Agnostic Learning from Tolerant Natural Proofs

Marco L. Carmosino[1], Russell Impagliazzo[2], Valentine Kabanets[3], and Antonina Kolokolova[4]

1 **Department of Computer Science, University of California San Diego, La Jolla, CA, USA**
`mcarmosi@cs.ucsd.edu`

2 **Department of Computer Science, University of California San Diego, La Jolla, CA, USA**
`russell@cs.ucsd.edu`

3 **School of Computing Science, Simon Fraser University, Burnaby, BC, Canada**
`kabanets@cs.sfu.ca`

4 **Department of Computer Science, Memorial University of Newfoundland, St. John's, NL, Canada**
`kol@mun.ca`

Abstract

We generalize the "learning algorithms from natural properties" framework of [4] to get *agnostic* learning algorithms from natural properties with extra features. We show that if a natural property (in the sense of Razborov and Rudich [28]) is useful also against functions that are *close* to the class of "easy" functions, rather than just against "easy" functions, then it can be used to get an agnostic learning algorithm over the uniform distribution with membership queries.

- For $\mathsf{AC}^0[q]$, any prime q (constant-depth circuits of polynomial size, with AND, OR, NOT, and MOD_q gates of unbounded fanin), which happens to have a natural property with the requisite extra feature by [27, 31, 28], we obtain the first agnostic learning algorithm for $\mathsf{AC}^0[q]$, for every prime q. Our algorithm runs in randomized quasi-polynomial time, uses membership queries, and outputs a circuit for a given boolean function $f\colon \{0,1\}^n \to \{0,1\}$ that agrees with f on all but at most $(\mathsf{poly}\log n)\cdot \mathsf{opt}$ fraction of inputs, where opt is the relative distance between f and the closest function h in the class $\mathsf{AC}^0[q]$.
- For the ideal case, a natural proof of strongly exponential correlation circuit lower bounds against a circuit class $\mathcal{C}$ containing $\mathsf{AC}^0[2]$ (i.e., circuits of size $\exp(\Omega(n))$ cannot compute some n-variate function even with $\exp(-\Omega(n))$ advantage over random guessing) would yield a polynomial-time query agnostic learning algorithm for $\mathcal{C}$ with the approximation error $O(\mathsf{opt})$.

1998 ACM Subject Classification F.1.3 Complexity Measures and Classes

Keywords and phrases agnostic learning, natural proofs, circuit lower bounds, meta-algorithms, $\mathsf{AC}^0[q]$, Nisan-Wigderson generator

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2017.35

1 Introduction

Recently many new connections have been discovered between the two complementary domains: proving circuit lower bounds and designing meta-algorithms for the corresponding circuit classes (see, e.g., [30, 33, 34, 15, 16, 5, 4]). In particular, [4] shows that a natural property (in the sense of Razborov and Rudich [28]) for a (sufficiently powerful) circuit class Λ yields an efficient PAC learning algorithm for the same circuit class, under the uniform distribution, with membership queries; this approach led to a first learning algorithm for the class $\mathsf{AC}^0[q]$ (of constant-depth circuits with AND, OR, NOT, and modulo q gates), for every prime q.

The "learning algorithms from natural proofs" technique of [4] applies only to *realizable case* learning: if a function f is computed exactly by an appropriate circuit class Λ for which there is a natural proof of a circuit lower bound, then we can learn f using membership queries in time dependent on the strength of the circuit lower bound. A more realistic learning model is *agnostic learning*, where we select some "touchstone" class Λ and attempt to find a hypothesis that isn't "too far off" from the best Λ-approximation to the target function.

We show that, even in this agnostic setting, we can (somewhat generically) obtain learning algorithms from natural proofs. We instantiate this framework to give the first membership-query agnostic learning algorithm over the uniform distribution for $\mathsf{AC}^0[q]$, the class of constant-depth circuits of polynomial-size with unbounded fanin AND, OR, NOT, and MOD_q gates. Previously, only the case of AC^0 circuits was known (albeit for an agnostic algorithm without membership queries, and with better approximation error) [19] (based on the LMN algorithm of [23]).

▶ **Theorem 1** ($\mathsf{AC}^0[q]$ agnostic learning)**.** *Let q be any prime. There is a randomized quasi-polynomial-time algorithm such that, given oracle access to a function $f\colon \{0,1\}^n \to \{0,1\}$ that agrees with some unknown function in $\mathsf{AC}^0[q]$ on at least $1-\beta$ fraction of inputs (for some non-negligible $\beta > 0$), the algorithm outputs a circuit that computes f on all but at most $\mathsf{poly}(\log n)\cdot\beta$ fraction of inputs.*

As an interesting special case, we get a quasipolynomial-time agnostic learning algorithm for n-variate polynomials over $\mathbf{GF}(q)$ of low degree (say, at most $\mathsf{poly}(\log n)$), for prime $q \geq 2$ (as every polynomial of degree d is computable by an $\mathsf{AC}^0[q]$ circuit of size $O(n^d)$). Before our result, no such learning algorithm for polynomials was known.

For an algorithm with error $c(n)\cdot\beta$, for some function c, we call the factor $c(n)$ the *weakness* parameter of the learning algorithm. It is desirable to have $c(n) = 1$. Our algorithm for $\mathsf{AC}^0[q]$ above has weakness $\mathsf{poly}(\log n)$. In general, we have a trade-off between the quality of a natural property for the circuit class, and the quality of the resulting agnostic learning algorithm for the same class. For simplicity, we state here just the result for the best-case scenario; see Theorem 11 below for the fully general statement.

▶ **Theorem 2** (Ideal-case trade-off)**.** *Suppose there is a natural property for a circuit class $\mathcal{C} \supseteq \mathsf{AC}^0[2]$ that is useful against functions that agree on $1/2 + \exp(-\Omega(n))$ of inputs with some function of $\mathcal{C}$-circuit complexity $\exp(\Omega(n))$. Then, for some constant $c > 0$, there is a polynomial-time query agnostic learning algorithm for $\mathcal{C}$ with weakness c.*

Theorem 2 yields a "search-to-decision" reduction for a version of the Minimal Circuit Size Problem (MCSP). Define the Minimal Approximate Circuit Size Problem (MACSP) as follows: Given a truth table of an n-variate boolean function f, and parameters $s \in \mathbb{N}$ and $\delta \in [0,1]$, decide if there exists a boolean circuit C of size at most s that agrees with f on all but at most δ fraction of inputs. (MCSP is a special case of MACSP for $\delta = 0$.) Clearly, if MACSP is easy (say, in P), then, for a given size bound s (our "budget"), we can determine the best approximation parameter δ for every given truth table of a boolean function f. But, since MACSP is an ideal-case tolerant natural property for general circuits, we get by Theorem 2 that a polynomial-time algorithm for MACSP would yield a polynomial-time algorithm to actually find a circuit of size $\mathsf{poly}(s)$, with an approximation guarantee $O(\delta)$.[1]

[1] In [4], a similar "search-to-decision" reduction was given for MCSP: if a given boolean function f is

Another way to interpret Theorem 2 is as follows. If MACSP is in P, then, given oracle access to a boolean function f, and a budget $s \in \mathbb{N}$, we can learn, in polynomial time, a circuit of size $\mathsf{poly}(s)$ that agrees with f on all but at most $O(\delta)$ fraction of inputs, where δ is the error of the best size s circuit for f. That is, we can learn essentially the best possible circuit for f, given our budget s on the circuit size.

1.1 Our approach

The key observation in adapting to the agnostic setting is that many natural properties contain even more useful distinguishers than required for realizable-case learning. As defined by [28], the distinguisher from a natural property rejects truth tables that are exactly computed by Λ-circuits. But existing natural properties give us something even stronger: they reject truth tables which are just *close* to those computed by Λ-circuits. Using this observation and the same "play to lose" distinguisher-to-predictor reduction as in [4], we obtain agnostic learning algorithms from such natural properties.

