Differentially Private Query Release Through Adaptive Projection

Sergul Aydore [1] **William Brown** [1 2] **Michael Kearns** [1 3] **Krishnaram Kenthapadi** [1]
Luca Melis [1] **Aaron Roth** [1 3] **Ankit Siva** [1]

Abstract

We propose, implement, and evaluate a new algorithm for releasing answers to very large numbers of statistical queries like k-way marginals, subject to differential privacy. Our algorithm makes adaptive use of a continuous relaxation of the *Projection Mechanism*, which answers queries on the private dataset using simple perturbation, and then attempts to find the synthetic dataset that most closely matches the noisy answers. We use a continuous relaxation of the synthetic dataset domain which makes the projection loss differentiable, and allows us to use efficient ML optimization techniques and tooling. Rather than answering all queries up front, we make judicious use of our privacy budget by iteratively finding queries for which our (relaxed) synthetic data has high error, and then repeating the projection. Randomized rounding allows us to obtain synthetic data in the original schema. We perform experimental evaluations across a range of parameters and datasets, and find that our method outperforms existing algorithms on large query classes.

1. Introduction

A basic problem in differential privacy is to accurately answer a large number m of statistical queries (also known as *linear* and *counting* queries), which have the form, "how many people in private dataset D have property P?" Marginal queries (also known as *conjunctions*) — which ask how many people in the dataset have particular combinations of feature values — are one of the most useful and most studied special cases. The simplest technique for answering such queries is to compute each answer on the private dataset, and then perturb them with independent Gaussian noise. For a dataset of size n, this results in error scaling as $\tilde{O}\left(\frac{\sqrt{m}}{n}\right)$ (Dwork et al., 2006a). This simple technique is useful for answering small numbers of queries. But it has been known since (Blum et al., 2008) that *in principle*, it is possible to privately and accurately answer very large classes of queries (of size exponential in n), and that an attractive way of doing so is to encode the answers in a *synthetic dataset*. Synthetic datasets have several advantages: most basically, they are a concise way of representing the answers to large numbers of queries. But they also permit one to evaluate queries *other* than those that have been explicitly answered by the mechanism, and to take advantage of *generalization*. Unfortunately, it is also known that improving on the error of the simple Gaussian perturbation technique is computationally hard in the worst case (Ullman, 2016). Moreover, constructing synthetic datasets is hard even when it would be possible to provide accurate answers with simple perturbation (Ullman & Vadhan, 2011) for simple classes of queries such as the set of all $\binom{d}{2}$ marginal queries restricted to 2 out of d binary features (so-called 2-way marginals). As a result we cannot hope for a differentially private algorithm that can provably answer large numbers of statistical queries or generate interesting synthetic data in polynomial time.

Nevertheless, there has been a resurgence of interest in private synthetic data generation and large-scale private queries due to the importance of the problem. Recent methods offer provable privacy guarantees, but have run-time and accuracy properties that must be evaluated empirically.

1.1. Our Contributions

Our starting point is the (computationally inefficient) *projection mechanism* of (Nikolov et al., 2013), which is informally described as follows. We begin with a dataset $D \in \mathcal{X}^n$. First, the values of each of the m queries of interest q_i are computed on the private dataset: $a = q(D) \in [0,1]^m$. Next, a privacy preserving vector of noisy answers $\hat{a} \in \mathbb{R}^m$ is computed using simple Gaussian perturbation. Finally, the vector of noisy answers $\hat{a}$ is *projected* into the set of answer vectors that are consistent with some dataset to obtain a final vector of answers a' — i.e., the projection guarantees that $a' = q(D')$ for *some* $D' \in \mathcal{X}^n$. This corresponds to solving the optimization problem of finding the synthetic

[1]Amazon Web Services AI/ML [2]Columbia University, New York, NY, USA [3]University of Pennsylvania, Philadelphia, PA, USA. Correspondence to: William Brown <w.brown@columbia.edu>, Ankit Siva <ankitsiv@amazon.com>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

dataset $D' \in \mathcal{X}^n$ that minimizes error $||q(D') - \hat{a}||_2$. This is known to be a near optimal mechanism for answering statistical queries (Nikolov et al., 2013) but for most data and query classes, the projection step corresponds to a difficult discrete optimization problem. We remark that the main purpose of the projection is not (only) to construct a synthetic dataset, but to improve accuracy. This is analogous to how learning with a restricted model class like linear classifiers can improve accuracy if the data really is labeled by some linear function, i.e., the projection improves accuracy because by projecting into a set that contains the *true* vector of answers a, it is imposing constraints that we know to be satisfied by the true (unknown) vector of answers.

Our core algorithm is based on a continuous relaxation of this projection step. This allows us to deploy first-order optimization methods, which empirically work very well despite the non-convexity of the problem. A further feature of this approach is that we can take advantage of sophisticated existing tooling for continuous optimization — including autodifferentiation (to allow us to easily handle many different query classes) and GPU acceleration, which has been advanced by a decade of research in deep learning. This is in contrast to related approaches like (Gaboardi et al., 2014; Vietri et al., 2020) which use integer program solvers and often require designing custom integer programs for optimizing over each new class of queries.

We then extend our core algorithm by giving an adaptive variant that is able to make better use of its privacy budget, by taking advantage of generalization properties. Rather than answering *all* of the queries up front, we start by answering a small number of queries, and then project them onto a vector of answers consistent with a relaxed synthetic dataset — i.e., a dataset in a larger domain than the original data — but one that still allows us to evaluate queries. At the next round, we use a private selection mechanism to find a small number of additional queries on which our current synthetic dataset performs poorly; we answer those queries, find a new synthetic dataset via our continuous projection, and then repeat. If the queries we have answered are highly accurate, then we are often able to find synthetic data representing the original data well after only having explicitly answered a very small number of them (i.e., we *generalize* well to new queries). This forms a virtuous cycle, because if we only need to explicitly answer a very small number of queries, we can answer them highly accurately with our privacy budget. By taking our relaxed data domain to be the set of *probability distributions* over one-hot encodings of the original data domain, we can finally apply randomized rounding to output a synthetic dataset in the original schema.

