SIAM J. COMPUT.
Vol. 42, No. 4, pp. 1494–1520

PRIVATELY RELEASING CONJUNCTIONS AND THE STATISTICAL QUERY BARRIER[*]

ANUPAM GUPTA[†], MORITZ HARDT[‡], AARON ROTH[§], AND JONATHAN ULLMAN[¶]

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 access the data itself only by using statistical queries. A trivial solution is to exhaustively ask all queries in C. In this paper, we investigate how and when we can do better than this naïve approach. 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 run-time is not a concern. 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. These results are interesting not only from a learning theoretic point of view, but also from the perspective of *privacy-preserving data analysis*. In this context, 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.

Key words. differential privacy, agnostic learning, SQ learning

AMS subject classifications. 68Q32, 68Q25, 68Q17

DOI. 10.1137/110857714

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 analysis on the data set while respecting individual privacy. In particular, we would like to guarantee *differential privacy* [13], a rigorous notion of privacy that guarantees that the outcome of a statistical analysis is nearly indistinguishable on any two data sets that differ in only a single individual's data.

One of the most important classes of statistical queries on the data set are (monotone) Boolean conjunctions, sometimes called contingency tables or marginal queries. See, for example, [3, 7, 22, 32]. A Boolean conjunction corresponding to a subset $S \subseteq [d]$ counts what fraction of the individuals have each attribute in S set to 1.

[*]Received by the editors December 2, 2011; accepted for publication (in revised form) March 15, 2013; published electronically July 9, 2013. An extended abstract of this work previously appeared in *Proceedings of the 43rd Annual ACM Symposium on Theory of Computing* (STOC 2011).
http://www.siam.org/journals/sicomp/42-4/85771.html

[†]Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213 (anupamg@cs.cmu.edu). This author's research was partially supported by NSF award CCF-1016799 and an Alfred P. Sloan Fellowship.

[‡]IBM Research Almaden, San Jose, CA 95120 (mhardt@us.ibm.com). This author's research was supported by NSF grants CCF-0426582 and CCF-0832797.

[§]Department of Computer and Information Sciences, University of Pennsylvania, Philadelphia, PA 19104 (aaroth@cis.upenn.edu). This author's research was partially supported by NSF awards CCF-1101389 and CNS-1065060 and an NSF CAREER award.

[¶]School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138 (jullman@seas.harvard.edu). This author's research was supported by NSF grant CNS-0831289.

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 run-time 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 convexity-like property that assigns values to subsets of a universe, with the marginal value of adding an element to a set decreasing as the set gets larger; a formal definition is found in section 2. Submodularity often arises in data analysis and machine learning problems [26], 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 accesses the database only 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 by exhibiting a 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 sufficient 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 [24]. Using an SQ lower bound for agnostically learning monotone conjunctions shown by Feldman [15], this connection implies that no polynomial-time algorithm operating in the SQ model can release *all* monotone conjunctions to subconstant error. Since monotone conjunction queries can be described by a submodular function, the lower bound applies to releasing the values of submodular functions on all inputs as well.

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

- First, it also characterizes what can be released in the *local privacy* model of Kasiviswanathan et al. [23]; this follows from the fact that [23] showed that SQ algorithms are precisely what can be computed in the local privacy model.
- Second, and perhaps even more importantly, it gives unconditional lower

[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 [25].

bounds on the *run-time* of any query release algorithm in the SQ model. To our knowledge, this class includes almost all privacy-preserving algorithms developed to date, including the median mechanism [29] and multiplicative weights mechanism [18],[2] which both run in time *exponential* in the dimension of the data. In fact, in their most natural implementations, each of these algorithms uses the trivial solution to the (non–privacy-preserving) query release problem—evaluate each query individually—as an intermediate step, and this is one (though not the only) bottleneck to achieving polynomial-time algorithms. One may have been optimistic that for simple classes of queries, such as conjunctions, this bottleneck could be eliminated by asking a small subset of the queries. However, our results show that this is not the case, and that even an efficient solution to the non–privacy-preserving query release problem will require new techniques. In particular, our lower bound shows that any mechanism that ultimately interacts with the private data only by making some number of linear queries and adding Laplace noise cannot solve the conjunction release problem in polynomial 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 the class of 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. Specifically, our upper bound reduces the problem of query release to the problem of learning the function that maps queries to their answers.

1.1. Overview of our results. In this section we give an informal statement of our theorems with pointers to the relevant sections. (Precise definitions of terms used in this section can be found in section 2.) We begin by proving a strong correspondence between the general query release problem and agnostic learning complexity in the SQ model in section 3.

Informal Theorem 1.1 (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.

As a consequence of this theorem, lower bounds for agnostic learning in the SQ model immediately imply lower bounds for query release in the SQ model. Thus, by applying a result of Feldman [15], we can obtain a corollary saying that there is no algorithm that makes a polynomial number of statistical queries and releases answers to all monotone Boolean conjunction queries with any *subconstant* additive error.

While both reductions preserve the query complexity of the problem, neither reduction preserves run-time. We also note that our equivalence characterization is more general than what we stated: we show how to use any algorithm A for agnostic learning of a class C as a black-box to release the answers to all queries in a class C, and vice versa. Since the reductions are black-box, any restrictions we place on the agnostic learner's access to the database will also be satisfied by the resulting query release algorithm, and vice versa. In our discussion, we have focused on the restriction

[2]A notable exception is the private parity-learning algorithm of [23], which explicitly escapes the SQ model.

that the algorithm A make only a polynomial number of statistical queries. Another interesting restriction would be that A be differentially private. In this case, we show that for every class C, the privacy cost of releasing the answers to C is not much larger than the privacy cost of agnostically learning C, and vice versa, even when we do not restrict ourselves to the SQ model.

Next, in section 4, we prove a theorem on approximating submodular functions. (Submodularity is often thought of as a combinatorial analogue of convexity, and a precise definition appears in section 2.)

INFORMAL THEOREM 1.2 (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 oracle algorithm $\mathcal{A}^f$ with run-time $d^{O(\log(1/\beta)/\alpha^2)}$ that produces an approximation $h\colon \{0,1\}^d \to [0,1]$ such that $\mathbb{P}\mathrm{r}_{x\in\{0,1\}^d}\{|f(x) - h(x)| \leq \alpha\} \geq 1-\beta$. Moreover, $\mathcal{A}$ produces the desired approximation as long as its oracle approximates f pointwise to within an additive error $\tau = \Theta(\alpha)$.*

Then, in section 5 we use this approximation to submodular functions to give a differentially private release mechanism for Boolean conjunctions.

INFORMAL THEOREM 1.3 (differentially private query release for conjunctions). *Let $\alpha, \beta > 0$. There is an ε-differentially private algorithm $\mathcal{B}$ that, for every database D with $|D| \geq d^{O(\log(1/\beta)/\alpha^2)}/\varepsilon$, releases answers to all Boolean conjunction counting queries on D such that the additive error is at most α on at least a $1-\beta$ fraction of the queries. Moreover, the run-time of the algorithm is $d^{O(\log(1/\beta)/\alpha^2)}$.*

The guarantee in our theorem can be refined to show that our algorithm achieves, simultaneously for each $w \in \{1, 2, \ldots, d\}$, an α-approximation to a $1-\beta$ fraction of the set of w-way conjunctions (i.e., conjunctions of w terms). While our result is clearly weaker than achieving an at most α error on *all* conjunctions (an L_∞-type of guarantee), it is stronger than giving L_1 or L_2 error guarantees, which have been studied in the past [20]. Indeed, as long as $\beta \leq \alpha^p/2$, our guarantee is at least as strong as achieving error α in the L_p-norm. Moreover, from a practical point of view, it turns out that some privacy-preserving algorithms in the literature indeed require only the ability to answer *random* conjunction queries privately; see, e.g., [21].

1.2. Our techniques. Our characterization of the query complexity of the release problem in the SQ model uses the multiplicative weights method [27, 1]: similar to its application in [18], we maintain a distribution over the universe on which the queries are defined. Novel to this work is the observation that an agnostic learning algorithm for a class C can be used to find a query from C that distinguishes as much as possible between the true data set and our distribution. Such a query can be used to induce a significant *update step* in the multiplicative weights algorithm, and because of the regret guarantees of this algorithm, we know that it is not possible to induce very many significant update steps. Therefore, it must be that we have to call the agnostic learning algorithm only a small number of times before we find a distribution over database elements that well approximates *every* query of interest. We note that in addition to proving a relationship between agnostic learning and query release, this gives a new *offline* mechanism for private query release that can be instantiated with any agnostic learning algorithm.

Our release algorithm is based on a structural theorem about general submodular functions $f\colon 2^U \to [0,1]$ 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 [33] that Lipschitz submodular functions are *self-bounding*, which allows us to apply dimension-free concentration bounds for self-bounding functions [8, 9]. 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.

1.3. Related work.

Learning submodular functions. The problem of learning submodular functions was introduced by Balcan and Harvey [2]; their PAC-style definition was different from previously studied pointwise learning approaches by Goemans et al. [16] and Svitkina and Fleischer [30]. 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. They also give strong lower bounds and matching algorithmic results for nonproduct 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, nonmonotone) submodular functions to constant *additive* error. Moreover, our decomposition makes value queries to the submodular function, which are prohibited in the model studied by [2].

Information-theoretic characterizations in privacy. The work of Kasiviswanathan et al. [23] 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 [24].

Blum, Ligett, and Roth [7] 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 following question: Which classes of functions can be released using mechanisms that have run-time only poly-logarithmic in the size of the data universe and the class of interest? In particular, they asked whether 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. [6] and Feldman [15].

Lower bounds and hardness results. There are also several conditional lower bounds on the run-time of private mechanisms for solving the query release problem. Dwork et al. [14] showed that under cryptographic assumptions, there exists a class of queries that can be privately released using the inefficient mechanism of [7] 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 [32] extended this result to the class of conjunctions: they showed that under standard 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 [32] 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 and release the answers.

