Privately Releasing Conjunctions and the Statistical Query Barrier

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

December 14, 2010

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 the first 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 attributes. The goal of privacy-preserving data analysis is to enable rich statistical analyses on the data set while respecting individual privacy. The formal privacy guarantee we study is *differential privacy* [DMNS06]. Differential privacy is a rigorous notion of privacy which guarantees that 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 the first 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 in 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 can do substantially better if *the algorithm permits an implementation that only accesses the database through statistical queries*. A statistical query is given by a function $q\colon \{0,1\}^d \to \{0,1\}$ and the answer is $\mathbb{E}_{x \in D}\, q(x)$.

Independent of any privacy concerns we therefore raise the natural 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]. In particular, this connection allows us to infer that no polynomial-time algorithm operating in the SQ-model can release even monotone conjunctions to subconstant error, due to an SQ lower bound for agnostically learning monotone conjunctions shown by Feldman [Fel10]; since releasing submodular functions is more general, the lower bound carries over to that setting as well.

While our 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 very natural *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.

- Secondly, and perhaps even more importantly, it gives us 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

[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].

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 $\mathbb{P}\mathrm{r}_{x\in\{0,1\}^d}\{|f(x) - h(x)| \leq \alpha\} \geq 1 - \beta$.*

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 $|D| \geq d^{O(\log(1/\beta)/\alpha^2)}/\epsilon$.*

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 $2q$ statistical queries implies an agnostic learner that makes at most q queries.

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

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

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 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 that Lipschitz-continuous submodular functions are *self-bounding*, which allows us to apply recent dimension-free concentration bounds for self-bounding functions [Von10]. 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.

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 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 modulo cryptographic assumptions, there exists a class of functions that can be privately released using the mechanism of [BLR08], but that no mechanism *which outputs a data set* and which runs in polynomial time (in the dimension of the data space) can privately release. Ullman and Vadhan [UV10] extended this result to the class of conjunctions: they showed that modulo cryptographic assumptions, no polynomial time mechanism *which outputs a data set* can answer even the set of d^2 conjunctions of size 2!

Both of these lower bounds apply only to the class of mechanisms which output data sets, rather than some other data structure encoding their answers. 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. 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.

Kasiviswanathan et. al. [KRSU10] show that average error of $\Omega(1/\sqrt{n})$ is necessary from a privacy point of view (i.e. independent of running time) for mechanisms which answer all conjunctions of constant size, extending work by Dinur and Nissim [DN03] who showed 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 $D \subset X$ of size $|D| = n$. Two databases D and D' are called *adjacent* if one can be obtained from the other by adding a single data item, i.e., $D' = D \cup \{x\}$ for some $x \in X$. 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 $\Pr(\mathcal{M}(D) \in S) \leq e^{\epsilon} \Pr(\mathcal{M}(D') \in S)$.

A *counting query* is specified by a predicate $q\colon X \to [0, 1]$. We will denote the answer to a count 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 $1/\epsilon n$, denoted $Lap(1/\epsilon n)$, 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 $\mathbb{E}_{x \sim D} q(x) + Lap(k/n\epsilon)$. 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 $n \geq \frac{k(\log k + \log(1/\beta))}{\epsilon\tau}$.*

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 $\sigma = k/n\epsilon$ 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 distribution over *concepts* (or predicates) from $X \to [0, 1]$, e.g., the uniform distribution over a finite set of predicates.

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 when the function $q \mapsto q(D)$ can be described by a *submodular function* 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 $f(S \cup T) + f(S \cap T) \leq f(S) + f(T)$. 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 $\partial_x f(S) \geq \partial_x f(T)$ for all $S \subseteq T \subseteq U$ and all $x \in U$.*

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

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 Section A). In this section, we prove our structure theorem, present our algorithm, and prove its correctness. We begin with a simpler version of the structure theorem:

Lemma 3.1. *For every submodular function $f : 2^U \to [0,1]$ and every $\alpha \geq 0$, there is a collection of submodular functions $\mathcal{G}$ with the following properties:*

1. *for every $S \subseteq U$ there is $V_S \subseteq U$ and $g^S : 2^{V_S} \to [0,1] \in \mathcal{G}$ such that $S \subseteq V_S$ and $f(S) = g^S(S)$,*
2. *every $g \in \mathcal{G}$ satisfies $\sup_{x\in V_S, S\subseteq V_S} \partial_x g(S) \leq \alpha$,*
3. *given oracle access to f, we can construct $\mathcal{G}$ in time $O(|\mathcal{G}|)$ together with the mapping $S \mapsto g^S$.*

Moreover, we always have $|\mathcal{G}| \leq O(|U|^{1/\alpha})$.

Proof. We give a proof by construction. We analyze a collection of sets $\mathcal{I}$ generated by the following (non-deterministic) procedure:

$i \leftarrow 0, B_0 \leftarrow \emptyset$
$\mathcal{I} \leftarrow \{B_0\}$
while there exists an $x_i \in U$ such that $\partial_{x_i} f(B) > \alpha$ **do**
  $B_{i+1} \leftarrow B_i \cup \{x_i\}$
  $\mathcal{I} \leftarrow \mathcal{I} \cup \{B_{i+1}\}$, $i \leftarrow i + 1$.

