Privately Releasing Conjunctions and the Statistical Query Barrier

Anupam Gupta* Moritz Hardt† Aaron Roth‡ Jonathan Ullman§

November 26, 2024

Abstract

Suppose we would like to know *all* answers to a set of statistical queries C on a data set up to small error, but we can only access the data itself using statistical queries. A trivial solution is to exhaustively ask all queries in C. Can we do any better?

1. We show that the number of statistical queries necessary and sufficient for this task is—up to polynomial factors—equal to the agnostic learning complexity of C in Kearns' statistical query (SQ) model. This gives a complete answer to the question when running time is not a concern.
2. We then show that the problem can be solved efficiently (allowing arbitrary error on a small fraction of queries) whenever the answers to C can be described by a submodular function. This includes many natural concept classes, such as graph cuts and Boolean disjunctions and conjunctions.

While interesting from a learning theoretic point of view, our main applications are in *privacy-preserving data analysis*: Here, our second result leads to an algorithm that efficiently releases differentially private answers to all Boolean conjunctions with 1% average error. This presents significant progress on a key open problem in privacy-preserving data analysis. Our first result on the other hand gives unconditional lower bounds on any differentially private algorithm that admits a (potentially non-privacy-preserving) implementation using only statistical queries. Not only our algorithms, but also most known private algorithms can be implemented using only statistical queries, and hence are constrained by these lower bounds. Our result therefore isolates the complexity of agnostic learning in the SQ-model as a new barrier in the design of differentially private algorithms.

*Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213. Research was partly supported by NSF award CCF-1016799 and an Alfred P. Sloan Fellowship.

†Center for Computational Intractability, Department of Computer Science, Princeton University. Supported by NSF grants CCF-0426582 and CCF-0832797. Email: mhardt@cs.princeton.edu.

‡Microsoft Research New England, Cambridge MA. Email: alroth@cs.cmu.edu.

§School of Engineering and Applied Sciences, Harvard University, Cambridge MA. Supported by NSF grant CNS-0831289. Email: jullman@seas.harvard.edu.

1 Introduction

Consider a data set $D \subseteq \{0,1\}^d$ in which each element corresponds to an individual's record over d binary attributes. The goal of privacy-preserving data analysis is to enable rich statistical analyses on the data set while respecting individual privacy. In paritcular, we would like to guarantee *differential privacy* [DMNS06], a rigorous notion of privacy that guarantees the outcome of a statistical analysis is nearly indistinguishable on any two data sets that differ only in a single individual's data.

One of the most important classes of statistical queries on the data set are Boolean conjunctions, sometimes called contingency tables or marginal queries. See, for example, [BCD+07, BLR08, KRSU10, UV10]. A boolean conjunction corresponding to a subset $S \subseteq [d]$ counts what fraction of the individuals have each attribute in S set to 1. A major open problem in privacy-preserving data analysis is to efficiently create a differentially private synopsis of the data set that accurately encodes answers to all Boolean conjunctions. In this work we give an algorithm with runtime polynomial in d, which outputs a differentially private data structure that represents all boolean conjunctions up to an average error of 1%.

Our result is significantly more general and applies to any collection of queries that can be described by a low sensitivity *submodular* function. Submodularity is a property that often arises in data analysis and machine learning problems [KG07], including problems for which privacy is a first-order design constraint[1]. Imagine, for example, a social network on d vertices. A data analyst may wish to analyze the size of the cuts induced by various subsets of the vertices. Here, our result provides a data structure that represents all cuts up to a small average error. Another important example of submodularity is the *set-coverage* function, which given a set system over elements in some universe U, represents the number of elements that are covered by the union of any collection of the sets.

The size of our data structure grows exponentially in the inverse error desired, and hence we can represent submodular functions only up to constant error if we want polynomial query complexity. *Can any efficient algorithm do even better?* We give evidence that in order to do better, fundamentally new techniques are needed. Specifically, we show that no polynomial-time algorithm that guarantees small error for *every* boolean conjunction can do substantially better if the algorithm permits an implementation that only accesses the database through statistical queries. This statement holds regardless of whether such an implementation is privacy-preserving. (A statistical query is given by a function $q\colon \{0,1\}^d \to \{0,1\}$, to which the answer is $\mathbb{E}_{x \in D}[q(x)]$.)

We show this limitation using connection between the data release problem and standard problems in learning theory. Putting aside privacy concerns, we pose the following question: *How many statistical queries to a data set are necessary and sufficent in order to approximately answer* all *queries in a class C?* We show that the number of statistical queries necessary and sufficient for this task is, up to a factor of $O(d)$, equal to the agnostic learning complexity of C (over arbitrary distributions) in Kearns' statistical query (SQ) model [Kea98]. Using an SQ lower bound for agnostically learning monotone conjunctions shown by Feldman [Fel10], this connection implies that no polynomial-time algorithm operating in the SQ-model can release even monotone conjunctions to subconstant error. Since monotone conjunction queries can be described by a submodular function, the lower bound applies to releasing submodular functions as well.

While the characterization above is independent of privacy concerns, it has two immediate implications for private data release:

- Firstly, it also characterizes what can be released in the *local privacy* model of Kasiviswanathan et al. [KLN+08]; this follows from the fact that [KLN+08] showed that SQ algorithms are precisely what can be computed in the local privacy model.

[1] For example, Kempe, Kleinberg, and Tardos show that for two common models of influence propagation on social networks, the function capturing the "influence" of a set of users (perhaps the targets of a viral marketing campaign) is a monotone submodular function [KKT03].

- Secondly, and perhaps even more importantly, it gives us the claimed unconditional lower bounds on the running time of any query-release algorithm that permits an implementation using only statistical queries—regardless of whether its privacy analysis can be carried out in the local privacy model. To our knowledge, this class includes almost all privacy preserving algorithms developed to date, including the recently introduced Median Mechanism [RR10] and Multiplicative Weights Mechanism [HR10][2]. Note that these mechanisms cannot be implemented in the local privacy model while preserving their privacy guarantees, because they will have to make too many queries. Indeed, they are capable of releasing conjunctions to subconstant error! Yet, they can be implemented using only statistical queries, and so our lower bounds apply to their running time.

To summarize, our results imply that if we want to develop efficient algorithms to solve the query release problem for classes as expressive as monotone conjunctions (itself an extremely simple class!), we need to develop techniques that are able to sidestep this *statistical query barrier.* On a conceptual level, our results present new reductions from problems in differential privacy to problems in learning theory.

1.1 Overview of our results

In this section we give an informal statement of our theorems with pointers to the relevant sections. Our theorem on approximating submodular functions is proved in Section 3. The definition of submodularity is found in the Preliminaries (Section 2).

Informal Theorem 1.1 (Approximating submodular functions)**.** *Let $\alpha > 0, \beta > 0$. Let $f\colon \{0,1\}^d \to [0,1]$ be a submodular function. Then, there is an algorithm with runtime $d^{O(\log(1/\beta)/\alpha^2)}$ which produces an approximation $h\colon \{0,1\}^d \to [0,1]$ such that $\Pr_{x\in\{0,1\}^d}\{|f(x) - h(x)| \leq \alpha\} \geq 1 - \beta$.*

In Section 4 we then show how this algorithm gives the following differentially private release mechanism for Boolean conjunctions. The definition of differential privacy is given in Section 2.

Informal Theorem 1.2 (Differentially private query release for conjunctions)**.** *Let $\alpha > 0, \beta > 0$. There is an ε-differentially private algorithm with runtime $d^{O(\log(1/\beta)/\alpha^2)}$ which releases the set of Boolean conjunctions with error at most α on a $1 - \beta$ fraction of the queries provided that $|D| \geq d^{O(\log(1/\beta)/\alpha^2)}/\varepsilon$.*

The guarantee in our theorem can be refined to give an α-approximation to a $1 - \beta$ fraction of the set of w-way conjunctions (conjunctions of width w) for all $w \in \{1, ..., d\}$. Nevertheless, our algorithm has the property that the error may be larger than α on a small fraction of the queries. We note, however, that for $\beta \leq \alpha^p/2$ our guarantee is stronger than error α in the L_p-norm which is also a natural objective that has been considered in other works. For example, Hardt and Talwar study error bounds on mechanisms with respect to the Euclidean norm across all answers [HT10]. From a practical point of view, it also turns out that some privacy-preserving algorithms in the literature indeed only require the ability to answer *random* conjunction queries privately, e.g., [JPW09].

Finally, in Section 5, we study the general query release problem and relate it to the agnostic learning complexity in the Statistical Query model.

Informal Theorem 1.3 (Equivalence between query release and agnostic learning)**.** *Suppose there exists an algorithm that learns a class C up to error α under arbitrary distributions using at most q statistical queries. Then, there is a release mechanism for C that makes at most $O(qd/\alpha^2)$ statistical queries.*

Moreover, any release mechanism for C that makes at most q statistical queries implies an agnostic learner that makes at most $2q$ queries.

[2] A notable exception is the private parity-learning algorithm of [KLN+08], which explicitly escapes the statistical query model.

While both reductions preserve the query complexity of the problem neither reduction preserves runtime. We also note that our equivalence characterization is more general than what we stated: the same proof shows that agnostic learning of a class C is (up to small factors) information theoretically equivalent to releasing the answers to all queries in a class C for any class of algorithms that may access the database only in some restricted manner. The ability to make only SQ queries is one restriction, and the requirement to be differentially private is another. Thus, we also show that on a class by class basis, the privacy cost of releasing the answers to a class of queries using any technique is not much larger than the privacy cost of simply optimizing over the same class to find the query with the highest value, and vice versa.

Our techniques. Our release algorithm is based on a structural theorem about general submodular functions $f : 2^U \to [0, 1]$ that may be of independent interest. Informally, we show that any submodular function has a "small" "approximate" representation. Specifically, we show that for any $\alpha > 0$, there exist at most $|U|^{2/\alpha}$ submodular functions g_i such that each g_i satisfies a strong Lipschitz condition, and for each $S \subset U$, there exists an i such that $f(S) = g_i(S)$. We then take advantage of Vondrak's observation in [Von10] that Lipschitz submodular functions are *self-bounding*, which allows us to apply recent dimension-free concentration bounds for self-bounding functions [BLM00, BLM09]. These concentration results imply that if we associate each function g_i with its expectation, and respond to queries $f(S)$ with $\mathbb{E}[g_i(S)]$ for the appropriate g_i, then most queries are answered to within only α additive error. This yields an algorithm for *learning* submodular functions over product distributions, which can easily be made privacy preserving when the values $f(S)$ correspond to queries on a sensitive database.

