PAC Learning Depth-3 AC^0 Circuits of Bounded Top Fanin

Ning Ding DINGNING@SJTU.EDU.CN
Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, 200240, China
State Key Laboratory of Cryptology
P.O.Box 5159, Beijing, 100878, China

Yanli Ren RENYANLI@SHU.EDU.CN
School of Communication and Information Engineering
Shanghai University
Shanghai, China

Dawu Gu DWGU@SJTU.EDU.CN
Department of Computer Science and Engineering
Shanghai Jiao Tong University
Shanghai, 200240, China

Editors: Steve Hanneke and Lev Reyzin

Abstract

An important and long-standing question in computational learning theory is how to learn AC^0 circuits with respect to any distribution (i.e. PAC learning). All previous results either require that the underlying distribution is uniform Linial et al. (1993) (or simple variants of the uniform distribution) or restrict the depths of circuits being learned to 1 Valiant (1984) and 2 Klivans and Servedio (2004). As for the circuits of depth 3 or more, it is currently unknown how to PAC learn them.

In this paper we present an algorithm to PAC learn depth-3 AC^0 circuits of bounded top fanin over $(x_1, \cdots, x_n, \overline{x}_1, \cdots, \overline{x}_n)$. Our result is that every depth-3 AC^0 circuit of top fanin K can be computed by a polynomial threshold function (PTF) of degree $\widetilde{O}(K \cdot n^{\frac{1}{2}})$, which means that it can be PAC learned in time $2^{\widetilde{O}(K \cdot n^{\frac{1}{2}})}$. In particular, when $K = O(n^{\epsilon_0})$ for any $\epsilon_0 < \frac{1}{2}$, the time for learning is sub-exponential. We note that instead of employing some known tools we use some specific approximation in expressing such circuits in PTFs which can thus save a factor of polylog(n) in degrees of the PTFs.

Keywords: PAC Learning, AC^0 Circuits, Polynomial Threshold Functions, Rational Functions

1. Introduction

The seminal result of Linial et al. (1993) shows the Fourier spectrum of any function in AC^0 is concentrated on low-degree coefficients and then introduces the Low Degree Algorithm to learn the low-degree coefficients under the uniform distribution and thus generate a function approximately identical to the concept function. Later the Fourier concentration bound for

AC^0 has been improved in Boppana (1997); Håstad (2001); Tal (2014). Following Linial et al. (1993), some works present various Fourier concentration results for more expressive circuits augmented from AC^0 and thus gain corresponding learning results with the Low Degree Algorithm Jackson et al. (2002); Beigel (1994); Gopalan and Servedio (2010). In all these results, the uniform distribution is required.

There has been a few successful attempts to learning AC^0 under product distributions or other restricted distributions Furst et al. (1991); Blais et al. (2010); Ding et al. (2017). In the pursuit of learning AC^0 under arbitrary distributions (i.e. PAC learning), Bun and Thaler (2015) points out that if AC^0 could be computed by a polynomial threshold function (PTF) (of arbitrary degree) with weight at most W then under any distribution, some conjunction has correlation at least $1/W$ with some circuit being learned due to the discriminator lemma of Hajnal et al. (1993), and thus one can then apply an agnostic learning algorithm for conjunctions such as Kalai et al. (2008) combined with standard boosting techniques, to PAC learn it in $\max(\exp(\widetilde{O}(n^{1/2})), W)$ time. However, currently it is not known how to construct PTFs for AC^0 with moderate W. In fact, it is only known that AC^0 can be approximately computed by PTFs (of poly-logarithmic degrees) Ajtai and Ben-Or (1984); Aspnes et al. (1994a); Toda and Ogiwara (1992); Tarui (1993); Harsha and Srinivasan (2016) (these approximations will err on some fraction of inputs).

So a natural and long-standing question is whether we can PAC learn AC^0. Some works present efficient/sub-exponential-time learning algorithms for AC^0 of very restricted depth. Valiant (1984) presents a polynomial-time algorithm to PAC learn conjunctions. Klivans and Servedio (2004) presents an algorithm to PAC learn DNF formulae in time $n^{O(n^{1/3}\log n)}$. Since the question of learning disjunctions and CNF formulae can be reduced to the one of learning conjunctions and DNF formulae, we have that the AC^0 circuits of depths 1 and 2 are PAC learnable. For Boolean formulae of bounded size, O'Donnell and Servedio (2003) presents an algorithm to PAC learn s-size and d-depth Boolean formulae in time $n^{s^{1/2}(\log s)^{O(d)}}$ (thus for learning in sub-exponential-time, s should be slower than $n^2/\mathrm{polylog}(n)$). So far, it is unknown how to PAC learn AC^0 circuits of depth 3.

1.1. Our Results

In this paper we present an algorithm to PAC learn depth-3 AC^0 circuits of bounded top fanin in sub-exponential time. The top fanin of a circuit refers to the fanin of its output gate. A depth-3 AC^0 circuit is a circuit over $(x_1, \cdots, x_n, \overline{x}_1, \cdots, \overline{x}_n)$ which gates are arranged in three levels: from the bottom to the top, the three levels are OR-AND-OR or AND-OR-AND (the former is called a Σ_3-circuit and the latter is called a Π_3-circuit). Namely, a Σ_3-circuit is an OR of CNF formulae and a Π_3-circuit is an AND of DNF formulae over a same input $x \in \{0,1\}^n$.

Our result is that every depth-3 AC^0 circuit of top fanin K can be computed by a polynomial threshold function (PTF) of degree $\widetilde{O}(K \cdot n^{\frac{1}{2}})$, which means that it can be PAC learned in time $2^{\widetilde{O}(K \cdot n^{\frac{1}{2}})}$. Thus the interesting case is that $K = O(n^{\epsilon_0})$ for any $\epsilon_0 < \frac{1}{2}$, which leads the learning to sub-exponential-time.

