Learning Unions of $\omega(1)$-Dimensional Rectangles

Alp Atıcı[1] and Rocco A. Servedio[⋆]

Columbia University, New York, NY, USA
{atici@math,rocco@cs}.columbia.edu

Abstract. We consider the problem of learning unions of rectangles over the domain $[b]^n$, in the uniform distribution membership query learning setting, where both b and n are "large". We obtain poly$(n, \log b)$-time algorithms for the following classes:

- poly$(n \log b)$-Majority of $O(\frac{\log(n \log b)}{\log\log(n \log b)})$-dimensional rectangles.
- Union of poly$(\log(n \log b))$ $O(\frac{\log^2(n \log b)}{(\log\log(n \log b) \log\log\log(n \log b))^2})$-dimensional rectangles.
- poly$(n \log b)$-Majority of poly$(n \log b)$-Or of disjoint $O(\frac{\log(n \log b)}{\log\log(n \log b)})$-dimensional rectangles.

Our main algorithmic tool is an extension of Jackson's boosting- and Fourier-based Harmonic Sieve algorithm [12] to the domain $[b]^n$, building on work of Akavia *et al.* [1]. Other ingredients used to obtain the results stated above are techniques from exact learning [3] and ideas from recent work on learning augmented AC^0 circuits [13] and on representing Boolean functions as thresholds of parities [15].

1 Introduction

Motivation. The learnability of Boolean valued functions defined over the domain $[b]^n = \{0, 1, \ldots, b-1\}^n$ has long elicited interest in computational learning theory literature. In particular, much research has been done on learning various classes of "unions of rectangles" over $[b]^n$ (see e.g. [3, 5, 6, 9, 12, 18]), where a rectangle is a conjunction of properties of the form "the value of attribute x_i lies in the range $[\alpha_i, \beta_i]$". One motivation for studying these classes is that they are a natural analogue of classes of DNF (Disjunctive Normal Form) formulae over $\{0, 1\}^n$; for instance, it is easy to see that in the case $b = 2$ any union of s rectangles is simply a DNF with s terms.

Since the description length of a point $x \in [b]^n$ is $n \log b$ bits, a natural goal in learning functions over $[b]^n$ is to obtain algorithms which run in time poly$(n \log b)$. Throughout the paper we refer to such algorithms with poly$(n \log b)$ runtime as *efficient* algorithms. In this paper we give efficient algorithms which can learn several interesting classes of unions of rectangles over $[b]^n$ in the model of uniform distribution learning with membership queries.

Previous results. In a breakthrough result a decade ago, Jackson [12] gave the Harmonic Sieve (HS) algorithm and proved that it can learn any s-term DNF formula over n Boolean variables in poly(n, s) time. In fact, Jackson showed

[⋆] Supported in part by NSF award CCF-0347282 and NSF award CCF-0523664.

that the algorithm can learn any s-way majority of parities in $\mathrm{poly}(n, s)$ time; this is a richer set of functions which includes all s-term DNF formulae. The HS algorithm works by boosting a Fourier-based weak learning algorithm, which is a modified version of an earlier algorithm due to Kushilevitz and Mansour [17].

In [12] Jackson also described an extension of the HS algorithm to the domain $[b]^n$. His main result for $[b]^n$ is an algorithm that can learn any union of s rectangles over $[b]^n$ in $\mathrm{poly}(s^{b \log \log b}, n)$ time; note that this runtime is $\mathrm{poly}(n, s)$ if and only if b is $\Theta(1)$ (and the runtime is clearly exponential in b for any s).

There has also been substantial work on learning various classes of unions of rectangles over $[b]^n$ in the more demanding model of exact learning from membership and equivalence queries. Some of the subclasses of unions of rectangles which have been considered in this setting are

The dimension of each rectangle is $O(1)$: Beimel and Kushilevitz [3] give an algorithm learning any union of s $O(1)$-dimensional rectangles over $[b]^n$ in $\mathrm{poly}(n, s, \log b)$ time steps, using equivalence queries only.

The number of rectangles is limited: In [3] an algorithm is also given which exactly learns any union of $O(\log n)$ many rectangles in $\mathrm{poly}(n, \log b)$ time using membership and equivalence queries. Earlier, Maass and Warmuth [18] gave an algorithm which uses only equivalence queries and can learn any union of $O(1)$ rectangles in $\mathrm{poly}(n, \log b)$ time.

The rectangles are disjoint: If no input $x \in [b]^n$ belongs to more than one rectangle, then [3] can learn a union of s such rectangles in $\mathrm{poly}(n, s, \log b)$ time with membership and equivalence queries.

Our techniques and results. Because efficient learnability is established for union of $O(\log n)$ arbitrary dimensional rectangles by [3] in a more demanding model, we are interested in achieving positive results when the number of rectangles is strictly larger. Therefore all the cases we study involve at least $\mathrm{poly}(\log(n \log b))$ and sometimes as many as $\mathrm{poly}(n \log b)$ rectangles.

We start by describing a new variant of the Harmonic Sieve algorithm for learning functions defined over $[b]^n$; we call this new algorithm the Generalized Harmonic Sieve, or GHS. The key difference between GHS and Jackson's algorithm for $[b]^n$ is that whereas Jackson's algorithm used a weak learning algorithm whose runtime is $\mathrm{poly}(b)$, the GHS algorithm uses a $\mathrm{poly}(\log b)$ time weak learning algorithm described in recent work of Akavia *et al.* [1].

We then apply GHS to learn various classes of functions defined in terms of "b-literals" (see Section 2 for a precise definition; roughly speaking a b-literal is like a 1-dimensional rectangle). We first show the following result:

Theorem 1. *The concept class of s-*MAJORITY *of r-*PARITY *of b-literals where $s = \mathrm{poly}(n \log b)$, $r = O(\frac{\log(n \log b)}{\log \log(n \log b)})$ is efficiently learnable using* GHS.

Learning this class has immediate applications for our goal of "learning unions of rectangles"; in particular, it follows that

Theorem 2. *The concept class of s-*MAJORITY *of r-rectangles where $s = \mathrm{poly}(n \log b)$, $r = O(\frac{\log(n \log b)}{\log \log(n \log b)})$ is efficiently learnable using* GHS.

This clearly implies efficient learnability for unions (as opposed to majorities) of s such rectangles as well.

We then employ a technique of restricting the domain $[b]^n$ to a much smaller set and adaptively expanding this set as required. This approach was used in the exact learning framework by Beimel and Kushilevitz [3]; by an appropriate modification we adapt the underlying idea to the uniform distribution membership query framework. Using this approach in conjunction with GHS we obtain almost a quadratic improvement in the dimension of the rectangles if the number of terms is guaranteed to be small:

Theorem 3. *The concept class of unions of $s = \mathrm{poly}(\log(n \log b))$ many r-rectangles where $r = O(\frac{\log^2(n\log b)}{(\log\log(n\log b)\log\log\log(n\log b))^2})$ is efficiently learnable via Algorithm 2 (see Section 5).*

Finally we consider the case of disjoint rectangles (also studied by [3] as mentioned above), and improve the depth of our circuits by 1 provided that the rectangles connected to the same Or gate are disjoint:

Corollary 1. *The concept class of s-Majority of t-Or of disjoint r-rectangles where $s, t = \mathrm{poly}(n \log b)$, $r = O(\frac{\log(n\log b)}{\log\log(n\log b)})$ is efficiently learnable under GHS.*

Organization. In Section 3 we describe the Generalized Harmonic Sieve algorithm GHS which will be our main tool for learning unions of rectangles. In Section 4 we show that s-Majority of r-Parity of b-literals is efficiently learnable using GHS for suitable r, s; this concept class turns out to be quite useful for learning unions of rectangles. In Section 5 we improve over the results of Section 4 slightly if the number of terms is small, by adaptively selecting a small subset of $[b]$ in each dimension which is sufficient for learning, and invoke GHS over the restricted domain. In Section 6 we explore the consequences of the results in Sections 4 and 5 for the ultimate goal of learning unions of rectangles.

2 Preliminaries

The learning model. We are interested in Boolean functions defined over the domain $[b]^n$, where $[b] = \{0, 1, \ldots, b-1\}$. We view Boolean functions as mapping into $\{-1, 1\}$ where -1 is associated with True and 1 with False.

A *concept class* $\mathfrak{C}$ is a collection of classes (sets) of Boolean functions $\{C_{n,b} \colon n > 0, b > 1\}$ such that if $f \in C_{n,b}$ then $f \colon [b]^n \to \{-1, 1\}$. Throughout this paper we view both n and b as asymptotic parameters, and our goal is to exhibit algorithms that learn various classes $C_{n,b}$ in $\mathrm{poly}(n, \log b)$ time. We now describe the uniform distribution membership query learning model that we will consider.

