Circuit Lower Bounds from Nontrivial Learning Algorithms

Igor C. Oliveira — Charles University in Prague

Rahul Santhanam — University of Oxford

Motivation and Background

"*Nontrivial*" **learning algorithm** for a circuit class **C** → **Lower bounds** against **C** ?

"*Nontrivial*" **learning algorithm** for a circuit class **C** → Improved *algorithmic* **upper bounds** ?

(Non-uniform) Circuit Classes:

$AC^0 \subseteq AC^0[p] \subseteq ACC^0 \subseteq TC^0 \subseteq NC^1 \subseteq \ldots \subseteq$ P/poly

Circuit Size: number of wires.

C(s(n)): C-circuits on n-variables of size ≤ s(n).

(Uniform) Complexity Classes:

P ⊆ NP, P ⊆ BPP; NP ⊆ MA, BPP ⊆ MA; MA ⊆ PSPACE ⊆ EXP; EXP ⊆ NEXP, EXP ⊆ BPEXP; NEXP ⊆ MAEXP, BPEXP ⊆ MAEXP

Theorem. MAEXP $\not\subseteq$ **P/poly** [BFT'98].

Theorem. MA/1 $\not\subseteq$ **SIZE(n^k)** [San'07].

Theorem. NEXP $\not\subseteq$ **ACC^0** [Wil'11].

BPEXP: Exponential time version of BPP.
Believed to collapse to EXP (if **P** = **BPP** then **EXP** = **BPEXP**).

Learning Algorithms:

- **Uniform Distribution**, **Randomized**, **Membership Queries.**

(unknown) **f** ∈ **C**

Oracle Access

Learning Algorithm A^f

Runs in time: t_A**(n, size(f), 1/ε, 1/δ)**

ε: "Accuracy Parameter".
δ: "Confidence Parameter".

Goal: Output with probability > 1 - δ a (general) circuit h such that $\Pr_x[h(x) \neq f(x)] \leq \varepsilon$.

Algorithm A learns f ∈ C with *advantage* γ if w.h.p $\Pr_x[h(x) = f(x)] \geq 1/2 + \gamma$.

- Algorithm *weakly learns* C if $\gamma \geq 1/\mathrm{poly}(n)$.
- It is *nontrivial* if running time of A ≤ $2^n/n^{\omega(1)}$.

Previous Work

Some connections between algorithms and circuit lower bounds:

"Fast SAT implies lower bounds" [KL'80]
If **Circuit-SAT** can be solved *efficiently* then **EXP** $\not\subseteq$ **P/poly**.

"Derandomization implies lower bounds" [KI'03]
If **PIT** ∈ **NSUBEXP** then either
(*i*) **NEXP** $\not\subseteq$ **P/poly**; or
(*ii*) **Permanent** is not computed by poly-size arithmetic circuits.

"Nontrivial SAT implies lower bounds" [Wil'10]
If **Circuit-SAT** for poly-size circuits can be solved in time $2^n/n^{\omega(1)}$ then **NEXP** $\not\subseteq$ **P/poly**.

Lower Bounds from Learning Algorithms:

[FK'06] Learning circuit class C in **poly-time** implies **BPEXP** $\not\subseteq$ **C** (and related results for other learning models).

[HH'11] Stronger lower bounds from Exact Learning.

[KKO'13] Weaker assumptions and stronger conclusions in different learning models. In particular,
Learning C in **subexponential** time implies LB or unlikely collapse.

[Vol'14] **Efficiently** learning C implies **BPTIME($n^{\omega(1)}$)/1** $\not\subseteq$ **C.**

[Vol'15] New results for learning arithmetic circuits.

Fast learning algorithms:

$AC^0 \subseteq AC^0[p] \subseteq ACC^0 \subseteq TC^0 \subseteq NC^1 \subseteq \ldots \subseteq$ P/poly

? ? ?

[CIKK'16] Quasi-poly size **$AC^0[p]$** circuits learnable in quasi-poly time (requires MQs).

[LMN'89] Quasi-poly size **AC^0** circuits learnable in quasi-poly time.

Our Results

[Nontrivial Learning Implies Lower Bounds]

Theorem 1.
Let **C** be a circuit class, γ: ℕ→ (0,1/2] be arbitrary.

Assume C-circuits of size $\mathbf{n^{\omega(1)}}$ can be learned with advantage γ in time O(γ^2 $2^n/n^2$).

Then **BPEXP** $\not\subseteq$ **C[poly]**.

Corollary of Theorem 1.
If **ACC^0** circuits of size $\mathbf{n^{\log \log \log n}}$ can be weakly learned in time $2^n/n^{\omega(1)}$ then **BPEXP** $\not\subseteq$ **ACC^0**.