Theorem 1 (Main Result.) *The class of all depth-3 AC^0 circuits of top fanin K can be learned to any accuracy and confidence (ϵ, δ) under any distribution in time $2^{\widetilde{O}(K \cdot n^{\frac{1}{2}})} \cdot poly(n, \frac{1}{\epsilon}, \log\frac{1}{\delta})$. (When $K = O(n^{\epsilon_0})$ for any $\epsilon_0 < \frac{1}{2}$, the running-time is sub-exponential.)*

Our Techniques. Our main work is to show that each such circuit of top fanin K can be computed by a PTF of degree $\widetilde{O}(\sqrt{n} \cdot K)$ which thus implies the PAC learning result in Theorem 1. So the key point is to establish the existence of such PTFs. (We note that using the known tools and methods in Beigel et al. (1995); Klivans and Servedio (2004); Klivans et al. (2004) can bring similar results but require a more factor of polylog(n) in degrees of the PTFs, which will be interpreted in footnotes later. So we use some specific construction.) In the following we sketch this with respect to Σ_3-circuits (learning Π_3-circuits of top fanin K can be reduced to the task of learning the Σ_3-circuits).

Let C be a Σ_3-circuit of top fanin K. Recall the gates of C from the bottom to the top is OR-AND-OR. We first use NOT and OR to replace the AND gates in the middle level. By the DeMorgan Law, AND can be replaced by a sub-circuit of three levels of NOT-OR-NOT. Thus C is equivalent to a 5-level circuit, in which from the bottom to the top the gates are of type OR-NOT-OR-NOT-OR. We then present approximation to the gates in C from the bottom to the top level by level.

First consider the OR gates in the bottom level. Recall that Nisan and Szegedy (1994) shows that each OR gate in the bottom level can be ϵ_1-uniformly approximated by a real multivariate polynomial of degree $O(\sqrt{n}\ln\frac{1}{\epsilon_1})$. By ϵ_1-uniform approximation, we mean that for each $x \in \{0,1\}^n$, the absolute value of the difference between the outputs of the polynomial and the OR gate is bounded by ϵ_1. In this paper we choose $\epsilon_1 = n^{-2\log n}$. Since level 2 consists of NOT gates, flipping outputs of the polynomials for level 1 (bottom level) gives polynomials for the NOT gates.

Then the difficult task is to construct polynomials for the gates in higher levels. If we use the result in Nisan and Szegedy (1994) again to approximate the OR gates in level 3, the composed polynomials would be of degree n, which is thus trivial.

So we adopt a new way to approximate these OR gates. Instead of doing this with polynomials, we do it with rational functions. A rational function is a function of form $\frac{f(x)}{g(x)}$, in which $f(x), g(x)$ are polynomials. We show that each OR gate in level 3 can be $n^{-\Theta(\log n)}$-uniformly approximated by a rational function f/g, in which f, g are of degree $O(\sqrt{n}\ln\frac{1}{\epsilon_1})$. Then flipping outputs of these rational functions gives correct approximation to the NOT gates in level 4. (The method of rational function approximation was previously adopted by Klivans *et al.* Klivans et al. (2004) in learning intersections of halfspaces, based on the earlier works of Beigel et al. (1995); Newman (1964).)

Let us finally consider the output OR gate. Let $\frac{f_1}{g_1}, \cdots, \frac{f_K}{g_K}$ denote the approximation to all NOT gates in level 4, in which each $\frac{f_j}{g_j}$ differs from the corresponding NOT gate by $n^{-\Theta(\log n)}$ for all j for any input x. If $C(x) = 0$ then all NOT gates in level 4 output 0, which indicates $\sum_{j=1}^{K}\frac{f_j}{g_j} < \frac{1}{2}$. If $C(x) = 1$ then at least one such NOT gate outputs 1, which indicates $\sum_{j=1}^{K}\frac{f_j}{g_j} > \frac{1}{2}$. Equivalently, $C(x) = 0$ if and only if $\sum_{j=1}^{K}(f_j\prod_{i\in[K],i\neq j} g_i) < \frac{1}{2}\prod_{i\in[K]} g_i$, i.e. $\sum_{j=1}^{K}(f_j\prod_{i\in[K],i\neq j} g_i) - \frac{1}{2}\prod_{i\in[K]} g_i < 0$.

Thus let $F(x)$ denote the polynomial in the left side of the above inequality. Then it is of degree $O(\sqrt{n}\ln\frac{1}{\epsilon_1})\cdot K$. Therefore, it can be seen that $\mathsf{Sign}(F(x))$ is a PTF that computes $C(x)$ exactly.

1.2. Organization

The rest of the paper is arranged as follows. Section 2 presents the preliminaries. In Section 3 we present the details of how to construct the PTF for each Σ_3-circuit of bounded top fanin and sketch the learning strategy. In Section 4 we show the learning algorithm in detail.

2. Preliminaries

This section contains the notations and definitions used throughout this paper.

2.1. Basic Notions

We use $[n]$ to denote the integers in $[1, n]$. Let $\mathsf{Sign}(y)$ denote the sign function that outputs 1 if $y > 0$ and outputs 0 if $y < 0$.

A polynomial threshold function over n boolean variables $x = (x_1, \cdots, x_n)$, is a boolean function $h : \{0,1\}^n \to \{0,1\}$ defined as $h(x) = \mathsf{Sign}(g(x_1, \cdots, x_n))$, where $g(x_1, \cdots, x_n)$ is a real polynomial over $(x_1, \cdots, x_n)$. The degree of h refers to that of g.

We say that a real multivariate function $f(x)$ ϵ-uniformly approximates $g(x)$ if for all $x \in \{0,1\}^n$, $|f(x) - g(x)| \le \epsilon$.

A multivariate rational function over $x = (x_1, \cdots, x_n)$ is a function of form $\frac{f(x)}{g(x)}$, where f, g are polynomials over x.

2.2. PAC Learning

Let $\mathcal{C}$ denote a class of functions. In the PAC learning model Valiant (1984), a labeled example is a pair $(x, f(x))$, where $x \in X$ is an input and $f(x)$ is the value of the target function $f \in \mathcal{C}$ on the input x. A training sample labeled by f is of the form $((x^1, f(x^1)), \cdots, (x^m, f(x^m)))$, in which each $(x^i, f(x^i))$ denotes the ith labeled example $1 \le i \le m$.