More precisely, we show that if a natural property for a circuit class Λ (containing $\mathsf{AC}^0[q]$) is *tolerant* in the sense that it distinguishes from random the truth tables of functions "close" to the class Λ (of "large" circuit complexity), then it can be used to get an agnostic membership-query algorithm for learning Λ. We argue that such a tolerant natural property exists for $\mathsf{AC}^0[q]$ [27, 31, 28], which is then used to prove our Theorem 1. For $\mathsf{AC}^0[2]$, we need to dig inside the arguments of [27], and show that his original circuit lower bound proof does yield a certain tolerant natural property. For $\mathsf{AC}^0[q]$, for prime $q > 2$, we actually need to re-do the "natural proof" argument of [28] by adapting it to the case of $\mathbf{GF}(q)$-valued functions (rather than boolean functions). Not only does it allow us to get tolerant natural properties for $\mathsf{AC}^0[q]$, but also simplifies and streamlines the analysis in [4] of the learning algorithm for $\mathsf{AC}^0[q]$.

By definition, tolerant natural properties can be used for proving *average-case* circuit lower bounds (as opposed to the worst-case circuit lower bounds implied by standard natural properties). Thus the main message of the present paper can be summarized as follows:

Natural proofs of *average-case* circuit lower bounds imply *agnostic* learning algorithms!

In contrast, the main result of [4] says that natural proofs of worst-case circuit lower bounds imply standard (non-agnostic) learning algorithms.

1.2 Our techniques

We build upon the framework of [4] who use a natural property for a given circuit class Λ in order to devise a learning algorithm for the same class. Recall that a natural property (in the sense of [28]) is an efficient algorithm that tells apart truth tables of functions in the class Λ (of some "large" circuit complexity u) from those of random functions. To learn a function $f \in \Lambda$, for some circuit class Λ that has an associated natural property, the idea is to apply (as only a thought experiment!) an appropriate "function generator" that maps f to a family of functions all of which are "easy" (of small Λ circuit complexity) and so will be rejected by

exactly computable by a polynomial-size circuit, then one can find a polynomial-size circuit *approximately* computing f, given a polynomial-time algorithm for MCSP. In contrast, here we say that if f can be non-trivially *approximated* by a polynomial-size circuit, we can find another polynomial-size circuit that achieves the *same approximation error* up to a constant factor, given a polynomial-time algorithm for MACSP.

the natural property for the class. Thus an efficient algorithm from the natural property acts as a distinguisher "breaking" the function generator. If the function generator has an "efficient reconstruction" property, meaning that a distinguisher for the generator can be used to build a small circuit approximately computing the original function f, we get a learning algorithm for f. Thus, the actual learning algorithm is using the natural property algorithm as a distinguisher, and applies the efficient reconstruction procedure (associated with the given function generator) to build a circuit approximating f. Usually, such a reconstruction procedure requires oracle access to the function generator; if, however, the function generator is "local" in the sense that such oracle access to the generator can be efficiently reduced to oracle access to the original function f, one gets a query learning algorithm for the concept class Λ.

To adapt this approach to the case of agnostic learning, where a function f to be learned is not in the class Λ, but rather just somewhat close to the class, we need to satisfy the following requirements:

1. the outputs of the function generator applied to f must be close to the class Λ (of some circuit size u), and
2. the natural property for Λ must reject not only functions in Λ (of size u), but also functions that are close to those.

We call natural properties satisfying condition (2) above *tolerant*. We say that a natural property has ρ-tolerant u-usefulness for the circuit class Λ if it rejects all truth tables of functions that agree with some function in $\Lambda[u]$ (computable by a Λ circuit of size u) on all but at most ρ fraction of inputs. We show that the natural property for the circuit class $\mathsf{AC}^0[2]$ from [27] is in fact ρ-tolerant, for some small but nontrivial $\rho > 0$, and with large (weakly-exponential) usefulness u.

With tolerant natural properties in hand, we turn to requirement (1) above: getting the truth tables output by the function generator on a given function f to be close to those from the circuit class $\Lambda[u]$. We need to take a closer look at the function generator used in [4]. It comprises two components: (1) amplification, and (2) Nisan-Wigderson (NW) generator [25] applied to the amplified version $\mathsf{Amp}(f)$ of the function f. The purpose of the amplification component is to "error-correct" f so that even a circuit that computes $\mathsf{Amp}(f)$ with small advantage over random guessing can be used to construct a circuit that computes f almost everywhere. The NW generator applied to $\mathsf{Amp}(f)$ has the properties required of the function generator: locality and efficient reconstruction.

In our case, suppose that f agrees with some function $h \in \Lambda$ on a large fraction of inputs. Once we apply amplification to both f and h, we get $\mathsf{Amp}(f)$ and $\mathsf{Amp}(h)$ that are pushed further apart (as one would expect when using error-correcting codes). In order to keep the amplified functions close to each other, we will tone down the amplification procedure, which will adversely affect the approximation error of our learning algorithm, but the error can still be kept relatively small.

Next we need to ensure that the NW generator when applied to $\mathsf{Amp}(f)$ generates a family of functions such that most of them are sufficiently close to the family generated on $\mathsf{Amp}(h)$. In other words, we would like the generator to almost preserve the relative distance between the functions it is applied to. This can be achieved as follows. First, we observe that the definition of the NW generator guarantees that on a random seed z, the functions generated for $\mathsf{Amp}(f)$ and $\mathsf{Amp}(h)$ have the expected distance (over random z) equal to the actual distance between $\mathsf{Amp}(f)$ and $\mathsf{Amp}(h)$. Thus we have distance preservation in expectation. To make it concentrated around the expectation, we modify the NW construction by adding a pairwise-independent generator inside the NW construction. This ensures that the truth tables output by the modified NW generator are evaluations of $\mathsf{Amp}(f)$ (or $\mathsf{Amp}(h)$) on a sequence of pairwise independent inputs. The required concentration then follows by the Chebyshev bound. (A similar modification of the NW generator was done in [17], where an expander-walk generator was used for even better concentration; we use a simple pairwise generator as it can be easily implemented in AC^0[2], and it provides sufficient concentration for our purposes.)

1.3 Related work

The concept of *agnostic* learning was introduced by Kearns et al. [20], where it was also shown that piecewise linear functions are agnostically learnable. Agnostic learning is also known for certain geometric patterns [10], and restricted neural networks [21]. More results are known for the restricted versions of agnostic learning, for instance, when the distribution over examples is uniform. The class of AC^0 functions was shown to be (weakly) agnostically learnable under the uniform distribution by [20]. It was later shown by [19] that the well-known LMN learning algorithm of [23] achieves a constant-factor approximation of the optimal error (improved to the constant factor 2 in [18]), and that a modification of the algorithm (using L_1 regression) achieves the optimal error; the runtime of the algorithm is quasipolynomial. In fact, the result of [19] is generic in the following sense: any concept class of functions with certain "Fourier concentration" (as is the case, e.g., for AC^0 functions by the results of [23]) admits an agnostic learning algorithm under the uniform distribution, with an optimal error, whose runtime depends on the strength of the Fourier concentration for the concept class.

In distribution-independent setting, allowing membership queries does not give extra power to agnostic learning, yet membership queries can help when the distribution is uniform [6]. In particular, under the uniform distribution, Gopalan, Kalai and Klivans [12] and Feldman [7] give polynomial-time agnostic learning algorithms with membership queries for decision trees.

Agnostic learning of parities is closely related to the well-studied problem of learning noisy parities, which has a number of applications beyond learning theory, from decoding random linear codes to cryptography[2, 9, 1, 24, 26].

Under the uniform distribution, agnostic learning of parities (that is, learning parities with adversarial noise) reduces to learning parities with random noise [8]. Blum, Kalai and Wasserman [3] give an algorithm that properly learns length k parities with random noise under uniform distribution in time and sample size $\mathsf{poly}((1/(1-2\eta))^{2^a}, 2^b)$, where $\eta < 1/2$ is the noise probability, and $ab \geq k$. This is in contrast to the NP-hardness of properly learning noisy parities under arbitrary distributions, which follows from [13]. Later, Lyubashevsky [24] improved query complexity of the [3] algorithm to $n^{1+\epsilon}$, at the expense of bringing the running time up to $2^{O(n/\log\log n)}$, for $\eta < 1/2 - 2^{-(\log n)^{\delta}}$ for a constant δ. A corollary of the latter result is a subexponential algorithm for decoding $n \times n^{1+\epsilon}$ random binary linear codes, in the random noise setting.

Regev [29] considered an extension of learning parity with noise to mod p, which he called LWE (learning with error). He has shown that an efficient solution to LWE (for some range of parameters) implies an efficient quantum approximation of two variants of the shortest vector problem (GapSVP and the shortest independent vectors problem) and presented a public-key cryptosystem based on its hardness.

