Learning Coverage Functions and Private Release of Marginals

Vitaly Feldman
vitaly@post.harvard.edu

Pravesh Kothari
kothari@cs.utexas.edu

Abstract

We study the problem of approximating and learning coverage functions. A function $c : 2^{[n]} \to \mathbb{R}^+$ is a coverage function, if there exists a universe U with non-negative weights $w(u)$ for each $u \in U$ and subsets $A_1, A_2, \ldots, A_n$ of U such that $c(S) = \sum_{u \in \cup_{i \in S} A_i} w(u)$. Alternatively, coverage functions can be described as non-negative linear combinations of monotone disjunctions. They are a natural subclass of submodular functions and arise in a number of applications.

We give an algorithm that for any $\gamma, \delta > 0$, given random and uniform examples of an unknown coverage function c, finds a function h that approximates c within factor $1 + \gamma$ on all but δ-fraction of the points in time $\mathsf{poly}(n, 1/\gamma, 1/\delta)$. This is the first fully-polynomial algorithm for learning an interesting class of functions in the demanding PMAC model of Balcan and Harvey [2012]. Our algorithms are based on several new structural properties of coverage functions. Using the results in [Feldman and Kothari, 2014], we also show that coverage functions are learnable agnostically with excess ℓ_1-error ϵ over all product and symmetric distributions in time $n^{\log(1/\epsilon)}$. In contrast, we show that, without assumptions on the distribution, learning coverage functions is at least as hard as learning polynomial-size disjoint DNF formulas, a class of functions for which the best known algorithm runs in time $2^{\tilde{O}(n^{1/3})}$ [Klivans and Servedio, 2004].

As an application of our learning results, we give simple differentially-private algorithms for releasing monotone conjunction counting queries with low *average* error. In particular, for any $k \leq n$, we obtain private release of k-way marginals with average error $\bar{\alpha}$ in time $n^{O(\log(1/\bar{\alpha}))}$.

1 Introduction

We consider learning and approximation of the class of *coverage* functions over the Boolean hypercube $\{-1, 1\}^n$. A function $c : 2^{[n]} \to \mathbb{R}^+$ is a coverage function if there exists a family of sets $A_1, A_2, \ldots, A_n$ on a universe U equipped with a weight function $w : U \to \mathbb{R}^+$ such that for any $S \subseteq [n]$, $c(S) = w(\cup_{i \in S} A_i)$, where $w(T) = \sum_{u \in T} w(u)$ for any $T \subseteq U$. We view these functions over $\{-1, 1\}^n$ by associating each subset $S \subseteq [n]$ with vector $x^S \in \{-1, 1\}^n$ such that $x^S_i = -1$ iff $i \in S$. We define the size (denoted by $\mathsf{size}(c)$) of a coverage function c as the size of a smallest-size universe U that can be used to define c. As is well-known, coverage functions also have an equivalent and natural representation as non-negative linear combinations of monotone disjunctions with the size being the number of disjunctions in the combination.

Coverage functions form a relatively simple but important subclass of the broad class of submodular functions. Submodular functions have been studied in a number of contexts and play an important role in combinatorial optimization [Lovász, 1983, Goemans and Williamson, 1995, Fleischer et al., 2001, Edmonds, 1970, Frank, 1997, Feige, 1998] with several applications to machine learning [Guestrin et al., 2005, Krause et al., 2006, Krause and Guestrin, 2011, Iyer and Bilmes, 2013] and in algorithmic game theory, where they are used to model valuation functions [B. Lehmann and Nisan, 2006, Dobzinski and Schapira, 2006, Vondrák, 2008]. Coverage functions themselves figure in several applications such as facility location [Cornuejols et al., 1977], private data release of conjunctions [Gupta et al., 2011] and algorithmic game theory where they are used to model the utilities of agents in welfare maximization and design of combinatorial auctions [Dughmi and Vondrák, 2011].

In this paper, we investigate the learnability of coverage functions from random examples. The study of learnability from random examples of the larger classes of functions such as submodular and fractionally-subadditive functions has been initiated by Balcan and Harvey [2012] who were motivated by applications in algorithmic game theory. They introduced the PMAC model of learning in which, given random and independent examples of an unknown function, the learner is required to output a hypothesis that is multiplicatively close (which is the standard notion of approximation in the optimization setting) to the unknown target on at least $1-\delta$ fraction of the points. This setting is also considered in [Balcan et al., 2012, Badanidiyuru et al., 2012]. Learning of submodular functions with less demanding (and more common in machine learning) additive guarantees was first considered by Gupta et al. [2011], who were motivated by problems in private data release. In this setting the goal of the learner is equivalent to producing a hypothesis that ϵ-approximates the target function in ℓ_1 or ℓ_2 distance. That is for functions f, g, $\mathbf{E}_{x\sim\mathcal{D}}[|f(x)-g(x)|]$ or $\sqrt{\mathbf{E}_{x\sim\mathcal{D}}[(f(x)-g(x))^2]}$ where $\mathcal{D}$ is the underlying distribution on the domain (with the uniform distribution being the most common). The same notion of error and restriction to the uniform distribution are also used in several subsequent works on learning of submodular functions [Cheraghchi et al., 2012, Raskhodnikova and Yaroslavtsev, 2012, Feldman et al., 2013, Feldman and Vondrák, 2013]. We consider both these models in the present work. For a more detailed survey of submodular function learning the reader is referred to [Balcan and Harvey, 2012].

1.1 Our Results

1.1.1 Distribution-independent learning

Our main results are for the uniform, product and symmetric distribution learning of coverage functions. However it is useful to first understand the complexity of learning these functions without any distributional assumptions (for a formal definition and details of the models of learning see Sec. 2). We prove (see Sec. 6) that distribution-independent learning of coverage functions is at least as hard as PAC learning the class of polynomial-size disjoint DNF formulas over arbitrary distributions (that is DNF formulas, where each point satisfies at most 1 term). Polynomial-size disjoint DNF formulas is an expressive class of Boolean functions that includes the class of polynomial-size decision trees, for example. Moreover, there is no known algorithm for learning polynomial-size disjoint DNFs that runs faster than the algorithm for learning general DNF formulas, the best known algorithm for which runs in time $2^{\tilde{O}(n^{1/3})}$ [Klivans and Servedio, 2004]. Let $\mathcal{CV}$ denote the class of coverage functions over $\{-1,1\}^n$ with range in $[0,1]$.

Theorem 1.1. *Let $\mathcal{A}$ be an algorithm that learns all coverage functions in $\mathcal{CV}$ of size at most s with ℓ_1-error ϵ in time $T(n,s,\frac{1}{\epsilon})$. Then, there exists an algorithm $\mathcal{A}'$ that PAC learns the class of s-term disjoint-DNF in time $T(2n,s,\frac{2s}{\epsilon})$.*

This reduction gives a computational impediment to fully-polynomial PAC (and consequently PMAC) learning of coverage functions of polynomial size or any class that includes coverage functions. Previously, hardness results for learning various classes of submodular and fractionally-subadditive functions were information-theoretic [Balcan and Harvey, 2012, Badanidiyuru et al., 2012, Balcan et al., 2012] or required encodings of cryptographic primitives in the function [Balcan and Harvey, 2012]. On the positive side, in Sec. 6.2 we show that learning (both distribution-specific and distribution-independent) of coverage functions of size s is at most as hard as learning the class of linear thresholds of s monotone Boolean disjunctions (which for example include monotone CNF with s clauses). A special case of this simple reduction appears in [Hardt et al., 2012].

1.1.2 PAC and PMAC learning over the uniform distribution

Learning of submodular functions becomes substantially easier when the distribution is restricted to be uniform (denoted by $\mathcal{U}$). For example, all submodular functions are learnable with ℓ_1-error of ϵ in time $2^{O(1/\epsilon^4)} \cdot \mathsf{poly}(n)$ [Feldman et al., 2013] whereas there is a constant α and a distribution $\mathcal{D}$ such that no polynomial-time algorithm can achieve ℓ_1-error of α when learning submodular functions relative to $\mathcal{D}$ [Balcan and Harvey, 2012]. At the same time achieving fully-polynomial time is often hard even under this strong assumption on the distribution. For example, polynomial-size disjoint DNF or monotone DNF/CNF are not known to be learnable efficiently in this setting and the best algorithms run in $n^{O(\log(n/\epsilon))}$ time. But, as we show below, when restricted to the uniform distribution, coverage functions are easier than disjoint DNF and are PAC learnable efficiently. Further, they are learnable in fully-polynomial time even with the stronger multiplicative approximation guarantees of the PMAC learning model [Balcan and Harvey, 2012]. We first state the PAC learning result which is easier to prove and serves as a step toward the PMAC algorithm.

Theorem 1.2. *There exists an algorithm which, given $\epsilon > 0$ and access to random uniform examples of any coverage function $c \in \mathcal{CV}$, with probability at least $2/3$, outputs a hypothesis h such that $\mathbf{E}_{\mathcal{U}}[|h(x) - c(x)|] \leq \epsilon$. The algorithm runs in $\tilde{O}(n/\epsilon^4 + 1/\epsilon^8)$ time and uses $\log n \cdot \tilde{O}(1/\epsilon^4)$ examples.*

We note that for general submodular functions exponential dependence on $1/\epsilon$ is necessary information-theoretically [Feldman et al., 2013]. To obtain an algorithm with multiplicative guarantees we show that for every monotone submodular (and not just coverage) function multiplicative approximation can be easily reduced to additive approximation. The reduction decomposes $\{-1, 1\}^n$ into $O(\log(1/\delta))$ subcubes where the target function is relatively large with high probability, specifically the value of f on each subcube is $\Omega(1/\log(1/\delta))$ times the maximum value of f on the subcube. The reduction is based on concentration results for submodular functions [Boucheron et al., 2000, Vondrák, 2010, Balcan and Harvey, 2012] and the fact that for any non-negative monotone submodular function f, $\mathbf{E}_{\mathcal{U}}[f] \geq \|f\|_\infty/2$ [Feige, 2006]. This reduction together with Thm. 1.2 yields our PMAC learning algorithm for coverage functions.

Theorem 1.3. *There exists an algorithm which, given $\gamma, \delta > 0$ and access to random uniform examples of any coverage function c, with probability at least $2/3$, outputs a hypothesis h such that $\mathbf{Pr}_{\mathcal{U}}[h(x) \leq c(x) \leq (1+\gamma)h(x)] \geq 1 - \delta$. The algorithm runs in $\tilde{O}(\frac{n}{\gamma^4\delta^4} + \frac{1}{\gamma^8\delta^8})$ time and uses $\log n \cdot \tilde{O}(\frac{1}{\gamma^4\delta^4})$ examples.*

This is the first fully-polynomial (that is polynomial in n, $1/\epsilon$ and $1/\delta$) algorithm for PMAC learning a natural subclass of submodular functions even when the distribution is restricted to be uniform. As a point of comparison, the sketching result of Badanidiyuru et al. [2012] shows that for every coverage function c and $\gamma > 0$, there exists a coverage function of size $\mathsf{poly}(n, 1/\gamma)$ size that approximates c within factor $1 + \gamma$ everywhere. Unfortunately, it is unknown how to compute this strong approximation even in subexponential time and even with value queries[1] and the distribution is restricted to be uniform. . Our result shows that if one relaxes the approximation to be over $1 - \delta$ fraction of points then in time polynomial in $n, 1/\gamma$ and $1/\delta$ one can find a $(1+\gamma)$-approximating function using random examples alone.

The key property that we identify and exploit in designing the PAC algorithm is that the Fourier coefficients of coverage functions have a form of (anti-)monotonicity property.

[1] A value query on a point in a domain returns the value of the target function at the point. For Boolean functions it is usually referred to as a membership query.

Lemma 1.4. *For a coverage function $c : \{-1,1\}^n \to [0,1]$ and non-empty $T \subseteq V$, $|\hat{c}(T)| \geq |\hat{c}(V)|$.*

This lemma allows us to find all significant Fourier coefficients of a coverage function efficiently using a search procedure analogous to that in the Kushilevitz-Mansour algorithm [Kushilevitz and Mansour, 1993] (but without the need for value queries). An additional useful property we prove is that any coverage function can be approximated by a function of few variables (referred to as *junta*).

Theorem 1.5. *For any coverage function $c : \{-1,1\}^n \to [0,1]$ and $\epsilon > 0$, there exists a coverage function c', that depends only on $O(1/\epsilon^2)$ variables and satisfies $\mathbf{E}_{\mathcal{U}}[|c(x) - c'(x)|] \leq \epsilon$.*

By identifying the variables of an approximating junta we make the learning algorithm computationally more efficient and achieve logarithmic dependence of the number of random examples on n. This, in particular, implies *attribute efficiency* [Blum and Langley, 1997] of our algorithm. Our bound on junta size is tight since coverage functions include monotone linear functions which require a $\Omega(1/\epsilon^2)$-junta for ϵ-approximation (e.g. [Feldman and Vondrák, 2013]). This clearly distinguishes coverage functions from disjunctions themselves which can always be approximated using a function of just $O(\log(1/\epsilon))$ variables. We note that in a subsequent work Feldman and Vondrák [2013] showed approximation by $O(\log(1/\epsilon)/\epsilon^2)$-juntas for all submodular functions using a more involved approach. They also show that this approximation leads to a $2^{\tilde{O}(1/(\delta\gamma)^2)} \cdot \mathsf{poly}(n)$ PMAC learning algorithm for all submodular functions.

Exploiting the representation of coverage functions as non-negative linear combinations of monotone disjunctions, we show that we can actually get a PAC learning algorithm that outputs a hypothesis that is guaranteed to be a coverage function. That is, the algorithm is *proper*. The running time of this algorithm is polynomial in n and, in addition, depends polynomially on the size of the target coverage function.

Theorem 1.6. *There exists an algorithm, that for any $\epsilon > 0$, given random and uniform examples of any $c \in \mathcal{CV}$, with probability at least $2/3$, outputs a* coverage *function h such that $\mathbf{E}_{\mathcal{U}}[|h(x) - c(x)|] \leq \epsilon$. The algorithm runs in time $\tilde{O}(n) \cdot \mathsf{poly}(s/\epsilon)$ and uses $\log(n) \cdot \mathsf{poly}(s/\epsilon)$ random examples, where $s = \min\{\mathsf{size}(c), (1/\epsilon)^{\log(1/\epsilon)}\}$.*

1.1.3 Agnostic Learning on Product and Symmetric Distributions

We then consider learning of coverage functions over general product and symmetric distributions (that is those whose PDF is symmetric with respect to the n variables). These are natural generalizations of the uniform distribution studied in a number of prior works. In our case the motivation comes from the application to differentially-private release of (monotone) k-conjunction counting queries referred to as *k-way marginals* in this context. Releasing k-way marginals with average error corresponds to learning of coverage functions over the uniform distribution on points of Hamming weight k which is a symmetric distribution (we describe the applications in more detail in the next subsection).

As usual with Fourier transform-based techniques, on general product distributions the running time of our PAC learning algorithm becomes polynomial in $(1/p)^{O(\log(1/\epsilon))}$, where p is the smallest bias of a variable in the distribution. It also relies heavily on the independence of variables and therefore does not apply to general symmetric distributions. Therefore, we use a different approach to the problem which learns coverage functions by learning disjunctions in the agnostic learning model [Haussler, 1992, Kearns et al., 1994b]. This approach is based on a simple and known observation that if disjunctions can be approximated in ℓ_1 distance by linear combinations of some basis functions then so are coverage functions. As a result, the learning algorithm for coverage functions also has agnostic guarantees relative to ℓ_1-error.

Theorem 1.7. *There exists an algorithm which for any product or symmetric distribution $\mathcal{D}$ on $\{-1,1\}^n$, given $\epsilon > 0$ and access examples of a function $f : \{-1,1\}^n \to [0,1]$ on points sampled from $\mathcal{D}$, with probability at least $2/3$, outputs a hypothesis h such that $\mathbf{E}_{\mathcal{D}}[|h(x) - f(x)|] \leq \min_{c \in \mathcal{CV}}\{\mathbf{E}_{\mathcal{D}}[|c(x) - f(x)|]\} + \epsilon$. The algorithm runs in $n^{O(\log(1/\epsilon))}$ time.*

For product distributions, this algorithm relies on the fact that disjunctions can be ℓ_1-approximated within ϵ by degree $O(\log(1/\epsilon))$ polynomials [Blais et al., 2008]. A simpler proof for this approximation appears in [Feldman and Kothari, 2014] where it is also shown that the same result holds for all symmetric distributions.

For the special case of product distributions that have their one dimensional marginal expectations bounded away from 0 and 1 by some constants, we show that we can in fact make our agnostic learning algorithm *proper*, that is, the hypothesis returned by our algorithm is a coverage function. In particular, we give a proper agnostic learning algorithm for coverage functions over the uniform distribution running in time $n^{O(\log(1/\epsilon))}$. It is not hard to show that this algorithm is essentially the best possible assuming hardness of learning sparse parities with noise.

1.1.4 Applications to Differentially Private Data Release

We now briefly overview the problem of differentially private data release and state our results. Formal definitions and details of our applications to privacy appear in Sec. 5 and a more detailed background discussion can for example be found in [Thaler et al., 2012]. The objective of a private data release algorithm is to release answers to all *counting queries* from a given class $\mathcal{C}$ with low error while protecting the privacy of participants in the data set. Specifically, we are given a data set D which is a subset of a fixed domain X (in our case $X = \{-1,1\}^n$). Given a query class $\mathcal{C}$ of Boolean functions on $\{-1,1\}^n$, the objective is to output a data structure H that allows answering *counting queries* from $\mathcal{C}$ on D with low error. A counting query for $c \in \mathcal{C}$ gives the fraction of elements in D on which c equals to 1. The algorithm producing H should be *differentially private* [Dwork et al., 2006]. The efficiency of a private release algorithm for releasing a class of queries $\mathcal{C}$ with error α on a data set $D \subseteq X$ is measured by its running time (in the size of the data set, the dimension and the error parameter) and the minimum data set size required for achieving certain error. Informally speaking, a release algorithm is differentially private if adding an element of X to (or removing an element of X from) D does not affect the probability that any specific H will be output by the algorithm significantly. A natural and often useful way of private data release for a data set D is to output another data set $\hat{D} \subseteq X$ (in a differentially private way) such that answers to counting queries based on $\hat{D}$ approximate answers based on D. Such release is referred to as data *sanitization* and the data set is referred to as a *synthetic data set*.