Definition 2 (PAC Learning) An algorithm L is called a learner for $\mathcal{C}$ under distribution D over X, if it is given a training sample in which each x is sampled from D independently and its label is $f(x)$ for some unknown $f \in \mathcal{C}$, $\epsilon, \delta \in (0,1)$, with probability at least $1-\delta$, L outputs a function h (not necessarily in $\mathcal{C}$) such that $\Pr[f(x) \ne h(x)] < \epsilon$ for $x \leftarrow D$.

If L can work under any D, we say L PAC (Probably Approximately Correct) learns $\mathcal{C}$ or simply learns $\mathcal{C}$. We refer to ϵ as the accuracy parameter and δ as the confidence parameter. We call L efficient if its running-time is poly$(n, \frac{1}{\epsilon}, \frac{1}{\delta})$ and call L non-trivial if its running-time is bounded by a sub-exponential in $(n, \frac{1}{\epsilon}, \frac{1}{\delta})$.

2.3. AC^0 Circuits

Let AC^0 denote the class of all functions computable by polynomial-size constant-depth unbounded fanin circuits (of AND, OR, NOT gates and of binary output). We also use AC^0 to denote the class of all polynomial-size constant-depth unbounded fanin circuits.

This paper focuses on those circuits over $(x_1, \cdots, x_n, \overline{x}_1, \cdots, \overline{x}_n)$ whose gates are arranged in three levels. From the bottom to the top, the three levels are OR-AND-OR or AND-OR-AND. We call a circuit of form OR-AND-OR a Σ_3-circuit and call a circuit of AND-OR-AND a Π_3-circuit. Namely, a Σ_3-circuit is an OR of CNF formulae and a Π_3-circuit is an AND of DNF formulae.

The fanin of a gate is the number of input wires to it. The top fanin refers to the fanin of the top gate.

3. Polynomial Threshold Function Representation

In this section we show that each Σ_3-circuit of the bounded top fanin can be computed by a PTF exactly. In Section 3.1 we present the uniform approximation to gates of each such circuit level by level (except for the output gate) using rational functions. In Section 3.2 we present the PTF and learning strategy.

3.1. Approximation with Rational Functions

Let C be a Σ_3-circuit over $(x_1, \cdots, x_n, \overline{x}_1, \cdots, \overline{x}_n)$, where x_i denotes the ith bit of x and $\overline{x}_i$ denotes $1 - x_i$, $1 \le i \le n$.

For C, assume there are s_1 OR gates in level 1, s_2 AND gates in level 2 and one OR gate in the top. Let $\mathrm{OR}_i^1, 1 \le i \le s_1$ denote all gates in level 1, $\mathrm{AND}_i^1, 1 \le i \le s_2$ denote all gates in level 2. s_1 is polynomial in n and s_2 is the top fanin.

Note that for any AND (or OR) operation, which has arbitrary k bits $(y_1, \cdots, y_k)$ as input, we have that $\mathrm{AND}(y_1, \cdots, y_k) = \mathrm{AND}(y_1, \cdots, y_k, y_j)$ for any $j \in [1, k]$. This means that by repeating any input bit many times, we can assume any OR/AND gate has a specified fan-in. So w.l.o.g. assume all $\mathrm{AND}_i^2, 1 \le i \le s_2$ have the same fixed fan-in s'.

In the following we will present polynomials/rational functions approximating gates in C level by level from the bottom to the top. Notice that all input bits to C are $(x_1, \cdots, x_n, \overline{x}_1, \cdots, \overline{x}_n)$. If an OR gate in level 1 has more than n bits as input, there is at least an i such that $x_i, \overline{x}_i$ simultaneously appear in the input to this OR gate, which leads to that it outputs 1 always. So it can be functionally equivalently replaced by another OR gate which just has two bits $x_i, \overline{x}_i$ as input. Thus we can consider the number of input bits to each OR gate in level 1 is always bounded by n.

First let us recall the following result of uniformly approximating the OR operation over n bits (which proof can be referred to Jukna (2012) Lemma 2.6) that gives polynomials for approximating the OR gates in level 1.

Claim 1 *(Nisan and Szegedy (1994)) For each $\epsilon_1 < \frac{1}{2}$, there is a real multivariate polynomial $p(\cdot)$ of degree $O(\sqrt{n} \ln \frac{1}{\epsilon_1})$ such that for any $z \in \{0, 1\}^n$, $|OR(z) - p(z)| \le \epsilon_1$.*

We will frequently use $\mathrm{OR}_i^1, 1 \le i \le s_1$ to denote its output in the computation of $C(x)$ for any $x \in \{0, 1\}^n$ (and also adopt this usage for the gates in higher levels). By Claim 1, there are real multivariate polynomials $p^1_1, \cdots, p^1_{s_1}$ of degree $O(\sqrt{n}\ln\frac{1}{\epsilon_1})$ over x such that $|p^1_i(x) - \mathrm{OR}^1_i| \le \epsilon_1, 1 \le i \le s_1$ for all $x \in \{0,1\}^n$. In this paper for simplicity of statement choose $\epsilon_1 = n^{-2\log n}$. [1]

Then consider the computation in the AND gates in level 2. Assume that AND^2_j has as input the outputs of $\mathrm{OR}^1_{j_1}, \cdots, \mathrm{OR}^1_{j_{s'}}$, $1 \le j \le s_2$. Note that by applying the DeMorgan Law, we have that for any j,

$$\mathrm{AND}^2_j(\mathrm{OR}^1_{j_1}, \cdots, \mathrm{OR}^1_{j_{s'}}) = \neg\neg\mathrm{AND}^2_j(\mathrm{OR}^1_{j_1}, \cdots, \mathrm{OR}^1_{j_{s'}}) = \neg\mathrm{OR}(\neg\mathrm{OR}^1_{j_1}, \cdots, \neg\mathrm{OR}^1_{j_{s'}})$$

So we replace each AND gate in the middle level by the following sub-circuit: on input many bits (coming from some OR gates in level 1) compute the NOT of each input bit first, then OR of the outputs of all the NOT gates and finally output the NOT of the output of the OR gate.