Our characterization of the query complexity of the release problem in the SQ model uses the multiplicative weights method [LW94, AHK05] similar to how it was used recently in [HR10]. That is we maintain a distribution over the universe on which the queries are defined. What is new is the observation that an agnostic learning algorithm for a class C can be used to find a query from C that distinguishes between the true data set and our distribution as much as possible. Such a query can then be used in the multiplicative weights update to reduce the relative entropy between the true data set and our distribution significantly. Since the relative entropy is nonnegative there can only be a few such steps before we find a distribution which provides a good approximation to the true data set on *all* queries in the class C.

1.2 Related Work

Learning Submodular Functions. The problem of learning submodular functions was recently introduced by Balcan and Harvey [BH10]; their PAC-style definition was different from previously studied point-wise learning approaches [GHIM09, SF08]. For product distributions, Balcan and Harvey give an algorithm for learning monotone, Lipschitz continuous submodular functions up to constant *multiplicative* error using only random examples. [BH10] also give strong lower bounds and matching algorithmic results for non-product distributions. Our main algorithmic result is similar in spirit, and is inspired by their concentration-of-measure approach. Our model is different from theirs, which makes our results incomparable. We introduce a decomposition that allows us to learn arbitrary (i.e. potentially non-Lipschitz, non-monotone) submodular functions to constant *additive* error. Moreover, our decomposition makes value queries to the submodular function, which are prohibited in the model studied by [BH10].

Information Theoretic Characterizations in Privacy. Kasiviswanathan et al. [KLN+08] introduced the *centralized* and *local* models of privacy and gave information theoretic characterizations for which classes of functions could be *learned* in these models: they showed that information theoretically, the class of functions that can be learned in the centralized model of privacy is equivalent to the class of functions that can be agnostically PAC learned, and the class of functions that can be learned in the local privacy model is equivalent to the class of functions that can be learned in the SQ model of Kearns [Kea98].

Blum, Ligett, and Roth [BLR08] considered the *query release* problem (the task of releasing the approximate value of all functions in some class) and characterized exactly which classes of functions can be information theoretically released while preserving differential privacy in the *centralized* model of data privacy. They also posed the question: which classes of functions can be released using mechanisms that have running time only polylogarithmic in the size of the data universe and the class of interest? In particular, they asked if conjunctions were such a class.

In this paper, we give an exact information theoretic characterization of which classes of functions can be released in the SQ model, and hence in the local privacy model: we show that it is exactly the class of functions that can be *agnostically learned* in the SQ model. We note that the agnostic SQ learnability of a class C (and hence, by our result, the SQ releasability of C) can also be characterized by combinatorial properties of C, as done by Blum et al. [BFJ+94] and recently Feldman [Fel10].

Lower bounds and hardness results. There are also several conditional lower bounds on the running time of private mechanisms for solving the query release problem. Dwork et al. [DNR+09] showed that under cryptographic assumptions, there exists a class of queries that can be privately released using the inefficient mechanism of [BLR08], but cannot be privately released by any mechanism that runs in time polynomial in the dimension of the data universe (e.g. d, when the data universe is $\{0,1\}^d$). Ullman and Vadhan [UV10] extended this result to the class of conjunctions: they showed that under cryptographic assumptions, no polynomial time mechanism *that outputs a data set* can answer even the set of d^2 conjunctions of two-literals!

The latter lower bound applies only to the class of mechanisms that output data sets, rather than some other data structure encoding their answers, and only to mechanisms that answer *all* conjunctions of two-literals with small error. In fact, because there are only d^2 conjunctions of size 2 in total, the hardness result of [UV10] does not hold if the mechanism is allowed to output some other data structure – such a mechanism can simply privately query each of the d^2 questions.

We circumvent the hardness result of [UV10] by outputting a data structure rather than a synthetic data set, and by releasing all conjunctions with small *average* error. Although there are no known computational lower bounds for releasing conjunctions with small average error, even for algorithms that output a data set, since our algorithm does not output a data set, our approach may be useful in circumventing the lower bounds of [UV10].

We also prove a new unconditional (information theoretic) lower bound on algorithms for privately releasing monotone conjunctions that applies to the class of algorithms that interact with the data using only SQ queries: no such polynomial time algorithm can release monotone conjunctions with $o(1)$ average error. We note that our lower bound does not depend on the output representation of the algorithm. Because almost all known private algorithms can indeed be implemented using statistical queries, this provides a new perspective on sources of hardness for private query release. We note that information theoretic lower bounds on the query complexity imply lower bounds on the running time of such differentially private algorithms.

There are also many lower bounds on the *error* that must be introduced by any private mechanism, independent of its running time. In particular, Kasiviswanathan et. al. [KRSU10] showed that average error of $\Omega(1/\sqrt{n})$ is necessary for private mechanisms that answer all conjunction queries of constant size. Recently, this work was extended by De [De11] to apply to mechanisms that are allowed to have arbitrarily large error on a constant fraction of conjunction queries of constant size. These results extend earlier results by Dinur and Nissim [DN03] showing that average error $\Omega(1/\sqrt{n})$ is necessary for random queries.

Interactive private query release mechanisms. Recently, Roth and Roughgarden [RR10] and Hardt and Rothblum [HR10] gave interactive private query release mechanisms that allow a data analyst to ask a large

number of questions, while only expending their privacy budgets slowly. Their privacy analyses depend on the fact that only a small fraction of the queries asked necessitate updating the internal state of the algorithm. However, to answer large classes of queries, these algorithms need to make a large number of statistical queries to the database, even though only a small number of statistical queries result in update steps! Intuitively, our characterization of the query complexity of the release problem in the SQ model is based on two observations: first, that it would be possible to implement these interactive mechanisms using only a small number of statistical queries if the data analyst was able to ask only those queries that would result in update steps, and second, that finding queries that induce large update steps is exactly the problem of agnostic learning.

2 Preliminaries

Differential privacy and counting queries. We study the question of answering *counting queries* over a database while preserving differential privacy. Given an arbitrary domain X, we consider databases $D \in X^n$. We write $n = |D|$. Two databases $D = (x_1, \ldots, x_n)$ and $D' = (x'_1, \ldots, x'_n)$ are called *adjacent* if they differ only in one entry. That is, there exists $i \in [n]$ such that for every $j \neq i$, $x_j = x'_j$. We are interested in algorithms (or *mechanisms*) that map databases to some abstract range $\mathcal{R}$ while satisfying ε-differential privacy:

Definition 2.1 (Differential Privacy [DMNS06])**.** A mechanism $\mathcal{M} : X^* \to \mathcal{R}$ satisfies ε-differential privacy if for all $S \subset \mathcal{R}$ and every pair of two adjacent databases D, D', we have $\Pr(\mathcal{M}(D) \in S) \leq e^{\varepsilon} \Pr(\mathcal{M}(D') \in S)$.

A *counting query* is specified by a predicate $q\colon X \to [0,1]$. We will denote the answer to a counting query (with some abuse of notation) by $q(D) = \frac{1}{n}\sum_{x \in D} q(X)$. Note that a count query can differ by at most $1/n$ on any two adjacent databases. In particular, adding Laplacian noise of magnitude $1/\varepsilon n$, denoted $Lap(1/\varepsilon n)$, guarantees ε-differential privacy on a single count query (see [DMNS06] for details).

The statistical query model and its connection to differential privacy. We will state our algorithms in Kearns' statistical query (SQ) model. In this model an algorithm A^{O} can access a distribution D over a universe X only through *statistical queries* to an oracle O. That is, the algorithm may ask any query $q\colon X \to [0,1]$ and the oracle may respond with any answer a satisfying $|a - \mathbb{E}_{x \sim D}\, q(x)| \leq \tau$. Here, τ is a parameter called the *tolerance* of the query.

In the context of differential privacy, the distribution D will typically be the uniform distribution over a data set of size n. A statistical query is then just the same as a counting query as defined earlier. Since SQ algorithms are tolerant to noise it is not difficult to turn them into differentially private algorithms using a suitable oracle. This observation is not new, and has been used previously, for example by Blum et al. [BDMN05] and Kasiviswanathan et al. [KLN+08].

Proposition 2.1. *Let A denote an algorithm that requires k queries of tolerance τ. Let O denote the oracle that outputs $\mathbb{E}_{x \sim D}\, q(x) + Lap(k/n\varepsilon)$. Then, the algorithm A^{O} satisfies ε-differential privacy and with probability at least $1 - \beta$, the oracle answers all q queries with error at most τ provided that $n \geq \frac{k(\log k + \log(1/\beta))}{\varepsilon\tau}$.*

Proof. The first claim follows directly from the properties of the Laplacian mechanism and the composition property of ε-differential privacy. To argue the second claim note that $\Pr(|Lap(\sigma)| \geq \tau) \leq \exp(-\tau/\sigma)$. Using that $\sigma = k/n\varepsilon$ and the assumption on n, we get that this probability is less than β/k. The claim now follows by taking a union bound over all k queries. □

Query release. A *concept class* (or *query class*) is a set of predicates from $X \to [0,1]$.

Definition 2.2 (Query Release)**.** Let C be a concept class. We say that an algorithm A (α, β)-*releases* C over a data set D if $\Pr_{q \sim C}\{|q(D) - A(q)| \leq \alpha\} \geq 1 - \beta$.

Specifically, we are interested in algorithms which release C using few statistical queries to the underlying data set. We will study the query release problem by considering the function $f(q) = q(D)$. In this setting, releasing a concept class C is equivalent to *approximating* the function q is the following sense

Definition 2.3. We say that an algorithm A (α, β)-*approximates* a function $f\colon 2^U \to [0,1]$ over a distribution $\mathcal{D}$ if $\Pr_{S\sim\mathcal{D}}\{|f(S) - A(S)| \le \alpha\} \ge 1 - \beta$.

For many concept classes of interest, the function $f(q) = q(D)$ will be *submodular*, defined next.

Submodularity. Given a universe U, a function $f : 2^U \to \mathbb{R}$ is called *submodular* if for all $S, T \subset U$ it holds that $f(S \cup T) + f(S \cap T) \le f(S) + f(T)$. We define the *marginal value* of x (or *discrete derivative*) at S as $\partial_x f(S) = f(S \cup \{x\}) - f(S)$.

Fact 2.1. *A function f is submodular if and only if $\partial_x f(S) \ge \partial_x f(T)$ for all $S \subseteq T \subseteq U$ and all $x \in U$.*

Definition 2.4. A function $f : 2^U \to \mathbb{R}$ is γ-Lipschitz if for every $S \subseteq U$ and $x \in U$, $|\partial_x f(S)| \le \gamma$.

Concentration bounds for submodular functions. The next lemma was shown by Vondrak [Von10] building on concentration bounds for so-called self-bounding functions due to [BLM00, BLM09].