The procedure is nondeterministic because it does not specify which x to choose, but we may construct the entire collection of sets $\mathcal{I}$ that it might ever generate by merely exploring its computation tree using depth first search. First observe that $|\mathcal{I}| \leq O(|U|^{1/\alpha})$, and so we can construct the entire set in time $O(|U|^{1/\alpha})$. This is because for any i:

$$1 \geq f(B_i) = \sum_{j=0}^{|B_i|-1} \partial_{x_j} f(B_j) \geq |B| \cdot \alpha\,. \tag{1}$$

Therefore, it must be that $|B_i| \leq 1/\alpha$, and there are only at most $O(|U|^{1/\alpha})$ such sets over $|U|$ elements.

Consider any $B_i \in \mathcal{I}$ and let $V_{B_i} = \{x \in U : \partial_x f(B_i) \leq \alpha\}$ be the elements that have "low" marginal increase with respect to the set B_i. Define the function $g^{B_i} : 2^{V_{B_i}} \to [0,1]$ as $g^{B_i}(S) = f(S \cup B_i)$. First observe that g^B is a submodular function since it is a shifted version of f.

Claim 3.2. $\sup_{x\in V_{B_i}, S\subseteq V_{B_i}} \partial_x g^{B_i}(S) \leq \alpha$

Proof. By definition, we have $\partial_x g^{B_i}(S) = \partial_x f(B_i \cup S)$ for all $S \subseteq V_{B_i}$. But $\partial_x f(B_i \cup S) \leq \partial_x f(B_i)$, by the submodularity of f, and $\partial_x f(B_i) \leq \alpha$ for every $x \in V_{B_i}$, by the definition of V_{B_i}. □

The previous claim justifies putting $\mathcal{G} = \{g^{B_i} : B_i \in \mathcal{I}\}$. It only remains to argue that for every $S \subseteq U$ we have a $g^{B_i} \in \mathcal{G}$ such that $S \subseteq V_{B_i}$ and $f(S) = g^{B_i}(S)$. Indeed, let B_i denote a maximal set in $\mathcal{I}$ such that $B_i \subseteq S$. Note that such a set must exist because $\emptyset \in \mathcal{I}$, and that this set can be obtained efficiently by following the construction of B using the greedy procedure described above while maintaining $B \subseteq S$. By

construction, $B_i \subseteq S$, and we have $S \subseteq V_{B_i}$ by the maximality of B_i within S: indeed, if there existed an $x \in S$ such that $x \notin V_{B_i}$, then by definition, $\partial_x(B_i) > \alpha$. However, such an x would have been selected by our greedy procedure, and so B_i would not have been a maximal set in S. We now have $g^{B_i}(S) = f(B_i \cup S) = f(S)$ as desired. □

For non-monotone functions, we need a more refined argument. Our main structure theorem replaces Condition 2 in Lemma 3.1 by the stronger guarantee, namely 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 $\overline{f}(S) = f(V \backslash S)$ is also submodular; moreover

$$\inf_{x,S} \partial_x \overline{f}(S) = -\sup_{x,S} \partial_x f(S)\,. \tag{2}$$

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

Theorem 3.3. *For every submodular function $f : 2^U \to [0,1]$ and every $\alpha \geq 0$, there is a collection of submodular functions $\mathcal{G}$ of size at most $O(|U|^{2/\alpha})$ such that*

1. *For every $S \subseteq U$ there is $g^S : 2^{V_S} \to [0,1] \in \mathcal{G}$ such that $S \subseteq V_S \subseteq U$ and $f(S) = g^S(S)$*
2. *every $g \in \mathcal{G}$ satisfies $\sup_{x \in V_S, S \subseteq V_S} |\partial_x g(S)| \leq \alpha$,*
3. *given oracle access to f, we can construct $\mathcal{G}$ in time $O(|\mathcal{G}|)$ together with the mapping $S \mapsto g^S$.*

Proof. First apply Lemma 3.1 to obtain a collection of functions $\mathcal{G}(f)$ satisfying the conditions of the lemma. Hence, for every $g^B \in \mathcal{G}(f)$, we have $\sup_{x \in V_B, S \subseteq V_B} \partial_x g^B(S) \leq \alpha$, but not necessarily $\partial_x g^B(S) \geq -\alpha$. Consider instead the function $\overline{g^B}$. This is submodular since g^B was submodular, and moreover, $\partial_x \overline{g^B}(S) \geq -\alpha$ for every $x \in V_B, S \subseteq V_B$.