We evaluate our algorithm on several datasets, comparing it to two state-of-the-art algorithms from the literature. A key advantage of our algorithm is that we can scale to large query workloads (in our experiments we answer roughly 20 million queries on some datasets and do not hit computational bottlenecks). We outperform the state of the art algorithm FEM ("Follow-the-Perturbed-Leader with Exponential Mechanism") from (Vietri et al., 2020), which is one of the few previous techniques able to scale to large workloads. We also compare to algorithms that are unable to scale to large workloads, comparing to one of the state of the art methods, optimized variants of the HDMM ("High Dimensional Matrix Mechanism") from (McKenna et al., 2019). When run on a workload of roughly 65 thousand queries provided by the authors of (McKenna et al., 2019), HDMM outperforms our algorithm. The result is an algorithm that we believe to be state of the art for large query workloads, albeit one that can be outperformed for smaller workloads.

1.2. Additional Related Work

Differential privacy offers a formal semantics for data privacy and was introduced by (Dwork et al., 2006b). The differential privacy literature is far too large to survey here; see (Dwork & Roth, 2014) for a textbook introduction.

The problem of answering large numbers of queries on a private dataset (often via synthetic data generation) dates back to (Blum et al., 2008). A line of early theoretical work (Blum et al., 2008; Roth & Roughgarden, 2010; Hardt & Rothblum, 2010; Gupta et al., 2012; Nikolov et al., 2013) established statistical rates for answering very general classes of queries, showing that it is possible in principle (i.e., ignoring computation) to provide answers to *exponentially many* queries in the size of the dataset. This line of work establishes statistically optimal rates for the problem (i.e., matching statistical lower bounds), but provides algorithms that have running time that is generally *exponential* in the data dimension, and hence impractical for even moderately high dimensional data. Moreover, this exponential running time is known to be necessary in the worst case (Dwork et al., 2009; Ullman & Vadhan, 2011; Ullman, 2016). As a result, a line of work has emerged that tries to avoid this exponential running time in practice. The "Multiplicative Weights Exponential Mechanism" (Hardt et al., 2012) uses optimizations to avoid exponentially large representations when the query class does not require it. Dwork, Nikolov, and Talwar give a theoretical analysis of a convex relaxation of the projection mechanism that can answer k-way marginals in time polynomial in d^k — albeit with accuracy that is sub-optimal by a factor of $d^{k/2}$ (Dwork et al., 2015)."Dual Query" (Gaboardi et al., 2014) used a dual representation of the optimization problem implicitly solved by (Roth & Roughgarden, 2010; Hardt & Rothblum, 2010; Hardt et al., 2012) to trade off the need to manipulate exponentially large state with the need to solve concisely defined but NP-hard integer programs. This was an "oracle efficient" algorithm. The theory of oracle efficient synthetic data release was further developed in (Neel et al., 2019), and (Vietri et al., 2020) give further improvements on oracle efficient algorithms in this dual representation, and promising experimental results. We compare against the algorithm from (Vietri et al., 2020) in our empirical results. We remark that marginal queries (the focus of our experimental evaluation) have been considered a canonical special case of the general query release problem, and the explicit focus of a long line of work (Barak et al., 2007; Thaler et al., 2012; Cormode et al., 2018; Chandrasekaran et al., 2014; Gupta et al., 2013).

A parallel line of work on *matrix mechanisms* focused on optimizing error within a restricted class of mechanisms. Informally speaking, this class answers a specially chosen set of queries explicitly with simple perturbation, and then deduces the answers to other queries by taking linear combinations of those that were explicitly answered. One can optimize the error of this approach by optimizing over the set of queries that are explicitly answered (Li et al., 2015). Doing this exactly is also intractable, because it requires manipulating matrices that are exponential in the data dimension. This line of work too has seen heuristic optimizations, and the "high dimensional matrix mechanism" (McKenna et al., 2018) together with further optimizations (McKenna et al., 2019) is able to scale to higher dimensional data and larger collections of queries — although to date the size of the query classes that these algorithms can answer is smaller by several orders of magnitude compared to our algorithm and others in the oracle efficient line of work.

Finally, there is a line of work that has taken modern techniques for distribution learning (GANs, VAEs, etc.) and has made them differentially private, generally by training using private variants of stochastic gradient descent (Beaulieu-Jones et al., 2019; Jordon et al., 2018; Torkzadehmahani et al., 2019; Neunhoeffer et al., 2020; Takagi et al., 2020). This line of work has shown some promise for image data as measured by visual fidelity, and for limited kinds of downstream machine learning tasks — but generally has not shown promising results for enforcing consistency with simple classes of statistics like marginal queries. As a result we do not compare to approaches from this line of work.

2. Preliminaries

2.1. Statistical Queries and Synthetic Data

Let $\mathcal{X}$ be a data domain. In this paper, we will focus on data points containing d categorical features: i.e. $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_d$, where each $\mathcal{X}_i$ is a set of t_i categories. A *dataset* (which we will denote by D) consists of a multiset of n points from $\mathcal{X}$: $D \in \mathcal{X}^n$.

Definition 2.1 (Statistical Query (Kearns, 1998))**.** A *statistical query* (also known as a *linear query* or *counting query*) is defined by a function $q_i : \mathcal{X} \to [0, 1]$. Given a dataset D, we will abuse notation and write $q_i(D)$ to denote the average value of the function q_i on D:

$$q_i(D) = \frac{1}{n} \sum_{x \in D} q_i(x)$$

Given a collection of m statistical queries $\{q_i\}_{i=1}^m$, we write $q(D) \in [0, 1]^m$ to denote the vector of values $q(D) = (q_1(D), \ldots, q_m(D))$.

An important type of statistical query is a k-way marginal, which counts the number of data points $x \in D$ that have a particular realization of feature values for some subset of k features.[1]

Definition 2.2. A k-way marginal query is defined by a subset $S \subseteq [d]$ of $|S| = k$ features, together with a particular value for each of the features $y \in \prod_{i \in S} \mathcal{X}_i$. Given such a pair (S, y), let $\mathcal{X}(S, y) = \{x \in \mathcal{X} : x_i = y_i \ \ \forall i \in S\}$ denote the set of points that match the feature value y_i for each of the k features in S. The corresponding statistical query $q_{S,y}$ is defined as:

$$q_{S,y}(x) = \mathbb{1}(x \in \mathcal{X}(S, y))$$

Observe that for each collection of features (*marginal*) S, there are $\prod_{i \in S} |\mathcal{X}_i|$ many queries.

Given a set of m statistical queries q, we will be interested in vectors of answers $a' \in [0, 1]^m$ that represent their answers on D *accurately*:

Definition 2.3. Given a dataset D, a collection of m statistical queries represented as $q : \mathcal{X}^n \to [0, 1]^m$, and a vector of estimated answers $a' \in [0, 1]^m$, we say that a' has ℓ_∞ or *max* error α if $\max_{i \in [m]} |q_i(D) - a'_i| \le \alpha$.

