On the Efficiency of Noise-Tolerant PAC Algorithms Derived from Statistical Queries

Jeffrey Jackson [*]

Math. & Comp. Science Dept., Duquesne University, Pittsburgh, PA 15282-1754
E-mail: jackson@mathcs.duq.edu

The Statistical Query (SQ) model provides an elegant means for generating noise-tolerant PAC learning algorithms that run in time inverse polynomial in the noise rate. Whether or not there is an SQ algorithm for every noise-tolerant PAC algorithm that is efficient in this sense remains an open question. However, we show that PAC algorithms derived from the Statistical Query model are not always the most efficient possible. Specifically, we give a general definition of SQ-based algorithm and show that there is a subclass of parity functions for which there is an efficient PAC algorithm requiring asymptotically less running time than any SQ-based algorithm. While it turns out that this result can be derived fairly easily by combining a recent algorithm of Blum, Kalai, and Wasserman with an older lower bound, we also provide alternate, Fourier-based approaches to both the upper and lower bounds that strengthen the results in various ways. The lower bound in particular is stronger than might be expected, and the amortized technique used in deriving this bound may be of independent interest.

Keywords: PAC learning, Statistical Query model, Fourier analysis

AMS Subject classification: Primary 68Q32; Secondary 68T05

1. Introduction

Kearns's Statistical Query (SQ) model [10] is a well-studied, elegant abstraction from Valiant's foundational Probably Approximately Correct (PAC) learning model [14]. In the PAC model, a learning algorithm is given access (through a so-called oracle) to examples of an unknown function f, generally assumed to be

[*] This material is based upon work supported by the National Science Foundation under Grants No. CCR-9800029 and CCR-9877079. A preliminary version of this material appeared in the Proceedings of the Thirteenth Annual Conference on Computational Learning Theory.

Boolean. Each example consists of a vector of attribute values—assumed Boolean throughout this paper, but not in the general PAC model—and the value of f for the specified attribute values. The goal of the learning algorithm is to, with high probability, produce a function h that closely approximates f on examples "similar" to those given to the learning algorithm. A rigorous definition of a particular PAC learning problem is given in the next section, but suffice it to say for now that it is generally accepted that the PAC model is a reasonably good mathematical model of many of the sorts of supervised classification learning problems often studied by machine learning empiricists. For example, some positive results obtained in the PAC model, most notably hypothesis boosting [13,7], have had a substantial impact on machine learning practice.

In Kearns's Statistical Query model, instead of having access directly to examples of the unknown function f, the learning algorithm is given access to an oracle that provides estimates of various "statistics" about the unknown function. Again, a formal definition is given in the next section, but for now we note that unlike the PAC example oracle, the SQ oracle does not directly correspond to anything readily available in the real world. However, Kearns showed that any function class efficiently learnable in the SQ model is also learnable in the PAC model despite noise uniformly applied to the class labels of the examples. What makes this result particularly interesting from an applications point of view is that the proof essentially outlines a generic method that can be used to simulate an SQ algorithm using a noisy PAC example oracle (a "noisy" oracle is one that mislabels each example with a fixed probability). The PAC algorithm resulting from the SQ simulation runs in time at most polynomially greater than the time required of an SQ algorithm using an SQ oracle, and also inverse polynomially in the noise probability. Thus, if an efficient SQ learning algorithm can be developed for a learning problem, an efficient noise-tolerant PAC algorithm can also be immediately derived for this problem.

In the same paper where Kearns showed that SQ learnability implies noise-tolerant PAC learnability, he developed SQ algorithms for almost all function classes known to be efficiently learnable in the PAC model. By doing this, he provided the first known noise-tolerant PAC algorithms for some of these classes. In fact, the SQ approach to developing noise-tolerant PAC algorithms was so successful that Kearns asked whether or not the SQ and PAC+noise models might be equivalent [10]. That is, is it possible that every class of functions that is efficiently learnable from a noisy PAC example oracle is also efficiently learnable from an SQ oracle? If so, this would mean that hardness results for learnability in the SQ model [2] would imply hardness in the PAC with noise model. It might also seem to imply that all future searching for noise-tolerant PAC algorithms should focus exclusively on finding SQ algorithms from which PAC algorithms could then be derived.

However, Blum, Kalai, and Wasserman [3] have recently shown that in some sense the PAC+noise and SQ models are different. Specifically, they show that there is a class that is in some sense efficiently PAC learnable with noise but not efficiently SQ learnable. However, they only show that this class can be learned efficiently when the noise rate is constant. That is, the time required by their algorithm is not polynomial in the inverse of the noise rate. This leaves open the question of whether or not there is an efficient SQ algorithm for every function class that is learnable in time inverse polynomial in the noise rate.

Like Blum *et al.*, this paper does not answer this intriguing question. However, we do show that using the SQ model to develop (inverse-polynomial) noise-tolerant PAC algorithms sometimes does not produce optimally efficient algorithms. Specifically, a formal definition of *SQ-based* PAC algorithms is developed. Informally, SQ-based algorithms are PAC algorithms that are derived through a generic process from SQ algorithms. This process is even generous to the SQ-based algorithm in that it assumes that the target function is noiseless and also assumes that an appropriate sample size for simulating each statistical query does not need to be computed but is instead known by the algorithm. Despite the generosity of this definition, we show that there is an efficient (in inverse noise rate as well as other standard learning parameters) PAC algorithm for the class of all parity functions on the first $O(\log n)$ bits of an n-bit input space, and that this algorithm is more efficient than any SQ-based algorithm for this class.

We actually present several somewhat different approaches to this result. First, while the Blum *et al.* results [3] focus on producing a superpolynomial separation between PAC+noise and SQ learning in the constant noise setting, they have in fact developed a family of parameterized algorithms that can be used to derive a variety of learnability results. In particular, some members of this family of algorithms efficiently tolerate inverse-polynomial noise rates for certain function classes. Given our definition of an SQ-based algorithm, it is relatively easy to combine these Blum, Kalai, and Wasserman algorithms with a lower bound from an older Blum *et al.* paper [2] to give a polynomial separation between SQ-based algorithms and inverse-polynomial noise-tolerant PAC algorithms when learning the $O(\log n)$ subclass of parity functions with respect to the uniform distribution over the n-bit input space (that is, for both training and testing purposes, the attribute values of examples are assigned uniformly at random).

We then improve on the lower bound. Specifically, a time bound of $\Omega(2^{n/2})$ for SQ-based learning of the class of parity functions over n bits that can be derived in a fairly straightforward way from [2], but improving on this seems to require deeper analysis. We improve this bound to $\Omega(2^n)$ using an amortized analysis approach that may also be useful in other settings.

Finally, we show that an algorithm based on the well-studied Goldreich-Levin parity-learning algorithm [8,11], which on the surface is quite different from the algorithm of Blum, Kalai, and Wasserman, achieves running time sufficient to also give a polynomial separation result between noisy PAC learning and SQ-based learning. The Goldreich-Levin algorithm was initially designed to allow the algorithm to select its own examples (that is, to use *membership queries*) rather than being restricted to examples provided by an oracle. The fact that Goldreich-Levin can be used without membership queries in this way is somewhat interesting in itself. Furthermore, the Goldreich-Levin algorithm appears to be resistant to much broader forms of noise than the Blum *et al.* algorithm, thus strengthening the separation between SQ and PAC+noise.