We now apply Lemma 3.1 again to each $\overline{g^B} \in \mathcal{G}(f)$, and hence obtain a collection of functions for each $\overline{g^B}$ denoted by $\mathcal{G}(\overline{g^B})$. Since each $h^C \in \mathcal{G}(\overline{g^B})$ is a shift of $\overline{g^B}$, and $\overline{g^B}$ is submodular, we have $\inf_{x \in V_C, S \subseteq V_C} \partial_x h^C(S) \geq -\alpha$. But moreover by the statement of the lemma, $\sup_{x \in V_C, S \subseteq V_C} \partial_x h^C(S) \leq \alpha$. Consequently, $\left|\partial_x h^C(S)\right| \leq \alpha$ for every $x \in V_C, S \subseteq V_C$, and (13) implies the same for $\overline{h^C}$ too. Moreover, using Property 1 for the two invocations of Lemma 3.1, we get that for every S, there is an h^C such that $f(S) = \overline{h^C}(S)$. Therefore take $\mathcal{G} = \cup_{g^B \in \mathcal{G}(f)} \{\overline{h^C} \mid h^C \in \mathcal{G}(\overline{g^B})\}$ 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 avoid

Algorithm 1 Learning a submodular function

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

Let $\alpha' = \frac{\alpha^2}{\log(2/\beta)(5\alpha/3+2)}$.
Construct the collection of functions $\mathcal{G}$ given by Theorem 3.3 with parameter α'.
Estimate the value $\mu_{g^B} = \mathbb{E}_{S \sim \mathcal{D}_{V_B}}[g^B(S)]$ for each $g^B \in \mathcal{G}$.
Output the data structure h which maps a given query $S \subseteq U$ to the value μ_g where g is the function in $\mathcal{G}$ that is assigned to S.

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

Proof. For a set $S \subseteq V$, we let $g^{B(S)}$ denote the function that's assigned to S by our algorithm. We claim that

$$\Pr_{S\sim\mathcal{D}}\{|f(S) - h(S)| > \alpha\} = \Pr_{S\sim\mathcal{D}_{V_{B(S)}}}\left\{\left|g^{B(S)}(S) - \mu_g\right| > \alpha\right\}. \tag{3}$$

To see this, recall that each $g^B \in \mathcal{G}$ is defined by a set $B \subseteq V$ so that $g(S) = f(S \cup B)$ for all S, and that S is associated with g^B only if B is a maximal set such that $S \subseteq V_B$. Hence, the distribution of $g^B(S)$ for $S \sim \mathcal{D}_{V_B}$ does not change under the condition that $S \supseteq B$. On the other hand, Theorem 3.3 with Lemma A.2 implies that each g is a self-bounding function. Applying the concentration inequality for submodular functions stated as Corollary A.3, we get

$$\Pr_{S\sim\mathcal{D}}\left\{\left|g(S) - \mu_g\right| \geq \alpha' t\right\} \leq 2\exp\left(-\frac{t^2}{2(1/\alpha' + 5/6t)}\right). \tag{4}$$

Plugging in $t = \alpha/\alpha' = \frac{\log(2/\beta)(5\alpha/3+2)}{\alpha}$ and simplifying we get $\Pr\left\{\left|g^S(S) - \mu_g\right| > \alpha\right\} \leq \beta$. Combining this with Equation 3, the claim follows. □

Approximating submodular functions from queries with tolerance. In the above we assumed that we have an exact query oracle to the submodular function f. For our applications we need that the algorithm works when each query is answered with tolerance τ.

Theorem 3.5 (Theorem 3.4 with tolerance)**.** *For any $\alpha > 0, \beta > 0$, Algorithm 1 (α,β)-approximates any submodular function $f\colon 2^U \to [0,1]$ under any product distribution in time $|U|^{O(\alpha^{-2}\log(1/\beta))}$ using oracle queries to f of tolerance $\alpha/4$.*

Proof (sketch). The proof works the same way as that of Theorem 3.4 and Theorem 3.3 except we need to take into account an additive error of $\alpha/4$ on each oracle query. Since each value $\partial_x f(S)$ consists of two queries to the oracle our estimate of $\partial_x f(S)$ is correct up to an error of $\alpha/2$. We can therefore follow the proof Theorem 3.3 with error parameter $\alpha/2$. This will guarantee that every $g \in \mathcal{G}$ satisfies $\partial_x g(S) \leq \alpha$. From there on the proof is identical. □

Generalized submodular functions and certain non-product distributions. For our main application to Boolean conjunctions we need Theorem 3.5 to hold for somewhat generalized submodular functions as well as certain non-product distributions.

Consider an alphabet $[p] = \{0, \ldots, p-1\}$ of size p and a universe U as before. Identify an element $S \in [p]^U$ with a function $S : U \to [p]$ in the usual way. We define a partial order on $[p]^U$ by putting $S \subseteq T$ if and only if $S(x) \neq 0 \Rightarrow S(x) = T(x)$. (All other elements are incomparable.) We say that $S \in [p]^U$ has *weight* w if $S(x) \neq 0$ for exactly w elements $x \in U$. Finally, for a function $f\colon [p]^U \to \mathbb{R}$ and $x \in U, i \in [p]$ Finally, let $\partial_{x,i} f(S) = f(S[x \mapsto i]) - f(S)$. Here, $S[x \mapsto i]$ is the function that maps x to i and agrees with S everywhere else. We then call f *submodular* if and only if $\partial_{x,i} f(S) \geq \partial_{x,i}(T)$ for all $S \subseteq T$, all $i \in [p]$ and all $x \in U$ satisfying $S(x) = 0$. We call f *monotone* if $f(T) \geq f(S)$ for all $S \subseteq T$. Note that for $p = 2$ this is the same as before.