Lemma 1 **[Speedup Phenomenon in Learning Theory].**

Assume **C[poly(n)]** can be (weakly) learned in time $\mathbf{2^n/n^{\omega(1)}}$.

Let **k** ∈ N and **ε** > 0 be arbitrary constants.

Then C-circuits of size $\mathbf{n^k}$ can be learned to accuracy $\mathbf{n^{-k}}$ in time at most $\mathbf{exp(n^{\varepsilon})}$.

Running Time	ACC^0-SAT	ACC^0-Learning
Nontrivial: $2^n/n^{\omega(1)}$	✓	?
SETH: $2^{(1-\varepsilon)n}$	?	?
ETH: $2^{\varepsilon n}$	?	?
SUBEXP: $2^{n^{\varepsilon}}$	?	?

Speedup Phenomenon in Learning Theory

Proof of Theorem 1

(assuming Lemma 1)

Main Techniques:

- Counting / Concentration Bound
- Diagonalization via majority vote
- Learning via self-correction and downward-reducibility
- Special PSPACE language
- Hardness amplification
- Nisan-Wigderson pseudorandom generator

Proof Sketch:

Let **C** ∈ {AC^0, $AC^0[p]$, ACC^0, TC^0, NC^1, …}.

Assume that **ACC^0** ⊆ **C[poly]** (since otherwise **BPEXP** $\not\subseteq$ **C**).

"C is not too weak."

If **PSPACE** $\not\subseteq$ **C[poly]** we are done (using that **PSPACE** ⊆ **BPEXP**).

⟹ Proceed under the assumption that **PSPACE** ⊆ **C[poly]**.

Let's use the **Speedup Lemma (1)** (*stated above*).

We can learn **C**-circuits of **poly(n)** size in time $\mathbf{exp(n^{o(1)})}$.

(recall that **PSPACE** ⊆ **C[poly]**).

Now employ a technique from [KKO'13], [FK'06], [IW'01]:

Lemma [PSPACE Simulation] (2) (*the proof is sketched later*)
If **PSPACE** ⊆ **C[poly]** and **C[poly]** can be learned in subexponential time then **PSPACE** ⊆ **BPTIME[exp($n^{o(1)}$)]**.

From **PSPACE** ⊆ **BPTIME[exp($n^{o(1)}$)]**,
simple *padding argument* implies: **DSPACE[$n^{\omega(1)}$]** ⊆ **BPEXP.**

Lemma [Diagonalization] (3) (*the proof is sketched later*).
There is L ∈ **DSPACE[$n^{\omega(1)}$]** that is not in **C[poly].**

Since **DSPACE[$n^{\omega(1)}$]** ⊆ **BPEXP,** we get **BPEXP** $\not\subseteq$ **C[poly]**, which completes the proof of **Theorem 1**. □

It remains to prove the following lemmas.

(1) Speedup Lemma (relies on recent work [CIKK'16]).

(2) PSPACE Simulation Lemma (follows [KKO'13]).

(3) Diagonalization Lemma [Folklore].

Lemma [Diagonalization]
There is L ∈ **DSPACE[$n^{\omega(1)}$]** that is not in **C[poly].**

Sketch. Diagonalization via **majority vote**. Define **L** such that on the first ≪ $n^{\log n}$ strings of size n it differs from *every* circuit in **C[poly]**:

- On input 0^n, output the bit that disagrees with at least *half* of the circuits.
- On input $0^{n-1}1$, output the bit that disagrees with at least *half* of the **remaining** circuits.

⋮ ⟹ **L** can be computed in **DSPACE[$n^{\omega(1)}$]**. □

Lemma [PSPACE Simulation]
If **PSPACE** ⊆ **C[poly]** and **C[poly]** can be learned in subexponential time then **PSPACE** ⊆ **BPTIME[exp($n^{o(1)}$)]**.

Sketch. "Learn how to compute a **PSPACE** complete language."

[TV'02] **PSPACE**-complete language **L*** that is both: **downward-self-reducible (dsr)** and **random-self-reducible (rsr)**.

Assume we have learned how to compute **L*** on inputs of length **k** ← dsr — **Goal:** length **k+1** inputs — Run *learning algorithm* and simulate oracle access to **L*** on strings of length **k+1** via **dsr**. — **h: $\{0,1\}^{k+1} \to \{0,1\}$** Self-correct **h** via **rsr**. □

The proof of the Speedup Lemma requires additional ideas and will not be presented here. Check the paper for more details!

Open Problems and Research Directions

1. Investigate the existence of speedups in other learning models.

2. Design a nontrivial learning algorithm for $AC^0[6]$ circuits of size $n^{\log \log \log n}$.

(This would show that **BPEXP** $\not\subseteq$ $AC^0[6]$.)

3. Which classes of functions admit nontrivial learning algorithms?

Example. Depth-2 Threshold Circuits?