The mathematical analysis underlying these results depends heavily on discrete Fourier techniques, specifically properties of the Walsh transform. Fourier analysis was first used to analyze machine learning questions by Linial, Mansour, and Nisan [12]. Even this first application produced a very substantial result, showing that the class of functions that can be represented by polynomial-size, constant-depth Boolean circuits is learnable in time quasi-polynomial in the input size n. Several other substantial Fourier-based results have been obtained since then (e.g., [11,2,9,5]) as well as a number of smaller results, although several of the algorithms derived require the use of membership queries, limiting their practical application. As already noted, the results in this paper do not have this limitation.

The remainder of this paper is organized as follows. First, formal definitions of specific learning problems in the PAC and SQ models are given, along with definitions of the Fourier (Walsh) transform and statements of some standard Fourier results. Next, an initial separation between the PAC+noise and SQ-based learning models is presented by combining PAC learning results from Blum, Kalai, and Wasserman [3] with earlier Blum *et al.* lower bound SQ results [2].

Next, the lower bound on SQ-based learning is improved by directly analyzing sample complexity rather than relying on the number of statistical queries as a lower bound on run time. Finally, an alternative to the Blum, Kalai, and Wasserman PAC learning algorithm is presented, giving a somewhat more noise-tolerant algorithm for the upper bound.

2. Preliminaries

This paper focuses on the learnability of the class of parity functions in various learning models. Following standard Fourier learning theory notation, we use $\chi_a : \{0,1\}^n \to \{-1,+1\}$ to represent the *parity function* defined as follows:

$$\chi_a(x) = (-1)^{a \cdot x}$$

where $a \cdot x$ represents the dot product of the n-bit Boolean vectors a and x. We define the *class of parity functions on n bits* $PAR_n = \{f \mid f = \chi_a \text{ or } f = -\chi_a, a \in \{0,1\}^n\}$ and the *class of parity functions* $PAR = \cup_{n=0}^{\infty} PAR_n$.

The two models of learnability we will focus on are both variants of Valiant's Probably Approximately Correct (PAC) model [14]. The first question we consider is the PAC *uniform random classification-noise learnability of PAR with respect to the uniform distribution.* In this model, the learning algorithm $\mathcal{A}$ is given access to a noisy example oracle $EX^{\eta}(f, U_n)$. Here η is the noise rate of the oracle (assumed known, although knowing a lower bound suffices for all of our results), f is a fixed unknown parity function in PAR_n for some value of n, and U_n represents the uniform distribution over $\{0,1\}^n$. On each draw from the oracle, a vector $x \in \{0,1\}^n$ is drawn uniformly at random. The oracle then randomly chooses between returning the noiseless example $\langle x, f(x)\rangle$, with probability $1-\eta$, and returning the noisy example $\langle x, -f(x)\rangle$, with probability η. $\mathcal{A}$ is also given parameters $\epsilon, \delta > 0$. The goal of the learner is to produce a function $h : \{0,1\}^n \to \{-1,+1\}$ such that, with probability at least $1-\delta$, $\mathbf{Pr}_{x \sim U_n}[f(x) \neq h(x)] \leq \epsilon$. Such an h is called an *ϵ-approximator to f with respect to the uniform distribution.*

The second question considered is the Statistical Query learnability of PAR. A *uniform-distribution Statistical Query (SQ) oracle*— denoted $SQ(g, \tau)$—is an oracle for an unknown target function $f : \{0,1\}^n \to \{0,1\}$. Given a function $g : \{0,1\}^{n+1} \to \{-1,+1\}$ and a *tolerance* $\tau \geq 0$, $SQ(g,\tau)$ returns a value $\tilde{\mu}$ such that $|\mathbf{E}_{x \sim U_n}[g(x, f(x))] - \tilde{\mu}| \leq \tau$, where $\mathbf{E}_{x \sim U_n}[\cdot]$ represents expected value with respect to the uniform distribution over $\{0,1\}^n$. A Boolean function class $\mathcal{C}$ is *uniform-distribution learnable in the Statistical Query model (SQ-learnable)* if there is an algorithm $\mathcal{A}$ that, given any $\epsilon > 0$ and access to an SQ oracle for any function $f \in \mathcal{C}$, produces a function $h : \{0,1\}^n \to \{0,1\}$ such that $\mathbf{Pr}_{x\sim U_n}[f(x) \neq h(x)] \leq \epsilon$. In this paper we consider only the original additive error version of statistical queries and not the relative error model, which is in some sense polynomially equivalent [1].

These definitions can be generalized to arbitrary probability distributions rather than U_n in an obvious way. However, in this paper, our focus is on the uniform distribution. In the sequel, probabilities and expectations that do not specify a distribution are over the uniform distribution on $\{0,1\}^n$ for a value of n that will be obvious from context.

We will also make use of an algorithm that uses queries to a membership oracle in order to weakly learn certain function classes with respect to the uniform distribution. A *membership oracle* for $f : \{0,1\}^n \to \{-1,+1\}$ ($MEM(f)$) is an oracle that given any n-bit vector x returns the value $f(x)$. A function $h : \{0,1\}^n \to \{-1,+1\}$ is a *weak approximation with respect to the uniform distribution* to a function $f : \{0,1\}^n \to \{-1,+1\}$ if $\mathbf{Pr}_{x\sim U_n}[f(x) = h(x)] \geq \frac{1}{2} + \theta$, where θ is inverse polynomial in parameters appropriate for the learning problem considered (the specific parameters will not be important for our purposes). A uniform-distribution learning algorithm for a class $\mathcal{A}$ that produces weak approximators as hypotheses—rather than ϵ-approximators as in the models above—is said to *weakly learn* $\mathcal{A}$. Learning algorithms that produce ϵ-approximators are sometimes referred to as *strong*.

Several of our results will use Fourier analysis. Given a Boolean function $f : \{0,1\}^n \to \{-1,+1\}$ and an n-bit vector a, we define the *Fourier coefficient with index* a ($\hat{f}(a)$) to be $\mathbf{E}_x[f(x)\cdot\chi_a(x)]$. Parseval's identity for the Fourier transform is $\mathbf{E}_x[f^2(x)] = \sum_a \hat{f}^2(a)$. For $f \in \{-1,+1\}$, this gives that $\sum_a \hat{f}^2(a) = 1$.

Notice that if a Fourier coefficient $\hat{f}(a)$ is reasonably large (bounded away from 0 by an inverse polynomial), then the corresponding parity function χ_a is a weak approximator to the function f. To see this, note that

$$\begin{aligned}\hat{f}(a) &= \mathbf{E}_x[f(x)\chi_a(x)]\\ &= \mathbf{Pr}_x[f(x) = \chi_a(x)] - \mathbf{Pr}_x[f(x) \neq \chi_a(x)]\\ &= 2\mathbf{Pr}_x[f(x) = \chi_a(x)] - 1.\end{aligned}$$

Therefore, if $|\hat{f}(a)| \geq \gamma$, then either the parity function χ_a or its negation is a $((1-\gamma)/2)$-approximator to f with respect to the uniform distribution. We say in this case that the parity χ_a (or its negation) is γ-*correlated* with f.

3. An initial separation of PAC and SQ-based algorithms

We begin this section by defining our notion of an SQ-based algorithm and discussing some of the implications of this definition. We then apply results from Blum *et al.* [2] and a very simple sample complexity argument to show a lower bound on the run time of any SQ-based algorithm for PAR. Next, time bounds on the recent Blum *et al.* [3] family of algorithms for learning parity are compared with the lower bound in the context of learning parity functions on the first $O(\log n)$ bits of n-bit input vectors. This comparison gives a separation between SQ-based algorithms and PAC algorithms resistant to inverse-polynomial classification noise. We then improve on this separation in various ways in subsequent sections.

