JMLR: Workshop and Conference Proceedings vol 30 (2013) 1–34

# Learning using Local Membership Queries

**Pranjal Awasthi**[*] PAWASTHI@CS.CMU.EDU
*Carnegie Mellon University*

**Vitaly Feldman** VITALY@POST.HARVARD.EDU
*IBM Almaden Research Center*

**Varun Kanade**[†] VKANADE@EECS.BERKELEY.EDU
*University of California, Berkeley*

## Abstract

We introduce a new model of membership query (MQ) learning, where the learning algorithm is restricted to query points that are *close* to random examples drawn from the underlying distribution. The learning model is intermediate between the PAC model (Valiant, 1984) and the PAC+MQ model (where the queries are allowed to be arbitrary points).

Membership query algorithms are not popular among machine learning practitioners. Apart from the obvious difficulty of adaptively querying labelers, it has also been observed that querying *unnatural* points leads to increased noise from human labelers (Lang and Baum, 1992). This motivates our study of learning algorithms that make queries that are close to examples generated from the data distribution.

We restrict our attention to functions defined on the $n$-dimensional Boolean hypercube and say that a membership query is local if its Hamming distance from some example in the (random) training data is at most $O(\log(n))$. We show the following results in this model:

(i) The class of sparse polynomials (with coefficients in $\mathbb{R}$) over $\{0,1\}^n$ is polynomial time learnable under a large class of *log-Lipschitz* distributions using $O(\log(n))$-local queries. This class also includes the class of $O(\log(n))$-depth decision trees.

(ii) The class of polynomial-sized decision trees is polynomial time learnable under product distributions using $O(\log(n))$-local queries.

(iii) The class of polynomial size DNF formulas is learnable under the uniform distribution using $O(\log(n))$-local queries in time $n^{O(\log(\log(n)))}$.

(iv) In addition we prove a number of results relating the proposed model to the traditional PAC model and the PAC+MQ model.

**Keywords:** PAC learning, membership queries, decision trees, DNF

[†] The author is supported in part by the National Science Foundation under grants CCF-1116892 and IIS-1065251. Part of this work was done while the author was visiting Microsoft Research, New England.

[†] The author is supported by a Simons Postdoctoral Fellowship. Part of this work was performed while the author was at Harvard University supported by grant NSF-CCF-09-64401.

## 1. Introduction

Valiant's Probably Approximately Correct (PAC) model (Valiant, 1984) has been used widely to study computational complexity of learning. In the PAC model, the goal is to design algorithms which can "learn" an unknown target function, $f$, from a concept class, $C$ (for example, $C$ may be polynomial-size decision trees or linear separators), where $f$ is a boolean function over some instance space, $X$ (typically $X = \{-1, 1\}^n$ or $X \subseteq \mathbb{R}^n$). The learning algorithm has access to random *labeled* examples, $(x, f(x))$, through an oracle, $\mathsf{EX}(f, D)$, where $f$ is the unknown target concept and $D$ is the target distribution. The goal of the learning algorithm is to output a hypothesis, $h$, with low error with respect to the target concept, $f$, under distribution, $D$.

Several interesting concept classes have been shown to be *learnable* in the PAC framework (e.g. boolean conjunctions and disjunctions, $k$-CNF and $k$-DNF formulas (for constant $k$), decision lists and the class of linear separators). On the other hand, it is known that very rich concept classes such as polynomial-sized circuits are not PAC-learnable under cryptographic assumptions (Valiant, 1984; Goldreich et al., 1986). The most interesting classes for which both efficient PAC learning algorithms and cryptographic lower bounds have remained elusive are polynomial-size decision trees (even log-depth decision trees) and polynomial-size DNF formulas.

**Membership Query Model**: This learning setting is an extension of the PAC model and allows the learning algorithm to query the label of any point $x$ of its choice in the domain. These queries are called *membership queries*. With this additional power it has been shown that the classes of finite automata (Angluin, 1987), monotone DNF formulas (Angluin, 1988), polynomial-size decision trees (Bshouty, 1993), and sparse polynomials over GF(2) (Schapire and Sellie, 1996) are learnable in polynomial time. In a celebrated result, Jackson (1997) showed that the class of DNF formulas is learnable in the PAC+MQ model under the uniform distribution. Jackson (1997) used Fourier analytic techniques to prove this result building upon previous work of Kushilevitz and Mansour (1993) on learning decision trees using membership queries under the uniform distribution.

**Our Model**: Despite several interesting theoretical results, the membership query model has not been received enthusiastically by machine learning practitioners. Of course, there is the obvious difficulty of getting labelers to perform their task while the learning algorithm is being executed. But another, and probably more significant, reason for this disparity is that quite often, the queries made by these algorithms are for labels of points that do not look like typical points sampled from the underlying distribution. This was observed by Lang and Baum (1992), where experiments on handwritten characters and digits revealed that the query points generated by the algorithms often had no structure and looked meaningless to the human eye. This can cause problems for the learning algorithm as it may receive noisy labels for such query points.

Motivated by the above observations, we propose a model of membership queries where the learning algorithm is restricted to query labels of points that "look" like points drawn from the distribution. In this paper, we focus our attention to the case when the instance space is the boolean cube, i.e. $X = \{-1, 1\}^n$, or $X = \{0, 1\}^n$. However, similar models could be defined in the case when $X$ is some subset of $\mathbb{R}^n$. Suppose $x$ is a *natural* example, *i.e.* one that was received as part of the training dataset (through the oracle $\mathsf{EX}(f, D)$). We restrict the learning algorithm to make queries $x'$, where $x$ and $x'$ are close in Hamming distance. More precisely, we say that a membership query $x'$ is $r$-local with respect to a point $x$, if the Hamming distance, $|x - x'|_H$, is at most $r$.

One can imagine settings where these queries could be realistic, yet powerful. Suppose you want to learn a hypothesis that predicts a particular medical diagnosis using patient records. It could be helpful if the learning algorithm could generate a new medical record and query its label. However, if the learning algorithm is entirely unconstrained, it might come up with a record that looks gibberish to any doctor. On the other hand, if the query chosen by the learning algorithm is obtained by changing an existing record in a few locations (local query), it is more likely that a doctor may be able to make sense of such a record. In fact, this might be a powerful way for the learning algorithm to identify the most important features of the record.

It is interesting to study what power these local membership queries add to the learning setting. At the two extremes, are the PAC model (with 0-local queries), and MQ-model (with $n$-local queries). It can be easily observed that using only 1-local queries, the class of parities can be learned in polynomial time even in the presence random classification noise. This problem is known to be notoriously difficult in the PAC learning setting (Blum et al., 2003). At the same time, most PAC+MQ algorithms we are aware of, such as the algorithms for learning decision trees (Bshouty, 1993) and for learning DNF formulas (Jackson, 1997), rely crucially on using MQs in a strongly non-local way. Also, it is easy to show that in a formal sense, allowing a learner to make 1-local queries gives it strictly more power than in the PAC setting. In fact, essentially the same argument can be used to show that $r + 1$-local queries are more powerful than $r$-local queries. These separation results can be easily proved under standard cryptographic assumptions, and are presented in Appendix F.

Our results are for learning on *log-Lipschitz* distributions over the boolean cube, which we denote by $\{-1, 1\}^n$ (or sometimes by $\{0, 1\}^n$). We say that a distribution, $D$, over the boolean cube, $X = \{b_0, b_1\}^n$ is $\alpha$*-log-Lipschitz* if the logarithm of the density function is $\log(\alpha)$-Lipschitz with respect to the Hamming distance. A straightforward implication of a distribution being $\alpha$-log-Lipschitz is that for any two points $x$ and $x'$ which differ in only one bit, $D(x)/D(x') \leq \alpha$. Intuitively, this means that points that are close to each other cannot have vastly different frequencies. Frequency of a point reflects (and sometimes defines) its "naturalness" and so, in a sense, our assumption on the distribution is the same as the assumption underlying the use of local queries. The notion of log-Lipschitzness is a natural one and its variants have been studied before in different contexts Feldman and Schulman (2012); Koltun and Papadimitriou (2007). Furthermore, log-Lipschitz distributions contain a wide variety of popularly studied distributions as special cases. For example, the uniform distribution is *log-Lipschitz* with $\alpha = 1$. For constant $\alpha$, log-Lipschitz distributions have the property that changing $O(\log(n))$ bits can change the weight of a point by at most a polynomial factor. Such distributions include product distributions when the mean of each bit is some constant bounded away from $\pm 1$ (or $0, 1$). Convex combinations of $\alpha$-log-Lipschitz distributions are also $\alpha$-log-Lipschitz. They also include smooth distributions which have previously been studied for designing learning algorithms Kalai et al. (2009b).

**Our Results:** We give several learning algorithms for general log-Lipschitz distributions and for the special case of product/uniform distributions. Our main result for the log-

Lipschitz distributions is that sparse[1] polynomials are efficiently learnable with membership queries that are logarithmically local.

**Theorem 1** *The class of $t$-sparse polynomials (with real coefficients) over $\{0,1\}^n$ is efficiently learnable under the class of $\alpha$-log-Lipschitz distributions, for any constant $\alpha$, by a learning algorithm that only uses $O(\log(n) + \log(t))$-local membership queries.*

An important subclass of sparse polynomials is $O(\log n)$-depth decision trees. For this subclass we also give a conceptually simpler analysis of our algorithm (Appendix D.1). Richer concept classes are also included in the class of sparse polynomials. This includes the class of disjoint $\log(n)$-DNF expressions and log-depth decision trees, where each node is a monomial (rather than a variable). A special case of such decision trees is $O(\log(n))$-term DNF expressions.

When the polynomials represent boolean functions this algorithm can easily be made to work in the presence of *persistent* random classification noise, as described in Appendix D.4.

For the special case of constant bounded product distributions we show that polynomial-size decision trees are efficiently learnable.

**Theorem 2** *Let $\mathcal{P}$ be the class of product distributions over $X = \{-1,1\}^n$, such that the mean of each bit is bounded away from $-1$ and $1$ by a constant. Then, the class of polynomial-size decision trees is learnable with respect to the class of distributions $\mathcal{P}$, by an algorithm that uses only $O(\log(n))$-local membership queries.*

We also consider polynomial size DNF which are known to be learnable in PAC+MQ.

**Theorem 3** *The class of polynomial sized DNF formulas is learnable under the uniform distribution using $O(\log(n))$-local queries in time $n^{O(\log\log n)}$.*

**Techniques**: All our results are based on learning polynomials. It is well known that log-depth decision trees can be expressed as sparse polynomials of degree $O(\log(n))$.

Our results on learning sparse polynomials (Section 3) under log-Lipschitz distributions rely on being able to identify all the important monomials (those with non-zero coefficient) of low-degree, using $O(\log(n))$-local queries. We identify a *set* of monomials, the size of which is bounded by a polynomial in the required parameters, which includes all the important monomials. The crucial idea is that using $O(\log(n))$-local queries, we can identify given a subset of variables $S \subseteq [n]$, whether the function on the remaining variables (those in $[n] \setminus S$), is zero or not. We use the fact that the distribution is log-Lipschitz to show that performing $L_2$ regression over the set of monomials will give a good hypothesis.

For uniform (or product) distributions (Section 4; Appendices D.2, D.3), we can make use of Fourier techniques and numerous algorithms based on them. A natural approach to the problem is to try to adapt the famous algorithm of Kushilevitz and Mansour (1993) for learning decision trees (the KM algorithm) to the use of local MQs. The KM algorithm relies on a procedure that isolates all Fourier coefficients that share a common prefix and computes the sum of their squares. Isolation of coefficients that share a prefix of length $k$ requires $k$-local MQs and therefore we cannot use the KM algorithm directly. Instead we isolate

[1] Sparsity refers to the number of non-zero coefficients.

Fourier coefficients that contain a certain set of variables and we grow these sets variable-by-variable as long as the sum of squares of coefficients in them is large enough. Using $k$-local MQs it is possible to grow sets up to size $k$. More importantly, the use of prefixes in the KM algorithm ensures that Fourier coefficients are split into disjoint collections and therefore not too many collections will be relevant. In our case the collections of coefficients are not disjoint and so to prove that our algorithm will not take superpolynomial time we rely on strong concentration properties of the Fourier transform of decision trees.

For the case of DNF formulas, we use the result of Feldman (2012); Kalai et al. (2009b) which shows that one can learn a DNF formula given its heavy logarithmic-degree Fourier coefficients. To recover those coefficients we use the same algorithm as in the decision tree case. However in this case a more delicate analysis is required to obtain even the $n^{O(\log \log n)}$ running time bound we give in Theorem 3. We rely on a concentration bound by Mansour (1992) that shows that the total weight of Fourier coefficients of degree $d$ decays exponentially as $d$ grows. We remark that Mansour also gives a PAC+MQ algorithm for learning DNF running in time $n^{O(\log \log n)}$ but aside from the use of the concentration bound our algorithm and analysis are different (the dependence on the error $\epsilon$ in our algorithm is also substantially better).

All known efficient algorithms for learning DNF under the uniform distribution rely on agnostic learning of parities using the KM algorithm (or a related algorithm of Levin (1993)) (Blum et al., 1994; Jackson, 1997). In the agnostic case one cannot rely on the concentration properties crucial for our analysis and therefore it is unclear whether poly-size DNF formulas can be learned efficiently from logarithmically-local MQs. As some evidence of the hardness of this problem, in Section 5 we show that for a constant $k$, $k$-local queries do not help in agnostic learning under the uniform distribution.

One point to note is that under $\alpha$-log-Lipschitz distributions for a constant $\alpha$, the main difficulty is designing algorithms which are faster than time $n^{O(\log(n))}$. Designing an $n^{O(\log(n))}$ time algorithm is trivial for decision trees and DNF formulas. In fact, one does not even require local-membership queries to do this. This follows from the observation that *agnostic* learning of $O(\log(n))$-size parities is easy in $n^{O(\log n)}$ time.

**Related work**: Models that address the problems that arise when membership queries are answered by humans have been studied before. The work of Blum et al. (1998) proposed a noise model wherein membership queries made on points lying in the low probability region of the distribution are unreliable. For this model the authors design algorithms for learning an intersection of two halfspaces in $\mathbb{R}^n$ and also for learning a very special subclass of monotone DNF formulas. Our result on learning sparse polynomials can be compared with that of Schapire and Sellie (1996), who provided an algorithm to learn sparse polynomials over GF(2) under arbitrary distributions in Angluin's *exact* learning model. However, their algorithm is required to make membership queries that are not local. Bshouty (1993) gave an algorithm for learning decision trees using membership queries. In both these cases, it seems unlikely that the algorithms can be modified to use only local membership queries, even for the class of locally smooth distributions.