Lemma 2.1 (Concentration for submodular functions)**.** *Let $f\colon 2^U \to \mathbb{R}$ be a 1-Lipschitz submodular function. Then for any product distribution $\mathcal{D}$ over 2^U, we have*

$$\Pr_{S\sim\mathcal{D}}\{|f(S) - \mathbb{E}\, f(S)| \ge t\} \le 2\exp\left(-\frac{t^2}{2(\mathbb{E}\, f(S) + 5t/6)}\right), \tag{1}$$

where the expectations are taken over $S \sim \mathcal{D}$.

We obtain as a simple corollary

Corollary 2.2. *Let $f\colon 2^U \to [0,1]$ be a γ-Lipschitz submodular function. Then for any product distribution $\mathcal{D}$ over 2^U, we have*

$$\Pr_{S\sim\mathcal{D}}\{|f(S) - \mathbb{E}\, f(S)| \ge \gamma t\} \le 2\exp\left(-\frac{t^2}{2(1/\gamma + 5t/6)}\right), \tag{2}$$

where the expectations are taken over $S \sim \mathcal{D}$.

3 Approximating Submodular Functions

Our algorithm for approximating submodular functions is based on a structural theorem, together with some strong concentration inequalities for submodular functions (see Lemma 2.1). The structure theorem essentially says that we can decompose any bounded submodular function into a small collection of Lipschitz submodular functions, one for each region of the domain. In this section, we prove our structure theorem, present our algorithm, and prove its correctness.

Algorithm 1 Decomposition for monotone submodular functions

Input: Oracle access to a submodular function $f\colon 2^U \to [0,1]$ and parameter $\gamma > 0$.

Let $\prec$ denote an arbitrary ordering of U.
Let $\mathcal{I} \leftarrow \{\emptyset\}$
for $x \in U$ (in ascending order under $\prec$) **do**
  $\mathcal{I}' \leftarrow \emptyset$
  for $B \in \mathcal{I}$ **do**
    if $\partial_x f(B) > \gamma$ **then** $\mathcal{I}' \leftarrow \mathcal{I}' \cup \{B \cup \{x\}\}$
  $\mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{I}'$
Let $V(S) = \{x \in U \mid \partial_x f(S) \le \gamma\}$ denote the set of elements that have small marginal value with respect to $S \subseteq U$.

Output: the collection of functions $\mathcal{G} = \{g^B \mid B \in \mathcal{I}\}$, where for $B \in \mathcal{I}$ we define the function $g^B : 2^{V(B)} \to [0,1]$ as $g^B(S) = f(S \cup B)$.

3.1 Monotone Submodular Functions

We begin with a simpler version of the structure theorem. This version will be sufficient for approximating bounded monotone submodular functions from value queries, and will be the main building block in our stronger results, which will allow us to approximate arbitrary bounded submodular functions, even from "tolerant" value queries.

Our structure theorem follows from an algorithm that decomposes a given submodular function into Lipschitz submodular functions. The algorithm is presented next and analyzed in Lemma 3.1.

Lemma 3.1. *Given any submodular function $f\colon 2^U \to [0,1]$ and $\gamma > 0$, Algorithm 1 makes the following guarantee. There are maps $F, T\colon 2^U \to 2^U$ such that:*

1. (Lipschitz) *For every $g^B \in \mathcal{G}$, g^B is submodular and satisfies $\sup_{x \in V(B), S \subseteq V(B)} \partial_x g^B(S) \le \gamma$.*
2. (Completeness) *For every $S \subseteq U$, $F(S) \subseteq S \subseteq V(F(S))$ and $g^{F(S)}(S) = f(S)$.*
3. (Uniqueness) *For every $S \subseteq U$ and every $B \in \mathcal{I}$, we have $F(S) = B$ if and only if $B \subseteq S \subseteq V(B)$ and $S \cap T(B) = \emptyset$.*
4. (Size) *The size of $\mathcal{G}$ is at most $|\mathcal{G}| = |U|^{O(1/\gamma)}$. Moreover, given oracle access to f, we compute F, V, T in time $|U|^{O(1/\gamma)}$.*

Note that the lemma applies to non-monotone submodular functions f as well; however, since our release algorithm will require the stronger condition $\sup_{x \in V(B), S \subseteq V(B)} |\partial_x g(S)| \le \gamma$, the lemma will only be sufficient for releasing monotone submodular functions (where it holds that $|\partial_x g(S)| \le \gamma \iff \partial_x g(S) \le \gamma$). We will return to the non-monotone case later.

Proof. Algorithm 1 always terminates and we have the following bound on the size of $\mathcal{I}$.

Claim 3.2. $|\mathcal{I}| \le |U|^{1/\gamma}$

Proof. Let $B \in \mathcal{I}$ be a set, $B = \{x_1, \ldots, x_{|B|}\}$. Let $B_0 = \emptyset$ and $B_i = \{x_1, \ldots, x_i\}$ for $i = 1, \ldots, |B| - 1$. Then

$$1 \ge f(B) = \sum_{i=0}^{|B|-1} \partial_{x_{i+1}} f(B_i) > |B| \cdot \gamma\,. \tag{3}$$

Therefore, it must be that $|B| \le 1/\gamma$, and there are at most $|U|^{1/\gamma}$ such sets over $|U|$ elements. □

Item 1 is shown next.

Claim 3.3 (Lipschitz)**.** *For every $g^B \in \mathcal{G}$, g^B is submodular and* $\sup_{x \in V(B), S \subseteq V(B)} \partial_x g^B(S) \leq \gamma$.

Proof. Submodularity follows from the fact that g^B is a "shifted" version of f. Specifically, if $T \subseteq S$, then $\partial_x g^B(S) = \partial_x f(B \cup S) \leq \partial_x f(B \cup T) = \partial_x g^B(T)$, where the inequality is by submodularity of f.

To establish the Lipschitz property, we note that by the definition of V, $\partial_x f(B) \leq \gamma$ for every $x \in V(B)$. Also, by the submodularity of f, we have $\partial_x g^B(S) = \partial_x f(B \cup S) \leq \partial_x f(B) \leq \gamma$. □

Definition of F and proof of Item 2. Now we turn to constructing the promised mappings F and T in order to Properties 2 and 3. Roughly, we want $F(S)$ to choose a maximal set in $\mathcal{I}$ such that $F(S) \subseteq S$, in order to assure that $S \subseteq V(F(S))$. This task is complicated by the fact that there could be many such sets. We want to be able to choose a unique such set, and moreover, given any such set B, determine efficiently if $F(S) = B$. To achieve the former task, we define a specific, deterministic mapping $F(S)$ and to achieve the latter we will carefully define the mapping T.

We define $F(S)$ as follows:

let $j \leftarrow 0$, $B_j \leftarrow \emptyset$
for $x \in U$ (in ascending order under $\prec$) **do**
  if $x \notin V(B_j)$ and $x \in S$ **then** $B_{j+1} \leftarrow B_j \cup \{x\}$, $j \leftarrow j + 1$
return $F(S) = B_j$.

Note that this procedure is similar to the procedure we use to construct $\mathcal{I}$. To construct $\mathcal{I}$, we gradually constructed a tree of sets, where each set $B \in \mathcal{I}$ had a child for every set $B \cup \{x\}$ such that x has high influence on B ($x \notin V(B)$.. The procedure $F(S)$ differs in that it only constructs a single root-leaf path in this tree, where for each B_j in the path, the next set in the path is $B_j \cup \{x\}$ where x is the *minimal* $x \in S$ that has high influence on B_j (and has not already been considered by $F(S)$. We will use $P(S) = (B_0 \subset B_1 \subset \cdots \subset F(S))$ to denote this path, which is the sequence of intermediate sets B_j in the execution of $F(S)$. Given these observations, we can state the following useful facts about F.

Fact 3.1. *If $F(S) = B$, then $P(S) = P(B)$. Moreover, for every $S \in U$, $P(S) \subseteq \mathcal{I}$.*

We can now establish Property 2 by the following claim.

Claim 3.4 (Completeness)**.** *For every $S \subseteq U$, $F(S) \subseteq S \subseteq V(F(S))$, and $g^{F(S)}(S) = f(S)$.*

Proof. Let $P(S) = B_0 \subset B_1 \subset \cdots \subset F(S)$. $F(S)$ always checks that $x \in S$ before including an element x, so $F(S) \subseteq S$. To see that $S \subseteq V(F(S))$, assume there exists $x \in S \setminus V(F(S))$. By submodularity we have $\partial_x f(B_j) \geq \partial_x f(F(S)) > \gamma$ for every set B_j. But if $\partial_x f(B_j) > \gamma$ for every B_j and $x \in S$, it must be that $x \in F(S)$. But then $\partial_x f(F(S)) = 0$, contradicting the fact that $x \notin V(F(S))$.

Finally, we note that since $S \subseteq V(F(S))$, $g^{F(S)}(S)$ is defined (S is in the domain of $g^{F(S)}$) and since $F(S) \subseteq S$, $g^{F(S)}(S) = f(F(S) \cup S) = f(S)$. □

Definition of T and proof of Item 3. We will now define the mapping T. The idea is to consider a set $B \in \mathcal{I}$ and $P(B)$ and consider all the elements we had to "reject" on the way from the root to B. We say that an element $x \in U$ is "rejected" if, when x is considered by $F(S)$, it has high influence on the current set, but is not in B. Since any set S such that $B = F(S)$ satisfies $P(S) = P(B)$ (Fact 3.1), and any set S that contains a rejected element would have taken a different path, we will get that the elements $x \in T(B)$ "witness" the fact that $B \neq F(S)$. We define the map $T(B)$ as follows:

let $j \leftarrow 0$, $B_j \leftarrow \emptyset$, $R \leftarrow \emptyset$
for $x \in U$ (in ascending order under $\prec$) **do**
  if $x \notin V(B_j)$ and $x \notin B$ **then** $R \leftarrow R \cup \{x\}$

else if $x \notin V_{B_j}$ and $x \in B$ **then** $B_{j+1} \leftarrow B_j \cup \{x\}$, $j \leftarrow j+1$

return $T(B) = R$.

We'll establish Property 3 via the following two claims.

Claim 3.5. *If $B = F(S)$, then $B \subseteq S \subseteq V(B)$ and $S \cap T(B) = \emptyset$.*

Proof. We have already demonstrated the first part of the claim in Claim 3.4, so we focus on the claim that $S \cap T(B) = \emptyset$. By Fact 3.1, every set S s.t. $B = F(S)$ satisfies $P(S) = P(B)$. Let $(B_0 \subset B_1 \subset \cdots \subset B) = P(B)$. Suppose there is an element $x \in S \cap T(B)$. Then there is a set B_j such that $x \notin V(B_j)$ and $x \notin B$. But since $x \notin V(B_j)$ and $x \in S$, it must be that $x \in B_{j+1}$, contradicting the fact that $B_{j+1} \subseteq B$. □

Now we establish the converse.