3.1. SQ-based algorithms

We begin by formalizing the notion of an SQ-based algorithm that will be used in the lower bound proofs. The definition in some ways makes overly simple assumptions about the difficulty of simulating statistical queries, as discussed further below. However, these simplistic assumptions can be made without loss of generality for lower bound purposes and will streamline our later analysis.

Definition 1. An algorithm $\mathcal{A}$ is *SQ-based* if it is a PAC (example oracle) simulation of an SQ algorithm $\mathcal{S}$. Specifically, $\mathcal{A}$ is derived from $\mathcal{S}$ by replacing the ith query (g_i, τ_i) to the SQ oracle with the explicit computation of the sample mean of g_i over m_i (noiseless) examples obtained from the example oracle. Given a confidence δ, the m_i's must be chosen such that with probability at least $1-\delta$ all of the simulated statistical queries succeed at producing values within τ_i of the true expected values. The algorithm $\mathcal{A}$ therefore succeeds with probability at least $1-\delta$.

One simplifying assumption made in this definition is that the example oracle is noiseless, while the later PAC algorithms will be required to deal with noisy examples. Also notice that the definition does not require the SQ-based algorithm to compute an appropriate value of m_i (which would be necessary in a typical "real" simulation), but only to use an appropriate number of examples in its calculations.

Another point worth noting is that this definition does not exclude the possibility of simulating a number of queries (g_i, τ_i) as a batch rather than sequentially. That is, while the definition does require that all of the statistical queries be simulated, it does not specify the order in which they are simulated, and does not even preclude the computations for different query simulations being interleaved. The definition does, however, imply that each query should be simulated by computing the sum of g_i over m_i examples (this is the intention of the term "explicit computation" in the definition). That is, we do not allow any clever use of computations related to g_j to be used in the computation of the sample mean of g_i, $i \neq j$. This is because our goal is to capture the essence of a generic simulation of statistical queries, and any cleverness introduced would presumably be problem-specific rather than generic.

Finally, notice that this definition *does* allow for the reuse of examples between simulations of queries i and j, $i \neq j$. So the sample complexity of an SQ-based algorithm may be much less than $\sum_i m_i$. However, a key to our lower bound arguments is to note that the time complexity of an SQ-based algorithm is (at least) the sum of the times required to simulate all of the queries made by the algorithm, and therefore is at least $\sum_i m_i$.

3.2. A simple lower bound

We now consider SQ-based learning algorithms for PAR. Our analysis makes heavy use of Fourier-based ideas from Blum *et al.* [2], who showed that any class containing super-polynomially many distinct parity functions cannot be learned with a polynomial number of statistical queries having polynomial error tolerance. We will be interested in both the number of queries made and in the time required to simulate these queries with a (noiseless) PAC example oracle.

3.2.1. SQ learning of PAR

First, consider the SQ learnability with respect to the uniform distribution of the class PAR of parity functions. Let $f : \{0,1\}^n \to \{-1,+1\}$ be such a parity function–call it χ_b, where b is the n bit vector indicating which of the n input bits are relevant to χ_b–and let $f'(x) = (1 - f(x))/2$ be the $\{0,1\}$-valued version of f. A corollary of analysis in Blum *et al.* [2] then gives that for any function $g : \{0,1\}^{n+1} \to \{-1,+1\}$,

$$\begin{aligned}&\mathbf{E}_{z\sim U_n}[g(z, f'(z))]\\ &= \hat{g}(0_{n+1}) + \sum_{a\in\{0,1\}^n} \hat{g}(a1)\mathbf{E}_{z\sim U_n}[f(z)\chi_a(z)]\end{aligned}$$

where $a1$ represents the concatenation of the n-bit vector a and the bit 1. Furthermore, it follows by the orthogonality of the Fourier basis functions χ_a that the expectation $\mathbf{E}_{z\sim U_n}[f(z)\chi_a(z)] = \mathbf{E}_{z\sim U_n}[\chi_b(z)\chi_a(z)]$ is 0 unless $a = b$, in which case it is 1. So we have

$$\mathbf{E}_{z\sim U_n}[g(z, f'(z))] = \hat{g}(0_{n+1}) + \hat{g}(b1). \tag{1}$$

This means that if an SQ learning algorithm makes a query (g, τ) and $\tau \geq |\hat{g}(b1)|$ then the SQ oracle can return $\hat{g}(0_{n+1})$. But by a simple application of Parseval's identity, for any fixed function g as above there are at most τ^{-2} distinct Fourier coefficients of magnitude at least τ. Thus, a response of $\hat{g}(0_{n+1})$ by the SQ oracle to a query (g, τ) made by the SQ learner allows the learner to eliminate ("cover") at most τ^{-2} parity functions from further consideration (those corresponding to a's such that $|\hat{g}(a)| \geq \tau$). This leaves at least $2^n - \tau^{-2}$ parity functions, any one of which might be the target function.

Therefore, if our goal is to find the actual target function and all of our statistical queries use the same tolerance τ, in the worst case at least $2^n/\tau^2$ queries are required. This also implies that if we were to set the tolerance τ to $2^{-n/2}$, then conceivably we could learn the target parity function in a single statistical query. So sample complexity alone is not enough for our SQ-based lower bound argument; we also need to consider the number of examples required to simulate a query.

3.2.2. SQ-Based Learning of PAR

We can obtain a lower bound on the run time of any SQ-based algorithm for PAR by combining the above analysis with consideration of the number of examples needed to simulate a statistical query. Clearly, to simulate a statistical query (g_i, τ_i) requires $m_i = \Omega(1/\tau_i)$ examples; fewer than this means that even the discretization error of the sample mean is larger than τ_i. Thus, even in the case of a single statistical query being used with tolerance $2^{-n/2}$, an SQ-based algorithm will require time at least $\Omega(2^{n/2})$. We therefore have

Theorem 2. Let n represent the number of input bits of a function in PAR. Every SQ-based algorithm requires time $\Omega(2^{n/2})$ to PAC learn the class PAR with respect to the uniform distribution.

We will improve on this bound in section 4.

3.3. Noise-tolerant PAC algorithms for PAR

Blum, Kalai, and Wasserman [3], as part of their results, prove the following:

Theorem 3 (BKW). Let n represent the number of input bits of a function in PAR. For any integers a and b such that $ab = n$, the class PAR can be learned with respect to the uniform distribution under uniform random classification noise of rate η in time polynomial in $(1 - 2\eta)^{-(2^a)}$ and 2^b as well as the normal PAC parameters.

While Blum *et al.* used this theorem to analyze the case in which a is logarithmic in n, note that choosing a to be a constant gives us an algorithm with running time polynomial in the inverse noise rate and $O(2^{n/a})$ in terms of n. In particular, choosing $a > 2$ gives us an algorithm that has better asymptotic performance in n than the best possible SQ-based algorithm for PAR with respect to uniform. Furthermore, the algorithm's run time does not depend on the PAC parameter ϵ, as it produces a single parity function as its hypothesis, which is either a 0-approximator to the target f or is not at all correlated with f with respect to the uniform distribution. And the algorithm can be shown to have run time logarithmic in terms of $1/\delta$, as is typical of PAC algorithms.

Given this understanding of the Blum *et al.* results, we are ready for a formal comparison of this PAC algorithm with the SQ-based algorithm above.