In this paper we will represent vectors of estimated answers a' *implicitly* using some data structure D' on which we can evaluate queries, and will write $q(D')$ for a'. If $D' \in \mathcal{X}^*$, then we refer to D' as a *synthetic dataset* — but we will also make use of D' lying in continuous relaxations of $\mathcal{X}^n$ (and will define how query evaluation applies to such "relaxed datasets").

2.2. Differential Privacy

Two datasets $D, D' \in \mathcal{X}^n$ are said to be *neighboring* if they differ in at most one data point. We will be interested in *randomized algorithms* $\mathcal{A} : \mathcal{X}^n \to R$ (where R can be an arbitrary range).

[1]We define marginals for datasets with discrete features. In our experimental results we encode continuous features as discrete by standard binning techniques.

Definition 2.4 (Differential Privacy (Dwork et al., 2006b;a)). A randomized algorithm $\mathcal{A} : \mathcal{X}^n \to R$ is (ϵ, δ) differentially private if for all pairs of neighboring datasets $D, D' \in \mathcal{X}^n$ and for all measurable $S \subseteq R$:

$$\Pr[\mathcal{A}(D) \in S] \leq \exp(\epsilon) \Pr[\mathcal{A}(D') \in S] + \delta.$$

If $\delta = 0$ we say that $\mathcal{A}$ is ϵ-differentially private.

Differential privacy is not convenient for tightly handling the degradation of parameters under composition, and so as a tool for our analysis, we use the related notion of (zero) Concentrated Differential Privacy:

Definition 2.5 (Zero Concentrated Differential Privacy (Bun & Steinke, 2016)). An algorithm $\mathcal{A} : \mathcal{X}^n \to R$ satisfies ρ-zero Concentrated Differential Privacy (zCDP) if for all pairs of neighboring datasets $D, D' \in \mathcal{X}^n$, and for all $\alpha \in (0, \infty)$:

$$\mathbb{D}_\alpha(\mathcal{A}(D), \mathcal{A}(D')) \leq \rho\alpha$$

where $\mathbb{D}_\alpha(\mathcal{A}(D), \mathcal{A}(D'))$ denotes the α-Renyi divergence between the distributions $\mathcal{A}(D)$ and $\mathcal{A}(D')$.

zCDP enjoys clean composition and postprocessing properties:

Lemma 2.6 (Composition (Bun & Steinke, 2016)). Let $\mathcal{A}_1 : \mathcal{X}^n \to R_1$ be ρ_1-zCDP. Let $\mathcal{A}_2 : \mathcal{X}^n \times R_1 \to R_2$ be such that $\mathcal{A}_2(\cdot, r)$ is ρ_2-zCDP for every $r \in R_1$. Then the algorithm $\mathcal{A}(D)$ that computes $r_1 = \mathcal{A}_1(D)$, $r_2 = \mathcal{A}_2(D, r_1)$ and outputs (r_1, r_2) satisfies $(\rho_1 + \rho_2)$-zCDP.

Lemma 2.7 (Post Processing (Bun & Steinke, 2016)). Let $\mathcal{A} : \mathcal{X}^n \to R_1$ be ρ-zCDP, and let $f : R_1 \to R_2$ be an arbitrary randomized mapping. Then $f \circ \mathcal{A}$ is also ρ-zCDP.

Together, these lemmas mean that we can construct zCDP mechanisms by modularly combining zCDP sub-routines. Finally, we can relate differential privacy with zCDP:

Lemma 2.8 (Conversions (Bun & Steinke, 2016)).

1. If $\mathcal{A}$ is ϵ-differentially private, it satisfies $(\frac{1}{2}\epsilon^2)$-zCDP.
2. If $\mathcal{A}$ is ρ-zCDP, then for any $\delta > 0$, it satisfies $(\rho + 2\sqrt{\rho \log(1/\delta)}, \delta)$-differential privacy.

We will make use of two basic primitives from differential privacy, which we introduce here in the context of statistical queries. The first is the Gaussian mechanism.

Definition 2.9 (Gaussian Mechanism). The Gaussian mechanism $G(D, q_i, \rho)$ takes as input a dataset $D \in \mathcal{X}^n$, a statistical query q_i, and a zCDP parameter ρ. It outputs $a_i = q_i(D) + Z$, where $Z \sim N(0, \sigma^2)$, where $N(0, \sigma^2)$ is the Gaussian distribution with mean 0 and variance $\sigma^2 = \frac{1}{2n^2\rho}$.

Lemma 2.10 ((Bun & Steinke, 2016)). For any statistical query q_i and parameter $\rho > 0$, the Gaussian mechanism $G(\cdot, q_i, \rho)$ satisfies ρ-zCDP.

The second is a simple private "selection" mechanism called report noisy max — we define a special case here, tailored to our use of it.

Definition 2.11 (Report Noisy Max With Gumbel Noise). The "Report Noisy Max" mechanism $RNM(D, q, a, \rho)$ takes as input a dataset $D \in \mathcal{X}^n$, a vector of m statistical queries q, a vector of m conjectured query answers a, and a zCDP parameter ρ. It outputs the index of the query with highest noisy error estimate. Specifically, it outputs $i^* = \arg\max_{i \in [m]}(|q_i(D) - a_i| + Z_i)$ where each $Z_i \sim \text{Gumbel}\left(1/\sqrt{2\rho}n\right)$.

Lemma 2.12. For any vector of statistical queries q, vector of conjectured answers a, and zCDP parameter ρ, $RNM(\cdot, q, a, \rho)$ satisfies ρ-zCDP.

Proof. The report noisy max mechanism with Gumbel noise is equivalent to the exponential mechanism for sensitivity 1/n queries, and hence satisfies the bounded range property as defined in (Durfee & Rogers, 2019). Lemma 3.2 of (Cesar & Rogers, 2021) converts bounded range guarantees to zCDP guarantees, from which the claim follows. □

3. Relaxing the Projection Mechanism

The projection mechanism of (Nikolov et al., 2013) can be described simply in our language. Given a collection of m statistical queries q and zCDP parameter ρ, it consists of two steps:

1. For each i, evaluate q_i on D using the Gaussian mechanism: $\hat{a}_i = G(D, q_i, \rho/m)$.
2. Find the synthetic dataset[2] D' whose query values are closest to $\hat{a}$ in ℓ_2 norm — i.e., let $D' = \arg\min_{D' \in \mathcal{X}^*} ||q(D') - \hat{a}||_2$.