Claim 3.6. *If $B \subseteq S \subseteq V(B)$, $S \cap T(B) = \emptyset$, then $B = F(S)$.*

Proof. Suppose for the sake of contradiction that there a set $B' \neq B$ such that $B' = F(S)$. There exists an element $x \in B \triangle B'$, and we consider the minimal such x under $\prec$. Let $P(B) = (B_0 \subset B_1 \subset \cdots \subset B)$ and $P(S) = P(B') = (B'_0 \subset B'_1 \subset \cdots \subset B')$. Since x is minimal in $B \triangle B'$, there must be j be such that $B_i = B'_i$ for all $i \leq j$, but $x \in B_{j+1} \triangle B'_{j+1}$. Consider two cases:

1. $B \supset B'$. Thus $x \in B \setminus B'$ Moreover, since $x \in B \subseteq S$, it must be that when x was considered in the execution of $F(S)$, and B'_j was the current set, it was the case that $x \in V(B'_j)$. But $B_j = B'_j$, so $x \in V(B_j)$, contradicting the fact that $x \in B_{j+1}$.

2. $B \not\supset B'$. Thus $x \in B' \setminus B$. Since $x \in B' = F(S) \subseteq S$ (Claim 3.4), we have $x \in S$. Moreover, since $x \in B'_{j+1}$ we must have $x \notin V(B'_j) = V(B_j)$. Thus we have $x \notin V(B_j)$ and $x \notin B$, which implies $x \in T(B)$, by construction. Thus $S \cap T(B) \neq \emptyset$, a contradiction.

□

The previous two claims establish Item 3.

Finally we observe that the enumeration of $\mathcal{I}$ requires time at most $|U| \cdot |\mathcal{I}| = |U|^{O(1/\gamma)}$, since we iterate over each element of U and then iterate over each set currently in $\mathcal{I}$. We also note that we can compute the mappings F and T in time linear in $|\mathcal{I}| = |U|^{O(1/\gamma)}$ and can compute $V(B)$ in time linear in $|U|$. These observations establish Property 4 and complete the proof of Lemma 3.1. □

Lemma 3.7 (Lemma 3.1 with tolerance)**.** *Given any submodular function $f\colon 2^U \to [0,1]$ and $\gamma > 0$, Algorithm 2 makes the following guarantee. There are maps $F, T\colon 2^U \to 2^U$ satisfying properties 1-4 of Lemma 3.1 and moreover, can be computed using tolerant queries to f with tolerance $\gamma/12$.*

Proof. Throughout the proof, we will assume that the oracle always gives the same answer to each query. Thus the function $\tilde{f}$ defined in Algorithm 2 is well defined. Note that $\tilde{f}(S)$ need not be submodular even if f is, however, we can assume that we have exact oracle access to $\tilde{f}(S)$. Also note that, since we can compute $\partial_x f(S)$ using two queries to f, we are guaranteed that for every $S \subseteq U$, and $x \in U$,

$$|\partial_x \tilde{f}(S) - \partial_x f(S)| \leq \gamma/6. \tag{4}$$

Observe that Algorithm 2 differs from Algorithm 1 only in the choice of parameters. The analysis required to establish the Lemma is also a natural modification of the analysis of Lemma 3.1, so we will refer the reader to the proof of that Lemma for several details and only call attention to the steps of the proof that require modification.

Algorithm 2 Decomposition for monotone submodular functions from tolerant queries

Input: Tolerant oracle access to a submodular function $f\colon 2^U \to [0,1]$ with tolerance at most $\gamma/12$ and parameter $\gamma > 0$.

Let $\tilde{f}$ denote the function specified by tolerant oracle queries to f such that for every $S \subseteq U$

$$|f(S) - \tilde{f}(S)| \leq \gamma/12.$$

Let $\prec$ denote an arbitrary ordering of U.
Let $\mathcal{I} \leftarrow \{\emptyset\}$
for $x \in U$ (in ascending order under $\prec$) **do**
  $\mathcal{I}' \leftarrow \emptyset$
  for $B \in \mathcal{I}$ **do**
    if $\partial_x \tilde{f}(B) > \gamma/3$ **then** $\mathcal{I}' \leftarrow \mathcal{I}' \cup \{B \cup \{x\}\}$
  $\mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{I}'$
Let $V(S) = \{x \in U \mid \partial_x \tilde{f}(S) \leq 2\gamma/3\}$ denote the set of elements that have small marginal value with respect to $S \subseteq U$.

Output: the collection of functions $\mathcal{G} = \{g^B \mid B \in \mathcal{I}\}$, where for $B \in \mathcal{I}$ we define the function $g^B : 2^{V(B)} \to [0,1]$ as $g^B(S) = f(S \cup B)$.

We will proceed by running through the construction of Lemma 3.1 on $\tilde{f}(S)$ using $\gamma/3$ as the error parameter. Since the argument is a fairly straightforward modification to Lemma 3.1, we will refer the reader to the proof of that Lemma for several details, and only call attention to the steps of the proof that require modification.

First, we establish a bound on the size of $\mathcal{I}$

Claim 3.8. $|\mathcal{I}| \leq |U|^{6/\gamma}$

Proof. Let $B \in \mathcal{I}$ be a set, $B = \{x_1, \ldots, x_{|B|}\}$. Let $B_0 = \emptyset$ and $B_i = \{x_1, \ldots, x_i\}$ for $i = 1, \ldots, |B| - 1$. Then

$$1 \geq f(B) = \sum_{i=0}^{|B|-1} \partial_{x_{i+1}} f(B_i) \geq \sum_{i=0}^{|B|-1} \left(\partial_{x_{i+1}} \tilde{f}(B_i) - \gamma/6\right) > |B| \cdot (\gamma/3 - \gamma/6) = |B| \cdot \gamma/6\,. \tag{5}$$

Therefore, it must be that $|B| \leq 6/\gamma$, and there are at most $|U|^{6/\gamma}$ such sets over $|U|$ elements. □

Item 1 is shown next.

Claim 3.9 (Lipschitz)**.** *For every $g^B \in \mathcal{G}$, g^B is submodular and* $\sup_{x \in V(B), S \subseteq V(B)} \partial_x g^B(S) \leq \gamma$.

Proof. The proof of submodularity is identical to Claim 3.3

To establish the Lipschitz property, observe that for every $B \subseteq U$, and every $x \in V(B)$, $\partial_x f(B) \leq \partial_x \tilde{f}(B) + \gamma/6 \leq \gamma$. □

Definition of F and proof of Item 2. In addition to the sets $V(B) = \{x \in U \mid \partial_x \tilde{f}(B) \leq 2\gamma/3\}$, we will define the sets $V'(B) = \{x \in U \mid \partial_x \tilde{f}(B) \leq \gamma/3\}$, note that for every $B \subseteq U$, $V'(B) \subseteq V(B)$. We define the promised mapping $F(S)$ in the the same manner as in the proof of Lemma 3.1, but we use V' in place of V to decide whether or not we select an element x for inclusion in the set $F(S)$.

Now we establish Property 2 via the following claim, analogous to Claim 3.4 in the proof of Lemma 3.1

Claim 3.10 (Completeness)**.** *For every $S \subseteq U$, $F(S) \subseteq S \subseteq V(F(S))$. Moreover, $g^{F(S)}(S) = f(S)$.*

Proof. Let $P(S) = B_0 \subset B_1 \subset \cdots \subset F(S)$. The fact that $F(S) \subseteq S$ follows as in Claim 3.4. To see that $S \subseteq V(F(S))$, assume there exists $x \in S \setminus V(F(S))$. By submodularity of f, and (4), we have

$$\partial_x \tilde{f}(B_j) \geq \partial_x f(B_j) - \gamma/6 \geq \partial_x f(F(S)) - \gamma/6 > \partial_x \tilde{f}(F(S)) - \gamma/3 > \gamma/3.$$

Thus, $\partial_x \tilde{f}(B_j) > \gamma/3$ for every set B_j. But if $\partial_x f(B_j) > \gamma/3$ for every B_j and $x \in S$, then $x \notin V'(B_j)$ for every B_j, and it must be that $x \in F(S)$. But then $\partial_x f(F(S)) = 0$, contradicting the fact that $x \notin V(F(S))$.

The fact that $g^{F(S)}(S) = f(S)$ follows as in the proof of Claim 3.4. □

Definition of T and proof of Item 3. We also define the promised mapping $T(S)$ in the same manner as in the proof of Lemma 3.1, but using V' in place of V to decide whether or not we select an element x for inclusion in the set $F(S)$.

To establish Property 3, we note that the proofs of Claims 3.5 and 3.6 do not rely on the submodularity of f, therefore they apply as-is to the case where we compute on $\tilde{f}$, even though $\tilde{f}$ is not necessarily submodular.

Property 4 also follows as in the proof of Lemma 3.1. This completes the proof of the Lemma. □

We now present our algorithm for learning monotone submodular functions over product distributions. For a subset of the universe $V \subseteq U$, let $\mathcal{D}_V$ denote the distribution $\mathcal{D}$ restricted to the variables in V. Note that if $\mathcal{D}$ is a product distribution, then $\mathcal{D}_V$ remains a product distribution and is easy to sample from.

Algorithm 3 Approximating a monotone submodular function from tolerant queries

Learn$(f, \alpha, \beta, \mathcal{D})$

Let $\gamma = \frac{\alpha^2}{6\log(2/\beta)}$.

Construct the collection of functions $\mathcal{G}$ returned by Algorithm 2 and let F, V, T be the associated mappings given by Lemma 3.7 with parameter γ.

Estimate the value $\mu_{g^B} = \mathbb{E}_{S \sim \mathcal{D}_{V(B)\setminus T(B)}}[g^B(S)]$ for each $g^B \in \mathcal{G}$.

Output the data structure h that consists of the values μ_{g^B} for every $g^B \in \mathcal{G}$ as well as the mapping F.