A *membership oracle* $\mathsf{MEM}(f)$ is an oracle which, when queried with input x, outputs the label $f(x)$ assigned by the target f to the input. Let $f \in C_{n,b}$ be an unknown member of the concept class and let $\mathcal{A}$ be a randomized learning algorithm which takes as input accuracy and confidence parameters ϵ, δ and can invoke $\mathsf{MEM}(f)$. We say that $\mathcal{A}$ *learns* $\mathfrak{C}$ *under the uniform distribution on* $[b]^n$ provided that given any $0 < \epsilon, \delta < 1$ and access to $\mathsf{MEM}(f)$, with probability at least $1-\delta$ $\mathcal{A}$ outputs an ϵ-approximating hypothesis $h\colon [b]^n \to \{-1,1\}$ (which need not belong to $\mathfrak{C}$) such that $\Pr_{x\in[b]^n}[f(x)=h(x)] \geq 1-\epsilon$.

We are interested in computationally efficient learning algorithms. We say that $\mathcal{A}$ *learns* $\mathfrak{C}$ *efficiently* if for any target concept $f \in C_{n,b}$,

- $\mathcal{A}$ runs for at most $\mathrm{poly}(n, \log b, 1/\epsilon, \log 1/\delta)$ steps;
- Any hypothesis h that $\mathcal{A}$ produces can be evaluated on any $x \in [b]^n$ in at most $\mathrm{poly}(n, \log b, 1/\epsilon, \log 1/\delta)$ time steps.

The functions we study. The reader might wonder which classes of Boolean valued functions over $[b]^n$ are interesting. In this article we study classes of functions that are defined in terms of "b-literals"; these include rectangles and unions of rectangles over $[b]^n$ as well as other richer classes. As described below, b-literals are a natural extension of Boolean literals to the domain $[b]^n$.

Definition 1. *A function $\ell\colon [b] \to \{-1,1\}$ is a* basic b-literal *if for some $\sigma \in \{-1,1\}$ and some $\alpha \leq \beta$ with $\alpha, \beta \in [b]$ we have $\ell(x) = \sigma$ if $\alpha \leq x \leq \beta$, and $\ell(x) = -\sigma$ otherwise. A function $\ell\colon [b] \to \{-1,1\}$ is a* b-literal *if there exists a basic b-literal ℓ' and some fixed $z \in [b]$, $\gcd(z,b)=1$ such that for all $x \in [b]$ we have $\ell(x) = \ell'(xz)$.*

Basic b-literals are the most natural extension of Boolean literals to the domain $[b]^n$. General b-literals (not necessarily basic) were previously studied in [1] and are also quite natural; for example, if b is odd then the *least significant bit* function $lsb(x)\colon [b] \to \{-1,1\}$ (defined by $lsb(x) = -1$ iff x is even) is a b-literal.

Definition 2. *A function $f\colon [b]^n \to \{-1,1\}$ is a* k-rectangle *if it is an* AND *of k basic b-literals $\ell_1, \ldots, \ell_k$ over k distinct variables $x_{i_1}, \ldots, x_{i_k}$. If f is a k-rectangle for some k then we may simply say that f is a* rectangle. *A* union of s rectangles *$R_1, \ldots, R_s$ is a function of the form $f(x) = \mathrm{OR}_{i=1}^{s} R_i(x)$.*

The class of unions of s rectangles over $[b]^n$ is a natural generalization of the class of s-term DNF over $\{0,1\}^n$. Similarly MAJORITY of PARITY of basic b-literals generalizes the class of MAJORITY of PARITY of Boolean literals, a class which has been the subject of much research (see e.g. [12, 4, 15]).

If G is a logic gate with potentially unbounded fan-in (e.g. MAJORITY, PARITY, AND, etc.) we write "s-G" to indicate that the fan-in of G is restricted to be at most s. Thus, for example, an "s-MAJORITY of r-PARITY of b-literals" is a MAJORITY of at most s functions $g_1, \ldots, g_s$, each of which is a PARITY of at most r many b-literals. We will further assume that *any two b-literals which are inputs to the same gate depend on different variables.* This is a natural restriction to impose in light of our ultimate goal of learning unions of rectangles. Although our results hold without this assumption, it provides simplicity in the presentation.

The Fourier transform. We will make use of the Fourier expansion of complex valued functions over $[b]^n$.

Consider $f, g\colon [b]^n \to \mathbb{C}$ endowed with the inner product $\langle f, g\rangle = \mathbf{E}[f\overline{g}]$ and induced norm $\|f\| = \sqrt{\langle f, f\rangle}$. Let $\omega_b = e^{\frac{2\pi i}{b}}$ and for each $\alpha = (\alpha_1, \ldots, \alpha_n) \in [b]^n$, let $\chi_\alpha \colon [b]^n \to \mathbb{C}$ be defined as $\chi_\alpha(x) = \omega_b^{\alpha_1 x_1 + \cdots + \alpha_n x_n}$. Let $\mathcal{B}$ denote the set of functions $\mathcal{B} = \{\chi_\alpha \colon \alpha \in [b]^n\}$. It is easy to verify the following properties:

- For each $\alpha = (\alpha_1, \ldots, \alpha_n) \in [b]^n$, we have $\|\chi_\alpha\| = 1$.
- Elements in $\mathcal{B}$ are orthogonal: For $\alpha, \beta \in [b]^n$, we have $\langle \chi_\alpha, \chi_\beta \rangle = \begin{cases} 1 \text{ if } \alpha = \beta \\ 0 \text{ if } \alpha \neq \beta \end{cases}$.
- $\mathcal{B}$ constitutes an orthonormal basis for all functions $\{f \colon [b]^n \to \mathbb{C}\}$ considered as a vector space over $\mathbb{C}$, so every $f \colon [b]^n \to \mathbb{C}$ can be expressed uniquely as $f(x) = \sum_\alpha \hat{f}(\alpha)\chi_\alpha(x)$.

The values $\{\hat{f}(\alpha) \colon \alpha \in [b]^n\}$ are called the *Fourier coefficients* or the *Fourier transform* of f. As is well known, *Parseval's Identity* relates the values of the coefficients to the values of the function:

Lemma 1 (Parseval's Identity). $\sum_\alpha |\hat{f}(\alpha)|^2 = \mathbf{E}[|f|^2]$ *for any* $f \colon [b]^n \to \mathbb{C}$.

We write $L_1(f)$ to denote $\sum_\alpha |\hat{f}(\alpha)|$.

Additional tools: weak hypotheses and boosting. Let $f \colon [b]^n \to \{-1, 1\}$ and $\mathcal{D}$ be a probability distribution over $[b]^n$. A function $g \colon [b]^n \to \mathbb{R}$ is said to be a *weak hypothesis for f with advantage γ under $\mathcal{D}$* if $\mathbf{E}_{\mathcal{D}}[fg] \geq \gamma$.

The first *boosting* algorithm was described by Schapire [19] in 1990; since then boosting has been intensively studied (see [8] for an overview). The basic idea is that by combining a sequence of weak hypotheses $h_1, h_2, \ldots$ (the i-th of which has advantage γ with respect to a carefully chosen distribution $\mathcal{D}_i$) it is possible to obtain a high accuracy final hypothesis h which satisfies $\Pr[h(x) = f(x)] \geq 1-\epsilon$. The following theorem gives a precise statement of the performance guarantees of a particular boosting algorithm, which we call Algorithm $\mathcal{B}$, due to Freund. Many similar statements are now known about a range of different boosting algorithms but this is sufficient for our purposes.