3.4. Comparing the PAC and SQ-based algorithms

Comparing the bounds in Theorems 3 and 2, it is clear that the carefully crafted PAC algorithm of Blum *et al.* with a constant runs in polynomial time in all parameters on the class of parity functions over the first $O(\log n)$ input bits, and that this algorithm is generally faster than any SQ-based algorithm. However, the PAC algorithm bound includes the noise rate while our SQ analysis did not, so the PAC algorithm is not necessarily more efficient regardless of the noise rate. But note that if the noise term $1/(1-2\eta)$ is polynomially bounded in

n, say is $O(n^k)$ for some constant k, then there is a constant c such that the PAC algorithm on parity over the first $c \cdot k$ bits will be asymptotically more efficient than any SQ-based algorithm. This polynomial restriction on the noise term is relatively benign, particularly considering that n is exponentially larger than the size of the functions being learned. In any case, we have

Theorem 4. For any noise rate $\eta < 1/2$ such that $1/(1-2\eta)$ is bounded by a fixed polynomial in n, and for any confidence $\delta > 0$ such that $1/\delta$ is bounded by a fixed exponential in n, there exists a class $\mathcal{C}$ of functions and a constant k such that:

1. $\mathcal{C}$ can be PAC learned with respect to the uniform distribution with classification noise rate η in time $o(n^k)$ for some constant k; and
2. Every SQ-based algorithm for (noiseless) $\mathcal{C}$ with respect to the uniform distribution runs in time $\Omega(n^k)$.

We now turn to some improvements on this result. First, we show a stronger lower bound on the running time of SQ-based algorithms for PAR of $\Omega(2^n)$. We then show that a different algorithm for PAR that has running time dominated by $2^{n/2}$. In conjunction with the improved lower bound, this algorithm is also asymptotically faster than any SQ-based algorithm for parity with respect to the uniform distribution. We also note that this algorithm is robust against noise other than uniform random classification noise, and so appears to generalize somewhat the results obtained thus far.

4. A better lower bound on SQ-based algorithms

Our earlier analysis of the number of examples needed to simulate a statistical query was extremely simple, but coarse. Here we give a much more involved analysis which shows, perhaps somewhat surprisingly, that time fully $\Omega(2^n)$ is needed by any SQ-based algorithm to learn PAR. Our approach is to consider many cases of statistical queries and to show that in every case the number of parity functions "covered" by query i is $O(m_i)$, where m_i represents the number of examples needed to simulate i. Since by our earlier discussion essentially all 2^n parity functions must be covered by the algorithm, the overall result follows.

We will need several technical lemmas about the binomial distribution, which are stated and proved in the Appendix. Given these lemmas, we will now prove a general lower bound on the sample complexity—and therefore time complexity—needed to approximate certain random variables.

Lemma 5. Let X be a random variable in $\{0,1\}$ such that $\mathbf{Pr}[X=1]=p$, that is, a Bernoulli random variable with fixed mean p. Denote the mean value of a sample of X (of size m to be determined) by $\tilde{X}$. Assume that either or both of the following conditions hold for the parameters p, m, and λ: 1) $1/3 < p < 2/3$ and $0 < \lambda \leq 1/9$ (no condition on m); 2) m and p satisfy $mp(1-p) > 1$ and $0 < \lambda < p(1-p)/2$. Then for any $0 < \delta < 0.05$, a sample of size $m \geq p(1-p)/(2\lambda^2)$ is necessary to achieve $\mathbf{Pr}[|\tilde{X} - p| \leq \lambda] \geq 1 - \delta$.

Proof. Let $q = 1-p$ and call an m that guarantees that $\mathbf{Pr}[|\tilde{X} - p| \leq \lambda] \geq 1 - \delta$ *adequate*. We will begin by showing that if either of the conditions of the lemma is met and m is adequate then $mpq > 1$, which also implies $m \geq 5$. To see this, note that if $1/3 < p < 2/3$ and $mpq \leq 1$ then $m \leq 9/2$. For such small m, $\tilde{X}$ can take on only a small number of values, and the condition 1) bound on λ implies that at most one of these values of $\tilde{X}$ can be within λ of fixed p. A simple case analysis for $m = 2, 3, 4$ applying Lemma 11 shows that the probability of occurrence for the pertinent values of $\tilde{X}$ is much less than 0.95, and the case $m = 1$ is immediate. Thus if condition 1) is satisfied and m is adequate then it must be the case that $mpq > 1$.

Next, note that if the sample is of size m then $m\tilde{X}$, the sum of the random variables in the sample, has the binomial distribution $B(m,p)$ with mean mp. By Lemma 9, for $p < 1$, the maximum value of this distribution occurs at the first integer greater than $p(m+1)-1$, and this maximum value is shown in Lemma 13 to be no more than $0.41/\sqrt{mpq-1}$ for $mpq > 1$. Now the probability that $m\tilde{X}$ is within $\sqrt{mpq-1}$ of the true mean is just the integral of the distribution $B(m,p)$ from $mp - \sqrt{mpq-1}$ to $mp + \sqrt{mpq-1}$. Using the maximum on $B(m,p)$ given above, this probability is bounded above by 0.82. Therefore, we have that the probability that $|m\tilde{X} - mp| > \sqrt{mpq-1}$ is at least 0.18. In other words,

$$\mathbf{Pr}\left[|\tilde{X} - p| > \frac{\sqrt{mpq-1}}{m}\right] > 0.05.$$

So if either of the lemma's conditions is satisfied, then an adequate m must be such that $\sqrt{mpq-1}/m \leq \lambda$. This inequality is satisfied by all m if $\lambda \geq pq/2$, but it is easily seen that this relationship between λ and p is not possible is either of the conditions of the lemma holds. So solving this inequality, we see that it holds if either

$$m \leq \frac{pq - \sqrt{p^2q^2 - 4\lambda^2}}{2\lambda^2}$$

or

$$m \geq \frac{pq + \sqrt{p^2q^2 - 4\lambda^2}}{2\lambda^2}.$$

We will next show that no m satisfying the first of these inequalities is adequate, completing the proof.

Since we can assume that $\lambda \leq pq/2$, it follows that $m \leq (pq - \sqrt{p^2q^2 - 4\lambda^2})/(2\lambda^2)$ implies $m \leq 1/\lambda$. Now if $m \leq 1/\lambda$, since $\tilde{X}$ is an integer divided by m, there are at most two values of $\tilde{X}$ that differ from p by no more than λ. But since we know that $mpq > 1$, Lemma 13 gives that the maximum probability of any one value of the binomial distribution—and thus the maximum probability of any one value of $\tilde{X}$ occurring—is at most 0.46. Thus the maximum probability on two values of $\tilde{X}$ is at most 0.92, and a value of m less than $(pq - \sqrt{p^2q^2 - 4\lambda^2})/(2\lambda^2)$ cannot be adequate.

$\square$

With these lemmas in hand, we are ready to prove the main theorem of this section.

Theorem 6. Every SQ-based algorithm requires time $\Omega(2^n)$ to PAC learn the class PAR of parity functions with respect to the uniform distribution.