The output of the mechanism is the synthetic dataset D', which implicitly encodes the answer vector $a' = q(D')$. Because the perturbation in Step 1 is Gaussian, and the projection is with respect to the ℓ_2 norm, D' is the maximum likelihood estimator for the dataset D given the noisy statistics $\hat{a}$. The projection also serves to enforce consistency constraints across all query answers, which perhaps counter-intuitively, is accuracy-improving. For intuition, the

[2]In fact, in (Nikolov et al., 2013), the projection is onto a set of datasets that allows datapoints to have positive or negative weights — but their analysis also applies to projections onto the set of synthetic datasets in our sense. A statement of this can be found as Lemma 5.3 in (Błasiok et al., 2019).

reader can consider the case in which all queries q_i are identical: in this case, the scale of the initial Gaussian noise is $\Omega(\sqrt{m}/n)$, which is sub-optimal, because the single query of interest could have been privately answered with noise scaling only as $O(1/n)$. But the effect of the projection will be similar to *averaging* all of the perturbed answers $\hat{a}_i$, because $q_i(D')$ will be constrained to take a fixed value across all i (since the queries are identical), and the mean of a vector of noisy estimates minimizes the Euclidean distance to those estimates. This has the effect of averaging out much of the noise, recovering error $O(1/n)$. The projection mechanism is easily seen to be ρ-zCDP — the m applications of (ρ/m)-zCDP instantiations of the Gaussian mechanism in Step 1 compose to satisfy ρ-zCDP by the composition guarantee of zCDP (Lemma 2.6), and Step 2 is a post-processing operation, and so by Lemma 2.7 does not increase the privacy cost. This mechanism is nearly optimal amongst the class of all differentially private mechanisms, as measured by ℓ_2 error, in the worst case over the choice of statistical queries (Nikolov et al., 2013). Unfortunately, Step 2 is in general an intractable computation, since it is a minimization of a non-convex and non-differentiable objective over an exponentially large discrete space. The first idea that goes into our algorithm (Algorithm 1) is to relax the space of datasets $\mathcal{X}^n$ to be a continuous space, and to generalize the statistical queries q_i to be differentiable over this space. Doing so allows us to apply powerful GPU-accelerated tools for differentiable optimization to the projection step 2.

From Categorical to Real Valued Features Our first step is to embed categorical features into *binary* features using a one-hot encoding. This corresponds to replacing each categorical feature $\mathcal{X}_i$ with t_i binary features $\mathcal{X}_i^1 \times \ldots \times \mathcal{X}_i^{t_i} = \{0,1\}^{t_i}$, for each $x \in \mathcal{X}$. Exactly one of these new t_i binary features corresponding to categorical feature i is set to 1 for any particular data point $x \in \mathcal{X}$: If $x_i = v_j$ for some $v_j \in \mathcal{X}_i$, then we set $\mathcal{X}_i^j = 1$ and $\mathcal{X}_i^{j'} = 0$ for all $j' \neq j$. Let $d' = \sum_{i=1}^d t_i$ be the dimension of a feature vector that has been encoded using this one-hot encoding. Under this encoding, the datapoints x are embedded in the binary feature space $\{0,1\}^{d'}$. We will aim to construct synthetic data that lies in a continuous relaxation of this binary feature space. For example, choosing $\mathcal{X}^r = [0,1]^{d'}$ is natural. In our experiments, we choose $\mathcal{X}^r = [-1,1]^{d'}$, which empirically leads to an easier optimization problem. In Section 4, we discuss a more structured projection mechanism that preserves the original dataset schema.

Let $h : \mathcal{X} \to \{0,1\}^{d'}$ represent the function that maps a $x \in \mathcal{X}$ to its one-hot encoding. We abuse notation and for a dataset $D \in \mathcal{X}^n$, write $h(D)$ to denote the one-hot encoding of every $x \in D$.

From Discrete to Differentiable Queries Consider a marginal query $q_{S,y} : \mathcal{X} \to \{0,1\}$ defined by some $S \subseteq [d]$ and $y \in \prod_{i \in S} \mathcal{X}_i$. Such a query can be evaluated on a vector of categorical features $x \in \mathcal{X}$ in our original domain. Our goal is to construct an *equivalent extended differentiable query* $\hat{q}_{S,y} : \mathcal{X}^r \to \mathbb{R}$ that has two properties:

Definition 3.1 (Equivalent Extended Differentiable Query)**.** Given a statistical query $q_i : \mathcal{X} \to [0,1]$, we say that $\hat{q}_i : \mathcal{X}^r \to \mathbb{R}$ is an extended differentiable query that is equivalent to q_i if it satisfies the following two properties:

1. $\hat{q}_i$ is differentiable over $\mathcal{X}^r$ — i.e. for every $x \in \mathcal{X}^r$, $\nabla q_i(x)$ is defined, and
2. $\hat{q}_i$ agrees with q_i on every feature vector that results from a one-hot encoding. In other words, for every $x \in \mathcal{X}$: $q_i(x) = \hat{q}_i(h(x))$.

We will want to give equivalent extended differentiable queries for the class of k-way marginal queries. Towards this end, we define a product query:

Definition 3.2. Given a subset of features $T \subseteq [d']$, the product query $q_T : \mathcal{X}^r \to \mathbb{R}$ is defined as: $q_T(x) = \prod_{i \in T} x_i$.

By construction, product queries satisfy the first requirement for being extended differentiable queries: they are defined over the entire relaxed feature space $\mathcal{X}^r$, and are differentiable (since they are monomials over a real valued vector space). It remains to observe that for every marginal query $q_{S,y}$, there is an equivalent product query $\hat{q}_{S,y}$ that takes value $q_{S,y}(x)$ on the one-hot encoding $h(x)$ of x for every x.

Lemma 3.3. Every k-way marginal query has an equivalent extended differentiable query in the class of product queries. In other words, for every k-way marginal query $q_{S,y} : \mathcal{X}^n \to \{0,1\}$, there is a corresponding product query $\hat{q}_{S,y} = q_T(y) : \mathcal{X}^r \to \mathbb{R}$ with $|T| = k$ such that for every $x \in \mathcal{X}$: $q_{S,y}(x) = q_T(h(x))$.

