Limits on Representing Functions by Linear Combinations of Simple Functions

\[ f : \{0,1\}^n \rightarrow \{0,1\} \ ? \equiv \]

\[ \sum \]

\[ \text{simple simple simple simple simple simple simple} \]

Ryan Williams
MIT
Let $\mathcal{C}$ be a class of “simple” functions (take Boolean inputs, but need not be Boolean-valued).

Which “interesting” functions $f$ can (not) be represented by “short” $\mathbb{R}$-linear combinations of functions from $\mathcal{C}$?

$$f : \{0,1\}^n \rightarrow \{0,1\} \equiv \sum \circ \mathcal{C}$$

Call this a $\sum \circ \mathcal{C}$ circuit.

Note: If $\mathcal{C}$ spans the vector space of all functions $f : \{0,1\}^n \rightarrow \mathbb{R}$, then there is always a $\sum \circ \mathcal{C}$ circuit of $\leq 2^n$ size...
The $\mathbb{R}$-linear Representation Problem

Which “interesting” functions $f$ can (not) be represented by “short” $\mathbb{R}$-linear combinations of functions from $C$?

If $C$ is the class of $2^n$ AND functions on $n$ variables:

\[ \sum \circ AND \equiv 0/1 \text{ polynomials over } \mathbb{R} \]

If $C$ is the class of $2^n$ PARITY functions on $n$ variables:

\[ \sum \circ PARITY \equiv -1/1 \text{ polynomials over } \mathbb{R} \]

(Fourier analysis of Boolean functions)

These are well-understood:

$C$ is a basis for the vector space of functions $f : \{0,1\}^n \to \mathbb{R}$

$\Rightarrow$ the $\mathbb{R}$-linear representation of $f$ is unique,

so the “shortest” is also the “longest”…

More interesting cases: representations are not unique
This Paper: Three Simple Classes

1. Linear Threshold Functions \([LTF]\]
2. Rectified Linear Units \([ReLU]\]
3. \(GF(p)\)-Polynomials of Degree-\(d\) \([POLYd[p]]\)
   (\(p\) prime and \(d \geq 2\))

For all three classes:

- There are \(\gg 2^n\) functions on \(n\) variables, so \(\mathbb{R}\)-linear representations are not unique
  \(2^{\Theta(n^2)}\) LTFs, \(p^{\Theta(n^d)}\) degree-\(d\) polys, \(\infty\) ReLU functions

- \(\mathbb{R}\)-linear Representations have been studied!
  \[\sum \circ LTF = \text{Special Case of Depth-2 Threshold Circuits}\]
  \[\sum \circ ReLU = \text{“Depth-2 Neural Net with ReLU activation”}\]
  \[\sum \circ POLYd[p] = \text{“Higher-Order” Fourier Analysis for } d \geq 2\]
Sums of Linear Threshold Functions

**Def.** $f_n: \{0,1\}^n \rightarrow \{0,1\}$ is an LTF if $\exists w_1, \ldots w_n, t \in \mathbb{R}$ such that
\[
\forall (x_1, \ldots, x_n) \in \{0,1\}^n, \ f(x_1, \ldots, x_n) = 1 \iff \sum_i w_i x_i \geq t
\]

**Depth-Two LTF Circuits (LTF \circ LTF):** Major problem to find “nice” functions without $n^k$-gate $LTF \circ LTF$ circuits, for all $k$

[Hajnal et al.’91] $\exp(n)$ depth-two lower bounds for small $w_i$’s

[Roychowdhury-Orlitsky-Siu’94] What about $\sum \circ LTF$?

**Special case of LTF \circ LTF:**
the linear form for output LTF must always evaluate to 0 or 1

Still, no $n^{1.5}$-gate lower bounds were known for $\sum \circ LTF$!

**We prove:**

**Thm** $\forall k, \exists f_k \in NP$ without $n^k$-size $\sum \circ LTF$

**Thm** $\exists f \in NTIME[n^{\log^*n}]$ without $\text{poly}(n)$-size $\sum \circ LTF$

Note: It is a major open problem to prove $\not\exists f \in NP$ without $n^k$-size (unrestricted) circuits
Sums of ReLUs

**Def.** $f_n: \mathbb{R}^n \rightarrow \mathbb{R}^+$ is a ReLU if $\exists w_1, ... w_n, t \in \mathbb{R}$ such that $\forall (x_1, ..., x_n) \in \mathbb{R}^n$, $f(x_1, ..., x_n) = \max(0, \sum_i w_i x_i + t)$