There has been considerable work investigating learnability beyond the PAC framework. We consider our results in this body of work. Many of these models are motivated by theoretical as well as real-world interest. On the one hand, it is interesting to study the minimum extra power one needs to add to the PAC setting, to make the class of polynomial-size decision trees or DNF formulas efficiently learnable. The work of Aldous and Vazirani (1990) studies models of learning where the examples are generated according to a Markov process. An interesting special case of such models is when examples are generated by a random walk on $\{-1,1\}^n$. For this model Bshouty et al. (2005) give an algorithm for learning DNF formulas (see also (Jackson and Wimmer, 2009) for more recent developments). One could simulate random walks of length up to $O(\log(n))$ using local membership queries, but adapting their DNF learning algorithm to our model runs into the same issues as adapting the KM algorithm. The work of Kalai et al. (2009b) provided polynomial time algorithms for learning decision trees and DNF formulas in a framework where the learner gets to see examples from a *smoothed* distribution.[2] Their model was inspired by the celebrated smoothed analysis framework Spielman and Teng (2004). On the other hand, other models have been proposed to capture plausible settings when the learner may indeed have more power than in the PAC-setting. These situations arise for example in scientific studies where the learner may have more than just *black-box* access to the function. Two recent examples in this line of work are the learning using injection queries of Angluin et al. (2006), and learning using restriction access of Dvir et al. (2012).

**Organization**: Section 2 introduces notation, preliminaries and also formal definitions of the model we introduce in this paper. Section 3 presents the result on learning sparse multi-linear polynomials. Due to space limitations, Appendix D contains our results on learning decision trees, and also the implementation of these algorithms in the presence of random classification noise. Section 4 presents our algorithm for learning DNFs under the uniform distribution. Section 5 contains the result showing that agnostic learning in the PAC setting and with constant local queries is equivalent. Appendix F shows that the model we introduce is strictly more powerful than the PAC setting, and strictly weaker than the MQ setting. Finally, Section 6 discusses directions for future work.

## 2. Notation and Preliminaries

**Notation**: Let $X$ be an instance space. In this paper, $X$ is the boolean hypercube. In Section 4 and Appendix D, we will use $X = \{-1,1\}^n$, as we apply Fourier techniques. In Section 3, we will use $X = \{0,1\}^n$ (the class of sparse polynomials over $\{0,1\}^n$ is different from sparse polynomials over $\{-1,1\}^n$). A concept class, $C$, is a set of functions over $X \to Y$ (where $Y = \{-1,1\}$ or $Y = \mathbb{R}$). For a distribution, $D$, over $X$ and any hypothesis, $h : X \to \{-1,1\}$, we define, $\mathrm{err}_D(h,f) = \Pr_{x\sim D}[h(x) \neq f(x)]$. If $h : X \to \mathbb{R}$, we use squared loss as the error measure, *i.e.* $\mathbb{E}_{x\sim D}[(f(x) - h(x))^2]$.

For some bit vector $x$ (where bits may be $\{0,1\}$ or $\{-1,1\}$), and any subset $S \subseteq [n]$, $x_S$ denotes the bits of $x$ corresponding to the variables, $i \in S$. The set $\bar{S}$ denotes the set $[n] \setminus S$. For two disjoint sets, $S, T$, $x_S x_T$ denote the variables corresponding to the set $S \cup T$. In particular, $x_S x_{\bar{S}} = x$.

If $D$ is a distribution over $X$, for a subset $S$, $D_S$ denotes the marginal distribution over variables in the set $S$. Let $b_S$ denote a function, $b_S : S \to \{b_0, b_1\}$, (where $\{b_0, b_1\} = \{0,1\}$ or $\{b_0, b_1\} = \{-1,1\}$). Then, $x_S = b_S$, denotes that for each $i \in S, x_i = b_S(i)$, thus the

[2] The notion of *smoothness* in the work of Kalai et al. is not related to our notion of log-Lipschitzness. They consider product distributions where each bit has mean that is chosen randomly from a range bounded away from ±1 by a constant.

variables in the set $S$ are set to the values defined by the function $b_S$. Let $\pi : X \to \{0, 1\}$ denote some property (e.g. $\pi(x) = 1$, if $x_S = b_S$ and $\pi(x) = 0$ otherwise). The distribution $(D|\pi)$, denotes the conditional distribution, given that $\pi(x) = 1$, *i.e.* the property holds.

For the unfamiliar reader, Appendix A provides standard definitions of the PAC (Valiant, 1984) model and the PAC+MQ model.

**Local Membership Queries**: For any point $x$, we say that a query $x'$ is $r$-local with respect to $x$ if the Hamming distance, $|x - x'|_H$ is at most $r$. In our model, we only allow algorithms to make queries that are $r$-local with respect to some example that it received by querying $\mathsf{EX}(f, D)$, an oracle that returns a random example from $D$ labeled according to $f$. We think of examples coming through $\mathsf{EX}(f, D)$ as *natural* examples. Thus, the learning algorithm draws a set of natural examples from $\mathsf{EX}(f, D)$ and then makes queries that are close to some example from this set. The queries are made to the membership oracle, $\mathsf{MQ}(f)$, which on receiving a query $x$, returns $f(x)$. Formally, we define learning using $r$-local membership queries as follows:

**Definition 4 (PAC+$r$-local MQ Learning)** *Let $X$ be the instance space, $C$ a concept class over $X$, and $\mathcal{D}$ a class of distributions over $X$. We say that $C$ is PAC-learnable using $r$-local membership queries with respect to distribution class, $\mathcal{D}$, if there exist a learning algorithm, $\mathcal{L}$, such that for every $\epsilon > 0$, $\delta > 0$, for every distribution $D \in \mathcal{D}$ and every target concept $f \in C$, the following hold:*

1. *$\mathcal{L}$ draws a sample, $\mathcal{S}$, of size $m = \mathrm{poly}(n, 1/\delta, 1/\epsilon)$ using example oracle, $\mathsf{EX}(f, D)$*

2. *Each query, $x'$, made by $\mathcal{L}$ to the membership query oracle, $\mathsf{MQ}(f)$, is $r$-local with respect to some example, $x \in \mathcal{S}$*

3. *$\mathcal{L}$ outputs a hypothesis, $h$, that satisfies with probability at least $1 - \delta$, $\mathrm{err}_D(h, f) \leq \epsilon$*

4. *The running time of $\mathcal{L}$ (hence also the number of oracle accesses) is polynomial in $n$, $1/\epsilon$, $1/\delta$ and the output hypothesis, $h$, is polynomially evaluable.*

**Log-Lipschitz Distributions**: Since we want to talk about log-Lipschitz distributions over $\{-1, 1\}^n$ and $\{0, 1\}^n$ both, we consider $X = \{b_0, b_1\}^n$ and state the properties of interest in general terms. We say that a distribution, $D$, over $X = \{b_0, b_1\}^n$ is $\alpha$-log-Lipschitz, for $\alpha \geq 1$, if for every pair $x, x' \in X$, with Hamming distance, $|x - x'|_H = 1$, it holds that $|\log(D(x)) - \log(D(x'))| \leq \log(\alpha)$.

We will repeatedly use the following useful properties of $\alpha$-log-Lipschitz distributions. The proof of these are easy and hence are omitted.

**Fact 5** *Let $D$ be an $\alpha$-log-Lipschitz distribution over $X = \{b_0, b_1\}^n$. Then the following are true:*

1. *For $b \in \{b_0, b_1\}$, $\frac{1}{1+\alpha} \leq \Pr_D[x_i = b] \leq \frac{\alpha}{1+\alpha}$.*

2. *For any subset, $S \subseteq [n]$, the marginal distribution, $D_{\bar{S}}$ is $\alpha$-log-Lipschitz.*

3. *For any subset $S \subseteq [n]$, and for any property, $\pi_S$, that depends only on variables $x_S$ (e.g. $x_S = b_S$), the marginal (with respect of $\bar{S}$) of the conditional distribution, $(D|\pi_S)_{\bar{S}}$ is $\alpha$-log-Lipschitz.*

4. *(As a corollary of the above three)* $\left(\frac{1}{1+\alpha}\right)^{|S|} \leq \Pr_D[x_S = b_S] \leq \left(\frac{\alpha}{1+\alpha}\right)^{|S|}$.

We use Fourier analytic techniques and a brief introduction is provided in Appendix A.

## 3. Learning Sparse Polynomials under Log-Lipschitz Distributions

In this section, we consider the problem of learning $t$-sparse polynomials with coefficients over $\mathbb{R}$ (or $\mathbb{Q}$), when the domain is restricted to $\{0,1\}^n$. In this case, we may as well assume that the polynomials are multi-linear. We assume that the absolute values of the coefficients are bounded by $B$, and hence the polynomials take values in $[-tB, tB]$, on the domain $\{0,1\}^n$. For a subset $S \subseteq [n]$, let $\xi_S(x) = \prod_{i \in S} x_i$, thus $\xi_S(x)$ is the monomial corresponding to the variables in the set $S$. Note that any $t$-sparse multi-linear polynomial can be represented as,

$$f(x) = \sum_S c_S \xi_S(x),$$

where $c_S \in \mathbb{R}$, $|\{S \mid c_S \neq 0\}| \leq t$, and $|c_S| \leq B$ for all $S$. Let $\mathbb{R}^n_{t,B}[X]$ denote the class of multi-linear polynomials over $n$ variables with coefficients in $\mathbb{R}$, where at most $t$ coefficients are non-zero and all coefficients have magnitude at most $B$.

We assume that we have an infinite precision computation model for reals.[3] Also, since the polynomials may take on arbitrary real values, we use squared loss as the notion of error. For a distribution, $D$ over $\{0,1\}^n$, the squared loss between polynomials, $f$ and $h$, is $\mathbb{E}_{x \sim D}[(f(x) - h(x))^2]$. Our main result is:

**Theorem 6** *The class $\mathbb{R}^n_{t,B}[X]$, is learnable with respect the class of $\alpha$-log-Lipschitz distributions over $\{0,1\}^n$, using $O(\log(n/\epsilon) + \log(t/\epsilon))$-local $\mathsf{MQ}$s and in time poly$((ntB/\epsilon)^\alpha, \log(n/\delta))$. The output hypothesis is a multi-linear polynomial, $h$, such that, with probability $(1-\delta)$, $\mathbb{E}_{x \sim D}[(h(x) - f(x))^2] \leq \epsilon$.*

Recall that for a subset, $S$, $x_S$ denotes the variables that are in $S$; and that $\bar{S}$ denotes the set $[n] \setminus S$. Let $f_S(x_{\bar{S}})$ denote the multi-linear polynomial defined only on variables in $x_{\bar{S}}$,

$$f_S(x_{\bar{S}}) = \sum_{T \subseteq \bar{S}} c_{S \cup T} \xi_T(x_{\bar{S}})$$

Here, we describe the high-level idea of the proof of Theorem 6. The details of the argument are provided in Appendix B. Algorithm 1 outputs a hypothesis that approximates the polynomial $f$.

**Truncation**: First, we show that there are low-degree polynomials that approximate the multi-linear polynomial, $f$, up to arbitrary (inverse polynomial) accuracy. These polynomials are the truncations of $f$ itself. Let $f^d$ denote the multi-linear polynomial obtained

[3] The case when we have bounded precision can be handled easily since our algorithms run in time polynomial in $B$, but is more cumbersome.

```
Algorithm: Learning $t$-Sparse Polynomials
inputs: $d$, $\theta$, oracles $\mathsf{EX}(f, D)$, (local)$\mathsf{MQ}(f)$

    1. let $\mathcal{S} = \emptyset$

    2. repeat (while some new set is added to $\mathcal{S}$)

        (a) For every $S' \in \mathcal{S}$, $|S'| \leq d - 1$ and for every $j \in [n] \setminus S'$
            i. let $S = S' \cup \{j\}$
            ii. if $\Pr_{D_{\bar{S}}}[f_S(x_{\bar{S}}) \neq 0] \geq \theta$, then $\mathcal{S} = \mathcal{S} \cup \{S\}$

    3. Perform regression to identify a polynomial $h = \sum_S h[S]\xi_S(x)$, that minimizes
       $\mathbb{E}[(f(x) - h(x))^2]$, subject to:

        (a) $h[S] = 0$ for $S \notin \mathcal{S}$.
        (b) $\sum_S |h[S]| \leq tB$

output $h(x)$
```

Figure 1: Algorithm: Learning $t$-Sparse Polynomials

from $f$ by discarding all the terms of degree at least $d+1$. Note that $f^d$ is multi-linear and $t$-sparse, and has coefficients of magnitude at most $B$. Thus,

$$f^d(x) = \sum_{\substack{S \subseteq [n] \\ |S| \leq d}} c_S \xi_S(x)$$

Now, observe that because $D$ is locally $\alpha$-smooth, the probability that $\xi_S(x) = 1$ is at most $(\alpha/(1+\alpha))^{|S|}$ (see Fact 5). Thus, the probability that at least one term of degree $\geq d+1$ in $f$ is non-zero, is at most $t(\alpha/(1+\alpha))^d$ by a union bound. Thus, $\Pr_{x \sim D}[f(x) \neq f^d(x)] \leq t(\alpha/(1+\alpha))^d$. Also, since $|f(x)| \leq tB$ and $|f^d(x)| \leq tB$, this implies that $\mathbb{E}_{x \sim D}[(f(x) - f^d(x))^2] \leq 4t^3 B^2 (\alpha/(1+\alpha))^d$. By choosing $d$ appropriately, when $\alpha$ is a constant this quantity can be made arbitrarily (inverse polynomial) small.

Step 3 of the Algorithm (see Fig. 1) identifies all the important coefficients of the polynomial, $f$. Suppose, we could guarantee that the set, $\mathcal{S}$, contains all coefficients $S$, such that $c_S \neq 0$ and $|S| \leq d$, *i.e.* all non-zero coefficients of $f^d$ are identified. This guarantees that the regression in step 4 will give a good approximation to $f$, since the error of the hypothesis obtained by regression has to be smaller than $\mathbb{E}_{x \sim D}[(f(x) - f^d(x))^2]$. (The generalization guarantees are fairly standard and are described in the long version.)

**Identifying Important Monomials**: In order to test whether or not a monomial, $S$, is important, the algorithm checks whether $\Pr_{D_{\bar{S}}}[f_S(x_{\bar{S}}) \neq 0] \geq \theta$. Here, $D_{\bar{S}}$ is the marginal distribution over the variables $x_{\bar{S}}$. We assume that this test can be performed perfectly accurately. (The analysis using samples is standard by applying appropriate Chernoff-Hoeffding bounds.)

In Lemma 12, we show that if the polynomial $f_S(x_{\bar{S}})$ has a non-zero coefficient of degree at most $d - |S|$, then the probability that $f_S(x_{\bar{S}}) \neq 0$ is at least $(1/(1+\alpha))^{d+\log(t)}$. In Lemma 13, we show that if $f_S(x_{\bar{S}})$ has no non-zero coefficient of degree less than $d' = O((d + \ln(t))\ln(1+\alpha))$, then $f_S(x_{\bar{S}}) \neq 0$ with probability at most $0.5(1/(1+\alpha))^{d+\log(t)}$. Thus, we will never add any subset $S$, unless there is some co-efficient $T$ in $f$ of size at most $d' = O((d + \ln(t))\ln(1+\alpha))$ and $S \subseteq T$. However, the number of such $T$ is at most $t$, and each such set can have at most $2^{d'}$ subsets. This bounds the total number of subsets the algorithm may add to $\mathcal{S}$, and hence, also the running time of the algorithm (to polynomial in the required parameters).