We circumvent the hardness result of [32] 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 [32].

We also prove a new unconditional lower bound on algorithms for privately releasing monotone conjunctions that applies to the class of algorithms that interact with the data using only statistical 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 run-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 run-time. In particular, Kasiviswanathan et. al. [22] showed that average error of $\Omega(1/\sqrt{n})$ is necessary for private mechanisms that answer all conjunction queries of constant size. De [11] extended this work and showed that $\Omega(1/\sqrt{n})$ error is necessary even if the mechanism is also allowed to introduce arbitrary error on a constant fraction of the queries. These results extend earlier results by Dinur and Nissim [12] showing that average error $\Omega(1/\sqrt{n})$ is necessary for random queries.

Interactive private query release mechanisms. Roth and Roughgarden [29] and Hardt and Rothblum [18] gave interactive private query release mechanisms that allow a data analyst to ask a large number of questions, while 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 were 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.

1.4. Subsequent work. Subsequent to the publication of the extended abstract of this work in the *Proceedings of the* 43*rd Annual ACM Symposium on Theory of Computing*, there have been several follow-up works, which we briefly summarize here. Most directly related to this work are two papers by Cheraghchi et al. [10]

and Raskhodnikova and Yaroslavtsev [28]. Cheraghchi et al. prove that submodular functions are *noise-stable*, which directly leads to an improved algorithm for learning submodular functions in our model, and as a result, an improved algorithm for privately releasing conjunctions with low average error. Specifically, they improve on the number of queries needed to learn a submodular function from our bound of $d^{O(1/\alpha^2)}$ to $d^{O(\log 1/\alpha)}$. Raskhodnikova and Yaroslavtsev also give an algorithm for learning submodular functions, based on the decomposition introduced in this paper. Their algorithm improves on both ours and the algorithm of [10] in a certain range of parameters, which allows them to make progress on the question of *testing* submodular functions.

There has also been subsequent work on the problems of privately releasing the classes of queries considered in this paper. Hardt, Rothblum, and Servedio [19] and Thaler, Ullman, and Vadhan [31] give algorithms for releasing width-k conjunctions, to both low average error and worst case error. Reference [19] shows that an efficient privacy-preserving PAC learning algorithm for a concept class related to C implies an efficient privacy-preserving algorithm for releasing answers to every query in C, which yields more efficient algorithms for several interesting concept classes. The current best result specifically for conjunctions, from [31], gives an algorithm for releasing width-k conjunctions with 1% worst case error, which runs in time $d^{O(\sqrt{k})}$ and requires a database of size $d^{O(\sqrt{k})}$. This result is incomparable to the one in the present paper: our algorithm runs in polynomial time even for superconstant k and requires only a polynomially sized database, but guarantees only low average error. In contrast, [31] runs in superpolynomial time and requires a superpolynomial sized dataset when $k = \omega(1)$, but guarantees low error on all conjunctions.

The problem of privately releasing graph cuts has also continued to attract attention. Gupta, Roth, and Ullman [17] considered the problem of noninteractively releasing the answers to all graph cuts (with worst case error bounds), and improved on the error bounds which follow from the general techniques for learning submodular functions presented in this paper. Subsequently, Blocki et al. [4] gave a mechanism for noninteractively releasing graph cut queries that improves on the error bounds of [17] for small cuts.

2. Preliminaries.

2.1. Databases 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 $D = (x_1, \dots, x_n)$ and $D' = (x'_1, \dots, x'_n)$ are called *adjacent* if they differ only in one entry. That is, there exists $i \in [n]$ such that for every $j \neq i$, $x_j = x'_j$.

A *counting query* (sometimes called a *linear 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 $q(D) = \frac{1}{n}\sum_{x\in D} q(x)$. We can think of the database D also as a distribution over X, where the probability of $x \in X$ is the fraction of entries in D that equal X. Hence, $q(D)$ can be written as $\mathbb{E}_{x\sim D}[q(x)]$.

2.2. Differential privacy and counting queries.

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 [13]). *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 $\Pr(\mathcal{M}(D) \in S) \le e^{\varepsilon} \Pr(\mathcal{M}(D') \in S)$.*

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 [13] for details).

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

In the context of differential privacy, the distribution $\mathcal{D}$ will typically be the uniform distribution over a data set D of size n, and hence a statistical query is the same as a counting query. 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. [5] and Kasiviswanathan et al. [23].

PROPOSITION 2.2. *Let A denote an algorithm that requires k queries of tolerance τ. Let $\mathcal{O}$ denote the oracle that outputs $\mathbb{E}_{x\sim D}[q(x)] + Lap(k/n\varepsilon)$. Then, the algorithm $A^{\mathcal{O}}$ obtained by using this SQ oracle satisfies ε-differential privacy, and with probability at least $1-\beta$, the oracle answers all q queries with error at most τ, provided that $n \ge \frac{k(\log k + \log(1/\beta))}{\varepsilon\tau}$.*

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 $\mathbb{P}\mathrm{r}(|Lap(\sigma)| \ge \tau) \le \exp(-\tau/\sigma)$. 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. ∎

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

DEFINITION 2.3 (query release). *Let C be a concept class. We say that an algorithm $\mathcal{B}$ (α,β)*-releases *C over a data set D if it outputs a data structure $\mathcal{B}(D)$ (which provides an answer $\mathcal{B}(D)_q$ for each query $q \in C$) such that*

$$\mathop{\mathbb{P}\mathrm{r}}_{q\sim C}\{|q(D) - \mathcal{B}(D)_q| \le \alpha\} \ge 1-\beta\,.$$

Unless otherwise specified, the distribution over C is the uniform distribution.

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 in the following sense.

DEFINITION 2.4. *An algorithm A (α,β)*-approximates *a function $f\colon 2^U \to [0,1]$ over a distribution $\mathcal{D}$ on subsets of U if*

$$\mathop{\mathbb{P}\mathrm{r}}_{S\sim\mathcal{D}}\{|f(S) - A(S)| \le \alpha\} \ge 1-\beta\,.$$

2.5. 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 2.5. *A function $f : 2^U \to \mathbb{R}$ is γ*-Lipschitz *if for every $S \subseteq U$ and $x \in U$, $|\partial_x f(S)| \le \gamma$.*

2.6. Concentration bounds for submodular functions. The next lemma was shown by Vondrak [33] building on concentration bounds for so-called self-bounding functions due to [8, 9].

LEMMA 2.6 (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*

$$\Pr_{S\sim\mathcal{D}}\{|f(S) - \mathbb{E}\, f(S)| \geq t\} \leq 2\exp\left(-\frac{t^2}{2(\mathbb{E}\, f(S) + 5t/6)}\right), \tag{2.1}$$

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

Observing that if f is a γ-Lipschitz map into $[0,1]$, then $\frac{1}{\gamma} f$ is 1-Lipschitz with expectation at most $1/\gamma$, we get the following corollary.

COROLLARY 2.7. *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*

$$\Pr_{S\sim\mathcal{D}}\{|f(S) - \mathbb{E}\, f(S)| \geq \gamma t\} \leq 2\exp\left(-\frac{t^2}{2(1/\gamma + 5t/6)}\right), \tag{2.2}$$

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

3. Equivalence between agnostic learning and query release.

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

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

Recall that we use $\mathrm{STAT}_\tau(\mathcal{D})$ to denote the *statistical query oracle for distribution* $\mathcal{D}$ that takes as input a predicate $q\colon X \to \{0,1\}$ and returns a value v such that $|v - \mathbb{E}_{x\sim\mathcal{D}}[q(x)]| \leq \tau$, where τ is the *tolerance*.

DEFINITION 3.1 (strong agnostic SQ-learning). *Let C be a concept class, and let $\gamma, \tau > 0$ and $0 < \alpha \leq 1/2$. An algorithm $\mathcal{A}$ is an (α, γ, τ)-*weak agnostic SQ learner *for C if for every distribution $\mathcal{D}$, the algorithm $\mathcal{A}^{\mathrm{STAT}_\tau(\mathcal{D})}$ outputs a predicate $q\colon X \to \{0,1\}$ such that*

$$\Pr_{(x,b)\sim\mathcal{D}}\{q(x) = b\} \geq \sup_{q^*\in C}\left(\Pr_{(x,b)\sim\mathcal{D}}\{q^*(x) = b\}\right) - \alpha\,, \tag{3.1}$$

with probability at least $1 - \gamma$.

DEFINITION 3.2 (weak agnostic SQ-learning). *Let C be a concept class, and let $\gamma, \tau > 0$ and $0 < \beta < \alpha \leq 1/2$. An algorithm $\mathcal{A}$ is an $(\alpha, \beta, \gamma, \tau)$-*weak agnostic SQ learner *for C if, for every distribution $\mathcal{D}$ for which there exists $q^* \in C$ satisfying*

$$\Pr_{(x,b)\sim\mathcal{D}}\{q^*(x) = b\} \geq 1/2 + \alpha\,, \tag{3.2}$$

the algorithm $\mathcal{A}^{\mathrm{STAT}_\tau(\mathcal{D})}$ outputs a predicate $q\colon X \to \{0,1\}$ such that

$$\mathbb{P}\mathrm{r}_{(x,b)\sim\mathcal{D}}\{q(x)=b\} \geq 1/2 + \beta\,, \tag{3.3}$$

with probability at least $1-\gamma$.

Algorithm 1. Multiplicative weights update.

```
Let $D_0$ denote the uniform distribution over $X$.
**For** $t = 1,\dots,T = \lceil 8\log|X|/\beta^2\rceil + 1$:
  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 $\mathbf{1}_{\{q_t^+(x)=b\}}$, and let $v_t^-$ be the value returned by $\mathrm{STAT}_\tau(A_t^-)$ on the
  query $\mathbf{1}_{\{q_t^-(x)=b\}}$. Let $v_t = \max\{v_t^+, v_t^-\} - 1/2$, and let $q_t$ be the corresponding
  query.
  **If:**
  (3.4)      $v_t \leq \frac{\beta}{2} - \tau\,,$
  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 $c \in C$.
```

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, for any concept class C, a weak agnostic learner for C in the SQ model is sufficient to release C in the SQ model. (See Definition 2.3 for what it means to release C, and also recall that a database $D \in X^n$ can be viewed as a distribution over X.)