Theorem 4 (Boosting Algorithm [7]). *Suppose that Algorithm $\mathcal{B}$ is given:*

- $0 < \epsilon, \delta < 1$, *and membership query access* $\mathsf{MEM}(f)$ *to* $f \colon [b]^n \to \{-1, 1\}$*;*
- *access to an algorithm* `WL` *which has the following property: given a value δ' and access to* $\mathsf{MEM}(f)$ *and to* $\mathsf{EX}(f, \mathcal{D})$ *(the latter is an example oracle which generates random examples from $[b]^n$ drawn with respect to distribution $\mathcal{D}$), it constructs a weak hypothesis for f with advantage γ under $\mathcal{D}$ with probability at least $1 - \delta'$ in time polynomial in n, $\log b$, $\log(1/\delta')$.*

Then Algorithm $\mathcal{B}$ behaves as follows:

- *It runs for $S = O(\log(1/\epsilon)/\gamma^2)$ stages and runs in total time polynomial in n, $\log b$, ϵ^{-1}, γ^{-1}, $\log(\delta^{-1})$.*
- *At each stage $1 \leq j \leq S$ it constructs a distribution $\mathcal{D}_j$ such that $L_\infty(\mathcal{D}_j) < \mathrm{poly}(\epsilon^{-1})/b^n$, and simulates $\mathsf{EX}(f, \mathcal{D}_j)$ for* `WL` *in stage j. Moreover, there is a "pseudo-distribution" $\tilde{\mathcal{D}}_j$ satisfying $\tilde{\mathcal{D}}_j(x) = c\mathcal{D}_j(x)$ for all x (where $c \in [1/2, 3/2]$ is some fixed value) such that $\tilde{\mathcal{D}}_j(x)$ can be computed in time polynomial in $n \log b$ for each $x \in [b]^n$.*

– *It outputs a final hypothesis* $h = \text{sign}(h_1+h_2+\ldots+h_S)$ *which* ϵ*-approximates* f *under the uniform distribution with probability* $1-\delta$*; here* h_j *is the output of* `WL` *at stage* j *invoked with simulated access to* $\mathsf{EX}(f, \mathcal{D}_j)$.

We will sometimes informally refer to distributions $\mathcal{D}$ which satisfy the bound $L_\infty(\mathcal{D}) < \frac{\text{poly}(\epsilon^{-1})}{b^n}$ as *smooth* distributions.

In order to use boosting, it must be the case that there exists a suitable weak hypothesis with advantage γ. The "discriminator lemma" of Hajnal *et al.* [10] can often be used to assert that the desired weak hypothesis exists:

Lemma 2 (The Discriminator Lemma [10]). *Let* $\mathfrak{H}$ *be a class of* ± 1*-valued functions over* $[b]^n$ *and let* $f : [b]^n \to \{-1, 1\}$ *be expressible as* $f = \text{MAJORITY}(h_1, \ldots, h_s)$ *where each* $h_i \in \mathfrak{H}$ *and* $h_1(x) + \ldots + h_s(x) \neq 0$ *for all* x*. Then for any distribution* $\mathcal{D}$ *over* $[b]^n$ *there is some* h_i *such that* $|\mathbf{E}_{\mathcal{D}}[fh_i]| \geq 1/s$.

3 The Generalized Harmonic Sieve Algorithm

In this section our goal is to describe a variant of Jackson's Harmonic Sieve Algorithm and show that under suitable conditions it can efficiently learn certain functions $f : [b]^n \to \{-1, 1\}$. As mentioned earlier, our aim is to attain $\text{poly}(\log b)$ runtime dependence on b and consequently obtain efficient algorithms as described in Section 2. This goal precludes using Jackson's original Harmonic Sieve variant for $[b]^n$ since the runtime of his weak learner depends polynomially rather than polylogarithmically on b (see [12, Lemma 15]).

As we describe below, this $\text{poly}(\log b)$ runtime can be achieved by modifying the Harmonic Sieve over $[b]^n$ to use a weak learner due to Akavia *et al.* [1] which is more efficient than Jackson's weak learner. We shall call the resulting algorithm "The Generalized Harmonic Sieve" algorithm, or GHS for short.

Recall that in the Harmonic Sieve over the Boolean domain $\{-1, 1\}^n$, the weak hypotheses used are simply the Fourier basis elements over $\{-1, 1\}^n$, which correspond to the Boolean-valued parity functions. For $[b]^n$, we will use the real component of the complex-valued Fourier basis elements $\{\chi_\alpha, \alpha \in [b]^n\}$ as our weak hypotheses.

The following theorem of Akavia *et al.* [1, Theorem 5] will play a crucial role towards construction of the GHS algorithm.

Theorem 5 (See [1]). *There is a learning algorithm that, given membership query access to* $f : [b]^n \to \mathbb{C}$, $0 < \gamma$ *and* $0 < \delta < 1$*, outputs a list* L *of indices such that with probability at least* $1 - \delta$*, we have* $\{\alpha : |\hat{f}(\alpha)| > \gamma\} \subseteq L$ *and* $|\hat{f}(\beta)| \geq \frac{\gamma}{2}$ *for every* $\beta \in L$*. The running time of the algorithm is polynomial in* n*,* $\log b$*,* $\|f\|_\infty$*,* γ^{-1}*,* $\log(\delta^{-1})$.

Lemma 3 (Construction of the weak hypothesis). *Given*

– *Membership query access* $\mathsf{MEM}(f)$ *to* $f : [b]^n \to \{-1, 1\}$*;*
– *A smooth distribution* $\mathcal{D}$*; more precisely, access to an algorithm computing* $\tilde{\mathcal{D}}(x)$ *in time polynomial in* n*,* $\log b$ *for each* $x \in [b]^n$*. Here* $\tilde{\mathcal{D}}$ *is a "pseudo-distribution" for* $\mathcal{D}$ *as in Theorem 4, i.e. there is a value* $c \in [1/2, 3/2]$ *such that* $\tilde{\mathcal{D}}(x) = c\mathcal{D}(x)$ *for all* x.

– *A value* $0 < \gamma < 1/2$ *such that there exists a Fourier basis element* χ_β *satisfying* $|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\beta}]| > \gamma$.

There is an algorithm that outputs a weak hypothesis for f *with advantage* $\gamma/2$ *under* $\mathcal{D}$ *with probability* $1-\delta$ *and runs in time polynomial in* n, $\log b$, ϵ^{-1}, γ^{-1}, $\log(\delta^{-1})$.

Proof. Let $f_*(x) = b^n\tilde{\mathcal{D}}(x)f(x)$. Observe that

– Since $\mathcal{D}$ is smooth, $\|f_*\|_\infty < \mathrm{poly}(\epsilon^{-1})$.
– For any $\alpha \in [b]^n$, $\hat{f}_*(\alpha) = \mathbf{E}[f_*\overline{\chi_\alpha}] = \frac{1}{b^n}\sum_{x\in[b]^n} b^n\tilde{\mathcal{D}}(x)f(x)\overline{\chi_\alpha(x)} = \mathbf{E}_{\mathcal{D}}[cf\overline{\chi_\alpha}]$.

Therefore one can invoke the algorithm of Theorem 5 over $f_*(x)$ by simulating $\mathsf{MEM}(f_*)$ via $\mathsf{MEM}(f)$, each time with $\mathrm{poly}(n, \log b)$ time overhead, and obtain a list L of indices as in Theorem 5. It is easy to see that the algorithm runs in the desired time bound and outputs a nonempty list L. Let β be any element of L. Because $\hat{f}_*(\beta) = \mathbf{E}[b^n\tilde{\mathcal{D}}(x)f(x)\overline{\chi_\beta(x)}]$, one can approximate $\frac{\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\beta}]}{|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\beta}]|} = \frac{\hat{f}_*(\beta)}{|\hat{f}_*(\beta)|} = e^{i\theta}$ using uniformly drawn random examples. Let $e^{i\theta'}$ be the approximation thus obtained. Note that by assumption we have:

– $|\hat{f}_*(\beta)| > c\gamma$.
– For random $x \in [b]^n$, the random variable $(b^n\tilde{\mathcal{D}}(x)f(x)\overline{\chi_\beta(x)})$ always takes a value whose magnitude is $O(\mathrm{poly}(\epsilon^{-1}))$ in absolute value.

Using a straightforward Chernoff bound argument, this implies that $|\theta - \theta'|$ can be made smaller than any constant using $\mathrm{poly}(n, \log b, \epsilon^{-1})$ time and random examples.

Now note that we have

$$\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\beta}] = e^{i\theta}|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\beta}]| \Rightarrow \mathbf{E}_{\mathcal{D}}[f\overline{e^{i\theta}\chi_\beta}] = |\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\beta}]| > \gamma.$$

Therefore for a sufficiently small value of $|\theta - \theta'|$, we have

$$\mathbf{E}_{\mathcal{D}}[f\Re\{\overline{e^{i\theta'}\chi_\beta}\}] = \Re\{\mathbf{E}_{\mathcal{D}}[f\overline{e^{i\theta'}\chi_\beta}]\} = \Re\{e^{i(\theta-\theta')}\quad \underbrace{\mathbf{E}_{\mathcal{D}}[f\overline{e^{i\theta}\chi_\beta}]}_{\text{real valued and } > \gamma}\quad\} \geq \gamma/2.$$

We conclude that $\Re\{\overline{e^{i\theta'}\chi_\beta}\}$ constitutes a weak hypothesis for f with advantage $\gamma/2$ under $\mathcal{D}$ with high probability. □

Rephrasing the statement of Lemma 3, now we know: As long as for any function f in the concept class it is guaranteed that under any smooth distribution $\mathcal{D}$ there is a Fourier basis element χ_β that has nonnegligible correlation with f (i.e. $|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| > \gamma$), then it is possible to efficiently identify and use such a Fourier basis element to construct a weak hypothesis.