Remainder of the paper

We start with some basic definitions in Section 2. In Section 3, we prove our main result, Theorem 1, by instantiating the "agnostic learning from tolerant natural properties" framework to the case of $\mathsf{AC}^0[q]$ circuits, for any prime q. We present this framework in full generality in Section 4, where, in particular, we prove Theorem 2. In Section 5, we discuss the difficulty of removing membership queries from our agnostic learning algorithms for $\mathsf{AC}^0[2]$ (as it would have consequences for learning noisy parities). We conclude with some open questions in Section 6. The appendix contains some proofs omitted from the main body of the paper.

2 Preliminaries

For n-variate boolean functions f and g, we define the distance between them, denoted $\mathrm{DIST}(f, g)$, to be the number of inputs x where $f(x) \neq g(x)$. We denote by $\mathrm{dist}(f, g)$ the relative distance $\mathrm{DIST}(f, g)/2^n$. For a class $\mathcal{F}$ of n-variate boolean functions, and an n-variate boolean function f, we define the distance of f from the class $\mathcal{F}$, denoted $\mathrm{DIST}(f, \mathcal{F})$, as $\min_{h\in\mathcal{F}} \mathrm{DIST}(f, h)$. The relative distance of f from $\mathcal{F}$ is $\mathrm{dist}(f, \mathcal{F}) = \mathrm{DIST}(f, \mathcal{F})/2^n$.

▶ **Definition 3** (Distinguishers). Let $L \colon \mathbb{N} \to \mathbb{N}$ be a stretch function, let $0 < \epsilon < 1$ be an error bound, and let $\mathcal{G} = \{g_m \colon \{0,1\}^m \to \{0,1\}^{L(m)}\}$ be a sequence of functions. Define $\mathsf{DIS}(\mathcal{G}, \epsilon)$ to be the set of all Boolean circuits D on $L(m)$-bit inputs satisfying: $\Pr_{z\in\{0,1\}^m}[D(g_m(z))] - \Pr_{y\in\{0,1\}^{L(m)}}[D(y)] > \epsilon$. We say that $D \in \mathsf{DIS}(\mathcal{G}, \epsilon)$ is a *distinguisher for* $\mathcal{G}$ *with the distinguishing probability* ϵ.

2.1 Learning algorithms

The concept of *agnostic learning* was introduced by [20]. As in the PAC model of Valiant [32], we have a distribution over labeled examples $(x, f(x))$ for some function f, and we wish to learn f up to a small additive error over the given distribution. However, unlike in the PAC model, we don't assume that f belongs to some concept class $\mathcal{C}$, but rather that f is "close" to $\mathcal{C}$. More precisely, setting opt to be the disagreement probability between f and the best (closest) function $h \in \mathcal{C}$, the agnostic learning algorithm is supposed to output, with high probability $1 - \delta$, a hypothesis that disagrees with f with probability at most $\mathsf{opt} + \epsilon$, for given $\epsilon, \delta \in [0, 1]$. If the underlying distribution over examples is uniform, we say that the concept class $\mathcal{C}$ is agnostically learnable under the uniform distribution.

In the special case where we allow membership oracle, i.e., our learning algorithm has oracle access to the function f it is trying to learn, we call it a (membership) query agnostic learning algorithm. If, in addition, the hypothesis error is measure under the uniform distribution, we call it a query agnostic learning algorithm under the uniform distribution.

The learning algorithms considered in our paper are query algorithms under the uniform distribution. However, they don't achieve the ideal error $\mathsf{opt} + \epsilon$. Rather, we get the error of the form $c(n) \cdot \mathsf{opt}$, for some function c, which we call the *weakness* parameter of the agnostic learning algorithm; we also assume that opt is non-negligible and so we can drop the additive error ϵ to simplify the notation. For example, in the case of $\mathcal{C} = \mathsf{AC}^0[2]$, our learning algorithm has weakness $\mathsf{poly}(\log n)$.

2.2 Tolerant natural properties

We extend the definition of a natural property [28] to the case of a tolerant one, which intuitively says that not only all "easy" functions are rejected by the property, but also all functions "sufficiently close" to the "easy ones" are rejected. Such tolerant properties yield not just worst-case, but also average-case circuit lower bounds.

Let F_n be the collection of all Boolean functions on n variables. Λ and Γ denote complexity classes. A combinatorial property is a sequence of subsets of F_n for each n.

▶ **Definition 4** (Tolerant Natural Property). A combinatorial property $\{R_n\}_{n\geq 0}$ is Γ-natural with density δ and τ-tolerant u-usefulness, for some functions $\delta, \tau : \mathbb{N} \to [0,1]$ and $u : \mathbb{N} \to \mathbb{N}$, if it satisfies the following conditions:

Γ-Constructivity: Given the truth table of f_n, a Γ-algorithm decides if $f_n \in R_n$.

δ-Largeness: $|R_n| \geq \delta(n) \cdot |F_n|$.

τ-Tolerant u-Usefulness: For all $f_n \in F_n$ (for large n), if $\mathrm{dist}(f_n, \Lambda[u(n)]) \leq \tau(n)$, then $f_n \notin R_n$.

The standard natural property [28] is 0-tolerant in our language. For a number of complexity classes, including $\mathsf{AC}^0[q]$ for primes q, 0-tolerant natural properties were given in [28]. We prove that the natural property of [27] has in fact $(1/n^3)$-tolerant usefulness against d-depth $\mathsf{AC}^0[2]$ circuits of size $\exp(\Omega(n^{1/(2d)}))$; see Section A of the appendix for the proof of the following.

▶ **Lemma 5** (Tolerant natural property for $\mathsf{AC}^0[2]$). *There is a* P*-natural property* $\{R_n\}_{n\geq 0}$ *with largeness* $1/2$*, and* $(1/n^3)$*-tolerant* $\exp(\Omega(n^{1/(2d)}))$*-usefulness against* d*-depth* $\mathsf{AC}^0[2]$ *circuits.*

3 Agnostic learning from tolerant natural properties for $\mathsf{AC}^0[2]$

3.1 The CIKK framework

Recall the way non-agnostic learning algorithms follow from natural properties in the framework of [4]. Suppose we want to learn a function f in some circuit class Λ; for simplicity, assume f has polynomial-size circuits of type Λ.

As a thought experiment, imagine the following transformations applied to f. First, we *amplify* f, getting a new function $F = \textsc{Amp}(f)$, on polynomially larger inputs, with the property:

> If we are given a small circuit computing F on at least $1/2 + \epsilon$ fraction of inputs, then we can construct a circuit computing the original function f on at least $1 - 1/\mathsf{poly}(n)$ fraction of inputs, in randomized time $\mathsf{poly}(n, 1/\epsilon)$, using membership queries to f.

Then F is used as a "hard function" for the NW generator G. For each seed z of the NW generator, we view the output binary string $G(z)$ of length L as the truth table of an ℓ-variate boolean function, for $\ell = \log L$. The crucial observation in [4] is that the circuit complexity of this ℓ-variate boolean function is polynomial in the circuit size of the original function f, which is $\mathsf{poly}(n)$.

We need to express this circuit complexity $\mathsf{poly}(n)$ as the function of the input size ℓ. Note that if the stretch L is small, for example, if $L = \mathsf{poly}(n)$, then $\ell = O(\log n)$, and so the ℓ-variate function (whose truth table is) output by $G(z)$ has circuit complexity exponential in its input size ℓ. Thus, to reduce the circuit complexity of the function output by $G(z)$, we need to increase the stretch L of the NW generator. For example, by taking $L = \exp(\mathsf{poly}\log n)$, we can ensure that the circuit complexity of $G(z)$ (for each seed z) is only weakly exponential in the input size ℓ.

The point of using the NW generator to produce truth tables of relatively easy functions $G(z)$ is that we assumed the existence of an efficient natural property (with sufficient usefulness) which will accept many random truth tables, but will reject all truth tables of easy functions. In other words, this natural property provides an efficient (polynomial-time) algorithm that *distinguishes* the outputs of the NW generator G from truly random strings.

But then, the analysis of the NW generator construction implies that we get from this distinguisher a new algorithm that computes F (the function upon which the NW generator was based) on at least $1/2 + \Omega(1/L)$ fraction of inputs; where the reconstruction algorithm requires membership oracle for f. The latter implies (by the aforementioned properties of $F = \textsc{Amp}(f)$) that we can construct a circuit computing f on almost all inputs, in time $\mathsf{poly}(n, L)$ (again, using membership oracle for f). Thus we get a learning algorithm from the natural property, using the efficient reconstruction algorithm for the NW generator and the amplification procedure.