THEOREM 3.3. *Let C be a concept class. Let algorithm $\mathcal{A}$ be a $(\alpha/2,\beta,\gamma,\tau)$-weak agnostic SQ learner for C with $\tau \leq \beta/8$. Then there exists an algorithm $\mathcal{B}$ that, for every database D, invokes $\mathcal{A}$ at most $T = 8\log|X|/\beta^2$ times and $(\alpha,0)$-releases C over data set D with probability at least $1-T\gamma$.*

The proof strategy is as follows. We will start from the distribution D_0 being the uniform distribution over X. We will then construct a short sequence of distributions $D_1, D_2, \dots, D_T$ such that no concept in C can distinguish between the database D and the distribution D_T with advantage α. (Here, we say that q distinguishes D and D_T with advantage α if $|\mathbb{E}_{x\sim D}[q(x)] - \mathbb{E}_{x\sim D_T}[q(x)]| > \alpha$.) Each distribution D_t is obtained from the previous one using a multiplicative weights approach as in [18] and with the help of the learning algorithm that is provided by 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 (with advantage at least β). 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 technical reasons we also need to consider the distribution with 0 and 1 flipped.) Once we obtain q_t, we can use it as a penalty function in the update rule of the multiplicative weights method. Roughly speaking, if $\mathbb{E}[q_t(D_t)] < \mathbb{E}[q_t(D)] - \beta$, then we obtain D_{t+1} by increasing the probability of elements x such that $q_t(x) = 1$. This has the effect of bringing D and D_t closer in relative entropy distance. 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. The precise details of how to obtain the distributions D_i appears in Algorithm 1, and the proof of correctness appears below in section 3.1.

3.1. Proof of Theorem 3.3.

For query $q\colon X \to \{0,1\}$, define the error on q at time t to be

$$\Delta_t(q) := \mathop{\mathbb{E}}_{x\sim D} q(x) - \mathop{\mathbb{E}}_{x\sim D_t} q(x). \tag{3.5}$$

Observe that we want to reach a distribution D_t with absolute error $|\,\Delta_t(q)\,| \le \alpha$ for all queries q. To this end, consider the distributions A_t^+ and A_t^- defined in Algorithm 1. We begin by relating the probability that a query q predicts label b for point x under these distributions to the error on q at time $t-1$.

Lemma 3.4. *For any $q\colon X \to \{0,1\}$,*

$$\mathop{\mathbb{P}\mathrm{r}}_{(x,b)\sim A_t^+}\{q(x) = b\} = \frac{1}{2} + \frac{1}{2}\Delta_{t-1}(q), \tag{3.6}$$

$$\mathop{\mathbb{P}\mathrm{r}}_{(x,b)\sim A_t^-}\{q(x) = b\} = \frac{1}{2} - \frac{1}{2}\Delta_{t-1}(q). \tag{3.7}$$

Proof. For the first equality, observe that

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

For the second, the proof is similar:

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

For two distributions P, Q on a universe X, we define the *relative entropy*

$$\mathrm{RE}(P||Q) := \sum_{x\in X} P(x)\log(P(x)/Q(x))$$

and consider the potential function

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

Elementary properties of the relative entropy function imply that $\Psi_0 \le \log|X|$. Moreover, for all $t \ge 0$, $\Psi_t \ge 0$. We argue that in every step in which we do not terminate, the potential decreases by at least $\beta^2/8$. Hence, there can be at most $8\log|X|/\beta^2$ steps before we reach a distribution that satisfies the termination condition (3.4).

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 for the update rule in the algorithm. We use the following lemma from [18].

LEMMA 3.5 (see [18, Lemma IV.2]).

$$\Psi_{t-1} - \Psi_t \ge \eta\left|\mathop{\mathbb{E}}_{x\sim D} q_t(x) - \mathop{\mathbb{E}}_{x\sim D_{t-1}} q_t(x)\right| - \eta^2 = \eta\,|\Delta_{t-1}(q_t)| - \eta^2\,. \tag{3.8}$$

Define

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

The equality of the two expressions above follows from (3.6) and (3.7). Moreover, by equality (3.6), we get that

$$\mathrm{opt}_t = \frac{1}{2}\sup_{q\in C}|\,\Delta_{t-1}(q)\,|.$$

And hence if we reach a time t such that $\mathrm{opt}_t \le \alpha/2$, we are done (since then $\sup_{q\in C}|\Delta_{t-1}| \le \alpha$, and D and D_t have a small difference on all queries). So assume that $\mathrm{opt}_t \ge \alpha/2$.

By our assumption of the $\mathcal{A}$ being a weak learner, if $\mathrm{opt}_t \ge \alpha/2$, then (with probability at least $1-\gamma$) the algorithm $\mathcal{A}$ produces a concept q_t such that

$$\left|\mathop{\mathbb{P}\mathrm{r}}_{(x,b)\sim A_t^+}\{q_t(x)=b\} - \frac{1}{2}\right| \ge \beta\,. \tag{3.9}$$

For the remainder of the proof we assume that our algorithm returns a concept satisfying (3.9) in every stage for which $\mathrm{opt}_t \ge \alpha/2$. This is justified, because by a union bound over the stages of the algorithm, this event occurs with probability at least $1 - T\gamma$. Again, using (3.6), the inequality (3.9) implies that $|\,\frac{1}{2}\Delta_{t-1}(q_t)\,| \ge \beta$. Plugging back into (3.8),

$$\begin{aligned}\Psi_{t-1} - \Psi_t &\ge \eta\,|\,\Delta_{t-1}(q_t)\,| - \eta^2\\ &\ge \eta\,(2\beta) - \eta^2 \ge \beta^2 - \beta^2/4 = 3\beta^2/4\,,\end{aligned}$$

where we used $\eta = \beta/2$.

In fact, as long as $|\,\Delta_{t-1}(q_t)\,| \ge \beta - 2\tau$, the potential function goes down by $\beta^2/8$, and we make progress. (This is where we use $\tau = \beta/8$.) So finally, when $|\,\Delta_{t-1}(q_t)\,| < \beta - 2\tau$, this will ensure that $v_t < \frac{1}{2}(\beta - 2\tau) + \tau = \beta/2$ (taking into account the tolerance of τ), and the algorithm stops.

Moreover, when the algorithm terminates, $v_t \leq \beta/2$, and hence $|\Delta_{t-1}(q_t)| < 2(v_t + \tau) \leq 2\beta$. In turn, we get $\mathrm{opt}_t < \alpha/2$, or equivalently $\sup_{q \in C} |\Delta_{t-1}(q)| \leq \alpha$. This shows that the algorithm $(\alpha, 0)$-releases the concept class C over data set D in $8 \log |X| / \beta^2$ steps with probability $1 - T\gamma$.

Since we take a union bound at the end, the failure probability of the release algorithm is $T\gamma$—i.e., it grows linearly in the number of calls we make to the learning algorithm. However, this is not necessary and is only done for ease of exposition: we could have driven down the probability of error in each stage by independent repetitions of the agnostic learner.

3.2. From release to agnostic learning.

This relationship between release and agnostic learning also holds in the reverse direction as well. Again, recall that a database D can be viewed as a distribution over X, and *vice versa*.

THEOREM 3.6. *Let C be a concept class. Assume there exists an algorithm $\mathcal{B}$ that for every database D $(\alpha, 0)$-releases C over database D with probability $1 - \gamma$ and accesses the database D using at most k oracle accesses to $\mathrm{STAT}_\tau(D)$. Then there is an algorithm $\mathcal{A}$ that for every distribution D makes $2k$ queries to $\mathrm{STAT}_\tau(D)$ 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 $\mathrm{STAT}_\tau(D)$ to simulate oracles $\mathrm{STAT}_\tau(Y)$ and $\mathrm{STAT}_\tau(N)$ that condition the queried concept on the label as follows: $\mathrm{STAT}_\tau(Y)$, when invoked on concept q, returns an approximation to $\mathbb{P}\mathrm{r}_{x \sim D}[q(x) = 1 \wedge (x \in Y)]$, and $\mathrm{STAT}_\tau(N)$ returns an approximation to $\mathbb{P}\mathrm{r}_{x \sim D}[q(x) = 1 \wedge (x \in N)]$. We can simulate a query to either oracle using only one query to $\mathrm{STAT}_\tau(D)$ by running $\mathrm{STAT}_\tau(D)$ on the predicate $(q(x) = 1 \wedge (x \in Y))$ or $(q(x) = 1 \wedge (x \in N))$.

Now run the data release algorithm $\mathcal{B}(Y)$ to obtain answers $(a_q^Y)_{q \in C}$, and similarly run $\mathcal{B}(N)$ to obtain answers $(a_q^N)_{q \in C}$. Note that this takes at most $2k$ oracle queries using the simulation described above, by our assumption on $\mathcal{B}$. Now the accuracy guarantees for $\mathcal{B}$ (and a union bound) imply that for a $1 - 2\gamma$ fraction of the concepts $q_i \in C$: $|q_i(Y) - a_i^Y| \leq \alpha$ and $|q_i(N) - a_i^N| \leq \alpha$. Let $q^* = \arg\sup_{q_i \in C}(a_i^Y - a_i^N)$. Observe that $q^*(D) \geq \sup_{q \in C} q(D) - 2\alpha$, and so we have agnostically learned C (in the strong sense) up to error 2α. ∎

Consider the setting where $X = \{0,1\}^d$ (i.e., elements are d-bit vectors). For $S \subset \{1, 2, \ldots, d\}$, the monotone conjunction associated with S is the function $f_S : \{0,1\}^d \to \{0,1\}$, where $f_S(x) := \wedge_{i \in S} x_i$. Feldman proves that even monotone conjunctions cannot be agnostically learned to subconstant error with polynomially many statistical queries.