Releasing Boolean conjunction counting queries is likely the single best motivated and most well-studied problem in private data analysis [Barak et al., 2007, Ullman and Vadhan, 2011, Cheraghchi et al., 2012, Hardt et al., 2012, Thaler et al., 2012, Bun et al., 2013, Chandrasekaran et al., 2014, Dwork et al., 2013]. It is a part of the official statistics in the form of reported data in the US Census, Bureau of Labor statistics and the Internal Revenue Service.

Despite the relative simplicity of this class of functions, the best known algorithm for releasing all k-way marginals with a constant worst-case error runs in polynomial time for data sets of size at least $n^{\Omega(\sqrt{k})}$ [Thaler et al., 2012]. Starting with the work of Gupta et al. [2011], researchers have also considered the private release problem with low *average* error with respect to some distribution, most commonly uniform, on the class of queries [Cheraghchi et al., 2012, Hardt et al., 2012, Dwork et al., 2013]. However, in most applications only relatively short marginals are of interest and therefore the average error relative to the uniform distribution can be completely uninformative in this case. As can be easily seen (e.g. [Gupta et al., 2011]), the function mapping a monotone conjunction c to a counting query for c on a data set D can be written in terms of a convex combination of monotone disjunctions corresponding to points in D which is a coverage function. In this translation the distribution on conjunctions becomes a distribution over points on which the coverage function is defined and the ℓ_1 error in approximating the coverage function becomes the average error of the data release. Therefore using standard techniques, we adapt our learning algorithms to this problem. Thm. 1.7 gives the following algorithm for release of k-way marginals.

Theorem 1.8. *Let $\mathcal{C}_k$ be the class of all monotone conjunctions of length $k \in [n]$. For every $\epsilon > 0$, there is an ϵ-differentially private algorithm which for any data set $D \subseteq \{-1,1\}^n$ of size $n^{\Omega(\log(1/\bar{\alpha}))} \cdot \log(1/\delta)/\epsilon$, with probability at least $1-\delta$ outputs a data structure H that answers counting queries for $\mathcal{C}_k$ with respect to the uniform distribution on $\mathcal{C}_k$ with an average error of at most $\bar{\alpha}$. The algorithm runs in time $n^{O(\log(1/\bar{\alpha}))} \cdot \log(1/\delta)/\epsilon$ and the size of H is $n^{O(\log(1/\bar{\alpha}))}$.*

Note that there is no dependence on k in the bounds and it applies to any symmetric distribution. Without assumptions on the distribution, Dwork et al. [2013] give an algorithm that releases k-way marginals with average error $\bar{\alpha}$ given a data set of size at least $\tilde{\Omega}(n^{\lceil k/2 \rceil/2} \cdot 1/\bar{\alpha}^2)$ and runs in polynomial time in this size. (They also give a method to obtain the stronger worst-case error guarantees by using *private boosting*.)

We then adapt our PAC learning algorithms for coverage functions to give two algorithms for privately releasing monotone conjunction counting queries over the uniform distribution. Our first algorithm uses Thm. 1.2 to obtain a differentially private algorithm for releasing monotone conjunction counting queries in time polynomial in n (the data set dimension) and $1/\bar{\alpha}$.

Theorem 1.9. *Let $\mathcal{C}$ be the class of all monotone conjunctions. For every $\epsilon, \delta > 0$, there exists an ϵ-differentially private algorithm which for any data set $D \subseteq \{-1,1\}^n$ of size $\tilde{\Omega}(n\log(1/\delta)/(\epsilon\bar{\alpha}^6))$, with probability at least $1-\delta$, outputs a data structure H that answers counting queries for $\mathcal{C}$ with respect to the uniform distribution with an average error of at most $\bar{\alpha}$. The algorithm runs in time $\tilde{O}(n^2\log(1/\delta)/(\epsilon\bar{\alpha}^{10}))$ and the size of H is $\log n \cdot \tilde{O}(1/\bar{\alpha}^4)$.*

The previous best algorithm for this problem runs is time $n^{O(\log(1/\bar{\alpha}))}$ [Cheraghchi et al., 2012]. In addition, using a general framework from [Hardt et al., 2012], one can reduce private release of monotone conjunction counting queries to PAC learning with value queries of linear thresholds of a polynomial number of conjunctions over a certain class of "smooth" distributions. Hardt et al. [2012] show how to use their framework together with Jackson's algorithm for learning majorities of parities [Jackson, 1997] to privately release parity counting queries. Using a similar argument one can also obtain a polynomial-time algorithm for privately releasing monotone conjunction counting queries. Our algorithm is substantially simpler and more efficient than the one obtained via the reduction in [Hardt et al., 2012].

We can also use our proper learning algorithm to obtain a differentially private sanitization for releasing marginals in time polynomial in n and quasi-polynomial in $1/\bar{\alpha}$.

Theorem 1.10. *Let $\mathcal{C}$ be the class of all monotone conjunctions. For every $\epsilon, \delta > 0$, there exists an ϵ-differentially private algorithm which for any data set $D \subseteq \{-1,1\}^n$ of size $n \cdot \bar{\alpha}^{-\Omega(\log(1/\bar{\alpha}))} \cdot \log(1/\delta)/\epsilon$, with probability at least $1-\delta$, releases a synthetic data set $\hat{D}$ that can answer counting queries for $\mathcal{C}$ with respect to the uniform distribution with average error of at most $\bar{\alpha}$. The algorithm runs in time $n^2 \cdot \bar{\alpha}^{-O(\log(1/\bar{\alpha}))} \cdot \log(1/\delta)/\epsilon$.*

Note that our algorithm for privately releasing monotone conjunction queries with low-average error via a synthetic data set is polynomial time for any error $\bar{\alpha}$ that is $2^{-O(\sqrt{\log(n)})}$.

1.2 Related Work

Badanidiyuru et al. [2012] study *sketching* of coverage functions and prove that for any coverage function there exists a small (polynomial in the dimension and the inverse of the error parameter) approximate representation that multiplicatively approximates the function on all points. Their result implies an algorithm for learning coverage functions in the PMAC model [Balcan and Harvey, 2012] that uses a polynomial number of examples but requires exponential time in the dimension n. Chakrabarty and Huang [2012] study the problem of *testing* coverage functions (under what they call the *W-distance*) and show that the class of coverage functions of polynomial size can be reconstructed, that is, one can obtain in polynomial time, a representation of an unknown coverage function c such that $\mathsf{size}(c)$ is bounded by some polynomial in n (in general for a coverage function c, $\mathsf{size}(c)$ can be as high as 2^n), that computes c correctly at all points, using polynomially many value queries. Their reconstruction algorithm can be seen as an *exact* learning algorithm with value queries for coverage functions of small size.

In a recent (and independent) work, Yang et al. [2013] develop a subroutine for learning sums of monotone conjunctions that also relies on the monotonicity the Fourier coefficients (as in Lemma 1.4). Their application is in a very different context of learning DNF expressions from *numerical pairwise* queries, which given two assignments from $\{-1,1\}^n$ to the variables, expects in reply, the number of terms of the target DNF satisfied by both assignments.

A general result of Gupta et al. [2011] shows that releasing all counting queries from a concept class $\mathcal{C}$ using counting queries (when accessing the data set) requires as many counting queries as agnostically learning $\mathcal{C}$ using statistical queries. Using lower bounds on statistical query complexity of agnostic learning of conjunctions [Feldman, 2012] they derived a lower bound on counting query complexity for releasing all conjunction counting queries of certain length. This rules out a fully polynomial (in k, the data set size and the dimension n) algorithm to privately release short conjunction counting queries with low worst-case error.

Since our algorithms access the data set using counting queries, the lower bounds from [Gupta et al., 2011] apply to our setting. However the lower bound in [Gupta et al., 2011] is only significant when the length of conjunctions is at most logarithmic in n. Building on the work of Dwork et al. [2009], Ullman and Vadhan [2011] showed that there exists a constant γ such that there is no polynomial time algorithm for releasing a synthetic data set that answers all conjunction counting queries with worst-case error of at most γ under some mild cryptographic assumptions.

2 Preliminaries

We use $\{-1,1\}^n$ to denote the n-dimensional Boolean hypercube with "false" mapped to 1 and "true" mapped to -1. Let $[n]$ denote the set $\{1,2,\ldots,n\}$. For $S \subseteq [n]$, we denote by $\mathsf{OR}_S : \{-1,1\}^n \to \{0,1\}$, the monotone Boolean disjunction on variables with indices in S, that is, for any $x \in \{-1,1\}^n$, $\mathsf{OR}_S(x) = 0 \Leftrightarrow \forall i \in S\ \ x_i = 1$. A monotone Boolean disjunction is a simple example of a coverage function. To see this, consider a universe of size 1, containing a single element say u, the associated weight, $w(u) = 1$, and the sets $A_1, A_2, \ldots, A_n$ such that A_i contains u if and only if $i \in S$. In the following lemma we describe a natural and folklore characterization of coverage functions as non-negative linear combination of non-empty monotone disjunctions (e.g. [Gupta et al., 2011]). For completeness we include the proof in App. A.

Lemma 2.1. *A function $c : \{-1,1\}^n \to \mathbb{R}^+$ is a coverage function on some universe U, if and only if there exist non-negative coefficients α_S for every $S \subseteq [n], S \neq \emptyset$ such that $c(x) = \sum_{S\subseteq[n],S\neq\emptyset} \alpha_S \cdot \mathsf{OR}_S(x)$, and at most $|U|$ of the coefficients α_S are non-zero.*

For simplicity and without loss of generality we scale coverage functions to the range $[0,1]$. Note that in this case, for $c = \sum_{S \subseteq [n], S \neq \emptyset} \alpha_S \cdot \mathsf{OR}_S$ we have $\sum_{S \subseteq [n], S \neq \emptyset} \alpha_S = c((-1,\ldots,-1)) \leq 1$. In the discussion below we always represent coverage functions as linear combination of monotone disjunctions with the sum of coefficients upper bounded by 1. For convenience, we also allow the empty disjunction (or constant 1) in the combination. Note that $\mathsf{OR}_{[n]}$ differs from the constant 1 only on one point $(1,1,\ldots,1)$ and therefore this more general definition is essentially equivalent for the purposes of our discussion. Note that for every S, the coefficient α_S is determined uniquely by the function since OR_S is a monomial when viewed over $\{0,1\}^n$ with 0 corresponding to "true".

2.1 Learning Models

Our learning algorithms are in several models based on the PAC model [Valiant, 1984]. In the PAC learning model the learner has access to random examples of an unknown function from a known class of functions and the goal is to output a hypothesis with low error. The PAC model was defined for Boolean functions with the probability of disagreement being used to measure the error. For our real-valued setting we use ℓ_1 error which generalizes the disagreement error.

Definition 2.2 (PAC learning with ℓ_1-error)**.** *Let $\mathcal{F}$ be a class of real-valued functions on $\{-1,1\}^n$ and let $\mathcal{D}$ be a distribution on $\{-1,1\}^n$. An algorithm $\mathcal{A}$ PAC learns $\mathcal{F}$ on $\mathcal{D}$, if for every $\epsilon > 0$ and any target function $f \in \mathcal{F}$, given access to random independent samples from $\mathcal{D}$ labeled by f, with probability at least $2/3$, $\mathcal{A}$ returns a hypothesis h such that $\mathbf{E}_{x\sim\mathcal{D}}[|f(x) - h(x)|] \leq \epsilon$. $\mathcal{A}$ is said to be* proper *if $h \in \mathcal{F}$. $\mathcal{A}$ is said to be* efficient *if h can be evaluated in polynomial time on any input and the running time of $\mathcal{A}$ is polynomial in n and $1/\epsilon$.*

We also consider learning from random examples with multiplicative guarantees introduced by Balcan and Harvey [2012] and referred to as PMAC learning. For a class of non-negative functions $\mathcal{F}$, a PMAC learner with approximation factor $\alpha \geq 1$ and error $\delta > 0$ is an algorithm which, with probability at least $2/3$, outputs a hypothesis h that satisfies $\mathbf{Pr}_{x\sim\mathcal{D}}[h(x) \leq f(x) \leq \alpha h(x)] \geq 1 - \delta$. We say that h multiplicatively (α,δ)-approximates f over $\mathcal{D}$ in this case.

We are primarily interested in the regime when the approximation ratio α is close to 1 and hence use $1+\gamma$ instead. We say that the learner is *fully-polynomial* if it is polynomial in n,$1/\gamma$ and $1/\delta$.

2.2 Fourier Analysis on the Boolean Cube

When learning with respect to the uniform distribution we use several standard tools and ideas from Fourier analysis on the Boolean hypercube. For any functions $f,g : \{-1,1\}^n \to \mathbb{R}$, the inner product of f and g is defined as $\langle f,g\rangle = \mathbf{E}_{x\sim\mathcal{U}}[f(x)\cdot g(x)]$. The ℓ_1 and ℓ_2 norms of f are defined by $\|f\|_1 = \mathbf{E}_{x\sim\mathcal{U}}[|f(x)|]$ and $\|f\|_2^2 = \mathbf{E}_{x\sim\mathcal{U}}[f(x)^2]$, respectively. Unless noted otherwise, in this context all expectations are with respect to x chosen from the uniform distribution.

For $S \subseteq [n]$, the parity function $\chi_S : \{-1,1\}^n \to \{-1,1\}$ is defined as $\chi_S(x) = \prod_{i\in S} x_i$. Parities form an orthonormal basis for functions on $\{-1,1\}^n$ (for the inner product defined above). Thus, every function $f : \{-1,1\}^n \to \mathbb{R}$ can be written as a real linear combination of parities. The coefficients of the linear combination are referred to as the Fourier coefficients of f. For $f : \{-1,1\}^n \to \mathbb{R}$ and $S \subseteq [n]$, the Fourier coefficient $\hat{f}(S)$ is given by $\hat{f}(S) = \langle f, \chi_S\rangle = \mathbf{E}[f(x)\chi_S(x)]$.The Fourier expansion of f is given by $f(x) = \sum_{S\subseteq[n]} \hat{f}(S)\chi_S(x)$. For any function f on $\{-1,1\}^n$ its spectral ℓ_1-norm is defined as $\|\hat{f}\|_1 = \sum_{S\subseteq[n]} |\hat{f}(S)|$.

It is easy to estimate any Fourier coefficient of a function $f : \{-1,1\}^n \to [0,1]$, given access to an oracle that outputs the value of f at a uniformly random point in the hypercube. Given any parameters $\epsilon, \delta > 0$, we choose a set $R \subseteq \{-1,1\}^n$ of size $\Theta(\log \frac{1}{\delta}/\epsilon^2)$ drawn uniformly at random from $\{-1,1\}^n$ and estimate $\tilde{f}(S) = \frac{1}{|R|}\sum_{x\in R}[f(x)\cdot\chi_S(x)]$. Standard Chernoff bounds can then be used to show that with probability at least $1-\delta$, $|\hat{f}(S) - \tilde{f}(S)| \leq \epsilon$. For any $\epsilon > 0$, a Boolean function f is said to be ϵ-concentrated on a set $\mathbb{S} \subseteq 2^{[n]}$ of indices, if

$$\mathbf{E}\left[\left(f(x) - \sum_{S\in\mathbb{S}} \hat{f}(S)\chi_S(x)\right)^2\right] = \sum_{S\notin\mathbb{S}} \hat{f}(S)^2 \leq \epsilon.$$

The following simple observation (implicit in [Kushilevitz and Mansour, 1993]) can be used to obtain spectral concentration from bounded spectral ℓ_1-norm for any function f. In addition, it shows that approximating each large Fourier coefficient to a sufficiently small additive error yields a sparse linear combination of parities that approximates f. For completeness we include a proof in App. A.

Lemma 2.3. *Let $f : \{-1,1\}^n \to \mathbb{R}$ be any function with $\|f\|_2 \leq 1$. For any $\epsilon \in (0,1]$, let $L = \|\hat{f}\|_1$ and $\mathbb{T} = \{T \mid |\hat{f}(T)| \geq \frac{\epsilon}{2L}\}$. Then f is $\epsilon/2$-concentrated on $\mathbb{T}$ and $|\mathbb{T}| \leq \frac{2L^2}{\epsilon}$. Further, let $\mathbb{S} \supseteq \mathbb{T}$ and for each $S \in \mathbb{S}$, let $\tilde{f}(S)$ be an estimate of $\hat{f}(S)$ such that*

1. *$\forall S \in \mathbb{S}$, $|\tilde{f}(S)| \geq \frac{\epsilon}{3L}$ and*
2. *$\forall S \in \mathbb{S}$, $|\tilde{f}(S) - \hat{f}(S)| \leq \frac{\epsilon}{6L}$.*

Then, $\mathbf{E}[(f(x) - \sum_{S\in\mathbb{S}} \tilde{f}(S)\cdot\chi_S(x))^2] \leq \epsilon$ and, in particular, $\|f - \sum_{S\in\mathbb{S}} \tilde{f}(S)\cdot\chi_S\|_1 \leq \sqrt{\epsilon}$.

3 Learning Coverage Functions on the Uniform Distribution

Here we present our PAC and PMAC learning algorithms for $\mathcal{CV}$ over the uniform distribution.

3.1 Structural Results

We start by proving several structural lemmas about the Fourier spectrum of coverage functions. First, we observe that the spectral ℓ_1-norm of coverage functions is at most 2.