In this way, C is equivalent to a level-5 circuit, in which

- The bottom level is same as C.
- The second level consists of s_1 NOT gates, denoted $\mathrm{NOT}^2_1, \cdots, \mathrm{NOT}^2_{s_1}$, which flip the outputs of $\mathrm{OR}^1_1, \cdots, \mathrm{OR}^1_{s_1}$.
- The third level consists of s_2 OR gates, denoted $\mathrm{OR}^3_1, \cdots, \mathrm{OR}^3_{s_2}$, in which OR^3_j has as input the outputs of $\mathrm{NOT}^2_{j_1}, \cdots, \mathrm{NOT}^2_{j_{s'}}$.
- The fourth level consists of s_2 NOT gates, denoted $\mathrm{NOT}^4_1, \cdots, \mathrm{NOT}^4_{s_2}$, which flip the outputs of $\mathrm{OR}^3_1, \cdots, \mathrm{OR}^3_{s_2}$.
- The top level is the output OR gate, i.e. the output gate of C, denoted OR^5, which outputs the OR of the outputs of $\mathrm{NOT}^4_1, \cdots, \mathrm{NOT}^4_{s_2}$.

So in the following we think of C having this 5-level structure. Then let us consider the NOT gates in level 2. Since $p^1_i(x)$ is close to the output of OR^1_i, $1 - p^1_i(x)$ is also close to the output of NOT^2_i. Let $p^2_i(x)$ denote $1 - p^1_i(x)$. Then by Claim 1 we have the following claim.

Claim 2 *For each input $x \in \{0,1\}^n$, $|p^2_i(x) - NOT^2_i| \le \epsilon_1$.*

Now let us consider how to approximate the OR gates in level 3. This is actually a key step of the whole construction. Recall that previous works such as Aspnes et al. (1994b) present low-degree polynomials to approximate OR under any distribution. These polynomials can compute OR correctly for majority fraction of inputs, and however have large deviation for other inputs. So we do not adopt these known polynomial approximation. Instead, we will use rational functions to approximate the OR gates in this level.

1. The uniform approximation in Claim 1 also holds for the AND operation. Then any s-term t-DNF formula (i.e. one containing s terms, each of which consists of at most t literals) can be computed by a PTF, in which the polynomial is the sum of all the approximation to its terms when $\epsilon_1 < \frac{1}{2s}$. Since each approximation is of degree $O(\sqrt{t}\log(1/\epsilon_1))$, the PTF is of degree $O(\sqrt{t}\log(1/\epsilon_1)) = O(\sqrt{t}\log s)$. This achieves a same result shown in Klivans and Servedio (2004).

Recall that Beigel et al. (1995) presents an approximation to the sign function with rational functions which can be employed here to approximate OR. But it would result in a more factor of polylog(n) in degrees of the numerator and denominator in rational functions when compared to our specific construction in the following.[2] (Note that we can also employ the results and methods in Klivans and Servedio (2004); Klivans et al. (2004) to construct the PTF, but it still leads to the polylog(n)-factor augment in degrees.[3] [4])

So we use a new rational approximation. Take OR_j^3 for example. Let y_{j_k} denote the output of $\mathrm{NOT}_{j_k}^2, 1 \le k \le s'$, where s' is polynomial in n. Define $p(y_{j_1}, \cdots, y_{j_{s'}})$ as follows.

$$p(y_{j_1}, \cdots, y_{j_{s'}}) = \frac{y_{j_1} + \cdots + y_{j_{s'}}}{y_{j_1} + \cdots + y_{j_{s'}} + n^{\log n}\epsilon_1}$$

Then we can see that $0 < p(y) \le 1$ for any $y = (y_{j_1}, \cdots, y_{j_{s'}})$ and $p(y) = 0$ if $\mathrm{OR}_j^3(y) = 0$, and $p(y) \in [\frac{1}{1+n^{\log n}\epsilon_1}, \frac{s'}{s'+n^{\log n}\epsilon_1}]$ if $\mathrm{OR}_j^3(y) = 1$.

Let p_j^3 denote $p(p_{j_1}^2, \cdots, p_{j_{s'}}^2)$. Then $p_j^3(x)$ is a rational function, in which the polynomials in numerator and denominator are of degree $O(\sqrt{n} \ln \frac{1}{\epsilon_1})$, since each $p_{j_k}^2$ is of degree $O(\sqrt{n} \ln \frac{1}{\epsilon_1})$. Also note that each $p_{j_k}^2$ is ϵ_1-close to 0/1 and thus the denominator of p_j^3 will not equal 0, which ensures that the definition of p_j^3 is meaningful. Now we have the following claim.

Claim 3 *For each input $x \in \{0,1\}^n$, $|p_j^3(x)| < \frac{poly(n)}{n^{\log n}}$ if $OR_j^3 = 0$, and $p_j^3(x) \in (1 - \frac{n^{-\log n}}{poly(n)}, 1)$ if $OR_j^3 = 1$.*

Proof Recall that y_{j_k} denotes the output of $\mathrm{NOT}_{j_k}^2, 1 \le k \le s'$ and $y = (y_{j_1}, \cdots, y_{j_{s'}})$. When $\mathrm{OR}_j^3 = 0$, all y_{j_k} are 0. Thus we have that $p(y) = 0$, i.e. $\frac{y_{j_1}+\cdots+y_{j_{s'}}}{y_{j_1}+\cdots+y_{j_{s'}}+n^{\log n}\epsilon_1} = 0$. By