For example, using natural properties against $\mathsf{AC}^0[2]$ that are useful against circuits of weakly-exponential size [28], the above framework yields a learning algorithm, with membership queries, for functions computable by polynomial-size $\mathsf{AC}^0[2]$ circuits, running in quasipolynomial time.

3.2 Extension to the agnostic learning case

We wish to apply the same framework to the task of agnostic learning. Suppose we wish to learn a function f which is only somewhat close to a function h in some circuit class Λ (of polynomial-size circuits). Suppose that $\text{dist}(f, \Lambda) \leq \beta$, and that $h \in \Lambda$ is the closest function to f. Assume we are given a membership oracle for f.

To apply the [4] approach to learn f, we need to ensure the following:

For most seeds z, the function $G(z)$ (for the NW generator based on $F = \textsc{Amp}(f)$) is rejected by the appropriate natural property for our circuit class Λ.

If so, then we have a distinguisher for the NW generator based on F, and, as before, can efficiently construct a circuit for computing f almost everywhere.

As f is not in the class Λ, but rather just close to it, the best we can hope for is that the amplified function $F = \textsc{Amp}(f)$ is also somewhat close to Λ, and that the outputs of the NW generator $G(z)$ based on F are also somewhat close to the class Λ (of larger circuit size). If we can guarantee that (most of) the strings $G(z)$ are at the relative distance at most τ from $\Lambda[u]$, then our natural property with τ-tolerant u-usefulness will be a distinguisher for the NW generator, and we can reconstruct a circuit approximately computing f.

We need to balance the opposing constraints. On the one hand, to keep $F = \textsc{Amp}(f)$ close to Λ, we cannot amplify f too much, as the amplification, like an error-correcting encoding, pushes the originally close functions far apart. On the other hand, the stronger the amplification applied to f, the smaller the approximation error we get from a circuit for f constructed by the learning algorithm. As we are restricted by the tolerance parameter τ of our natural property, we are forced to keep the amplification relatively weak, which in turn implies a weak approximation error for the learned circuit for f.

Suppose that $f\colon \{0,1\}^n \to \{0,1\}$ is at the relative distance β from some n-variate function $h \in \Lambda[\mathsf{poly}]$. We will fine-tune the amplification procedure of [4] so that $F = \textsc{Amp}(f)$ and $H = \textsc{Amp}(h)$ are at the relative distance at most $\mu(n)$, for some $\mu : \mathbb{N} \to [0, 1]$ to be determined. Then we need to ensure that the outputs of the NW generator on F and on H, for most random seeds z, produce truth tables of length L that are at the relative distance at most $\tau(\ell)$ from each other, where $\ell = \log L$ is the input size of such a function output by $G(z)$.

To ensure that the NW generator based on close functions F and H produces strings that are close (for most seeds z), we modify the NW generator by adding a pairwise-independent generator as an extra component. (Similar modification to the NW generator, using an expander-walk generator, was done in [17], for a different purpose.) We will show that such a modified NW generator, when run on functions F and H that are at the relative distance $\mu(n)$ from each other, indeed outputs, for most seeds z, strings $G^F(z)$ and $G^H(z)$ of length L each, which are at the relative distance at most $2\mu(n)$ from each other. Expressing $2\mu(n)$ as a function of the input length $\ell = \log L$, we get an upper bound on the relative distance between $G^F(z)$ and $\Lambda[u]$ (as $G^H(z) \in \Lambda[u]$ by our assumption that $h \in \Lambda[\mathsf{poly}]$), for most seeds z. Here we choose the stretch L long enough so that the circuit complexity of the functions $G^H(z)$ is at most u, where u is usefulness of our natural property. For example, for $\mathsf{AC}^0[2]$, we have usefulness against weakly-exponential circuit size $\exp(n^{1/(2d)})$ for depth d circuits, and so we can make L to be quasi-polynomial, $\exp(\mathsf{poly} \log n)$.

3.3 Outline of the general method

In converting a tolerant natural property to an agnostic learning algorithm, we go through the following steps, mostly analogous to the steps in [4].

Initial assumptions. We start with access via membership queries to a Boolean function f. We are promised that there is a function $h \in C$ so that $\mathrm{dist}(f,h) \leq \beta$, for some parameter β. We do not have any access to h, but can refer to it in the analysis.

Amplification. The first step is to perform an *amplification construction*, $\mathsf{Amp}(f)$, to obtain a function F. Similarly, we can (conceptually) apply $\mathsf{Amp}(h)$ to obtain a function H. We need the following properties:

1. We can simulate membership queries to F via membership queries to f
2. $H \in C$
3. We can bound $\mathrm{dist}(F,H)$ away from $1/2$. The exact bound we will require will depend on the tolerance of the natural property.

Pseudo-random Function Generator. We next convert F to a *pseudo-random function generator*, $G_s^F(I)$ (and, conceptually, convert H into $G_s^H(I)$. For each seed s, G_s^F is a Boolean function on ℓ bits, producing a truth table of size $L = 2^\ell$. We call L the *stretch* of the generator. We need the following properties:

1. Given s, the truth table for G_s^F can be computed via membership queries to F (and hence, f).
2. For each s, $G_s^H(I)$ has small C circuit complexity (as a function of ℓ bit input I)
3. With good probability over s, $\mathrm{dist}(G_s^F, G_s^H)$ is small

Again, the exact quantitative requirements will depend on the quality of the tolerant natural property. The stronger the circuit lower bound the property is useful against, the smaller we can make the stretch and so the larger the relative circuit complexity of G_s^H in (2) can be. The more tolerant the property is, the larger the allowed distance in (3) can be. The greater the density, the smaller the probability over seeds of small distance between G_s^F and G_s^H in (3) can be.

Apply tolerant natural property to get a distinguisher. Now we use the tolerant natural property as a *distinguisher*, telling the difference between G_s^F and a random function of the same size. The second and third conditions above imply that, for many seeds s, G_s^F is close to a function with small C complexity. Thus, the property will not hold for many such functions (as long as close is within the tolerance, and small within the usefulness of the property). On the other hand, largeness implies that it will hold for many random functions. A gap between these two probabilities implies a distinguishing probability. The size of the distinguisher we obtain will depend on the stretch L and the constructivity of the property.

Convert distinguisher to a predictor. We use the contrapositive of the correctness proof of the PRFG construction to obtain a *predictor* that non-trivially predicts F. Note that non-trivially usually means with advantage at most $1/L$ over random guessing, so the smaller the stretch, the better the predictor will be.

Reverse the amplification. Finally, we apply the converse of the hardness amplification correctness proof to obtain a circuit that computes the original function f with good probability. Note that the agreement of the circuit for f will depend on the strength of the hardness amplifier we can use (which is largely determined by the tolerance) but also on the prediction advantage (largely determined by the stretch, itself determined by the usefulness of the property). Thus, the strongest results will only apply when the tolerance is exponentially close to 1/2 and the usefulness is exponential.

3.4 The case of $\mathsf{AC}^0[2]$

We first consider the case of amplification for $\mathsf{AC}^0[2]$. The case of $\mathsf{AC}^0[q]$ for primes $q > 2$ can be done in a similar way, where we work with $\mathbf{GF}(q)$-valued rather than Boolean functions; we sketch the argument in Section 3.5 below.

Given a boolean function $f\colon \{0,1\}^n \to \{0,1\}$, and a parameter $k = k(n) \in \mathbb{N}$, the amplification $\mathsf{Amp}_k(f)$ is defined as the Goldreich-Levin (Hadamard code) encoding of the k-wise direct product of f:

$$\textsc{Amp}_k(f) = F(x_1, \dots, x_k, b_1, \dots, b_k) = \sum_{i=1}^{k} b_i \cdot f(x_i),$$

where $x_1, \dots, x_k \in \{0,1\}^n$, $b_1, \dots, b_k \in \{0,1\}$, and the summation is modulo 2.

It is shown in [4] that the error parameter of the learning algorithm for f is a function of k and the stretch L of the generator.