Proof. We construct T in the straightforward way: for every $i \in S$, we include in T the coordinate corresponding to $y_i \in \mathcal{X}_i$. Now consider any x such that $q_{S,y}(x) = 1$. It must be that for every $i \in S$, $x_i = y_i$. By construction, the product $q_T(h(x)) = \prod_{j \in T} h(x)_j = 1$ because all terms in the product evaluate to 1. Similarly, if $q_{S,y}(x) = 0$, then it must be that for at least one coordinate $j \in T$, $h(x)_j = 0$, and so $q_T(h(x)) = \prod_{j \in T} h(x)_j = 0$. □

4. The Relaxed Adaptive Projection (RAP) Mechanism

We here introduce the "Relaxed Adaptive Projection" (RAP) mechanism (Algorithm 2), which has three hyperparameters: the *number of adaptive rounds* T, the *number of queries per round* K, and the *size of the (relaxed) synthetic dataset* n'. In the simplest case, when $T = 1$ and $K = m$, we recover the natural relaxation of the projection mechanism:

1. We evaluate each query $q_i \in Q$ on D using the Gaussian mechanism to obtain a noisy answer $\hat{a}_i$, and

2. Find a *relaxed* synthetic dataset $D' \in X^r$ whose equivalent extended differentiable query values are closest to $\hat{a}$ in ℓ_2 norm: $D' = \arg\min_{D' \in (\mathcal{X}^r)^{n'}} ||\hat{q}(D') - \hat{a}||_2$.

Because step 2 is now optimizing a continuous, differentiable function over a continuous space (of dimension $d' \cdot n'$, we can use existing tool kits for performing the optimization – for example, we can use auto-differentiation tools, and optimizers like Adam (Kingma & Ba, 2015). (Recall that the projection is a post-processing of the Gaussian mechanism, and so the privacy properties of the algorithm are independent of our choice of optimizer). Here n' is a hyperparameter that we can choose to trade off the expressivity of the synthetic data with the running-time of the optimization: If we choose $n' = n$, then we are assured that it is possible to express D exactly in our relaxed domain: as we choose smaller values of n', we introduce a source of representation error, but decrease the dimensionality of the optimization problem in our projection step, and hence improve the runtime of the algorithm. In this simple case, we can recover an accuracy theorem by leveraging the results of (Nikolov et al., 2013):

Theorem 4.1. *Fix privacy parameters $\epsilon, \delta > 0$, a synthetic dataset size n', and any set of m k-way product queries q. If the minimization in the projection step is solved exactly, then the average error for the RAP mechanism when $T = 1$ and $K = m$ can be bounded as:*

$$\sqrt{\frac{1}{m}||q(D) - q(D')||_2^2} \leq$$

$$O\left(\frac{(d'(\log k + \log n') + \log(1/\beta)\ln(1/\delta))^{1/4}}{\sqrt{\epsilon n}} + \frac{\sqrt{\log k}}{\sqrt{n'}}\right)$$

with probability $1 - \beta$ over the realization of the Gaussian noise.

See Appendix A for proof.

This is an "oracle efficient" accuracy theorem in the style of (Gaboardi et al., 2014; Vietri et al., 2020; Neel et al., 2020) in the sense that it assumes that our heuristic optimization succeeds (note that this assumption is not needed for the privacy of our algorithm, which we establish in Theorem 4.2). Compared to the accuracy theorem for the FEM algorithm proven in (Vietri et al., 2020), our theorem improves by a factor of $\sqrt{d'}$.

In the general case, our algorithm runs in T rounds: After each round t, we have answered some *subset* of the queries $Q_S \subseteq Q$, and perform a projection only with respect to the queries in Q_S for which we have estimates, obtaining an intermediate relaxed synthetic dataset D'_t. At the next round, we augment Q_S with K additional queries q_i from $Q \setminus Q_S$ chosen (using report noisy max) to maximize the disparity $|q_i(D'_t) - q_i(D)|$. We then repeat the projection. In total, this algorithm only explicitly answers $T \cdot K$ queries, which might be $\ll m$. But by selectively answering queries for which the consistency constraints imposed by the projection with respect to previous queries have not correctly fixed, we aim to expend our privacy budget more wisely. Adaptively answering a small number of "hard" queries has its roots in a long theoretical line of work (Roth & Roughgarden, 2010; Hardt & Rothblum, 2010; Gupta et al., 2012).

Algorithm 1 Relaxed Projection (RP)

Input: A vector of differentiable queries $q : \mathcal{X}^r \to \mathbb{R}^{m'}$, a vector of target answers $\hat{a} \in \mathbb{R}^{m'}$, and an initial dataset $D' \in (\mathcal{X}^r)^{n'}$.

Use any differentiable optimization technique (Stochastic Gradient Descent, Adam, etc.) to attempt to find:

$$D_S = \arg\min_{D' \in (\mathcal{X}^r)^{n'}} ||q(D') - \hat{a}||_2^2$$

Output D_S.

Theorem 4.2. *For any query class Q, any set of parameters K, T, n', and any privacy parameters $\epsilon, \delta > 0$, the RAP mechanism $RAP(\cdot, Q, K, T, n', \epsilon, \delta)$ (Algorithm 2) is (ϵ, δ)-differentially private.*

See Appendix B for proof.

Randomized Rounding to Output a Synthetic Dataset We use the SparseMax (Martins & Astudillo, 2016) transformation to generate relaxed synthetic data in which each set of one-hot columns, corresponding to the original features, is normalized to a (sparse) probability distribution. More specifically, after each step of the optimization technique in Algorithm. 1, we apply SparseMax independently to each set of encoded columns in the synthetic dataset D_S. Randomized rounding (i.e. for each feature independently, selecting a one-hot encoding with probability proportional to its probability in the relaxed synthetic data) can then be applied to produce a synthetic dataset consistent with the original schema. This preserves the expected value of marginal queries, and can preserve their values exactly in the limit as we take multiple samples. As we show in our experiments, preserving the worst case error over many marginals requires only moderate oversampling in practice (5 samples per data point).