2. Beigel et al. (1995) shows that for every $l, t \ge 1$ there is a rational function $P_t^l(z)$ over real z satisfying that it is in $[1, 1 + \frac{1}{l}]$ if $z \in [1, 2^t]$ and it is in $[-1 - \frac{1}{l}, -1]$ if $z \in [-2^t, -1]$ and degrees of the numerator and denominator of $P_t^l(z)$ sum into $O(t \log l)$. For each OR gate in level 3, letting z denote the sum of all its inputs, z is $[1, s']$ if the OR outputs 1 and $z = 0$ if it outputs 0. That implies that $1 - 2z = 1$ if the NOT gate in level 4 connected to this OR gate outputs 1 and $1 - 2z \in [1 - 2s', -1]$ if it outputs 0. Choose $t = \log^2 n, l = n^{\log n}$. Then $P_t^l(1 - 2z)$ of degree $O(t \log l)$ $\frac{1}{l}$-approximates the NOT gate. Thus since later we need to replace each input to the OR gate by p_{jk}^2 in defining p_j^3 and p_j^4, ϵ_1 should be set to $n^{-\Omega(t \log l)}$. So compared to our construction, adopting the general conclusion in Beigel et al. (1995) leads to a more factor of $O(t \log l \cdot \log(1/\epsilon_1)) = \mathrm{polylog}(n)$ in degrees when approximating NOT gates in level 4.
3. Consider the question of learning Π_3-circuits of top fanin $s_2 < n$, each of which is an AND of s_2 DNF formulae. Klivans and Servedio (2004) shows that every s-term DNF formula can be computed by a PTF of degree $O(\sqrt{n} \log s)$ and of weight $w = n^{O(\sqrt{n} \log s)}$. When s is an arbitrary polynomial, $w = n^{O(\sqrt{n} \log^2 n)}$. Choose $t = \log w$ and $l = O(n)$. By Beigel et al. (1995) and the arguments in Klivans et al. (2004), each such circuit, i.e. an AND of s_2 PTFs, can be computed by a PTF of degree $O(s_2 \cdot t \log l)$, which also contains a more factor of polylog(n) than our construction.
4. Klivans and Servedio (2004) also shows that every s-term DNF can be computed by a PTF of degree $O(n^{1/3} \log s)$. However, the weight of the PTF can be 2^{n^c} for $c \ge 1$. So if following the way in footnote 3 with this result, we only have that any Π_3-circuit of top fanin s_2 can be computed by a PTF of degree n, which results in exponential-time learning.

Claim 2, $|p^2_{j_k}(x) - y_{j_k}| < \epsilon_1$ for any k. Thus

$$|\frac{p^2_{j_1} + \cdots + p^2_{j_{s'}}}{p^2_{j_1} + \cdots + p^2_{j_{s'}} + n^{\log n}\epsilon_1}| < \frac{|y_{j_1} + \cdots + y_{j_{s'}}| + s'\epsilon_1}{-|y_{j_1} + \cdots + y_{j_{s'}}| - s'\epsilon_1 + n^{\log n}\epsilon_1}$$
$$= \frac{s'\epsilon_1}{-s'\epsilon_1 + n^{\log n}\epsilon_1} = \frac{s'}{-s' + n^{\log n}} < \frac{\text{poly}(n)}{n^{\log n}}$$

When $\text{OR}^3_j = 1$, there is at least one y_{j_k} which is 1. Thus $\frac{y_{j_1}+\cdots+y_{j_{s'}}}{y_{j_1}+\cdots+y_{j_{s'}}+n^{\log n}\epsilon_1} \in [\frac{1}{1+n^{\log n}\epsilon_1}, \frac{s'}{s'+n^{\log n}\epsilon_1}]$, which is less than 1. Then there exists a $s_0 \in [-s', s']$ such that

$$\frac{p^2_{j_1} + \cdots + p^2_{j_{s'}}}{p^2_{j_1} + \cdots + p^2_{j_{s'}} + n^{\log n}\epsilon_1} = \frac{y_{j_1} + \cdots + y_{j_{s'}} + s_0\epsilon_1}{y_{j_1} + \cdots + y_{j_{s'}} + s_0\epsilon_1 + n^{\log n}\epsilon_1}$$
$$= 1 - \frac{n^{\log n}\epsilon_1}{y_{j_1} + \cdots + y_{j_{s'}} + s_0\epsilon_1 + n^{\log n}\epsilon_1} > 1 - \frac{n^{-\log n}}{y_{j_1} + \cdots + y_{j_{s'}}} \geq 1 - \frac{n^{-\log n}}{\text{poly}(n)}$$

in which the last equality follows from that $n^{\log n}\epsilon_1 = n^{-\log n}$ and $y_{j_1} + \cdots + y_{j_{s'}}$ is at most a polynomial quantity. The claim holds.

■

Then consider the NOT gates in level 4. Let $p^4_j(x)$ denote $1 - p^3_j(x)$. Thus p^4_i is still a rational function, in which the polynomials in numerator and denominator are of degree $O(\sqrt{n}\ln\frac{1}{\epsilon_1})$. Then by Claim 3 we have the following result.

Claim 4 *For each input $x \in \{0,1\}^n$, $p^4_j(x) \in (0, \frac{n^{-\log n}}{poly(n)})$ if NOT$^4_j = 0$, and $p^4_j(x) \in (1 - \frac{poly(n)}{n^{\log n}}, 1 + \frac{poly(n)}{n^{\log n}})$ if NOT$^4_j = 1$.*

Then let us consider the final output gate, i.e. OR^5, which has the outputs of the NOT gates in level 4 as input. If all these outputs are 0, then it outputs 0. Otherwise, it outputs 1. Thus we have the following claim.