Proof. We begin by considering the SQ algorithm $\mathcal{S}$ that will be used to learn PAR, formalizing some of the earlier discussion. The algorithm must produce a good approximator h to the (noiseless) target—call it χ_b—which is one of the 2^n parity functions. By standard Fourier analysis based on Parseval's identity, if h is such that $\mathbf{Pr}[h = \chi_b] \geq 7/8$ then h cannot have a similar level of correlation with any other parity function. Specifically, recall that $\hat{h}(b) = 2\mathbf{Pr}[h = \chi_b] - 1$, and therefore $\hat{h}(b) \geq 3/4$ if $\mathbf{Pr}[h = \chi_b] \geq 7/8$. Since $\sum_a \hat{h}^2(a) = 1$ by Parseval's identity, we have that $\sum_{a \neq b} \hat{h}^2(a) \leq 7/16$, and therefore the maximal value for $\hat{h}(a)$ for any $a \neq b$ is less than $3/4$. Therefore, choosing $\epsilon < 1/8$ forces the SQ learning algorithm to produce a hypothesis that is well correlated with a single parity function.

Now as indicated above, each SQ query g of tolerance τ will either get a response that differs by at least τ from $\hat{g}(0_{n+1})$ or one that does not. We will call the former response *informative* and the latter *uninformative.* Also recall that each uninformative query $SQ(g,\tau)$ eliminates at most τ^{-2} parity functions from further consideration as possible targets.

Based on (1) and the subsequent analysis of statistical queries on PAR, we know that in the worst case a statistical query differs by more than τ_i from $\hat{g}_i(0_{n+1})$ only if $|\hat{g}_i(b1)| > \tau_i$, where χ_b is the target parity. Thus we define the *(worst-case) coverage C_i of a query $SQ(g_i,\tau_i)$* to be $C_i = \{a \mid |\hat{g}_i(a1)| > \tau_i\}$. Any SQ algorithm for the parity problem must in the worst case make queries that collectively cover all but one of the set of 2^n parity functions in order to with probability 1 successfully find a good approximator for $\epsilon < 1/8$. That is, in the worst case $|\cup_i C_i| = \Omega(2^n)$. Also note that in the worst case only the last of the covering queries—and possibly not even that one—will be informative.

Thus the SQ algorithm $\mathcal{S}$ can be viewed as at each step i choosing to make the query $SQ(g_i,\tau_i)$ in order to cover a certain set C_i of coefficients. That is, $\mathcal{S}$ first decides on the set C_i to be covered and then chooses g_i and τ_i in order to cover this set. We will assume without loss of generality that the SQ algorithm chooses τ_i for each query such that

$$\tau_i = \min\left\{|\hat{g}_i(a1)| \mid a \in C_i - \cup_{j=1}^{i-1} C_j\right\}.$$

That is, each τ_i is chosen to optimally cover its portion C_i of the function space. This change makes no difference in the total coverage after each query, and it will be seen below that it can only improve the run-time performance of the SQ-based algorithm.

Our goal now is to show that the time required by any SQ-based algorithm to simulate the queries made by any SQ algorithm for parity with respect to the uniform distribution is $\Omega(2^n)$. The analysis is similar in spirit to that of amortized cost: we show that each query $SQ(g_i,\tau_i)$ simulated by the SQ-based algorithm must "pay" an amount of running time proportionate to the coverage $|C_i|$.

We consider two different cases based on the nature of the queries made by the SQ algorithm. Let

$$p = \mathbf{E}_x\left[\frac{g(x,f(x))+1}{2}\right],$$

where f is the $\{0,1\}$ version of the target χ_b, so that p is the mean of a $\{0,1\}$-random variable. Then if the query g and target f are such that $1/3 < p < 2/3$, by Lemma 5 (or trivially, if τ is bounded by a large constant so that Lemma 5 does not apply) we need an estimate of the mean value of p over a sample of size $\Omega(1/\tau^2)$ in order to have high confidence that our estimate is within $\tau/2$ of the true mean (note that estimating p to within $\tau/2$ is equivalent to estimating $\mathbf{E}_x[g(x, f(x))]$ to within τ). And of course the estimate must be within $\tau/2$ if the algorithm is to decide whether or not the response to the query is informative. In other words, for queries satisfying the condition on p for this case, our SQ-based algorithm must pay a time cost of $\Omega(1/\tau^2)$. As already noted earlier, by Parseval's identity at most $1/\tau^2$ parity functions will be covered by a query with tolerance τ. Therefore, in this case, the running time of the SQ-based algorithm is proportionate to the coverage achieved.

On the other hand, assume without loss of generality that the SQ algorithm makes a query $SQ(g, \tau)$ such that $p < 1/3$ (the $p > 2/3$ case is symmetric). By (1) this implies that either the magnitude of $\hat{g}(0_{n+1})$ or of $\hat{g}(b1)$, or both, is larger than a constant. This in turn implies by Parseval's identity that there can be fewer additional "heavy" coefficients in g. In other words, even though we may be able to simulate the query $SQ(g, \tau)$ with a sample smaller than τ^2, we will also be covering fewer coefficients than we could if $\hat{g}(0_{n+1})$ and $\hat{g}(b1)$ were both small.

We formalize this intuition by again considering several cases. First, again given that the target parity is χ_b, note that

$$p = \frac{\hat{g}(0_{n+1}) + \hat{g}(b1) + 1}{2}.$$

We may assume that $|\hat{g}(b1)| \leq \tau$. This is certainly true if the query is uninformative, and it is true in the worst case for an informative query by our assumption about τ earlier. This then gives that $\hat{g}(0_{n+1}) \leq 2p - 1 + \tau$. Taking C to represent the coverage of $SQ(g, \tau)$, Parseval's identity gives that

$$|C| \leq \frac{1 - \hat{g}^2(0_{n+1})}{\tau^2}.$$

Now because $p < 1/3$, τ would need to be at least $1/3$ in order for $2p - 1 + \tau \geq 0$ to hold. But in this case only at most 9 coefficients can be covered, obviously with at least constant run-time cost. So we consider the case $2p - 1 + \tau < 0$. This implies that $\hat{g}^2(0_{n+1}) \geq (2p - 1 + \tau)^2$, and after some algebra and simplification gives

$$|C| \leq \frac{4pq}{\tau^2} + \frac{2}{\tau},$$

where $q = 1 - p$.

To convert this to a bound in terms of m, we consider two cases for the value of mpq. First, consider the case in which m is chosen such that $mpq < 1$, and assume for sake of contradiction that also $m < 1/(2\tau)$. Such a small m implies again that at most one of the possible values of the sample mean $\tilde{X}$ will be within τ of the true mean p. Furthermore, since $q > 2/3$, $mp < 3/2$, and it is easy to see from Lemma 9 that the sum $m\tilde{X}$ attains its maximum probability at either the value 0 or the value 1. Consider first the case where m and p are such that the maximum is at 1. The probability of drawing m consecutive 0's from a Bernoulli distribution with mean p is $(1-p)^m \geq \frac{10}{11}e^{-mp}$ (this form of the bound comes from [6]). Since $m \leq 3/(2p)$, this means that the probability of seeing all 0's is over 0.2. Thus the probability that $m\tilde{X} = 1$ is less than 0.8, so a larger m would be required in order for the sample mean to be within τ of the true mean with more than constant probability. If instead m and p are such that the maximum of the binomial is at 0, it must be that $mp < 1$. We consider a Bernoulli random variable with mean $p + \tau$. If m examples are chosen from this random variable then with probability at least $\frac{10}{11}e^{-m(p+\tau)}$ all m examples are 0's. This quantity is again over 0.2 if $m < 1/(2\tau)$. Thus we would need many more examples than $1/(2\tau)$ in order to have better than constant confidence that our sample came from a distribution with mean p rather than one with mean $p + \tau$.