Lemma 3.1. *For a coverage function $c : \{-1,1\}^n \to [0,1]$, $\|\hat{c}\|_1 \leq 2$.*

Proof. From Lem. 2.1 we have that there exist non-negative coefficients α_S for every $S \subseteq [n]$ such that $c(x) = \sum_{S\subseteq[n]} \alpha_S \cdot \mathsf{OR}_S(x)$. By triangle inequality, we have: $\|\hat{c}\|_1 \leq \sum_{S\subseteq[n]} \alpha_S \cdot \|\widehat{\mathsf{OR}_S}\|_1 \leq \max_{S\subseteq[n]} \|\widehat{\mathsf{OR}_S}\|_1 \cdot \sum_{S\subseteq[n]} \alpha_S \leq \max_{S\subseteq[n]} \|\widehat{\mathsf{OR}_S}\|_1$. To complete the proof, we verify that $\forall S \subseteq [n]$, $\|\widehat{\mathsf{OR}_S}\|_1 \leq 2$. For this note that $\mathsf{OR}_S(x) = 1 - \frac{1}{2^{|S|}} \cdot \Pi_{i\in S}(1+x_i) = 1 - \frac{1}{2^{|S|}} \sum_{T\subseteq S} \chi_T(x)$ and thus $\|\widehat{\mathsf{OR}_S}\|_1 \leq 1 + \frac{1}{2^{|S|}} 2^{|S|} = 2$. □

The small spectral ℓ_1-norm guarantees that any coverage function has its Fourier spectrum ϵ^2-concentrated on some set $\mathbb{T}$ of indices of size $O(\frac{1}{\epsilon^2})$ (Lem. 2.3). This means that given an efficient algorithm to find a set $\mathbb{S}$ of indices such that $\mathbb{S}$ is of size $O(\frac{1}{\epsilon^2})$ and $\mathbb{S} \supseteq \mathbb{T}$ we obtain a way to PAC learn coverage functions to ℓ_1-error of ϵ. In general, given only random examples labeled by a function f that is concentrated on a small set $\mathbb{T}$ of indices, it is not known how to efficiently find a small set $\mathbb{S} \supseteq \mathbb{T}$, without additional information about $\mathbb{T}$ (such as all indices in $\mathbb{T}$ being of small cardinality). However, for coverage functions, we can utilize a simple monotonicity property of their Fourier coefficients to efficiently retrieve such a set $\mathbb{S}$ and obtain a PAC learning algorithm with running time that depends only polynomially on $1/\epsilon$.

Lemma 3.2 (Lem. 1.4 restated)**.** *Let $c : \{-1,1\}^n \to [0,1]$ be a coverage function. For any non empty $T \subseteq V \subseteq [n]$, $|\hat{c}(V)| \leq |\hat{c}(T)| \leq \frac{1}{2^{|T|}}$.*

Proof. From Lem. 2.1 we have that there exist constants $\alpha_S \geq 0$ for every $S \subseteq [n]$ such that $\sum_{S \subseteq [n]} \alpha_S \leq 1$ and $c(x) = \sum_{S \subseteq [n]} \alpha_S \mathsf{OR}_S(x)$ for every $x \in \{-1,1\}^n$. The Fourier transform of c can now be obtained simply by observing, as before in Lem. 3.1 that $\mathsf{OR}_S(x) = 1 - \frac{1}{2^{|S|}} \sum_{T \subseteq S} \chi_T(x)$. Thus for every $T \neq \emptyset$, $\hat{c}(T) = -\sum_{S \supseteq T} \alpha_S \cdot (\frac{1}{2^{|S|}})$. Notice that since all the coefficients α_S are non-negative, $\hat{c}(T)$ and $\hat{c}(V)$ are non-positive and $|\hat{c}(T)| = \sum_{S \supseteq T} \alpha_S \cdot (\frac{1}{2^{|S|}}) \geq \sum_{S \supseteq V} \alpha_S \cdot (\frac{1}{2^{|S|}}) = |\hat{c}(V)|$. For an upper bound on the magnitude $|\hat{c}(T)|$, we have: $|\hat{c}(T)| = \sum_{S \supseteq T} \alpha_S \cdot \frac{1}{2^{|S|}} \leq \sum_{S \supseteq T} \alpha_S \cdot \frac{1}{2^{|T|}} \leq (\sum_{S \subseteq [n]} \alpha_S) \cdot \frac{1}{2^{|T|}} \leq \frac{1}{2^{|T|}}$. □

We will now use Lemmas 3.1 and 3.2 to show that for any coverage function c, there exists another coverage function c' that depends on just $O(1/\epsilon^2)$ variables and ℓ_1-approximates it within ϵ. Using Lem. 3.1, we also obtain spectral concentration for c. We start with some notation: for any $x \in \{-1,1\}^n$ and a subset $J \subseteq [n]$ of variables, let $x_J \in \{-1,1\}^J$ denote the projection of x on J. Given $y \in \{-1,1\}^J$ and $z \in \{-1,1\}^{\bar{J}}$, let $x = y \circ z$ denote the string in $\{-1,1\}^n$ such that $x_J = y$ and $x_{\bar{J}} = z$ (where $\bar{J}$ denotes the set $[n] \setminus J$). We will need the following simple lemma that expresses the Fourier coefficients of the function f_I which is obtained by averaging a function f over all variables outside of I (a proof can be found for example in [Kushilevitz and Mansour, 1993]).

Lemma 3.3. *For $f : \{-1,1\}^n \to [0,1]$ and $I \subseteq [n]$, let $f_I(x) = \mathbf{E}_{y \sim \{-1,1\}^{\bar{I}}}[f(x_I \circ y)]$. Then, $\hat{f}_I(S) = \hat{f}(S)$ for every $S \subseteq I$ and $\hat{f}_I(T) = 0$ for every $T \not\subseteq I$.*

We now show that coverage functions can be approximated by functions of few variables.

Theorem 3.4 (Thm. 1.5 restated)**.** *Let $c : \{-1,1\}^n \to [0,1]$ be a coverage function and $\epsilon > 0$. Let $I = \{i \in [n] \mid |\hat{c}(\{i\})| \geq \frac{\epsilon^2}{2}\}$. Let c_I be defined as $c_I(x) = \mathbf{E}_{y \sim \{-1,1\}^{\bar{I}}}[c(x_I \circ y)]$. Then c_I is a coverage function that depends only on variables in I, $|I| \leq 4/\epsilon^2$, $\mathsf{size}(c_I) \leq \mathsf{size}(c)$ and $\|c - c_I\|_1 \leq \epsilon$. Further, let $\mathbb{T} = \{T \subseteq [n] \mid |\hat{c}(T)| \geq \frac{\epsilon^2}{2}\}$. Then $\mathbb{T} \subseteq 2^I$ and c is ϵ^2-concentrated on $\mathbb{T}$.*

Proof. Since c is a coverage function, it can be written as a non-negative weighted sum of monotone disjunctions. Thus, for every $v \in \{-1,1\}^{\bar{I}}$ the function, $c_v : \{-1,1\}^n \to [0,1]$ defined as $c_v(z \circ y) = c(z \circ v)$ for every $y \in \{-1,1\}^{\bar{I}}$ is also a non-negative linear combination of monotone disjunctions, that is a coverage function. By definition, for every $z \in \{-1,1\}^I$ and $y \in \{-1,1\}^{\bar{I}}$, $c_I(z \circ y) = \frac{1}{2^{n-|I|}} \sum_{v \in \{-1,1\}^{\bar{I}}} c(z \circ v) = \frac{1}{2^{n-|I|}} \sum_{v \in \{-1,1\}^{\bar{I}}} c_v(z \circ y)$. In other words, c_I is a convex combination of c_v's and therefore is a coverage function itself. Note that for every $S \subseteq I$ if the coefficient of OR_S in c_I is non-zero then there must exist $S' \subseteq \bar{I}$ for which the coefficient of $\mathsf{OR}_{S \cup S'}$ in c is non-zero. This implies that $\mathsf{size}(c_I) \leq \mathsf{size}(c)$. We will now establish that c_I approximates c. Using Lem. 3.3, $\hat{c}(S) = \hat{c}_I(S)$ for every $S \subseteq I$. Thus, $\|c - c_I\|_2^2 = \sum_{T \not\subseteq I} \hat{c}(T)^2$. We first observe that $\mathbb{T} \subseteq 2^I$. To see this, consider any $T \not\subseteq I$. Then, $\exists i \notin I$ such that $i \in T$ and therefore, by Lem. 3.2, $|\hat{c}(T)| \leq |\hat{c}(\{i\})| < \epsilon^2/2$. Thus $|\hat{c}(T)| \leq \epsilon^2/2$. By Lem. 2.3, c is ϵ^2-concentrated on $\mathbb{T}$ and using Cauchy-Schwartz inequality, $\|c - c_I\|_1^2 \leq \|c - c_I\|_2^2 = \sum_{T \not\subseteq I} \hat{c}(T)^2 \leq \sum_{T \notin \mathbb{T}} \hat{c}(T)^2 \leq \epsilon^2$. □

3.2 PAC Learning

We now describe our PAC learning algorithm for coverage functions. This algorithm is used for our application to private query release and also as a subroutine for our PMAC learning algorithm. Given the structural results above the algorithm itself is quite simple. Using random examples of the target coverage function, we compute all the singleton Fourier coefficients and isolate the set $\tilde{I}$ of coordinates corresponding to large (estimated) singleton coefficients that includes $I = \{i \in [n] \mid |\hat{c}(\{i\})| \geq \frac{\epsilon^2}{4}\}$. Thm. 3.4 guarantees that the target coverage function is concentrated on the large Fourier coefficients, the indices of which are subsets of $\tilde{I}$. We then find a collection $\mathbb{S} \subseteq 2^{\tilde{I}}$ of indices that contains all $T \subseteq \tilde{I}$ such that $|\hat{c}(T)| \geq \epsilon^2/4$. This can be done efficiently since by Lem. 3.2, $|\hat{c}(T)| \geq \epsilon^2/4$ only if $|\hat{c}(V)| \geq \epsilon^2/4$ for all $V \subseteq T$, $V \neq \emptyset$. We can only estimate Fourier coefficients up to some additive error with high probability and therefore we keep all coefficients in the set $\mathbb{S}$ whose estimated magnitude is at least $\epsilon^2/6$. Once we have a set $\mathbb{S}$, on which the target function is ϵ^2-concentrated, we use Lem. 2.3 to get our hypothesis. We give the pseudocode of the algorithm below.

Algorithm 1 PAC Learning of Coverage Functions

```
Set $\theta = \frac{\epsilon^2}{6}$.
Draw a random sample of size $m_1 = O(\log(n)/\epsilon^4)$ and use it to estimate $\hat{c}(\{i\})$ for all $i$.
Set $\tilde{I} = \{i \in [n] \mid |\tilde{c}(\{i\})| \geq \theta\}$.
$\mathbb{S} \leftarrow \{\emptyset\}$.
Draw random sample $R$ of size $m_2 = O(\log(1/\epsilon)/\epsilon^4)$.
for $t = 1$ to $\log(2/\theta)$ do
    for each set $T \in \mathbb{S}$ of size $t-1$ and $i \in \tilde{I} \setminus T$ do
        Use $R$ to estimate the coefficient $\hat{c}(T \cup \{i\})$.
        If $|\tilde{c}(T \cup i)| \geq \theta$ then $\mathbb{S} \leftarrow \mathbb{S} \cup \{T \cup \{i\}\}$
    end for
end for
return $\sum_{S \in \mathbb{S}} \tilde{c}(S) \cdot \chi_S$.
```

Theorem 3.5 (Thm. 1.2 restated)**.** *There exists an algorithm that PAC learns $\mathcal{CV}$ in $\tilde{O}(n/\epsilon^4 + 1/\epsilon^8)$ time and using $\log n \cdot \tilde{O}(1/\epsilon^4)$ examples.*

Proof. Let c be the target coverage function and let $\mathbb{T} = \{T \subseteq [n] \mid |\hat{c}(T)| \geq \frac{\epsilon^2}{4}\}$. By Lem. 2.3, it is sufficient to find a set $\mathbb{S} \supseteq \mathbb{T}$ and estimates $\tilde{c}(S)$ for each $S \in \mathbb{S}$ such that:

1. $\forall S \in \mathbb{S}\ |\tilde{c}(S)| \geq \frac{\epsilon^2}{6}$ and

2. $\forall S \in \mathbb{S}$, $|\tilde{c}(S) - \hat{c}(S)| \leq \frac{\epsilon^2}{12}$.

Let $\theta = \epsilon^2/6$. In the first stage our algorithm finds a set $\tilde{I}$ of variables that contains $I = \{i \in [n] \mid |\hat{c}(\{i\})| \geq \frac{\epsilon^2}{4}\}$. We do this by estimating all the singleton Fourier coefficients, $\{\hat{c}(\{i\}) \mid i \in [n]\}$ within $\theta/2$ with (overall) probability at least $5/6$ (as before we denote the estimate of $\hat{c}(S)$ by $\tilde{c}(S)$). We set $\tilde{I} = \{i \in [n] \mid \tilde{c}(\{i\})| \geq \theta\}$. If all the estimates are within $\theta/2$ of the corresponding coefficients then for every $i \in I$, $\tilde{c}(\{i\}) \geq \epsilon^2/4 - \theta/2 = \epsilon^2/6 = \theta$. Therefore $i \in \tilde{I}$ and hence $I \subseteq \tilde{I}$.

In the second phase, the algorithm finds a set $\mathbb{S} \subseteq 2^{\tilde{I}}$ such that the set of all large Fourier coefficients $\mathbb{T}$ is included in $\mathbb{S}$. This is done iteratively starting with $\mathbb{S} = \{\emptyset\}$. In every iteration, for every set T that was added in the previous iteration and every $i \in \tilde{I} \setminus T$, it estimates $\hat{c}(T \cup \{i\})$ within $\theta/2$ (the success probability for estimates in this whole phase will be $5/6$). If $|\tilde{c}(T \cup \{i\})| \geq \theta$ then $T \cup \{i\}$ is added to $\mathbb{S}$. This iterative process runs until no sets are added in an iteration. At the end of the last iteration, the algorithm returns $\sum_{S \in \mathbb{S}} \tilde{c}(S)\chi_S$ as the hypothesis.

We first prove the correctness of the algorithm assuming that all the estimates are successful. Let $T \in \mathbb{T}$ be such that $|\hat{c}(T)| \geq \epsilon^2/4$. Then, by Thm. 3.4, $T \subseteq I \subseteq \tilde{I}$. In addition, by Lem. 3.2, for all $V \subseteq T$, $V \neq \emptyset$, $|\hat{c}(V)| \geq \epsilon^2/4$. This means that for all $V \subseteq T$, $V \neq \emptyset$ an estimate of $|\hat{c}(V)|$ within $\theta/2$ will be at least θ. By induction on t this implies that in iteration t, all subsets of T of size t will be added to $\mathbb{S}$ and T will be added in iteration $|T|$. Hence the algorithm outputs a set $\mathbb{S}$ such that $\mathbb{T} \subseteq \mathbb{S}$. By definition, $\forall S \in \mathbb{S}$, $|\tilde{c}(S)| \geq \theta = \frac{\epsilon^2}{6}$ and $\forall S \in \mathbb{S}$, $|\tilde{c}(S) - \hat{c}(S)| \leq \theta/2 = \frac{\epsilon^2}{12}$. By Lem. 2.3, $\|c - \sum_{S \in \mathbb{S}} \tilde{c}(S)\chi_S\|_1 \leq \epsilon$.

We now analyze the running time and sample complexity of the algorithm. We make the following observations regarding the algorithm.

- By Chernoff bounds, $O(\log(n)/\theta^2) = O(\log(n)/\epsilon^4)$ examples suffice to estimate all singleton coefficients within $\theta/2$ with probability at least $5/6$. To estimate a singleton coefficients of c, the algorithm needs to look at only one coordinate and the label of a random example. Thus all the singleton coefficients can be estimated in time $O(n\log(n)/\epsilon^4)$.

- For every S such that $\hat{c}(S)$ was estimated within $\theta/2$ and $|\tilde{c}(S)| \geq \theta$, we have that $|\hat{c}(S)| \geq \theta/2 = \epsilon^2/12$. This implies that $|\tilde{I}| \leq 2/(\theta/2) = 24/\epsilon^2$. This also implies that $|\mathbb{S}| \leq 4/\theta = 24/\epsilon^2$.

- By Lem. 3.2, for any $T \subseteq [n]$, $|\hat{c}(T)| \leq \frac{1}{2^{|T|}}$. Thus, if $|\hat{c}(T)| \geq \theta/2$ then $|T| \leq \log(2/\theta)$. This means that the number of iterations in the second phase is bounded by $\log(2/\theta)$ and for all $S \in \mathbb{S}$, $|S| \leq \log(2/\theta)$.

- In the second phase, the algorithm only estimates coefficients for subsets in

  $$\mathbb{S}' = \{S \cup \{i\} \mid |\tilde{c}(S)| \geq \theta \text{ and } i \in \tilde{I}\}.$$

  Let $\mathbb{T}' = \{T \cup \{i\} \mid |\hat{c}(T)| \geq \theta/2 \text{ and } i \in \tilde{I}\}$. By Chernoff bounds, a random sample of size $O(\log|\mathbb{T}'|/\theta^2) = \tilde{O}(1/\epsilon^4)$ can be used to ensure that, with probability at least $5/6$, the estimates of all coefficients on subsets in $\mathbb{T}'$ are within $\theta/2$. When the estimates are successful we also know that $\mathbb{S}' \subseteq \mathbb{T}'$ and therefore all coefficients estimated by the algorithm in the second phase are also within $\theta/2$ of true values with probability $\geq 5/6$. Overall in the second phase the algorithm estimates $|\mathbb{S}'| \leq |\mathbb{S}| \cdot |\tilde{I}| = O(1/\epsilon^4)$ coefficients. To estimate any single of those coefficients, the algorithm needs to examine only $\log(2/\theta) = O(\log(1/\epsilon))$ coordinates and the label of an example. Thus, the estimation of each Fourier coefficient takes $\tilde{O}(1/\epsilon^4)$ time and $\tilde{O}(1/\epsilon^8)$ time is sufficient to estimate all the coefficients.