Now as in Jackson's original Harmonic Sieve, one can invoke Algorithm $\mathcal{B}$ from Theorem 4: At stage j, we have a distribution $\mathcal{D}_j$ over $[b]^n$ for which $L_\infty(\mathcal{D}_j) < \mathrm{poly}(\epsilon^{-1})/b^n$. Thus one can pass the values of $\mathcal{D}_j$ to the algorithm in Lemma 3 and use this algorithm as `WL` in Algorithm $\mathcal{B}$ to obtain the weak hypothesis at each stage. Repeating this idea for every stage and combining the weak hypotheses generated for all the stages as described by Theorem 4, we have the GHS algorithm:

Corollary 2 (The Generalized Harmonic Sieve). *Let $\mathfrak{C}$ be a concept class. Suppose that for any concept $f \in C_{n,b}$ and any distribution $\mathcal{D}$ over $[b]^n$ with $L_\infty(\mathcal{D}) < \text{poly}(\epsilon^{-1})/b^n$ there exists a Fourier basis element χ_α such that $|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| \geq \gamma$. Then $\mathfrak{C}$ can be learned in time $\text{poly}(n, \log b, \epsilon^{-1}, \gamma^{-1})$.*

4 Learning Majority of Parity using GHS

In this section we identify classes of functions which can be learned efficiently using the GHS algorithm and prove Theorem 1.

To prove Theorem 1, we show that for any concept $f \in \mathfrak{C}$ and under any smooth distribution there must be some Fourier basis element which has high correlation with f; this is the essential step which lets us apply the Generalized Harmonic Sieve. We prove this in Section 4.2. In Section 4.3 we give an alternate argument which yields a Theorem 1 analogue but with a slightly different bound on r, namely $r = O(\frac{\log(n \log b)}{\log\log b})$.

4.1 Setting the stage

For ease of notation we will write $abs(\alpha)$ to denote $\min\{\alpha, b - \alpha\}$. We will use the following simple lemma from [1]:

Lemma 4 (See [1]). *For all $0 \leq \ell \leq b$, we have $|\sum_{y=0}^{\ell-1} \omega_b^{\alpha y}| < b/abs(\alpha)$.*

Corollary 3. *Let $f\colon [b] \to \{-1, 1\}$ be a basic b-literal. Then if $\alpha = 0$, $|\hat{f}(\alpha)| < 1$, while if $\alpha \neq 0$, $|\hat{f}(\alpha)| < O(\frac{1}{abs(\alpha)})$.*

Proof. The first inequality follows immediately from Lemma 1 (Parseval's Identity) because f is ± 1-valued. For the latter, note that $|\hat{f}(\alpha)| = |\mathbf{E}[f\overline{\chi_\alpha}]| =$

$$\frac{1}{b}\left|\sum_{x \in f^{-1}(1)} \chi_\alpha(x) - \sum_{x \in f^{-1}(-1)} \chi_\alpha(x)\right| \leq \frac{1}{b}\left|\sum_{x \in f^{-1}(1)} \chi_\alpha(x)\right| + \frac{1}{b}\left|\sum_{x \in f^{-1}(-1)} \chi_\alpha(x)\right|$$

where the inequality is simply the triangle inequality. It is easy to see that each of the sums on the RHS above equals $\frac{1}{b}|\omega_b^{\alpha c}||\sum_{y=0}^{\ell-1} \omega_b^{\alpha y}| = \frac{1}{b}|\sum_{y=0}^{\ell-1} \omega_b^{\alpha y}|$ for some suitable c and $\ell \leq b$, and hence by Lemma 4 each sum is $O(\frac{1}{abs(\alpha)})$. This gives the desired result. □

The following easy lemma is useful for relating the Fourier transform of a b-literal to the corresponding basic b-literal:

Lemma 5. *For $f, g\colon [b] \to \mathbb{C}$ such that $g(x) = f(xz)$ where $\gcd(z, b) = 1$, we have $\hat{g}(\alpha) = \hat{f}(\alpha z^{-1})$.*

Proof.

$$\begin{aligned}\hat{g}(\alpha) &= \mathbf{E}_x[g(x)\overline{\chi_\alpha(x)}] = \mathbf{E}_x[f(xz)\overline{\chi_\alpha(x)}] = \mathbf{E}_{xz^{-1}}[f(x)\overline{\chi_\alpha(xz^{-1})}] \\ &= \mathbf{E}_{xz^{-1}}[f(x)\overline{\chi_{\alpha z^{-1}}(x)}] = \mathbf{E}_x[f(x)\overline{\chi_{\alpha z^{-1}}(x)}] = \hat{f}(\alpha z^{-1}).\end{aligned}$$

□

A natural way to approximate a b-literal is by truncating its Fourier representation. We make the following definition:

Definition 3. *Let k be a positive integer. For $f\colon [b] \to \{-1,1\}$ a basic b-literal, the k-restriction of f is $\tilde{f}\colon [b] \to \mathbb{C}$, $\tilde{f}(x) = \sum_{abs(\alpha)\le k} \hat{f}(\alpha)\chi_\alpha(x)$. More generally, for $f\colon [b] \to \{-1,1\}$ a b-literal (so $f(x) = f'(xz)$ where f' is a basic b-literal) the k-restriction of f is $\tilde{f}\colon [b] \to \mathbb{C}$, $\tilde{f}(x) = \sum_{abs(\alpha z^{-1})\le k} \hat{f}(\alpha)\chi_\alpha(x) = \sum_{abs(\beta)\le k} \widehat{f'}(\beta)\chi_\beta(x)$.*

4.2 There exist highly correlated Fourier basis elements for functions in $\mathfrak{C}$ under smooth distributions

In this section we show that given any $f \in \mathfrak{C}$ and any smooth distribution $\mathcal{D}$, some Fourier basis element must have high correlation with f. We begin by bounding the error of the k-restriction of a basic b-literal:

Lemma 6. *For $f\colon [b] \to \{-1,1\}$ a b-literal and $\tilde{f}$ the k-restriction of f, we have $\mathbf{E}[|f-\tilde{f}|^2] = O(1/k)$.*

Proof. Without loss of generality assume f to be a basic b-literal. By an immediate application of Lemma 1 (Parseval's Identity) we obtain:

$$\mathbf{E}[|f-\tilde{f}|^2] = \sum_{abs(\alpha)>k} |\hat{f}(\alpha)|^2 \underbrace{=}_{\text{by Corollary 3}} \sum_{\alpha>k} O(1)/\alpha^2 = O(1/k). \qquad \square$$

Now suppose that f is an r-Parity of b-literals $f_1,\ldots,f_r$. Since Parity corresponds to multiplication over the domain $\{-1,1\}$, this means that $f = \prod_{i=1}^r f_i$. It is natural to approximate f by the product of the k-restrictions $\prod_{i=1}^r \tilde{f}_i$. The following lemma bounds the error of this approximation:

Lemma 7. *For $i = 1,\ldots,r$, let $f_i\colon [b] \to \{-1,1\}$ be a b-literal and let $\tilde{f}_i$ be its k-restriction. Then $\mathbf{E}[|f_1(x_1)f_2(x_2)\ldots f_r(x_r) - \tilde{f}_1(x_1)\tilde{f}_2(x_2)\ldots\tilde{f}_r(x_r)|] < \frac{(O(1))^r}{\sqrt{k}}$.*

Proof. First note that by the nonnegativity of variance and Lemma 6, we have that for each $i = 1,\ldots,r$:

$$\mathbf{E}_{x_i}[|f_i(x_i) - \tilde{f}_i(x_i)|] \le \sqrt{\mathbf{E}_{x_i}[|f_i(x_i) - \tilde{f}_i(x_i)|^2]} = O(1/\sqrt{k}).$$

Therefore we also have for each $i = 1,\ldots,r$:

$$\mathbf{E}_{x_i}[|\tilde{f}_i(x_i)|] < \underbrace{\mathbf{E}_{x_i}[|\tilde{f}_i(x_i) - f_i(x_i)|]}_{<O(\frac{1}{\sqrt{k}})} + \underbrace{\mathbf{E}_{x_i}[|f_i(x_i)|]}_{=1} = O(1).$$