Note that sampling, $x_{\bar{S}}$ according to $D_{\bar{S}}$ is trivial, just draw random example from $\mathsf{EX}(f, D)$ and ignore the variables $x_S$. Let $U_S$ denote the uniform distribution over variables in $x_S$. Then we have,

$$f_S(x_{\bar{S}}) = \mathbb{E}_{x_S \sim U_S}[2^{|S|} \prod_{i \in S}(2x_i - 1) f(x)] \tag{1}$$

The variables in $x_{\bar{S}}$ are fixed, the expectation is only taken over the uniform distribution over variables in $x_S$. Notice that for any $i \in S$, since $x_i \in \{0, 1\}$, $\mathbb{E}_{x_S}[(2x_i - 1)] = 0$ and $\mathbb{E}_{x_S}[(2x_i - 1)x_i] = 1/2$. Thus, in the RHS of (1), if $S \not\subseteq T$, $\mathbb{E}_{x_S}[2^{|S|}\prod_{i \in S}(2x_i - 1)\xi_T(x)] = 0$, and if $S \subseteq T$, $\mathbb{E}_{x_S}[2^{|S|}\prod_{i \in S}(2x_i - 1)\xi_T(x)] = \xi_{T \setminus S}(x_{\bar{S}})$. Thus, the relation in (1) is true. Also, this means that if the example, $x$, is received by querying the oracle, $\mathsf{EX}(f, D)$, $f_S(x_{\bar{S}})$ can be obtained by making $O(|S|)$-local membership queries to the oracle $\mathsf{MQ}(f)$.

The complete details of the proof appear in Appendix B.

## 4. Learning DNF Formulas under the Uniform Distribution

In this section, we present an algorithm for learning polynomial size DNF formulas under the uniform distribution. We will frequently use the following facts about DNF formulas.

1. For every size-$s$ DNF formula $f$, there exists a size-$s$, DNF formula $g$ with terms of size $l$, such that $\|f - g\|_2^2 = \mathbb{E}[(f(x) - g(x))^2] \leq \frac{4s}{2^l}$. This follows by setting $g$ to the DNF formula obtained by dropping all terms of size greater than $l$ from $f$.
2. Let $f$ be an $l$-term DNF formula. Then $\sum_{|S|>t} \hat{f}^2(S) \leq 2^{\frac{-t}{10l}}$. This fact follows from the proof of Lemma 3.2 in (Mansour, 1992).

We will further use the following result from (Kalai et al., 2009b; Feldman, 2012):

**Theorem 7 ((Feldman, 2012; Kalai et al., 2009b))** *If $f$ is a size-$s$ DNF formula, then there exists an efficient randomized algorithm,* LearnDNF *that given access to the heavy, low-degree Fourier coefficients, i.e. the set $\{\hat{f}(S) \mid |S| \leq \log(s/\epsilon), |\hat{f}(S)| \geq (\epsilon/(4s))\}$, outputs a hypothesis, $h$, such that $\mathrm{err}_U(h, f) \leq \epsilon$.*

The algorithm to obtain all *heavy, low-degree terms* is presented in Figure 2. The output of the algorithm, $\mathsf{LearnDNF}(\mathcal{S})$ is obtained by doing the following: (i) estimate all the Fourier coefficients, $\hat{f}(S)$, for $S \in \mathcal{S}$ (ii) use the algorithm from Theorem 7 to obtain $h$. The rest of the section is devoted to the proof of the following Theorem.

```
Algorithm: Learning DNF formulas
inputs: $d$, $\theta$, oracles $\mathsf{EX}(f, U)$, (local)-$\mathsf{MQ}(f)$
  1. let $\mathcal{S} = \{\emptyset\}$
  2. for $i = 1, \ldots, d$
       (a) for every $S' \in \mathcal{S}$, $|S'| = i - 1$ and for every $j \in [n] \setminus S'$
             i. let $S = S' \cup \{j\}$
             ii. ($L_2$ Test) if $\mathbb{E}_{x \sim U}[f_S(x)^2] > \theta^2$, then $\mathcal{S} = \mathcal{S} \cup \{S\}$
output: $h = \mathsf{LearnDNF}(\mathcal{S})$
```

Figure 2: Algorithm: Learning DNF formulas under the Uniform Distribution

**Theorem 8** *The class of size-$s$ DNF formulas is learnable in time $(s/\epsilon)^{O(\log\log(s/\epsilon))}$ using $O(\log(s/\epsilon))$-local membership queries under the uniform distribution.*

When $d = log(s/\epsilon)$ and $\theta = \epsilon/4s$, we argue that every Fourier coefficient that has magnitude at least $\theta$ is included in $\mathcal{S}$ and also that $|\mathcal{S}|$ is $(s/\epsilon)^{O(\log\log(s/\epsilon))}$. The first part is immediate (see Claim 18 in App. D.2) and the second is proved in Claim 9. Note that the proof of Theorem 8 follows immediately using Claims 18, 9 and Theorem 7. The proof of Claim 9 is provided in Appendix C.

**Claim 9** *After running algorithm 2, $|\mathcal{S}| = (s/\epsilon)^{O(\log\log(s/\epsilon))}$.*

## 5. Lower Bound for Agnostic Learning

In this section, we prove that any concept class, $C$, efficiently *agnostically* learnable over the uniform distribution with (constant) $k$-local MQs is also efficiently *agnostically* learnable from random examples alone. This result can be compared with that of Feldman (2008), where it is shown that membership queries do not help for distribution-independent agnostic learning.

We remark that 1-local MQs suffice for learning parities of any size with random classification noise. At the same time when learning from random examples alone agnostic learning of parities can be reduced to learning parities with random classification noise (Feldman et al., 2006). However this reduction does not lead to an agnostic algorithm for learning parities with 1-local MQs since it is highly-nonlocal: the (noisy) label of every example is influenced by labels of points chosen randomly and uniformly from the whole hypercube.

Our reduction is based on embedding the unknown function, $f$, in a higher dimensional domain such that the original points are mapped to points that are at least at distance $2k+1$ apart (and in particular that no single point in the domain is $k$-close to more than one of the original points). A crucial property of this embedding is that, up to scaling it preserves the correlation of any function with $f$. The embedding is achieved using a linear binary error-correcting code, specifically we use the classic binary BCH code (Hocquenghem, 1959; Bose and Ray-Chaudhuri, 1960). The proof of the following theorem is provided in Appendix E.

**Theorem 10** *For any constant, $k$, if a concept class $C$ is learnable agnostically under the uniform distribution in the* PAC*+$k$-local* MQ *model, then $C$ is also agnostically learnable in the* PAC *model.*

In particular, this implies that it is highly unlikely that the class of parities (even of size $O(\log(n))$) will be efficiently agnostically learnable using $k$-local MQs. The class of parities is of particular interest, because an efficient agnostic algorithm for learning $O(\log(n))$ sized parities would yield an efficient DNF-learning algorithm.

## 6. Conclusions and Future Work

We introduced the local membership query model, with the goal of studying query algorithms that may be useful in practice. With the rise of crowdsourcing tools, it is increasingly possible to get human labelers for a variety of tasks. Thus, membership queries beyond the standard active learning paradigm could prove to be useful to increase the efficiency and accuracy of learning. In order to make use of human labelers, it is necessary to make queries that make sense to them. In some ways, our algorithms can be understood as searching for higher-dimensional (deeper) features using queries that modify the examples locally.

Our model of local membership queries is also a very natural and simple theoretical model. There are several interesting open questions: (i) can the class of $t$-leaf decision trees (without depth restriction) be learned under the class of log-Lipschitz distributions? (ii) is the class of DNF formulas learnable in polynomial time, at least under the uniform distribution? Another interesting question is whether a general purpose boosting algorithm exists that only uses $\alpha$-log-Lipschitz distributions. This looks difficult since most boosting algorithms decrease weights of points substantially[4].

It is also interesting to see whether agnostic learning of any interesting concept classes is possible in this learning model. Our results show that constant local queries are not useful for agnostic learning. However can $O(\log(n))$-local queries help in learning $O(\log(n))$-sized parities in the agnostic setting? We observe that learning the class of $O(\log(n))$-sized parities and the class of decision-trees is equivalent in the agnostic learning setting (even under locally smooth distributions), since weak and strong agnostic learning is equivalent even with respect to a fixed distribution (Kalai and Kanade, 2009; Feldman, 2010). Agnostic learning $O(\log(n))$-sized parities (even with respect to a fixed distribution) would also imply (PAC) learning DNF in our model with local membership queries (with respect to the same distribution) (Kalai et al., 2009a).

## Acknowledgments

The authors would like to thank Avrim Blum, Adam Kalai, Ryan O'Donnell, Shang Hua Teng, and Leslie Valiant for several helpful discussions. The authors would also like to thank the anonymous reviewers for helpful feedback.

[4] Note that Smooth-boosting (Servedio, 2003) does not use distributions that are log-Lipschitz.

## References

D. Aldous and U. Vazirani. A markovian extension of valiant's learning model. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, 1990.

D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

Dana Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75 (2):87–106, 1987.

Dana Angluin, James Aspnes, Jiang Chen, and Wu Yinghua. Learning a circuit by injecting values. In *STOC*, 2006.

A. Blum, A. Kalai, and H. Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *Journal of the ACM*, 50(4):506–519, 2003.

Avrim Blum, Prasad Chalasani, Sally A. Goldman, and Donna K. Slonim. Learning with unreliable boundary queries. *J. Comput. Syst. Sci.*, 56, April 1998.

Avrim Blum, Merrick Furst, Jeffrey Jackson, Michael Kearns, Yishay Mansour, and Steven Rudich. Weakly learning dnf and characterizing statistical query learning using fourier analysis. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, STOC '94, 1994.

R. C. Bose and D. K. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, 1960.

Nader H. Bshouty. Exact learning via the monotone theory (extended abstract). In *FOCS*, 1993.

Nader H. Bshouty, Elchanan Mossel, Ryan O'Donnell, and Rocco A. Servedio. Learning dnf from random walks. *J. Comput. Syst. Sci.*, 71, October 2005.

Z. Dvir, A. Rao, A. Wigderson, and A. Yehudayoff. Restriction access. In *ITCS*, 2012.

Dan Feldman and Leonard J. Schulman. Data reduction for weighted and outlier-resistant clustering. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, 2012.

V. Feldman. Learning dnf expressions from fourier spectrum. In *COLT*, 2012.

V. Feldman, P. Gopalan, S. Khot, and A. Ponnuswami. New results for learning noisy parities and halfspaces. In *FOCS*, 2006.

Vitaly Feldman. On the power of membership queries in agnostic learning. In *COLT*, 2008.

Vitaly Feldman. Distribution-specific agnostic boosting. In *ICS*, 2010.

Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33:792–807, 1986. ISSN 0004-5411. URL http://doi.acm.org/10.1145/6490.6503.

A. Hocquenghem. Codes correcteurs d'erreurs (in french). *Chiffres*, 2:147–156, 1959.

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.

Jeffrey C. Jackson and Karl Wimmer. New results for random walk learning. In *COLT*, 2009.

Adam Kalai and Varun Kanade. Potential-based agnostic boosting. In *NIPS*, 2009.

Adam Kalai, Varun Kanade, and Yishay Mansour. Reliable agnostic learning. In *COLT*, 2009a.

Adam Tauman Kalai, Alex Samorodnitsky, and Shang-Hua Teng. Learning and smoothed analysis. In *FOCS*, 2009b.

Vladlen Koltun and Christos H. Papadimitriou. Approximately dominating representatives. *Theor. Comput. Sci.*, 371(3), February 2007.

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

K. Lang and E. Baum. Query learning can work poorly when a human oracle is used. *IEEE Intl. Joint Conference on Neural Networks*, 1992.

L. Levin. Randomness and non-determinism. *Journal of Symbolic Logic*, 58(3):1102–1103, 1993.

Y. Mansour. An $o(n^{\log \log n})$ learning algorithm for dnf under the uniform distribution. In *COLT*, 1992.

Yishay Mansour. Learning boolean functions via the fourier transform. Survey, 1994.

J. Massey. Shift-register synthesis and BCH decoding. *IEEE Trans. Inform. Theory*, 15: 122–127, 1969.

Robert E. Schapire and Linda M. Sellie. Learning sparse multivariate polynomials over a field with queries and counterexamples. In *COLT*, pages 17–26, 1996.

Rocco A. Servedio. Smooth boosting and learning with malicious noise. *J. Mach. Learn. Res.*, 4:633–648, December 2003. ISSN 1532-4435. URL http://dx.doi.org/10.1162/153244304773936072.

Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. *J. ACM*, 51(3), 2004.

L. G. Valiant. A theory of the learnable. In *STOC*, 1984.

## Appendix A. Definitions

$\mathsf{PAC}$ **Learning** (Valiant, 1984): Let $\mathcal{D}$ be a class of distributions over $X$ and $D \in \mathcal{D}$ be some distribution. Let $C$ be a concept class over $X$, and $f \in C$. An example oracle, $\mathsf{EX}(f, D)$, when queried, returns $(x, f(x))$, where $x$ is drawn randomly from distribution $D$. The learning algorithm in the $\mathsf{PAC}$ model has access to an example oracle, $\mathsf{EX}(f, D)$, where $f \in C$ is the unknown target concept and $D \in \mathcal{D}$ is the target distribution. The goal of the learning algorithm is to output a hypothesis, $h$, that has low error with respect to the target concept under the target distribution, *i.e.* $\mathrm{err}_D(h, f) = \Pr_{x \sim D}[h(x) \neq f(x)] \leq \epsilon$.

**Membership Queries**: Let $f \in C$ be a concept defined over instance space $X$. Then a *membership query* is a point $x \in X$. A membership query oracle $\mathsf{MQ}(f)$, on receiving query $x \in X$, responds with value $f(x)$. In the $\mathsf{PAC}$+$\mathsf{MQ}$ model of learning, along with the example oracle $\mathsf{EX}(f, D)$, the learning algorithm also has access to a membership oracle, $\mathsf{MQ}(f)$.