Thus, in total the algorithm runs in $\tilde{O}(n/\epsilon^4 + 1/\epsilon^8)$ time, uses $\log n \cdot \tilde{O}(1/\epsilon^4)$ random examples and succeeds with probability at least $2/3$. □

3.3 PMAC Learning

We now describe our PMAC learning algorithm that is based on a reduction from multiplicative to additive approximation. First we note that if we knew that the values of the target coverage function c are lower bounded by some $m > 0$ then we could obtain multiplicative $(1 + \gamma, \delta)$-approximation using a hypothesis h

with ℓ_1 error of $\gamma\delta m/2$. To see this note that, by Markov's inequality, $\mathbf{E}[|h(x)-c(x)|] \leq \gamma\delta m/2$ implies that $\mathbf{Pr}[|h(x)-c(x)| > \gamma m/2] \leq \delta$. Let $h'(x) = \max\{m, h(x) - \gamma m/2\}$. Then

$$\begin{aligned} 1-\delta &\leq \mathbf{Pr}[|h(x)-c(x)| \leq \gamma m/2] = \mathbf{Pr}[h(x) - \gamma m/2 \leq c(x) \leq h(x) + \gamma m/2] \\ &\leq \mathbf{Pr}[h'(x) \leq c(x) \leq h'(x) + \gamma m] \leq \mathbf{Pr}[h'(x) \leq c(x) \leq (1+\gamma)h'(x)] \end{aligned} \tag{1}$$

Now, we might not have such a lower bound on the value of c. To make this idea work for all coverage functions, we show that any monotone submodular function can be decomposed into regions where it is relatively large (compared to the maximum in that region) with high probability. The decomposition is based on the following lemma: given a monotone submodular function f with maximum value M, either $\mathbf{Pr}[f(x) \geq M/4] \geq 1 - \delta/2$ or there is an index $i \in [n]$ such that $f(x) \geq \frac{M}{16\ln(2/\delta)}$ for every x satisfying $x_i = -1$. In the first case we can obtain multiplicative approximation from additive approximation using a slight refinement of our observation above (since the lower bound on f only holds with probability $1-\delta/2$). In the second case we can reduce the problem to additive approximation on the half of the domain where $x_i = -1$. For the other half we use the same argument recursively. After $\lceil \log(2/\delta) \rceil$ levels of recursion at most $\delta/2$ fraction of the points will remain where we have no approximation. Those are included in the probability of error. We will need the following concentration inequality for 1-Lipschitz (with respect to the Hamming distance) submodular functions [Boucheron et al., 2000, Vondrák, 2010, Balcan and Harvey, 2012].

Theorem 3.6 (Vondrák, 2010)**.** *For a non-negative, monotone, 1-Lipschitz submodular function f and $0 \leq \beta < 1$, $\mathbf{Pr}_{\mathcal{U}}[f(x) \leq (1-\beta)\,\mathbf{E}_{\mathcal{U}}[f(x)]] \leq e^{-\beta^2\,\mathbf{E}[f]/2}$.*

Another property of non-negative monotone submodular functions that we need is that their expectation is at least half the maximum value [Feige, 2006]. For the special case of coverage functions this lemma follows simply from the fact that the expectation of any disjunction is at least $1/2$.

Lemma 3.7 (Feige, 2006)**.** *For f, a non-negative monotone submodular function, $\mathbf{E}_{\mathcal{U}}[f] \geq \|f\|_\infty/2$.*

We now prove our lemma that lower bounds the relative value of a monotone submodular function.

Lemma 3.8. *Let f be a non-negative monotone submodular function and $M = \|f\|_\infty$. Then for every $\delta > 0$, either $\mathbf{Pr}_{\mathcal{U}}[f(x) \leq M/4] \leq \delta$ or there exists an $i \in [n]$ such that $f(x) \geq \frac{M}{16\ln(1/\delta)}$ for every x such that $x_i = -1$.*

Proof. Let $e_i \in \{-1,1\}^n$ equal the bit string that has -1 in its i^{th} coordinate and 1 everywhere else. For any $x \in \{-1,1\}^n$ let $x \oplus y$ denote the string z such that $z_j = x_j \cdot y_j$ for every $j \in [n]$. Suppose that for every $i \in [n]$, exists x such that $x_i = -1$ and $f(x) \leq \frac{M}{16\ln(1/\delta)}$. By monotonicity of f that implies that for every $i \in [n]$, $f(e_i) \leq \frac{M}{16\ln(1/\delta)}$. Since f is a submodular function, for any x and i such that $x_i = 1$, we have: $f(x \oplus e_i) - f(x) \leq f(1^n \oplus e_i) - f(1^n) \leq f(e_i) \leq \frac{M}{16\ln(1/\delta)}$. This implies that f is $\frac{M}{16\ln(1/\delta)}$-Lipschitz. Then, $f' = f/\frac{M}{16\ln(1/\delta)}$ is a 1-Lipschitz, non-negative submodular function. Also, by Lem. 3.7, $\mathbf{E}[f] \geq M/2$ and $\mathbf{E}[f'] \geq 8\ln(1/\delta)$. Now, using Thm. 3.6, we obtain: $\mathbf{Pr}[f(x) \leq M/4] \leq \mathbf{Pr}[f'(x) \leq \frac{1}{2}\mathbf{E}[f']] \leq e^{-\frac{1}{8}\mathbf{E}[f']} \leq e^{-\ln(1/\delta)} = \delta$. □

Recall that for any set $J \subseteq [n]$ of variables and $x \in \{-1,1\}^n$, $x_J \in \{-1,1\}^J$ is defined as the substring of x that contains the bits in coordinates indexed by J. We are now ready to describe our reduction that gives a PMAC algorithm for coverage functions.

Theorem 3.9 (Thm. 1.3 restated)**.** *There exists an algorithm $\mathcal{A}$ which, given $\gamma, \delta > 0$ and access to random uniform examples of any coverage function c, with probability at least $2/3$, outputs a hypothesis h such that $\mathbf{Pr}_{\mathcal{U}}[h(x) \leq c(x) \leq (1+\gamma)h(x)] \geq 1-\delta$. Further, $\mathcal{A}$ runs in $\tilde{O}(\frac{n}{\gamma^4\delta^4} + \frac{1}{\gamma^8\delta^8})$ time and uses $\log n \cdot \tilde{O}(\frac{1}{\gamma^4\delta^4})$ examples.*

Proof. Algorithm $\mathcal{A}$ consists of a call to $\mathcal{A}'(0)$, where $\mathcal{A}'(k)$ is a recursive procedure described below.

Procedure $\mathcal{A}'(k)$ on examples labeled by $c : \{-1,1\}^n \to \mathbb{R}^+$:

1. If $k > \log(3/\delta)$, then, $\mathcal{A}'(k)$ returns the hypothesis $h \equiv 0$ and halts.

2. Otherwise, $\mathcal{A}'(k)$ computes a 3-approximation to the maximum M of the target function c (with confidence at least $1-\eta$ for η to be defined later). As we show later this can be done by drawing a sufficient number of random examples labeled by c and choosing $\tilde{M}$ to be the maximum label. Thus, $\frac{M}{3} \leq \tilde{M} \leq M$. If $\tilde{M} = 0$, **return** $h \equiv 0$. Otherwise, set $c' = \frac{c}{3\tilde{M}}$ (note that, with probability at least $1-\eta$, $c'(x) \in [0,1]$ for every x).

3. Estimate $p = \mathbf{Pr}[c(x) \leq \tilde{M}/4]$ within an additive error of $\frac{\delta}{9}$ by $\tilde{p}$ with confidence at least $1-\eta$. Then, $p - \frac{\delta}{9} \leq \tilde{p} \leq p + \frac{\delta}{9}$.

4. If $\tilde{p} < 2\delta/9$: run Algorithm from Thm. 3.5 on random examples labeled by c' with accuracy $\epsilon_1 = \frac{1}{12}\frac{\gamma}{2}\frac{\delta}{3}$ and confidence $1-\eta$ (note that Algorithm from Thm. 3.5 only gives $2/3$ confidence but the confidence can be boosted to $1-\eta$ using $O(\log(1/\eta))$ repetitions with standard hypothesis testing). Let h' be the hypothesis output by the algorithm. **Return** hypothesis $h = \max\{\tilde{M}/4, 3\tilde{M}(h' - \gamma/24)\}$.

5. If $\tilde{p} \geq 2\delta/9$,

   (a) Find $j \in [n]$ such that $c(x) \geq \tilde{M}/(16\ln(9/\delta))$ for every x such that $x_j = -1$ with confidence at least $1-\eta$. This can be done by drawing a sufficient number of random examples and checking the labels. If such j does not exist we output $h \equiv 0$. Otherwise, define $c_{j,-} : \{-1,1\}^{[n]\setminus j} \to \mathbb{R}^+$ to be the restriction of c to $\{-1,1\}^{[n]\setminus j}$ where $x_j = -1$ and $c'_{j,-} = c_{j,-}/(3\tilde{M})$. Run the algorithm from Thm. 3.5 on examples labeled by $c'_{j,-}$ with accuracy $\epsilon' = \frac{\gamma}{2}\cdot\frac{\delta}{3}\cdot\frac{1}{48\ln(9/\delta)}$ and confidence $1-\eta$. Let h'_- be the hypothesis returned by the algorithm. Set $h_- = \max\{\tilde{M}/4, 3\tilde{M}(h'_- - \frac{\gamma}{96\ln(9/\delta)})\}$.

   (b) Let $c_{j,+} : \{-1,1\}^{[n]\setminus j} \to \mathbb{R}^+$ to be the restriction of c to $\{-1,1\}^{[n]\setminus j}$ where $x_j = +1$. Run $\mathcal{A}'(k+1)$ on examples labeled by $c_{j,+}$ and let h_+ be the hypothesis returned by the algorithm.

   (c) **Return** hypothesis $h : \{-1,1\}^n \to \mathbb{R}^+$ defined by $h(x) = \begin{cases} h_-(x_{[n]\setminus j}) \text{ if } x_j = -1 \\ h_+(x_{[n]\setminus j}) \text{ if } x_j = 1 \end{cases}$

The algorithm can simulate random examples labeled by $c'_{j,-}$ (or $c_{j,+}$) by drawing random examples labeled by c, selecting (x, ℓ) such that $x_j = -1$ (or $x_j = 1$) and removing the j-th coordinate. Since $k \leq \log(3/\delta)$ bits will need to be fixed the expected number of random examples required to simulate one example from any function in the run of $\mathcal{A}(k)$ is at most $3/\delta$.

We now prove the correctness of the algorithm assuming that all random estimations and runs of the PAC learning algorithm are successful. To see that one can estimate the maximum M of a coverage function c within a multiplicative factor of 3, recall that by Lem. 3.7, $\mathbf{E}[c] \geq M/2$. Thus, for a randomly and uniformly chosen $x \in \{-1,1\}^n$, with probability at least $1/4$, $c(x) \geq M/3$. This means that $\log(2/\eta)$ random examples will suffice to get confidence $1-\eta$.

We now observe that if the condition in step 4 holds then h $(1+\gamma, 2\delta/3)$-multiplicatively approximates c. To see this, first note that in this case, $p = \mathbf{Pr}[c(x) \leq \tilde{M}/4] \leq \tilde{p} + \delta/9 \leq \delta/3$. Then, $\mathbf{Pr}[c'(x) \leq (\tilde{M}/4)/(3\tilde{M})] \leq \delta/3$. By Thm. 3.5, $\mathbf{E}[|c'(x) - h'(x)|] \leq \frac{1}{12}\frac{\gamma}{2} \cdot \frac{\delta}{3}$. Then, by Markov's inequality,

$$\mathbf{Pr}[h'(x) - \gamma/24 > c'(x) \text{ or } c'(x) > h'(x) + \gamma/24] \leq \delta/3.$$

Let $h''(x) = \max\{1/12, h'(x) - \gamma/24\}$. By the same argument as in eq. (1), we get that

$$\mathbf{Pr}[c'(x) \geq 1/12 \text{ and } (h''(x) > c'(x) \text{ or } c'(x) > (1+\gamma)h''(x))] \leq \delta/3.$$

Therefore,

$$\mathbf{Pr}[h''(x) \leq c'(x) \leq (1+\gamma)h''(x)] \geq 1 - 2\delta/3$$

or, equivalently,

$$\mathbf{Pr}[h(x) \leq c(x) \leq (1+\gamma)h(x)] \geq 1 - 2\delta/3.$$

If the condition in step 4 does not hold, then $p \geq 2\delta/9 - \delta/9 = \delta/9$. Thus, $\mathbf{Pr}[c(x) \leq M/4] \geq \mathbf{Pr}[c(x) \leq \tilde{M}/4] \geq \delta/9$, which by Lem. 3.8 yields that there exists $j \in [n]$ such that $c(x) \geq M/(16 \ln(9/\delta))$. Now, by drawing $O(\log(n/\eta)/\delta)$ examples and choosing j such that for all examples where $x_j = -1$, $c(x) \geq M/(16\ln(9/\delta))$ we can ensure that, with probability at least $1-\eta$,

$$\mathop{\mathbf{Pr}}_{y \in \{-1,1\}^{[n]\setminus j}}[c_{j,-}(y) \leq M/(16\ln(9/\delta))] \leq \delta/3.$$

Now, by the same analysis as in step 4, we obtain that h_- satisfies $\mathbf{Pr}[h_- \leq c_{j,-} \leq (1+\gamma)h_-] \geq 1 - 2\delta/3$.

Now, observe that the set of points in the domain $\{-1,1\}^n$ can be partitioned into two disjoint sets.

1. The set G such that for every $z \in G$, $\mathcal{A}$ has fixed the value of the hypothesis given by $h(z)$ based on some hypothesis returned by the PAC learning algorithm (Thm. 3.5) or $h(z) \equiv 0$ when $\tilde{M} = 0$.

2. The set $\bar{G}$ where the recursion has reached depth $k > \log(3/\delta)$ and step 1 sets $h(x) \equiv 0$ on every point in $\bar{G}$.

By the construction, the points in G can be divided into disjoint sub-cubes such that in each of them, the conditional probability that the hypothesis we output does not satisfy the multiplicative guarantee is at most $2\delta/3$. Therefore, the hypothesis h does not satisfy the multiplicative guarantee on at most $2\delta/3$ fraction of the points in G. It is easy to see that $\bar{G}$ has probability mass at most $\delta/3$. This is because $\mathcal{A} = \mathcal{A}(0)$ and thus, when $k > \log(3/\delta)$, the dimension of the subcube that $\mathcal{A}'(k)$ is invoked on, is at most $n - \log(3/\delta)$. Thus, the total probability mass of points where the multiplicative approximation does not hold is at most δ.

We now bound the running time and sample complexity of the algorithm. First note that for some $\eta = O(1/\log(1/\delta))$ all the random estimations and runs of the PAC learning algorithm will be successful with probability at least $2/3$ (by union bound).

From Thm. 3.5, any run of the PAC learning algorithm in some recursive call to $\mathcal{A}'$ requires at most $\log n \cdot \log(\frac{1}{\eta}) \cdot \tilde{O}(\frac{1}{\gamma^4 \cdot \delta^4})$ examples from their respective target functions. Each such example can be simulated using $\Theta(1/\delta)$ examples labeled by c. Thus, in total, in all recursive calls, $\log n \cdot \tilde{O}(\frac{1}{\gamma^4 \cdot \delta^5})$ examples will suffice.

Each run of the PAC learning algorithm requires $\tilde{O}(\frac{n}{\gamma^4\delta^4} + \frac{1}{\gamma^8\delta^8})$ time. The rest of the computations in any one recursive call to $\mathcal{A}'$ can be performed in time linear in the number of examples. Thus, total time required for an execution of $\mathcal{A}$ is bounded by $\tilde{O}(\frac{n}{\gamma^4\delta^4} + \frac{1}{\gamma^8\delta^8})$.

□

3.4 Proper PAC Learning Algorithm

We now present a PAC learning algorithm for coverage functions that guarantees that the returned hypothesis is also a coverage function. That is, the algorithm is *proper*. The running time of the algorithm will depend polynomially on the size of the target coverage function. We will need a variant of linear regression with ℓ_1 error for this algorithm which we now define formally.

3.4.1 ℓ_1 Linear Regression

It is easy to see that given a set of t examples $\{(x^i, y^i)\}_{i \le t}$ and a set of m functions $\phi_1, \phi_2, \ldots, \phi_m$ finding coefficients $\alpha_1, \ldots, \alpha_m$ which minimize

$$\sum_{i \le t} \left| \sum_{j \le m} \alpha_j \phi_j(x^i) - y^i \right|$$

can be formulated as a linear program. This LP is referred to as Least-Absolute-Error (LAE) LP or Least-Absolute-Deviation LP, or ℓ_1 linear regression [Wikipedia, 2010]. Together with standard uniform convergence bounds for linear functions [Vapnik, 1998], ℓ_1 linear regression gives a general technique for learning with ℓ_1-error.

Theorem 3.10. *Let $\mathcal{F}$ be a class of real-valued functions from $\{-1,1\}^n$ to $[-B, B]$ for some $B > 0$, $\mathcal{D}$ be distribution on $\{-1,1\}^n$ and $\phi_1, \phi_2, \ldots, \phi_m : \{-1,1\}^n \to \mathbb{R}$ be a set of functions that can be evaluated in time polynomial in n. Assume that there exists Δ such that for each $f \in \mathcal{F}$, there exist reals $\alpha_1, \alpha_2, \ldots, \alpha_m$ such that*

$$\mathop{\mathbf{E}}_{x \sim \mathcal{D}} \left[\left| \sum_{i \le m} \alpha_i \phi_i(x) - f(x) \right| \right] \le \Delta.$$

Then there is an algorithm that for every $\epsilon > 0$ and any distribution $\mathcal{P}$ on $\{-1,1\}^n \times [0,1]$ such that the marginal of $\mathcal{P}$ on $\{-1,1\}^n$ is $\mathcal{D}$, given access to random samples from $\mathcal{P}$, with probability at least $2/3$, outputs a function h such that $\mathbf{E}_{(x,y)\sim\mathcal{P}}[|h(x) - y|] \le \Delta + \epsilon$. The algorithm uses $O(m \cdot B^2/\epsilon^2)$ examples, runs in time polynomial in n, m, B/ϵ and returns a linear combination of ϕ_i's.