▶ **Theorem 6** ([4]). *Suppose the NW generator based on the function $F = \textsc{Amp}_k(f)$, with output strings of length L, is broken with a constant distinguishing probability. Then, using the distinguisher and membership queries to f, one can construct a circuit computing f on at least $1 - \epsilon$ fraction of inputs, for $\epsilon \leq O((\ln L)/k)$. The construction algorithm is a randomized $\mathsf{poly}(n, k, L)$-time algorithm.*

Suppose there is a function $h \in \mathsf{AC}^0[2]$ such that $\mathrm{dist}(f, h) = \beta$. As observed in [4], the function $H = \mathsf{Amp}_k(h) \in \mathsf{AC}^0[2]$ for any $k = k(n) \leq \mathsf{poly}(n)$. It is also easy to argue that $\mathrm{dist}(F, H) = 1/2 - (1-\beta)^k/2$. For a given $\tau = \tau(\ell)$, we want to choose k so that $\mathrm{dist}(F, H) \leq \tau/4$. That is, we want $(1-\beta)^k \geq 1 - \tau/2$. Using the inequalities $1 + x \leq e^x$ (true for all x), and $1 - x \geq e^{-2x}$ (true for all $0 \leq x \leq 0.7$), we are allowed to take $k = \tau/(4\beta)$.

Then the NW generator based on F outputs a truth table of an ℓ-variate function that has the expected (over random seeds z to the generator) relative distance at most $\tau/4$ from the class of $\mathsf{AC}^0[2]$ circuits of size u, for weakly-exponential circuit size u (for which we have a tolerant natural property given by Theorem 5). By Markov's inequality, we get that the actual distance is at most τ for at least 3/4 fraction of the random seeds z to the generator.[2] Thus, for $\mathsf{AC}^0[2]$, we can make the stretch L of our generator to be quasipolynomial, $L = \exp(\mathsf{poly}(\log n))$. Then $\ell = \log L = \mathsf{poly}(\log n)$.

[2] Here, and for the case of $\mathsf{AC}^0[q]$ for primes $q > 2$ later, we can use a simple averaging argument and keep the NW generator as is, because we have natural properties for these classes with very poor tolerance parameters. However, for the general case, when we may have better tolerance parameters, we achieve better concentration by combining the NW generator with a pairwise-independent generator.

As we have $(1/\ell^3)$-tolerant natural property for $\mathsf{AC}^0[2]$ circuits of size u computing ℓ-input boolean functions (Theorem 5), we set $\tau = (1/\ell^3)$, and get that $k = (4\beta\ell^3)^{-1}$. As the τ-tolerant natural property breaks the NW generator based on F, we get by Theorem 6 that f can be learned up to the error $O((\log L)/k) \leq O(\beta \cdot \ell^4) \leq \mathsf{poly}(\log n) \cdot \beta$.

Thus we have proved the following.

▶ **Theorem 7** (Agnostic learning of $\mathsf{AC}^0[2]$)**.** *There is a randomized quasipolynomial-time algorithm for agnostically learning, with membership queries, a function* $f\colon \{0,1\}^n \to \{0,1\}$ *with* $\mathrm{dist}(f, \mathsf{AC}^0[2]) \leq \beta$ *(for a non-negligible* $\beta > 0$*), producing a circuit that computes* f *on all but at most* $\mathsf{poly}(\log n) \cdot \beta$ *fraction of inputs.*

3.5 The case of $\mathsf{AC}^0[q]$ for prime $q > 2$

Next, we consider the case of agnostic learning for $\mathsf{AC}^0[q]$ for prime $q > 2$. While this follows the general outline of the $\mathsf{AC}^0[2]$ case, there are some differences. In particular, to keep the function generators close to functions in $\mathsf{AC}^0[q]$, we need to consider them as producing functions which take Boolean $\{1, -1\}$ inputs to outputs in the range $\{0, \ldots, q-1\}$ of integers modulo q. We need to adjust the natural property from [28] to handle such functions. This turns out to actually simplify the argument from [28] and to eliminate one step (the von Neumann construction) from the PRFG construction in [4].

Our learning algorithm follows the general outline.

Preconditions. We assume membership query access to a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$, and a value β and integer d so that we are promised that there is an h in $\mathsf{AC}^0[q]$ computable by a depth d circuit and $\mathrm{dist}(f,h) \leq \beta$.

Amplification. Given a parameter $k = k(n)$, the mod q amplification $\mathsf{Amp}_{k,q}(f)$ is defined as the mod q Goldreich-Levin (Hadamard code) encoding of the k-wise direct product of f:

$$\textsc{Amp}_{k,q}(f) = F(x_1, \ldots, x_k, b_1, \ldots, b_k) = \sum_{i=1}^{k} b_i \cdot f(x_i),$$

where $x_1, \ldots, x_k \in \{1,-1\}^n$, $b_1, \ldots, b_k \in \{0, \ldots, q-1\}$, and the summation is modulo q. Note that this function takes on values in $\{0, \ldots, q-1\}$. We will extend the class $AC^0[q]$ to include such functions in any of several obvious ways, e.g., by having q output gates with the one true one selecting the output. We can code inputs taking on such values similarly.

In our construction, we will set $k = 1/(10 \cdot \beta)$. Let the functions H and F be defined by $H = \mathsf{Amp}_{k,q}(h) \in \mathsf{AC}^0[q]$ and $F = \mathsf{Amp}_{k,q}(f)$. Then $\mathrm{dist}(F,H) = (1-(1-\beta)^k)(1-1/q) \leq k\beta = .01$, since if f and h agree on all k inputs, the functions F and H will agree, and otherwise, they agree with conditional probability $1/q$. Also, H is computable by a depth $d+2$ $\mathsf{AC}^0[q]$ circuit of polynomial size, and a query to F can be simulated with k queries to f.

Pseudo-random function generator. As in [4], we use a version of the NW generator with a design based on polynomials over $\mathbf{GF}(q)$. We are applying this to the function F with non-Boolean outputs from $\mathbf{GF}(q)$, so the resulting truth table will be, for each seed s, a vector of values mod q. We will set the stretch L to be quasi-polynomial in n, $L = \exp(C \cdot \log^{qd+c} n)$ for some constants C and c, where we need the q in the exponent of the $\mathsf{polylog}$ because of the overhead of GL reconstruction for circuits with outputs in $\mathbf{GF}(2)$. Note that we can construct such a truth table with L queries to F. A subtlety is that, while we look at the sets in the design as determined by polynomials over $\mathbf{GF}(q)$, we only consider those L polynomials of degree $\ell - 1$, where $\ell = \log_2 L$, with co-efficients in $\{1, -1\}$.

Call this pseudo-random function generator using F and H respectively, and seed s, G_s^F and G_s^H. As noted in [4], for each seed s, G_s^H can be computed by $\mathsf{poly}(n)$ sized circuits of depth $d + O(1)$.

Since for a random seed s and random position I, the value F is queried at is uniform, $\mathbb{E}_s\left[\mathrm{dist}(G_s^F, G_s^H)\right] = \mathrm{dist}(F, H) \leq .01$. By Markov, we get $\Pr\left[\mathrm{dist}(G_s^F, G_s^H) \geq .1\right] \leq .1$.

Apply natural property. At this point, we apply a tolerant natural property. We need a variant of natural property that applies to functions with Boolean inputs and outputs in $\mathbf{GF}(q)$. It turns out that the Razborov-Rudich [28] natural property for $\mathsf{AC}^0[q]$ is actually simpler in this case. We prove the following in Section B of the appendix.

▶ **Lemma 8** (Tolerant natural property for $\mathsf{AC}^0[q]$)**.** *There is a* P*-natural property* $\{R_n\}_{n\geq 0}$ *with largeness* $1/2$*, and* $(.15)$*-tolerant* $\exp(\Omega(n^{1/(2d)}))$*-usefulness against* d*-depth* $\mathsf{AC}^0[q]$ *circuits computing functions* $f\colon \{1,-1\}^n \to \mathbf{GF}(q)$*.*

We get that at most $1/10$ of the functions G_s^F will be of high complexity, whereas a random function will be of high complexity with probability $1/2$. So testing whether a function has high complexity gives us a $\mathsf{poly}(L)$ size distinguisher with constant advantage for distinguishing G_s^F from a random function.

Converting to a predictor. Using the standard hybrid argument and proof of correctness for the NW generator, we can convert this distinguisher into a predictor circuit of size $\mathsf{poly}(L)$ and advantage $\Omega(1/L)$ of predicting $F(z)$ over random guessing. (To compute this predictor, we need to query F and hence f at $\mathsf{poly}(L)$ positions; see [4]. This is the main step that requires membership queries.)