**Fourier Analysis**: Here, we assume that $X = \{-1, 1\}^n$ (and not $\{0, 1\}^n$). For $S \subseteq [n]$, let $\chi_S : X \to \{-1, 1\}$ denote the parity function on bits in $S$, *i.e.* $\chi_S(x) = \prod_{i \in S} x_i$. When working with the uniform distribution, $U_n$, over $\{-1, 1\}^n$, it is well known that $\langle \chi_S \rangle_{S \subseteq [n]}$ forms an orthonormal basis (Fourier basis) for functions $f : X \to \mathbb{R}$. Hence $f$ can be represented as a degree $n$, multi-linear polynomial over the variables $\{x_i\}$,

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \chi_S(x)$$

where $\hat{f}(S) = \mathbb{E}_{x \sim U_n}[\chi_S(x) f(x)]$. Define $L_1(f) = \sum_{S \subseteq [n]} |\hat{f}(S)|$, $L_2(f) = \sum_{S \subseteq [n]} \hat{f}(S)^2$, $L_\infty(f) = \max_{S \subseteq [n]} |\hat{f}(S)|$ and $L_0(f) = |\{S \subseteq [n] \mid \hat{f}(S) \neq 0\}|$. Parseval's identity, states that $\mathbb{E}_{x \sim U_n}[f^2(x)] = L_2(f)$. For Boolean functions, *i.e.* with range $\{-1, 1\}$, Parseval's identity implies that $\sum_{S \subseteq [n]} \hat{f}(S)^2 = 1$. Other useful observations that we use frequently are: (i) $L_2(f) \leq L_1(f) \cdot L_\infty(f)$; (ii) $L_2(f) \leq L_0(f) \cdot (L_\infty(f))^2$.

**Polynomials**: In this paper, we are only concerned with multi-linear polynomials, since the domain is $\{-1, 1\}^n$ or $\{0, 1\}^n$. A multi-linear polynomial over $n$ variables can be expressed as:

$$f(x) = \sum_{S \subseteq [n]} c_S \prod_{i \in S} x_i$$

When the domain is $\{-1, 1\}^n$, the monomials correspond to the parity function, $\chi_S(x)$, and if the distribution is uniform over $\{-1, 1\}^n$, $c_S = \mathbb{E}_{x \sim U}[f(x) \chi_S(x)] = \hat{f}(S)$. When the domain is $\{0, 1\}^n$, the monomials correspond to conjunctions over the variables included in the monomial. We denote such conjunctions by, $\xi_S(x) = \prod_{i \in S} x_i$.

For any $S \subseteq [n]$, define $f_S(x) := \sum_{T \supseteq S} c_T \prod_{i \in T \setminus S} x_i$, and $f_{-S}(x) = f(x) - (\prod_{i \in S} x_i) \cdot f_S(x)$.

## Appendix B. Learning Sparse Polynomials under Log-Lipschitz Distributions

### Proof of Theorem 6

First, we have the following useful general lemma.

**Lemma 11** *Let $f$ be a $t$-sparse multi-linear polynomial defined over any field, $\mathbb{F}$, with a non-zero constant term, $c_0$. Let $D$ be any $\alpha$-log-Lipschitz distribution over $\{0,1\}^n$, then*

$$\Pr_D[f(x) \neq 0] \geq \left(\frac{1}{1+\alpha}\right)^{\log_2(t)}$$

**Proof** We prove this by induction on the number of variables, $n$. When $n = 1$, the only possible polynomials are $f(x_1) = c_0 + c_1 x_1$. Then $f(x) = 0$ if and only if $x_1 = 1$ and $c_1 = -c_0$ (since $c_0 \neq 0$). Note that when $D$ is $\alpha$-log-Lipschitz, $\Pr[x_1 = 1] \leq \alpha/(1+\alpha)$ (see Fact 5). Thus, $\Pr_D[f(x_1) \neq 0] \geq 1/(1+\alpha)$. (And the sparsity is 2, and $\log(2) = 1$.) Thus the base case is verified.

Let $f$ be any multi-linear polynomial defined over $n$ variables. Suppose there exists a variable, without loss of generality, say $x_1$, such that $c_1 x_1$ is a term in $f$, where $c_1 \neq 0$. Then we can write $f$ as follows:

$$f(x) = f_{-1}(x) + x_1 f_1(x)$$

where $f_{-1}$ and $f_1$ are both multi-linear polynomials over $n-1$ variables and both have a non-zero constant term. (The constant term of $f_{-1}$ is just $c_0$, and $f_1$ has constant term $c_1$.) Then note that $1/(1+\alpha) \leq \Pr_D[x_1 = b | x_{-1}] \leq \alpha/(1+\alpha)$, for both $b = 1$ and $b = 0$. Now, it is easy to see that $\Pr_D[f(x) \neq 0] \geq \Pr_D[x_1 = 0 | x_{-1}] \Pr_D[f_{-1}(x) \neq 0] \geq (1/(1+\alpha)) \Pr[f_{-1}(x) \neq 0]$.

To see that $\Pr_D[f(x) \neq 0] \geq (1/(1+\alpha)) \Pr_D[f_1(x) \neq 0]$ consider the following: Fix $x_{-1}$, if $\Pr[f_1(x) \neq 0]$, then for at least one setting of $x_1$, it must be the case that $f(x) \neq 0$. Thus, conditioned on $x_{-1}$, $\Pr_D[f(x) \neq 0 | x_{-1}] \geq (1/(1+\alpha))\delta(f_{-1}(x) \neq 0)$ (here $\delta(\cdot)$ is the indicator function). Thus, $\Pr_D[f(x) \neq 0] \geq (1/(1+\alpha)) \Pr_D[f_{-1}(x) \neq 0]$. However, at least one of $f_{-1}$, $f_1$ must have sparsity at most $t/2$, thus by induction we are done.

In the case, that there is no $x_i$ such that $c_i x_i$ (with $c_i \neq 0$) appears in $f$ as a term, let $f_0$ be the polynomial obtained from $f$ by setting $x_1 = 0$ and $f_1$ be the polynomial obtained from $f$ by setting $x_1 = 1$. Note that both $f_0$ and $f_1$ have constant term $c_0 \neq 0$ and sparsity at most $t$, but they have one fewer variable than $f$. Thus, $\Pr_D[f_b(x) \neq 0] \geq (1/(1+\alpha))^{\log_2(t)}$, for $b = 0, 1$. However, note that

$$\begin{aligned}\Pr_D[f(x) \neq 0] &= \Pr_D[x_1 = 0] \Pr_{D_0}[f_0(x) \neq 0] + \Pr_D[x_1 = 1] \Pr_{D_1}[f_1(x) \neq 0] \\ &\geq \left(\frac{1}{1+\alpha}\right)^{\log(t)}\end{aligned}$$

This completes the induction. ■

Using Lemma 11, we can now show that step 3 in algorithm 1 correctly identifies all the important monomials (monomials of low-degree with non-zero coefficients in $f$).

**Lemma 12** *Suppose $S \subseteq [n]$, such that $f_S(x)$ has a monomial of degree at most $d - |S|$ with non-zero coefficient. Then,*

$$\Pr_{D_{\bar{S}}}[f_S(x_{\bar{S}}) \neq 0] \geq \left(\frac{1}{1+\alpha}\right)^{d-|S|+\log(t)}$$

**Proof** Note that, since $D$ is a $\alpha$-log-Lipschitz distribution, $D_{\bar{S}}$ is also a $\alpha$-log-Lipschitz distribution (see Fact 5). Let $S'$ be a subset of $\bar{S}$, such that $\xi_{S'}(x_{\bar{S}})$ is the smallest degree monomial in $f_S(x_{\bar{S}})$ with non-zero coefficient. Then, since $D_{\bar{S}}$ is $\alpha$-log-Lipschitz, $\Pr_{D_{\bar{S}}}[\xi_{S'}(x_{\bar{S}}) = 1] \geq (1/(1+\alpha))^{|S'|} \geq (1/(1+\alpha))^{d-|S|}$.

Now, the conditional distribution $(D_{\bar{S}}|\xi_{S'}(x_{\bar{S}}) = 1)$ is not necessarily $\alpha$-log-Lipschitz, but the marginal distribution with respect to variables $(\overline{S \cup S'})$, $(D_{\bar{S}}|(\xi_{S'}(x_{\bar{S}}) = 1))_{(\overline{S \cup S'})}$, is indeed $\alpha$-log-Lipschitz (see Fact 5). Let $f_S^{S'}(x_{(\overline{S \cup S'})})$ be the polynomial obtained from $f_S$ by setting $x_i = 1$ for each $i \in S'$. Note that the constant term of $f_S^{S'}$ is non-zero and it is $t$-sparse, and is only defined on the variables in $-(S \cup S')$. Hence, by applying Lemma 11 to $f_S^{S'}$ and the marginal (w.r.t the variables $\bar{S}'$) of the conditional distribution $(D_{\bar{S}}|\xi_{S'}(x_{\bar{S}}) = 1)$, *i.e.* $(D_{\bar{S}}|\xi_{S'}(x) = 1)_{(\overline{S \cup S'})}$, we get the required result. ■

Next, we show the following simple lemma (proof is in Section B) that will allow us to conclude that step 3 of the algorithm never adds too many terms.

**Lemma 13** *If each term of $f_S$ has degree at least $d'$, then the probability that $f_S(x) \neq 0$ is at most $t(\alpha/(1+\alpha))^{d'}$.*

**Proof** Note that each monomial of $f_S$ has degree at least $d'$. Under any $\alpha$-log-Lipschitz distribution, the probability that a monomial of degree $d'$ is not-zero is at most $(\alpha/(1+\alpha))^{d'}$ (see Fact 5). Since, $D_{\bar{S}}$ is a $\alpha$-log-Lipschitz distribution, by a simple union bound we get the required result. ■

Now in order to get an $\epsilon$-approximation in terms of squared error, using the argument about truncation, it suffices to choose $d = \log(4t^3B^2/\epsilon)/\log((1+\alpha)/\alpha)$, and consider the truncation $\alpha$. For this value of $d$, if $\theta$ is set to
$1/(4t^3B^2)^{2\log(1+\alpha)/\log((1+\alpha)/\alpha)}$, using Lemma 12, we are sure that all the monomials in $f$ of degree at most $d$ that have non-zero coefficients are identified in step 3 of the algorithm. Note that $\theta$ is still inverse polynomial in $(ntB/\epsilon)^{\alpha}$.

Finally, we note that if $d'$ is set to $\log(2t/\theta)/\log((1+\alpha)/\alpha)$, then for any subset, $S$, if the monomial with the least degree in $f_S$, has degree at least $d'$, then $\Pr_{D_{\bar{S}}}[f_S(x) \neq 0] \leq \theta/2$. In particular, this means that if a set, $S$, with $|S| \leq d$, is such that the smallest monomial, $\xi_T(x)$ in $f$ for which $S \subseteq T$, is such that $|T| \geq d + d'$, then $S$ will never be added to $\mathcal{S}$ by the algorithms. The fact that this probability was $\theta/2$ (instead of exactly $\theta$), means that sampling can be used carry out the test in the algorithm to reasonable accuracy. Finally, observe that $t2^{d+d'}$ is still polynomial in $(ntB/\epsilon)^{\alpha}$. Thus, the total number of sets added in $\mathcal{S}$, can never be more than polynomially many.

**Generalization** The generalization argument is pretty standard and so we just present an outline. First, we observe that it is fine to discretize real numbers to some $\Delta$, where $\Delta$ is inverse polynomial in $(ntB/\epsilon)^{\alpha}$, without blowing up the squared loss. Now, the regression in the algorithm requires that the sum of absolute values of the coefficients of the polynomial, $h$, be at most $tB$. Thus, we can view this as distributing $tB/\Delta$ blocks over $2^n$ possible coefficients (in fact the number of coefficients is smaller). The total number of such discretized polynomials is at most $2^{\mathrm{poly}((ntB/\epsilon)^{\alpha})}$. Thus, it suffices to minimize the squared error on a (reasonably large) sample.

## Appendix C. Learning DNF Formulas under the Uniform Distribution

**Proof** [of Claim 9] For a set $S$, denote $\|f_S\|^2 = \sum_{T \supseteq S} \hat{f}^2(T)$. If $S \in \mathcal{S}$, then we know that $\|f_S\|^2 \geq \theta^2$. Thus, it follows that $|\mathcal{S}| \leq (1/\theta^2) \sum_{S:|S| \leq d} \|f_S\|^2$. Now, $\sum_{S:|S|\leq d} \|f_S\|^2 = \sum_{d'=1}^{d} \sum_{S:|S|=d'} \|f_S\|^2$. We will show that for each $d'$, $\sum_{S:|S|=d'} \|f_S\|^2 \leq ns2^{O(d' \log(d'))}$. Then,

$$\begin{aligned}
\sum_{S:|S|=d'} \|f_S\|^2 &= \sum_{S:|S|=d'} \sum_{T \supseteq S} \hat{f}^2(T) \\
&= \sum_{T:|T|\geq d'} \binom{|T|}{d'} \hat{f}^2(T) \\
&= \sum_{t=d'}^{n} \sum_{T:|T|=t} \binom{t}{d'} \hat{f}^2(T)
\end{aligned}$$

For a given $t$, consider the inner summation $\sum_{T:|T|=t} \binom{t}{d'} \hat{f}^2(T)$. We will first apply Fact 1, to say that $f$ is close to some $g_t$ which is an $l_t$-term $\mathsf{DNF}$ formula (we will define the value of $l_t$ shortly). Hence, we get that,

$$\sum_{T:|T|=t} \binom{t}{d'} \hat{f}^2(T) \leq \binom{t}{d'} \sum_{T:|T|=t} \hat{g}_t^2(T) + \frac{4s}{2^{l_t}}$$

Next we use Fact 2 to claim that $\sum_{T:|T|=t} \hat{g}_t^2(T) \leq 2^{\frac{-t}{10 l_t}}$. So we have

$$\sum_{T:|T|=t} \binom{t}{d'} \hat{f}^2(T) \leq \binom{t}{d'} \left( 2^{\frac{-t}{10 l_t}} + \frac{4s}{2^{l_t}} \right)$$

Setting $l_t = C\sqrt{t}$ and differentiating, we get that the term is maximized at $t = O(d'^2)$ and the maximum value is $s2^{O(d' \log d')}$. Since there are at most $n$ such terms we get that $\sum_{S:|S|=d'} ||f_S||^2 = ns2^{O(d' \log d')}$. Finally, we have

$$\sum_{S:|S|\leq d} ||f_S||^2 \leq \sum_{d'=1}^{d} ns2^{O(d' \log(d'))} = nds2^{O(d \log d)}.$$

■

## Appendix D. Learning Decision Trees

In this section, we present two algorithms for learning decision trees. Section D.1, shows that $O(\log(n))$-depth decision trees can be efficiently learned under locally $\alpha$-smooth distributions, for constant $\alpha$. This result is actually a special case of the result in Section 3, but we present it separately because it is somewhat simpler. In Section D.2, we show that the class of polynomial-size decision trees can be learned under the uniform distribution. This algorithm is extended to product distributions; this fact is shown in Section D.3. Section D.4 shows that these algorithms can be made to work under random classification noise.