Claim 5 *For each input $x \in \{0,1\}^n$, the OR^5 outputs 0 if and only if $\sum_{j=1}^{s_2} p^4_j(x) = n^{-\Theta(\log n)}$ and OR^5 outputs 1 if and only if $\sum_{j=1}^{s_2} p^4_i(x) > 1 - n^{-\Theta(\log n)}$.*

Proof It can be seen that OR^5 outputs 0 if and only if all NOT^4_j output 0. By Claim 4, each NOT^4_j outputs 0 if and only if $|p^4_j(x)| < \frac{n^{-\log n}}{\text{poly}(n)}$. Thus OR^5 outputs 0 if and only if the following holds.

$$\sum_{j=1}^{s_2} p^4_j(x) < s_2 \cdot \frac{n^{-\log n}}{\text{poly}(n)} = n^{-\Theta(\log n)}$$

On the contrary, OR^5 outputs 1 if and only if at least one NOT^4_j outputs 1, which is equivalent to that

$$\sum_{j=1}^{s_2} p^4_j(x) > 1 - \frac{\text{poly}(n)}{n^{\log n}} = 1 - n^{-\Theta(\log n)}$$

The claim holds.

■

We remark that although we could now present the PTF computing C using Claim 5, for simplicity we put forward the following statement of clear quantity that is sufficient for further analysis.

Claim 6 *For each input $x \in \{0,1\}^n$ and for sufficiently large n, the OR^5 outputs 0 if and only if $\sum_{j=1}^{s_2} p_j^4(x) < \frac{1}{2}$ and OR^5 outputs 1 if and only if $\sum_{j=1}^{s_2} p_i^4(x) > \frac{1}{2}$.*

3.2. The Learning Strategy

We now present the learning strategy. Notice that each p_j^4 is a rational function, which can thus be represented as $\frac{f_j}{g_j}$, where f_j, g_j are polynomials of degree $O(\sqrt{n}\ln\frac{1}{\epsilon_1})$. Thus

$$\sum_{j=1}^{s_2} p_j^4(x) = \frac{f_1}{g_1} + \cdots + \frac{f_{s_2}}{g_{s_2}}$$

By Claim 6 $\mathrm{OR}^5 = 0$ is equivalent to $\frac{f_1}{g_1} + \cdots + \frac{f_{s_2}}{g_{s_2}} < \frac{1}{2}$ and $\mathrm{OR}^5 = 1$ is equivalent to $\frac{f_1}{g_1} + \cdots + \frac{f_{s_2}}{g_{s_2}} > \frac{1}{2}$. Multiplying $\prod_{i\in[s_2]} g_i$ to both sides, we have the following inequalities.

$$\begin{cases} \sum_{j=1}^{s_2}(f_j \prod_{i\in[s_2],i\neq j} g_i) < \frac{1}{2}\cdot\prod_{i\in[s_2]} g_i, \text{ if } \mathrm{OR}^5 = 0 \\ \sum_{j=1}^{s_2}(f_j \prod_{i\in[s_2],i\neq j} g_j) > \frac{1}{2}\cdot\prod_{i\in[s_2]} g_i, \text{ if } \mathrm{OR}^5 = 1 \end{cases}$$

Let $F(x)$ denote the following polynomial.

$$F(x) = \sum_{j=1}^{s_2}(f_j \prod_{i\in[s_2],i\neq j} g_i) - \frac{1}{2}\cdot\prod_{i\in[s_2]} g_i$$

Thus $F(x) < 0$ if $C(x) = 0$ and $F(x) > 0$ if $C(x) = 1$.

Let $T = O(\sqrt{n}\ln\frac{1}{\epsilon_1})\cdot s_2$. We have that $F(x)$ is of degree T. So $F(x)$ can be represented as

$$F(x) = \sum_{S:|S|\leq T} \alpha_S \prod_{i\in S} x_i$$

in which all S's are subsets of $[n]$ and α_S's are real coefficients.

For a labeled example $(x, C(x))$, if $C(x) = 0$, we can construct an inequality as follows.

$$\sum_{S:|S|\leq T} \alpha_S \prod_{i\in S} x_i < 0$$

If $C(x) = 1$, we construct the inequality as follows.

$$\sum_{S:|S|\leq T} \alpha_S \prod_{i\in S} x_i > 0$$

So when given m examples, we obtain m inequalities as above shows. An important thing is that due to the generation of these inequalities, we are ensured there exists at least one solution to all coefficients α_S's. Then by using any linear programming algorithm, we can recover a solution of all α_S's, which are actually consistent with all the m examples. Thus when m is large enough, these α_S's can be used to evaluate a new input x (with the strategy that the output is 1 if $\sum_{S:|S|\le T} \alpha_S \prod_{i\in S} x_i > 0$ and is 0 otherwise.) We will formalize this strategy in the next section.

4. The Learning Algorithm

In this section we present the learning algorithm for any Σ_3-circuit C which top fanin is s_2. In Section 4.1 we present the sample complexity for learning C. In Section 4.2 we present the learning algorithm.

4.1. Sample Complexity

Let $\mathcal{H}_n$ denote the class of all functions computable by $\mathsf{Sign}(\sum_{S:|S|\le T} \alpha_S \prod_{i\in S} x_i)$ with different coefficients α_S's, where recall $T = O(\sqrt{n}\ln\frac{1}{\epsilon_1})\cdot s_2$.

Notice that due to the construction of $F(x)$ in the previous section, we have $C(x) = \mathsf{Sign}(F(x))$ for any Σ_3-circuit C with top fanin s_2. Since $\mathsf{Sign}(F(x)) \in \mathcal{H}_n$, all such Σ_3-circuits are in $\mathcal{H}_n$. So it suffices to adopt the VC-dimension of $\mathcal{H}_n$ to drive the required number of examples for learning C. First recall the following result.

Claim 7 *(Mixon and Peterson (2015)) The VC-dimension of $\mathcal{H}_n$ is $2n^{T+1} - O(n^T \cdot T\log n) = O(n^{T+1})$.*

Let D denote any distribution over $\{0,1\}^n$, $(x^1, C(x^1), \cdots, x^m, C(x^m))$ be m examples labeled by any circuit C where each $x^i \leftarrow D$ independently. An algorithm L is called a consistent-hypothesis-finder if L on input any m labeled examples $(x^i, C(x^i)), 1 \le i \le m$ can output a hypothesis $h \in \mathcal{H}_n$ satisfying $h(x^i) = C(x^i)$ for all i.