We have the following theorem.

Theorem 3.6. *For any $\alpha, \beta > 0$, $p \in \mathbb{N}$ and $w \in \{1, \ldots, d\}$, Algorithm 1 (α,β)-approximates any submodular function $f\colon [p]^U \to [0,1]$ under the uniform distribution over strings of weight w in time $(p|U|)^{O(\alpha^{-2}\log(1/\beta))}$ using oracle queries to f of tolerance $\alpha/4$.*

Proof (sketch). We repeat the construction of Theorem 3.3. At every step we now explore all p possibilities of choosing an elment that we add to the set B we're maintaining. Hence the resulting tree is now of size at most $(p|U|)^{2/\alpha}$. We then repeat the proof of Theorem 3.5 where we choose the uniform distribution over strings of weight w (rather than a product distribution). For this to work we need that the function f has sufficient concentration properties under this distribution. This is demonstrated in Appendix A by combining Lemma A.4 with Lemma A.5. □

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 conjunctions over a boolean database. In Appendix C, we also show how our mechanism can be applied to release the *cut function* of an arbitrary graph.

Given our previous results, we only need to argue that conjunctions can be described by a submodular function. While this is not directly true for conjunctions it is true for disjunctions and submodular functions over a ternary alphabet. Indeed, every element $S \in \{0,1,2\}^d$ naturally corresponds to a Boolean conjunction $c_S : \{0,1\}^d \to \{0,1\}$ as follows $c_S(x) = \left(\bigwedge_{i:\, S(i)=1} x_i\right) \wedge \left(\bigwedge_{i:\, S(i)=2} \neg x_i\right)$. (Note that in contrast to Section 3 here we use x to denote an element of $\{0,1\}^d$.) Similarly, the disjunction predicate $d_S : \{0,1\}^d \to \{0,1\}$ is defined to be $d_S(x) = \left(\bigvee_{i:\, S(i)=1} x_i\right) \vee \left(\bigvee_{i:\, S(i)=2} \neg x_i)\right)$. For $S \in \{0,1,2\}^d$, let $\bar{S}$ denote the string such that $\bar{S}(i) = 1$ if $S(i) = 2$, $\bar{S}(i) = 2$ if $\bar{S}(i) = 1$, and $\bar{S}(i) = 0$ if $S(i) = 0$

It is easy to see that for any data set $D \subseteq \{0,1\}^d$, $c_S(D) = 1 - d_{\bar{S}}(D)$. Therefore, to release the class of conjunctions, it is sufficient to release the class of disjunctions. Let $F_{\mathrm{Disj}} : \{0,1,2\}^d \to [0,1]$ be the function such that $F_{\mathrm{Disj}}(S) = d_S(D)$. Then:

Lemma 4.1. *$F_{Disj}(D)$ 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 $F_{\mathrm{Disj}}(D)$ and the set coverage function $\mathrm{Cov} : 2^U \to [0, |D|]$ defined to be $\mathrm{Cov}(S) = |\bigcup_{X \in U} X|$, which is a monotone submodular function. □

We therefore obtain the following corallary directly by combining Theorem 3.6 with Proposition 2.1.

Corollary 4.2. *Let $\alpha, \beta, \epsilon > 0, 1 \le w \le d$. There is an ϵ-differentially private algorithm that (α,β)-releases the set of all Boolean conjunction of width w in time $d^{t(\alpha,\beta)}$ for any data set of size $|D| \ge d^{t(\alpha,\beta)}/\epsilon$ where $t(\alpha,\beta) = O(\alpha^{-2}\log(1/\beta))$.*

Of course, we can instantiate the theorem for every $w \in \{1,\ldots,k\}$ to obtain a statement for conjunctions of any width.

In the next section we give a lower bound for releasing *all* conjunctions with small error from SQ queries. Our theorem here gives only an upper bound on the error for almost all conjunctions. In Appendix B, however, we extend the lower bound from Section 5 to hold for *average* error as well. This makes our upper and lower bounds directly comparable.

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.

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 2 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:

$$v_t \leq \frac{\beta}{2} - \tau\,, \tag{5}$$

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 the most 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 $q\colon X \to \{0,1\}$,*

$$\Pr_{(x,b)\sim A_t^+}\{q(x) = b\} - \frac{1}{2} = \frac{1}{2}\left(\mathop{\mathbb{E}}_{x\sim D} q(x) - \mathop{\mathbb{E}}_{x\sim D_{t-1}} q(x)\right) \tag{6}$$

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 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 (5).