Converse of amplification. Applying the converse of the generalized GL construction and the direct product theorems, we can convert this predictor circuit into one that computes f on $1-\gamma$ inputs, where $(1-\gamma)^{\Omega k} = \Omega(1/L)$. Thus, $e^{-C_1\gamma k} = C_2/L$, or $\gamma = O(\log L/k) = O(\beta \cdot \log L) = O(\beta \cdot \log^{qd+c} n)$. So we get an agnostic learner that works in time and queries quasi-polynomial in n, and with error at most $O(\log^{qd+c} n)\cdot\beta$. (Note that this assumes β is non-negligible; otherwise, the time and circuit size depend on $1/\beta$ as well).

Combining all these pieces, we have the following.

▶ **Theorem 9** (Agnostic learning of $\mathsf{AC}^0[q]$)**.** *Let* $q > 2$ *be any prime. There is a randomized quasipolynomial-time algorithm for agnostically learning, with membership queries, a function* $f\colon \{0,1\}^n \to \{0,1\}$ *with* $\mathrm{dist}(f, \mathsf{AC}^0[q]) \leq \beta$ *(for a non-negligible* $\beta > 0$*), producing a circuit that computes* f *on all but at most* $\mathsf{poly}(\log n)\cdot\beta$ *fraction of inputs.*

4 Agnostic learning from tolerant natural properties

Next, we consider the case of agnostic learning for any Λ closed under $\mathsf{AC}^0[2]$-reductions for *any* natural property against Λ with super-constant tolerance and usefulness. This follows the general outline of the $\mathsf{AC}^0[2]$ case, but we need to use a variant of the NW pseudorandom generator to take advantage of (potentially) better tolerance. We will use Chebyshev instead of Markov to bound the probability, over random seeds z, that the functions mapped to by the generator have small distance. Our generic learning algorithm also follows the outline.

Preconditions. Let Λ be some complexity class closed under $\mathsf{AC}^0[2]$-reductions. Let $\mathcal{R}$ be a BPP-constructive, τ-tolerant, u-useful natural property against Λ, for super-constant

τ with largeness $\delta > (1/2)$. Write $\tau = (1/2) - \tau'$, because it will sometimes be easier to work with τ as an "advantage." We assume membership query access to a Boolean function $f \colon \{0,1\}^n \to \{0,1\}$, and a value β so that we are promised that there is an h in Λ with $\mathrm{dist}(f,h) \le \beta$.

Amplification. We use $\textsc{Amp}_k$ identically to the specific case of $\mathsf{AC}^0[2]$, except that we set k later based on abstract τ and u. Let $F = \textsc{Amp}_k(f)$, $H = \textsc{Amp}_k(h)$, as before $\mathrm{dist}(F,H) = (1/2) - (1/2)(1-\beta)^k$, which we call μ.

Pseudo-random function generator. As in [4], we use a version of the NW generator with a design based on polynomials over $\mathbf{GF}(2)$. Recall that the NW design for parameters $n, m, L \in \mathbb{N}$ is a family of sets $S_1, \ldots, S_L \subseteq [m]$, of size $|S_i| = n$, for all $1 \le i \le L$, and small overlap $|S_i \cap S_j| \le \log L = \ell$ for all $1 \le i \ne j \le L$. It was shown in [4] that such designs can be efficiently locally computed by $\mathsf{AC}^0[q]$ circuits, for any prime q.

▶ **Lemma 10** (NW design in $\mathsf{AC}^0[q]$ [25, 4]). *Let q be any prime. There is a constant $d_0 \in \mathbb{N}$ such that, for any n and $L < 2^n$, there exists an NW design $S_1, \ldots, S_L$ with parameters as defined above, so that the function $MX_{NW} \colon \{0,1\}^\ell \times \{0,1\}^m \to \{0,1\}^n$, defined by $MX_{NW}(i,z) = z|_{S_i}$, where $z|_{S_i}$ denotes the substring of z indexed by S_i, is computable by an $\mathsf{AC}^0[q]$ circuit of depth d_0 and size $\mathsf{poly}(\ell, n)$.*

The NW generator [25] based on a boolean function $F \colon \{0,1\}^n \to \{0,1\}$ is $G^F \colon \{0,1\}^m \to \{0,1\}^L$ defined as $G^F(z) = F(z|_{S_1}) \circ \cdots \circ F(z|_{S_L})$, where $S_1, \ldots, S_L$ is the NW design as above. Lemma 10 implies that if $F \in \mathsf{AC}^0[2]$, then, for each seed z, the output $G^F(z)$ is the truth table of an ($\ell = \log L$)-variate Boolean function of $\mathsf{AC}^0[2]$ circuit complexity at most $\mathsf{poly}(\ell, n)$.

Let $H \colon \{0,1\}^n \to \{0,1\}$ be another boolean function such that $\mathrm{dist}(F,H) \le \mu$, for some $\mu \in [0,1]$. By the definition of the NW generator, we have that the expected hamming distance between the L-bit strings $G^F(z)$ and $G^H(z)$, over random seeds z, is $\mathrm{dist}(F,H) \cdot L \le \mu \cdot L$. For our agnostic learning framework, it is important (as explained in the previous section) that the NW generator have the *concentration property*: for most seeds z, the hamming distance between $G^F(z)$ and $G^H(z)$ is close to the expected distance $\mu \cdot L$.

We achieve this concentration property by adding a pairwise-independent string generator as a component of the NW generator. Let $PI \colon \{0,1\}^\ell \times \{0,1\}^{m'} \to \{0,1\}^n$ be a *pairwise independent generator* such that

1. for each $i \in [L]$, the distribution $PI(i,z)$ over uniformly random $z \in \{0,1\}^{m'}$ is uniform over $\{0,1\}^n$, and
2. for all $i \ne j \in [L]$, the distribution of $PI(i,z)$ and $PI(j,z)$, over uniformly random seeds $z \in \{0,1\}^{m'}$, is uniform over $\{0,1\}^n \times \{0,1\}^n$.

Such generators exist for $m' \le n(\ell+1)$; for example, pick a random 0/1 matrix A of dimension $n \times \ell$ and a random 0/1 vector v of dimension n. Let $z = (A, v)$. Define $P(i, (A,v)) = A \cdot i + v$, where $A \cdot i$ denotes the matrix-vector multiplication, and all operations are over $\mathbf{GF}(2)$. It is easy to see that this generator $PI(i,z)$ is computable by an $\mathsf{AC}^0[2]$ circuit of polynomial size.

Define the modified NW generator $G'^F \colon \{0,1\}^m \times \{0,1\}^{m'} \to \{0,1\}^L$, based on the n-variate boolean function F, as follows:

$$G'^F(z_1, z_2) = F(z_1|_{S_1} \oplus PI(1, z_2)) \circ \cdots \circ F(z_1|_{S_L} \oplus PI(L, z_2)),$$

where S_i's form the NW design, and PI is the pairwise-independent generator as above, and $\oplus$ denotes the bit-wise XOR of the corresponding n-bit strings.

Observe that since the generator PI is efficiently locally computable in $\mathsf{AC}^0[2]$, we still get (by Lemma 10) that the ℓ-bit function output by G'^F, for $F \in \mathsf{AC}^0[2]$, has $\mathsf{AC}^0[2]$ circuit complexity at most $\mathsf{poly}(\ell, n)$. Next, the generator G' allows the same kind of reconstruction as the original NW generator: given a distinguisher for G' with a constant distinguishing probability, one can efficiently construct (using membership queries to F) a small circuit computing F on at least $1/2 + \Omega(1/L)$ fraction of inputs. Finally, the generator $G'^F(z_1, z_2)$, for uniformly random seeds z_1 and z_2, outputs L values of F on *pairwise-independent* uniformly random n-bit inputs.

From pairwise independence we get that the hamming distance between $G'^F(z_1, z_2)$ and $G'^H(z_1, z_2)$, over random z_1 and z_2, is concentrated around the expectation, by the Chebyshev bound. More precisely, for F and H with $\text{dist}(F, H) \le \mu$, we have by Chebyshev that

$$\mathbf{Pr}_z \left[\left| \text{DIST}(G'^F(z), G'^H(z)) - \mu \cdot L \right| > \zeta \cdot L \right] < \frac{1}{\zeta^2 \cdot L},$$

which we will require to be less than $1/4$. We parameterize the bound with $\zeta = (1/4)(1-\beta)^k$. For the selected ζ, and the stretch L we are forced to pick later, this is immediate.