Proposition 3 *(Blumer et al. (1989)) For any $\mathcal{H}_n$, choose $m = \frac{4}{\epsilon}(\text{VC-dim}(\mathcal{H}_n)\ln(\frac{12}{\epsilon}) + \ln(\frac{2}{\delta}))$. If L is a consistent-hypothesis-finder for m examples labeled by some function in $\mathcal{H}_n$ under any distribution D, then it is a PAC learning algorithm that can learn $\mathcal{H}_n$ to accuracy and confidence (ϵ, δ) under D.*

Returning to our setting, to learn Σ_3-circuits with top fanin s_2 (contained in $\mathcal{H}_n$), m can be set to $O(\frac{1}{\epsilon}(n^{T+1}\ln(\frac{12}{\epsilon}) + \ln(\frac{2}{\delta})))$.

4.2. Actual Description

Our learning algorithm for all Σ_3-circuits with top fanin s_2 is shown in Algorithm 1, in which m is the number specified above.

Then we show that Algorithm 1 can learn any such C under any distribution D.

Theorem 4 *Algorithm 1 can with probability at least $1-\delta$ output a hypothesis h satisfying $\Pr[h(x) \ne C(x)] < \epsilon$ for $x \leftarrow D$ in time poly(m).*

Algorithm 1: The learning algorithm for all Σ_3-circuits with top fanin s_2

Input:

- m labeled examples of form $(x, C(x))$ where x is drawn from D independently and C is a Σ_3-circuits with top fanin s_2.
- ϵ, δ and s_2.

Output: a hypothesis h'.

1. Choose $\epsilon_1 = n^{-2\log n}$ and let $T = O(\sqrt{n} \ln \frac{1}{\epsilon_1}) \cdot s_2$.
2. For each example $(x, C(x))$, if $C(x) = 1$, generate an inequality $\sum_{S:|S|\leq T} \alpha_S \prod_{i\in S} x_i > 0$. If $C(x) = 0$, generate an inequality $\sum_{S:|S|\leq T} \alpha_S \prod_{i\in S} x_i < 0$.

   Thus the learning algorithm finally generates m linear inequalities, in which all α_S's are unknown coefficients.
3. Run any linear programming algorithm on input the m inequalities to find a solution of all α_S's. (At least one solution exists.) Denote by α'_S's the solution.
4. Output the function h' as the learned hypothesis of C. h' has all α'_S's hardwired and on input any $x \in \{0,1\}^n$, outputs $\mathsf{Sign}(\sum_{S:|S|\leq T} \alpha'_S \prod_{i\in S} x_i)$.

End Algorithm

Proof First, it can be seen that when transforming the m examples to the inequalities, we are ensured that the inequalities have at least one solution. This is so because when given C, if we follow the construction strategy to generate $\mathsf{Sign}(F(x))$, it is of course consistent with the training examples. This ensures that the linear programming algorithm can find a solution of α_S's (no matter whether they are identical to the original ones or not).

Thus the $h' \in \mathcal{H}_n$ output by Algorithm 1 is actually a consistent hypothesis. Considering the choice of m, by Proposition 3, h' is indeed a learned hypothesis. Lastly we can see that Algorithm 1 runs in time polynomial in m.

■

We present several remarks on this learning result. First, for learning in sub-exponential time, the top fanin s_2 should be bounded by n^{ϵ_0} for any $\epsilon_0 < \frac{1}{2}$, which thus ensures the running-time is $n^{\widetilde{O}(n^{\frac{1}{2}+\epsilon_0})}$, a sub-exponential time.

Second, when obtaining a learning algorithm for Σ_3 circuits of top fanin s_2, we can also learn all Π_3 circuits of top fanin s_2, i.e. AND of s_2 DNF formulae. Let C' be such a Π_3-circuit. Notice that $\overline{C'}$ is a Σ_3-circuit of top fanin s_2. So when given m examples of form $(x, C'(x))$, we first generate m examples $(x, 1 - C'(x))$ in which $1 - C'(x)$ is equal to $\overline{C'}(x)$. Run Algorithm 1 on the new examples to obtain a hypothesis h' of $\overline{C'}$, and finally denote by $1 - h'$ the learned hypothesis of C'.

Third, for learning an arbitrary AC^0 circuit C with top fanin s_2, when given m examples, we can first assume C is a Σ_3-circuit and thus run Algorithm 1 to output a hypothesis h'.

If h' is consistent with all the examples, by Proposition 3, h' is indeed a desired learned hypothesis. Otherwise, it implies that C is a Π_3-circuit of top fanin s_2. Thus we can apply the learning strategy in the previous paragraph to generate a desired learned hypothesis for C. Therefore we complete the proof of Theorem 1.

Acknowledgments

We are grateful to the reviewers of ALT 2017 for their useful comments. This work is supported by National Natural Science Foundation of China (Grant No. 61572309) and National Cryptography Development Fund of China (Grant No. MMJJ20170128) and Major State Basic Research Development Program (973 Plan) of China (Grant No. 2013CB338004).

References

Miklós Ajtai and Michael Ben-Or. A theorem on probabilistic constant depth computations. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*, pages 471–474, 1984. doi: 10.1145/800057.808715. URL http://doi.acm.org/10.1145/800057.808715.

James Aspnes, Richard Beigel, Merrick Furst, and Steven Rudich. The expressive power of voting polynomials. *Combinatorica*, 14(2):1–14, 1994a.

James Aspnes, Richard Beigel, Merrick L. Furst, and Steven Rudich. The expressive power of voting polynomials. *Combinatorica*, 14(2):135–148, 1994b.

Richard Beigel. When do extra majority gates help? polylog(n) majority gates are equivalent to one. *Computational Complexity*, 4:314–324, 1994.

Richard Beigel, Nick Reingold, and Daniel A. Spielman. PP is closed under intersection. *J. Comput. Syst. Sci.*, 50(2):191–202, 1995. doi: 10.1006/jcss.1995.1017. URL https://doi.org/10.1006/jcss.1995.1017.