### D.1. Learning log-Depth Trees under Locally Smooth Distributions

In this section, the domain is assumed to be $\{-1, 1\}^n$. Let $f$ be a function that can be represented as a depth $d$ decision tree with $t$ leaves (note that $t \leq 2^d$). We are mostly interested in the case when $d = O(\log(n))$, [5] since our algorithms run in time polynomial in $2^d$. By slightly abusing notation, let $f$ also denote the polynomial representing the decision tree,

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \chi_S(x)$$

Also, we stick to the notation $\hat{f}(S)$ as the co-efficient of $\chi_S(x)$ even though we will consider distributions that are not uniform over $\{-1, 1\}^n$. Thus, the coefficients cannot be interpreted as a Fourier transform. The polynomial $f$ has degree $d$. Also, it is the case that $L_0(f) \leq t2^d$, $L_1(f) \leq t$, and $L_2(f) = 1$. (These facts hold even when the distribution is not uniform and are standard. The reader is referred to (Mansour, 1994). However, when the distribution is not uniform, it is not the case that $\mathbb{E}_{x \sim D}[f(x)^2] = L_2(f)$.) More importantly, it is also the case that if $\hat{f}(S) \neq 0$, then $|\hat{f}(S)| \geq 1/2^d$. We will use this fact effectively in showing that depth-$d$ decision trees can be efficiently learned under locally $\alpha$-smooth distributions, when $d = O(\log(n))$ and $\alpha$ constant.

The algorithm (see Fig. 3) finds all the monomials that are relevant. Call a subset $S \subseteq [n]$ maximal for $f$, if $\hat{f}(S) \neq 0$ and for all $T \supsetneq S$, $\hat{f}(T) = 0$. We prove the following two crucial points:

1. If $S$ is maximal for $f$, all subsets of $S$ pass the non-zero test (Step 2.(a).ii. of the algorithm).

2. If a set $T$ is not a subset of some $S$ that is maximal for $f$, $T$ fails the non-zero test (Step 2.(a).ii).

The above points can be used to prove two facts. First, that the relevant monomials can be found by building up from the empty set (since all subsets of maximal monomials pass the test). And second, that the total number of subsets that are added into the set of important monomials (in Step 2) is at most $t2^d$. This bounds the running time of the algorithm.

**Non-Zero Test**: In step 2.(a).ii. of the algorithm (Fig. 3), we check the following, which we call as the *non-zero* test: $\Pr_{x \sim D}[f_S(x) \neq 0]$. Recall, that

$$f_S(x) = \sum_{T \supseteq S} \hat{f}(T) \chi_{T \setminus S}(x)$$

[5] When $d = O(\log(n))$, whether we consider the domain to be $\{-1, 1\}^n$ or $\{0, 1\}^n$ is unimportant, since the sparsity of polynomial is preserved (up to polynomial factors) when going from one domain to the other.

```
Algorithm: Learning log-depth decision trees
Input: $d$ (depth of DT), $\alpha$, $\mathsf{EX}(f, D)$, (local)-$\mathsf{MQ}(f)$
  1. let $\mathcal{S} = \{\emptyset\}$; $\theta = (1+\alpha)^{-d-1}$
  2. for $i = 1, \ldots, d$
      (a) For every $S' \in \mathcal{S}$, $|S'| = i - 1$ and for every $j \in [n] \setminus S'$
          i. Let $S = S' \cup \{j\}$
          ii. (Non-Zero Test) If $\Pr_{x \sim D}[f_S(x) \neq 0] \geq \theta$, then $\mathcal{S} = \mathcal{S} \cup \{S\}$
  3. Let polynomial $h(x)$ over terms in $\mathcal{S}$ be obtained by minimizing $\mathbb{E}_{x \sim \mathcal{D}}[(f(x) - h(x))^2]$, constrained by $\sum_S |\hat{h}(S)| \leq t$.
Output: $\mathrm{sign}(h(x))$.
```

Figure 3: Algorithm: Learning log-depth decision trees

Note that in fact, $f_S$ is a function of $x_{\bar{S}}$. Let $x_{\bar{S}}$ be fixed. Let $U_S$ denote the uniform distribution over $x_S$. Observe that $f(x) = f_{-S}(x) + \chi_S(x_S) f_S(x_{\bar{S}})$. Now each monomial in $f_{-S}$ is missing at least one variable from $x_S$ (otherwise it would have been in $f_S$). Thus, $f_S(x_{\bar{S}}) = \mathbb{E}_{x_S \sim U_S}[f(x_S x_{\bar{S}})]$. Thus, if $x$ were a point drawn from the example oracle, $\mathsf{EX}(f, D)$, $f_S(x_{\bar{S}})$ can be computed using $|S|$-local membership queries. Now, $\Pr_{x \sim D}[f_S(x) \neq 0]$ can be estimated very accurately by sampling. We ignore the analysis that employs standard Chernoff bounds and assume that we can perform the test in Step 2.(a).ii. with perfect accuracy. We prove the following:

**Theorem 14** *The class of depth-$O(\log(n))$ decision trees is learnable using $O(\log(n/\epsilon))$-local membership queries under the class of locally $\alpha$-smooth distributions, for constant $\alpha$, in time that is polynomial in $n, 1/\epsilon, 1/\delta$.*

The main tools required to prove Theorem 14 are Lemmas 15 and 16. Lemma 15 shows that if some subset $S$ satisfies $\Pr[f_S(x) \neq 0] \geq \theta$, then there exists some $T \supseteq S$ such that $\hat{f}(T) \neq 0$. Lemma 16 can be used to show that if $S \subseteq T$, where $\hat{f}(T) \neq 0$ and $T$ is maximal for $f$, then $\Pr_{x \sim D}[f(x) \neq 0] \geq \theta$. Note that what Lemma 16 actually proves is that if $\Pr_{x \sim D}[f_S(x) \neq 0] \leq \theta$, then $\Pr_{x \sim D}[f_{S \cup \{i\}}(x) \neq 0] \leq \theta(1+\alpha)$, for any $i \notin S$. Now, if $T$ is maximal, and $S \subseteq T$, this would imply that $\Pr_{x \sim D}[f_T(x) \neq 0] \leq \theta(1+\alpha)^{|T|-|S|}$. But, if $T$ is maximal, the polynomial $f_T(x)$ is a non-zero constant polynomial. Thus, if $\theta$ is chosen to be smaller than $1/(1+\alpha)^d$, where $d = O(\log(n))$ is a bound on the depth of the decision tree being learned, the algorithm finds all the important monomials. Once all the important monomials have been found, the best polynomial can easily be obtained, for example by regression. Alternatively, one could also solve a system of linear equations on the monomials and this actually learns the decision tree *exactly*.[6]

[6] By this we mean, that the output hypothesis is such that $\Pr_{x \sim D}[h(x) \neq f(x)] = 0$, except with some small probability.

**Lemma 15** *For any set $S \subseteq [n]$, if for all $T \supseteq S$, $\hat{f}(T) = 0$, then* $\Pr_{x \sim D}[f_S(x) \neq 0] = 0$.

**Proof** The polynomial $f_S(x) \equiv 0$. ■

**Lemma 16** *Let $D$ be any locally $\alpha$-smooth distribution, then for any set $S$, and any $i \notin S$,* $\Pr_{x \sim D}[f_{S \cup \{i\}}(x) \neq 0] \geq (1/(1+\alpha)) \Pr_{x \sim D}[f_S(x) \neq 0]$.

**Proof** Recall, that $f_S(x) = \sum_{T \supseteq S} \hat{f}(T) \chi_{T \setminus S}(x)$. Also, note that $f_S$ is only a function of the variables, $x_{\bar{S}}$. Observe that,

$$f_S(x_{\bar{S}}) = \sum_{\substack{T \supseteq S \\ i \notin T}} \hat{f}(T) \chi_{T \setminus S}(x_{\bar{S}}) + x_i f_{S \cup \{i\}}(x_{-(S \cup \{i\})})$$

Note that $D_{\bar{S}}$, the marginal distribution, is also locally $\alpha$-smooth (see Fact 5). Let $(D_{\bar{S}} | f_{S \cup \{i\}}(x_{-(S \cup \{i\})}) = 1)$ be the conditional distribution, given $f_{S \cup \{i\}}(x_{-(S \cup \{i\})}) = 1$. Then, the marginal distribution, $(D_{\bar{S}} | f_{S \cup \{i\}}(x_{-(S \cup \{i\})}))_i$ is a distribution on only one variable, $x_i$, but is also locally $\alpha$-smooth (Fact 5). But, given that $f_{S \cup \{i\}}(x_{-(S \cup \{i\})}) \neq 0$, for one of the two values, $x_i = 1$ or $x_i = -1$, it must be the case that $f_S(x_{\bar{S}}) \neq 0$. Since, the distribution $(D_{\bar{S}} | f_{S \cup \{i\}}(x_{-(S \cup \{i\})}) = 1)_i$ is locally $\alpha$-smooth, the proof of the Lemma follows from Fact 5. (Note that $\Pr_{x \sim D}[f_S(x) \neq 0] = \Pr_{x_{\bar{S}} \sim D_{\bar{S}}}[f_S(x_{\bar{S}}) \neq 0]$, since $f_S$ does not depend on the variables $x_S$.) ■

### D.2. Learning Decision Trees under the Uniform Distribution

In this section, we present an algorithm for learning $t$-leaf decision trees (of arbitrary depth) under the uniform distribution. Although, the uniform distribution is a special case of product distributions considered in Section D.3, the exposition is simpler and conveys the high-level ideas better.

We use standard results from Fourier analysis; Kushilevitz and Mansour (1993) proved the following useful properties of the Fourier spectrum of decision trees. Let $f$ be a function that is represented by a $t$-leaf decision tree, then:

1. For any set $S \subseteq [n]$, $|\hat{f}(S)| \leq t/2^{|S|}$.
2. $L_1(f) = \sum_{S \subseteq [n]} |\hat{f}(S)| \leq t$.

Using the above relations, we can immediately prove the following useful (and well-known) fact.

**Fact 17** *Suppose $f$ is boolean function that is represented by $t$-leaf decision tree. Then, for any $\tau > 0$, $\sum_{S, |S| \geq \log(t^2/\tau)} \hat{f}(S)^2 \leq \tau$*

```
Algorithm: Learning Decision Trees
inputs: $d$, $\theta$, oracles $\mathsf{EX}(f, U)$, (local)-$\mathsf{MQ}(f)$

  1. let $\mathcal{S} = \{\emptyset\}$

  2. for $i = 1, \ldots, d$

       (a) for every $S' \in \mathcal{S}$, $|S'| = i - 1$ and for every $j \in [n] \setminus S'$
             i. let $S = S' \cup \{j\}$
            ii. ($L_2$ Test) if $\mathbb{E}_{x \sim U}[f_S(x)^2] > \theta^2$, then $\mathcal{S} = \mathcal{S} \cup \{S\}$

  3. let $h(x) = \sum_{S \in \mathcal{S}} \hat{f}(S) \chi_S(x)$

 output: $\mathrm{sign}(h(x))$
```

Figure 4: Algorithm: Learning Decision Trees under the Uniform Distribution

**Proof** Consider,

$$\begin{aligned}\sum_{S, |S| \geq \log(t^2/\tau)} \hat{f}(S)^2 &\leq \max_{S, |S| \geq \log(t^2/\tau)} |\hat{f}(S)| \cdot \left( \sum_{T, |T| \geq \log(t^2/\tau)} |\hat{f}(S)| \right) \\ &\leq t \cdot (\tau/t^2) \cdot L_1(f) \leq \tau\end{aligned}$$

■

The algorithm in Figure 4 learns $t$-leaf decision trees under the uniform distribution. For simplicity of presentation, we assume that the expectations used in the algorithm and also the Fourier coefficients can be computed *exactly*. It is easy to see that using standard applications of Chernoff-Hoeffding bounds, the guarantees of the algorithm hold even when the expectations and values of the Fourier coefficients can only be computed approximately. The main step in Algorithm 4 that requires some explanation is how to compute the quantity $\mathbb{E}_{x \sim U}[f_S(x)^2]$ to check if it is greater than $\theta^2$. We refer to this as the $L_2$ Test.

$L_2$ **Test**: Let $x \in \{-1, 1\}^n$, and recall that for $S \subseteq [n]$, $f_S(x) = \sum_{T \supseteq S} \hat{f}(T) \chi_{T \setminus S}(x)$, and that this can be computed by using the fact that,

$$f_S(x_{\bar{S}}) = \mathbb{E}_{x_S \sim U_S}[\chi_S(x) f(x)]$$

Given a point $(x, f(x))$, we observe that the expectation $\mathbb{E}_{x_S \sim U_S}[f(x)\chi_S(x)]$ can be computed using $2^{|S|}$, $|S|$-local membership queries with respect to $x$ (only the bits in $S$ need to be flipped). The quantity $\mathbb{E}_{x \sim U}[f_S(x)^2]$ can thus be computed easily using only $|S|$-local membership queries and taking a sample from $\mathsf{EX}(f, D)$.

**High-Level Overview of Proof**: Fact 17 showed that the Fourier mass (sum of squares of the Fourier coefficients) of $t$-leaf decision trees is concentrated on low degree terms. Parseval's identity implies that this is sufficient to construct a polynomial, $h(x)$, that is a good $\ell_2$ approximation to the decision tree, $f$, i.e. $\mathbb{E}_{x \sim U}[(h(x) - f(x))^2] \leq \epsilon$. Also, Kushilevitz and Mansour (1993) showed that since $L_1(f)$ is bounded, most of the Fourier mass is concentrated on a small (polynomially many) number of terms.

The main insight here is that, these terms on which most of the Fourier mass is concentrated, can be identified using only $O(\log(n))$-local membership queries. It is relatively easy to see that any coefficient for which $|\hat{f}(S)| \geq \theta$ will be identified correctly by the test in line 2.(a).ii. (Figure 4). We show that the quantity $|\mathcal{S}|$ never grows too large. To show this, we prove that if any coefficient is inserted in $\mathcal{S}$ in line 2.(a).ii, it must be a subset of some coefficient of large magnitude. This follows quite easily by observing that $\mathbb{E}_{x \sim U}[f_S(x)^2] = \sum_{T \supseteq S} \hat{f}(T)^2$ and using the fact that $L_1(f)$ is bounded.

The rest of the section is devoted to a formal proof of the above overview.

**Claim 18** *Suppose that $S$ is such that $|\hat{f}(S)| \geq \theta$ and $|S| \leq d$, then $S \in \mathcal{S}$.*

**Proof** First observe that for any subset $S' \subseteq S$, it holds that $\mathbb{E}[f_{S'}(x)^2] \geq \theta^2$. This follows immediately by observing that

$$\mathbb{E}[f_{S'}(x)^2] = \sum_{T \supseteq S'} \hat{f}(T)^2 \geq \hat{f}(S)^2 \geq \theta^2$$

It follows by a simple induction argument that at iteration $i$, $\mathcal{S}$ contains every subset of $S$ of size at most $i$, for which $\mathbb{E}[f_S(x)^2] \geq \theta^2$. And, hence $S \in \mathcal{S}$. ■

**Claim 19** *If $S \in \mathcal{S}$, then there exists a $S' \supseteq S$ such that $\hat{f}(S') \geq \theta^2/t$.*

**Proof** Since $S \in \mathcal{S}$, we know that $\mathbb{E}[f_S(x)^2] = \sum_{T \supseteq S} \hat{f}(S)^2 \geq \theta^2$. But observe that,

$$\sum_{T \supseteq S} \hat{f}(T)^2 \leq \left( \sum_{T \supseteq S} |\hat{f}(T)| \right) \cdot \max_{T \supseteq S} |\hat{f}(T)|$$

The above inequality simply states the fact that $L_2(f_S) \leq L_1(f_S) L_\infty(f_S)$. Since $f$ is a $t$-leaf decision tree, $\sum_{T \supseteq S} |\hat{f}(T)| \leq L_1(f) \leq t$. The claim now follows immediately. ■

Using the above claims, it is easy to show our main theorem.

