different).

Parameter setting should be done under differential privacy for a truly realistic evaluation. Overall, we do not know of a principled approach to handling this issue; the problem of private parameter tuning is an area of active research (see e.g., Chaudhuri and Vinterbo [9]).

7 Discussion and conclusion

We have given a new private query release mechanism that can handle datasets with dimensionality multiple orders of magnitude larger than what was previously possible. Indeed, it seems we have not reached the limits of our approach—even on synthetic data with more than $500{,}000$ attributes, DualQuery continues to generate useful answers with about 30 minutes of overhead on top of query evaluation (which by itself is on the scale of hours). We believe that DualQuery makes private analysis of high dimensional data practical for the first time.

However, this remarkable improvement in running time is not free: our theoretical accuracy bounds are worse than those of previous approaches [18, 19]. For low dimensional datasets for which it is possible to maintain a distribution over records, the MWEM algorithm of Hardt et al. [19] likely remains the state of the art. Our work complements MWEM by allowing private data analysis on higher-dimensional data sets.

References

[1] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

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

[3] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems (PODS), Beijing, China*, pages 273–282, 2007.

[4] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Characterizing the sample complexity of private learners. In *ACM SIGACT Innovations in Theoretical Computer Science (ITCS), Berkeley, California*, pages 97–110, 2013.

[5] A. Blum, K. Ligett, and A. Roth. A learning theory approach to noninteractive database privacy. *Journal of the ACM*, 60(2):12, 2013.

[6] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the sulq framework. In *ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems (PODS), Baltimore, Maryland*, pages 128–138, 2005.

[7] Kamalika Chaudhuri and Daniel Hsu. Sample complexity bounds for differentially private learning. *Journal of Machine Learning Research*, 19:155–186, 2011.

[8] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In *Conference on Neural Information Processing Systems (NIPS), Vancouver, British Colombia*, pages 289–296, 2008.

[9] Kamalika Chaudhuri and Staal A. Vinterbo. A stability-based validation procedure for differentially private machine learning. pages 2652–2660, 2013.

[10] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12:1069–1109, 2011.

[11] Kamalika Chaudhuri, Anand Sarwate, and Kaushik Sinha. Near-optimal differentially private principal components. In *Conference on Neural Information Processing Systems (NIPS), Lake Tahoe, California*, pages 998–1006, 2012.

[12] J.C. Duchi, M.I. Jordan, and M.J. Wainwright. Local privacy and statistical minimax rates. In *IEEE Symposium on Foundations of Computer Science (FOCS), Berkeley, California*, 2013.

[13] C. Dwork, M. Naor, O. Reingold, G.N. Rothblum, and S.P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *ACM SIGACT Symposium on Theory of Computing (STOC), Bethesda, Maryland*, pages 381–390, 2009.

[14] C. Dwork, G.N. Rothblum, and S. Vadhan. Boosting and differential privacy. In *IEEE Symposium on Foundations of Computer Science (FOCS), Las Vegas, Nevada*, pages 51–60, 2010.

[15] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *IACR Theory of Cryptography Conference (TCC), New York, New York*, pages 265–284, 2006.

[16] Y. Freund and R.E. Schapire. Game theory, on-line prediction and boosting. In *Conference on Computational Learning Theory (CoLT), Desenzano sul Garda, Italy*, pages 325–332, 1996.

[17] A. Gupta, A. Roth, and J. Ullman. Iterative constructions and private data release. In *IACR Theory of Cryptography Conference (TCC), Taormina, Italy*, pages 339–356, 2012.

[18] Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *IEEE Symposium on Foundations of Computer Science (FOCS), Las Vegas, Nevada*, pages 61–70, 2010.

[19] Moritz Hardt, Katrina Ligett, and Frank McSherry. A simple and practical algorithm for differentially private data release. In *Conference on Neural Information Processing Systems (NIPS), Lake Tahoe, California*, pages 2348–2356, 2012.

[20] Justin Hsu, Aaron Roth, and Jonathan Ullman. Differential privacy for the analyst via private equilibrium computation. In *ACM SIGACT Symposium on Theory of Computing (STOC), Palo Alto, California*, pages 341–350, 2013.

[21] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[22] Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45 (6):983–1006, 1998.

[23] Daniel Kifer, Adam Smith, and Abhradeep Thakurta. Private convex empirical risk minimization and high-dimensional regression. *Journal of Machine Learning Research*, 1:41, 2012.

[24] Chao Li and Gerome Miklau. An adaptive mechanism for accurate query answering under differential privacy. volume 5, pages 514–525, 2012.