Theorem 3.11. *For any $\alpha, \beta \in (0,1]$, Algorithm 5 (α,β)-approximates any submodular function $f\colon 2^U \to [0,1]$ under any product distribution $\mathcal{D}$ in time $|U|^{O(\alpha^{-2}\log(1/\beta))}$ using oracle queries to f of tolerance $\alpha^2/72\log(2/\beta)$.*

Proof. For a set $S \subseteq U$, we let $B = F(S)$ and g^B be the corresponding submodular function as in Lemma 3.7. Note that since the queries have tolerance $\alpha^2/72\log(1/\beta) \leq \gamma/12$, the lemma applies. We will analyze the error probability as if the estimates μ_{g^B} were computed using exact oracle queries to f, and will note that using tolerant queries to f can only introduce an additional error of $\alpha^2/72\log(1/\beta) \leq \alpha/6$. We claim that, under this condition

$$\begin{aligned}\Pr_{S\sim\mathcal{D}}\{|f(S) - h(S)| > 5\alpha/6\} &= \Pr_{S\sim\mathcal{D}}\left\{\left|g^{F(S)}(S) - \mu_{g^{F(S)}}\right| > 5\alpha/6\right\} \\ &= \sum_{g^B \in \mathcal{G}} \Pr_{S\sim\mathcal{D}}\{B = F(S)\} \cdot \Pr_{S\sim\mathcal{D}}\left\{\left|g^B(S) - \mu_{g^B}\right| > 5\alpha/6 \mid B = F(S)\right\}. \qquad (6)\end{aligned}$$

To see this, recall that for every $S \subseteq U$, $g^{F(S)}(S) = f(S)$. By Property 3 of Lemma 3.1, the condition that $B = F(S)$ is equivalent to the conditions that $B \subseteq S \subseteq V(B)$ and $S \cap T(B) = \emptyset$. Hence,

$$\Pr_{S\sim\mathcal{D}}\left\{\left|g^B(S) - \mu_{g^B}\right| > 5\alpha/6 \mid B = F(S)\right\} = \Pr_{S\sim\mathcal{D}_{V(B)\setminus T(B)}}\left\{\left|g^B(S) - \mu_{g^B}\right| > 5\alpha/6\right\}.$$

Now, applying the concentration inequality for submodular functions stated as Corollary 2.2, we get

$$\Pr_{S\sim\mathcal{D}_{V_B\setminus T_B}}\left\{\left|g^B(S)-\mu_{g^B}\right|\geq\gamma t\right\}\leq 2\exp\left(-\frac{t^2}{2(1/\gamma+5/6t)}\right). \tag{7}$$

Plugging in $t=5\alpha/6\gamma=\frac{5\log(2/\beta)}{\alpha}$ and simplifying we get $\Pr_{S\sim\mathcal{D}_{V(B)\setminus T(B)}}\left\{\left|g^B(S)-\mu_{g^B}\right|>\alpha\right\}\leq\beta$. Combining this with (6), the claim follows. □

3.2 Non-monotone Submodular Functions

For non-monotone functions, we need a more refined argument. Our main structure theorem replaces Property 1 in Lemma 3.1 by the stronger guarantee that $|\partial_x g(S)|\leq\alpha$ for all $g\in\mathcal{G}$, even for non-monotone submodular functions. Observe that for a submodular function $f:2^V\to\mathbb{R}$, the function $\overline{f}\colon 2^V\to\mathbb{R}$ defined as $\overline{f}(S)=f(V\setminus S)$ is also submodular; moreover

$$\inf_{x\in V,S\subseteq V}\partial_x\overline{f}(S)=-\sup_{x\in V,S\subseteq V}\partial_x f(S). \tag{8}$$

Given these two facts, we can now prove our main structure theorem.

Algorithm 4 Decomposition for submodular functions from tolerant queries

Input: Tolerant oracle access to a submodular function $f\colon 2^U\to[0,1]$ with tolerance at most $\gamma/12$ and parameter $\gamma>0$.

Let $\tilde{f}$ denote the function specified by tolerant oracle queries to f such that for every $S\subseteq U$

$$|f(S)-\tilde{f}(S)|\leq\gamma/12.$$

Let $\prec$ denote an arbitrary ordering of U.
Let $\mathcal{G}(f)$ denote the collection of functions returned by Algorithm 2 with oracle f and parameter γ, and let F_f,V_f,T_f be the associated mappings promised by Lemma 3.7.
for $g^B\in\mathcal{G}(f)$ **do**
  Let $\mathcal{G}(B)$ be the collection of functions returned by Algorithm 2 with oracle $\overline{g}^B$ and parameter γ, and let F_B,V_B,T_B be the associated mappings promised by Lemma 3.7.
Let $V(S,T)=V_f(S)\cap V_S(T)$ denote the set of elements that have small marginal absolute value with respect to $S,T\subseteq U$.

Output: the collection of functions $\mathcal{G}=\bigcup_{g^B\in\mathcal{G}(f)}\{g^{B,C}=\overline{g}^C\mid g^C\in\mathcal{G}(B)\}$ where $g^{B,C}\colon 2^{V(B,C)}\to[0,1]$.

Theorem 3.12. *Given any submodular function $f\colon 2^U\to[0,1]$ and $\gamma>0$, Algorithm 4 makes the following guarantee. There are maps $F\colon 2^U\to 2^U\times 2^U$ and $T\colon 2^U\times 2^U\to 2^U$ such that:*

1. (Lipschitz) *For every $g^{B,C}\in\mathcal{G}$, $g^{B,C}$ is submodular and satisfies $\sup_{x\in V(B,C),S\subseteq V(B,C)}|\partial_x g^{B,C}(S)|\leq\gamma$.*
2. (Completeness) *For every $S\subseteq U$, $F(S)\subseteq S\subseteq V(F(S))$ and $g^{F(S)}(S)=f(S)$.*
3. (Uniqueness) *For every $g^{B,C}\in\mathcal{G}$, $F(S)=(B,C)$ if and only if $B,C\subseteq S\subseteq V(B,C)$ and $S\cap T(B,C)=\emptyset$.*
4. (Size) *The size of $\mathcal{G}$ is at most $|\mathcal{G}|=|U|^{O(1/\gamma)}$. Moreover, given tolerant oracle access to f with tolerance $\gamma/12$, we compute F,V,T in time $|U|^{O(1/\gamma)}$.*

Proof. First we show Item 1

Claim 3.13 (Lipschitz)**.** *For every $g^{B,C} \in \mathcal{G}$, $g^{B,C}$ is submodular and* $\sup_{x \in V(B,C), S \subseteq V(B,C)} |\partial_x g^{B,C}(S)| \leq \gamma$.

Proof. Submodularity follows directly from Property 1 of Lemma 3.7. The same property of the lemma guarantees that for every $g^B \in \mathcal{G}(f)$ and $g^C \in \mathcal{G}(B)$, $\sup_{x \in V_B(C), S \subseteq V_B(C)} \partial_x g^C(S) \leq \gamma$. Moreover, by (8), $\inf_{x \in V_f(B), S \subseteq V_f(B)} \partial_x \overline{g}^B(S) \geq -\gamma$. Taken together, we obtain $\sup_{x \in V(B,C), S \subseteq V(B,C)} |\partial_x g^{B,C}(S)| \leq \gamma$. □

Definition of F and proof of Item 2. Item 2 will follow from the analogous property in Lemma 3.7 almost directly. To construct the mapping $F(S)$, we want to first compute the appropriate function $g^B \in \mathcal{G}(f)$, using $F_f(S)$ and then find the appropriate function $g^C \in \mathcal{G}(B)$ using $F_B(S)$. Thus we can take $F(S) = (F_f(S), F_{F_f(S)}(S))$. By Lemma 3.7, Item 2 we have $B \subseteq S \subseteq V_f(B)$ and $C \subseteq S \subseteq V_B(C)$, so we conclude $B, C \subseteq S \subseteq V(B,C)$.

Definition of T and proof of Item 3. Item **??** will also follow from the analogous property in Lemma 3.7. By Lemma 3.7, Item 3, we have that $F_f(S) = B$ if and only if $B \subseteq S \subseteq V_f(B)$ and $S \cap T_f(B) = \emptyset$. By the same Lemma, we also have that $F_B(S) = C$ if and only if $C \subseteq S \subseteq V_B(C)$ and $S \cap T_B(C) = \emptyset$. So if we define $T(B,C) = T_f(B) \cup T_B(C)$, we can conclude that $F(S) = (B,C)$ if and only if $B, C \subseteq S \subseteq V(B,C)$ and $S \cap T(B,C) = \emptyset$.

Now it is clear that $F(S) = (B,C)$ if $F_f(S) = B$ and $F_B(S) = C$, which by Property 3 of Lemma 3.7 necessitates that $B \subseteq S \subseteq V_f(B)$, $S \cap T_f(B) = \emptyset$, $C \subseteq S \subseteq V_B(S)$, and $S \cap T_B(C) = \emptyset$. We have already defined $V(B,C)$ and now we define $T(B,C) = T_f(B) \cup T_B(C)$. It is clear now that $F(S) = (B,C)$ if and only if $B, C \subseteq S \subseteq V(B,C)$ and $S \cap T(B,C) = \emptyset$.

The size of $\mathcal{G}$ and running time bounds in Property **??** also follow directly from the analogous property of Lemma 3.1. The fact that we can compute the family $\mathcal{G}$ and the associated mappings F, V, T using oracle access to f with tolerance $\gamma/12$ follows from the fact that each invocation of Lemma 3.1 can be computed using queries with tolerance $\gamma/12$ and from the fact that Algorithm 4 only queries f in order to invoke Lemma 3.7. This completes the proof of the Theorem. □

We now present our algorithm for learning arbitrary submodular functions over product distributions. For a subset of the universe $V \subseteq C$, let $\mathcal{D}_V$ denote the distribution $\mathcal{D}$ restricted to the variables in V. Note that if $\mathcal{D}$ is a product distribution, then $\mathcal{D}_V$ remains a product distribution and is easy to sample from. To

Algorithm 5 Approximating a non-monotone submodular function

Learn$(f, \alpha, \beta, \mathcal{D})$

Let $\gamma = \frac{\alpha^2}{6 \log(2/\beta)}$.

Construct the collection of functions $\mathcal{G}$ and the associated mappings F, V, T given by Theorem 3.12 with parameter γ.

Estimate the value $\mu_{g^{B,C}} = \mathbb{E}_{S \sim \mathcal{D}_{V(B,C) \setminus T(B,C)}}[g^{B,C}(S)]$ for each $g^{B,C} \in \mathcal{G}$.

Output the data structure h that consists of the values $\mu_{g^{B,C}}$ for every $g^{B,C} \in \mathcal{G}$ as well as the mapping F.

avoid notational clutter, throughout this section we will not consider the details of how we construct our estimate μ_g. However, it is an easy observation that this quantity can be estimated to a sufficiently high degree of accuracy using a small number of random samples.