**Theorem 20** *Algorithm in Fig. 4 run with parameters $d = \log(2t^2/\epsilon)$ and $\theta = \epsilon/(2t)$, outputs a hypothesis, $\mathrm{sign}(h(x))$, where $\mathrm{err}_U(\mathrm{sign}(h(x), f) \leq \epsilon$. The running time is $\mathrm{poly}(t, n, 1/\epsilon)$ and the algorithm only makes $\log(2t^2/\epsilon)$-local queries to the membership oracle $\mathsf{MQ}(f)$.*

**Proof** First, we recall that for a $t$-leaf decision tree, $|\hat{f}(S)| \leq t/2^{|S|}$ (Kushilevitz and Mansour, 1993). Thus, if $|\hat{f}(S)| \geq \theta^2/t$, then $|S| \leq 2\log(t/\theta)$. Using Parseval's identity (see Section 2), we know that the number of Fourier coefficients that have magnitude greater than $\theta^2/t$ is at most $t^4/\theta^2$.

Consider the set $\mathcal{S}$ constructed by the algorithm (Fig. 4) at the end of $d$ iterations. If $S \in \mathcal{S}$, then there must exist some $T \supseteq S$ such that $|\hat{f}(T)| \geq \theta^2/t$ (Claim 19). But there can be at most $t^2/\theta^4$ such terms and each is of size at most $2\log(t/\theta)$. Hence, the $|\mathcal{S}| \leq (t^2/\theta^4)2^{2\log(t/\theta)} = t^4/\theta^6$.

For any coefficient, such that $|\hat{f}(S)| \geq \theta$, it must be that $|S| \leq \log(t/\theta) \leq d$. Claim 18 shows that all such coefficients are included in $\mathcal{S}$. Thus, $\max_{S \notin \mathcal{S}} |\hat{f}(S)| \leq \theta$. Hence, $\sum_{S \notin \mathcal{S}} \hat{f}(S)^2 \leq \sum_{S \notin \mathcal{S}} |\hat{f}(S)| \cdot \max_{S \notin \mathcal{S}} |\hat{f}(S)| \leq L_1(f) \cdot \theta \leq \theta t$. But $\mathbb{E}[(h(x) - f(x))^2] = \sum_{S \notin \mathcal{S}} \hat{f}(S)^2$ and also notice that $\Pr_{x \sim U}[\mathrm{sign}(h(x)) \neq f(x)] \leq \mathbb{E}_{x \sim U}[(h(x) - f(x))^2]$ (since $f(x)$ only takes values ±1). ■

### D.3. Learning Decision Trees under Product Distributions

In this section, we prove that the class of $t$-leaf decision trees can be learned under the class of product distributions, where each bit has mean bounded away from $-1$ and 1. Let $\mu = (\mu_1, \ldots, \mu_n)$ denote a product distribution over $X = \{-1, 1\}^n$, where $\mathbb{E}_{x \sim \mu}[x_i] = \mu_i \in [-1 + 2c, 1 - 2c]$, for some constant $c \in (0, 1/2]$. We use Fourier analysis using the modified basis for the product distribution. We begin by introducing required notation for using Fourier techniques.

**Fourier Analysis over** $\mu$: Let $\mu = (\mu_1, \ldots, \mu_n)$ be the product distribution over $X = \{-1, 1\}^n$, where $\mathbb{E}_{x \sim \mu}[x_i] = \mu_i$. Define,

$$\chi_S^\mu(x) = \prod_{i \in S} \frac{x_i - \mu_i}{\sqrt{(1 - \mu_i^2)}}.$$

Then, it is easy to observe that for any two sets $S_1 \neq S_2$, $\mathbb{E}_{x \sim \mu}[\chi_{S_1}^\mu(x)\chi_{S_2}^\mu] = 0$ and that, for any set $S$, $\mathbb{E}_{x \sim \mu}[\chi_S^\mu(x)^2] = 1$. Thus, the set of functions $\langle \chi_S^\mu(x) \rangle_{S \subseteq [n]}$ forms an orthonormal basis for functions defined on $\{-1, 1\}^n$ under the distribution $\mu$. For any function $f : \{-1, 1\}^n \to \mathbb{R}$, the Fourier coefficients under distribution $\mu$ are defined as $\hat{f}^\mu(S) = \mathbb{E}_{x \sim \mu}[f(x)\chi_S^\mu(x)]$. The following is Parseval's identity in this basis:

$$\mathbb{E}_{x \sim \mu}[f(x)^2] = \sum_{S \subseteq [n]} \hat{f}^\mu(S)^2 \tag{2}$$

In particular, when $f$ is a boolean function, i.e. with range $\{-1, 1\}$, the sum of Fourier coefficients is 1.

Let $L_1^\mu(f) = \sum_{S \subseteq [n]} |\hat{f}^\mu(S)|$, $L_2^\mu(f) = \sum_{S \subseteq [n]} \hat{f}^\mu(S)^2$ and $L_\infty^\mu(f) = \max_{S \subseteq [n]} |\hat{f}^\mu(S)|$ denote the 1, 2 and ∞ norm of the Fourier spectrum under distribution $\mu$. Also let $L_0^\mu(f) = |\{S \mid \hat{f}^\mu(S) \neq 0\}|$ denote the number of non-zero Fourier coefficients of $f$. We will frequently use the following useful observations:

1. $L_2^\mu(f) \leq L_1^\mu(f) \cdot L_\infty^\mu(f)$
2. $L_2^\mu(f) \leq L_0^\mu(f) \cdot (L_\infty^\mu(f))^2$

```
Algorithm: Learning Decision Trees
inputs: $d$, $\theta$, oracles $\mathsf{EX}(f, \mu)$, $\mathsf{MQ}(f)$
# $f$ is a $t$-leaf decision tree
# $\mu$ is a product distribution over $\{-1, 1\}^n$, $\mu_i \in [-1 + 2c, 1 - 2c]$

  1. let $\mathcal{S} = \{\emptyset\}$
  2. for $i = 1, \dots, d$
       (a) for every $S' \in \mathcal{S}$, $|S'| = i - 1$ and for every $j \in [n] \setminus S'$
             i. let $S = S' \cup \{j\}$
            ii. ($L_2$ Test) if $\mathbb{E}_{x \sim \mu}[f_S(x)^2] > \theta^2$, then $\mathcal{S} = \mathcal{S} \cup \{S\}$
  3. let $h(x) = \sum_{S \in \mathcal{S}} \hat{f}^{\mu}(S) \chi^{\mu}_S(x)$
 output: $\mathrm{sign}(h(x))$
```

Figure 5: Algorithm: Learning Decision Trees under Product Distributions

### D.3.1. Decision Tree Learning Algorithm

We present a high-level overview of our algorithm and a formal statement of the main result, before providing full details. The Algorithm is described in Figure 5.

**Truncation**: We show that a $t$-leaf decision tree, when truncated to logarithmic depth, is still a very good (inverse polynomially close) approximation to the original decision tree. This observation can be used to show that it suffices to identify low-degree (logarithmic) "heavy" Fourier coefficients of $f$, with respect to the distribution, $\mu$, and also that the number of such terms is not too large (at most polynomial). Note that this is not as simple as in the case of the uniform distribution, because it is not straightforward to bound $L_1^{\mu}(f) = \sum_{S \subseteq [n]} |\hat{f}^{\mu}(S)|$. (When $\mu$ is the uniform distribution, this is bounded by $t$.) Properties of such truncated decision trees were also used by Kalai et al. (2009b) in the smoothed analysis setting.

A $t$-leaf decision tree can be though of as $t$ (not disjoint) paths from root to leaves. A truncation of a decision tree at depth $d$, is a decision tree where for each path of length more than $d$, only the prefix (from root) of length $d$ is preserved. Note that this may collapse several paths to the same prefix, possibly reducing the number of leaves. A new leaf is added at the end of this path and labeled arbitrarily as $-1$ or $+1$.

For any function $g$, we denote by $\mathcal{S}^{\mu}_g$, the set of non-zero Fourier coefficients of $g$, with respect to the product distribution, $\mu$, *i.e.* $\mathcal{S}^{\mu}_g = \{T \subseteq [n] \mid \hat{g}(T) \neq 0\}$.

We prove two useful properties of the truncated decision trees with respect to product distribution. These appear as formal statements in Lemmas 21 and 22. Similar observations were also used by Kalai et al. (2009b) to prove learning of decision trees in the smoothed analysis setting.

(i) Truncation at logarithmic depth is a good approximation (inverse polynomial) to the original decision tree.

(ii) The number of nonzero Fourier coefficients of the truncated decision tree, $|\mathcal{S}_g^\mu|$ is small (polynomial).

**Lemma 21** *Let $f$ be a $t$-leaf decision tree, let $\mu$ be a product distribution over $X = \{-1, 1\}^n$ such that $\mu_i \in [-1 + 2c, 1 - 2c]$. Then for every $\tau > 0$, there exists a $t$-leaf decision tree of depth at most $\log(t/\tau)/\log(1/(1 - c))$, such that $\Pr_{x \sim \mu}[g(x) \neq f(x)] \leq \tau$*

**Proof** Let $g$ be the decision tree obtained by truncating $f$ at depth $d$. The new leaves added at depth $d$ can be labeled arbitrarily. Now, the points $x$ for which $g(x) \neq f(x)$ are precisely those, for which $g$ would lead to the newly added leaf node at depth $d$. But since $\mathbb{E}_{x \sim \mu}[x_i] \in [-1 + 2c, 1 - 2c]$, the probability that a random point from $\mu$ reaches such a node is at most $(1 - c)^d$. The number of new leaf nodes added cannot be more than $t$ (since any truncation only reduces the number of leaves). Thus, $\Pr_{x \sim \mu}[g(x) \neq f(x)] \leq t(1 - c)^d$. When, $d = \log(t/\tau)/\log(1/(1 - c))$ we get the result. ■

**Lemma 22** *Let $g$ be a decision tree of depth $d$ and $t$ leaves; then the number of non-zero Fourier coefficients of $g$ is at most $t \cdot 2^d$ and each is of size at most $d$.*

**Proof** We consider any path in $g$ from root to leaf, and let $P$ denote the subset of indexes corresponding to the variable that occur in the path. First, we expand decision tree $g$ as a polynomial.

$$g(x) = \sum_{\text{path } P} y_P \prod_{i \in P} \frac{1 + \sigma_{P,i} x_i}{2},$$

where $\sigma_{P,i}$ is $+1$ or $-1$, depending on whether the path leading out of node labeled $x_i$ on path $P$ was labeled $+1$ or $-1$, and $y_P$ is the label of the leaf at the end of the path $P$.

The only nonzero coefficients in $g$ are of the from $\prod_{i \in T} x_i$ for some $T \subseteq P$ for some path $P$. This also means that the only non-zero Fourier coefficients can be those corresponding to such subsets. This is because $\mathbb{E}_{x \sim \mu}[\chi_T^\mu(x) \prod_{i \in S} x_i] = 0$, unless $T \subseteq S$ (because $\mu$ is a product distribution). Since the number of paths in $g$ is at most $t$ and the length of each path is at most $d$, we get the required result. ■

**Lemma 23** *Let $f$ be a $t$-leaf decision tree, let $g$ be a truncation of $f$ to depth* $\log(4t/\tau)/\log(1/(1-c))$*. Then,*

$$\sum_{S, S \notin \mathcal{S}_g^\mu} \hat{f}_\mu(S)^2 \leq \tau$$

**Proof** Let $g$ be a truncation of $f$ at depth $\log(4t/\tau)/\log(1/(1-c))$. Let $\mathcal{S}^{\mu}_{g}$ denote the set of non-zero Fourier coefficients of $g$ under distribution $\mu$. Using Lemma 21, we know that $\Pr_{x\sim\mu}[f(x) \neq g(x)] \leq \tau/4$, hence $\mathbb{E}[(f(x)-g(x))^2] \leq \tau$. Now, by Parseval's identity:

$$\begin{aligned}\tau &\geq \mathbb{E}_{x\sim\mu}[(f(x)-g(x))^2] \\ &= \sum_{S\subseteq \mathcal{S}_g} (\hat{f}(S)-\hat{g}(S))^2 + \sum_{S\notin \mathcal{S}_g} \hat{f}(S)^2 \\ &\geq \sum_{S\notin \mathcal{S}_g} \hat{f}(S)^2\end{aligned}$$

The proof is complete by observing that every coefficient $S \in \mathcal{S}^{\mu}_{g}$ satisfies $|S| \leq \log(4t/\tau)/\log(1/(1-c))$ by Lemma 22. ■

$L_2$ **Test**: As in the case of uniform distribution, we write $f(x)$ as:

$$f(x) = f^{\mu}_{-S}(x) + \chi^{\mu}_{S}(x) f^{\mu}_{S}(x),$$

where, $f^{\mu}_{-S}(x) = \sum_{T, S\not\subseteq T} \hat{f}^{\mu}(T)\chi^{\mu}_{T}(x)$ and $f^{\mu}_{S}(x) = \sum_{T\supseteq S} \hat{f}^{\mu}(T)\chi^{\mu}_{T\setminus S}(x)$. Then as in the case of uniform distribution, $f_S(x) = f_S(x_{-S}) = \mathbb{E}_{x_S\sim\mu_S}[f(x)\chi^{\mu}_{S}(x)]$, where now $x_S$ is drawn from the restriction $\mu_S$ of the product distribution to the bits $x_S$. Note that for any given point $x$, $f_S(x)$ can be computed easily using $2^{|S|}$ membership queries that are $|S|$-local (since only the bits $x_S$ need to be changed). We point out that there is a subtle point in the case of product distributions. Recall that $f_S(x) = \mathbb{E}_{x_S\sim\mu_S}[f(x)\chi^{\mu}_{S}(x)]$. In the case when $\mu$ is the uniform distribution, the parity functions, $\chi_S$ are $\{-1, 1\}$ valued, and so $f_S(x) \in [-1, 1]$. Thus, application of Chernoff-Hoeffding bounds is straightforward. In the case, of product distributions the range of $\chi^{\mu}_{S}(x)$ can be $[-\prod_{i\in S}((1-|\mu_i|)/(\sqrt{1-\mu_i^2})), \prod_{i\in S}((1+|\mu_i|)/(\sqrt{1-\mu_i^2}))]$. Since, we never consider sets $S$ that are larger than $O(\log(n/\epsilon))$, the range of $f_S$ in our case is still polynomially bounded and arbitrarily good (inverse polynomial) estimates to the true expectation of $\mathbb{E}_{x\sim\mu}[f_S(x)^2]$ can be obtained by taking a sample and applying Chernoff-Hoeffding bounds. Thus, to simplify the presentation, we assume we can compute the expectation (in Line 2.a.ii in Fig. 5) and the Fourier coefficients exactly.