Apply natural property. At this point, we apply a tolerant natural property to produce a distinguisher circuit for the generator above. This induces the following system of constraints, which relate the usefulness, tolerance, and density of the property to the stretch and concentration of the generator. Let $\Lambda\text{-SIZE}(G'^H(z)) = s_H$. We require that $s_H \le u(\ell)$, to respect the size lower bound. We re-arrange the Chebyshev bound above and see that we should require $\mu + \zeta < \tau(\ell)$, respect tolerance, and ensure a good distinguishing gap from the property. We satisfy the first requirement by setting $\ell \ge u^{-1}(s_H)$. The second one is equivalent to $(1/4)(1-\beta)^k > \tau'(\ell)$. In this case, the tolerant property can only accept $G^F(z)$ with probability $(1/4)$ but accepts a random function with probability at least $(1/2)$, giving us a $(1/4)$ distinguishing gap. We can satisfy both constraints by setting $k = \Theta(\log(\tau'(\ell))/\beta)$.

Converting to a predictor. Using a small modification of the standard hybrid argument and proof of correctness for the NW generator, we can convert this distinguisher into a predictor circuit of size $\mathsf{poly}(L)$ and advantage $\Omega(1/L)$ of predicting $F(z)$ over random guessing. The modified predictor just embeds a construction of PI and shifts/unshifts inputs to the distinguisher circuit as necessary. (To compute this predictor, we need to query F and hence f at $\mathsf{poly}(L)$ positions; see [4]. This is the main step that requires membership queries.) From this step we know that our runtime is at most $\mathsf{poly}(L)$, and the circuit output at this stage is already size $\mathsf{poly}(L)$.

Converse of amplification. Identical to the case of $\mathsf{AC}^0[q]$, but with the additional constraints mentioned above. Note that the runtime of these algorithms is randomized time in the size of the input circuit, so runtime, number of queries, and output circuit size of this stage will also be dominated by L. Use of this algorithm imposes the following constraint from the direct product reconstruction stage: $\mathsf{poly}(1/L) > e^{-k\epsilon/c}$. So $\epsilon > \Theta(\log(L)/k)$. Substituting in our value for k, this gives us $\epsilon = \Theta(\ell\beta/\log(\tau'(\ell)))$ for $\ell = u^{-1}(s_H)$.

Summarizing, we get a generic reduction from tolerant natural properties to agnostic learning.

▶ **Theorem 11** (Tolerant natural properties imply agnostic learning algorithms). *Let $\mathcal{R}$ be a natural property against Λ closed under $\mathsf{AC}^0[2]$ reductions with $(1/2-\tau')$-tolerant u-usefulness and* *largeness $\delta \geq 1/2$. Then there is a randomized algorithm such that, for any n-ary boolean functions f and h with* $\mathrm{dist}(f,h) < \beta$ *and $s_h = \Lambda$-SIZE(h), the algorithm, given oracle access to f, produces a circuit ϵ-approximating f, for any* $\epsilon > \beta \cdot u^{-1}(\mathsf{poly}(s_h))/\log(\tau'(u^{-1}(\mathsf{poly}(s_h))))$, *in time* $\mathsf{poly}(\max\{2^{u^{-1}(\mathsf{poly}(s_h))},\ 1/\epsilon\})$.

In particular, this means if we have a "perfect" natural property, with exponential usefulness u and inverse exponential tolerance τ', we have a polynomial-time learning algorithm with error bound $\Theta(\beta)$. Thus Theorem 2 is a special case of Theorem 11.

5 Hardness of removing membership queries

Is it possible to eliminate membership queries from our algorithm, learning just from random examples? We note that removing membership queries would give us quasipolynomial-time algorithms for two notoriously difficult problems: learning parities with noise (LPN) for the case of $\mathsf{AC}^0[2]$ and a variant of learning with errors (LWE) for $\mathsf{AC}^0[q]$.

Though learning parities with noise under uniform distribution can be done in polynomial time with membership queries (by the Goldreich-Levin algorithm [11]), without membership queries this problem is believed to be hard. Learning parities with noise efficiently under uniform distribution would give learning algorithms for DNFs and k-juntas (and in general, for any problem reducible to finding a heavy Fourier coefficient of a function) [8].

In the worst case, LPN is known to be NP-hard (and MAX-SNP-hard). The average-case hardness of LPN has been considered as early as 1993, when Blum, Furst, Kearns and Lipton have given a simple construction of a pseudorandom bit generator based on the assumption that learning parities with constant noise rate is hard [2]. In practical cryptography, average-case hardness of LPN is the basis for Hopper and Blum authentication protocol [14]. There, the noise rate is usually set to a constant $\eta \in (0, 1/2)$, in particular $\eta = 1/8$ has been used in applications [22]. Though for $\mathsf{AC}^0[2]$ our algorithm works for noise up to $1/\mathsf{polylog}(n)$, we can tolerate constant noise for $\mathsf{AC}^0[q]$.

Hardness of LWE problem follows from worst-case hardness of variants of the lattice shortest vector problem [29]. Whereas LPN has been used to build "minicrypt" cryptographic primitives, LWE has been used for public-key cryptosystems [1, 29].

6 Open questions

While there are correlation bounds for $\mathsf{AC}^0[q]$ circuits that say that some explicit functions cannot be computed by "small" circuits on significantly more that $1/2 + 1/\sqrt{n}$ fraction of inputs, we do not know how to get natural properties with tolerance close to $1/2$. Getting natural properties with better tolerance parameters would immediately imply improved parameters for our agnostic learning algorithms for the corresponding circuit classes. (Of course, getting stronger correlation bounds for $\mathsf{AC}^0[q]$, whether obtained by natural proofs or not, is in itself a very important problem in circuit complexity.)

Can one get a query agnostic learning algorithm for $\mathsf{AC}^0[q]$ with the optimal error $\mathsf{opt} + \epsilon$? It seems that, even with ideal tolerance and usefulness, our approach of getting learning algorithms from natural properties will at best achieve the error $O(\mathsf{opt}) + \epsilon$. So one needs a new approach, perhaps inspired by the learning algorithm in this paper.

In fact, probably the main open problem is to get a more "natural" (understandable) learning algorithm for $\mathsf{AC}^0[q]$ than our construction, which combines the NW-style generator analysis with circuit lower bound proofs. As a possible first step, it would be interesting to get an alternative agnostic learning algorithm for low-degree polynomials over $\mathbf{GF}(2)$.

References

1 Michael Alekhnovich. More on average case vs approximation complexity. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 298–307. IEEE, 2003.

2 Avrim Blum, Merrick Furst, Michael Kearns, and Richard J Lipton. Cryptographic primitives based on hard learning problems. In *Annual International Cryptology Conference*, pages 278–291. Springer, 1993.

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

4 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In Ran Raz, editor, *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, volume 50, pages 10:1–10:24. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016.

5 Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. Mining circuit lower bound proofs for meta-algorithms. *Computational Complexity*, 24(2):333–392, 2015.

6 Vitaly Feldman. On the power of membership queries in agnostic learning. *Journal of Machine Learning Research*, 10:163–182, 2009.

7 Vitaly Feldman. Distribution-specific agnostic boosting. In *Innovations in Computer Science – ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 241–250, 2010.

8 Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. New results for learning noisy parities and halfspaces. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 563–574. IEEE, 2006.

9 Jean-Bernard Fischer and Jacques Stern. An efficient pseudo-random generator provably as secure as syndrome decoding. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 245–255. Springer, 1996.

10 Sally A. Goldman, Michael J. Kearns, and Robert E. Schapire. On the sample complexity of weakly learning. *Inf. Comput.*, 117(2):276–287, 1995.

11 Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In David S. Johnson, editor, *Proceedings of the 21st Annual ACM Symposium on Theory of Computing, May 14-17, 1989, Seattle, Washigton, USA*, pages 25–32. ACM, 1989.

12 Parikshit Gopalan, Adam Tauman Kalai, and Adam R. Klivans. Agnostically learning decision trees. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 527–536. ACM, 2008.

13 Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

14 Nicholas J. Hopper and Manuel Blum. Secure human identification protocols. In *Advances in Cryptology – ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, pages 52–66, 2001.

15 Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC^0. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 961–972. SIAM, 2012.

16 Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 111–119. IEEE Computer Society, 2012.

17 Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229. ACM, 1997.