$\sum \circ \text{ReLU}$ generalizes $\sum \circ \text{LTF}$

$\sum \circ \text{ReLU} = \text{“Depth-Two Neural Nets with ReLU Activations”}$

Very widely studied, thousands of references

Several recent references [see paper] give lower bounds for some “weird” $f: \mathbb{R}^n \rightarrow \mathbb{R}$ which vary sharply / sensitive

No lower bounds known for discrete-domain / Boolean functions (note: “most sensitive” Boolean fn PARITY has $O(n)$-size $\sum \circ \text{LTF}$)

We can generalize the $\sum \circ \text{LTF}$ limits to $\sum \circ \text{ReLU}$:

**Thm** $\forall k, \exists f_k \in \text{NP}$ without $n^k$-size $\sum \circ \text{ReLU}$

**Thm** $\exists f \in \text{NTIME}[n^{\log^* n}]$ without $\text{poly}(n)$-size $\sum \circ \text{ReLU}$

Again: major open problem to prove

$\exists f \in \text{NP}$ without $n^k$-size (unrestricted) circuits
Sums of Low-Degree GF(p)-Polys

$\Sigma \circ POLY_d[p]$: Linear combination of $f: \{0,1\}^n \to \{0,1, \ldots, p - 1\}$ where for every $f$ there is a degree-$d$ polynomial $q(x)$ such that $\forall x \in \{0,1\}^n, f(x) = q(x) \mod p$

Case of $d = 2, p = 2$ is already very interesting!

Compelling Conjecture [“Degree-Two Uncertainty Principle”]:

AND (on $n$ inputs) requires $n^{\omega(1)}$-size $\Sigma \circ POLY2[2]$

Known: AND requires $\Omega(2^n)$-size $\Sigma \circ POLY1[2]$

AND has $O(2^{n/2})$-size $\Sigma \circ POLY2[2]$

No non-trivial lower bounds were known for $\Sigma \circ POLY2[p]$

We prove:

Thm $\forall d, k, \forall p$ prime, $\exists f_k \in NP$ without $n^k$-size $\Sigma \circ POLY_d[p]$

Thm $\exists f \in NTIME[n^{\log^* n}]$ without $poly(n)$-size $\Sigma \circ POLY_d[p]$

for all fixed $d$ and fixed prime $p$
A Key Theorem

Key Theorem: Let \( C \) be a class of functions \( f : \{0, 1\}^n \rightarrow \mathbb{R} \).

Assume: there is an \( \varepsilon > 0 \) and an algorithm \( A \) so that for any given \( f_1, \ldots, f_4 \in C \), \( A \) can compute the “sum-product”

\[
\sum_{\alpha \in \{0,1\}^n} \prod_{i=1}^{4} f_i(\alpha)
\]

in \( 2^{n(1-\varepsilon)} \) time.

Then: \( \forall k, \exists f \in NP \) without \( n^k \)-size \( \sum \circ C \), and \( \exists f \in NTIME[n^{\log^* n}] \) without \( poly(n) \)-size \( \sum \circ C \).

Applies the new Easy Witness Lemma of [Murray-W’18]

We show how to compute sum-products in \( 2^{n(1-\varepsilon)} \) time for LTFs, ReLUs, and low-degree polynomials.
Major Ideas in the Key Theorem

Assume: (1) There is a $2^{n(1-\varepsilon)}$-time sum-product algorithm $A$ for $C$
(2) For some fixed $k$, all $f \in NP$ have $n^k$-size $\sum \circ C$

Goal: Derive a contradiction.

(1) and (2) $\Rightarrow$ Given (unrestricted) circuit $T$ with $n$ inputs and $m$ size
Can guess-and-check $m^k$-size $\sum \circ C$ computing $T$, in $2^{n(1-\varepsilon)}m^{O(1)}$ time

Note: to guess, we need that the coefficients in our linear combinations have “small” bit complexity, WLOG

(1) $\Rightarrow$ Can solve Circuit-UNSAT in nondeterministic $2^{n(1-\varepsilon)}m^{O(1)}$ time
We can even solve $\#\text{Circuit-SAT}$, because we can compute
$$\sum_{a \in \{0,1\}^n} (\sum \circ C (a)) = \sum \sum_a C(a)$$
by solving sum-product for $n^k$ times

[Murray-W'18] $\Rightarrow \forall k, \exists f \in NP$ without $n^k$-size unrestricted circuits

Contradicts (2) when $\sum \circ C$ can be simulated by Boolean circuits!