Remark 3.1. *Additional linear constraints on α_i's can be added to this LP as long as these constraints are satisfied by linear combinations that approximate each $f \in \mathcal{F}$ within Δ. In particular we will use this LP with each α_i being constrained to be non-negative and their sum being at most 1.*

We remark that this approach to learning can equivalently be seen as learning based on Empirical Risk Minimization with absolute loss [Vapnik, 1998]. For a Boolean target function, a hypothesis with ℓ_1 error of ϵ also gives a hypothesis with classification error of ϵ (e.g. [Kalai et al., 2008]). Therefore, as demonstrated by Kalai et al. [2008], ℓ_1 linear regression is also useful for agnostic learning of Boolean functions.

3.4.2 The Algorithm

The basic idea of the algorithm is to find a small set $\mathbb{S}$ of indices for which there exists non-negative reals α_S with $\sum_{S \in \mathbb{S}} \alpha_S \le 1$ such that $\mathbf{E}[|c(x) - \sum_{S \in \mathbb{S}} \alpha_S \cdot \mathsf{OR}_S(x)|] \le \epsilon$. Given $\mathbb{S}$, we can find a good non-negative linear combination as above using ℓ_1 linear regression as described in Thm. 3.10 and Remark 3.1. We will show that for any coverage function c, there exists a set of indices $\mathbb{S}$ (that depends on c) as above and moreover, that we can find such a set $\mathbb{S}$ using just random examples labeled by c. This will give us our proper learning algorithm.

To find a set $\mathbb{S}$ as above, we use the Fourier coefficients of the target coverage function. First, we will invoke Thm. 3.4 to find a set $\tilde{I} \subseteq [n]$ of coordinates of size $O(1/\epsilon^2)$ just as in Thm. 3.5 and show that we need only look at subsets of $\tilde{I}$ to find $\mathbb{S}$. Following this, we will prove that $\mathbb{S}$ can be identified using the following property: if $S \in \mathbb{S}$, then the Fourier coefficient $\hat{c}(S)$ is large enough in magnitude.

We begin with a simple lemma that shows that if a disjunction has a small number of variables and a significant coefficient in a coverage function then the corresponding (i.e. with the same index set) Fourier coefficient is significant.

Lemma 3.11. *Let $c = \sum_{S \subseteq [n]} \alpha_S \mathsf{OR}_S$ be a coverage function. Then, for any $T \subseteq [n]$, $|\hat{c}(T)| \geq 2^{-|T|} \cdot \alpha_T$.*

Proof. From the proof of Lem. 1.4, we have for any non-empty $T \neq \emptyset$, $\hat{c}(T) = -\sum_{S \supseteq T} \alpha_S \cdot (\frac{1}{2^{|S|}})$. Each term in the summation above is non-negative. Thus, $|\hat{c}(T)| \geq 2^{-|T|} \cdot \alpha_T$. □

We now show that for a coverage function $c \in \mathcal{CV}$, using significant Fourier coefficients of c, we can identify a coverage function c' of size at most $\min\{\mathsf{size}(c), (1/\epsilon)^{O(\log(1/\epsilon))}\}$ that ℓ_1 approximates c within ϵ. Moreover, c' depends on just $O(1/\epsilon^2)$ variables.

Lemma 3.12. *Let $c : \{-1,1\}^n \to [0,1]$ be a coverage function and $\epsilon > 0$. Let $I = \{i \in [n] \mid |\hat{c}(\{i\})| \geq \epsilon^2/18\}$. Set $s_\epsilon = \min\{\mathsf{size}(c), |I|^{\log(3/\epsilon)}\}$ and let*

$$\mathbb{T}_\epsilon = \{T \subseteq I \mid |\hat{c}(T)| \geq \epsilon^2/(9s_\epsilon) \text{ and } |T| \leq \log(3/\epsilon)\} \cup \{\emptyset\}.$$

Then,

1. *$|I| \leq 36/\epsilon^2$,*
2. *There exists $c' \in \mathcal{CV}$ such that $c' = \sum_{T \in \mathbb{T}_\epsilon} \alpha'_T \cdot \mathsf{OR}_T$ and $\|c - c'\|_1 \leq \epsilon$.*

Proof. Let α_T for each $T \subseteq [n]$ be the coefficients of the disjunctions of c, that is, $c = \sum_{T \subseteq [n]} \alpha_T \cdot \mathsf{OR}_T$ for non-negative α_T satisfying $\sum_{T \subseteq [n]} \alpha_T \leq 1$. Using Thm. 3.4, we know that the coverage function c_I (defined as $c_I(x) = \mathbf{E}_{y \sim \{-1,1\}^{\bar{I}}}[c(x_I \circ y])$ depends only on variables in I and $\|c - c_I\|_1 \leq \epsilon/3$. Therefore, $c_I = \sum_{T \subseteq I} \beta_T \cdot \mathsf{OR}_T$ for some constants $\beta_T \geq 0$, $\sum_{T \subseteq I} \beta_T \leq 1$. Using Lem. 3.1, we obtain that $|I| \leq 36/\epsilon^2$ and thus $s_\epsilon \leq \min\{\mathsf{size}(c), (6/\epsilon)^{2\log(3/\epsilon)}\}$.

Let $\mathbb{T}_1 = \{T \subseteq I \mid 0 < \beta_T \leq \frac{\epsilon}{3s_\epsilon}$ and $|T| \leq \log(\frac{3}{\epsilon})\}$ and $\mathbb{T}_2 = \{T \subseteq I \mid |T| > \log(\frac{3}{\epsilon})\}$. Consider any $T \subseteq I$ such that $T \notin T_1 \cup T_2$. Then $|T| < \log(3/\epsilon)$ and $\beta_T > \epsilon/(3s_\epsilon)$. This, by Lem. 3.11 applied to c_I, implies that $|\hat{c}_I(T)| > \epsilon^2/(9s_\epsilon)$. By Lem. 3.3, $|\hat{c}(T)| = |\hat{c}_I(T)| \geq \epsilon^2/(9s_\epsilon)$. This implies that $T \in \mathbb{T}_\epsilon$ and thus every $T \subseteq I$ is in $\mathbb{T}_\epsilon \cup \mathbb{T}_1 \cup \mathbb{T}_2$.

Set $v = \sum_{T \in \mathbb{T}_2} \beta_T$ and let $c' = \sum_{T \in \mathbb{T}_\epsilon} \beta_T \mathsf{OR}_T(x) + v$. Clearly $c' \in \mathcal{CV}$ and $c' = \sum_{T \in \mathbb{T}_\epsilon} \alpha'_T \cdot \mathsf{OR}_T$ for some coefficients α'_T. For c' we have:

$$\begin{aligned}
\|c_I - c'\|_1 &= \mathbf{E}\left[\left|\sum_{T \subseteq I} \beta_T \cdot \mathsf{OR}_T(x) - \left(\sum_{T \in \mathbb{T}_\epsilon} \beta_T \mathsf{OR}_T(x) + \sum_{T \in \mathbb{T}_2} \beta_T\right)\right|\right] \\
&\leq \mathbf{E}\left[\left|\sum_{T \in \mathbb{T}_1} \beta_T \cdot \mathsf{OR}_T(x)\right|\right] + \mathbf{E}\left[\left|\sum_{T \in \mathbb{T}_2} \beta_T \cdot (\mathsf{OR}_T(x) - 1)\right|\right] \\
&= \sum_{T \in \mathbb{T}_1} \mathbf{E}\left[|\beta_T \mathsf{OR}_T(x)|\right] + \sum_{T \in \mathbb{T}_2} \beta_T \cdot \mathbf{Pr}[\mathsf{OR}_T(x) = 0]
\end{aligned}$$

By Thm. 3.4, $\mathsf{size}(c_I) \leq \mathsf{size}(c)$ and therefore $|\mathbb{T}_1| \leq s_\epsilon$. For each $T \in \mathbb{T}_2$, $|T| > \log\left(\frac{3}{\epsilon}\right)$ which gives $\mathbf{Pr}[\mathsf{OR}_T(x) = 0] < \epsilon/3$. Therefore,

$$\|c_I - c'\|_1 \leq |\mathbb{T}_1| \cdot \frac{\epsilon}{3s_\epsilon} + \epsilon/3 \cdot 1 \leq 2\epsilon/3.$$

Thus, $\|c - c'\|_1 \leq \|c - c_I\|_1 + \|c_I - c'\|_1 \leq \epsilon$. □

We can now describe and analyze our proper PAC learning algorithm for $\mathcal{CV}$.

Theorem 3.13 (Proper PAC Learning)**.** *There exists an algorithm, that for any $\epsilon > 0$, given random and uniform examples of any $c \in \mathcal{CV}$, with probability at least $2/3$, outputs $h \in \mathcal{CV}$ such that $\|h - c\|_1 \leq \epsilon$. Further,* $\mathsf{size}(h) = O(s_\epsilon/\epsilon^2)$ *and the algorithm runs in time* $\tilde{O}(n) \cdot \mathsf{poly}(s_\epsilon/\epsilon)$ *and uses* $\log(n) \cdot \tilde{O}(s_\epsilon^2/\epsilon^4)$ *random examples, where* $s_\epsilon = \min\{\mathsf{size}(c), (12/\epsilon)^{2\lceil \log(6/\epsilon)\rceil}\}$.

Algorithm 2 *Proper* PAC Learning of Coverage Functions

```
Set $\theta = \frac{\epsilon^2}{108}$, $s_\epsilon = \min\{\mathsf{size}(c), (12/\epsilon)^{\lceil \log(6/\epsilon)\rceil}\}$.
Draw a random sample of size $m_1 = O(\log(n)/\epsilon^4)$ and use it to estimate, $\hat{c}(\{i\})$ for all $i$.
Set $\tilde{I} = \{i \in [n] \mid |\tilde{c}(\{i\})| \geq \theta\}$.
$\mathbb{S} \leftarrow \{\emptyset\}$.
Draw random sample $R$ of size $m_2 = O(s_\epsilon^2 \log(s_\epsilon/\epsilon)/\epsilon^4)$.
for $t = 1$ to $\lceil \log(6/\epsilon)\rceil$ do
    for each set $T \in \mathbb{S}$ of size $t-1$ and $i \in \tilde{I} \setminus T$ do
        Use $R$ to estimate the coefficient $\hat{c}(T \cup \{i\})$.
        If $|\tilde{c}(T \cup i)| \geq \theta$ then $\mathbb{S} \leftarrow \mathbb{S} \cup \{T \cup \{i\}\}$
    end for
end for
Draw a random sample $R$ of size $m_3 = O(s_\epsilon/\epsilon^4)$ and use $\ell_1$ linear regression to minimize $\sum_{(x,y)\in R}[|y - \sum_{S\in\mathbb{S}} \alpha_S \cdot \mathsf{OR}_S(x)|]$ subject to $\sum_{S\in\mathbb{S}} \alpha_S \leq 1$ and $\alpha_S \geq 0$ for all $S \in \mathbb{S}$. Let $\alpha_S^*$ for each $S \in \mathbb{S}$ be the solution.
return $\sum_{S\in\mathbb{S}} \alpha_S^* \cdot \mathsf{OR}_S$.
```

Proof. We first describe the algorithm and then present the analysis. We break the description of the algorithm into three stages. The first two stages are similar to those of Algorithm 1.

Let $\theta = \epsilon^2/108$. In the first stage, our algorithm finds a set $\tilde{I}$ of variables that contains $I = \{i \in [n] \mid |\hat{c}(\{i\})| \geq (\epsilon/2)^2/18\}$. We do this by estimating all the singleton Fourier coefficients, $\{\hat{c}(\{i\}) \mid i \in [n]\}$ within $\theta/2$ with (overall) probability at least $8/9$. As before, we denote the estimate of $\hat{c}(S)$ by $\tilde{c}(S)$ for any S. We then set $\tilde{I} = \{i \in [n] \mid \tilde{c}(\{i\})| \geq \theta\}$. If all the estimates are within $\theta/2$ of the corresponding coefficients then for every $i \in I$, $\tilde{c}(\{i\}) \geq \epsilon^2/72 - \theta/2 = \epsilon^2/108 = \theta$. Therefore $i \in \tilde{I}$ and hence $I \subseteq \tilde{I}$.

In the second stage, the algorithm finds a set $\mathbb{S} \subseteq 2^{\tilde{I}}$ such that the set of all large Fourier coefficients $\mathbb{T}$ is included in $\mathbb{S}$. Just as in Thm. 3.5, this is done iteratively starting with $\mathbb{S} = \{\emptyset\}$. In every iteration, for every set S that was added in the previous iteration and every $i \in \tilde{I} \setminus S$, it estimates $\hat{c}(S \cup \{i\})$ within $\epsilon^2/(108 s_\epsilon)$ (the success probability for all estimates in this phase will be $8/9$). If $|\tilde{c}(S \cup \{i\})| \geq \epsilon^2/(54 s_\epsilon)$ then $S \cup \{i\}$ is added to $\mathbb{S}$. The iterative process is run for at most $\lceil \log(6/\epsilon)\rceil$ iterations.

Finally, the algorithm draws a random sample R of size m_3 and uses ℓ_1 linear regression (Thm. 3.10) to minimize $\sum_{(x,y)\in R}[|y - \sum_{S\in\mathbb{S}} \alpha_S \cdot \mathsf{OR}_S(x)|]$ subject to $\sum_{S\in\mathbb{S}} \alpha_S \leq 1$ and $\alpha_S \geq 0$ for all $S \in \mathbb{S}$. Let

α^*_S for each $S \in \mathbb{S}$ be the solution. Here m_3 is chosen so that, with probability at least $8/9$, $\mathbf{E}[|c(x) - \sum_{S\in\mathbb{S}} \alpha^*_S \cdot \mathsf{OR}_S(x)]$ (the true error of $\sum_{S\in\mathbb{S}} \alpha^*_S \cdot \mathsf{OR}_S(x)$) is within $\epsilon/2$ of the optimum. Standard uniform convergence bounds imply that $m_3 = O(|\mathbb{S}|/\epsilon^2)$ examples suffice. The algorithm returns $\sum_{S\in\mathbb{S}} \alpha^*_S \cdot \mathsf{OR}_S$ as the hypothesis.

We can now prove the correctness of the algorithm assuming that all the estimates are successful. By an argument similar to the one presented in the proof of Thm. 3.5, we can verify that $I \subseteq \tilde{I}$ and that $\mathbb{T} \subseteq \mathbb{S}$. Using Thm. 3.12 and the facts that $\mathbb{S} \supseteq \mathbb{T}$ and $\tilde{I} \supseteq I$, we obtain that there must exist non-negative α'_S with $\sum_{S\in\mathbb{S}} \alpha'_S \leq 1$ such that $\mathbf{E}[|c(x) - \sum_{S\in\mathbb{S}} \alpha_S \cdot \mathsf{OR}_S(x)|] \leq \epsilon/2$. Thus ℓ_1 linear regression in the third stage will return coefficients α^*_S for each $S \in \mathbb{S}$ such that $\mathbf{E}[|c(x) - \sum_{S\in\mathbb{S}} \alpha^*_S \cdot \mathsf{OR}_S(x)|] \leq \epsilon$.

We now analyze the running time and sample complexity of the algorithm. The analysis for the first two stages is similar to the one presented in Thm. 3.5.

- Just as in the proof of Thm. 3.5, all the singleton coefficients can be estimated in time $O(n \log{(n)}/\epsilon^4)$ and samples $O(\log{(n)}/\epsilon^4)$ with confidence at least $8/9$.

- For every i such that $\hat{c}(\{i\})$ was estimated within $\theta/2$ and $|\tilde{c}(\{i\})| \geq \theta$, we have that $|\hat{c}(\{i\})| \geq \theta/2$. This implies that $|\tilde{I}| \leq 2/(\theta/2) = O(1/\epsilon^2)$. Similarly, for every $S \subseteq \tilde{I}$ such that $\hat{c}(S)$ was estimated within $\epsilon^2/(108 s_\epsilon)$ and $|\tilde{c}(S)| \geq \epsilon^2/(54 s_\epsilon)$, we have that $|\hat{c}(S)| \geq \epsilon^2/108 s_\epsilon$. Thus using Lem. 3.1, $|\mathbb{S}| = O(s_\epsilon/\epsilon^2)$.

- In the second stage the algorithm only estimates coefficients with indices that are subsets of

$$\mathbb{S}' = \{S \cup \{i\} \mid |\tilde{c}(S)| \geq \epsilon^2/(54 s_\epsilon) \text{ and } i \in \tilde{I} \text{ and } |S| \leq \log{(6/\epsilon)}\}.$$

  We can conclude that $|\mathbb{S}'| = O(s_\epsilon/\epsilon^4)$ and, as in the proof of Thm. 3.5, the estimation succeeds with probability at least $8/9$ using $\tilde{O}(s_\epsilon^2/\epsilon^4)$ examples and running in time $\tilde{O}(s_\epsilon^2/\epsilon^8)$.

- Finally, using Thm. 3.10, ℓ_1 linear regression will require $O(|\mathbb{S}|/\epsilon^2) = O(s_\epsilon/\epsilon^4)$ random examples and runs in time polynomial in $|S|$ and $1/\epsilon$, that is $\mathsf{poly}(s_\epsilon/\epsilon)$.

Overall, the algorithm succeeds with probability at least $2/3$, runs in time $\tilde{O}(n) \cdot \mathsf{poly}(s_\epsilon/\epsilon)$ and uses $\log{(n)} \cdot \tilde{O}(s_\epsilon^2/\epsilon^4)$ random examples. □

4 Agnostic Learning on Product and Symmetric Distributions

In this section, we give optimal algorithms for agnostically learning (we give a formal definition below) coverage functions on arbitrary product and symmetric distributions. Recall that a distribution is symmetric if the associated probability density function is symmetric on $\{-1,1\}^n$. We begin by recalling the definition of agnostic learning with ℓ_1-error.