Theorem 24 is the statement of the formal result about learning decision trees under product distributions. The main ideas are similar to the proof in the case of uniform distribution; but, the proof is more involved as explained above.

**Theorem 24** *Algorithm in Fig. 5 with parameters* $\theta = \sqrt{\epsilon/(2t(8t/\epsilon)^{1/\log(1/(1-c))})}$, $d = \log(8t/\epsilon)/\log(1/(1-c))$, *outputs a hypothesis* $\mathrm{sign}(h(x))$, *such that* $\mathrm{err}_{\mu}(\mathrm{sign}(h(x)), f) \leq \epsilon$. *The running time of the algorithm is polynomial in $n$, $t$ and $1/\epsilon$ and the algorithm makes only $O(\log(nt/\epsilon))$-local membership queries to the oracle* $\mathsf{MQ}(f)$.

The rest of this section is devoted to the proof of Theorem 24.

**Claim 25** *If $S$ is such that $|\hat{f}^{\mu}(S)| \geq \theta$ and $|S| \leq d$, then $S \in \mathcal{S}$.*

**Proof** This proof is thee same as the proof of Claim 18. ■

**Claim 26** *If $S \in \mathcal{S}$, then there exists $S' \supseteq S$, such that $\hat{f}^\mu(S')^2 \geq (\theta^2/2)/(t\cdot(8t/\theta^2)^{1/\log(1/(1-c))})$ and $|S'| \leq \log(8t/\theta^2)/\log(1/(1-c))$.*

**Proof** Let $\tau = \theta^2/2$ and let $g'$ be the decision tree obtained by truncation of $f$ as described in Lemma 23. Then, by Lemma 22, we know the depth of $g'$ is $\log(8t/\theta^2)/\log(1/(1-c))$ and that $\mathcal{S}^\mu_{g'}$ is of size at most $t\cdot 2^{\log(8t/\theta^2)/\log(1/(1-c))} = t\cdot(8t/\theta^2)^{1/\log(1/(1-c))}$. Also, by Lemma 23 we know that $\sum_{T \not\in \mathcal{S}^\mu_{g'}} \hat{f}^\mu(T)^2 \leq \theta^2/2$, and hence if $S$ passes the $L_2$-test, i.e. $\sum_{T\supseteq S} \hat{f}^\mu(T)^2 \geq \theta^2$, it must be that $\sum_{T\supseteq S, T\in\mathcal{S}^\mu_{g'}} \hat{f}^\mu(T)^2 \geq \theta^2/2$. Hence, there must be some set $S'$ of size at most $\log(8t/\theta^2)/\log(1/(1-c))$ for which $\hat{f}^\mu(S')^2 \geq (\theta^2/2)/(t\cdot(8t/\theta^2)^{1/\log(1/(1-c))})$. ■

**Proof** [Proof of Theorem 24] Let $g$ be the truncation of the target decision tree, $f$, to depth $d$. Then using Lemma 23, we know that $\sum_{S\not\in\mathcal{S}^\mu_{g'}} \hat{f}^\mu(S)^2 \leq \epsilon/2$. Now, every coefficient in $S \in \mathcal{S}^\mu_g$ for which $|\hat{f}^\mu(S)| \geq \theta$ is in $\mathcal{S}$ (see Algorithm 5 and Claim 25). $|\mathcal{S}^\mu_{g'}| \leq t2^d$. Tedious calculations show that $\sum_{S\in\mathcal{S}^\mu_{g'}, |\hat{f}(S)|<\theta} \hat{f}(S)^2 \leq t2^d\theta^2 \leq \epsilon/2$. Thus, $\sum_{S\in\mathcal{S}} \hat{f}(S)^2 \geq \sum_{S\in\mathcal{S}\cap\mathcal{S}^\mu_{g'}} \hat{f}(S)^2 \geq 1-\epsilon$. This implies by Parseval, that $\mathbb{E}_{x\sim\mu}[(h(x)-f(x))^2] \leq \epsilon$, where $h(x)$ is as defined in Algorithm 5.

The only thing remaining to show is that $|\mathcal{S}|$ always remains bounded by $\mathrm{poly}(t, n, 1/\epsilon)$. This can be shown easily using Claim 26, since if $S \in \mathcal{S}$, there exists $S' \supseteq S$, such that $|S'| \leq \log(8t/\theta^2)/\log(1/(1-c))$ and $\hat{f}^\mu(S)^2 \geq (\theta^2/2)/(t\cdot(8t/\theta^2)^{1/\log(1/(1-c))}$. Thus, the magnitude of $\hat{f}(S')^2$ is at least $1/\mathrm{poly}(t, n, 1/\epsilon)$, so by Parseval there can be at most $\mathrm{poly}(t, n, 1/\epsilon)$. Also the size of $|S'|$ is $O(\log(tn/\epsilon))$, thus the total number of irrelevant subsets added to $\mathcal{S}$ is at most $\mathrm{poly}(t, n, 1/\epsilon)$. ■

### D.4. Learning under Random Classification Noise

In this section, we show how the algorithms for learning decision trees can be implemented even with access to a noisy oracle. The learning algorithm we use is allowed queries to the membership oracle, $\mathsf{MQ}(f)$, therefore we consider a persistent random noise model. An easy way to conceptualize this model is as follows: Let $\zeta : \{-1,1\}^n \to \{-1,1\}$ be a function where for each $x \in \{-1,1\}^n$, the value of $\zeta(x) = 1$ with probability $1-\eta$ and $-1$ with probability $\eta$, independently. Once this noise function, $\zeta$, has been fixed, we assume that we have access to the function: $f^\eta = f\cdot\zeta$, rather than the function $f$. We show how the tests mentioned in this section can be implemented using $\mathsf{EX}(f^\eta, D)$ and $\mathsf{MQ}(f^\eta)$, rather than $\mathsf{EX}(f, D)$ and $\mathsf{MQ}(f)$.

#### D.4.1. Non-Zero Test

Recall that we are interested in estimating $\Pr[f_S(x) \neq 0]$, where $S \subseteq [n]$, and

$$f_S(x) = \mathbb{E}_{x_S\sim U_S}[f(x)\chi_S(x)] \tag{3}$$

Instead, if we have access to $f^\eta$, we are able to compute,

$$f^\eta_S(x) = \mathbb{E}_{x_S \sim U_S}[f^\eta(x)\chi_S(x)]$$

Although, the random classification noise is persistent and fixed according to $\zeta$, for the purpose of analysis it is easier to imagine that for each $x$, $\zeta(x)$ is only determined when the algorithm makes a query for the point $x$ (or $x$ is drawn by $\mathsf{EX}(f^\eta, D)$). Lemma 27 allows us to conclude that the test required in Section D.1 can be performed using access to $f^\eta$ instead of $f$. The lemma assumes that $\zeta(x)$ is chosen independently, each time $x$ is queried, *i.e.* the noise is not *persistent*. However, we show later that our algorithm queries each example only once, so the noise may as well have been persistent.

**Lemma 27** *The following are true:*

1. $\Pr_{x,\zeta}[f^\eta_S(x) \neq 0] \geq (1 - p_0) + \frac{(2\eta-1)^2 c_0}{2^{3|S|/2}} \Pr[f_S(x) \neq 0]$
2. $\Pr_{x,\zeta}[f^\eta_S(x) \neq 0] \leq (1 - p_0) + \Pr[f_S(x) \neq 0]$

*Here, $c_0$ is an absolute constant, $p_0$ depends only on $|S|$ and $\eta$. The probability is taken over the choice of $x \sim D$ and choice of $\zeta$.*

**Proof** Note that $f_S(x) = \mathbb{E}_{x_S \sim U_S}[f(x)\chi_S(x)]$, and so $f_S(x)$ is evaluated by using $2^{|S|}$ different values of $f(x)$. For every $x$, $f(x) \in \{-1, 1\}$, and hence if $f_S(x) = 0$, it must be that the $2^{|S|}$ values used in the expectation have exactly $2^{|S|-1}$ $+1$s and $-1$s each.

On the other hand, if $f_S(x) \neq 0$, then the number of $+1$s is different than $-1$s. If $f_S(x) \neq 0$, without loss of generality, we only consider the case when $f_S(x) > 0$, so that there are more $+1$s than $-1$s. Thus, we are left with the following combinatorial question:

Suppose we begin with $2k$ variables, $x_1, \ldots, x_{2k}$, where each $x_i$ is $+1$ or $-1$. Let $k_1$ be the number of $+1$s and $2k - k_1$ is the number of $-1$s. We will assume throughout that $k \geq 2$. We perform the following process, each $x_i$ is left as is with probability $1 - \eta$ and its sign flipped with probability $\eta$, independently. Let $x'_i$ be the values of the resulting variables, and let $X' = \sum_i x'_i$. Let $p^k_i$ denote the probability that $X'$ is 0 having started with $(k + i)$ $+1$s and $(k - i)$ $-1$s. Thus, $p^k_0$ is the probability of getting a 0, when we start with equal number of $+1$s and $-1$s.

Then the following are true:

1. $p^k_i$ decreases as $i$ increases.
2. $p^k_0 - p^k_1 \geq (2\eta - 1)^2 c_0 / k^{3/2}$ for some absolute constant $c_0$.

For proof of the above facts see Lemmas 29 and 30 below, though it should be fairly clear that the conclusions make sense. When $\eta = 1/2$, the initial values are irrelevant of the $x_i$ are irrelevant and each $x'_i = \pm 1$ with probability $1/2$, but for $\eta < 1/2$, if one started with the sum $\sum_i x_i = 0$, it is more likely that $\sum_i x'_i = 0$, than if one started from some value, $\sum_i x_i$ that was greater than 0.

We apply the above to the setting when $k = 2^{|S|-1}$. We drop the superscripts $p_0^{2^{|S|-1}}$ and $p_1^{2^{|S|-1}}$ in the rest of this discussion. First, imagine that we have fixed the variables $x_{-S}$ so that the expectation (3) is only a function of the noise function $\zeta$. If $f_S(x_{-S}) = 0$, then

$\Pr_\zeta[f_S^\eta(x_{-S}) = 0] = p_0$. On the other hand, if $f_S(x_{-S}) \neq 0$, then $0 \leq \Pr_\zeta[f^\eta(x_{-S}) = 0] \leq p_1$. So, we have the following:

$$\begin{aligned}\Pr_{x,\zeta}[f_S^\eta(x) \neq 0] &\geq \Pr_x[f_S(x) \neq 0](1 - p_1) + \Pr_x[f_S(x) = 0](1 - p_0) \\ &= (1 - p_0) + (p_0 - p_1) \Pr_{x \sim D}[f_S(x) \neq 0]\end{aligned}$$

On the other hand,

$$\begin{aligned}\Pr_{x,\zeta}[f_S^\eta(x) \neq 0] &\leq \Pr_x[f_S(x) \neq 0] + (1 - p_0) \Pr_x[f_S(x) \neq 0] \\ &\leq (1 - p_0) + \Pr_x[f_S(x) \neq 0]\end{aligned}$$

This completes the proof of the assertion. ■

We note that this allows us to distinguish between the cases where $\Pr_{x\sim D}[f_S(x) \neq 0] \geq \alpha$ from $\Pr_{x\sim D}[f_S(x) \neq 0] \leq \beta$, as long as $\alpha - \beta$ is sufficiently large. This can be done by choosing $\beta = \alpha \cdot (2\eta - 1)^2 c_0/(2 \cdot 2^{3|S|/2})$, and then computing the value $\Pr_{x\sim D}[f_S^\eta(x) \neq 0]$. Note that $p_0$ can be computed exactly, if the size $|S|$ and the noise rate $\eta$ are known. We assume that the noise rate is known; if not, the standard trick of binary searching the noise rate can be employed. Note that these tests can be carried out to high accuracy from samples. Now, in the case when $D$ is an locally $\alpha$-smooth distribution for constant $\alpha$, any two points $x$ and $x'$ drawn from $\mathsf{EX}(f, D)$ will have Hamming distance $\Omega(n)$ with very high probability. The local queries to $\mathsf{MQ}(f^\eta)$ are only made for points that are at Hamming distance $O(\log(n))$ from sampled points (see Fact 5). Thus, with very high probability, the queries made to compute $f_S^\eta(x)$ and $f_S^\eta(x')$ do not have any point in common, *i.e.* no example is queried twice by the learning algorithm. So we can employ Lemma 27 as if the noise was chosen independently each time a point was queried.

### D.4.2. $L_2$ Test

Recall that $f_S^\eta(x_{-S}) = \frac{1}{2^{|S|}} \sum_{x_S \in \{-1,1\}^{|S|}}[f^\eta(x_S x_{-S})]$. For a fixed $x_{-S}$, $f^\eta(x_{-S})$ is a random variable depending only on the noise function $\zeta$. Let $2^{|S|} f_S(x) = 2k$, where $2k$ is some even integer in the range $[-2^{|S|}, 2^{|S|}]$. Let $k_1 = 2^{|S|-1} + k = 2^{|S|-1}(1 + f_S(x))$ and $k_2 = 2^{|S|-1} - k = 2^{|S|-1}(1 - f_S(x))$, so that $2^{|S|} f_S(x)$ is a sum of $k_1$, $+1$s and $k_2$, $-1$s. Let $Z_1 \sim \mathsf{Bin}(k_1, \eta)$ and $Z_2 \sim \mathsf{Bin}(k_2, \eta)$ be binomial random variables. Then $2^{|S|} f^\eta(x_{-S}) = 2^{|S|} f_S(x) - 2Z_1 + 2Z_2$. This follows immediately from the definition of the noise model. The following can then be verified by straightforward calculations,

$$\begin{aligned}\mathbb{E}_\zeta[f_S^\eta(x_{-S})] &= (1 - 2\eta) f_S(x) \\ \mathbb{E}_\zeta[f_S^\eta(x_{-S})^2] &= (1 - 2\eta)^2 f_S(x)^2 + 2^{-|S|+1}\eta(1 - \eta)\end{aligned}$$

Thus, if we can obtain accurate estimates of $\mathbb{E}_{x\sim D}[f_S^\eta(x)^2]$, we can also obtain accurate estimates of $\mathbb{E}_{x\sim D}[f_S(x)^2]$. Again, as in the previous case, we observe that the algorithm (with high probability) never makes a query twice for the same example. Thus, we can assume that the noise model is in fact not persistent. It is clear that $\mathbb{E}_{x\sim D}[f_S^\eta(x)^2]$ can be estimated highly accurately by sampling.