Algorithm 2 Relaxed Adaptive Projection (RAP)

Input: A dataset D, a collection of m statistical queries Q, a "queries per round" parameter $K \leq m$, a "number of iterations" parameter $T \leq m/K$, a synthetic dataset size n', and differential privacy parameters ϵ, δ.
Let ρ be such that:

$$\epsilon = \rho + 2\sqrt{\rho \log(1/\delta)}$$

if $T = 1$ **then**
  for $i = 1$ to m **do**
    Let $\hat{a}_i = G(D, q_i, \rho/m)$.
  end for
  Randomly initialize $D' \in (\mathcal{X}^r)^{n'}$.
  Output $D' = RP(q, \hat{a}, D')$.
else
  Let $Q_S = \emptyset$ and $D'_0 \in (\mathcal{X}^r)^{n'}$ be an arbitrary initialization.
  for $t = 1$ to T **do**
    for $k = 1$ to K **do**
      Define $\hat{q}^{Q \setminus Q_S}(x) = (\hat{q}_i(x) : q_i \in Q \setminus Q_S)$ where $\hat{q}_i$ is an equivalent extended differentiable query for q_i.
      Let $q_i = RNM(D, \hat{q}^{Q \setminus Q_S}, \hat{q}^{Q \setminus Q_S}(D'_{t-1}), \frac{\rho}{2T \cdot K})$.
      Let $Q_S = Q_S \cup \{q_i\}$.
      Let $\hat{a}_i = G(D, q_i, \frac{\rho}{2T \cdot K})$.
    end for
    Define $q^{Q_S}(x) = (q_i(x) : q_i \in Q_S)$ and $\hat{a} = \{\hat{a}_i : q_i \in Q_S\}$ where $\hat{q}_i$ is an equivalent extended differentiable query for q_i. Let $D'_t = RP(q^{Q_S}, \hat{a}, D'_{t-1})$.
  end for
  Output D'_T.
end if

5. Empirical Evaluation

5.1. Implementation and Hyperparameters

We implement[3] Algorithm 2 in Python (Van Rossum & Drake, 2009), using the JAX library (Bradbury et al., 2018) for auto-differentiation of queries and the Adam optimizer (Kingma & Ba, 2015) (with learning rate 0.001) for the call to RP (Algorithm 1). For each call to RP, we do early stopping if the relative improvement on the loss function between consecutive Adam steps is less than 10^{-7}. The number of maximum Adam steps per RP round is set to 5000. Fig. 1 contains a Jax code snippet, which computes 3-way product queries on a dataset D. A benefit of using JAX (or other packages with autodifferentiation capabilities) is that to instantiate the algorithm for a new query class, all that is required is to write a new python function which computes queries in the class — we do not need to perform any other reasoning about the class. In contrast, approaches like (Gaboardi et al., 2014; Vietri et al., 2020) require deriving an integer program to *optimize* over each new class of interest, and approaches like (McKenna et al., 2019) require performing an expensive optimization over each new workload of interest. This makes our method more easily extensible.

```
import jax.numpy as np
def threeway_marginals(D):
    return np.einsum('ij,ik,il->jkl', D, D,
        D)/D.shape[0]
```

Figure 1. Python function used to compute 3-way product queries

JAX also has the advantages of being open source and able to take advantage of GPU acceleration. We run our experiments for Algorithm 2 on an EC2 p2.xlarge instance (1 GPU, 4 CPUs, 61 GB RAM). For FEM we use the code from the authors of (Vietri et al., 2020) available at https://github.com/giusevtr/fem, using the hyperparameters given in their tables 2 and 3 for the experimental results we report in Figures 2 and 3, respectively. Their code requires the Gurobi integer program solver; we were able to obtain a license to Gurobi for a personal computer, but not for EC2 instances, and so we run FEM on a 2019 16" MacBook Pro (6 CPUs, 16GB RAM) (Gurobi does not support GPU acceleration) — as a result we do not report timing comparisons. We remark that an advantage of our approach is that it can leverage the robust open-source tooling (like JAX and Adam) that has been developed for deep learning, to allow us to easily take advantage of large-scale distributed GPU accelerated computation.

For HDMM+LSS and HDMM+PGM implementations, we used code provided by the authors of (McKenna et al., 2019) which was hard-coded with a query strategy for a particular set of 62876 marginal queries on the Adult dataset, which we also run on a MacBook Pro.

For most experiments, we set the size of the synthetic data $n' = 1000$ — significantly smaller than n for both of our datasets (see Table 1). See Appendix C for an investigation of performance as a function of n'. For the remaining hyperparameters K and T, we optimize over a small grid of values (see Table 2) and report the combination with the smallest error. This is also how error is reported for FEM. For all experiments we optimize over $\mathcal{X}^r = [-1, 1]^{d'}$, which empirically had better convergence rates compared to using $\mathcal{X}^r = [0, 1]^{d'}$ — likely because gradients of our queries vanish at 0.

[3] github.com/amazon-research/relaxed-adaptive-projection

Dataset	Records	Features	Transformed Binary Features
ADULT	48842	15	588
LOANS	42535	48	4427

Table 1. Datasets. Each dataset starts with the given number of original (categorical and real valued) features. After our transformation, it is encoded as a dataset with a larger number of binary features.

Parameter	Description	Values
K	Queries per round	5 10 25 50 100
T	Number of iterations	2 5 10 25 50

Table 2. RAP hyperparameters tested in our experiments

5.2. Selecting Marginals

For our main set of experiments comparing to the FEM algorithm of (Vietri et al., 2020), we mirror their experimental design in (Vietri et al., 2020), and given k, we select a number of marginals S (i.e., subsets of categorical features), referred to as the *workload*, at random, and then enumerate all queries consistent with the selected marginals (i.e., we enumerate all $y \in \prod_{i \in S} \mathcal{X}_i$). For each experiment, we fix the query selection process and random seed so that both algorithms in our comparisons are evaluated on exactly the same set of queries. See Fig. 7 in Appendix C for the total number of selected queries across different workloads on both of our datasets, which vary in a range between 10^5 and 10^8. For our comparison to the HDMM variants of (McKenna et al., 2019), we compare on the particular set of 62876 3-way marginal queries on Adult for which the hard-coded query strategy in their provided code is optimized on.

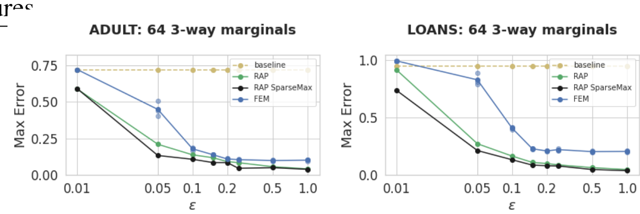
5.3. Experimental Results

We evaluate both our algorithm and FEM on the two datasets used by (Vietri et al., 2020) in their evaluation: ADULT and LOANS (Dua & Graff, 2017).