Definition 4.1. *Let $\mathcal{F}$ be a class of real-valued functions on $\{-1,1\}^n$ with range in $[0,1]$ and let $\mathcal{D}$ be any fixed distribution on $\{-1,1\}^n$. For any distribution $\mathcal{P}$ over $\{-1,1\}^n \times [0,1]$, let opt$(\mathcal{P},\mathcal{F})$ be defined as: opt$(\mathcal{P},\mathcal{F}) = \inf_{f\in\mathcal{F}} \mathbf{E}_{(x,y)\sim\mathcal{P}}[|y - f(x)|]$. An algorithm $\mathcal{A}$, is said to agnostically learn $\mathcal{F}$ on $\mathcal{D}$ if for every* excess error $\epsilon > 0$ and any distribution $\mathcal{P}$ on $\{-1,1\}^n \times [0,1]$ such that the marginal of $\mathcal{P}$ on $\{-1,1\}^n$ is $\mathcal{D}$, given access to random independent examples drawn from $\mathcal{P}$, with probability at least $\frac{2}{3}$, $\mathcal{A}$ outputs a hypothesis h such that $\mathbf{E}_{(x,y)\sim\mathcal{P}}[|h(x) - y|] \leq \mathsf{opt}(\mathcal{P},\mathcal{F}) + \epsilon$.

Our learning result is based on a simple observation (a special case of which is implicit in [Cheraghchi et al., 2012]) that an ℓ_1-approximation on a distribution $\mathcal{D}$ for all monotone disjunctions by linear combination of functions from a fixed set of functions yields a similar approximation for $\mathcal{CV}$ on $\mathcal{D}$. For completeness a proof is included in App. A.

Lemma 4.2. *Fix a distribution $\mathcal{D}$ on $\{-1,1\}^n$. Suppose there exist functions $\phi_1, \phi_2, \ldots, \phi_m : \{0,1\}^n \to \mathbb{R}$ such that for any $S \subseteq [n]$, there are reals $\beta_1^S, \beta_2^S, \ldots, \beta_m^S$ such that $\|\mathsf{OR}_S - \sum_{j=1}^m \beta_j^S \cdot \phi_j\|_1 = \mathbf{E}_{x\sim\mathcal{D}}[|\mathsf{OR}_S(x) - \sum_{j=1}^m \beta_j^S \cdot \phi_j(x)|] \leq \epsilon$. Then, for every coverage function $c \in \mathcal{CV}$, there exist reals $\beta_1, \beta_2, \ldots, \beta_m$ such that $\|c - \sum_{j=1}^m \beta_j \cdot \phi_j\|_1 = \mathbf{E}_{x\sim\mathcal{D}}[|c(x) - \sum_{j=1}^m \beta_j \cdot \phi_j(x)|] \leq \epsilon$.*

A natural and commonly used set of basis functions is the set of all monomials on $\{-1,1\}^n$ of some bounded degree. It is easy to see that on product distributions with constant bias, disjunctions longer than some constant multiple of $\log(1/\epsilon)$ are ϵ-close to constant 1. Therefore degree $O(\log(1/\epsilon))$ suffices for ℓ_1 approximation on such distributions. This simple argument does not work for general product distributions. However it was shown by Blais et al. [2008] that the same degree (up to a constant factor) still suffices in this case. Their argument is based on the analysis of noise sensitivity under product distributions and implies additional interesting results. A simpler proof of this fact also appears in [Feldman and Kothari, 2014], who also show that the same holds if the distribution is uniform over points of Hamming weight k, for any fixed $k \in \{0, \ldots, n\}$.

Lemma 4.3 (Feldman and Kothari, 2014)**.** *For $0 \leq k \leq n$, let Π_k denote the uniform distribution over points of Hamming weight k. For every disjunction f and $\epsilon > 0$, there exists a polynomial p of degree at most $O(\log(1/\epsilon))$ such that $\mathbf{E}_{x\sim\Pi_k}[|f(x) - p(x)] \leq \epsilon$.*

This result implies a basis for approximating disjunctions over arbitrary symmetric distributions. All we need is to partition the domain $\{-1,1\}^n$ into $\cup_{0\leq k\leq n} S_k$ layers and use a (different) polynomial for each layer. Formally, the basis now contains functions of the form $\mathrm{IND}(k) \cdot \chi$, where IND is the indicator function of being in layer of Hamming weight k and χ is a monomial of degree $O(\log(1/\epsilon))$. In contrast, as shown in [Feldman and Kothari, 2014], there exist symmetric distributions over which disjunctions cannot be ℓ_1-approximated by low-degree polynomials.

These results together with the ℓ_1 regression in Thm. 3.10 immediately yield an agnostic learning algorithm for the class of coverage functions $\mathcal{CV}$ over any product or symmetric distribution.

Theorem 4.4 (Thm. 1.7 restated)**.** *There exists an algorithm that for any product or symmetric distribution $\mathcal{D}$ agnostically learns $\mathcal{CV}$ with excess ℓ_1 error ϵ in time $n^{O(\log 1/\epsilon)}$.*

We now remark that any algorithm that agnostically learns the class of coverage functions on n inputs on the uniform distribution on $\{0,1\}^n$ in time $n^{o(\log(\frac{1}{\epsilon}))}$ would yield a faster algorithm for the notoriously hard problem of learning sparse parities with noise. The reduction only uses the fact that coverage functions include all monotone disjunctions and follows from the results in [Kalai et al., 2008, Feldman, 2012] (see [Feldman and Kothari, 2014] for details).

4.1 Proper Agnostic Learning

For the special case of *bounded* product distributions, that is, product distributions with one dimensional marginal expectations bounded away from 0 and 1 by constants, we can in fact obtain a *proper* agnostic learner for $\mathcal{CV}$. The proof is based on approximating coverage functions by truncating the expansion of a coverage function in terms of monotone disjunctions (Lem. 2.1) to keep only the terms corresponding to short disjunctions. We show that such a truncation is enough to approximate the function with respect to ℓ_1 error. Recall that by Lem. 2.1, such a truncation is itself a coverage function.

Lemma 4.5. *Let $c(x) = \sum_{S \subseteq [n]} \alpha_S \cdot \mathsf{OR}_S(x)$ be a coverage function with range in $[0,1]$ and $\epsilon > 0$. Let $\mathcal{D}$ be any product distribution such that the one-dimensional marginal expectations of $\mathcal{D}$ are at most $1-\kappa$. Then, for $k = 2/\kappa \cdot \lceil \log\left(\frac{1}{\epsilon}\right) \rceil$ the coverage function $c' = \sum_{|S|>k} \alpha_S + \sum_{|S| \leq k} \alpha_S \cdot \mathsf{OR}_S$ satisfies $\mathbf{E}_{x \sim \mathcal{D}}[|c - c'|] \leq \epsilon$.*

Proof. By Lem. 2.1 and the fact that c is a coverage function we obtain that c' is a non-negative combination of monotone disjunctions with the sum of coefficients being at most 1, that is a coverage function itself. For each $x \in \{-1,1\}^n$, non-negativity of the coefficients α_S implies $c(x) \leq c'(x)$.

For a monotone disjunction OR_S, observe that if $|S| > k$, then $\mathbf{Pr}_{x \sim \mathcal{D}}[\mathsf{OR}_S(x) = 0] \leq (1-\kappa)^k \leq \epsilon$. Thus: Now,

$$\begin{aligned} \|c - c'\|_1 = \mathbf{E}[|c(x) - c'(x)|] &= \mathop{\mathbf{E}}_{x \sim \mathcal{D}}[c'(x) - c(x)] \\ &= \sum_{S \subseteq [n], |S| > k} \mathop{\mathbf{E}}_{x \in \mathcal{D}}[\alpha_S - \alpha_S \cdot \mathsf{OR}_S] = \sum_{S \subseteq [n], |S| > k} \mathop{\mathbf{Pr}}_{\mathcal{D}}[\mathsf{OR}_S = 0] \cdot \alpha_S \\ &\leq \sum_{S \subseteq [n], |S| > k} \alpha_S \cdot \epsilon \leq \epsilon. \end{aligned}$$

□

As an immediate corollary of Lem. 4.5 and Thm. 3.10, we obtain an algorithm for agnostic learning of coverage functions on the uniform distribution.

Theorem 4.6. *There exists an algorithm, that agnostically learns the class $\mathcal{CV}$ on any bounded product distributions in time $n^{O(\log\left(\frac{1}{\epsilon}\right))}$. Further, the hypothesis returned by the algorithm is itself a coverage function.*

Proof. Lem. 4.5 shows that every coverage function can be approximated by a non-negative linear combination of monotone disjunctions of length $\log\left(\frac{1}{\epsilon}\right)$ within ϵ in the ℓ_1-norm. Now, Thm. 3.10 with Remark 3.1 immediately yields an agnostic learning algorithm. □

5 Privately Releasing Monotone Conjunction Counting Queries

In this section we use our learning algorithms to derive privacy-preserving algorithms for releasing monotone conjunction (equivalently, disjunction) counting queries. We begin with the necessary formal definitions.

5.1 Preliminaries

Differential Privacy: We use the standard formal notion of privacy, referred to as *differential privacy*, proposed by Dwork et al. [2006]. For some domain X, we will call $D \subseteq X$ a *data set*. data sets $D, D' \subset X$ are *adjacent* if one can be obtained from the other by adding a single element. In this paper, we will focus on Boolean data sets, thus, $X \subseteq \{-1,1\}^n$ for $n \in \mathbb{N}$. We now define a differentially private algorithm. In the following, A is an algorithm that takes as input a data set D and outputs an element of some set R.

Definition 5.1 (Differential privacy [Dwork et al., 2006])**.** *An (randomized) algorithm $A : 2^X \to R$ is* ϵ-differentially private *if for all $r \in R$ and every pair of adjacent data sets D, D', we have* $\mathbf{Pr}[A(D) = r] \leq e^\epsilon\, \mathbf{Pr}[A(D') = r]$.

Private Counting Query Release: We are interested in algorithms that answer predicate *counting* queries on Boolean data sets. A predicate counting query finds the fraction of elements in a given data set that satisfy the predicate. More generally, given a query $c : \{-1,1\}^n \to [0,1]$, a counting query corresponding to c on a data set $D \subseteq \{-1,1\}^n$ of size $m := |D|$, expects in reply $q_c(D) = 1/m \sum_{r \in D} c(r)$. In our applications, we will only insist on answering the counting queries approximately, that is, for some $\tau > 0$, an approximate counting query in the setting above expects a value v that satisfies $|v - 1/m \sum_{r \in D} c(r)| \leq \tau$. We refer to τ as the *tolerance* of the counting query. A class of queries $\mathcal{C}$ mapping $\{-1,1\}^n$ into $[0,1]$, thus induces a *counting query function* $\mathbf{CQ}_D : \mathcal{C} \to [0,1]$ given by $\mathbf{CQ}_D(c) = q_c(D)$ for every $c \in \mathcal{C}$. For a class $\mathcal{C}$ of such functions and a data set D, the goal of a data release algorithm is to output a summary $H : \mathcal{C} \to [0,1]$ that provides (approximate) answers to queries in $\mathcal{C}$. A *private* data release algorithm additionally requires that H be produced in a differentially private way with respect to the participants in the data set. One very useful way of publishing a summary is to output a *synthetic data set* $\hat{D} \subseteq \{-1,1\}^n$ such that for any query $c \in \mathcal{C}$, $q_c(\hat{D})$ is a good approximation for $q_c(D)$. Synthetic data sets are an attractive method for publishing private summaries as they can be directly used in software applications that are designed to run on Boolean data sets in addition to being easily understood by humans.

For a class $\mathcal{C}$ of queries mapping $\{-1,1\}^n$ into $[0,1]$, and a distribution Π on $\mathcal{C}$, an algorithm A (α,β)-answers queries from $\mathcal{C}$ over a data set D on the distribution Π, if for $H = A(D)$, $\mathbf{Pr}_{f \sim \Pi}[|\mathbf{CQ}_D(f) - H(f)| \leq \alpha] \geq 1 - \beta$. For convenience we will only measure the average error $\bar{\alpha}$ and require that $\mathbf{E}_{f \sim \Pi}[|\mathbf{CQ}_D(f) - H(f)|] \leq \bar{\alpha}$. Clearly, one can obtain an (α,β)-query release algorithm from an $\bar{\alpha}$-average error query release algorithm by setting $\bar{\alpha} = \alpha \cdot \beta$.

The key observation for obtaining conjunction query release algorithms from learning algorithms for coverage functions is that for any data set D and the query class of monotone conjunctions, $1 - \mathbf{CQ}_D$ is a coverage function. Namely, for any $S \subseteq [n]$, let AND_S be the monotone conjunction $\wedge_{i \in S} x_i$ which equals 1 iff each $x_i = -1$ for $i \in S$ and for $x \in \{-1,1\}^n$ let $S_x \subseteq [n]$ be the set such that $x_i = -1$ iff $i \in S_x$. Then $c_D(x) \doteq 1 - \mathbf{CQ}_D(\mathsf{AND}_{S_x})$ is a coverage function. We include a simple proof of this fact for completeness.

Lemma 5.2. *For a data set D, let $c_D : \{-1,1\}^n \to [0,1]$ be defined as $c_D(x) = 1 - \mathbf{CQ}_D(\mathsf{AND}_{S_x})$. Then c_D is a coverage function.*

Proof. Let $x \in \{-1,1\}^n$. By definition,

$$c_D(x) = 1 - \mathbf{CQ}_D(\mathsf{AND}_{S_x}) = 1 - \frac{1}{|D|}\sum_{z \in D} \mathsf{AND}_{S_x}(z) = \frac{1}{|D|}\sum_{z \in D}(1 - \mathsf{AND}_{S_x}(z)).$$

Note that

$$1 - \mathsf{AND}_{S_x}(z) = 1 - \bigwedge_{x_i = -1} [z_i = -1] = \bigvee_{z_i = 1} [x_i = -1] = \mathsf{OR}_{S_{-z}}(x).$$

Then,

$$c_D(x) = \sum_{z \in D} \frac{1}{|D|} \cdot \mathsf{OR}_{S_{-z}}(x).$$

□

Lem. 5.2 implies that for the class of monotone conjunctions $\mathcal{C}$, the set of functions $\{1 - \mathbf{CQ}_D | D \subseteq \{-1,1\}^n\}$ is a subset of $\mathcal{CV}$. Additive error approximation for $c_D = 1 - \mathbf{CQ}_D$ is equivalent to additive error approximation of $\mathbf{CQ}_D$. Therefore to obtain a private release algorithm with average error $\bar{\alpha}$ relative to a distribution Π over monotone conjunctions, it is sufficient to produce a hypothesis h that satisfies, $\mathbf{E}_{x\sim\Pi}[|c_D(x) - h(x)|] \leq \bar{\alpha}$, where we view Π also as a distribution over vectors corresponding to conjunctions (with x corresponding to AND_{S_x}). Note that the average error is exactly the ℓ_1-error in approximation of $\mathbf{CQ}_D$ over distribution Π. A monotone conjunction query of length k corresponds to a point in $\{-1,1\}$ that has exactly k (-1)s.

To convert our learning algorithms to differentially-private release algorithms we rely on the following proposition that Gupta et al. [2011] prove using technique from [Blum et al., 2005].

Proposition 5.1 (Gupta et al., 2011)**.** *Let $\mathcal{A}$ denote an algorithm that uses q counting queries of tolerance τ in its computation. Then for every $\epsilon, \delta > 0$, with probability $1-\delta$, $\mathcal{A}$ can be simulated in an ϵ-differentially private way provided that the size of data set $|D| \geq q(\log q + \log(1/\delta))/(\epsilon \cdot \tau)$. Simulation of each query of $\mathcal{A}$ takes time $O(|D|)$.*

5.2 Releasing k-way Marginals with Low Average Error

We now describe a differentially private algorithm for releasing monotone conjunction counting queries of length k with low average error. The result is based on a simple implementation of the ℓ_1 linear regression algorithm using tolerant counting query access to the data set D. Let $\mathcal{C}_k$ be the class of all monotone conjunctions of length $k \in [n]$ and let Π_k denote the uniform distribution over $\mathcal{C}_k$.

Theorem 5.3. *For every $\epsilon > 0$, there is an ϵ-differentially private algorithm which for any data set $D \subseteq \{-1,1\}^n$ of size $n^{\Omega(\log(1/\bar{\alpha}))} \cdot \log 1/\delta/\epsilon$, with probability at least $1-\delta$ publishes a data structure H that answers counting queries for $\mathcal{C}_k$ with an average error of at most $\bar{\alpha}$ relative to Π_k. The algorithm runs in time $n^{O(\log(1/\bar{\alpha}))} \cdot \log(1/\delta)/\epsilon$ and the size of H is $n^{O(\log(1/\bar{\alpha}))}$.*

Proof. In the light of the discussion above, we show how to implement the algorithm described in Thm. 1.7 to learn $\{c_D \mid D \subseteq \{-1,1\}^n\}$ with tolerant value query access to the data set D. We will simulate the algorithm described in Thm. 1.7 over distribution Π_k and with excess ℓ_1-error of $\bar{\alpha}/2$. The algorithm uses ℓ_1 linear regression to find a linear combination of $t = n^{O(\log(1/\bar{\alpha}))}$ monomials that best fits random examples (x^i, y^i).

We can simulate random examples of c_D by drawing x from Π_k and making the counting query on the conjunction AND_{S_x}. Since we can only use τ-tolerant queries, we are guaranteed that the value we obtain, denote it by $\tilde{c}_D(x)$, satisfies $|c_D(x) - \tilde{c}_D(x)| \leq \tau$. This additional error in values has average value of at most τ and hence can cause ℓ_1 linear regression to find a solution whose average absolute error is up to 2τ worse than the average absolute error of the optimal solution. This means that we are guaranteed that the returned polynomial H satisfies $\mathbf{E}[|c_D(x) - H(x)|] \leq 2\tau + \bar{\alpha}/2$. We set $\tau = \bar{\alpha}/4$ and obtain that the error is at most $\bar{\alpha}$. This implementation makes $n^{O(\log(1/\bar{\alpha}))}$ $(\bar{\alpha}/4)$-tolerant counting queries to the data set D and uses $n^{O(\log(1/\bar{\alpha}))}$ time to output H. Applying Proposition 5.1, we obtain that there exists a ϵ-differentially private algorithm to compute an H as above with the claimed bounds on the size of D and running time. □

5.3 Releasing All Marginals with Low Average Error

Next, we show that we can implement the algorithm from Thm. 3.5 using tolerant counting query access to the data set D and thereby obtain a private data release algorithm for monotone conjunctions with low average error relative to the uniform distribution. Notice that since we only promise low-average error over all monotone conjunction counting queries, for some k's the average error on conjunctions of length k can be very large.