Theorem 3.14. *For any $\alpha, \beta \in (0,1]$, Algorithm 5 (α, β)-approximates any submodular function $f : 2^U \to [0,1]$ under any product distribution in time $|U|^{O(\alpha^{-2} \log(1/\beta))}$ using oracle queries to f of tolerance $\alpha^2/72 \log(1/\beta)$*

Proof. For a set $S \subseteq U$, we let $(B,C) = F(S)$ and $g^{B,C}$ be the corresponding submodular function as in Theorem 3.12. Note that since the queries have tolerance $\alpha^2/72 \log(1/\beta) \leq \gamma/12$, the lemma applies. We

will analyze the error probability as if the estimates μ_{g^B} were computed using exact oracle queries to f, and will note that using tolerant queries to f can only introduce an additional error of $\alpha^2/72\log(1/\beta) \leq \alpha/6$. We claim that, under this condition We claim that

$$\begin{aligned}\Pr_{S\sim\mathcal{D}}\{|f(S)-h(S)|>5\alpha/6\} &= \Pr_{S\sim\mathcal{D}}\left\{\left|g^{F(S)}(S)-\mu_{g^{F(S)}}\right|>5\alpha/6\right\}\\ &= \sum_{g^{B,C}\in\mathcal{G}}\Pr_{S\sim\mathcal{D}}\{(B,C)=F(S)\}\cdot\Pr_{S\sim\mathcal{D}}\left\{\left|g^{B,C}(S)-\mu_{g^{B,C}}\right|>5\alpha/6\mid(B,C)=F(S)\right\}.\end{aligned} \tag{9}$$

To see this, recall that for every $S \subseteq U$, $g^{F(S)}(S) = f(S)$. By Property 3 of Lemma 3.1, the condition that $B = F(S)$ is equivalent to the conditions that $B, C \subseteq S \subseteq V(B,C)$ and $S \cap T(B,C) = \emptyset$. Hence,

$$\Pr_{S\sim\mathcal{D}}\left\{\left|g^{B,C}(S)-\mu_{g^{B,C}}\right|>5\alpha/6\mid(B,C)=F(S)\right\} = \Pr_{S\sim\mathcal{D}_{V(B,C)\setminus T(B,C)}}\left\{\left|g^{B}(S)-\mu_{g^{B}}\right|>5\alpha/6\right\}.$$

Now, applying the concentration inequality for submodular functions stated as Corollary 2.2, we get

$$\Pr_{S\sim\mathcal{D}_{V(B,C)\setminus T(B,C)}}\left\{\left|g^{B,C}(S)-\mu_{g^{B,C}}\right|\geq\gamma t\right\}\leq 2\exp\left(-\frac{t^2}{2(1/\gamma+5t/6)}\right). \tag{10}$$

Plugging in $t = 5\alpha/6\gamma = \frac{5\log(2/\beta)}{\alpha}$ and simplifying we get $\Pr_{S\sim\mathcal{D}_{V(B,C)\setminus T(B,C)}}\left\{\left|g^{B,C}(S)-\mu_{g^{B,C}}\right|>\alpha\right\} \leq \beta$. Combining this with Equation (9), the claim follows. □

4 Applications to privacy-preserving query release

In this section, we show how to apply our algorithm from Section 3 to the problem of releasing monotone conjunctions over a boolean database. In Section 4.1, we also show how our mechanism can be applied to release the *cut function* of an arbitrary graph.

Let us now begin with the monotone disjunctions. We will then extend the result to monotone conjunctions. Given our previous results, we only need to argue that monotone disjunctions can be described by a submodular function. Indeed, every element $S \in \{0,1\}^d$ naturally corresponds to a monotone Boolean disjunction $d_S : \{0,1\}^d \to \{0,1\}$ by putting

$$d_S(x) \stackrel{\text{def}}{=} \bigvee_{i:\, S(i)=1} x_i\,.$$

Note that in contrast to Section 3 here we use x to denote an element of $\{0,1\}^d$. Let $F_{\text{Disj}} : \{0,1\}^d \to [0,1]$ be the function such that $F_{\text{Disj}}(S) = d_S(D)$. It is easy to show that $F_{\text{Disj}}(S)$ is a monotone submodular function.

Lemma 4.1. *F_{Disj} is a monotone submodular function.*

Proof. Let X_i^+ denote the set of elements $x \in D$ such that $x_i = 1$, and let X_i^- denote the set of elements $x \in D$ such that $x_i = 0$. Consider the set system $U = \{X_i^+, X_i^-\}_{i=1}^d$ over the universe of elements $x \in D$. Then there is a natural bijection between $F_{\text{Disj}}(D)$ and the set coverage function $\text{Cov} : 2^U \to [0, |D|]$ defined to be $\text{Cov}(S) = |\bigcup_{X\in U} X|$, which is a monotone submodular function. □

We therefore obtain the following corollary directly by combining Theorem 3.11 with Proposition 2.1.

Corollary 4.2. *Let $\alpha, \beta, \varepsilon > 0$. There is an ε-differentially private algorithm that (α,β)-releases the set of monotone Boolean disjunctions over any product distribution in time $d^{t(\alpha,\beta)}$ for any data set of size $|D| \geq d^{t(\alpha,\beta)}/\varepsilon$ where $t(\alpha,\beta) = O(\alpha^{-2}\log(1/\beta))$.*

Algorithm 6 Privately Releasing Monotone Disjunctions

Release$(D, \alpha, \beta, \varepsilon, \mathcal{D})$

Simulate the oracle queries $F_{Disj}(S)$ by answering with $d_S(D) + Lap(t(\alpha,\beta)/\varepsilon|D|)$.
Let $\gamma = \frac{\alpha^2}{6\log(2/\beta)}$.
Construct the collection of functions $\mathcal{G}$ and the associated mappings F, V, T given by Lemma 3.7 on the function F_{Disj} with parameter γ.
Estimate the value $\mu_{g^B} = \mathbb{E}_{S\sim\mathcal{D}_{V(B)\setminus T(B)}}[g^B(S)]$ for each $g^B \in \mathcal{G}$.
Output the data structure h that consists of the estimated values μ_{g^B} for every $g^B \in \mathcal{G}$ as well as the mapping F. To evaluate any monotone disjunction query $d_S(D)$, compute $\mu_{g^{F(S)}}$.

For completeness, we will present the algorithm for privately releasing monotone disjunctions over a product distribution $\mathcal{D}$ for a data set D, though we will rely on Corollary 4.2 for the formal analysis.

We will next see that this corollary directly transfers to monotone conjunctions. A monotone Boolean conjunction $c_S : \{0,1\}^d \to \{0,1\}$ is defined as

$$c_S(x) \stackrel{\text{def}}{=} \bigwedge_{i\in S} x_i = 1 - \bigvee_{i\in S}(1-x_i)\,.$$

Given the last equation, it is clear that in order to release conjunctions over some distribution, it is sufficient to release disjunctions over the same distribution after replacing every data item $x \in D$ by its negation $\bar{x}$, i.e., $\bar{x_i} = 1 - x_i$. Hence, Corollary 4.2 extends directly to monotone conjunctions.

Extension to width w. Note that the uniform distribution on disjunctions of width w is not a product distribution, which is what we require to apply Theorem 3.14 directly. However, in Lemma 4.3 we show that for monotone submodular functions (such as F^D_{Disj}) the concentration of measure property required in the proof Theorem 3.14 is still satisfied. Of course, we can instantiate the theorem for every $w \in \{1,\ldots,k\}$ to obtain a statement for conjunctions of any width.

Indeed, given a monotone submodular function $f\colon 2^U \to \mathbb{R}$, let $S \in 2^U$ be the random variable where for every $x \in U$, independently $x \in S$ with probability w/d and $x \notin S$ with probability $1 - w/d$. On the other hand, let $T \in 2^U$ denote the uniform distribution over strings in 2^U of weight w. The following lemma is due to Balcan and Harvey [BH10].

Lemma 4.3. *Assume $f : 2^U \to \mathbb{R}$ is monotone function, and S and T are chosen at random as above. Then,*

$$\Pr[f(T) \geq \tau] \leq 2\Pr[f(S) \geq \tau] \tag{11}$$

$$\Pr[f(T) \leq \tau] \leq 2\Pr[f(S) \leq \tau] \tag{12}$$

Remark 4.1. *Throughout this section we focus on the case of* monotone *disjunctions and conjunctions. Our algorithm can be extended to non-monotone conjunctions/disjunctions as well. However, this turns out to be less interesting than the monotone case. Indeed, a random non-monotone conjunction of width w is false on any fixed data item with probability 2^{-w}, thus when $w \geq \log(1/\alpha)$, the constant function 0 is a good approximation to F_{Disj} on a random non-monotone conjunction of width w. We therefore omit the non-monotone case from our presentation.*

4.1 Releasing the cut function of a graph

Consider a graph $G = (V, E)$ in which the edge-set represents the private database (We assume here that each individual is associated with a single edge in G. The following discussion generalizes to the case in which

individuals may be associated with multiple edges, with a corresponding increase in sensitivity). The *cut function* associated with G is $f_G : 2^V \to [0,1]$, defined as:

$$f_G(S) = \frac{1}{|V|^2} \cdot |\{(u,v) \in E : u \in S, v \notin S\}|$$

We observe that the graph cut function encodes a collection of counting queries over the database E and so has sensitivity $1/|V|^2$.

Fact 4.1. *For any graph G, f_G is submodular.*

Lemma 4.4. *The decomposition from Theorem 3.12 constructs a collection of functions $\mathcal{G}$ of size $|\mathcal{G}| \leq 2^{2/\alpha}$.*

Proof. Let $u \in V$, and $S \subset V$ such that $|\partial_u f_G(S)| \geq \alpha$. It must be that the degree of u in G is at least $\alpha \cdot |E|$. But there can be at most $2/\alpha$ such high-influence vertices, and therefore at most $2^{2/\alpha}$ subsets of high influence vertices. □

Corollary 4.5. *Algorithm 5 can be used to privately (α, β)-release the cut function on any graph over any product distribution in time $t(\alpha, \beta, \varepsilon)$ for any database of size $|D| \geq t(\alpha, \beta, \varepsilon)$, while preserving ε-differential privacy, where:*

$$t(\alpha, \beta, \varepsilon) = \frac{2^{O(\alpha^{-2}\log(1/\beta))}}{\varepsilon}$$

Proof. This follows directly from a simple modification of Theorem 3.14, by applying Lemma 4.4 and plugging in the size of the decomposition $\mathcal{G}$. The algorithm can then be made privacy preserving by applying proposition 2.1. □

5 Equivalence between agnostic learning and query release

In this section we show an information-theoretic equivalence between *agnostic learning* and *query release* in the statistical queries model. In particular, given an agnostic learning algorithm for a specific concept class we construct a query release algorithm for the same concept class.

Consider a distribution A over $X \times \{0,1\}$ and a concept class C. An *agnostic learning* algorithm (in the strong sense) finds the concept $q \in C$ that approximately maximizes $\Pr_{(x,b)\sim A}[q(x) = b]$ to within an additive error of α. Our reduction from query release to agnostic learning actually holds even for *weak agnostic learning*. A weak agnostic learner is not required to maximize $\Pr_{(x,b)\sim A}[q(x) = b]$, but only to find a sufficiently good predicate q provided that one exists.