THEOREM 3.7 (see [15]). *Let C be the class of all 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 $\mathrm{STAT}_{1/k(d)}$.*

COROLLARY 3.8. *For any polynomial $k(d)$ in d, no algorithm that makes $k(d)$ statistical queries to a database of size $k(d)$ can release the class of all monotone conjunctions to error $o(1)$.*

Note that formally, Corollary 3.8 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, it can be shown that the lower bound from Corollary 3.8 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 (run-time $\mathrm{poly}(2^d)$)

that makes poly(d) statistical queries and releases random conjunctions up to a small additive error. We summarize this statement in the following informal theorem.

INFORMAL THEOREM 3.1. *For any concept class C, there is an (inefficient) algorithm that makes* $\mathrm{poly}(\log|X|, 1/\alpha)$ *statistical queries and* (α, α)*-releases* C. *In particular, this algorithm releases monotone conjunctions with average error* $1/\mathrm{poly}(d)$ *using* $\mathrm{poly}(d)$ *statistical queries.*

The algorithm is similar to Algorithm 1 in that it uses up to T rounds of the multiplicative weights update method. But instead of invoking an agnostic learner for C at each step, it will simply sample $O(\log(T)/\alpha)$ random queries and check to see whether any of these queries are answered inaccurately by the current distribution. If there is no such query, we terminate, and, with high probability, the current distribution D_t will give accurate answers to most queries. Otherwise we proceed and use the inaccurate query as a penalty function for the multiplicative weights update. The analysis in [18] or above shows that this can proceed for only $T = O(\log|X|/\alpha^2)$ rounds before terminating and thus makes at most $\mathrm{poly}(\log|X|, 1/\alpha)$ queries. We omit a formal analysis of this algorithm.

We also remark that the proofs of Theorems 3.3 and 3.6 are not particular to the SQ 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, we can give the following informal theorem.

INFORMAL THEOREM 3.2. *For any concept class C, if there is an ε-differentially private algorithm that agnostically learns C, then there is a* $(\varepsilon T \approx \varepsilon \log|X|)$*-differentially private algorithm that releases C.*

The algorithm is again similar to Algorithm 1, where we use the private agnostic learner to find queries for the multiplicative weights update. Privacy of the query release algorithm will follow from the assumed privacy of the agnostic learner and standard composition properties of differential privacy.

4. Approximating submodular functions.

Our query release algorithm for conjunctions is based on a general structural theorem about approximating submodular functions using a small data structure, which we present in this section.

The idea behind this structural result is simple: suppose a submodular function $f\colon 2^U \to [0,1]$ were itself γ-Lipschitz—i.e., for each $S \subset U$ and $e \in U$, $|f(S \cup \{x\}) - f(S)| \leq \gamma$. Then one can show that f is sharply concentrated around its mean and hence can be approximated well by a constant function $\mathbb{E} f$. (The notion of approximation is that of Definition 2.4.) If f is not Lipschitz, we show we can decompose its domain into a small collection of subregions (each of these a combinatorial rectangle), such that f restricted to any of these subregions is γ-Lipschitz and hence well approximated by its mean. How many subregions do we need? Loosely speaking, submodularity and boundedness ensure that for any set S there exists a set T with at most $1/\gamma$ elements in it such that, conditioning on the elements in T, the rest of the elements in S now have low marginal value—i.e., $\max_{x \in S} |f(T \cup \{x\}) - f(T)| \leq \gamma$. This suggests that $|U|^{O(1/\gamma)}$ subregions should suffice (one for each possible choice of this set T), which is indeed what we show.

In the rest of the section, we first give the construction for bounded monotone submodular functions, followed by its guarantee of correctness (even when we obtain noisy answers for queries to f) and then the extension to arbitrary bounded submodular functions. This construction will be the main ingredient in our algorithm for releasing conjunctions, which follows in section 5. In the discussion below, we assume that we are given *value queries* to the submodular function $f\colon 2^U \to [0,1]$ — i.e., there is a *value oracle* that, given a set $S \subseteq U$, returns the value $f(S)$. Initially we assume this value is given precisely, but we then extend our results to assume only the existence of a τ-*tolerant* value oracle for f for some tolerance parameter $\tau > 0$, which is guaranteed to return only a value $v \in f(S) \pm \tau$.

4.1. Monotone submodular functions. To demonstrate the basic ideas, we begin with a simple existential construction for monotone submodular functions. We then refine this in various ways over the course of the section.

Consider the collection $\mathcal{I}_0$ of all subsets of the universe U of size at most $\lfloor 1/\gamma \rfloor$. Clearly, $|\mathcal{I}_0| = |U|^{O(1/\gamma)}$. For each $B \subseteq U$, define the set $V(B)$ as $\{x \in U \mid \partial_x f(B) \leq \gamma\}$. (Recall that $\partial_x f(B) = f(B \cup \{x\}) - f(B)$.) Observe that $B \subseteq V(B) \subseteq U$. For each $B \subseteq U$, define the function $g^B\colon 2^{V(B)} \to [0,1]$ as

$$g^B(S) := f(B \cup S)\,. \tag{4.1}$$

Claim 4.1. *The function $g^B\colon 2^{V(B)} \to [0,1]$ is monotone submodular and γ-Lipschitz.*

Proof. Monotonicity of g^B is inherited from that of f. To establish submodularity, we use the characterization of submodularity from Fact 2.1—for $T \subseteq S \subseteq V(B)$, $\partial_x g^B(S) = \partial_x f(B \cup S) \leq \partial_x f(B \cup T) = \partial_x g^B(T)$, where the inequality is by submodularity of f.

Now, to prove the Lipschitz property, observe that the definition of $V(B)$ implies that $\partial_x f(B) \leq \gamma$ for every $x \in V(B)$. Now $\partial_x g^B(S) = \partial_x f(B \cup S) \leq \partial_x f(B) \leq \gamma$, where the first inequality again uses the submodularity of f. Finally, by monotonicity $\partial_x g^B(S) \geq 0$, and so $|\,\partial_x g^B(S)\,| \leq \gamma$ as desired. ∎

We now claim that $f(S)$ can be approximated by one of the functions $\{g^B \mid B \in \mathcal{I}_0\}$. One simple way is the following: start with $B = \emptyset$. And while there exists an $x \in S$ such that $\partial_x f(B) > \gamma$, pick an arbitrary such x and set $B \leftarrow B \cup \{x\}$. So eventually $\partial_x f(B) \leq \gamma$ for all $x \in S$, and hence $S \subseteq V(B)$. And by construction, $B \subseteq S$. Note that $|B| \leq 1/\gamma$, because the value of $f(B)$ increases by γ every time an element is added to B, and this value cannot go beyond 1 (since f is bounded). Hence $B \in \mathcal{I}_0$ and the function g^B is well defined.

Claim 4.2. *For any S, if $B \subseteq U$ satisfies $B \subseteq S \subseteq V(B)$, then $g^B(S) = f(S)$.*

Proof. By definition, $g^B(S) = f(B \cup S)$, but since $B \subseteq S$, this equals $f(S)$. ∎

We can summarize our progress so far.

Lemma 4.3 (structure lemma, version 0). *Given any monotone submodular function $f\colon 2^U \to [0,1]$ and a parameter $\gamma > 0$, there exists a collection of functions $\mathcal{G}_0 := \{g^B \mid B \in \mathcal{I}_0\}$ such that the following hold:*

1. (Lipschitz) *For every $g^B \in \mathcal{G}_0$, g^B is submodular and γ-Lipschitz.*
2. (Completeness) *For every $S \subseteq U$, there exists a $g^B \in \mathcal{G}_0$ such that $f(S) = g^B(S)$.*
3. (Size) *The size of $\mathcal{G}_0$ is at most $|U|^{O(1/\gamma)}$.*

4.1.1. A canonical mapping and its inverse. Note that for any S, there could be many different B's such that $f(S) = g^B(S)$—indeed, this will hold for every $B \subseteq S$. Looking ahead, for each function g^B we want to associate a value μ_{g^B}, which is the expectation of g^B over some distribution on inputs and invokes concentration bounds for submodular functions to argue that this single value approximates $f(S)$ for a large number of inputs S. To approximate $f(S)$, we will use the value μ_{g^B} for some appropriately chosen B. In order for this to work, we will require a mapping from sets S to functions g^B with two properties: (1) Each S gets mapped to a unique value μ_{g^B}; otherwise we might introduce additional error by always looking at the "worst" g^B for S. (2) The distribution on inputs S, conditioned on S being mapped to g^B, must remain a product distribution; otherwise the concentration bounds won't apply.

To achieve these goals, we want to be able to pin down a particular B for S and, moreover, given any such set B, to determine efficiently which sets S map to B under this map. To achieve the former task, in this section we define a specific, deterministic "canonical" mapping $\phi(S)$, and to achieve the latter we will define a "witness" mapping $\rho(B)$.

Fix any total ordering $\prec$ on the elements of the universe U, and consider the canonical mapping $\phi\colon 2^U \to 2^U$ defined as follows.