Theorem 5.4 (Thm. 1.9 restated)**.** *Let $\mathcal{C}$ be the class of all monotone conjunctions. For every $\epsilon, \delta > 0$, there exists an ϵ-differentially private algorithm which for any data set $D \subseteq \{-1,1\}^n$ of size $\tilde{\Omega}(n \log(1/\delta)/(\epsilon\bar{\alpha}^6))$, with probability at least $1-\delta$, publishes a data structure H that answers counting queries for $\mathcal{C}$ with respect to the uniform distribution with average error of at most $\bar{\alpha}$. The algorithm runs in time $\tilde{O}(n^2 \log(1/\delta)/(\epsilon\bar{\alpha}^{10}))$ and the size of H is $\log n \cdot \tilde{O}(1/\bar{\alpha}^4)$.*

Proof. In the algorithm from Thm. 3.5 the random examples of the target coverage function c are used only to estimate Fourier coefficients of c within tolerance $\theta/2$. Thus to implement the algorithm from Thm. 3.5, it is sufficient to show that for any index set $T \subseteq [n]$, we can compute $\widehat{c_D}(T)$ within $\theta/2$ using tolerant counting query access to D.

Consider any set $T \subseteq [n]$ and recall that for any $x \in \{-1,1\}^n$, $S_x = \{i \mid x_i = -1\}$. From the proof of Lemma 5.2, we have: $c_D = 1/|D| \sum_{z \in D} \mathsf{OR}_{S_{-z}}$. Then, we have

$$\begin{aligned}\widehat{c_D}(T) &= \mathop{\mathbf{E}}_{x \sim \mathcal{U}_n}[c_D(x) \cdot \chi_T(x)] = \mathop{\mathbf{E}}_{x \sim \mathcal{U}_n}\left[\frac{1}{|D|}\sum_{z \in D} \mathsf{OR}_{S_{-z}}(x) \cdot \chi_T(x)\right] \\ &= \frac{1}{|D|} \mathop{\mathbf{E}}_{x \sim \mathcal{U}_n}[\mathsf{OR}_{S_{-z}}(x) \cdot \chi_T(x)] = \frac{1}{|D|}\sum_{z \in D} \widehat{\mathsf{OR}}_{S_{-z}}(T). \end{aligned} \tag{2}$$

Define $F_T(z) = (1 + \widehat{\mathsf{OR}}_{S_{-z}}(T))/2$. Now F_T is a function with range $[0,1]$ and from equation (2) above, we observe that $\widehat{c_D}(T)$ can be estimated with tolerance $\theta/2$ by making a counting query for F_T on D with tolerance $\theta/4$. We now note that ℓ_1-error of hypothesis h over the uniform distribution on $\{-1,1\}^n$ is the same as the average error $\bar{\alpha}$ of answering counting queries using h over the uniform distribution on monotone disjunctions. Therefore $\theta/4 = (\bar{\alpha})^2/24$. The number of queries made by the algorithm is exactly equal to the number of Fourier coefficients estimated by it which is $O(n + 1/\bar{\alpha}^4)$. The output of the PAC learning algorithm is a linear combination of $O(1/\bar{\alpha}^2)$ parities over a subset of $O(1/\bar{\alpha}^2)$ variables and hence requires $\log n \cdot \tilde{O}(1/\bar{\alpha}^4)$ space. Note that given correct estimates of Fourier coefficients, the PAC learning algorithm is always successful. By applying Proposition 5.1, we can obtain an ϵ-differentially private execution of the PAC learning algorithm that succeeds with probability at least $1-\delta$ provided that the data set size is

$$\Omega\left(\frac{(n + \bar{\alpha}^4)\log(n + \bar{\alpha}^4) + \log(1/\delta)}{\epsilon\bar{\alpha}^2}\right) = \tilde{\Omega}(n \log(1/\delta)/(\epsilon\bar{\alpha}^6)).$$

The running time is dominated by the estimation of Fourier coefficients and hence is $O(n + 1/\bar{\alpha}^4) = O(n/\bar{\alpha}^4)$ times the size of the data set. □

We now use our proper PAC learning algorithm for coverage functions to obtain an algorithm for synthetic data set release for answering monotone conjunction counting queries.

Theorem 5.5. *Let $\mathcal{C}$ be the class of all monotone conjunctions. For every $\epsilon, \delta > 0$, there exists an ϵ-differentially private algorithm which for any data set $D \subseteq \{-1,1\}^n$ of size $n \cdot \bar{\alpha}^{-\Omega(\log(1/\bar{\alpha}))} \cdot \log(1/\delta)/\epsilon$, with probability at least $1-\delta$, releases a synthetic data set $\hat{D}$ of size $\bar{\alpha}^{-O(\log(1/\bar{\alpha}))}$ that can answer counting queries for $\mathcal{C}$ with respect to the uniform distribution with average error of at most $\bar{\alpha}$. The algorithm runs in time $n^2 \cdot \bar{\alpha}^{-O(\log(1/\bar{\alpha}))} \cdot \log(1/\delta)/\epsilon$.*

Proof. We will show that we can implement our proper PAC learning algorithm for coverage functions (Algorithm 2) to learn $\{c_D \mid D \subseteq \{-1,1\}^n\} \subseteq \mathcal{CV}$ using tolerant counting query access to the data set D. Algorithm 2 returns a hypothesis $H : \{-1,1\}^n \to [0,1]$ given by a non-negative linear combination of monotone disjunctions that ℓ_1-approximates c_D. Then, H' defined by $H'(x) = 1 - H(x)$, ℓ_1 approximates $\mathbf{CQ}_D$. We will then show that we can construct a data set $\hat{D}$ using H, such that $c_{\hat{D}}$ computes the same function as H, up to a small discretization error. Now, $\mathbf{CQ}_{\hat{D}}(x) = 1 - c_{\hat{D}}(x)$, and thus, answering monotone conjunction queries based on the data set $\hat{D}$ incurs low average error. As before, an ϵ-differentially private version of this algorithm is then obtained by invoking Proposition 5.1.

Algorithm 2 uses the random examples from the target function to first estimate certain Fourier coefficients and then to run the ℓ_1 linear regression (Thm. 3.10) to find the coefficients of the linear combination or disjunctions. We have already shown that tolerant counting queries on the data set can be used to estimate Fourier coefficients.

Recall that we proved that there exist monotone disjunctions $\mathsf{OR}_{S_1}, \mathsf{OR}_{S_2}, \ldots, \mathsf{OR}_{S_t}$ and non-negative reals γ_i for $i \in [t]$ satisfying $\sum_{i\in[t]} \gamma_i \leq 1$ such that $\mathbf{E}[|c_D(x) - \sum_{i\in[t]} \gamma_i \cdot \mathsf{OR}_{S_i}(x)|] \leq \Delta$, for some t and Δ. The algorithm, at this stage, runs ℓ_1 linear regression to compute non-negative coefficients for appropriately chosen monotone disjunctions. As we showed in Thm. 5.3, ℓ_1 linear regression can be implemented via τ-tolerant counting queries and results in the additional error of 2τ. That is we obtain non-negative reals γ_i^* for $i \in [t]$ such that $\mathbf{E}[|c_D(x) - \sum_{i\in[t]} \gamma_i^* \cdot \mathsf{OR}_{S_i}(x)|] \leq \Delta + \eta + 2\tau$.

Thus, we simulate Algorithm 2 with excess ℓ_1-error of $\bar{\alpha}/2$ and let τ be the minimum of tolerance required for estimating Fourier coefficients and $\bar{\alpha}/8$. The error of the coverage hypothesis function H is then at most $\bar{\alpha}/2 + 2\tau \leq 3\bar{\alpha}/4$. Inspecting the proof of Thm. 3.13 shows that $\tau = \bar{\alpha}^{O(\log(1/\alpha))}$ suffices. Thus, to summarize, we obtain that there exists an algorithm that makes $n + \bar{\alpha}^{-O(\log 1/\bar{\alpha})}$ τ-tolerant counting queries to the data set D (for τ as above) and uses $n \cdot \bar{\alpha}^{-O(\log 1/\bar{\alpha})}$ time to output a non-negative linear combination H of at most $\bar{\alpha}^{-O(\log 1/\bar{\alpha})}$ monotone disjunctions that satisfies $\|c_D - H\|_1 \leq 3\bar{\alpha}/4$. Applying Proposition 5.1, we obtain that there exists a ϵ-differentially private algorithm to compute an H as above with the claimed bounds on the size of D and running time.

We now convert our hypothesis $H(x) = \sum_{i\in[t]} \gamma_i^* \cdot \mathsf{OR}_{S_i}(x)$ into a data set by using the converse of Lem. 5.2. For each OR_S, let $x^S \in \{-1,1\}^n$ be defined by: for all $j \in [n]$, $x_j^S = -1$ if and only if $j \in S$. Our goal is to construct a data set $\hat{D}$ in which each x^{S_i} for $i \in [t]$ has a number of copies that is proportional to γ_i^* since this would imply that $c_{\hat{D}} = H$. To achieve this we first round down each γ_i^* to the nearest multiple of $\bar{\alpha}/(4t)$ and let $\tilde{\gamma}_i$ denote the result. The function $\tilde{H}(x) = \sum_{i\leq t} \tilde{\gamma}_i \cdot \mathsf{OR}_{S_i}(x)$ satisfies $\|\tilde{H} - H\|_1 \leq \bar{\alpha}/4$ and hence $\|\tilde{H} - c_D\|_1 \leq \bar{\alpha}$. Now we let $\hat{D}$ be the data set in which each x^{S_i} for $i \in [t]$ has $4t\tilde{\gamma}_i/\bar{\alpha}$ copies (an non-negative integer by our discretization). From Lem. 5.2 we see that $c_{\hat{D}} = \tilde{H}$. Note that the size of $\hat{D}$ is at most $4t/\bar{\alpha} = \bar{\alpha}^{-O(\log(1/\bar{\alpha}))}$.

□

6 Distribution-Independent Learning

6.1 Reduction from Learning Disjoint DNFs

In this section we show that distribution-independent learning of coverage functions is at least as hard as distribution-independent learning of disjoint DNF formulas.

Theorem 6.1 (Thm. 1.1 restated)**.** *Let $\mathcal{A}$ be an algorithm that distribution-independently PAC learns the class of all size-s coverage functions from $\{-1,1\}^n$ to $[0,1]$ in time $T(n, s, \frac{1}{\epsilon})$. Then, there exists an algorithm $\mathcal{A}'$*

that PAC learns of s-term disjoint DNFs in time $T(2n, s, \frac{2s}{\epsilon})$.

Proof. Let $d = \vee_{i \leq s} T_i$ be a disjoint DNF with s terms. Disjointness of terms implies that $d(x) = \sum_{i \leq s} T_i(x)$ for every $x \in \{-1,1\}^n$. By using de Morgan's law, we have: $d = s - \sum_{i \leq s} D_i$ where each D_i is a disjunction on the negated literals in T_i. We will now use a standard reduction [Kearns et al., 1994a] through a one-to-one map $m : \{-1,1\}^n \rightarrow \{-1,1\}^{2n}$ and show that there exists a sum of *monotone* disjunctions d' on $\{-1,1\}^{2n}$ such that for every $x \in \{-1,1\}^n$, $d'(m(x)) = s - d(x)$. The mapping m maps $x \in \{-1,1\}^n$ to $y \in \{-1,1\}^{2n}$ such that for each $i \in [n]$, $y_{2i-1} = x_i$ and $y_{2i} = -x_i$. To define d', we modify each disjunction D_j in the representation of d to obtain a monotone disjunction D'_j and set $d' = \sum_{j \leq s} D'_j$. For each x_i that appears in D_j we include y_{2i-1} in D'_j and for each $\neg x_i$ in D_j we include y_{2i}. Thus D'_j is a monotone disjunction on $y_1, \ldots, y_{2n}$. It is easy to verify that $d(x) = s - d'(m(x))$ for every $x \in \{-1,1\}^n$. Now $d'/s = 1 - d/s$ is a convex combination of monotone disjunctions, that is, a coverage function.

We now describe the reduction itself. As usual, we can assume that the number of terms in the target disjoint DNF, is known to the algorithm. This assumption can be removed via the standard "guess-and-double" trick. Given random examples drawn from a distribution $\mathcal{D}$ on $\{-1,1\}^n$ and labeled by a disjoint DNF d and $\epsilon > 0$, $\mathcal{A}'$ converts each such example (x, y) to example $(m(x), 1 - \frac{y}{s})$. On the modified examples, $\mathcal{A}'$ runs the algorithm $\mathcal{A}$ with error parameter $\epsilon/(2s)$ and obtains a hypothesis h'. Finally, $\mathcal{A}$ returns the hypothesis $h(x) =$ "$s(1 - h'(m(x))) \geq 1/2$" (that is $h(x) = 1$ if $s(1 - h'(m(x))) \geq 1/2$ and $h(x) = 0$ otherwise). To establish the correctness of $\mathcal{A}'$ we show that $\mathbf{Pr}_{x \sim \mathcal{D}}[d(x) \neq h(x)] \leq \epsilon$. By the definition of $h(x)$ we have that $h(x) \neq d(x)$ only if $|d(x) - s(1 - h'(m(x)))| \geq 1/2$. Thus, by the correctness of $\mathcal{A}$, we have

$$\mathop{\mathbf{Pr}}_{x \sim \mathcal{D}}[d(x) \neq h(x)] \leq 2 \mathop{\mathbf{E}}_{x \sim \mathcal{D}}[|(d(x) - (s - sh'(m(x)))|] = 2s \cdot \mathop{\mathbf{E}}_{x \sim \mathcal{D}}[|h'(m(x)) - (1 - \frac{d(x)}{s})|] \leq \epsilon.$$

Finally, the running time of our simulation is dominated by the running time of $\mathcal{A}$.

□

6.2 Reduction to Learning of Thresholds of Monotone Disjunctions

We give a general reduction of the problem of learning a class of bounded real-valued functions $\mathcal{C}$ to the problem of learning linear thresholds of $\mathcal{C}$. It is likely folklore in the context of PAC learning and was employed by Hardt et al. [2012] in a reduction from the problem of private data release. Here we give a slightly more involved analysis that also shows that this reduction applies in the agnostic learning setting. The reduction preserves the distribution on the domain and hence can be applied both for learning with respect to a fixed distribution and also in the distribution-independent setting. We remark that this reduction might make the problem (computationally) harder than the original problem. For example while, as we demonstrated, coverage functions are learnable efficiently over the uniform distribution. At the same time linear thresholds of monotone disjunctions appear to be significantly harder to learn, in particular they include monotone CNF formulas that are not known to be learnable efficiently.

For any $y \in \mathbb{R}$, let $\mathtt{thr}(y) : \mathbb{R} \rightarrow \{0,1\}$ be the function that is 1 iff $y \geq 0$. For any class $\mathcal{C}$ of functions mapping $\{-1,1\}^n$ into $[0,1]$, let $\mathcal{C}_{\geq}$ denote the class of Boolean functions $\{\mathtt{thr}(c - \theta) \mid c \in \mathcal{C}, \theta \in [0,1]\}$.

Theorem 6.2. *Let $\mathcal{C}$ be a class of functions mapping $\{-1,1\}^n$ into $[0,1]$. Let $\mathcal{D}$ be any fixed distribution on $\{-1,1\}^n$ and suppose $\mathcal{C}_{\geq}$ is agnostically learnable on $\mathcal{D}$ in time $T(n, \frac{1}{\epsilon})$. Then, $\mathcal{C}$ is agnostically learnable on $\mathcal{D}$ with ℓ_1-error ϵ in time $O(T(n, \frac{3}{\epsilon})/\epsilon)$.*

Proof. We will use the algorithm $\mathcal{A}$ that agnostically learns $\mathcal{C}_{\geq}$ to obtain an algorithm $\mathcal{A}'$ that agnostically learns $\mathcal{C}$. Let $\mathcal{P}$ be a distribution on $\{-1,1\}^n \rightarrow [0,1]$ whose marginal distribution on $\{-1,1\}^n$ is $\mathcal{D}$. Let $c^* \in$

$\mathcal{C}$ be the function that achieves the optimum error, that is $\mathbf{E}_{(x,y)\sim\mathcal{P}}[|c^*(x)-y|] = \min_{f\in\mathcal{C}} \mathbf{E}_{(x,y)\sim\mathcal{P}}[|f(x)-y|]$. For any $\theta \in [0,1]$, let $\mathcal{P}_\theta$ denote the distribution on $\{-1,1\}^n \times \{0,1\}$ obtained by taking a random sample (x,y) from $\mathcal{P}$ and outputting the sample $(x, \mathtt{thr}(y-\theta))$.

Our algorithm learns $\mathcal{C}$ as follows.

1. For each $1 \le i \le \lfloor \frac{1}{\epsilon} \rfloor = t$ and $\theta = i\cdot\epsilon$ simulate random examples from $\mathcal{P}_\theta$ and use $\mathcal{A}$ with an accuracy of ϵ to learn a hypothesis h_i. Notice that the marginal distribution of $\mathcal{P}_\theta$ on $\{-1,1\}^n$ is $\mathcal{D}$.