The proof of the following two lemmas are elementary and are omitted.

**Lemma 28** *Suppose, $X_0 = 0$. Consider the following random walk, $X_{i+1} = X_i$ with probability $1 - \alpha$, $X_{i+1} = X_i + 2$, with probability $\alpha/2$ and $X_{i+1} = X_i - 2$, with probability $\alpha/2$, where $\alpha \in [0, 1/2]$. Then, for $i \geq 0$, $\Pr[X_n = 0] - \Pr[X_n = 2]$ is a decreasing function of $\alpha$.*

The idea of the proof is to notice that the probability, $\Pr[X_n = 2j]$ follows a bell shaped curve, and the curve gets steeper (more mass is concentrated at 0) as $\alpha$ goes to 0.

**Lemma 29** *Let $x_1, \ldots, x_{2n}$, be such that $x_1 = \cdots = x_{n+d} = 1$ and $x_{n+d+1} = x_{n+i+2} = \cdots x_{2n} = -1$. The sign of each $x_i$ is flipped independently with probability $\eta < 1/2$, to get $x'_i$. Let $p^n_d$ be the probability that the $\sum_i x'_i = 0$. Then for $d \geq 0$, as $d$ increases, $p^n_d$ decreases.*

This expresses the quite obvious idea that if the probability of flipping is less than half, then the further from 0 the initial sum ($\sum_i x_i$), the less likely it is that $\sum_i x'_i = 0$.

**Lemma 30** *Let $x_1, \ldots, x_{2n}$, be such that $x_1 = \cdots x_{n+1} = 1$ and $x_{n+2} = \cdots = x_{2n} = -1$. The sign of each $x_i$ is flipped independently with probability $\eta < 1/2$, to get $x'_i$. Let $p_1$ denote the probability that $\sum_i x'_i = 0$. Let $y_1, \ldots, y_{2n}$ be such that, $y_1 = \cdots y_n = 1$ and $y_{n+1} = \cdots = y_{2n} = -1$. Then, let $y'_i$ be obtained by flipping $y_i$ independently with probability $\eta < 1/2$, and let $p_0$ denote the probability that $\sum_i y'_i = 0$. Then $p_0 - p_1 \geq (2\eta - 1)^2 c_0 / n^{3/2}$, for some absolute constant $c_0$.*

**Proof** First we leave aside the values, $x'_n$, $x'_{n+1}$, $y'_n$ and $y'_{n+1}$. The remaining variables, both in the case of $x_i$s and $y_i$s, were obtained by starting with exactly $(n-1)$ +1s and $(n-1)$ −1s and flipping each independently with probability $\eta < 1/2$. We can form pairs of $(+1, -1)$, to get a random variable $z_i = x'_i + x'_{n+1+i}$, $i = 1, \ldots, n-1$, where $z_i = 0$ with probability $\eta^2 + (1-\eta)^2 > 1/2$, $z_i = +2$ with probability $\eta(1-\eta)$ and $z_i = -2$ with probability $\eta(1-\eta)$. (A similar argument can be made in the case of $y'_i$s.) We can view the sum of these $z_i$ random variables as a random walk described in Lemma 28, where $X_{i+1} = X_i$ with probability $\eta^2 + (1-\eta)^2$ and $X_{i+1} = X_i + 2$, with probability $\eta(1-\eta)$ and $X_{i+1} = X_i - 2$, with probability $\eta(1-\eta)$. Now, $p_1 = \Pr[X_{n-1} = 0](2\eta(1-\eta)) + \Pr[X_{n-1} = 2]\eta^2 + \Pr[X_{n-1} = -2](1-\eta)^2$. On the other hand, $p_0 = \Pr[X_{n-1} = 0](\eta^2 + (1-\eta)^2) + \Pr[X_{n-1} = 2]\eta(1-\eta) + \Pr[X_{n-1} = -2]\eta(1-\eta)$. Noticing that, $\Pr[X_{n-1} = 2] = \Pr[X_{n-1} = -2]$, we get that $p_0 - p_1 = (2\eta - 1)^2(\Pr[X_{n-1} = 0] - \Pr[X_{n-1} = 2])$. But this difference is a decreasing function of $\alpha = 1 - (\eta^2 + (1-\eta)^2)$. But, even when $\alpha = 1/2$, i.e. $\eta = 1/2$, this difference is given by,

$$\begin{aligned}
\Pr[X_{n-1} = 0] - \Pr[X_{n-1} = 2] &= \frac{1}{2^{2n-2}}\left(\binom{2n-2}{n-1} - \binom{2n-2}{n}\right) \\
&= \frac{1}{2^{2n}}\binom{2n-2}{n-1}\left(1 - \frac{n-2}{n}\right)
\end{aligned}$$

The claim now follows easily. ■

## Appendix E. Lower Bound for Agnostically Learning Parities

**Proof** [ of Theorem 10] We begin by describing the required properties of the error correcting code. For every integer $t$ and $m$ that is a power of two, there exists a binary BCH code that maps a binary string $x$ of length $m-1-(t-1)\log m$ to a binary string $z = x \cdot e(x)$ of length $m$ and has distance of $2t$. In particular, if we denote the length of message $x$ by $n$ then for any $k$ we can obtain a code with a codeword of length $m \leq n + k\log n$ and distance $2k+1$.

Given a function $f : \{-1,1\}^n \to \{-1,1\}$ we define a function $f_e : \{-1,1\}^m \to \{-1,0,1\}$ as follows: $f_e(z) = f(x)$ if $z = x \cdot e(x)$ for some $x \in \{-1,1\}^n$ and $f_e(z) = 0$ otherwise. Here the value 0 is interpreted as function being equal to a random and independent coin flip.

We first note the following properties of this embedding.

- For every function $g : \{-1,1\}^n \to \{-1,1\}$,

$$\mathbb{E}_{x \cdot y \sim U_m}[f_e(x \cdot y)g(x)] = 2^{n-m}\mathbb{E}_{x \sim U}[f(x)g(x)] \ .$$

- For every function $h : \{-1,1\}^m \to \{-1,1\}$,

$$\mathbb{E}_{z \sim U_m}[f_e(z)h(z)] = 2^{n-m}\mathbb{E}_{x \sim U_n}[f(x)h(x \cdot e(x))] \ .$$

We can now describe the reduction from agnostic learning with $k$-local MQs to learning from random examples alone. Let $C$ be a concept class, let $f$ denote an unknown target function and $\epsilon$ denote the error parameter.

The main idea is to simulate random examples and $k$-local queries to $f_e$ using random examples alone of $f$. The simulation requires observing that points in $\{-1,1\}^m$ can be split into a set $Z$ which includes all codewords together with Hamming balls of radius $t$ around them and the rest of points (which we denote by $\bar{Z}$). We simulate a random example of $f_e(x)$ as follows:

1. Flip a coin with probability of heads equal to $\beta = |Z|/2^m$.

2. If the outcome is 1: ask for a random example and denote it by $(x, \ell)$. Choose a random point $z'$ in the Hamming ball of radius $k$ around $x \cdot e(x)$. If $z' = x \cdot e(x)$ then return the example $(z', \ell)$ otherwise return $(z', b)$ where $b$ is a random coin flip.

3. If the outcome is 0: sample a random point in $\bar{Z}$ and output $(z, b)$ where $b$ is a random coin flip. One can sample randomly from $\bar{Z}$ as follows: sample a random point in $z$ in $\{-1,1\}^m$, use a decoding algorithm for the BCH code to obtain a message $x$. If the decoding algorithm failed, that is $x \cdot e(x)$ is not within distance $k$ of $z$ then return $z$. Otherwise, try again.

It is important to note that the expected number of tries of this algorithm is $2^m/|\bar{Z}| = 2^m/(2^m - 2^n\beta_{m,k}) = 1/(1 - 2^{n-m}\beta_{m,k})$, where $\beta_{m,k}$ denotes the size of the Hamming ball of radius $k$. We can always assume that $2^{n-m}\beta_{m,k} \leq 2/3$ by for example adding 1 to $m$ (this does not affect the code, increases the term $2^{m-n}$ by 2 and $\beta_{m,k}$ by at most $(1 + k/(m+1))$). Therefore, with high probability, the simulation step will not require more than a logarithmic number of tries. Note that the BCH code we chose is efficiently decodable from up to $k$ errors (Massey, 1969).

Now we simulate a $k$-local MQ $z$ as follows. If $z$ is $k$-close to random example we generated in $\bar{Z}$ we return a random coin flip since there are no non-zero values of $f_e$ within distance $k$ of any point in $\bar{Z}$. If $z$ is $k$-close to random example we generated in $Z$ then it can only be $k$-close to $x \cdot e(x)$ in the Hamming ball of which $z$ was generated and for which we have an example $(x, \ell)$. This means that we can easily answer this $\mathsf{MQ}$: if $z = x \cdot e(x)$ then we return label $\ell$, otherwise a random coin flip.

On these simulated examples we run the agnostic learning algorithm $\mathcal{A}$ for $C$ on $\{-1,1\}^m$ with $\epsilon' = \epsilon \cdot 2^{n-m} \geq \epsilon / n^k$. Let $h$ denote the hypothesis returned by the algorithm.

Let $c^* = argmax_{c \in C}\{\mathbb{E}_U[f(x)c(x)]\}$ and let $\Delta = \mathbb{E}_U[f(x)c^*(x)]$. Then we know that

$$\mathbb{E}_{x \cdot y \sim U_m}[f_e(x \cdot y)c^*(x)] = 2^{n-m}\Delta.$$

By the agnostic guarantee of $\mathcal{A}$, we know that

$$\mathbb{E}_{z \sim U_m}[f_e(z)h(z)] \geq 2^{n-m}\Delta + \epsilon' = 2^{n-m}(\Delta + \epsilon)\ .$$

Now, again by the properties of the embedding,

$$\mathbb{E}_{x \sim U_n}[f(x)h(x \cdot e(x))] = 2^{m-n}\mathbb{E}_{z \sim U_m}[f_e(z)h(z)] \geq \Delta + \epsilon\ .$$

Hence $h'(x) = h(x \cdot e(x))$ is a valid hypothesis for agnostic learning of $C$.

Finally the running time of this simulation is $poly(n) \cdot T(n, n^k/\epsilon)$ where $T(n, n^k/\epsilon)$ is the running time of $\mathcal{A}$. In particular, if $T$ is polynomial then for a constant $k$ this simulation takes polynomial time. ■

## Appendix F. Separation Results

In this section, we show that $\mathsf{PAC}$+$r$-local $\mathsf{MQ}$ model is strictly more powerful than the $\mathsf{PAC}$ model, assuming that one-way functions exist. In the following discussion we show that even 1-local membership queries are more powerful than the standard $\mathsf{PAC}$ setting.

In this section, we assume that we are working with the domain $\{0,1\}^n$, rather than $\{-1,1\}^n$. Let $\mathcal{F}_n = \{f_s : \{0,1\}^n \to \{0,1\}\}_{s \in \{0,1\}^n}$ be a pseudo-random family of functions. It is well-known that such families can be constructed under the assumptions that one-way functions exist (Goldreich et al., 1986). Let $A_1, \ldots, A_n$ be a partition of $\{0,1\}^n$ that is easily computable. For example, if the strings in $\{0,1\}^n$ are lexicographically ordered, then $A_i$ contains strings with rank in the range $[(i-1)2^n/n, i2^n/n)$. For an $n+1$ bit string $x$, $x_{-1}$ denotes the $n$-length suffix of $x$. Then, for some string $s$, define the function $g_s : \{0,1\}^{n+1} \to \{0,1\}$ as follows:

$$g_s(x) = \begin{cases} f_s(x_{-1}) & \text{If } x_1 = 0 \\ f_s(x_{-1}) \oplus s_i & \text{If } x_1 = 1 \text{ and } x_{-1} \in A_i \end{cases}$$

Define $\mathcal{G}_{n+1} = \{g_s : \{0,1\}^{n+1} \to \{0,1\}\}_{s \in \{0,1\}^n}$. We show below that the class $\mathcal{G}_{n+1}$ is not learnable in the $\mathsf{PAC}$ setting, but is learnable in the $\mathsf{PAC}$+1-local $\mathsf{MQ}$ model under the uniform distribution.

**Theorem 31** *Assuming that one-way functions exist, the class $\mathcal{G}_{n+1}$ is not learnable in the* $\mathsf{PAC}$ *model, but is learnable in the* $\mathsf{PAC}$+1*-local* $\mathsf{MQ}$ *model, under the uniform distribution.*

**Proof** First, we show that $\mathcal{G}_{n+1}$ is learnable in the PAC+1-local MQ model. Let $1x$ and $0x$ be the two strings of length $n+1$, with suffix $x \in \{0,1\}^n$. Then for any $g_s \in \mathcal{G}_{n+1}$, $g_s(1x) \oplus g_s(0x) = s_i$, if $x \in A_i$. Thus, drawing a random example from $U$ and making a one local query reveals one bit of the string $s$. By drawing $O(n \log(n))$ random examples, all the bits of the string $s$ can be recovered with high probability. Thus, revealing the function $g_s$ itself.

On the other hand, in the PAC model, the probability that seeing two examples $1x$ and $0x$ is exponentially small. Thus, all the labels appear perfectly random (since $f_s$ is from a pseudorandom family). Thus, no learning is possible in the PAC model. ■

In fact, the above construction also shows that the random walk learning model (Bshouty et al., 2005) is also more powerful than the PAC learning setting, assuming that one way function exist. Bshouty et al. (2005) had already shown that the random walk model is provably weaker than the full MQ model assuming that one-way functions exist. In fact, essentially the same argument also shows that full MQ is more powerful than PAC+$o(n)$-local MQ. The following simple concept class (which is the same as that of Bshouty et al.) shows the necessary separation.

Let $e^i$ be the vector that has 1 in the $i^{th}$ position, and 0s elsewhere. Again, let $\mathcal{F}_n = \{f_s : \{0,1\}^n \rightarrow \{0,1\}\}_{s \in \{0,1\}^n}$ be the pseudorandom family of functions. Then define, $\mathcal{G}'_n = \{g_s\}$ as follows:

$$g_s(x) = \begin{cases} s_i & \text{If } x = e^i \\ f_s(x) & \text{Otherwise} \end{cases}$$

**Theorem 32** *The concept class $\mathcal{G}'_n$ is learnable in the full* MQ *model, but not in* PAC+$o(n)$-*local* MQ *model under the uniform distribution.*

**Proof** It is easy to see that by making membership queries to the points, $e^1, \ldots, e^n$, the string $s$ is revealed and hence also the function $g_s$. On the other hand, random points from the Boolean cube have Hamming weight $\Omega(n)$, except with exponentially small probability. Thus, $o(n)$-local MQs are of no use to query the points $e^i$. The labels for any point obtained from the distribution, or using $o(n)$-local MQs are essentially random. Hence, $\mathcal{G}'_n$ is not learnable in the PAC+$o(n)$-local MQ model. ■