Algorithm 2. Defining the canonical mapping $\phi(S)$.

```
let $j \leftarrow 0$, $B_j \leftarrow \emptyset$
for $x \in U$ (in the order given by $\prec$) do
    if $x \in S$ and $x \notin V(B_j)$ then $B_{j+1} \leftarrow B_j \cup \{x\}$, $j \leftarrow j+1$
return $\phi(S) := B_j$.
```

To visualize this process, one can imagine a "choice" tree $\mathcal{T}_0$ of depth $\lfloor 1/\gamma \rfloor$, where there is a one-to-one correspondence between vertices of the tree and sets in $\mathcal{I}_0$. The root node corresponds to the empty set; if vertex v corresponds to a set (call it B_v) from $\mathcal{I}_0$, the edge (v, w) corresponds to adding a single element x to B_v, with the constraint that all elements in B_v precede x in the order $\prec$. Hence, the $|U|$ children at depth 1 are the singleton sets, and the root-leaf path to the leaf v corresponds to choosing the elements in A_v in the order given by $\prec$. See Figure 4.1 for a visualization. The procedure for defining $\phi(S)$ constructs a single root-leaf path in this tree, where for each intermediate 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 $f(B_j)$ (and has not already been considered by $\phi(S)$). We will use $\mathbf{P}(S) = (\emptyset \subset B_1 \subset \cdots \subset \phi(S))$ to denote this path, which is the sequence of intermediate sets B_j in the construction of $\phi(S)$. Given these observations, we can state the following useful facts about ϕ, whose correctness is directly from the definition of Algorithm 2.

FACT 4.1. *If $\phi(S) = B$, then $\mathbf{P}(S) = \mathbf{P}(B)$. Moreover, for every $S \subseteq U$, $\mathbf{P}(S) \subseteq \mathcal{I}_0$.*

CLAIM 4.4. *For every $S \subseteq U$, $\phi(S) \subseteq S \subseteq V(\phi(S))$. Hence $g^{\phi(S)}(S) = f(S)$ by Claim* 4.2.

Proof. Let $\mathbf{P}(S) = B_0 \subset B_1 \subset \cdots \subset \phi(S)$. Algorithm 2 ensures that $\phi(S) \subseteq S$ (by only selecting elements $x \in S$ to be included in $\phi(S)$). To prove $S \subseteq V(\phi(S))$, assume there exists $x \in S \setminus V(\phi(S))$—i.e., $\partial_x f(\phi(S)) > \gamma$. By submodularity we have $\partial_x f(B_j) \geq \partial_x f(\phi(S)) > \gamma$ for every intermediate set B_j. But $x \in S$ must have been considered by the algorithm for some B_j and hence would have been added. This would mean $x \in \phi(S)$, but then $\partial_x f(\phi(S)) = 0 < \gamma$, a contradiction. ∎

Now we turn to characterizing which sets S map to a certain set $B \in \mathcal{I}_0$. First, observe that if any set S maps to B, then B must also map to itself. Let us define

$$\mathcal{I}_\phi := \{B \in \mathcal{I}_0 \mid \phi(B) = B\}.$$

The collection $\mathcal{I}_\phi$ is defined in terms of the canonical mapping ϕ. However, to reduce notational clutter we will drop the subscript and simply write $\mathcal{I}$. Next we define the

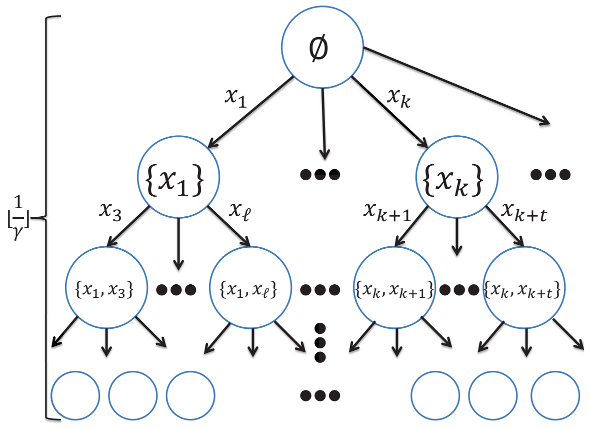

FIG. 4.1. *A visualization of the mapping defined by Algorithm* 2.

witness mapping ρ. To see the idea behind this witness map, consider the choice tree $\mathcal{T}_0$ used to visualize Algorithm 2, and keep only the nodes corresponding to the sets in $\mathcal{I}$, with the new tree being called $\mathcal{T}$. Consider a set $B \in \mathcal{I}$ and its associated path $\mathbf{P}(B)$ and consider all the elements we had to "reject" on the way from the root to B on this path: "rejected" elements are those that had high marginal value with respect to the current set when they were considered, but still do not belong to $\phi(B) = B$. Since any set S such that $B = \phi(S)$ satisfies $\mathbf{P}(S) = \mathbf{P}(B)$ (Fact 4.1), and any set S that contains a rejected element would have taken a different path, the absence of these rejected elements "witness" the fact that $B = \phi(S)$. With this intuition, we define the map $\rho(B)$ as follows.

Algorithm 3. Defining the witness mapping $\rho(B)$.

```
let $j \leftarrow 0$, $B_j \leftarrow \emptyset$, $R \leftarrow \emptyset$
for $x \in U$ (in the order given by $\prec$) do
    if $x \notin V(B_j)$ and $x \notin B$ then $R \leftarrow R \cup \{x\}$
    if $x \notin V(B_j)$ and $x \in B$ then $B_{j+1} \leftarrow B_j \cup \{x\}$, $j \leftarrow j + 1$
    if $B_j = B$ then terminate for loop.
return $\rho(B) := R$.
```

CLAIM 4.5. *A set $S \subseteq U$ satisfies $\phi(S) = B$ if and only if $B \in \mathcal{I}$, $B \subseteq S \subseteq V(B)$, and $S \cap \rho(B) = \emptyset$.*

Proof. Suppose $\phi(S) = B$; then Fact 4.1 implies that $\phi(B) = B$ and so $B \in \mathcal{I}$. Moreover, Claim 4.4 gives us that $B \subseteq S \subseteq V(B)$. For the third property, observe that every set S such that $B = \phi(S)$ has $\mathbf{P}(S) = \mathbf{P}(B)$ (by Fact 4.1). Say $\mathbf{P}(B) = (B_0 \subset B_1 \subset \cdots \subset B)$, and suppose there is an element $x^* \in S \cap \rho(B)$. Since it was added to $\rho(B)$ by Algorithm 3, there must be a set B_j such that $x^* \notin V(B_j)$ and $x^* \notin B$. Since the paths for S and B are identical, Algorithm 2 would encounter the set B_j and element $x^* \in S$ and would add it. Then $x^* \in \phi(S)$, contradicting that $B = \phi(S)$.

Now for the converse. For the sake of contradiction, suppose S and B satisfy the three properties, but a set $B' \neq B$ such that $\phi(S) = B'$. There exists an element $x \in B \triangle B'$, and we consider the minimal such x under $\prec$. Since $B \in \mathcal{I}$, it holds that $\phi(B) = B$, and by Fact 4.1 and the assumption that $\phi(S) = B'$, we get that $\phi(B') = B'$. Let $\mathbf{P}(B) = (\emptyset \subset B_1 \subset \cdots \subset B)$ and $\mathbf{P}(S) = \mathbf{P}(B') = (\emptyset \subset B'_1 \subset \cdots \subset B')$. By the minimality of x, there must be j such that $B_i = B'_i$ for all $i \leq j$, but $x \in B_{j+1} \triangle B'_{j+1}$. There are two cases to consider:

1. $B' \subset B$. Thus $x \in B \backslash B'$. Moreover, since $B \subseteq S$, $x \in S$. Consequently, when x was considered in the execution of $\phi(S)$, the current set was B'_j and x was not added; hence it must have been 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. $B' \not\subset B$. Thus $x \in B' \setminus B$. Since $B' = \phi(S) \subseteq S$ (the latter containment by Claim 4.4), $x \in S$. Moreover, since $x \in B'_{j+1}$, it holds that $x \notin V(B'_j) = V(B_j)$. Thus we have $x \notin V(B_j)$ and $x \notin B$, which would cause $x \in \rho(B)$ by Algorithm 3. Thus $S \cap \rho(B) \neq \emptyset$, a contradiction.

This proves the desired equivalence. ∎

Finally, on the algorithmic front we have the following: given oracle access to f, one can construct the tree $\mathcal{T}$ and, for each $B \in \mathcal{I}$, annotate the node in $\mathcal{T}$ corresponding to the set B with the sets $V(B)$ and $\rho(B)$, all in time $|U|^{O(1/\gamma)}$. Note that the tree gives a representation of the map ϕ: given a set $S \subseteq U$ one can walk down this tree in linear time to find $\phi(S)$. The following lemma summarizes our discussion so far.

LEMMA 4.6 (structure lemma, version 1). *Given any monotone submodular function $f\colon 2^U \to [0,1]$ and a parameter $\gamma > 0$, there exist $\mathcal{G} := \{g^B \mid B \in \mathcal{I}\}$ and maps $\phi\colon 2^U \to \mathcal{I}$ and $\rho\colon 2^U \to 2^U$ satisfying the following:*

1. (Lipschitz) *For every $g^B \in \mathcal{G}$, g^B is submodular and γ-Lipschitz.*
2. (Completeness) *For every $S \subseteq U$, $g^{\phi(S)}(S) = f(S)$.*
3. (Uniqueness) *For every $B \in \mathcal{I}$, a set $S \subseteq U$ has $\phi(S) = B$ if and only if $B \subseteq S \subseteq V(B)$ and $S \cap \rho(B) = \emptyset$.*
4. (Size) *The size of $\mathcal{G}$ is $|U|^{O(1/\gamma)}$. Moreover, given oracle access to f, we output a representation of V, ϕ, ρ in time $|U|^{O(1/\gamma)}$.*

4.1.2. An approximation theorem using value queries. Since each of the monotone submodular functions $\{g^B \mid B \in \mathcal{I}\}$ is γ-Lipschitz, we will use Corollary 2.7 to show that they are well approximated under product distributions by their expectations. But how does this translate back to the quality of the approximation for f? It is here that we use the careful choice of ϕ: the intuition is that the sets S that map to any particular B form a subcube of the Boolean cube $\{0,1\}^{|U|}$, and that this subcube behaves well under product distributions. Below, we give the details.