For any $(x_1,\ldots,x_r)$ we can bound the difference in the lemma as follows:

$$\begin{aligned}
&|f_1(x_1)\ldots f_r(x_r) - \tilde{f}_1(x_1)\ldots\tilde{f}_r(x_r)| \le \\
&|f_1(x_1)\ldots f_r(x_r) - f_1(x_1)\ldots f_{r-1}(x_{r-1})\tilde{f}_r(x_r)| + \\
&|f_1(x_1)\ldots f_{r-1}(x_{r-1})\tilde{f}_r(x_r) - \tilde{f}_1(x_1)\ldots\tilde{f}_r(x_r)| \le \\
&|f_r(x_r) - \tilde{f}_r(x_r)| + |\tilde{f}_r(x_r)||f_1(x_1)\ldots f_{r-1}(x_{r-1}) - \tilde{f}_1(x_1)\ldots\tilde{f}_{r-1}(x_{r-1})|
\end{aligned}$$

Therefore the expectation in question is at most:

$$\underbrace{\mathbf{E}_{x_r}[|f_r(x_r) - \tilde{f}_r(x_r)|]}_{=O(1/\sqrt{k})} + \underbrace{\mathbf{E}_{x_r}[|\tilde{f}_r(x_r)|]}_{=O(1)} \cdot \mathbf{E}_{(x_1,\ldots,x_{r-1})}[|f_1(x_1)\ldots f_{r-1}(x_{r-1}) - \tilde{f}_1(x_1)\ldots \tilde{f}_{r-1}(x_{r-1})|].$$

We can repeat this argument successively until the base case $\mathbf{E}_{x_1}[|f_1(x_1) - \tilde{f}_1(x_1)|] \leq O(\frac{1}{\sqrt{k}})$ is reached. Thus for some $K, L = 0(1)$, we have

$$\begin{aligned}\mathbf{E}_{(x_1,\ldots,x_r)\in[b]^r}[|f_1(x_1)\ldots f_r(x_r) - \tilde{f}_1(x_1)\ldots \tilde{f}_r(x_r)|] &\leq \frac{K + KL + KL^2 + \ldots + KL^{r-1}}{\sqrt{k}} \\ &= K\frac{L^r - 1}{(L-1)\sqrt{k}}\end{aligned}$$

from which the lemma follows. □

Now we are ready for the main theorem asserting the existence (under suitable conditions) of a highly correlated Fourier basis element. The basic approach of the following proof is reminiscent of the main technical lemma from [13].

Theorem 6. *Let τ be a parameter to be specified later and $\mathfrak{C}$ be the concept class consisting of s-*MAJORITY *of r-*PARITY *of b-literals where $s = \mathrm{poly}(\tau)$ and $r = O(\frac{\log(\tau)}{\log\log(\tau)})$. Then for any $f \in C_{n,b}$ and any distribution $\mathcal{D}$ over $[b]^n$ with $L_\infty(\mathcal{D}) = \mathrm{poly}(\tau)/b^n$, there exists a Fourier basis element χ_α such that $|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| > \Omega(1/\mathrm{poly}(\tau))$.*

Proof. Assume f is a MAJORITY of $h_1, \ldots, h_s$ each of which is a r-PARITY of b-literals. Then Lemma 2 implies that there exists h_i such that $|\mathbf{E}_{\mathcal{D}}[fh_i]| \geq 1/s$. Let h_i be PARITY of the b-literals $\ell_1, \ldots, \ell_r$.

Since s and $b^n \cdot L_\infty(\mathcal{D})$ are both at most $\mathrm{poly}(\tau)$ and $r = O(\frac{\log(\tau)}{\log\log(\tau)})$, Lemma 7 implies that there are absolute constants C_1, C_2 such that if we consider the k-restrictions $\tilde{\ell}_1, \ldots, \tilde{\ell}_r$ of $\ell_1, \ldots, \ell_r$ for $k = C_1 \cdot \tau^{C_2}$, we will have $\mathbf{E}[|h_i - \prod_{j=1}^r \tilde{\ell}_j|] \leq 1/(2sb^n L_\infty(\mathcal{D}))$ where the expectation on the left hand side is with respect to the uniform distribution on $[b]^n$. This in turn implies that $\mathbf{E}_{\mathcal{D}}[|h_i - \prod_{j=1}^r \tilde{\ell}_j|] \leq 1/2s$. Let us write h' to denote $\prod_{j=1}^r \tilde{\ell}_j$. We then have

$$\begin{aligned}|\mathbf{E}_{\mathcal{D}}[f\overline{h'}]| &\geq |\mathbf{E}_{\mathcal{D}}[f\overline{h_i}]| - |\mathbf{E}_{\mathcal{D}}[f\overline{(h_i - h')}]| \geq |\mathbf{E}_{\mathcal{D}}[f\overline{h_i}]| - \mathbf{E}_{\mathcal{D}}[|f\overline{(h_i - h')}|] \\ &= |\mathbf{E}_{\mathcal{D}}[fh_i]| - \mathbf{E}_{\mathcal{D}}[|h_i - h'|] \geq 1/s - 1/2s = 1/2s.\end{aligned}$$

Now observe that we additionally have

$$|\mathbf{E}_{\mathcal{D}}[f\overline{h'}]| = |\mathbf{E}_{\mathcal{D}}[f\overline{\sum_\alpha \hat{h'}(\alpha)\chi_\alpha}]| = |\sum_\alpha \overline{\hat{h'}(\alpha)}\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| \leq L_1(h') \max_\alpha |\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]|$$

Moreover, for each $j = 1, \ldots, r$ we have the following (where we write ℓ'_j to denote the basic b-literal associated with the b-literal ℓ_j):

$$L_1(\tilde{\ell}_j) = \sum_{abs(\alpha)\leq k} |\widehat{\ell'}_j(\alpha)| \underbrace{=}_{\text{by Corollary 3}} 1 + \sum_{\alpha=1}^{k} O(1)/\alpha = O(\log k).$$

Therefore, for some absolute constant $c > 0$ we have $L_1(h') \leq \prod_{j=1}^{r} L_1(\tilde{l}_j) \leq (c \log k)^r$, where the first inequality holds since the L_1 norm of a product is at most the product of the L_1 norms. Combining inequalities, we obtain

$$\max_{\alpha} |\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| \geq 1/(2s(c\log k)^r) = \Omega(1/\mathrm{poly}(\tau))$$

which is the desired result. □

Since we are interested in algorithms with runtime $\mathrm{poly}(n, \log b, \epsilon^{-1})$, setting $\tau = n\epsilon^{-1}\log b$ in Theorem 6 and combining its result with Corollary 2, gives rise to Theorem 1.

4.3 The second approach

A different analysis, similar to that which Jackson uses in the proof of [12, Fact 14], gives us an alternate bound to Theorem 6:

Lemma 8. *Let $\mathfrak{C}$ be the concept class consisting of s-*Majority *of r-*Parity *of b-literals. Then for any $f \in C_{n,b}$ and any distribution $\mathcal{D}$ over $[b]^n$, there exists a Fourier basis element χ_α such that $|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| = \Omega(1/s(\log b)^r)$.*

Proof. Assume f is a Majority of $h_1, \ldots, h_s$ each of which is a r-Parity of b-literals. Then Lemma 2 implies that there exists h_i such that $|\mathbf{E}_{\mathcal{D}}[fh_i]| \geq 1/s$. Let h_i be Parity of the b-literals $\ell_1, \ldots, \ell_r$. Now observe:

$$1/s \leq |\mathbf{E}_{\mathcal{D}}[fh_i]| = |\mathbf{E}_{\mathcal{D}}[f\overline{h_i}]| = |\sum_{\alpha} \overline{\hat{h_i}(\alpha)}\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| \leq L_1(h_i)\max_{\alpha}|\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]|$$

Also note that for $j = 1, \ldots, r$ we have the following (where as before we write ℓ'_j to denote the basic b-literal associated with the b-literal ℓ_j):

$$L_1(\ell_j) \underbrace{=}_{\text{by Lemma 5}} \sum_{\alpha} |\hat{\ell_j}(\alpha)| \underbrace{=}_{\text{by Corollary 3}} 1 + \sum_{\alpha=1}^{b-1} O(1)/\alpha = O(\log b).$$

Therefore for some constant $c > 0$ we have $L_1(h_i) \leq \prod_{j=1}^{r} L_1(\ell_j) = O((\log b)^r)$, from which we obtain $\max_\alpha |\mathbf{E}_{\mathcal{D}}[f\overline{\chi_\alpha}]| = \Omega(1/s(\log b)^r)$. □

Combining this result with that of Corollary 2 we obtain the following result:

Theorem 7. *The concept class $\mathfrak{C}$ consisting of s-*Majority *of r-*Parity *of b-literals can be learned in time $\mathrm{poly}(s, n, (\log b)^r)$ using the* GHS *algorithm.*

As an immediate corollary we obtain the following close analogue of Theorem 1:

Theorem 8. *The concept class $\mathfrak{C}$ consisting of s-*Majority *of r-*Parity *of b-literals where $s = \mathrm{poly}(n \log b)$, $r = O(\frac{\log(n \log b)}{\log \log b})$ is efficiently learnable using the* GHS *algorithm.*

5 Locating sensitive elements and learning with GHS on a restricted grid

In this section we consider an extension of the GHS algorithm which lets us achieve slightly better bounds when we are dealing only with basic b-literals. Following an idea from [3], the new algorithm works by identifying a subset of "sensitive" elements from $[b]$ for each of the n dimensions.