Eric Blais, Ryan O'Donnell, and Karl Wimmer. Polynomial regression under arbitrary product distributions. *Machine Learning*, 80(2-3):273–294, 2010. doi: 10.1007/s10994-010-5179-6. URL http://dx.doi.org/10.1007/s10994-010-5179-6.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989.

Ravi B. Boppana. The average sensitivity of bounded-depth circuits. *Inf. Process. Lett.*, 63 (5):257–261, 1997. doi: 10.1016/S0020-0190(97)00131-2. URL http://dx.doi.org/10.1016/S0020-0190(97)00131-2.

Mark Bun and Justin Thaler. Hardness amplification and the approximate degree of constant-depth circuits. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 268–280, 2015. doi: 10.1007/978-3-662-47672-7_22. URL http://dx.doi.org/10.1007/978-3-662-47672-7_22.

Ning Ding, Yanli Ren, and Dawu Gu. Learning ac^0 under k-dependent distributions. In T. V. Gopal, Gerhard Jäger, and Silvia Steila, editors, *Theory and Applications of Models of Computation - 14th Annual Conference, TAMC 2017, Bern, Switzerland, April 20-22, 2017, Proceedings*, volume 10185 of *Lecture Notes in Computer Science*, pages 187–200, 2017. ISBN 978-3-319-55910-0. doi: 10.1007/978-3-319-55911-7_14. URL https://doi.org/10.1007/978-3-319-55911-7_14.

Merrick L. Furst, Jeffrey C. Jackson, and Sean W. Smith. Improved learning of ac^0 functions. In Manfred K. Warmuth and Leslie G. Valiant, editors, *Proceedings of the Fourth Annual Workshop on Computational Learning Theory, COLT 1991, Santa Cruz, California, USA, August 5-7, 1991*, pages 317–325. Morgan Kaufmann, 1991. ISBN 1-55860-213-5. URL http://dl.acm.org/citation.cfm?id=114866.

Parikshit Gopalan and Rocco A. Servedio. Learning and lower bounds for ac^0 with threshold gates. In Maria J. Serna, Ronen Shaltiel, Klaus Jansen, and José D. P. Rolim, editors, *APPROX-RANDOM*, volume 6302 of *Lecture Notes in Computer Science*, pages 588–601. Springer, 2010. ISBN 978-3-642-15368-6.

András Hajnal, Wolfgang Maass, Pavel Pudlák, Mario Szegedy, and György Turán. Threshold circuits of bounded depth. *J. Comput. Syst. Sci.*, 46(2):129–154, 1993. doi: 10.1016/0022-0000(93)90001-D. URL https://doi.org/10.1016/0022-0000(93)90001-D.

Prahladh Harsha and Srikanth Srinivasan. On polynomial approximations to \${AC}ˆ0\$. *CoRR*, abs/1604.08121, 2016. URL http://arxiv.org/abs/1604.08121.

Johan Håstad. A slight sharpening of LMN. *J. Comput. Syst. Sci.*, 63(3):498–508, 2001. doi: 10.1006/jcss.2001.1803. URL http://dx.doi.org/10.1006/jcss.2001.1803.

Jeffrey C. Jackson, Adam Klivans, and Rocco A. Servedio. Learnability beyond ac0. In *IEEE Conference on Computational Complexity*, page 26. IEEE Computer Society, 2002. ISBN 0-7695-1468-5.

Stasys Jukna. *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer, 2012. ISBN 978-3-642-24507-7. doi: 10.1007/978-3-642-24508-4. URL https://doi.org/10.1007/978-3-642-24508-4.

Adam Tauman Kalai, Adam R. Klivans, Yishay Mansour, and Rocco A. Servedio. Agnostically learning halfspaces. *SIAM J. Comput.*, 37(6):1777–1805, 2008. doi: 10.1137/060649057. URL http://dx.doi.org/10.1137/060649057.

Adam R. Klivans and Rocco A. Servedio. Learning DNF in time $2^{\tilde{o}(n^{1/3})}$. *J. Comput. Syst. Sci.*, 68(2):303–318, 2004. doi: 10.1016/j.jcss.2003.07.007. URL https://doi.org/10.1016/j.jcss.2003.07.007.

Adam R. Klivans, Ryan O'Donnell, and Rocco A. Servedio. Learning intersections and thresholds of halfspaces. *J. Comput. Syst. Sci.*, 68(4):808–840, 2004. doi: 10.1016/j.jcss.2003.11.002. URL https://doi.org/10.1016/j.jcss.2003.11.002.

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

Dustin G. Mixon and Jesse Peterson. Learning boolean functions with concentrated spectra. *CoRR*, abs/1507.04319, 2015. URL http://arxiv.org/abs/1507.04319.

D. J. Newman. Rational function approximation to $|x|$. *Michigan Mathematical Journal*, 11:11–14, 1964.

Noam Nisan and Mario Szegedy. On the degree of boolean functions as real polynomials. *Computational Complexity*, 4:301–313, 1994. doi: 10.1007/BF01263419. URL https://doi.org/10.1007/BF01263419.

Ryan O'Donnell and Rocco A. Servedio. New degree bounds for polynomial threshold functions. In Lawrence L. Larmore and Michel X. Goemans, editors, *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 325–334. ACM, 2003. ISBN 1-58113-674-9. doi: 10.1145/780542.780592. URL http://doi.acm.org/10.1145/780542.780592.

Avishay Tal. Tight bounds on the fourier spectrum of ac^0. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:174, 2014. URL http://eccc.hpi-web.de/report/2014/174.

Jun Tarui. Probablistic polynomials, AC0 functions, and the polynomial-time hierarchy. *Theor. Comput. Sci.*, 113(1):167–183, 1993. doi: 10.1016/0304-3975(93)90214-E. URL http://dx.doi.org/10.1016/0304-3975(93)90214-E.

Seinosuke Toda and Mitsunori Ogiwara. Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM J. Comput.*, 21(2):316–328, 1992. doi: 10.1137/0221023. URL http://dx.doi.org/10.1137/0221023.

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