We use $\mathrm{STAT}_\tau(A)$ to denote the *statistical query oracle for distribution A* that takes as input a predicate $q : X \to \{0,1\}$ and returns a value v such that $|v - \mathbb{E}_{x\sim A}[q(x)]|$

Definition 5.1 (Weak Agnostic SQ-Learning)**.** Let C be a concept class and $\gamma, \tau > 0$ and $0 < \beta < \alpha \leq 1/2$. An algorithm $\mathcal{A}$ with oracle access to $\mathrm{STAT}_\tau(A)$ is an $(\alpha, \beta, \gamma, \tau)$-*weak agnostic learner for C* if for every distribution A such that there exists $q^* \in C$ satisfying $\Pr_{(x,b)\sim A}[q^*(x) = b] \geq 1/2 + \alpha$, $\mathcal{A}(A)$ outputs a predicate $q : X \to \{0,1\}$ such that $\Pr_{(x,b)\sim A}[q(x) = b] \geq 1/2 + \beta$, with probability at least $1 - \gamma$.

Note that if we can agnostically learn C in the strong sense from queries of tolerance τ to within additive error $\alpha - \beta$ with probability $1 - \gamma$, then there is also an $(\alpha, \beta, \gamma, \tau)$-weak agnostic learner.

We are now ready to state the main result of this section, which shows that a weak agnostic SQ-learner for any concept class is sufficient to release the same concept class in the SQ model.

Theorem 5.1. *Let C be a concept class. Let $\mathcal{A}$ be an algorithm that $(\alpha/2, \beta, \gamma, \tau)$ weak agnostic-SQ learns C with $\tau \leq \beta/8$. Then there exists an algorithm $\mathcal{B}$ that invokes $\mathcal{A}$ at most $T = 8\log|X|/\beta^2$ times and $(\alpha, 0)$-releases C with probability at least $1 - T\gamma$.*

Algorithm 7 Multiplicative weights update

Let D_0 denote the uniform distribution over X.

For $t = 1, ..., T = \lceil 8 \log |X|/\beta^2 \rceil + 1$:

Consider the distributions

$$A_t^+ = 1/2(D, 1) + 1/2(D_{t-1}, 0) \qquad A_t^- = 1/2(D, 0) + 1/2(D_{t-1}, 1)\,.$$

Let $q_t^+ = \mathcal{A}(A_t^+)$ and $q_t^- = \mathcal{A}(A_t^-)$. Let v_t^+ be the value returned by $\mathrm{STAT}_\tau(A_t^+)$ on the query q_t^+ and v_t^- be the value returned by $\mathrm{STAT}_\tau(A_t^-)$ on the query q_t^-. Let $v_t = \max\{v_t^+, v_t^-\} - 1/2$ and q_t be the corresponding query.

If:

$$v_t \le \frac{\beta}{2} - \tau\,, \tag{13}$$

proceed to "output" step.

Update: Let D_t be the distribution obtained from D_{t-1} using a multiplicative weights update step with penalty function induced by q_t and penalty parameter $\eta = \beta/2$ as follows:

$$D'_t(x) = \exp(\eta q_t(x)) \cdot D_{t-1}(x)$$

$$D_t(x) = \frac{D'_t}{\sum_{x \in X} D'_t(x)}$$

Output $a_c = \mathbb{E}_{x \sim D_T}\, c(x)$ for each $c \in C$.

The proof strategy is as follows. We will start from D_0 being the uniform distribution over X. We will then construct a short sequence of distributions $D_1, D_2, \ldots, D_T$ such that no concept in C can distinguish between D and D_T up to bias α. Each distribution D_t is obtained from the previous one using a multiplicative weights approach as in [HR10] and with the help of the learning algorithm that's given in the assumption of the theorem. Intuitively, at every step we use the agnostic learner to give us the predicate $q_t \in C$ which distinguishes between D_t and D. In order to accomplish this we feed the agnostic learner with the distribution A_t that labels elements sampled from D by 1 and elements sampled from D_t by 0. For a technical reason we also need to consider the distribution with 0 and 1 flipped. Once we obtained q_t we can use it as a penalty function in the update rule of the multiplicative weights method. This has the effect of bringing D and D_t closer in relative entropy. A typical potential argument then bounds the number of update steps that can occur before we reach a distribution D_t for which no good distinguisher in C exists.

5.1 Proof of Theorem 5.1

Proof. We start by relating the probability that q_t predicts b from x on the distribution A_t^+ to the difference in expectation of q_t on D and D_{t-1}.

Lemma 5.2. *For any* $q\colon X \to \{0, 1\}$,

$$\Pr_{(x,b) \sim A_t^+}\{q(x) = b\} - \frac{1}{2} = \frac{1}{2}\left(\mathop{\mathbb{E}}_{x \sim D} q(x) - \mathop{\mathbb{E}}_{x \sim D_{t-1}} q(x)\right) \tag{14}$$

Proof. If $q_t = q_t^+$ then

$$\begin{aligned}\Pr_{(x,b)\sim A_t^+}\{q(x) = b\} &= \frac{1}{2}\Pr_{x\sim D}\{q(x) = 1\} + \frac{1}{2}\Pr_{x\sim D_{t-1}}\{q(x) = 0\} \\ &= \frac{1}{2}\mathop{\mathbb{E}}_{x\sim D}[q(x)] + \frac{1}{2}\mathop{\mathbb{E}}_{x\sim D_{t-1}}[1 - q(x)] \\ &= \frac{1}{2} + \frac{1}{2}\left(\mathop{\mathbb{E}}_{x\sim D} q(x) - \mathop{\mathbb{E}}_{x\sim D_{t-1}} q(x)\right)\end{aligned}$$

Note that $\Pr_{(x,b)\sim A_t^-}\{q(x) = b\} = 1 - \Pr_{(x,b)\sim A_t^-}\{q(x) = (1-b)\} = 1 - \Pr_{(x,b)\sim A_t^+}\{q(x) = b\}$, so if $q_t = q_t^-$ then

$$\begin{aligned}\Pr_{(x,b)\sim A_t^+}\{q(x) = b\} = 1 - \Pr_{(x,b)\sim A_t^-}\{q(x) = b\} &= 1 - \left(\frac{1}{2} - \frac{1}{2}\left(\mathop{\mathbb{E}}_{x\sim D} q(x) - \mathop{\mathbb{E}}_{x\sim D_{t-1}} q(x)\right)\right) \\ &= \frac{1}{2} + \frac{1}{2}\left(\mathop{\mathbb{E}}_{x\sim D} q(x) - \mathop{\mathbb{E}}_{x\sim D_{t-1}} q(x)\right)\end{aligned}$$

□

The rest of the proof closely follows [HR10]. For two distributions P, Q on a universe X we define the *relative entropy* to be $\mathrm{RE}(P\|Q) = \sum_{x\in X} P(x)\log(P(x)/Q(x))$. We consider the potential

$$\Psi_t = \mathrm{RE}(D\|D_t)\,.$$

Fact 5.1. $\Psi_t \geq 0$

Fact 5.2. $\Psi_0 \leq \log|X|$

We will argue that in every step the potential drops by at least $\beta^2/4$ Hence, we know that there can be at most $4\log|X|/\alpha^2$ steps before we reach a distribution that satisfies (13).

The next lemma gives a lower bound on the potential drop in terms of the concept, q_t, returned by the learning algorithm at time t. Recall, that η (used below) is the penalty parameter used in the multiplicative weights update rule.

Lemma 5.3 ([HR10])**.**

$$\Psi_{t-1} - \Psi_t \geq \eta\left|\mathop{\mathbb{E}}_{x\sim D} q_t(x) - \mathop{\mathbb{E}}_{x\sim D_{t-1}} q_t(x)\right| - \eta^2 \tag{15}$$

Let

$$\mathrm{opt}_t = \sup_{q\in C}\left|\Pr_{(x,b)\sim A_t^+}\{q(x) = b\} - \frac{1}{2}\right|.$$

Note that $\Pr_{(x,b)\sim A_t^-}\{q(x) = b\} = 1 - \Pr_{(x,b)\sim A_t^+}\{\neg q(x) = b\}$. For the remainder of the proof we treat the two cases symmetrically and only look at how far from $1/2$ these probabilities are. The next lemma shows that either opt_t is large or else we are done in the sense that D_t is indistinguishable from D for any concept from C.

Lemma 5.4. *Let $\alpha > 0$. Suppose*

$$\mathrm{opt}_t \leq \frac{\alpha}{2}\,.$$

Then, for all $q \in C$,

$$\left|\mathop{\mathbb{E}}_{x\sim D} q(x) - \mathop{\mathbb{E}}_{x\sim D_t} q_t(x)\right| \leq \alpha \tag{16}$$

Proof. From Lemma 5.2 we have that for every $q \in C$

$$\frac{\alpha}{2} \geq \mathrm{opt}_t \geq \Pr_{(x,b)\sim A_t^+}\{q(x) = b\} - \frac{1}{2} = \frac{1}{2}\left(\mathop{\mathbb{E}}_{x\sim D} q(x) - \mathop{\mathbb{E}}_{x\sim D_t} q_t(x)\right)$$

Thus $\alpha \geq (\mathbb{E}_{x\sim D}\, q(x) - \mathbb{E}_{x\sim D_t}\, q_t(x))$. Similarly,

$$\frac{\alpha}{2} \geq \mathrm{opt}_t \geq \Pr_{(x,b)\sim A_t^-}\{q(x) = b\} - \frac{1}{2} = \frac{1}{2}\left(\mathop{\mathbb{E}}_{x\sim D_t} q(x) - \mathop{\mathbb{E}}_{x\sim D} q_t(x)\right)$$

Thus $-\alpha \leq (\mathbb{E}_{x\sim D}\, q(x) - \mathbb{E}_{x\sim D_t}\, q_t(x))$. So we conclude $\alpha \geq \left|\mathbb{E}_{x\sim D}\, q(x) - \mathbb{E}_{x\sim D_t}\, q_t(x)\right|$. □

We can now finish the proof of Theorem 5.1. By our assumption, we have that so long as $\mathrm{opt}_t \geq \alpha/2$ the algorithm $\mathcal{A}$ produces a concept q_t such that with probability $1-\gamma$

$$\left|\Pr_{(x,b)\sim A_t^+}\{q_t(x) = b\} - \frac{1}{2}\right| \geq \beta\,. \tag{17}$$

For the remainder of the proof we assume that our algorithm returns a concept satisfying Equation (17) in every stage for which $\mathrm{opt}_t \geq \alpha/2$. By a union bound over the stages of the algorithm, this event occurs with probability at least $1 - T\gamma$.