Recall that we would like an (α, β)-approximation of the function f under the product distribution $\mathcal{D}$. We start off with the representation of f guaranteed by Lemma 4.6, choosing $\gamma = \frac{\alpha^2}{4\ln(2/\beta)}$. For each $B \in \mathcal{I}$, recall that the sets mapped to it are precisely $\{B \cup T \mid T \subseteq V(B) \setminus \rho(B)\}$. Hence, we represent the function g^B by its expectation

$$\mu_{g^B} := \mathop{\mathbb{E}}_{A \sim D_{V(B)\setminus\rho(B)}} [g^B(A)],$$

where we take the expectation over the product distribution $\mathcal{D}$ restricted to $V(B) \setminus \rho(B)$. (To avoid further clutter, we assume here that we compute the expectation exactly. Indeed, we can estimate this expectation to within an additive ε error with probability $1 - \delta$ by taking the sample mean of $O(\varepsilon^{-2} \log \delta^{-1})$ samples; we omit these standard details.) Finally, we output this data structure, which uses $\mu_{g^{\phi(S)}}$ as the estimate for $f(S)$, and claim that it (α, β)-approximates f on $\mathcal{D}$.

To show the approximation guarantee, using that $g^{\phi(S)}(S) = f(S)$ for all S, we get

$$\begin{aligned} &\Pr_{S\sim\mathcal{D}} \{|f(S) - h(S)| > \alpha\} \\ &\quad = \Pr_{S\sim\mathcal{D}} \left\{\left|g^{\phi(S)}(S) - \mu_{g^{\phi(S)}}\right| > \alpha\right\} \\ &\quad = \sum_{g^B\in\mathcal{G}} \Pr_{S\sim\mathcal{D}} \{B = \phi(S)\} \cdot \Pr_{S\sim\mathcal{D}} \left\{\left|g^B(S) - \mu_{g^B}\right| > \alpha \mid B = \phi(S)\right\} . \end{aligned} \tag{4.2}$$

And now using the structure of sets mapped to B given by Lemma 4.6 (property 3), we get

$$\Pr_{S\sim\mathcal{D}} \left\{\left|g^B(S) - \mu_{g^B}\right| > \alpha \mid B = \phi(S)\right\} = \Pr_{S\sim\mathcal{D}_{V(B)\setminus\rho(B)}} \left\{\left|g^B(S) - \mu_{g^B}\right| > \alpha\right\} .$$

Finally, applying the concentration inequality from Corollary 2.7 with $t = \alpha/\gamma$,

$$\Pr_{S\sim\mathcal{D}_{V(B)\setminus T(B)}} \left\{\left|g^B(S) - \mu_{g^B}\right| \ge \alpha\right\} \le 2\exp\left(-\frac{(\alpha/\gamma)^2}{2(1/\gamma + 5\alpha/6\gamma)}\right) \le 2\exp\left(-\frac{\alpha^2}{4\gamma}\right) = \beta, \tag{4.3}$$

where we used that $\alpha \le 1$ and our chosen setting of $\gamma = \frac{\alpha^2}{4\ln(2/\beta)}$. This gives our first approximation result, that for monotone submodular functions with (exact) value queries.

THEOREM 4.7. *For any* $\alpha, \beta \in (0, 1]$*, given exact oracle queries to a monotone submodular function* $f\colon 2^U \to [0, 1]$ *and a product distribution* $\mathcal{D}$ *on* U*, there is a randomized algorithm that outputs an* (α, β)*-approximation to* f *under distribution* $\mathcal{D}$*, with a run-time of* $|U|^{O(\alpha^{-2}\log\beta^{-1})}$.

In the next section, we will extend this result to noisy oracle queries and to potentially nonmonotone submodular functions.

4.1.3. Tackling noisy answers. When dealing with noisy answers, it will be useful to assume that the noisy oracle always returns the same answer to each query. Thus we can use $\tilde{f}$ to denote its answers and assume that we have exact oracle access to $\tilde{f}$. Note that $\tilde{f}(S)$ need not be submodular even if f is. However, since we can compute $\partial_x f(S)$ using two queries to f, we are guaranteed that for every $S \subseteq U$ and every $x \in U$,

$$|\partial_x\tilde{f}(S) - \partial_x f(S)| \le \gamma/6. \tag{4.4}$$

Thus, $\tilde{f}$ will be *approximately submodular* in the sense that for every $S \subseteq T \subseteq U$ and every $x \in U$,

$$\partial_x\tilde{f}(S) \ge \partial_x\tilde{f}(T) - \gamma/3.$$

Thus, most of the work of this section is to tweak the structure lemma for submodular functions to handle these approximately submodular functions.

Again, we begin with an existential statement. Consider the collection $\mathcal{I}_1$ of all subsets of the universe U of size at most $\lfloor 6/\gamma\rfloor$. Clearly, $|\mathcal{I}_1| = |U|^{O(1/\gamma)}$. We will again associate a function $g^B\colon 2^{V(B)} \to [0, 1]$ with each set $B \in \mathcal{I}_1$, but we will slightly change the definition of $V(B)$. For each $B \subseteq U$, define the set $V(B)$ as $\{x \in U \mid \partial_x\tilde{f}(B) \le 2\gamma/3\}$. The purpose of this modification is to ensure that whenever $\partial_x \tilde{f}(B) \le 2\gamma/3$, we also have $\partial_x f(B) \le \gamma$. That is, elements of $V(B)$ also have low marginal value with respect to the real function f. As before, observe that $B \subseteq V(B) \subseteq U$, and for each $B \subseteq U$, define the function $g^B \colon 2^{V(B)} \to [0,1]$ as

$$g^B(S) := f(B \cup S)\,. \tag{4.5}$$

CLAIM 4.8. *The function $g^B \colon 2^{V(B)} \to [0,1]$ is monotone submodular and γ-Lipschitz.*

Proof. The proof of monotonicity and submodularity are as in Claim 4.1.

To prove the Lipschitz property, observe that the definition of $V(B)$ implies that $\partial_x \tilde{f}(B) \le 2\gamma/3$ for every $x \in V(B)$. Now $\partial_x g^B(S) = \partial_x f(B \cup S) \le \partial_x f(B) \le \partial_x \tilde{f}(B) + \gamma/6 \le \gamma$, where the first inequality uses the submodularity of f. By monotonicity $\partial_x g^B(S) \ge 0$, and so $|\,\partial_x g^B(S)\,| \le \gamma$ as desired. ☐

We now claim that $f(S)$ can be approximated by one of the functions $\{g^B \mid B \in \mathcal{I}_1\}$. One simple way is the following: start with $B = \emptyset$. And while there exists an $x \in S$ such that $\partial_x \tilde{f}(B) > \gamma/3$, pick an arbitrary such x and set $B \leftarrow B \cup \{x\}$. So eventually $\partial_x f(B) \le \gamma$ for all $x \in S$, and hence $S \subseteq V(B)$. And by construction, $B \subseteq S$. Now we have to be slightly more careful when using $\tilde{f}$ in place of f. Note that the value of $\tilde{f}(B)$ increases by $\gamma/3$ every time an element is added to B, and by (4.4), the value of $f(B)$ also increases by $\gamma/6$. Since this value cannot go beyond 1, we have $B \le 6/\gamma$. Hence $B \in \mathcal{I}_1$, and the function g^B is well defined.

CLAIM 4.9. *For any S, if $B \subseteq U$ satisfies $B \subseteq S \subseteq V(B)$, then $g^B(S) = f(S)$.*

Proof. By definition, $g^B(S) = f(B \cup S)$, but since $B \subseteq S$, this equals $f(S)$. ☐

So far we have not made any progress beyond version 0 of our structural lemma (Lemma 4.3); however, by changing how we define the collection $\mathcal{I}_1$ and $V(B)$ to be in terms of $\tilde{f}$, rather than f, we are in a better position to recover the results of version 1 of the structure lemma (Lemma 4.6) using only noisy queries to f.

Again we want to pin down a specific mapping from inputs S to functions g^B and identify a way to determine which inputs map to any given function g^B. Before giving the new mappings ϕ and ρ, we define the sets $V'(B)$. For each $B \subseteq U$, define the set $V'(B)$ as $\{x \in U \mid \partial_x \tilde{f}(B) \le \gamma/3\}$.

Algorithm 4. Defining the canonical mapping $\phi(S)$.

```
let $j \leftarrow 0$, $B_j \leftarrow \emptyset$
for $x \in U$ (in the order given by $\prec$) do
    if $x \in S$ and $x \notin V'(B_j)$ then $B_{j+1} \leftarrow B_j \cup \{x\}$, $j \leftarrow j + 1$
return $\phi(S) := B_j$.
```

We can again visualize this process as walking down a tree $\mathcal{T}_1$ defined on the subsets $\mathcal{I}_1$, where the walk starts at the root node $\emptyset$ and terminates at a node B. Recall that we use $\mathbf{P}(S) = (\emptyset \subset B_1 \subset \cdots \subset \phi(S))$ to denote this path.

CLAIM 4.10. *For every $S \subseteq U$, $\phi(S) \subseteq S \subseteq V(\phi(S))$. Hence $g^{\phi(S)}(S) = f(S)$ by Claim* 4.9.

Proof. Let $\mathbf{P}(S) = (B_0 \subset B_1 \subset \cdots \subset \phi(S))$. The fact that $\phi(S) \subseteq S$ follows as in Claim 4.4. To see that $S \subseteq V(\phi(S))$, assume that there exists $x \in S \setminus V(\phi(S))$. By submodularity of f, and (4.4), we have

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

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 $x \in \phi(S)$. But then $\partial_x f(\phi(S)) = 0$, contradicting the fact that $x \notin V(\phi(S))$.

The fact that $g^{\phi(S)}(S) = f(S)$ follows as in the proof of Claim 4.4. ∎

The intuition for the witness mapping is also the same as in the case of exact value queries, but to match the new mapping ϕ, we will define ρ with V' in place of V.