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])**.**

$$\Psi_{t-1} - \Psi_t \geq \eta\left|\mathop{\mathbb{E}}_{x\sim D} q_t(x) - \mathop{\mathbb{E}}_{x\sim D_{t-1}} q_t(x)\right| - \eta^2 \tag{7}$$

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 $q \in C$,

$$\left| \mathop{\mathbb{E}}_{x\sim D} q(x) - \mathop{\mathbb{E}}_{x\sim D_t} q_t(x) \right| \leq \alpha \tag{8}$$

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 $1 - \gamma$

$$\left| \Pr_{(x,b)\sim A_t^+}\{q_t(x) = b\} - \frac{1}{2} \right| \geq \beta . \tag{9}$$

For the remainder of the proof we assume that our algorithm returns a concept satisfying Equation 9 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 5 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

$$\begin{aligned}
\Psi_{t-1} - \Psi_t &\geq \eta \left| \mathop{\mathbb{E}}_{D} q_t(x) - \mathop{\mathbb{E}}_{D_{t-1}} q_t(x) \right| - \eta^2 && \text{(Lemma 5.3)} \\
&\geq \eta \left| 4 \Pr_{A_t}\{q_t(x) = b\} - 2 \right| - \eta^2 && \text{(Lemma 5.2)} \\
&\geq \eta \cdot \beta - \eta^2 && \text{(Equation 5 not satisfied)} \\
&\geq \frac{\beta^2}{2} - \frac{\beta^2}{4} && (\eta = \beta/2) \\
&= \frac{\beta^2}{4}
\end{aligned}$$

Hence, if we put $T \geq 4 \log |X|/\beta^2$, we must reach a distribution that satisfies (5). 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 9 is satisfied whenever $\text{opt}_t \geq 1/2 + \alpha/2$, we conclude that $\text{opt}_t < 1/2 + \alpha/2$. 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 $\text{STAT}_\tau(A)$ 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 $\text{STAT}_\tau(A)$ to simulate oracles $\text{STAT}_\tau(Y)$ and $\text{STAT}_\tau(N)$ that condition the queried concept on the label. That is, $\text{STAT}_\tau(Y)$, when invoked on concept q, returns an approximation to $\Pr_{x\sim A}\{q(x) = 1 \wedge (x \in Y)\}$ and $\text{STAT}_\tau(N)$ returns an approximation to $\Pr_{x\sim A}\{q(x) = 1 \wedge (x \in Y)]$. We can simulate a query to either oracle using only one query to $\text{STAT}_\tau(A)$.

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. However, in appendix B we prove from first principles that in fact, no polynomial time algorithm making only SQ queries can release monotone conjunctions to α error *even if* it is allowed to

make arbitrary errors on an α fraction of conjunctions of each size k. This lower bound applies in particular to our algorithm, and shows that the difficulty of agnostic learning in the SQ model cannot be circumvented in the release setting by algorithms that make arbitrary errors on a fraction of their output. Combined with theorem 5.1, this also provides a new proof of the impossibility of agnostically learning monotone conjunctions to subconstant error in the SQ model.

We 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 $T\epsilon \approx \log|X|\epsilon$-differential privacy.

Acknowledgments

We would like to thank Guy Rothblum 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.

[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.

[DP09] D. Dubhashi and A. Panconesi. Concentration of Measure for the Analysis of Randomized Algorithms. 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] J. Ullman and S. Vadhan. PCPs and the Hardness of Generating Synthetic Data . *Manuscript*, 2010.

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

A Concentration properties of submodular functions

In this section we present the concentration inequalities for submodular functions that were needed in the analysis of our main algorithm. We will employ "dimension-free" concentration bounds which apply to *self-bounding* functions (i.e., those whose concentration does not explicitly depend on the number of random variables):

Definition A.1 (Self Bounding Functions [DP09])**.** Let $\Omega = \prod_{i=1}^{d} \Omega_i$ be an arbitrary product space. A function $f : \Omega \to \mathbb{R}$ is (a, b)-*self-bounding* if there are functions $f_i : \prod_{j \neq i} \Omega_i \to \mathbb{R}$ such that if we denote $x^{(i)} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d)$, then for all $x \in \Omega$, i,

$$0 \leq f(x) - f_i(x^{(i)}) \leq 1 \tag{10}$$

and

$$\sum_{i=1}^{d} (f(x) - f_i(x^{(i)})) \leq a f(x) + b \tag{11}$$

Self-bounding functions satisfy the following strong concentration properties:

Theorem A.1 ([BLM00, BLM09])**.** *If f is a (a, b)-self-bounding function for $a > 1/3$, and $Z = f(X_1, \ldots, X_d)$ where each X_i taking values in Ω_i is independently random, then:*

$$\Pr\{|Z - \mathbb{E}\, Z| \geq t\} \leq 2 \exp\left(-\frac{t^2}{2(\mathbb{E}\, Z + b + ct)}\right), \tag{12}$$

where $c = (3a - 1)/6$.

We remark that Theorem A.1 is dimension-free: it shows that $f(Z)$ is concentrated around $\mathbb{E}[Z]$ with standard deviation $O(\sqrt{\mathbb{E}[Z]})$, rather than merely $O(\sqrt{d})$, which holds for any Lipschitz function. This tighter concentration is crucial to our application.

Submodular functions satisfying a Lipschitz condition are in fact self-bounding. This was observed by Vondrak [Von10] for $p = 2$.

Lemma A.2 (Vondrak [Von10])**.** *If $f \colon 2^U \to \mathbb{R}$ is a submodular function such that*

$$\sup_{S,x} |\partial_x f(S)| \leq 1\,, \tag{13}$$

then f is $(2, 0)$-self-bounding.