Definition 4 (See [3]). *A value $\sigma \in [b]$ is called* i-sensitive *with respect to $f\colon [b]^n \to \{-1,1\}$ if there exist values $c_1, c_2, \ldots, c_{i-1}, c_{i+1}, \ldots, c_n \in [b]$ such that $f(c_1, \ldots, c_{i-1}, \sigma-1, c_{i+1}, \ldots, c_n) \neq f(c_1, \ldots, c_{i-1}, \sigma, c_{i+1}, \ldots, c_n)$. A value σ is called* sensitive *with respect to f if σ is i-sensitive for some i. If there is no i-sensitive value with respect to f, we say index i is* trivial.

The main idea is to run GHS over a restricted subset of the original domain $[b]^n$, which is the grid formed by the sensitive values and a few more additional values, and therefore lower the algorithm's complexity.

Definition 5. *A* grid *in $[b]^n$ is a set $\mathcal{S} = L_1 \times L_2 \times \cdots \times L_n$ with $0 \in L_i \subseteq [b]$ for each i. We refer to the elements of $\mathcal{S}$ as* corners. *The* region covered by a corner *$(x_1, \ldots, x_n) \in \mathcal{S}$ is defined to be the set $\{(y_1, \ldots, y_n) \in [b]^n \colon \forall i, x_i \leq y_i < \lceil x_i \rceil\}$ where $\lceil x_i \rceil$ denotes the smallest value in L_i which is larger than x_i (by convention $\lceil x_i \rceil := b$ if no such value exists). The* area covered by the corner *$(x_1, \ldots, x_n) \in \mathcal{S}$ is therefore defined to be $\prod_{i=1}^{n}(\lceil x_i \rceil - x_i)$. A* refinement *of $\mathcal{S}$ is a grid in $[b]^n$ of the form $L'_1 \times L'_2 \times \cdots \times L'_n$ where each $L_i \subseteq L'_i$.*

Lemma 9. *Let $\mathcal{S}$ be a grid $L_1 \times L_2 \times \cdots \times L_n$ in $[b]^n$ such that each $|L_i| \leq \ell$. Let $\mathcal{I}_{\mathcal{S}}$ denote the set of indices for which $L_i \neq \{0\}$. If $|\mathcal{I}_{\mathcal{S}}| \leq \kappa$, then $\mathcal{S}$ admits a refinement $\mathcal{S}' = L'_1 \times L'_2 \times \cdots \times L'_n$ such that*

1. *All of the sets L'_i which contain more than one element have the same number of elements: $\mathbf{L_{max}}$, which is at most $\ell + C\kappa\ell$, where $C = \frac{b}{\kappa\ell} \cdot \frac{1}{\lfloor b/4\kappa\ell \rfloor} \geq 4$.*
2. *Given a list of the sets $L_1, \ldots, L_n$ as input, a list of the sets $L'_1, \ldots, L'_n$ can be generated by an algorithm with a running time of $O(n\kappa\ell \log b)$.*
3. *$L'_i = \{0\}$ whenever $L_i = \{0\}$.*
4. *Any ϵ fraction of the corners in $\mathcal{S}'$ cover a combined area of at most $2\epsilon b^n$.*

Proof. Consider Algorithm 1 which, given $\mathcal{S} = L_1 \times L_2 \times \cdots \times L_n$, generates $\mathcal{S}'$.

The purpose of the code between lines 18–22 is to make every $L'_i \neq \{0\}$ contain equal number of elements. Therefore the algorithm keeps track of the number of elements in the largest L'_i in a variable called $\mathbf{L_{max}}$ and eventually adds more (arbitrary) elements to those $L'_i \neq \{0\}$ which have fewer elements.

It is clear that the algorithm satisfies Property 3 above.

Now consider the state of Algorithm 1 at line 18. Let i be such that $|L'_i| = \mathbf{L_{max}}$. Clearly L'_i includes the elements in L_i which are at most ℓ many. Moreover every new element added to L'_i in the loop spanning lines 8-12 covers a section of $[b]$ of width τ, and thus $b/\tau = C\kappa\ell$ elements can be added. Thus $\mathbf{L_{max}} \leq \ell + C\kappa\ell$. At the end of the algorithm every L'_i contains either 1 element (which is $\{0\}$) or $\mathbf{L_{max}}$ elements. This gives us Property 1. Note that $C \geq 4$ by construction.

Algorithm 1 Computing a refinement of the grid $\mathcal{S}$ with the desired properties.

$\mathbf{L_{max}} \leftarrow 0$.
for all $1 \leq i \leq n$ **do**
  if $L_i = \{0\}$ **then**
    $L'_i \leftarrow \{0\}$.
  else
    Consider $L_i = \{x^i_0, x^i_1 \ldots, x^i_{\ell-1}\}$, where $x^i_0 < x^i_1 < \cdots < x^i_{\ell-1}$ (Also let $x^i_\ell = b$).
    Set $L'_i \leftarrow L_i$ and $\tau \leftarrow \lfloor b/4\kappa\ell \rfloor$.
    for all $r = 0, \ldots, \ell - 1$ **do**
      if $|x^i_{r+1} - x^i_r| > \tau$ **then**
        $L'_i \leftarrow L'_i \cup \{x^i_r + \tau, x^i_r + 2\tau, \ldots\}$ (up to and including the largest $x^i_r + j \cdot \tau$ which is less than x^i_{r+1})
      end if
    end for
    if $|L'_i| > \mathbf{L_{max}}$ **then**
      $\mathbf{L_{max}} \leftarrow |L'_i|$.
    end if
  end if
end for
for all $1 \leq i \leq n$ with $|L'_i| > 1$ **do**
  while $(|L'_i| < \mathbf{L_{max}})$ **do**
    $L'_i \leftarrow L'_i \cup \{$an arbitrary element from $[b]\}$.
  end while
end for
$\mathcal{S}' \leftarrow L'_1 \times L'_2 \times \cdots \times L'_n$.

It is easy to verify that it satisfies Property 2 as well (the $\log b$ factor in the runtime is present because the algorithm works with $(\log b)$-bit integers).

Property 1 and the bound $|\mathcal{I}_\mathcal{S}| \leq \kappa$ together give that the number of corners in $\mathcal{S}$ is at most $(\ell + C\kappa\ell)^\kappa$. It is easy to see from the algorithm that the area covered by each corner in $\mathcal{S}'$ is at most $\frac{b^n}{(C\kappa\ell)^\kappa}$ (again using the bound on $|\mathcal{I}_\mathcal{S}|$). Therefore any ϵ fraction of the corners in $\mathcal{S}'$ cover an area of at most:

$$\epsilon(\ell + C\kappa\ell)^\kappa \times \frac{b^n}{(C\kappa\ell)^\kappa} = \epsilon\left(1 + \frac{1}{C\kappa}\right)^\kappa \times b^n \underbrace{<}_{C \geq 4} e^{1/3}\epsilon b^n < 2\epsilon b^n.$$

This gives Property 4. □

The following lemma is easy and useful; similar statements are given in [3]. Note that the lemma critically relies on the b-literals being basic.