[25] Chao Li, Michael Hay, Vibhor Rastogi, Gerome Miklau, and Andrew McGregor. Optimizing linear counting queries under differential privacy. In *ACM SIGACT–SIGMOD–SIGART Symposium on Principles of Database Systems (PODS), Indianapolis, Indiana*, pages 123–134, 2010.

[26] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *IEEE Symposium on Foundations of Computer Science (FOCS), Providence, Rhode Island*, pages 94–103, 2007.

[27] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy (S&P), Oakland, California*, pages 111–125, 2008.

[28] Netflix. Netflix prize.

[29] Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In *ACM SIGACT Symposium on Theory of Computing (STOC), Cambridge, Massachusetts*, pages 765–774.

[30] Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for SVM learning. *Journal of Privacy and Confidentiality*, 4 (1):4, 2012.

[31] Abhradeep G. Thakurta and Adam Smith. (Nearly) optimal algorithms for private online learning in full-information and bandit settings. In *Conference on Neural Information Processing Systems (NIPS), Lake Tahoe, California*, pages 2733–2741, 2013.

[32] J. Ullman. Answering $n^{2+o(1)}$ counting queries with differential privacy is hard. In *ACM SIGACT Symposium on Theory of Computing (STOC), Palo Alto, California*, pages 361–370, 2013.

[33] J. Ullman and S.P. Vadhan. PCPs and the hardness of generating private synthetic data. In *IACR Theory of Cryptography Conference (TCC), Providence, Rhode Island*, pages 400–416, 2011.

[34] Grigory Yaroslavtsev, Graham Cormode, Cecilia M. Procopiuc, and Divesh Srivastava. Accurate and efficient private release of datacubes and contingency tables. In *IEEE International Conference on Data Engineering (ICDE), Brisbane, Australia*, pages 745–756, 2013.

A Case study: Parity queries

In this section, we show how to apply DualQuery to another well-studied class of queries: *parities*. Each specified by a subset S of features, these queries measure the number of records with an even number of bits on in S compared to the number of records with an odd number of bits on in S.

Definition A.1. Let $\mathcal{X} = \{-1, +1\}^d$. A *$k$-wise parity query* is a linear query specified by a subset of features $S \subseteq [d]$ with $|S| = k$, taking values

$$q_S(x) = \begin{cases} +1 & : \text{even number of } x_i = +1 \text{ for } i \in S \\ -1 & : \text{otherwise.} \end{cases}$$

Like before, we can define a *negated k-wise parity query*:

$$\overline{q_S}(x) = \begin{cases} +1 & : \text{odd number of } x_i = +1 \text{ for } i \in S \\ -1 & : \text{otherwise.} \end{cases}$$

Barak et al. [3] observed that answering k-way marginal queries can be reduced to answering k-wise parity queries to the same accuracy, in the following sense.

Theorem A.2 (Barak et al. [3])**.** *Let q_S be a k-way marginal query specified by the set of features S, and let D be a database of records. Then,*

$$q_S(D) = \frac{1}{2^k} \sum_{T \subseteq S} p_T(D),$$

where p_T is the parity query for features T.

Note that the coefficients sum to 1: hence, answering parity queries with additive error α is enough to answer marginal queries with additive error α. We now consider how to handle these queries with our algorithm. For the remainder, we specialize to $k = 3$. Like marginal queries, it suffices to give the best-response optimization problem; unlike marginal queries, we need to handle k-wise parities for every $k \leq 3$ in order to apply Theorem A.2.

Given sampled parity queries $\{\widehat{u_i}\}$ and negated parity queries $\{\widehat{v_i}\}$, the best response problem is to find the record $x \in \mathcal{X}$ that takes value 1 on as many of these queries as possible. We can construct an integer program for this task: introduce d variables x_i, and two variables c_q, d_q for each sampled query. The following integer program encodes the best-response problem.

$$\begin{aligned} \max \sum_i c_i & \\ \text{such that } \forall \widehat{u_i} = q_S. \ \sum_j x_{S_j} &= 2d_i + c_i - 1 \\ \forall \widehat{v_i} = \overline{q_S}. \ \sum_j x_{S_j} &= 2d_i + c_i \\ x_i, c_i, d_i &\in \{0, 1\} \end{aligned}$$

Consider the (non-negated) parity queries first. The idea is that each variable c_i can be set to 1 exactly when the corresponding parity query takes value 1, i.e., when x has an even number of bits in S set to $+1$. Since $|S| \leq 3$, this even number will either be 0 or 2, hence is equal to $2d_i$ for $d_i \in \{0, 1\}$. A similar argument holds for the negated parity queries.