Assuming Equation (13) is not satisfied we have that

$$\frac{\beta}{4} \leq \frac{\beta}{2} - 2\tau \leq v_t - \tau \leq \left|\Pr_{A_t^+}\{q_t(x) = b\}\right|\,.$$

The leftmost inequality follows because $\tau \leq \beta/8$. We then get

$$\begin{aligned}
\Psi_{t-1} - \Psi_t &\geq \eta\left|\mathop{\mathbb{E}}_{D} q_t(x) - \mathop{\mathbb{E}}_{D_{t-1}} q_t(x)\right| - \eta^2 && \text{(Lemma 5.3)}\\
&\geq \eta\left|4\Pr_{A_t}\{q_t(x) = b\} - 2\right| - \eta^2 && \text{(Lemma 5.2)}\\
&\geq \eta\cdot\beta - \eta^2 && \text{(Equation 13 not satisfied)}\\
&\geq \frac{\beta^2}{2} - \frac{\beta^2}{4} && (\eta = \beta/2)\\
&= \frac{\beta^2}{4}
\end{aligned}$$

Hence, if we put $T \geq 4\log|X|/\beta^2$, we must reach a distribution that satisfies (13). But at that point, call it t, the subroutine $\mathcal{A}$ outputs a concept q_t such that

$$\left|\Pr_{(x,b)\sim A_t^+}(q_t(x) = b) - \frac{1}{2}\right| \leq v_t + \tau < \frac{\beta}{2} + \tau < \beta$$

In this case, by our assumption that Equation 17 is satisfied whenever $\mathrm{opt}_t \geq 1/2 + \alpha/2$, we conclude that $\mathrm{opt}_t < 1/2 + \alpha/2$. By Lemma 5.4, we get

$$\sup_{q\in C}\left|\mathop{\mathbb{E}}_{x\sim D} q(x) - \mathop{\mathbb{E}}_{x\sim D_t} q_t(x)\right| \leq \alpha\,.$$

But this is what we wanted to show, since it means that our output on all concepts in C will be accurate up to error α. □

We remark that for clarity, we let the failure probability of the release algorithm grow linearly in the number of calls we made to the learning algorithm (by the union bound). However, this is not necessary: we could have driven down the probability of error in each stage by independent repetition of the agnostic learner.

This equivalence between release and agnostic learning also can easily be seen to hold in the reverse direction as well.

Theorem 5.5. *Let C be a concept class. If there exists an algorithm $\mathcal{B}$ that $(\alpha, 0)$-releases C with probability $1 - \gamma$ and accesses the database using at most k oracle accesses to $STAT_\tau(A)$, then there is an algorithm that makes $2k$ queries to $\mathrm{STAT}_\tau(A)$ and agnostically learns C in the strong sense with accuracy 2α with probability at least $1 - 2\gamma$.*

Proof. Let Y denote the set of examples with label 1, and let N denote the set of examples with label 0. We use $\mathrm{STAT}_\tau(A)$ to simulate oracles $\mathrm{STAT}_\tau(Y)$ and $\mathrm{STAT}_\tau(N)$ that condition the queried concept on the label. That is, $\mathrm{STAT}_\tau(Y)$, when invoked on concept q, returns an approximation to $\mathbb{P}\mathrm{r}_{x\sim A}\{q(x) = 1 \wedge (x \in Y)\}$ and $\mathrm{STAT}_\tau(N)$ returns an approximation to $\mathbb{P}\mathrm{r}_{x\sim A}\{q(x) = 1 \wedge (x \in Y)]$. We can simulate a query to either oracle using only one query to $\mathrm{STAT}_\tau(A)$.

Run $\mathcal{B}(Y)$ to obtain answers $a_1^Y, \ldots, a_{|C|}^Y$, and run $\mathcal{B}(N)$ to obtain answers $a_1^N, \ldots, a_{|C|}^N$. Note that this takes at most $2k$ oracle queries, using the simulation described above, by our assumption on $\mathcal{B}$. By the union bound, except with probability 2γ, we have for all $q_i \in C$: $|q_i(Y) - a_i^Y| \leq \alpha$ and $|q_i(B) - a_i^N| \leq \alpha$. Let $q^* = \arg\max_{q_i \in C}(a_i^Y - a_i^N)$. Observe that $q^*(D) \geq \max_{q \in C} q(D) - 2\alpha$, and so we have agnostically learned C up to error 2α. □

Feldman proves that even monotone conjunctions cannot be agnostically learned to subconstant error with polynomially many SQ queries:

Theorem 5.6 ([Fel10])**.** *Let C be the class of monotone conjunctions. Let $k(d)$ be any polynomial in d, the dimension of the data space. There is no algorithm $\mathcal{A}$ which agnostically learns C to error $o(1)$ using $k(D)$ queries to $STAT_{1/k(d)}$.*

Corollary 5.7. *For any polynomial in d, $k(d)$, no algorithm that makes $k(d)$ statistical queries to a database of size $k(d)$ can release the class of monotone conjunctions to error $o(1)$.*

Note that formally, Corollary 5.7 only precludes algorithms which release the approximately correct answers to *every* monotone conjunction, whereas our algorithm is allowed to make arbitrary errors on a small fraction of conjunctions.

Remark 5.1. *It can be shown that the lower bound from Corollary 5.7 in fact does* not *hold when the accuracy requirement is relaxed so that the algorithm may err arbitrarily on* 1% *of all the conjunctions. Indeed, there is an inefficient algorithm (runtime* $\mathrm{poly}(2^d)$*) that makes* $\mathrm{poly}(d)$ *statistical queries and releases random conjunctions up to a small additive error. The algorithm roughly proceeds by running multiplicative weights privately (as in [HR10] or above) while sampling, say,* 1000 *random conjunctions at every step and checking if any of them have large error. If so, an update occurs. We omit the formal description and analysis of the algorithm.*

We also remark that the proofs of Theorems 5.1 and 5.5 are not particular to the statistical queries model: we showed generically that it is possible to solve the query release problem using a small number of black-box calls to a learning algorithm, *without accessing the database except through the learning algorithm*. This has interesting implications for any class of algorithms that may make only restricted access to the database. For example, this also proves that if it is possible to agnostically learn some concept class C while preserving ε-differential privacy (even using algorithms that do not fit into the SQ model), then it is possible to release the same class while preserving $T\varepsilon \approx \log|X|\varepsilon$-differential privacy.

Acknowledgments

We would like to thank Guy Rothblum and Salil Vadhan for many insightful discussions, and Nina Balcan and Nick Harvey for pointing out key distinctions between our algorithmic model and that of [BH10].

References

[AHK05] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta algorithm and applications. Technical report, Princeton University, 2005.

[BCD+07] B. Barak, K. Chaudhuri, C. Dwork, S. Kale, F. McSherry, and K. Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 273–282. ACM New York, NY, USA, 2007.

[BDMN05] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the SuLQ framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 128–138. ACM New York, NY, USA, 2005.

[BFJ+94] A. Blum, M. Furst, J. Jackson, M. Kearns, Y. Mansour, and S. Rudich. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, page 262. ACM, 1994.

[BH10] M.F. Balcan and N.J.A. Harvey. Learning submodular functions. *Arxiv preprint arXiv:1008.2159*, 2010.

[BLM00] S. Boucheron, G. Lugosi, and P. Massart. A sharp concentration inequality with applications. *Random Structures and Algorithms*, 16(3):277–292, 2000.

[BLM09] S. Boucheron, G. Lugosi, and P. Massart. On concentration of self-bounding functions. *Electronic Journal of Probability*, 14:1884–1899, 2009.

[BLR08] A. Blum, K. Ligett, and A. Roth. A learning theory approach to non-interactive database privacy. In *Proceedings of the 40th annual ACM symposium on Theory of computing*, pages 609–618. ACM, 2008.

[De11] Anindya De. Lower bounds in differential privacy. *CoRR*, abs/1107.2183, 2011.

[DMNS06] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the Third Theory of Cryptography Conference TCC*, volume 3876 of *Lecture Notes in Computer Science*, page 265. Springer, 2006.

[DN03] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.

[DNR+09] C. Dwork, M. Naor, O. Reingold, G.N. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing*, pages 381–390. ACM New York, NY, USA, 2009.

[Fel10] V. Feldman. A complete characterization of statistical query learning with applications to evolvability. In *Foundations of Computer Science, 2009. FOCS'09. 50th Annual IEEE Symposium on*, pages 375–384. IEEE, 2010.

[GHIM09] Michel X. Goemans, Nicholas J. A. Harvey, Satoru Iwata, and Vahab S. Mirrokni. Approximating submodular functions everywhere. In *SODA*, pages 535–544, 2009.

[HR10] Moritz Hardt and Guy Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *Proc.* 51*st Foundations of Computer Science (FOCS)*. IEEE, 2010.

[HT10] Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *Proc.* 42*nd STOC*. ACM, 2010.

[JPW09] G. Jagannathan, K. Pillaipakkamnatt, and R.N. Wright. A Practical Differentially Private Random Decision Tree Classifier. In *2009 IEEE International Conference on Data Mining Workshops*, pages 114–121. IEEE, 2009.

[Kea98] M. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998.

[KG07] A. Krause and C. Guestrin. Near-optimal observation selection using submodular functions. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 1650. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

[KKT03] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM New York, NY, USA, 2003.

[KLN+08] S.P. Kasiviswanathan, H.K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What Can We Learn Privately? In *IEEE 49th Annual IEEE Symposium on Foundations of Computer Science, 2008. FOCS'08*, pages 531–540, 2008.

[KRSU10] S. Kasiviswanathan, M. Rudelson, A. Smith, and J. Ullman. The Price of Privately Releasing Contingency Tables and the Spectra of Random Matrices with Correlated Rows. In *The 42nd ACM Symposium on the Theory of Computing, 2010. STOC'10*, 2010.

[LW94] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Inf. Comput.*, 108(2):212–261, 1994.

[RR10] A. Roth and T. Roughgarden. Interactive Privacy via the Median Mechanism. In *The 42nd ACM Symposium on the Theory of Computing, 2010. STOC'10*, 2010.

[SF08] Z. Svitkina and L. Fleischer. Submodular approximation: Sampling-based algorithms and lower bounds. In *Foundations of Computer Science, 2008. FOCS '08. IEEE 49th Annual IEEE Symposium on*, pages 697 –706, 2008.

[UV10] Jonathan Ullman and Salil P. Vadhan. Pcps and the hardness of generating synthetic data. *Electronic Colloquium on Computational Complexity (ECCC)*, 17:17, 2010.

[Von10] J. Vondrak. A note on concentration of submodular functions. *Arxiv preprint arXiv:1005.2791*, 2010.