18 Jeffrey C. Jackson. Uniform-distribution learnability of noisy linear threshold functions with restricted focus of attention. In *Proceedings of the 19th Annual Conference on Learning Theory*, COLT'06, pages 304–318, Berlin, Heidelberg, 2006. Springer-Verlag.

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

20 Michael J. Kearns, Robert E. Schapire, and Linda Sellie. Toward efficient agnostic learning. *Machine Learning*, 17(2-3):115–141, 1994.

21 Wee Sun Lee, Peter L. Bartlett, and Robert C. Williamson. Efficient agnostic learning of neural networks with bounded fan-in. *IEEE Trans. Information Theory*, 42(6):2118–2132, 1996.

22 Éric Levieil and Pierre-Alain Fouque. An improved lpn algorithm. In *International Conference on Security and Cryptography for Networks*, pages 348–359. Springer, 2006.

23 Nathan Linial, Yishay Mansour, and Noam Nisan. Constant depth circuits, Fourier transform, and learnability. *J. ACM*, 40(3):607–620, 1993.

24 Vadim Lyubashevsky. The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 378–389. Springer, 2005.

25 Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994.

26 Krzysztof Pietrzak. Cryptography from learning parity with noise. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 99–114. Springer, 2012.

27 A. A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematical notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

28 Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

29 Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM (JACM)*, 56(6):34, 2009.

30 Rahul Santhanam. Fighting perebor: New and improved algorithms for formula and QBF satisfiability. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 183–192. IEEE Computer Society, 2010.

31 Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*, pages 77–82, 1987.

32 Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984.

33 Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. In Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 231–240. ACM, 2010.

34 Ryan Williams. Non-uniform ACC circuit lower bounds. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, June 8-10, 2011*, pages 115–125. IEEE Computer Society, 2011.

A Tolerant natural property for $\mathsf{AC}^0[2]$

Razborov [27] showed the following natural property for $\mathsf{AC}^0[2]$:

> Given an n-variate boolean function f, construct certain matrices $A_1, \dots, A_b$, for $b = n/2 - \sqrt{n}$, of dimensions at most $2^n \times 2^n$, and check if at least one of the matrices has rank at least $2^n/(140 \cdot n^2)$ over $\mathbf{GF}(2)$.

More precisely, for $a = n/2 - \sqrt{n}$ and all $i \leq a$, define A_i to be the matrix whose rows are labeled by size a subsets of $[n]$, and whose columns are labeled by size i subsets of $[n]$. For $K \subseteq [n]$, let $Z(K) = \{x \in \{0,1\}^n \mid x|_K = \vec{0}\}$. For a row $I \subseteq [n]$ and a column $J \subseteq [n]$, define $(A_i)_{I,J} = \oplus_{x \in Z(I \cup J)} f(x)$.

It is possible to show that at least 1/2 of all n-variate boolean functions satisfy this property; so we have largeness (see [4]). The usefulness of this property is due to the following two lemmas. Below we denote by $\mathcal{P}(D)$ the linear space of all n-variate degree D multilinear polynomials over $\mathbf{GF}(2)$.

▶ **Lemma 12** ([27]). *For an n-variate boolean function f and the corresponding matrices $A_1, \dots, A_b$, for $b = n/2 - \sqrt{n}$, we have for all $1 \leq i \leq b$ that*

$$\mathrm{DIST}(f, \mathcal{P}(\sqrt{n})) \geq \mathrm{rank}(A_i).$$

▶ **Lemma 13** ([27]). *For an n-variate boolean function f, if f is computable by a d-depth $\mathsf{AC}^0[2]$ circuit of size s, then*

$$\mathrm{dist}(f, \mathcal{P}((O(\log(s/\epsilon))^d)) \leq \epsilon.$$

So for $\epsilon = 1/n^3$ and size $s < \exp(\Omega(n^{1/(2d)}))/n^3$, we get by Lemma 13 that any f computable by a d-depth $\mathsf{AC}^0[2]$ circuit of size s is such that $\mathrm{DIST}(f, \mathcal{P}(\sqrt{n})) \leq 2^n/n^3$. Hence, by Lemma 12, all the corresponding matrices A_i for f have rank at most $2^n/n^3 \leq 2^n/(140 \cdot n^2)$ (for all sufficiently large n), and so f is rejected by the natural property.

Now suppose that h is an n-variate boolean function that is close to f, i.e., for some $0 \leq \beta \leq 1$,

$$\mathrm{dist}(h, f) \leq \beta,$$

where f is as above. Then we get by the triangle inequality that

$$\mathrm{DIST}(h, \mathcal{P}(\sqrt{n})) \leq (\beta + n^{-3}) \cdot 2^n,$$

which, in particular, means that for any $\beta \leq 1/n^3$, such a function h will also be rejected by the natural property above.

Thus we have proved the following.

▶ **Lemma 14** (Tolerant natural property for $\mathsf{AC}^0[2]$). *There is a P-natural property $\{R_n\}_{n \geq 0}$ with largeness 1/2, and $(1/n^3)$-tolerant $\exp(\Omega(n^{1/(2d)}))$-usefulness against d-depth $\mathsf{AC}^0[2]$ circuits.*

B Tolerant natural property for $\mathsf{AC}^0[q]$ for prime $q > 2$

Here we prove the following.

▶ **Lemma 15** (Tolerant natural property for $\mathsf{AC}^0[q]$)**.** *There is a* P*-natural property* $\{R_n\}_{n\geq 0}$ *with largeness* $1/2$*, and* $(.15)$*-tolerant* $\exp(\Omega(n^{1/(2d)}))$*-usefulness against* d*-depth* $\mathsf{AC}^0[q]$ *circuits computing functions* $f\colon \{1,-1\}^n \to \mathbf{GF}(q)$*.*

Proof. Let $\mathcal{M}$ be the vector space of all n-variate multilinear polynomials over $GF(q)$, and let $\mathcal{L}$ be the subspace of those polynomials of degree at most $n/2$. Given such a multilinear polynomial f (and any truth table indexed by $\{1,-1\}^n$ over $GF(q)$ defines such a polynomial), we say that f is high complexity if the dimension $\dim(\mathcal{L} + f \cdot \mathcal{L}) \geq 3/4 \cdot N$, where $N = 2^n$.

Note that, for any function f of degree d, $\mathcal{L} + f \cdot \mathcal{L}$ is contained within the space of multilinear polynomials of degree $l/2 + d$, which has dimension at most $N(1/2 + O(d/\sqrt{n}))$. Changing any D values can increase this dimension by at most D (since adding the dimension D vector space of all functions on these D points to the subspace for the original function includes the subspace functions for the changed function). So in particular, any high complexity function must have distance at least $1/5$ from any function of degree $c\sqrt{n}$ for some $c > 0$. Since by work by Razborov [27] and Smolensky [31], any function in $\mathsf{AC}^0[q]$ of depth d and size s is within ϵ distance of a multilinear polynomial over $\mathbf{GF}(q)$ of degree $O(\log(s/\epsilon)^d)$, any high complexity function must be distance .15 from any function computed by size $\exp(\Omega(n^{1/(2d+C)}))$ depth $d + C$ circuits with mod q gates.

At least half of such functions have high complexity. From [31], for p the product of all l inputs (i.e., the parity of the number of -1 inputs), $\mathcal{L} + p \cdot \mathcal{L} = \mathcal{M}$. Then for f any function, either f has high complexity or $p - f$ does. Because if both have low complexity, then

$$\dim(\mathcal{L} + f \cdot \mathcal{L}) = \dim \mathcal{L} + \dim((f \cdot \mathcal{L})/\mathcal{L}) < \frac{3}{4} \cdot N,$$

so $\dim((f \cdot \mathcal{L})/\mathcal{L}) < (1/4) \cdot N$, and similarly for $p - f$. Then

$$\begin{aligned}
\dim \mathcal{M} &= \dim(\mathcal{L} + p \cdot \mathcal{L}) \\
&\leq \dim(\mathcal{L} + f \cdot \mathcal{L} + (p - f) \cdot \mathcal{L}) \\
&\leq \dim \mathcal{L} + \dim((f \cdot \mathcal{L})/\mathcal{L}) + \dim(((p - f) \cdot \mathcal{L})/\mathcal{L}) \\
&< N/2 + N/4 + N/4 \\
&= N,
\end{aligned}$$

a contradiction. Since all functions can be paired up into $f, p - f$ pairs, at least half the functions have high complexity. Clearly, we can test whether a function has high complexity in $\mathsf{poly}(N)$ time. ◀