Algorithm 5. Defining the witness mapping $\rho(B)$.

```
let $j \leftarrow 0$, $B_j \leftarrow \emptyset$, $R \leftarrow \emptyset$
for $x \in U$ (in the order given by $\prec$) do
    if $x \notin V'(B_j)$ and $x \notin B$ then $R \leftarrow R \cup \{x\}$
    if $x \notin V'(B_j)$ and $x \in B$ then $B_{j+1} \leftarrow B_j \cup \{x\}$, $j \leftarrow j + 1$
    if $B_j = B$ then terminate for loop.
return $\rho(B) := R$.
```

In the original analysis of ρ, we did not actually use the submodularity of f; thus we obtain the following claim "for free."

Claim 4.11. *A set $S \subseteq U$ satisfies $\phi(S) = B$ if and only if $B \in \mathcal{I}$, $B \subseteq S \subseteq V(B)$, and $S \cap \rho(B) = \emptyset$.*

We can now summarize our progress again.

Lemma 4.12 (structure lemma, version 2). *Given any monotone submodular function $f\colon 2^U \to [0,1]$ and a parameter $\gamma > 0$, there exist $\mathcal{G} := \{g^B \mid B \in \mathcal{I}\}$ and maps $\phi\colon 2^U \to \mathcal{I}$ and $\rho\colon 2^U \to 2^U$ that satisfy properties 1–4 of Lemma 4.6 and, moreover, can be computed using noisy queries to f with tolerance $\gamma/12$.*

4.1.4. An approximation theorem using noisy value queries. Given the results of the previous section, it is easy to extend Theorem 4.7 to handle noisy oracle queries. The algorithm for approximating f uses the oracle in two ways: (1) The oracle is used to compute the decomposition. Lemma 4.12 shows that for a sufficiently accurate oracle, a suitable decomposition can still be computed correctly. (2) The oracle is used to estimate the values μ_B. If the oracle is accurate to within $\pm\tau$, then the noisy answers can introduce only an additional error of $\pm\tau$. We can summarize this discussion in the following theorem.

Theorem 4.13. *For any $\alpha, \beta \in (0, 1]$, given noisy oracle queries to a monotone submodular function $f\colon 2^U \to [0,1]$ and a product distribution $\mathcal{D}$ on U, there is a randomized algorithm that outputs an (α, β)-approximation to f under distribution $\mathcal{D}$, with a run-time of $|U|^{O(\alpha^{-2}\log\beta^{-1})}$, as long as the tolerance is at most $\alpha^2/72\log(2/\beta)$.*

4.2. Nonmonotone submodular functions. For nonmonotone functions, we need a more refined argument. Recall that, in order to apply concentration inequalities, we needed to decompose f into a collection of functions $g\colon 2^V \to [0,1]$ that satisfied $|\partial_x g(S)| \le \gamma$. We did so by ensuring that $\partial_x g(S) \le \gamma$ and used monotonicity to guarantee that $\partial_x g(S) \ge 0$. Without monotonicity, we have to work a bit harder, since it may be the case that, for some S, x, we have $\partial_x g(S) < -\gamma$. To solve this problem, we consider the function $\overline{g}\colon 2^V \to [0,1]$ defined as $\overline{g}(S) = g(V \setminus S)$. The function $\overline{g}$ is also submodular and satisfies

$$\inf_{x \in V, S \subseteq V} \partial_x \overline{f}(S) = - \sup_{x \in V, S \subseteq V} \partial_x f(S)\,. \tag{4.6}$$

Thus, if $\partial_x g(S) \le \gamma$, we have $\partial_x \overline{g}(S) \ge -\gamma$. Now, we need only ensure that $\partial_x \overline{g}(S) \le \gamma$. But we have already shown how to decompose the domain of $\overline{g}$ into

Algorithm 6. Decomposition for submodular functions from noisy queries.

```
Input: Noisy oracle access to a submodular function $f\colon 2^U \to [0,1]$ with tolerance at most $\gamma/12$ and parameter $\gamma > 0$.
  Let $\prec$ denote an arbitrary ordering of $U$.
  Let $\mathcal{G}(f)$ be the collection of functions satisfying the conditions of Lemma 4.12 with oracle $f$ and parameter $\gamma$, and let $\phi_f, V_f, \rho_f$ be the associated mappings satisfying the conditions of Lemma 4.12.
  for $g^B \in \mathcal{G}(f)$ do
      Let $\mathcal{G}(B)$ be the collection of functions satisfying the conditions of Lemma 4.12 with oracle $f$ and parameter $\gamma$, and let $\phi_B, V_B, \rho_B$ be the associated mappings satisfying the conditions of Lemma 4.12.
  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 $g^{B,C}\colon 2^{V(B,C)} \to [0,1]$.
```

a small collection of functions that satisfy this condition! We now specify the algorithm more precisely.

First, we clarify notation. Earlier we had dropped the subscript f on the mappings ϕ, V, ρ, since there was only one choice of f. Now we use $\phi_f, V_f, \rho_f : 2^U \to 2^U$ to denote mappings associated with the input function f. The mapping ϕ_f then induces a collection of subdomains $\mathcal{I}_{\phi_f}$, and for each such domain $B \in \mathcal{I}_{\phi_f}$, we will obtain a new set of mappings $\phi_B, V_B, \rho_B : 2^B \to 2^B$.

The collection of functions returned by Algorithm 6 will satisfy the necessary Lipschitz property.

CLAIM 4.14. *For every $g^{B,C} \in \mathcal{G}$, it follows that $g^{B,C}$ is submodular and that $\sup_{x \in V(B,C), S \subseteq V(B,C)} |\partial_x g^{B,C}(S)| \le \gamma$.*

Proof. Submodularity follows directly from property 1 of Lemma 4.3. The same property of the lemma guarantees that for every $g^B \in \mathcal{G}(f)$ and $g^C \in \mathcal{G}(B)$, $\sup_{x \in V(B,C), S \subseteq V(B,C)} \partial_x g^C(S) \le \gamma$. Moreover, by (4.6), $\inf_{x \in V_f(B), S \subseteq V_f(B)} \partial_x \overline{g}^B(S) \ge -\gamma$. By combining these two facts, we have that $\sup_{x \in V(B,C), S \subseteq V(B,C)} |\partial_x g^{B,C}(S)| \le \gamma$. ∎

4.2.1. A canonical mapping and its inverse for nonmonotone submodular functions.

As in the case of monotone submodular functions, we obtain a collection $\mathcal{G} = \{g^{B,C}\}$ and need to give a procedure for associating inputs S with functions $g^{B,C}$ (and vice versa). We can build such a mapping from the mappings ϕ_f and $\{\phi_B\}$. The idea is to use $\phi_f(S)$ to find a function $g^B \in \mathcal{G}(f)$, then use $\phi_B(S)$ to find a function $g^C \in \mathcal{G}(B)$, and then associate S with (B, C).

Algorithm 7. Defining the canonical mapping $\phi(S)$.

```
  Given: $\phi_f$, $\{\phi_B\}_{g^B \in \mathcal{G}(f)}$
  Let: $B = \phi_f(S)$, $C = \phi_B(S)$
  Return: $\phi(S) := (B, C)$
```

CLAIM 4.15. *For every $S \subseteq U$, $\phi(S) \subseteq S \subseteq V(\phi(S))$. Hence $g^{\phi(S)}(S) = f(S)$.*

Proof. By Lemma 4.3, property 1, we have $B \subseteq S \subseteq V_f(B)$ and $C \subseteq S \subseteq V_B(C)$, so we conclude that $B, C \subseteq S \subseteq V(B, C)$. ∎

We also need to define $\rho(B,C)$, so that we can efficiently represent the collection of inputs S that are mapped to each function $g^{B,C}$. Here, the idea is to use ρ_f to determine the items that witness the fact that S cannot be mapped to g^B, and use ρ_B to determine the set of items that witness the fact that once S has been mapped to g^B, S cannot then be mapped to g^C. Then we take $\rho(B,C)$ to be the union of these two sets of witnesses.

Algorithm 8. Defining the witness mapping $\phi(B,C)$.

```
  Given: $\rho_f$, $\{\rho_B\}_{g^B \in \mathcal{G}(f)}$
  Return: $\rho(B,C) := \rho_f(B) \cup \rho_B(C)$
```

Claim 4.16. *A set $S \subseteq U$ satisfies $\phi(S) = (B,C)$ if and only if $B, C \subseteq S \subseteq V(B,C)$, and $S \cap \rho(B,C) = \emptyset$.*

Proof. By Lemma 4.6, property 3, we have that $\phi_f(S) = B$ if and only if $B \subseteq S \subseteq V_f(B)$ and $S \cap \rho_f(B) = \emptyset$. By the same lemma, we also have that $\phi_B(S) = C$ if and only if $C \subseteq S \subseteq V_B(C)$ and $S \cap \rho_B(C) = \emptyset$. So if we define $\rho(B,C) = \rho_f(B) \cup \rho_B(C)$, we can conclude that $\phi(S) = (B,C)$ if and only if $B, C \subseteq S \subseteq V(B,C)$ and $S \cap \rho(B,C) = \emptyset$.

Now it is clear that $\phi(S) = (B,C)$ if $\phi_f(S) = B$ and $\phi_B(S) = C$, which by property 3 of Lemma 4.6 necessitates that $B \subseteq S \subseteq V_f(B)$, $S \cap \rho_f(B) = \emptyset$, $C \subseteq S \subseteq V_B(S)$, and $S \cap \rho_B(C) = \emptyset$. We have already defined $V(B,C)$, and now we define $\rho(B,C) = \rho_f(B) \cup \rho_B(C)$. It is clear now that $\phi(S) = (B,C)$ if and only if $B, C \subseteq S \subseteq V(B,C)$ and $S \cap \rho(B,C) = \emptyset$. ☐

We can now state the final version of our structure lemma.