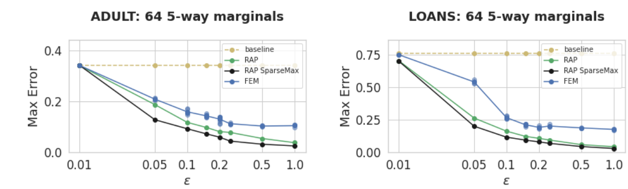
Just as in (Vietri et al., 2020), both datasets are transformed so that all features are categorical — real valued features are first bucketed into a finite number of categories. The algorithms are then run on a one-hot encoding of the discrete features, as we described in Section 3. To ensure consistency, we use the pre-processed data exactly as it appears in their repository for (Vietri et al., 2020). See Table 1 for a summary of the datasets.

We mirror the evaluation in (Vietri et al., 2020) and focus our experiments comparing to FEM on answering 3-way and 5-way marginals.

We also compare to the High Dimensional Matrix Mechanism (HDMM) with Local Least Squares (HDMM+LLS) and Probabilistic Graphical Model (HDMM+PGM) inference from (McKenna et al., 2019), but these mechanisms do not scale to large workloads, and the existing implementations are hard-coded with optimizations for a fixed set of queries on Adult. Hence in our comparison to HDMM+LSS and HDMM+PGM, we can only run these algorithms on the fixed set of 62876 3-way marginals defined on the Adult dataset that the code supports.



(a) ADULT dataset on 3-way marginals (b) LOANS dataset on 3-way marginals



(c) ADULT dataset on 5-way marginals (d) LOANS dataset on 5-way marginals

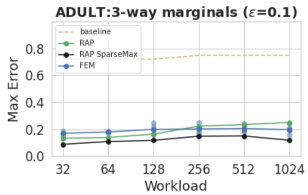
Figure 2. Max-error for 3 and 5-way marginal queries on different privacy levels. The number of marginals is fixed at 64.

We use the maximum error between answers to queries on the synthetic data and the correct answers on the real data across queries ($\max_i |q_i(D') - q_i(D)|$) as a performance measure. For calibration, we also report a naive baseline corresponding to the error obtained by answering every query with "0". Error above this naive baseline is uninteresting. For all experiments, we fix the privacy parameter δ to $\frac{1}{n^2}$, where n is the number of records in the dataset, and vary ϵ as reported.

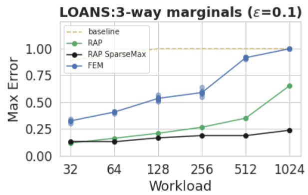
In Figs. 2-5(a) we show how our performance scales with the privacy budget ϵ for a fixed number of marginals. Figs. 3, 4 show our performance for a fixed privacy budget as we increase the number of marginals being preserved.

We significantly outperform FEM in all comparisons considered, and performance is particularly strong in the important high-privacy and high workload regimes (i.e., when ϵ is small and m is large). However, both HDMM+PGM and HDMM+LLS outperform RAP in the small workload regime in the comparison we are able to run.

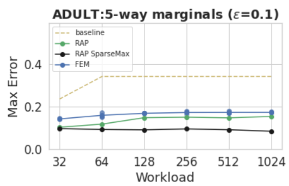
Figure 5 (b) shows how randomized rounding, when applied on the synthetic dataset generated by RAP and SparseMax, affects the error on the marginals for different levels of over-



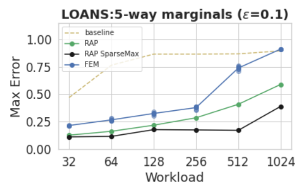
(a) ADULT dataset on 3-way marginals



(b) LOANS dataset on 3-way marginals

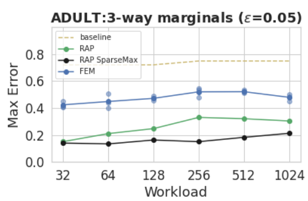


(c) ADULT dataset on 5-way marginals

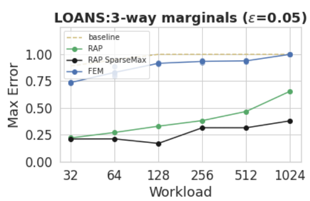


(d) LOANS dataset on 5-way marginals

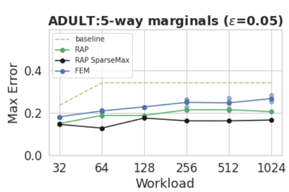
Figure 3. Max error for increasing number of 3 and 5-way marginal queries with $\epsilon = 0.1$



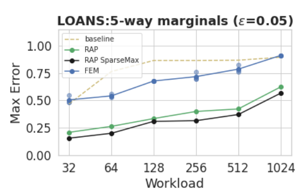
(a) ADULT dataset on 3-way marginals



(b) LOANS dataset on 3-way marginals



(c) ADULT dataset on 5-way marginals



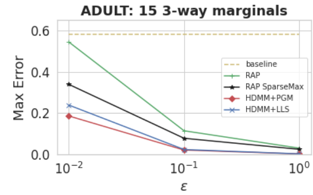
(d) LOANS dataset on 5-way marginals

Figure 4. Max error for increasing number of 3 and 5-way marginal queries with $\epsilon = 0.05$

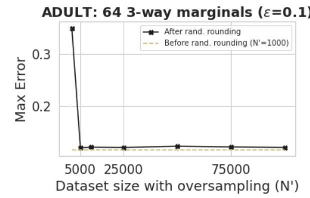
sampling. The error after randomly rounding each data point 5 times (obtaining a synthetic dataset of size $n' = 5,000$) approaches the error before applying randomized rounding and slowly converges for larger oversampling rates.

We also investigate the run-time and accuracy of our algorithm as a function of the synthetic dataset size n' — see Figure 6, and Appendix C for more details. Here we note two things: (i) We can take n' quite small as a function of the true dataset size n, until a certain point (below $n' = 1000$) at which point error starts increasing, (ii) Run time also decreases with n', until we take n' quite small, at which point the optimization problem appears to become more difficult.

Finally, as we have noted already, an advantage of our approach is its easy extensibility: to operate on a new query class, it is sufficient to write the code to evaluate queries in that class. To demonstrate this, in the Appendix we plot results for a different query class: linear threshold functions.

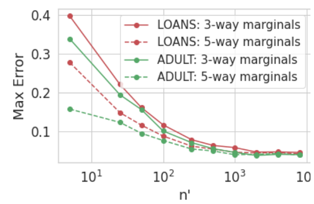


(a)

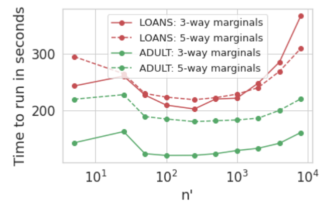


(b)

Figure 5. (a):Max-error of HDMM variants and RAP for the set of 15 3-way marginal queries on ADULT provided by (McKenna et al., 2019) at different privacy levels. (b) Max Error of RAP before and after randomized rounding with different levels of oversampling.