2. Return $h = \epsilon \cdot \sum_{i\in[t]} h_i$ as the final hypothesis.

To see why h is a good hypothesis for $\mathcal{P}$, first observe that for any $y \in [0,1]$,

$$0 \le y - \epsilon \cdot \sum_{i\in[t]} \mathtt{thr}(y - i\cdot\epsilon) \le \epsilon. \tag{3}$$

Let $c_i^* : \{-1,1\}^n \to \{0,1\}$ be defined by $c_i^*(x) = \mathtt{thr}(c^*(x) - i\cdot\epsilon)$ for every $i \in [t]$. Thus, $0 \le c^*(x) - \epsilon\cdot\sum_{i\in[t]} c_i^*(x) \le \epsilon$ for every $x \in \{-1,1\}^n$. Now, since $\mathcal{A}$ returns a hypothesis with error of at most ϵ higher than the optimum for $\mathcal{C}_\ge$ and $c_i^* \in \mathcal{C}_\ge$ for every i we have:

$$\begin{aligned}\mathop{\mathbf{E}}_{(x,y)\sim\mathcal{P}}[|\mathtt{thr}(y-i\cdot\epsilon) - h_i(x)|] &= \mathop{\mathbf{Pr}}_{(x,\ell)\sim\mathcal{P}_{i\cdot\epsilon}}[\ell \ne h_i(x)] \le \mathop{\mathbf{Pr}}_{(x,\ell)\sim\mathcal{P}_{i\cdot\epsilon}}[\ell \ne c_i^*(x)] + \epsilon \\ &= \mathop{\mathbf{Pr}}_{(x,y)\sim\mathcal{P}}[\mathtt{thr}(y-i\cdot\epsilon) \ne c_i^*(x)] \end{aligned} \tag{4}$$

for every $i \in [t]$. Now, for any fixed y, the number of $i \in [t]$ for which $\mathtt{thr}(y - i\cdot\epsilon) \ne c_i^*(x)$ is at most $\lceil \frac{|y-c^*(x)|}{\epsilon} \rceil$. Thus,

$$\sum_{i\in[t]} \mathop{\mathbf{Pr}}_{(x,y)\sim\mathcal{P}}[\mathtt{thr}(y-i\cdot\epsilon) \ne c_i^*(x)] \le \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{P}}\left[\left\lceil \frac{|y-c^*(x)|}{\epsilon} \right\rceil\right] \le \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{P}}\left[\frac{|y-c^*(x)|}{\epsilon}\right] + 1. \tag{5}$$

Now, by equations (3), then (4), and (5).

$$\begin{aligned}\mathop{\mathbf{E}}_{(x,y)\sim\mathcal{P}}[|y - \epsilon\cdot\sum_{i\in[t]} h_i(x)|] &\le \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{P}}\left[\left|y - \epsilon\sum_{i\in[t]}\mathtt{thr}(y-i\cdot\epsilon)\right|\right] + \epsilon \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{P}}\left[\left|\sum_{i\in[t]}\mathtt{thr}(y-i\cdot\epsilon) - \sum_{i\in[t]} h_i(x)\right|\right] \\ &\le^{(3)} \epsilon + \epsilon\sum_{i\in[t]} \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{P}}[|\mathtt{thr}(y-i\cdot\epsilon) - h_i(x)|] \\ &\le^{(4)} \epsilon + \epsilon\sum_{i\in[t]}\left(\mathop{\mathbf{Pr}}_{(x,y)\sim\mathcal{P}}[\mathtt{thr}(y-i\cdot\epsilon) \ne c_i^*(x)] + \epsilon\right) \\ &\le^{(5)} \epsilon + \epsilon\left(\mathop{\mathbf{E}}_{(x,y)\sim\mathcal{P}}\left[\frac{|y-c^*(x)|}{\epsilon}\right] + 1 + t\cdot\epsilon\right) \\ &\le \mathop{\mathbf{E}}_{(x,y)\sim\mathcal{P}}[|y-c^*(x)|] + 3\epsilon\end{aligned}$$

This establishes the correctness of our algorithm when used with $\epsilon/3$ instead of ϵ. Notice that the running time of the algorithm is dominated by $\lfloor \frac{3}{\epsilon} \rfloor$ runs of $\mathcal{A}$ and thus is at most $O(\frac{1}{\epsilon}\cdot T(n, \frac{3}{\epsilon}))$. This completes the proof. □

The same reduction clearly also works in the PAC setting (where $\mathbf{E}_{(x,y)\sim\mathcal{P}}[|y - c^*(x)|] = 0$). We state it below for completeness.

Lemma 6.3. *Let $\mathcal{C}$ be a class of functions mapping $\{-1,1\}^n$ into $[0,1]$, let $\mathcal{D}$ be any fixed distribution on $\{-1,1\}^n$ and suppose that $\mathcal{C}_\geq$ is PAC learnable on $\mathcal{D}$ in time $T(n, \frac{1}{\epsilon})$. Then, $\mathcal{C}$ is PAC learnable on $\mathcal{D}$ with ℓ_1-error ϵ in time $O(T(n, \frac{3}{\epsilon})/\epsilon)$.*

Using these results and Lem. 2.1, we can now relate the complexity of learning coverage functions with ℓ_1-error on any fixed distribution to the complexity of PAC learning of the class the class of thresholds of non-negative sums of monotone disjunctions on the same distribution.

Corollary 6.4. *Let $\mathcal{CV}(s)$ denote the set of all coverage functions in $\mathcal{CV}$ of size at most s. Suppose there exists an algorithm that PAC learns the class $\mathcal{CV}(s)_\geq$ in time $T(n, s, \frac{1}{\epsilon})$ over a distribution $\mathcal{D}$. Then, there exists an algorithm that PAC learns $\mathcal{CV}(s)$ with ℓ_1-error in time $O(T(n, s, \frac{3}{\epsilon})/\epsilon)$ over $\mathcal{D}$.*

7 Conclusions

In this work we described algorithms that provably learn coverage functions efficiently in PAC and PMAC learning models when the distribution is restricted to be uniform. While the uniform distribution assumption is a subject of intensive research in computational learning theory, it is unlikely to ever hold in practical applications. That said, our algorithms make sense and can be used even when the distribution of examples is not uniform (or product) possibly with some tweaks to the parameters. In fact, our algorithms include some of the standard ingredients used in practical machine learning such identification of relevant variables and polynomial regression. Therefore it would be interesting to evaluate the algorithms on real-world data.

Our work also leaves many natural questions about structure and learnability of coverage and related classes of functions (such as submodular, OXS and XOS) open. For example: (1) can coverage functions of unbounded size be PAC/PMAC learned properly and efficiently over the uniform distribution? (2) can OXS functions be PAC/PMAC learned efficiently over the uniform distribution? (3) which other natural distributions can coverage functions be learned on efficiently?

Acknowledgements

We are grateful to Jan Vondrak for helpful advice and numerous discussions about this work. We also thank Or Sheffet for bringing to our attention the problem of achieving low average error in differentially private query release for short conjunction queries.

References

D. J. Lehmann B. Lehmann and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55:1884–1899, 2006.

A. Badanidiyuru, S. Dobzinski, H. Fu, R. Kleinberg, N. Nisan, and T. Roughgarden. Sketching valuation functions. In *SODA*, pages 1025–1035, 2012.

M.F. Balcan and N. Harvey. Submodular functions: Learnability, structure, and optimization. *CoRR*, abs/1008.2159, 2012. Earlier version in proceedings of STOC 2011.

M.F. Balcan, Florin Constantin, Satoru Iwata, and Lei Wang. Learning valuation functions. *Journal of Machine Learning Research - COLT Proceedings*, 23:4.1–4.24, 2012.

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 *PODS*, pages 273–282, 2007.

E. Blais, R. O'Donnell, and K. Wimmer. Polynomial regression under arbitrary product distributions. In *COLT*, pages 193–204, 2008.

A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97(1-2):245–271, 1997.

A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the sulq framework. In *PODS*, pages 128–138, 2005.

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

M. Bun, J. Ullman, and S. P. Vadhan. Fingerprinting codes and the price of approximate differential privacy. *CoRR*, abs/1311.3158, 2013.

D. Chakrabarty and Z. Huang. Testing coverage functions. In *ICALP (1)*, pages 170–181, 2012.

K. Chandrasekaran, J. Thaler, J. Ullman, and A. Wan. Faster private release of marginals on small databases. *ITCS*, 2014.

M. Cheraghchi, A. Klivans, P. Kothari, and H. Lee. Submodular functions are noise stable. In *SODA*, pages 1586–1592, 2012.

G. Cornuejols, M. Fisher, and G. Nemhauser. Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms. *Management Science*, 23(8):789–810, 1977.

S. Dobzinski and M. Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *SODA*, pages 1064–1073, 2006.

S. Dughmi and J. Vondrák. Limitations of randomized mechanisms for combinatorial auctions. In *FOCS*, pages 502–511, 2011.

C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

C. Dwork, M. Naor, O. Reingold, G. Rothblum, and S. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, pages 381–390, 2009.

C. Dwork, A. Nikolov, and K. Talwar. Efficient algorithms for privately releasing marginals via convex relaxations. *CoRR*, abs/1308.1385, 2013.

J. Edmonds. Matroids, submodular functions and certain polyhedra. *Combinatorial Structures and Their Applications*, pages 69–87, 1970.

U. Feige. A threshold of $\ln n$ for approximating set cover. *JACM*, 45(4):634–652, 1998.

U. Feige. On maximizing welfare when utility functions are subadditive. In *ACM STOC*, pages 41–50, 2006.

V. Feldman. A complete characterization of statistical query learning with applications to evolvability. *Journal of Computer System Sciences*, 78(5):1444–1459, 2012.

V. Feldman and P. Kothari. Agnostic learning of disjunctions on symmetric distributions. *arXiv, CoRR*, abs/1405.6791, 2014.

V. Feldman and J. Vondrák. Optimal bounds on approximation of submodular and xos functions by juntas. In *FOCS*, pages 227–236, 2013.

V. Feldman, P. Kothari, and J. Vondrák. Representation, approximation and learning of submodular functions using low-rank decision trees. In *COLT*, pages 30:711–740, 2013.

L. Fleischer, S. Fujishige, and S. Iwata. A combinatorial, strongly polynomial-time algorithm for minimizing submodular functions. *JACM*, 48(4):761–777, 2001.

A. Frank. Matroids and submodular functions. *Annotated Biblographies in Combinatorial Optimization*, pages 65–80, 1997.

M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

C. Guestrin, A. Krause, and A. Singh. Near-optimal sensor placements in gaussian processes. In *ICML*, pages 265–272, 2005.

A. Gupta, M. Hardt, A. Roth, and J. Ullman. Privately releasing conjunctions and the statistical query barrier. In *STOC*. ACM, 2011.

M. Hardt, G. Rothblum, and R. Servedio. Private data release via learning thresholds. In *SODA*, pages 168–187, 2012.

D. Haussler. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation*, 100(1):78–150, 1992. ISSN 0890-5401.

R. K. Iyer and J. A. Bilmes. Submodular optimization with submodular cover and submodular knapsack constraints. In *NIPS*, pages 2436–2444, 2013.

J. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55:414–440, 1997.

A. Kalai, A. Klivans, Y. Mansour, and R. Servedio. Agnostically learning halfspaces. *SIAM J. Comput.*, 37 (6):1777–1805, 2008.

M. Kearns, M. Li, and L. Valiant. Learning boolean formulas. *J. ACM*, 41(6):1298–1328, November 1994a.

M. Kearns, R. Schapire, and L. Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3): 115–141, 1994b.

A. Klivans and R. Servedio. Learning dnf in time $2^{\tilde{O}(n^{1/3})}$. *J. Comput. Syst. Sci.*, 68(2):303–318, 2004.

A. Krause and C. Guestrin. Submodularity and its applications in optimized information gathering. *ACM TIST*, 2(4):32, 2011.

A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: maximizing information while minimizing communication cost. In *IPSN*, pages 2–10, 2006.

E. Kushilevitz and Y. Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, 1993.

L. Lovász. Submodular functions and convexity. *Mathematical Programmming: The State of the Art*, pages 235–257, 1983.

S. Raskhodnikova and G. Yaroslavtsev. Learning pseudo-boolean k-dnf and submodular functions. *SODA*, 2012.

J. Thaler, J. Ullman, and S. Vadhan. Faster algorithms for privately releasing marginals. In *ICALP (1)*, pages 810–821, 2012.

J. Ullman and S. Vadhan. Pcps and the hardness of generating private synthetic data. In *TCC*, pages 400–416, 2011.

L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

V. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, New York, 1998.

J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, pages 67–74, 2008.

Jan Vondrák. A note on concentration of submodular functions. *CoRR*, abs/1005.2791, 2010.

Wikipedia. Least absolute deviations, 2010. URL http://en.wikipedia.org/wiki/Least_absolute_deviations.

L. Yang, A. Blum, and J. Carbonell. Learnability of DNF with representation-specific queries. *ITCS*, 2013.

A Omitted Proofs

Proof of Lemma 2.1. Suppose $c : \{-1,1\}^n \to \mathbb{R}^+$ is a coverage function. Then, there exist a universe U and sets $A_1, A_2, \ldots, A_n$ with an associated weight function $w : U \to \mathbb{R}^+$, such that for any $x \in \{-1,1\}^n$, $c(x) = \sum_{u \in \cup_{j:x_j=-1} A_j} w(u)$. For any $u \in U$, let $S_u = \{j \in [n] \mid u \in A_j\}$. Then, $c(x) = \sum_{u \in U} w(u) \cdot \mathsf{OR}_{S_u}(x)$ since $\mathsf{OR}_{S_u}(x) = 1$ if and only if $x_j = -1$ for some $j \in S$. This yields a representation of c as a linear combination of $|U|$ disjunctions.

For the converse, now let c be any function such that there exist non-negative coefficients α_S for every $S \subseteq [n], S \neq \emptyset$ such that $c(x) = \sum_{S \subseteq [n], S \neq \emptyset} \alpha_S \cdot \mathsf{OR}_S(x)$. We will now construct a universe U and sets $A_1, A_2, \ldots, A_n$ with an associated weight function $w : U \to [0,1]$ and show that $f(x) = \sum_{u \in \cup_{i:x_i=-1} A_i} w(u)$. For every non zero α_S add a new element u_S to each A_i such that $i \in S$ and let $w(u_S) = \alpha_S$. Let $U = \cup_{1 \leq i \leq n} A_i$. Now, by our construction, $c(x) = \sum_{u_S : \alpha_S \neq 0} \alpha_S \mathsf{OR}_S(x) = \sum_{u \in \cup_{i:x_i=-1} A_i} w(u)$ as required. □

Proof of Lem. 2.3. First, observe that:

$$\begin{aligned}\sum_{T\notin\mathbb{T}} \hat{f}(T)^2 &\le \max_{T\notin\mathbb{T}} |\hat{f}(T)| \cdot \sum_{T\in\mathbb{T}} |\hat{f}(T)| \\ &\le \frac{\epsilon}{2L} \sum_{T\subseteq[n]} |\hat{f}(T)| = \frac{\epsilon}{2L}\|\hat{f}\|_1 \le \epsilon/2.\end{aligned}$$

Thus f is $\epsilon/2$-concentrated on $\mathbb{T}$. Further, $|\mathbb{T}| \le \frac{2\|f\|_1}{\epsilon}$ since for each $T \in \mathbb{T}$, $|\hat{f}(T)| \ge \frac{\epsilon}{2\|f\|_1}$.

By Plancherel's theorem, we have:

$$\mathbf{E}[(f(x) - \sum_{S\in\mathbb{S}} \tilde{f}(S)\cdot\chi_S(x))^2] = \sum_{S\in\mathbb{S}}[(\hat{f}(S) - \tilde{f}(S))^2] + \sum_{S\notin\mathbb{S}} \hat{f}(S)^2 \,.$$

The first term,

$$\sum_{S\in\mathbb{S}}[(\hat{f}(S) - \tilde{f}(S))^2] \le \max_{S\in\mathbb{S}}\{|\hat{f}(S) - \tilde{f}(S)|\} \cdot \sum_{S\in\mathbb{S}}\{|\hat{f}(S) - \tilde{f}(S)|\} \le \frac{\epsilon}{6L} \cdot \sum_{S\in\mathbb{S}} |\hat{f}(S) - \tilde{f}(S)|. \quad (6)$$

For each $S \in \mathbb{T}$, $|\tilde{f}(S)| \ge \frac{\epsilon}{3L}$ and $|\hat{f}(S) - \tilde{f}(S)| \le \frac{\epsilon}{6L}$. Thus $|\hat{f}(S)| \ge \frac{\epsilon}{6L} \ge |\hat{f}(S) - \tilde{f}(S)|$ and $\sum_{S\in\mathbb{S}} |\hat{f}(S) - \tilde{f}(S)| \le \|\hat{f}\|_1 = L$. Using equation (6), this gives $\sum_{S\in\mathbb{S}}[(\hat{f}(S) - \tilde{f}(S))^2] \le \epsilon/6$.

For the second term, $\sum_{S\notin\mathbb{S}} \hat{f}(S)^2 \le \epsilon/2$ as $\mathbb{S} \supset \mathbb{T}$ and f is $\epsilon/2$-concentrated on $\mathbb{T}$. Thus, $\mathbf{E}[(f(x) - \sum_{S\in\mathbb{S}} \tilde{f}(S)\cdot\chi_S(x))^2] \le \epsilon$.

Finally, by Jensen's inequality:

$$\mathbf{E}[|f(x) - \sum_{S\in\mathbb{S}} \tilde{f}(S)\cdot\chi_S(x)|] \le \sqrt{\mathbf{E}[(f(x) - \sum_{S\in\mathbb{S}} \tilde{f}(S)\cdot\chi_S(x))^2]} \le \sqrt{\epsilon}.$$

□

Proof of Lem. 4.2. Using Lemma 2.1, let $c = \sum_{S\subseteq[n]} \alpha_S \cdot \mathsf{OR}_S$ for some non-negative α_S, $S \subseteq [n]$ such that $\sum_{S\subseteq[n]} \alpha_S \le 1$.

Set $\beta_j = \sum_{S\subseteq[n]} \alpha_S \cdot \beta_j^S$ for every $1 \le j \le m$. We verify that $f = \sum_{j=1}^m \beta_j \cdot \phi_j$ has the required approximation guarantee.

$$\begin{aligned}\mathop{\mathbf{E}}_{x\sim\mathcal{D}}[|c(x) - f(x)|] &= \mathop{\mathbf{E}}_{x\sim\mathcal{D}}\left[\left|\sum_{S\subseteq[n]} \alpha_S \cdot (\mathsf{OR}_S - \sum_{j=1}^m \beta_j \cdot \phi_j(x))\right|\right] \\ &\le \sum_{S\subseteq[n]} \alpha_S \cdot \mathop{\mathbf{E}}_{x\sim\mathcal{D}}\left[\left|\mathsf{OR}_S(x) - \sum_{j=1}^m \beta_j^S \cdot \phi_j(x)\right|\right] \\ &\le \sum_{S\subseteq[n]} \alpha_S \cdot \epsilon \le \epsilon.\end{aligned}$$

□