We conclude from this that if $p < 1/3$ and $mpq < 1$ then it must be that $m > 1/(2\tau)$ in order for the PAC algorithm's sampling to succeed with sufficiently high probability. Thus we get that in this case,

$$|C| \leq \frac{4pq}{\tau^2} + \frac{2}{\tau} \leq \frac{4}{m\tau^2} + 4m \leq 20m.$$

Therefore, once again the coverage is proportional to the sample size used.

Finally, if $p < 1/3$ and $mpq \geq 1$ then we consider two last cases. First, if $\tau \geq pq$ then $4pq/\tau^2 \leq 4/\tau$ and also $1/\tau \leq m$. Therefore, in this case, $|C| \leq 6m$. On the other hand, if $p < 1/3$, $mpq \geq 1$, and $\tau < pq$ then we can apply Lemma 5 with $\lambda = \tau/2$. This gives that $m \geq 2pq/\tau^2$, and combining this with $mpq \geq 1$

implies that $m \geq \sqrt{2}/\tau$. Combining these bounds gives

$$|C| \leq \frac{4pq}{\tau^2} + \frac{2}{\tau} \leq (2+\sqrt{2})m.$$

So in all cases run time is proportional to the coverage $|C|$, and the total coverage $|\cup_i C_i|$ has already been shown to be $\Omega(2^n)$ in the worst case. □

5. A more robust noise-tolerant PAC algorithm for PAR

In this section we present another noise-tolerant PAC algorithm for learning the class PAR of parity functions with respect to the uniform distribution. While the algorithm's running time is $O(2^{n/2})$ in terms of n, by the lower bound on SQ-based algorithms of the previous section and an analysis similar to that of Theorem 4, the algorithm can be shown to be asymptotically faster than any SQ-based algorithm for this problem. Furthermore, the algorithm can be shown to be tolerant of a wide range of noise, not just uniform random classification noise.

Our algorithm is based on one by Goldreich and Levin [8] that uses membership queries. We first review the Goldreich-Levin algorithm and then show how to remove the need for membership queries when large samples are available.

5.1. Goldreich-Levin weak parity algorithm

A version of the Goldreich-Levin algorithm [8] is presented in Figure 1. The algorithm is given a membership oracle $MEM(f)$ for a target function $f : \{0,1\}^n \rightarrow \{-1,+1\}$ along with a threshold θ and a confidence δ. Conceptually, the algorithm begins by testing to see the extent to which the first bit is or is not *relevant* to f. A bit is particularly relevant if it is frequently the case that two input vectors that differ only in this bit produce different values of f. The function call

$$\texttt{WP-aux}(1, 0, n, MEM(f), \theta, \delta)$$

will, with high probability, return the empty set if the first bit is particularly relevant. To see this, notice that if the first bit is very relevant, then with high probability over uniform choice of $(n-1)$-bit x and 1-bit y and z,

$$f(yx)f(zx)\chi_0(y \oplus z) = f(yx)f(zx)$$

```
Invocation: $S \leftarrow \texttt{WP}(n, MEM(f), \theta, \delta)$
Input: Number $n$ of inputs to function $f : \{0,1\}^n \to \{-1,+1\}$; membership
oracle $MEM(f)$; $0 < \theta \leq 1$; $\delta > 0$
Output: Set $S$ of $n$-bit vectors such that, with probability at least $1-\delta$, every
$a$ such that $|\hat{f}(a)| \geq \theta$ is in $S$, and for every $a \in S$, $|\hat{f}(a)| \geq \theta/\sqrt{2}$.

 1. return $\texttt{WP-aux}(1, 0, n, MEM(f), \theta, \delta) \cup \texttt{WP-aux}(1, 1, n, MEM(f), \theta, \delta)$

Invocation: $S \leftarrow \texttt{WP-aux}(k, b, n, MEM(f), \theta, \delta)$
Input: Integer $k \in [1, n]$; $k$-bit vector $b$; number $n$ of inputs to function $f$ :
$\{0,1\}^n \to \{-1,+1\}$; membership oracle $MEM(f)$; $0 < \theta \leq 1$; $\delta > 0$
Output: Set $S$ of $n$-bit vectors such that, with probability at least $1-\delta$, for
every $a$ such that the first $k$ bits of $a$ match input vector $b$ and $|\hat{f}(a)| \geq \theta$, $a$ is in
$S$, and for every $a \in S$, $|\hat{f}(a)| \geq \theta/\sqrt{2}$.

 1. $s \leftarrow 0$; $m \leftarrow 32\theta^{-4}\ln(4n/\delta\theta^2)$
 2. for $m$ times do
 3.    Draw $x \in \{0,1\}^{n-k}$, $y, z \in \{0,1\}^k$ uniformly at random.
 4.    $s \leftarrow s + f(yx)f(zx)\chi_b(y \oplus z)$
 5. enddo
 6. $\mu' \leftarrow s/m$
 7. if $\mu' < 3\theta^2/4$ then
 8.    return $\emptyset$
 9. else if $k = n$ then
10.    return $\{b\}$
11. else
12.    return $\texttt{WP-aux}(k+1, b0, n, MEM(f), \theta, \delta) \cup \texttt{WP-aux}(k+1, b1, n, MEM(f), \theta, \delta)$
13. endif
```

Figure 1. The `WP` weak-parity algorithm.

will be very small. This is because with probability $\frac{1}{2}$ $y = z$ and with probability $\frac{1}{2}$ $y \neq z$, and when $y \neq z$ the high relevance of the first bit implies that frequently $f(yx) \neq f(zx)$. Thus approximately half the time we expect that the product is 1 and half -1, giving an expected value near 0. As the loop in `WP-aux` is estimating this expected value, we expect that the condition at line 7 will typically be satisfied.

On the other hand, consider the function

$$\texttt{WP-aux}(1, 1, n, MEM(f), \theta, \delta).$$

Now the loop in `WP-aux` is estimating the expected value of

$$f(yx)f(zx)\chi_1(y \oplus z).$$

Since $\chi_1(y \oplus z)$ is 1 when $y = z$ and -1 otherwise, we now expect a value for μ' very near 1. Thus, in the situation where the first bit is highly relevant, we expect that every Fourier index a in the set S returned by `WP` will begin with a 1. To determine exactly what these coefficients are, the `WP-aux` function calls itself recursively, this time fixing the first two bits of the subset of coefficients considered to either 10 or 11.

Thus we can intuitively view the `WP` algorithm as follows. It first tests to see the extent to which "flipping" the first input bit changes the value of the function. If the bit is either highly relevant or highly irrelevant then half of the coefficients can be eliminated from further consideration. Similar tests are then performed recursively on any coefficients that remain as candidates, with each recursive call leading to one more bit being fixed in a candidate parity index a. After n levels of recursion all n bits are fixed in all of the surviving candidate indices; these indices are then the output of the algorithm. With probability at least $1 - \delta$, this list contains indices of all of the parity functions that are θ-correlated with f and no parity function that is not at least $(\theta/\sqrt{2})$-correlated, and runs in time $O(n\theta^{-6}\log(n/\delta\theta))$ [9].

Furthermore, a θ^{-2} factor comes from the fact that in general the algorithm must maintain up to this many candidate sets of coefficients at each level of the recursion. In the case of learning parity, it can be shown that with high probability only one candidate set will survive at each level. Therefore, when learning PAR, the running time becomes $O(n\theta^{-4}\log(n/\delta\theta))$.