(a)



(b)

Figure 6. (a) Error and (b) run-time as a function of the synthetic dataset size n'.

6. Conclusion

We have presented a new, extensible method for privately answering large numbers of statistical queries, and producing synthetic data consistent with those queries. Our method relies on a continuous, differentiable relaxation of the projection mechanism, which allows us to use existing powerful tooling developed for deep learning. We demonstrate on a series of experiments that our method out-performs existing techniques across a wide range of parameters in the large workload regime.

References

Barak, B., Chaudhuri, K., Dwork, C., Kale, S., McSherry, F., and Talwar, K. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 273–282, 2007.

Beaulieu-Jones, B. K., Wu, Z. S., Williams, C., Lee, R., Bhavnani, S. P., Byrd, J. B., and Greene, C. S. Privacy-preserving generative deep neural networks support clinical data sharing. *Circulation: Cardiovascular Quality and Outcomes*, 12(7):e005122, 2019.

Błasiok, J., Bun, M., Nikolov, A., and Steinke, T. Towards instance-optimal private query release. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 2480–2497. SIAM, 2019.

Blum, A., Ligett, K., and Roth, A. A learning theory approach to non-interactive database privacy. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pp. 609–618, 2008.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Bun, M. and Steinke, T. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pp. 635–658. Springer, 2016.

Cesar, M. and Rogers, R. Bounding, concentrating, and truncating: Unifying privacy loss composition for data analytics. In *Algorithmic Learning Theory*, pp. 421–457. PMLR, 2021.

Chandrasekaran, K., Thaler, J., Ullman, J., and Wan, A. Faster private release of marginals on small databases. In *Proceedings of the 5th conference on Innovations in theoretical computer science*, pp. 387–402, 2014.

Cormode, G., Kulkarni, T., and Srivastava, D. Marginal release under local differential privacy. In *Proceedings of the 2018 International Conference on Management of Data*, pp. 131–146, 2018.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Durfee, D. and Rogers, R. M. Practical differentially private top-k selection with pay-what-you-get composition. *Advances in Neural Information Processing Systems*, 32: 3532–3542, 2019.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014. ISSN 1551-305X. doi: 10.1561/0400000042. URL https://doi.org/10.1561/0400000042.

Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., and Naor, M. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 486–503. Springer, 2006a.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. D. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006b.

Dwork, C., Naor, M., Reingold, O., Rothblum, G. N., and Vadhan, S. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 381–390, 2009.

Dwork, C., Nikolov, A., and Talwar, K. Efficient algorithms for privately releasing marginals via convex relaxations. *Discrete & Computational Geometry*, 53(3): 650–673, 2015.

Gaboardi, M., Gallego-Arias, E. J., Hsu, J., Roth, A., and Wu, Z. S. Dual query: Practical private query release for high dimensional data. In *International Conference on Machine Learning*, pp. 1170–1178, 2014.

Gupta, A., Roth, A., and Ullman, J. Iterative constructions and private data release. In *Theory of cryptography conference*, pp. 339–356. Springer, 2012.

Gupta, A., Hardt, M., Roth, A., and Ullman, J. Privately releasing conjunctions and the statistical query barrier. *SIAM Journal on Computing*, 42(4):1494–1520, 2013.

Hardt, M. and Rothblum, G. N. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pp. 61–70. IEEE, 2010.

Hardt, M., Ligett, K., and McSherry, F. A simple and practical algorithm for differentially private data release. In *Advances in Neural Information Processing Systems*, pp. 2339–2347, 2012.

Jordon, J., Yoon, J., and van der Schaar, M. Pate-gan: Generating synthetic data with differential privacy guarantees. In *International Conference on Learning Representations*, 2018.

Kearns, M. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998.

Kingma, D. P. and Ba, J. L. Adam: A method for stochastic gradient descent. In *ICLR: International Conference on Learning Representations*, 2015.

Li, C., Miklau, G., Hay, M., McGregor, A., and Rastogi, V. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB journal*, 24 (6):757–781, 2015.

Martins, A. and Astudillo, R. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*. PMLR, 2016.

McKenna, R., Miklau, G., Hay, M., and Machanavajjhala, A. Optimizing error of high-dimensional statistical queries under differential privacy. *Proceedings of the VLDB Endowment*, 11(10):1206–1219, 2018.

McKenna, R., Sheldon, D., and Miklau, G. Graphical-model based estimation and inference for differential privacy. In *International Conference on Machine Learning*, pp. 4435–4444. PMLR, 2019.

Neel, S., Roth, A., and Wu, Z. S. How to use heuristics for differential privacy. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 72–93. IEEE, 2019.

Neel, S., Roth, A., Vietri, G., and Wu, S. Oracle efficient private non-convex optimization. In *International Conference on Machine Learning*, pp. 7243–7252. PMLR, 2020.

Neunhoeffer, M., Wu, Z. S., and Dwork, C. Private post-GAN boosting. *arXiv preprint arXiv:2007.11934*, 2020.

Nikolov, A., Talwar, K., and Zhang, L. The geometry of differential privacy: the sparse and approximate cases. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 351–360, 2013.

Roth, A. and Roughgarden, T. Interactive privacy via the median mechanism. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pp. 765–774, 2010.

Takagi, S., Takahashi, T., Cao, Y., and Yoshikawa, M. P3GM: Private high-dimensional data release via privacy preserving phased generative model. *arXiv preprint arXiv:2006.12101*, 2020.

Thaler, J., Ullman, J., and Vadhan, S. Faster algorithms for privately releasing marginals. In *International Colloquium on Automata, Languages, and Programming*, pp. 810–821. Springer, 2012.

Torkzadehmahani, R., Kairouz, P., and Paten, B. DP-CGAN: Differentially private synthetic data and label generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

Ullman, J. Answering n^2+o(1) counting queries with differential privacy is hard. *SIAM Journal on Computing*, 45(2):473–496, 2016.

Ullman, J. and Vadhan, S. PCPs and the hardness of generating private synthetic data. In *Theory of Cryptography Conference*, pp. 400–416. Springer, 2011.

Van Rossum, G. and Drake, F. L. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

Vietri, G., Tian, G., Bun, M., Steinke, T., and Wu, S. New oracle-efficient algorithms for private synthetic data release. In *International Conference on Machine Learning*, pp. 9765–9774. PMLR, 2020.