Lemma 4.17 (structure lemma, final version). *Given any submodular function $f\colon 2^U \to [0,1]$ and $\gamma > 0$, Algorithm 6 makes the following guarantee. There are maps $\phi\colon 2^U \to 2^U \times 2^U$ and $\rho\colon 2^U \times 2^U \to 2^U$ such that the following hold:*

1. (Lipschitz) *For every $g^{B,C} \in \mathcal{G}$, $g^{B,C}$ is submodular and satisfies*

$$\sup_{x \in V(B,C), S \subseteq V(B,C)} |\partial_x g^{B,C}(S)| \le \gamma\,.$$

2. (Completeness) *For every $S \subseteq U$, $\phi(S) \subseteq S \subseteq V(\phi(S))$ and $g^{\phi(S)}(S) = f(S)$.*
3. (Uniqueness) *For every $g^{B,C} \in \mathcal{G}$, $\phi(S) = (B,C)$ if and only if $B, C \subseteq S \subseteq V(B,C)$ and $S \cap \rho(B,C) = \emptyset$.*
4. (Size) *The size of $\mathcal{G}$ is at most $|\mathcal{G}| = |U|^{O(1/\gamma)}$. Moreover, given noisy oracle access to f with tolerance $\gamma/12$, we compute ϕ, V, ρ in time $|U|^{O(1/\gamma)}$.*

4.2.2. An approximation theorem for general submodular functions.

Our final version of the approximation theorem now follows from Lemma 4.17.

Theorem 4.18. *For any $\alpha, \beta \in (0,1]$, given noisy oracle queries to an arbitrary submodular function $f\colon 2^U \to [0,1]$ and a product distribution $\mathcal{D}$ on U, there is a randomized algorithm that outputs an (α,β)-approximation to f under distribution $\mathcal{D}$, with a run-time of $|U|^{O(\alpha^{-2}\log\beta^{-1})}$, as long as the tolerance is at most $\alpha^2/72\log(2/\beta)$.*

5. Applications to privacy-preserving query release.

In this section, we show how to apply our algorithm from section 4 to the problem of releasing monotone conjunctions over a Boolean database. In section 5.1, we also show how our mechanism can be applied to release the *cut function* of an arbitrary graph.

Let us now begin with monotone disjunctions. We will then extend the result to monotone conjunctions. Given our previous results, we need only 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 $d_S\colon \{0,1\}^d \to \{0,1\}$ by putting

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

Note that in contrast to section 4, here we use x to denote an element of $\{0,1\}^d$. Given a database D with elements from $\{0,1\}^d$, recall that $d_S(D) = \mathbb{E}_{x\in D}\, d_S(x)$. Let $F_{\text{Disj}} : \{0,1\}^d \to [0,1]$ be the function such that $F_{\text{Disj}}(S) := d_S(D)$.

LEMMA 5.1. *The function 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 4.7 with Proposition 2.2.

COROLLARY 5.2. *Let $\alpha, \beta, \varepsilon > 0$. There is a 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 $t(\alpha,\beta) = O(\alpha^{-2}\log(1/\beta))$.*

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 5.2 for the formal analysis.

Algorithm 9. 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 $\phi, V, \rho$ given
    by Lemma 4.12 on the function $F_{Disj}$ with parameter $\gamma$.
    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 $\mu_{g^B}$ for every
    $g^B \in \mathcal{G}$ as well as the mapping $\phi$. To evaluate any monotone disjunction query
    $d_S(D)$, compute $\mu_{g^{\phi(S)}}$.
```

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 5.2 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 4.13

directly. However, in Lemma 5.3 we show that for monotone submodular functions (such as F^D_{Disj}) the concentration of measure property required in the discussion of Theorem 4.13 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 [2].

Lemma 5.3. *Assume $f : 2^U \to \mathbb{R}$ is a monotone function, and S and T are chosen at random as above. Then,*

$$\Pr[f(T) \geq \tau] \leq 2 \Pr[f(S) \geq \tau], \tag{5.1}$$

$$\Pr[f(T) \leq \tau] \leq 2 \Pr[f(S) \leq \tau]. \tag{5.2}$$

Remark 5.1. Throughout this section we focus on the case of *monotone* disjunctions and conjunctions. Our algorithm can be extended to nonmonotone conjunctions/disjunctions as well. However, this turns out to be less interesting than the monotone case. Indeed, a random nonmonotone 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 nonmonotone conjunction of width w. We therefore omit the nonmonotone case from our presentation.

5.1. 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 5.1. *For any graph G, f_G is submodular.*

Lemma 5.4. *The decomposition from Lemma* 4.6 *constructs a collection of functions $\mathcal{G}$ of size $|\mathcal{G}| \leq 2^{2/\alpha}$.*

Proof. Let $u \in V$, and let $S \subset V$ 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 5.5. *Algorithm* 6 *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 4.13, by applying Lemma 5.4 and plugging in the size of the decomposition $\mathcal{G}$. The algorithm can then be made privacy-preserving by applying Proposition 2.2. ∎

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 [2].

REFERENCES

[1] S. Arora, E. Hazan, and S. Kale, *The multiplicative weights update method: A meta-algorithm and applications*, Theory Comput., 8 (2012), pp. 121–164.

[2] M.-F. Balcan and N. J. A. Harvey, *Learning submodular functions*, in Proceedings of the 43rd ACM Annual Symposium on Theory of Computing (STOC), 2011, pp. 793–802.

[3] 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, ACM, New York, 2007, pp. 273–282.

[4] J. Blocki, A. Blum, A. Datta, and O. Sheffet, *The Johnson-Lindenstrauss Transform Itself Preserves Differential Privacy*, preprint, arXiv:1204.2136v1 [cs.DS], 2012.

[5] 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, ACM, New York, 2005, pp. 128–138.

[6] 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, ACM, New York, 1994, pp. 253–262.

[7] 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, ACM, New York, 2008, pp. 609–618.

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

[9] S. Boucheron, G. Lugosi, and P. Massart, *On concentration of self-bounding functions*, Electron. J. Probab., 14 (2009), pp. 1884–1899.

[10] M. Cheraghchi, A. Klivans, P. Kothari, and H. K. Lee, *Submodular functions are noise stable*, in Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, 2012, pp. 1586–1592.

[11] A. De, *Lower bounds in differential privacy*, in Theory of Cryptography, Lecture Notes in Comput. Sci. 7194, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 321–338.

[12] I. Dinur and K. Nissim, *Revealing information while preserving privacy*, in Proceedings of the Twenty-Second ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, ACM, New York, 2003, pp. 202–210.

[13] C. Dwork, F. McSherry, K. Nissim, and A. Smith, *Calibrating noise to sensitivity in private data analysis*, in Theory of Cryptography, Lecture Notes in Comput. Sci. 3876, Springer, Berlin, 2006, pp. 265–284.

[14] 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 Theory of Computing, ACM, New York, 2009, pp. 381–390.

[15] V. Feldman, *A complete characterization of statistical query learning with applications to evolvability*, J. Comput. System Sci., 78 (2012), PP. 1444–1459.

[16] M. X. Goemans, N. J. A. Harvey, S. Iwata, and V. Mirrokni, *Approximating submodular functions everywhere*, in Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, 2009, pp. 535–544.

[17] A. Gupta, A. Roth, and J. Ullman, *Iterative constructions and private data release*, in Theory of Cryptography, Lecture Notes in Comput. Sci. 7194, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 339–356.

[18] M. Hardt and G. Rothblum, *A multiplicative weights mechanism for privacy-preserving data analysis*, in Proceedings of the 51st IEEE Annual Symposium on Foundations of Computer Science (FOCS), 2010, pp. 61–70.

[19] M. Hardt, G. N. Rothblum, and R. A. Servedio, *Private data release via learning thresholds*, in Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, 2012, pp. 168–187.

[20] M. Hardt and K. Talwar, *On the geometry of differential privacy*, in Proceedings of the 42nd ACM Symposium on Theory of Computing, ACM, New York, 2010, pp. 705–714.

[21] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright, *A practical differentially private random decision tree classifier*, in Proceedings of the 2009 IEEE International Conference on Data Mining Workshops, 2009, pp. 114–121.

[22] S. P. 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 Proceedings of the 42nd ACM Symposium on Theory of Computing, ACM, New York, 2010, pp. 775–784.

[23] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith, *What can we learn privately?*, SIAM J. Comput., 40 (2011), pp. 793–826.

[24] M. Kearns, *Efficient noise-tolerant learning from statistical queries*, J. ACM, 45 (1998), pp. 983–1006.

[25] 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, ACM, New York, 2003, pp. 137–146.

[26] A. Krause and C. Guestrin, *Near-optimal observation selection using submodular functions*, in Proceedings of the 22nd National Conference on Artificial Intelligence, Vol. 2, AAAI Press, Menlo Park, CA, 2007, pp. 1650–1654.

[27] N. Littlestone and M. K. Warmuth, *The weighted majority algorithm*, Inform. and Comput., 108 (1994), pp. 212–261.

[28] S. Raskhodnikova and G. Yaroslavtsev, *Learning Pseudo-Boolean k-DNF and Submodular Functions*, preprint, arXiv:1208.2294v1 [cs.LG], 2012.

[29] A. Roth and T. Roughgarden, *Interactive privacy via the median mechanism*, in Proceedings of the 42nd ACM Symposium on Theory of Computing, ACM, New York, 2010, pp. 765–774.

[30] Z. Svitkina and L. Fleischer, *Submodular approximation: Sampling-based algorithms and lower bounds*, SIAM J. Comput., 40 (2011), pp. 1715–1737.

[31] J. Thaler, J. Ullman, and S. Vadhan, *Faster algorithms for privately releasing marginals*, in Automata, Languages, and Programming, Lecture Notes in Comput. Sci. 7391, Springer, Berlin, Heidelberg, 2012, pp. 810–821.

[32] J. Ullman and S. P. Vadhan, *PCPs and the hardness of generating private synthetic data*, in Theory of Cryptography, Lecture Notes in Comput. Sci. 6597, Y. Ishai, ed., Springer, Heidelberg, 2011, pp. 400–416.

[33] J. Vondrak, *A Note on Concentration of Submodular Functions*, preprint, arXiv:1005.2791v1 [cs.DM], 2010.