It is well known that this algorithm can be used to weakly learn certain function classes with respect to the uniform distribution using parity functions as hypotheses [11,9]. However, we note here that it can also be used to (strongly) learn PAR with respect to the uniform distribution in the presence of random classification noise. Let f^η represent the randomized Boolean function produced by applying classification noise of rate η to the target parity function f. That is, assume that on each call to $MEM(f^\eta)$ the oracle returns the value of $MEM(f)$ with noise of rate η applied, and noise is applied independently at each call to

$MEM(f^\eta)$. Let $\mathbf{E}_\eta[\cdot]$ represent expectation with respect to noise of rate η applied to f. Then it is straightforward to see that

$$\mathbf{E}_\eta[\mathbf{E}_{x\sim U_n}[f^\eta(x)\chi_b(x)]] = (1-2\eta)\hat{f}(b).$$

Furthermore, the only use the `WP` algorithm makes of $MEM(f)$ is to compute μ' in `WP-aux`, which is an estimate of $\mathbf{E}_{x\sim U_{n-k}, y\sim U_k, z\sim U_k}[f(yx)f(zx)\chi_b(y\oplus z)]$. Replacing f by f^η in this expression, we get an expectation that also depends on the randomness of f^η. However, since classification noise is applied independently to $f^\eta(yx)$ and $f^\eta(zx)$—even if $y = z$, as long as the values $f^\eta(yx)$ and $f^\eta(zx)$ are returned from separate calls to $MEM(f^\eta)$—it follows that $\mathbf{E}_{\eta,x,y,z}[f^\eta(yx)f^\eta(zx)\chi_b(y\oplus z)] = (1-2\eta)^2\mathbf{E}_{x,y,z}[f(yx)f(zx)\chi_b(y\oplus z)]$, where the first expectation is over the randomness of f^η as well as the inputs. Finally, none of the analysis [9] used to prove properties about the output of the `WP` algorithm precludes the target f from being randomized; independence of samples and the range of values produced by f are the key properties used in the proof, and these are the same for f^η as they are for the deterministic f.

In short, running `WP` with $\theta = 1-2\eta$ and membership oracle $MEM(f^\eta)$ representing a target $f = \chi_b$ will, with high probability, result in an output list that contains b. Furthermore, since by Parseval's identity and the above analysis $\mathbf{E}_\eta[\mathbf{E}_{x\sim U_n}[f^\eta(x)\chi_a(x)]] = 0$ for all $a \in \{0,1\}^n$ such that $a \neq b$, with high probability only index b will appear in the output list.

Of course, a noisy membership oracle as above can be used to simulate a noiseless oracle by simple resampling, so the observation that the `WP` algorithm can be used to learn PAR in the presence of noise is not in itself particularly interesting. However, we next show that we can simulate the `WP` algorithm with one that does not use membership queries, giving us a uniform-distribution noise-tolerant PAC algorithm for PAR that will be shown to be relatively efficient compared with SQ-based algorithms.

5.2. Removing the membership oracle

As discussed above, Goldreich-Levin uses the membership oracle $MEM(f)$ to perform the sampling needed to estimate the expected value $\mathbf{E}_{x,y,z}[f(yx)f(zx)\chi_b(y\oplus z)]$. At the first level of the recursion, $|x| = n-1$, we (conceptually) "flip" the first bit (technically, y and z will often have the same value, but it is the times when they differ that information about a deterministic function is actually obtained). Notice that we would not need the membership oracle if we had—or could simulate—a sort of example oracle that could produce pairs of examples $(\langle yx, f(yx)\rangle, \langle zx, f(zx)\rangle)$ drawn according to the uniform distribution over x, y, and z. We will denote by D_1 the induced distribution over pairs of examples.

Lemma 8 in the Appendix proves that if $2^{k/2+1}$ k-bit vectors are drawn uniformly at random, then with probability at least 1/2 one vector will occur twice in the sample. Therefore, if we draw a sample S of examples of f of size $2^{(n+1)/2}$ then with probability at least 1/2 a pair of examples will be drawn having the same final $n-1$ bits. And for any such pair, it is just as likely that the first bits of the two functions will differ as it is that they will be the same. Thus, with probability at least 1/2 we can simulate one draw from D_1 by creating a list of all pairs of examples in S that share the same values on the final $n-1$ attributes and choosing one of these pairs uniformly at random.

Slightly more precisely, we will draw $2^{(n+1)/2}$ examples and record in a list each $n-1$ bit pattern that appears at the end of more than one example; each such pattern appears once in the list. We then choose one such pattern uniformly at random from the list. Finally, from among the examples ending with the chosen pattern, we select two examples uniformly at random without replacement.

Note that the probability of selecting any particular $n-1$ bit pattern from the list is the same as selecting any other pattern. Therefore, we are selecting this pattern—corresponding to the choice of x in the expectation above—uniformly at random. Note also that our method of choosing the two examples ending with this pattern guarantee that the first bits of these examples—corresponding to y and z above—are independently and uniformly distributed. Therefore, with probability at least 1/2, this procedure simulates D_1. Furthermore, if a particular set S fails to have an appropriate pair of examples, we can easily detect this condition and simply draw another set of examples. With probability at least $1-\delta$, we will obtain an appropriate set within $\log(1/\delta)$ draws.

Now let D_2 represent the the probability distribution on pairs of examples corresponding to choosing an $(n-2)$-bit x uniformly at random, choosing 2-bit y and z uniformly at random, and producing the pair $(\langle yx, f(yx)\rangle, \langle zx, f(zx)\rangle)$. The procedure above can be readily modified to simulate a draw from D_2 using draws of only $2^{n/2}$ uniform examples. Similar statements can be made for the other distributions to be simulated.

In short, we have shown how to simulate draws from any of the probability distributions induced by the weak parity procedure, at a cost of roughly $2^{n/2}$

draws from the uniform distribution. Also note that the argument above has nothing to do with the labels of the examples, and so holds for randomized f^η as well as for deterministic f. Thus we have the following theorem.

Theorem 7. For any noise rate $\eta < 1/2$ and confidence $\delta > 0$, the class PAR of parity functions can be learned in the uniform random classification-noise model with respect to the uniform distribution in time $O(2^{n/2} n \theta^{-4} \log^2(n/\delta\theta))$, where $\theta = 1 - 2\eta$.

5.3. Improved noise tolerance

Consider the following noise model: given a target parity function $f = \chi_a$, the noise process is allowed to generate a deterministic noisy function f^η (η here denotes only that the function is a noisy version of f and not a uniform noise rate) subject only to the constraints that $\mathbf{E}_x[f^\eta(x)\chi_a(x)] \geq \theta$ for some given threshold θ, and for all $b \neq a$, $\mathbf{E}_x[f^\eta(x)\chi_b(x)] < \theta/\sqrt{2}$. That is, f^η must be at least θ-correlated with f and noticeably less correlated with every other parity function. It should be clear that the algorithm of this section, given θ, can strongly learn PAR with respect to uniform in this fairly general noise setting in time $O(2^{n/2} n \theta^{-6} \log^2(n/\delta\theta))$. The Blum *et al.* algorithm, on the other hand, seems to be more dependent on the uniform random classification noise model. However, a version of their algorithm is distribution independent, which raises the interesting question of whether or not the modified `WP` algorithm above can also be made distribution independent.