Corollary A.3 (Concentration for submodular functions)**.** *If $f\colon 2^U \to \mathbb{R}$ be a submodular function satisfying the Lipschitz condition 13. Then for any product distribution S over 2^U, we have*

$$\Pr\{|f(S) - \mathbb{E}\, f(S)| \geq t\} \leq 2\exp\left(-\frac{t^2}{2(\mathbb{E}\, Z + 5t/6)}\right), \tag{14}$$

The above bounds are all that are needed for our algorithm for approximating submodular functions over product distributions.

For application to Boolean conjunctions, we need a generalization to $p > 2$. However, we only need this claim for *monotone submodular functions*.

Lemma A.4 (Concentration for generalized monotone submodular functions)**.** *Let $f\colon [p]^U \to \mathbb{R}$ be a monotone submodular function satisfying the Lipschitz condition*

$$\sup_{S,x,i} \left|\partial_{x,i} f(S)\right| \leq 1\,. \tag{15}$$

Then for any product distribution S over $[p]^U$, we have

$$\Pr\{|f(S) - \mathbb{E}\, f(S)| \geq t\} \leq 2\exp\left(-\frac{t^2}{2(\mathbb{E}\, Z + t/3)}\right), \tag{16}$$

Proof. Using Theorem A.1 it suffices to show that the function is $(1, 0)$-self bounding.

To argue this claim it will be convenient to use the notation from Theorem A.1. We will therefore denote an element from $[p]^U$ by a string $x \in [p]^d$. In this notation we write $\partial_{i,a} f(x) = f(x[i := a]) - f(x)$ whenever $x_i = 0$ and $a \in [p]$. For each $x \in [p]^d$, denote by $x^{(i)}$ (as above) the vector $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots x_d) \in [p]^{d-1}$ and by $x_{(i)}$ the vector $(x_1, \ldots, x_{i-1}, 0, \ldots, 0) \in [p]^d$. Put $f_i(x^{(i)}) = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_u)$. We then have:

$$\begin{aligned}
\sum_{i=1}^{d} f(x) - f_i(x^{(i)}) &= \sum_{i=1}^{d} \partial_{i,x_i} f(x^{(i)}) \\
&\leq \sum_{i=1}^{u} \partial_{i,x_i} f(x_{(i)}) \\
&= \sum_{i=1}^{u} f(x_1, \ldots, x_i, 0, \ldots, 0) - f(x_1, \ldots, x_{i-1}, 0, \ldots, 0) \\
&\leq f(x)
\end{aligned}$$

□

We now show that we have roughly the same concentration bounds when we choose elements of weight w randomly rather than elements from a related product distribution over $[p]^U$. While this doesn't use submodularity, it does require the function to be *monotone*. Indeed, given a monotone submodular function $f\colon [p]^U \to \mathbb{R}$, let $S \in [p]^U$ be the random variable where $S(x) = 0$ with probability $1 - (w/d)$, and with probability w/d we choose $S(x)$ uniformly at random from $\{1, \ldots, p-1\}$. On the other hand, let $T \in [p]^d$ denote the uniform distribution over strings in $[p]^d$ of weight w. The following claim can be found in [BH10].

Lemma A.5. *Assume $f : [p]^U \to \mathbb{R}$ is monotone function, and S and T are chosen at random as above. Then,*

$$\Pr[f(T) \geq \tau] \leq 2\Pr[f(S) \geq \tau] \tag{17}$$

$$\Pr[f(T) \leq \tau] \leq 2\Pr[f(S) \leq \tau] \tag{18}$$

Proof. First we prove inequality 17.

$$\begin{aligned}
\Pr[f(S) \geq \tau] &= \sum_{i=0}^{d} \Pr[f(S) \geq \tau \mid |S| = i] \cdot \Pr[|S| = i] \\
&= \sum_{i=w}^{d} \Pr[f(S) \geq \tau \mid |S| = i] \cdot \Pr[|S| = i] \\
&\geq \sum_{i=w}^{d} \Pr[f(T) \geq \tau] \cdot \Pr[|S| = i] && \text{(monotonicity)} \\
&= \Pr[f(T) \geq \tau] \cdot \Pr[|S| \geq w] \\
&\geq 1/2 \Pr[f(T) \geq \tau]
\end{aligned}$$

The proof of (18) works the same way. □

B A Comparable Lower Bound

In this section, we prove directly that no polynomial time algorithm that can be implemented in the SQ model (or any algorithm that operates in the local privacy model) can release conjunctions to subconstant error α, even if it is allowed to make arbitrary errors on a fraction of conjunctions of each size, as our algorithm does. That is, we prove a lower bound that is directly comparable to our algorithm.