Lemma 10. *Let $f \colon [b]^n \to \{-1, 1\}$ be expressed as an s-*Majority *of* Parity *of basic b-literals. Then for each index $1 \leq i \leq n$, there are at most $2s$ i-sensitive values with respect to f.*

Proof. A literal ℓ on variable x_i induces two i-sensitive values. The lemma follows directly from our assumption (see Section 2) that for each variable x_i, each of the s Parity gates has no more than one incoming literal which depends on x_i. □

Algorithm 2 An improved algorithm for learning Majority of Parity of basic b-literals.

```
L_1 ← {0}, L_2 ← {0}, ..., L_n ← {0}.
loop
    S ← L_1 × L_2 × ··· × L_n.
    S' ← the output of refinement algorithm with input S.
    One can express S' = L'_1 × L'_2 × ··· × L'_n. If L_i ≠ {0} then L'_i =
    {x^i_0, x^i_1 ..., x^i_(L_max − 1)}. Let x^i_0 < x^i_1 < ··· < x^i_{t−1} and let τ_i : Z_{L_max} → L'_i
    be the translation function such that τ_i(j) = x^i_j. If L_i = L'_i = {0} then τ_i is the
    function simply mapping 0 to 0.
    Invoke GHS over f|_S' with accuracy ε/8. This is done by simulating
    MEM(f|_S'(x_1, ..., x_n)) with MEM(f(τ_1(x_1), τ_2(x_2), ..., τ_n(x_n))). Let the output
    of the algorithm be g.
    Let h be a hypothesis function over [b]^n such that h(x_1, ..., x_n) =
    g(τ_1^{−1}(⌊x_1⌋), ..., τ_n^{−1}(⌊x_n⌋)) (⌊x_i⌋ denotes largest value in L'_i less than or equal
    to x_i).
    if h ε-approximates f then
        Output h and terminate.
    end if
    Perform random membership queries until an element (x_1, ..., x_n) ∈ [b]^n is found
    such that f(⌊x_1⌋, ..., ⌊x_n⌋) ≠ f(x_1, ..., x_n).
    Find an index 1 ≤ i ≤ n such that f(⌊x_1⌋, ..., ⌊x_{i−1}⌋, x_i, ..., x_n) ≠
    f(⌊x_1⌋, ..., ⌊x_{i−1}⌋, ⌊x_i⌋, x_{i+1}, ..., x_n). This requires O(log n) membership
    queries using binary search.
    Find a value σ such that ⌊x_i⌋ + 1 ≤ σ ≤ x_i and f(⌊x_1⌋, ..., ⌊x_{i−1}⌋, σ −
    1, x_{i+1}, ..., x_n) ≠ f(⌊x_1⌋, ..., ⌊x_{i−1}⌋, σ, x_{i+1}, ..., x_n). This requires O(log b)
    membership queries using binary search.
    L_i ← L_i ∪ {σ}.
end loop
```

Algorithm 2 is our extension of the GHS algorithm. It essentially works by repeatedly running GHS on the target function f but restricted to a small (relative to $[b]^n$) grid. To upper bound the number of steps in each of these invocations we will be referring to the result of Theorem 8. After each execution of GHS, the hypothesis defined over the grid is extended to $[b]^n$ in a natural way and is tested for ϵ-accuracy. If h is not ϵ-accurate, then a point where h is incorrect is used to identify a new sensitive value and this value is used to refine the grid for the next iteration. The bound on the number of sensitive values from Lemma 10 lets us bound the number of iterations. Our theorem about Algorithm 2's performance is the following:

Theorem 9. *Let concept class $\mathfrak{C}$ consist of s-*Majority *of r-*Parity *of basic b-literals such that $s = \mathrm{poly}(n \log b)$ and each $f \in C_{n,b}$ has at most $\kappa(n, b)$ non-trivial indices and at most $\ell(n, b)$ i-sensitive values for each $i = 1, \ldots, n$. Then $\mathfrak{C}$ is efficiently learnable if $r = O(\frac{\log(n \log b)}{\log \log \kappa\ell})$.*

Proof. We assume $b = \omega(\kappa\ell)$ without loss of generality. Otherwise one immediately obtains the result with a direct application of GHS through Theorem 8.

We clearly have $\kappa \leq n$ and $\ell \leq 2s$. By Lemma 10 there are at most $\kappa\ell = O(ns)$ sensitive values. We will show that the algorithm finds a new sensitive value at each iteration and terminates before all sensitive values are found. Therefore the number of iterations will be upper bounded by $O(ns)$. We will also show that each iteration runs in $\mathrm{poly}(n, \log b, \epsilon^{-1})$ steps. This will establish the desired result.

Let's first establish that step 6 takes at most $\mathrm{poly}(n, \log b, \epsilon^{-1})$ steps. To observe this it is sufficient to combine the following facts:

– Due to the construction of Algorithm 1 for every non-trivial index i of f, L'_i has fixed cardinality $= \mathbf{L_{max}}$. Therefore GHS could be invoked over the restriction of f onto the grid, $f|_{\mathcal{S}'}$, without any trouble.
– If f is s-Majority of r-Parity of basic b-literals, then the function obtained by restricting it onto the grid: $f|_{\mathcal{S}'}$ could be expressed as t-Majority of u-Parity of basic L-literals where $t \leq s$, $u \leq r$ and $L \leq O(\kappa\ell)$ (due to the 1$^{\text{st}}$ property of the refinement).
– Running GHS over a grid with alphabet size $O(\kappa\ell)$ in each non-trivial index takes $\mathrm{poly}(n, \log b, \epsilon^{-1})$ time if the dimension of the rectangles are $r = O(\frac{\log(n \log b)}{\log\log \kappa\ell})$ due to Theorem 8. The key idea here is that running GHS over this $\kappa\ell$-size alphabet lets us replace the "b" in Theorem 8 with "$\kappa\ell$".

To check whether if h ϵ-approximates f at step 8, we may draw $O(1/\epsilon) \cdot \log(1/\delta)$ uniform random examples and use the membership oracle to empirically estimate h's accuracy on these examples. Standard bounds on sampling show that if the true error rate of h is less than (say) $\epsilon/2$, then the empirical error rate on such a sample will be less than ϵ with probability $1-\delta$. Observe that if all the sensitive values are recovered by the algorithm, h will ϵ-approximate f with high probability. Indeed, since g $(\epsilon/8)$-approximates $f|_{\mathcal{S}'}$, Property 4 of the refinement guarantees that misclassifying the function at $\epsilon/8$ fraction of the corners could at most incur an overall error of $2\epsilon/8 = \epsilon/4$. This is because when all the sensitive elements are recovered, for every corner in $\mathcal{S}'$, h either agrees with f or disagrees with f in the entire region covered by that corner. Thus h will be an $\epsilon/4$ approximator to f with high probability. This establishes that the algorithm must terminate within $O(ns)$ iterations of the outer loop.

Locating another sensitive value occurs at steps 11, 12 and 13. Note that h is not an ϵ-approximator to f because the algorithm moved beyond step 8. Even if we were to correct all the mistakes in g this would alter at most $\epsilon/8$ fraction of the corners in the grid $\mathcal{S}'$ and therefore $\epsilon/4$ fraction of the values in h – again due to the 4$^{\text{th}}$ property of the refinement and the way h is generated. Therefore for at least $3\epsilon/4$ fraction of the domain we ought to have $f(\lfloor x_1 \rfloor, \ldots, \lfloor x_n \rfloor) \neq f(x_1, \ldots, x_n)$ where $\lfloor x_i \rfloor$ denotes largest value in L'_i less than or equal to x_i. Thus the algorithm requires at most $O(1/\epsilon)$ random queries to find such an input in step 11.

Thus we have observed that steps 6, 8, 11, 12, 13 take at most $\mathrm{poly}(n, \log b, \epsilon^{-1})$ steps. Therefore each iteration of Algorithm 2 runs in $\mathrm{poly}(n, \log b, \epsilon^{-1})$ steps as claimed.

We note that we have been somewhat cavalier in our treatment of the failure probabilities for various events (such as the possibility of getting an inaccurate estimate of h's error rate in step 9, or not finding a suitable element $(x_1, \dots, x_n)$ soon enough in step 11). A standard analysis shows that all these failure probabilities can be made suitably small so that the overall failure probability is at most δ within the claimed runtime. $\square$

6 Applications to learning unions of rectangles

In this section we apply the results we have obtained in Sections 4 and 5 to obtain results on learning unions of rectangles and related classes.

6.1 Learning majorities and unions of many low-dimensional rectangles

The following lemma will let us apply our algorithm for learning MAJORITY of PARITY of b-literals to learn MAJORITY of AND of b-literals:

Lemma 11. *Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be expressible as an s-*MAJORITY *of r-*AND *of Boolean literals. Then f is also expressible as a $O(ns^2)$-*MAJORITY *of r-*PARITY *of Boolean literals.*

We note that Krause and Pudlák gave a related but slightly weaker bound in [16]; they used a probabilistic argument to show that any s-MAJORITY of AND of Boolean literals can be expressed as an $O(n^2s^4)$-MAJORITY of PARITY. Our boosting-based argument below closely follows that of [12, Corollary 13].