Acknowledgements

This work was initially inspired by work of Dan Ventura on the quantum learnability of the parity class. The author also thanks Tino Tamon for helpful discussions. Thanks also to the anonymous referee of the conference proceedings version of this paper who pointed out how the Blum, Kalai, and Wasserman algorithms could be applied to the problem considered in this paper.

References

[1] Javed A. Aslam and Scott E. Decatur. Specification and simulation of statistical query algorithms for efficiency and noise tolerance. *Journal of Computer and System Sciences*,

56(2):191–208, April 1998.

[2] 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 26th Annual ACM Symposium on Theory of Computing*, pages 253–262, 1994.

[3] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the Statistical Query model. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 435–440, 2000.

[4] Béla Bollobás. *Random Graphs.* Academic Press, 1985.

[5] Nader Bshouty and Christino Tamon. On the Fourier spectrum of monotone functions. *Journal of the ACM*, 43(4):747–770, July 1996.

[6] N. Cesa-Bianchi, E. Dichterman, P. Fischer, E. Shamier, and H.U. Simon. Sample-efficient strategies for learning in the presence of noise. *Journal of the ACM*, 46(5):684–719, 1999.

[7] Yoav Freund and Robert E. Schapire:. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997. First appeared in EuroCOLT '95.

[8] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the Twenty First Annual ACM Symposium on Theory of Computing*, pages 25–32, 1989.

[9] Jeffrey Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *Journal of Computer and System Sciences*, 55(3):414–440, 12 1997. Earlier version appeared in *Proceedings of the 35th Ann. Symp. on Foundations of Computer Science*, pages 42–53, 1994.

[10] Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 392–401, 1993.

[11] Eyal Kushilevitz and Yishay Mansour. Learning decision trees using the Fourier spectrum. *SIAM Journal on Computing*, 22(6):1331–1348, December 1993. Earlier version appeared in *Proceedings of the Twenty Third Annual ACM Symposium on Theory of Computing*, pages 455–464, 1991.

[12] Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM*, 40(3):607–620, July 1993. Earlier version appeared in *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 574–579, 1989.

[13] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5:197–227, 1990.

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

APPENDIX

This is a lemma that is needed to prove Theorem 7.

Lemma 8. If $2^{k/2+1}$ k-bit vectors are drawn uniformly at random, then with probability at least $1 - 1/e$ one vector will occur twice in the sample.

Proof. The proof is similar to that of the birthday paradox. First, note that the probability that two randomly drawn k-bit vectors do not match is $1-1/2^k$. The probability that a third vector does not match either of the first two is $1-2/2^k$, and thus the probability that none of the vectors match is $(1-1/2^k)(1-2/2^k)$. In general, if $2^{k/2+1}$ vectors are drawn, then the probability that none match is

$$\prod_{i=1}^{2^{k/2+1}} 1 - \frac{i}{2^k} \leq \prod_{i=2^{k/2}}^{2 \cdot 2^{k/2}} 1 - \frac{i}{2^k} \leq \left(1 - \frac{2^{k/2}}{2^k}\right)^{2^{k/2}} \leq \frac{1}{e}.$$

Thus the probability of a match is at least $1 - 1/e$. □

The following technical lemmas about the binomial distribution are used in Section 4. The first lemma is well known (for a proof, see, *e.g.*, [4]).

Lemma 9. Let $B(k; m, p) \equiv \binom{m}{k} p^k (1-p)^{m-k}$ represent the binomial distribution. For m and $p < 1$ fixed, the maximum value of the binomial distribution occurs at the first integer $k = k_m$ greater than $p(m+1) - 1$.

Lemma 10. For $m > 0$ and $p < 1$ fixed, the maximum value of the integer k_m at which the maximum of the binomial distribution occurs is such that $(k_m/m)(1 - k_m/m) \geq p(1-p) - 1/m$.

Proof. By Lemma 9, the k_m that maximizes $B(k; m, p)$ satisfies $p(m+1) - 1 < k_m \leq p(m+1)$. Some algebra and simplification then gives the result. □

Lemma 11. For any integer $m > 1$, integer $0 < k < m$, and real $0 \leq p \leq 1$,

$$B(k; m, p) \leq B(k; m, k/m).$$

Proof. We want to know the value of p that maximizes B for fixed k and m satisfying the constraints above. We can find the maximizing value of p by examining values of p that make the derivative dB/dp zero. It is easily shown that the derivative is zero only at $p = k/m$, $p = 0$, and $p = 1$ and that $p = k/m$ is the only local maximum. □

Lemma 12. For any integer $m \geq 5$, integer $0 < k < m$, and real $1/3 \leq p \leq 2/3$,

$$B(k; m, p) \leq \frac{0.406}{\sqrt{mp(1-p) - 1}}.$$

Proof. We will maximize $B(k; m, p)$ over both k and p, then develop a bound on this maximized quantity.

We already showed in Lemma 11 that maximizing the binomial distribution over p gives $p = k/m$. Using a fairly tight version of the Stirling bound

$$m^m e^{-m} \sqrt{2\pi m} e^{1/(12m+1)} \leq m! \leq m^m e^{-m} \sqrt{2\pi m} e^{1/12m}$$

we get that for any $0 < k < m$ and $m \geq 1$,

$$\binom{m}{k} \leq \frac{e^{1/12m}}{\sqrt{2\pi m(1-k/m)(k/m)}(1-k/m)^{m-k}(k/m)^k}$$

and for $m \geq 5$ this gives

$$B(k; m, k/m) \leq \frac{0.406}{\sqrt{m(k/m)(1-k/m)}}.$$

Thus we have that

$$\begin{aligned} B(k; m, p) &\leq B(k_m; m, p) \\ &\leq B(k_m; m, k_m/m) \\ &\leq \frac{0.406}{\sqrt{m(k_m/m)(1-k_m/m)}} \\ &\leq \frac{0.406}{\sqrt{mp(1-p) - 1}} \end{aligned}$$

where we applied Lemma 10 in the final step. Finally, note that $\sqrt{mp(1-p) - 1}$ is well-defined for all m and p as constrained by the statement of the lemma. □

Similarly, we can show

Lemma 13. For any integer m and any real $0 \leq p \leq 1$, $mp(1-p) > 1$ implies that $B(k; m, p) \leq 0.46$. Furthermore, in terms of p,

$$B(k; m, p) \leq \frac{0.406}{\sqrt{mp(1-p) - 1}}.$$

Proof. First, note that for all $0 < p < 1$, $1/(p(1-p)) \geq 4$. Thus if $m > 1/(p(1-p))$ then $m \geq 5$, and therefore Lemma 12—and its proof—apply if $mp(1-p) > 1$.

Therefore, if $mp(1-p) > 1$ then $B(k; m, p) \leq 0.406/\sqrt{m(k_m/m)(1-k_m/m)} \leq 0.406/\sqrt{mp(1-p)-1}$. Next, notice that if $m \geq 1/p$ then by Lemma 9 the value $k = k_m$ at which the binomial $B(k; m, p)$ is maximized must be at least 1. Similarly, if $m \geq 1/(1-p)$, $k_m \leq m-1$. It is easy to see that, as a function of k_m, $\sqrt{m(k_m/m)(1-k_m/m)}$ is minimized at its extremes, which we have just shown to be at worst $k_m = 1$ and $k_m = m-1$ if $mp(1-p) > 1$. Plugging either $k_m = 1$ or $k_m = m-1$ into $0.406/\sqrt{m(k_m/m)(1-k_m/m)}$ with $m \geq 5$ gives a value less than 0.46. $\square$