Definition B.1. Let C be a class of boolean predicates $f : \{0,1\}^n \to \{0,1\}$, and let D be a distribution over $\{0,1\}^n$. The SQ-Dimension of C with respect to D is defined to be SQ-DIM$(C, D) = d$, where d is the largest integer such that C contains d functions $f_1, \ldots, f_d$ with the property that for all $i \neq j$ we have:

$$|\Pr_D[f_i(x) = f_j(x)] - \Pr_D[f_i(x) \neq f_j(x)]| \leq \frac{1}{d^3}$$

Theorem B.1 ([BFJ+94])**.** *Let C be a class of boolean functions over $\{0,1\}^n$ and D be a distribution such that SQ-DIM$(C, D) = d \geq 16$. Then any algorithm that makes statistical queries with tolerance $1/d^{1/3}$ must make at least $d^{1/3}/2$ queries to learn C to error less than $1/2 - 1/d^3$.*

Definition B.2. For each $S \subseteq [d]$ there exists a parity function $\chi_S(x) : \{0,1\}^d \to \{0,1\}$ defined as follows:

$$\chi_S(x) = \left(\sum_{i \in S} x_i \right) \mod 2$$

The class of parity functions is defined to be $C_p = \{\chi_S : S \subseteq [d]\}$. Note that $|C_p| = 2^d$.

We first observe that any subset of the parity functions $C'_p \subseteq C_p$ has SQ-Dimension $|C'_p|$ with respect to the uniform distribution:

Observation B.1. *Let $C'_p \subseteq C_p$ be some subset of the parity functions, and let U be the uniform distribution over $\{0,1\}^n$. Then we have SQ-DIM$(C'_p, D) = |C'_p|$.*

This follows since every two distinct parity functions $\chi_T(x), \chi_{T'}(x)$ contain some variable x_i in their symmetric difference, and so have no correlation over the uniform distribution.

In this section, we prove from first principles that no algorithm that makes only polynomially many SQ queries can release even monotone conjunctions to subconstant error, even if it allowed to make arbitrary

errors on a small fraction of conjunctions of each size. We remark that combined with Theorem 5.1, this gives a new proof that no algorithm operating in the SQ model can strongly agnostically learn monotone conjunctions to subconstant error.

First, let us say that an algorithm is (α,β)-useful with respect to a class C if it correctly releases the answers to *every* query in C up to additive error α, except with probability β: That is, with probability $1-\beta$, it releases $a_1,\ldots,a_{|C|}$ such that $\max_{i\in C}|f(D)-a_i|\leq\alpha$.

We first observe that absent computational constraints, the problem of learning parity functions is strictly easier than the problem of usefully releasing the class of parity functions. We may view the bitstring/label pairs (x,y) in the database as single bitstrings $x\in\{0,1\}^{d+1}$, with $x_{d+1}=y$. Let $S\subseteq[d]$ be the set that defines some parity function $\chi_S(x)$ over $\{0,1\}^d$, and consider $\chi_{S\cup\{d+1\}}(x)$, a parity function over $\{0,1\}^{d+1}$. We have:

$$\begin{aligned}\chi_{S\cup\{d+1\}}(D) &= \frac{1}{n}\left|\{x\in D:\chi_{S\cup\{d+1\}}(x)=1\}\right| \\ &= \frac{1}{n}\left|\{x\in D:\chi_S(x)\neq x_{d+1}\}\right| \\ &= \frac{1}{n}\left|\{x\in D:\chi_S(x)\neq y\}\right| \\ &= err(\chi_S(x))\end{aligned}$$

If we have an (α,β)-useful mechanism with respect to a subset of parity queries, we therefore also have an (α,β) learning algorithm: we simply exhaustively search through the set of all parity queries χ_S such that $y\in S$, and select the one with the lowest value, which as we have shown, corresponds to the parity query with lowest error up to an additive ϵ.

We therefore have the following corollary:

Corollary B.2. *Let $C'_P\subset C_P$ be any subset of parity functions with superpolynomial cardinality: $|C'_P|\geq d^{\omega(1)}$. There does not exist an algorithm in the statistical query model model that makes only polynomially many queries and is (α,β) useful with respect to C'_P, for any α bounded away from $1/2$ by a non-negligible factor.*

We will now show how Corollary B.2 precludes a locally private release algorithm for monotone conjunctions to subconstant error.

Definition B.3. For each $S\subseteq[d]$ there exists a monotone conjunction $c_S(x):\{0,1\}^d\to\{0,1\}$ defined as follows:

$$c_S(x)=\prod_{i\in S}x_i$$

The class of conjunctions is defined to be $C_c=\{c_S:S\subseteq[d]\}$. Note that $|C_c|=2^d$. If $|S|=w$, then we say that c_S has width w.

Note that monotone conjunctions are a very simple class, contained within the class of conjunctions, disjunctions, decision lists, halfplanes, and other standard predicate classes of interest. Therefore a lower bound for the class of monotone conjunctions is a serious impediment to locally-private data release. Moreover, unlike parity functions, conjunctions are *easy* to learn in the SQ-model (and therefore in the local privacy model). Therefore, our result shows a formal separation between what is possible to *learn* in the local privacy model, and what is possible to *release*.