Proof of Lemma 11: Let f be the MAJORITY of $h_1, \dots, h_s$ where each h_i is an AND gate of fan-in r. By Lemma 2, given any distribution $\mathcal{D}$ there is some AND function h_j such that $|\mathbf{E}_{\mathcal{D}}[fh_j]| \geq 1/s$. It is not hard to show that the L_1-norm of any AND function is at most 4 (see, e.g., [17, Lemma 5.1] for a somewhat more general result), so we have $L_1(h_j) \leq 4$. Now the argument from the proof of Lemma 8 shows that there must be some parity function χ_a such that $|\mathbf{E}_{\mathcal{D}}[f\chi_a]| \geq 1/4s$, where the variables in χ_a are a subset of the variables in h_j – and thus χ_a is a parity of at most r literals. Consequently, we can apply the boosting algorithm of [7] stated in Theorem 4, choosing the weak hypothesis to be a PARITY with fan-in at most r at each stage of boosting, and be assured that each weak hypothesis has advantage at least $1/4s$ at every stage of boosting. If we boost to accuracy $\epsilon = \frac{1}{2^n+1}$, then the resulting final hypothesis will have zero error with respect to f and will be a MAJORITY of $O(\log(1/\epsilon)/s^2) = O(ns^2)$ many r-PARITY functions. Note that while this argument does not lead to a computationally efficient construction of the desired MAJORITY of r-PARITY, it does establish its existence, which is all we need. $\square$

Note that clearly any union (OR) of s many r-rectangles can be expressed as an $O(s)$-MAJORITY of r-rectangles as well.

Theorem 1 and Lemma 11 together give us Theorem 2. (In fact, these results give us learnability of s-MAJORITY of r-AND of b-literals which need not necessarily be basic.)

6.2 Learning unions of fewer rectangles of higher dimension

We now show that the number of rectangles s and the dimension bound r of each rectangle can be traded off against each other in Theorem 2 to a limited extent. We state the results below for the case $s = \mathrm{poly}(\log(n \log b))$, but one could obtain analogous results for a range of different choices of s.

We require the following lemma:

Lemma 12. *Any s-term r-DNF can be expressed as an $r^{O(\sqrt{r}\log s)}$-MAJORITY of $O(\sqrt{r}\log s)$-PARITY of Boolean literals.*

Proof. [15, Corollary 13] states that any s-term r-DNF can be expressed as an $r^{O(\sqrt{r}\log s)}$-MAJORITY of $O(\sqrt{r}\log s)$-ANDs. By considering the Fourier representation of an AND, it is clear that each t-AND in the MAJORITY can be replaced by at most $2^{O(t)}$ many t-PARITYs, corresponding to the parities in the Fourier representation of the AND. This gives the lemma. □

Now we can prove Theorem 3, which gives us roughly a quadratic improvement in the dimension r of rectangles over Theorem 2 when $s = \mathrm{poly}(\log(n \log b))$.

Proof of Theorem 3: First note that by Lemma 10, any function in $C_{n,b}$ can have at most $\kappa = O(rs) = \mathrm{poly}(\log(n \log b))$ non-trivial indices, and at most $\ell = O(s) = \mathrm{poly}(\log(n \log b))$ many i-sensitive values for all $i = 1, \ldots, n$. Now use Lemma 12 to express any function in $C_{n,b}$ as an s'-MAJORITY of r'-PARITY of basic b-literals where $s' = r^{O(\sqrt{r}\log s)} = \mathrm{poly}(n \log b)$ and $r' = O(\sqrt{r}\log s) = O(\frac{\log(n \log b)}{\log\log\log(n \log b)})$. Finally, apply Theorem 9 to obtain the desired result. □

Note that it is possible to obtain a similar result for learning $\mathrm{poly}(\log(n \log b))$ union of $O(\frac{\log^2(n \log b)}{(\log\log(n \log b))^4})$-AND of b-literals if one were to invoke Theorem 1.

6.3 Learning majorities of unions of disjoint rectangles

A set $\{R_1, \ldots, R_s\}$ of rectangles is said to be *disjoint* if every input $x \in [b]^n$ satisfies at most one of the rectangles. Learning unions of disjoint rectangles over $[b]^n$ was studied by [3], and is a natural analogue over $[b]^n$ of learning "disjoint DNF" which has been well studied in the Boolean domain (see e.g. [14, 2]).

We observe that when disjoint rectangles are considered Theorem 2 extends to the concept class of majority of unions of disjoint rectangles; enabling us to improve the depth of our circuits by 1. This extension relies on the easily verified fact that if $f_1, \ldots, f_t$ are functions from $[b]^n$ to $\{-1, 1\}^n$ such that each x satisfies at most one f_i, then the function $\mathrm{OR}(f_1, \ldots, f_t)$ satisfies $L_1(\mathrm{OR}(f_1, \ldots, f_t)) = O(L_1(f_1) + \cdots + L_1(f(t)))$. This fact lets us apply the argument behind Theorem 6 without modification, and we obtain Corollary 1. Note that only the rectangles connected to the same OR gate must be disjoint in order to invoke Corollary 1.

6.4 Conclusions and future work

For future work, besides the obvious goals of strengthening our positive results, we feel that it would be interesting to explore the limitations of current techniques for learning unions of rectangles over $[b]^n$. At this point we cannot rule out the possibility that the Generalized Harmonic Sieve algorithm is in fact a poly$(n, s, \log b)$-time algorithm for learning unions of s arbitrary rectangles over $[b]^n$. Can evidence for or against this possibility be given? For example, can one show that the representational power of the hypotheses which the Generalized Harmonic Sieve algorithm produces (when run for poly$(n, s, \log b)$ many stages) is – or is not – sufficient to express high-accuracy approximators to arbitrary unions of s rectangles over $[b]^n$?

References

[1] A. Akavia, S. Goldwasser, S. Safra, *Proving Hard Core Predicates Using List Decoding*, Proc. **44**th FOCS: 146–156 (2003).

[2] H. Aizenstein, A. Blum, R. Khardon, E. Kushilevitz, L. Pitt, D. Roth, *On Learning Read-k Satisfy-j DNF*, SIAM Journal on Computing, **27**(6): 1515–1530 (1998).

[3] A. Beimel, E. Kushilevitz, *Learning Boxes in High Dimension*, Algorithmica, **22**(1/2): 76–90 (1998).

[4] J. Bruck. *Harmonic analysis of polynomial threshold functions*, SIAM Journal on Discrete Mathematics, **3**(2): 168–177 (1990).

[5] Z. Chen and S. Homer, *The Bounded Injury Priority Method and The Learnability of Unions of Rectangles*, Annals of Pure and Applied Logic, **77**(2): 143–168 (1996).

[6] Z. Chen and W. Maass, *On-line Learning of Rectangles and Unions of Rectangles*, Machine Learning, **17**(2/3): 23–50 (1994).

[7] Y. Freund, *Boosting a Weak Learning Algorithm by Majority*, Proceedings of the **3**rd Annual Workshop on Computational Learning Theory, 202–216 (1990).

[8] Y. Freund and R. Schapire. *A short introduction to boosting*, Journal of the Japanese Society for Artificial Intelligence, **14**(5): 771-780 (1999).

[9] P. W. Goldberg, S. A. Goldman, H. D. Mathias, *Learning Unions of Boxes with Membership and Equivalence Queries*, Proceedings of the **7**th Annual ACM Conference on Computational Learning Theory: 198 – 207 (1994).

[10] A. Hajnal, W. Maass, P. Pudlák, M. Szegedy, G. Turan, *Threshold Circuits of Bounded Depth*, J. Comp. & Syst. Sci. **46**: 129–154 (1993).

[11] J. Håstad, *Computational Limitations for Small Depth Circuits*, MIT Press, Cambridge, MA (1986).

[12] J. C. Jackson, *An Efficient Membership-Query Algorithm for Learning* DNF *with Respect to the Uniform Distribution*, J. Comp. & Syst. Sci. **55**(3): 414–440 (1997).

[13] J. C. Jackson, A. R. Klivans, R. A. Servedio, *Learnability Beyond* AC^0, Proceedings of the **34**th Symposium on Theory of Computing (STOC): 776–784 (2002).

[14] R. Khardon. *On Using the Fourier Transform to Learn Disjoint DNF*, Information Processing Letters,**49**(5): 219–222 (1994).

[15] A. R. Klivans, R. A. Servedio, *Learning* DNF *in Time* $2^{\tilde{O}(n^{1/3})}$, Proceedings of the **33**rd Symposium on Theory of Computing (STOC): 258–265 (2001).

[16] M. Krause and P. Pudlák, *Computing Boolean Functions by Polynomials and Threshold Circuits*, Computational Complexity **7**(4): 346–370 (1998).

[17] E. Kushilevitz and Y. Mansour, *Learning Decision Trees using the Fourier Spectrum*, SIAM Journal on Computing **22**(6): 1331-1348 (1993).

[18] W. Maass and M. K. Warmuth, *Efficient Learning with Virtual Threshold Gates*, Information and Computation, **141**(1): 66–83 (1998).

[19] R. E. Schapire, *The Strength of Weak Learnability*, Machine Learning **5**: 197–227 (1990).