We first observe that for each parity function $\chi_S(x)$, we can define a real valued polynomial $f_S(x):\mathbb{R}^d\to\mathbb{R}$ such that for every $x\in\{0,1\}^d$, $f_S(x)=\chi_S(x)$: We do this in the natural way, using multilinear polynomial

interpolation:

$$\begin{aligned} f_S(x) &= \sum_{y\in\{0,1\}^d} \chi_S(y)\cdot\left(\prod_{i:y_i=1} x_i\right)\cdot\left(\prod_{i:y_i=0}(1-x_i)\right) \\ &= \sum_{y:\sum_{i\in S} y_i \mod 2=1}\left(\prod_{i:y_i=1} x_i\right)\cdot\left(\prod_{i:y_i=0}(1-x_i)\right) \\ &= \sum_{i=1}^{|S|}\sum_{T\subseteq S:|T|=i}(-2)^{i-1}\cdot\left(\prod_{j\in T} x_j\right) \\ &= \sum_{i=1}^{|S|}\sum_{T\subseteq S:|T|=i}(-2)^{i-1}\cdot c_T(x) \end{aligned}$$

By restricting the above derivation to the domain $\{0,1\}^d$ and by invoking linearity, it is therefore also easy to see:

$$\chi_S(D) = \sum_{i=1}^{|S|}\sum_{T\subseteq S:|T|=i}(-2)^{i-1}\cdot c_T(D) \tag{19}$$

for any database D.

That is, we have shown how to express parities over sets of size $|S|$ as linear combinations of conjunctions over sets of size at most $|S|$. Suppose for point of contradiction that there exists a mechanism that simultaneously (α,α)-releases width w monotone conjunctions for each $w\in[d]$ (i.e. one that may make arbitrary errors on an α fraction of conjunctions of size k, for each k – note that our mechanism makes exponentially fewer errors than this), for some $\alpha = o(1)$. Then using equation 19, we could compute the approximate value of each parity function $\chi_S(x)$:

$$\begin{aligned} \sum_{i=1}^{|S|}\sum_{T\subseteq S:|T|=i}(-2)^{i-1}\cdot a_T &\leq \sum_{i=1}^{|S|}\sum_{T\subseteq S:|T|=i}(-2)^{i-1}\cdot(c_T(D)+(-1)^{i-1}\alpha)+\sum_{i=1}^{|S|}2^{i-1}\alpha \\ &= \sum_{i=1}^{|S|}\sum_{T\subseteq S:|T|=i}(-2)^{i-1}\cdot c_T(D)+\sum_{i=1}^{|S|}\sum_{T\subseteq S:|T|=i}2^{i-1}\cdot\alpha+\sum_{i=1}^{|S|}2^{i-1}\alpha \\ &= \chi_S(D)+\alpha\cdot\left(\sum_{i=1}^{S}\binom{|S|}{i}\cdot 2^{i-1}\right)+\alpha(2^{|S|}-1) \\ &= \chi_S(D)+\left(\frac{3^{|S|}-1}{2}+2^{|S|}-1\right)\cdot\alpha \end{aligned}$$

where the first inequality follows from the hypothesized usefulness of the release mechanism, and the second equality follows from equation 19. We can similarly derive:

$$\sum_{i=1}^{|S|}\sum_{T\subseteq S:|T|=i}(-2)^{i-1}\cdot a_T \geq \chi_S(D)-\left(\frac{3^{|S|}-1}{2}+2^{|S|}-1\right)\cdot\alpha$$

Together, we have shown that a mechanism that (α,α)-releases conjunctions with up to s variables yields an (α',β)-useful mechanism for parities of up to s variables, with $\alpha' = O(3^s\cdot\alpha)$.

By setting $s = O(\log\frac{1}{\alpha})$, we can take α' to be $1/4$. However, since $\alpha = o(1)$ by assumption, we have that $\log\frac{1}{\alpha} = \omega(1)$. Let $C'_p = \{\chi_S\in C_p : |S|\leq s\}$. We have: $|C'_p| = d^{\omega(1)}$. By Corollary B.2, we know that α' cannot be bounded away from $1/2$ by a constant, and so we have derived a contradiction. We have therefore proven the following theorem:

Theorem B.3. *No release mechanism in the SQ model that makes only $d^{O(\log(1/\alpha))}$ queries can simultaneously (α, α)-release monotone conjunctions of width w for each $w \in \lceil 1/\alpha \rceil$ variables.*

This has several implications. Recall that Kasiviswanathan et al. showed that their Local Privacy model is polynomially equivalent to the SQ model [KLN+08]. Therefore we have:

Corollary B.4. *No release mechanism in the SQ model with a database of poly(d) many individuals can release even monotone conjunctions to subconstant error.*

Because we also proved the polynomial equivalence between SQ agnostic learning and SQ release, by applying Theorem 5.1 we also get the following corollary, originally due to Feldman [Fel10], proven here in a novel way:

Corollary B.5. *No agnostic learning algorithm operating in the SQ model can learn monotone conjunctions to subconstant error, even over the uniform distribution.*

C 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 C.1. *For any graph G, f_G is submodular.*

Lemma C.1. *The decomposition from Theorem 3.3 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 C.2. *Algorithm 1 can be used to privately (α, β)-release the cut function on any graph over any product distribution in time $t(\alpha, \beta, \epsilon)$ for any database of size $|D| \geq t(\alpha, \beta, \epsilon)$, while preserving ϵ-differential privacy, where:*

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

Proof. This follows directly from a simple modification of Theorem 3.5, by applying Lemma C.1 and plugging in the size of the decomposition $\mathcal{G}$. The algorithm can then be made privacy preserving by applying proposition 2.1. □