The proof crucially relies on $\sum \circ C$ computing a circuit exactly
Sum-Product Algorithm for LTF

Uses (old) fact that \#Subset-Sum is solvable in $poly(n) \cdot 2^{n/2}$ time!

Thm [HS’76] \#Subset-Sum on $n$ numbers is in $poly(n) \cdot 2^{n/2}$ time

Proof Given $w_1, \ldots, w_n, t$, we want to know the number of $S \subseteq [n]$ such that $\sum_{i \in S} w_i = t$

1. Enumerate all possible $2^{n/2}$ subsets $S$ of $\{w_1, \ldots, w_{n/2}\}$.
   Make a list $L_1$ of the $2^{n/2}$ subset sums, and SORT all sums in $L_1$

2. Enumerate all possible $2^{n/2}$ subsets $T$ of $\{w_{n/2+1}, \ldots, w_n\}$.
   For each $T$ summing to a value $v$, 
   BINARY SEARCH for a value $v'$ in $L_1$ such that $v + v' = t$

3. To compute the total number of subsets summing to $t$:
   For each sum value $v'$ appearing in $L_1$, 
   store the number $n_{v'}$ of subsets in $L_1$ which have value $v'$.
   Later, if value $v'$ is found in the binary search, 
   add $n_{v'}$ to a running sum.

Takes $poly(n) \cdot 2^{n/2}$ time in total
Sum-Product Algorithm for LTF

Uses (old) fact that \#Subset-Sum is solvable in \( \text{poly}(n) \cdot 2^{n/2} \) time!

**Thm** For any \( f_1, \ldots, f_4 \in LTF \), we can compute

\[
\sum_{a \in \{0,1\}^n} \prod_{i=1}^4 f_i(a) \quad \text{in} \ \text{poly}(n) \cdot 2^{n/2} \ \text{time}.
\]

**Proof** An **Exact LTF** (ELTF) has the form \( g(x) = 1 \iff \sum_i w_i x_i = t \)

\#Subset-Sum in \( \text{poly}(n) \cdot 2^{n/2} \) time \( \Rightarrow \sum_a g(a) \) in \( \text{poly}(n) \cdot 2^{n/2} \) time

[HP, CCC’10]: Every LTF on \( n \) inputs can be written as \( \sum_{\text{poly}(n)} \) ELTF

So we can write

\[
\sum_{a \in \{0,1\}^n} \prod_{i=1}^4 f_i(a) = \sum_{a \in \{0,1\}^n} \prod_{i=1}^4 \left( \sum_{\text{poly}(n)} g_{i,j}(a) \right) \quad \text{for ELTFs} \ g_{i,j}
\]

Simple algebra:

\[
= \sum_{a \in \{0,1\}^n} \sum_{\text{poly}(n)} \prod_{i=1}^4 g_{i,j}(a) = \sum_{\text{poly}(n)} \sum_{a \in \{0,1\}^n} \prod_{i=1}^4 g_{i,j}(a)
\]

Each \( \prod_{i=1}^4 g_{i,j}(x) = h(x) \) for some ELTF \( h \)

Can compute in \( \text{poly}(n) \cdot 2^{n/2} \) time!
Sum-Product Algorithm for Polys

Uses (recent) fact that counting Boolean roots of $n$-variable degree-$d$ \(GF(p)\)-polynomials is solvable in \(2^{n(1-1/O(pd))}\) time \([\text{LPTWY’17}]\)

Algorithm uses a derandomized version of Razborov-Smolensky’s probabilistic representation of \(AC0[p]\) by low-degree \(GF(p)\) polynomials, along with a divide-and-conquer approach for fast evaluation

**Thm** For any \(f_1, \ldots, f_4 \in POLYd[p]\), we can compute

\[
\sum_{a \in \{0,1\}^n} \prod_{i=1}^4 f_i(a) \quad \text{in} \quad \text{poly}(n) \cdot 2^{n\left(1 - \frac{1}{O(pd)}\right)} \quad \text{time.}
\]

**Proof Idea**
Reduce the sum-product problem on four degree-$d$ polys to counting Boolean roots of \(O(1)\) degree-\(O(pd)\) polys
Open Problems

We proved fixed-polynomial lower bounds for functions in $NP$

For each $k$, there is an $f \in NTIME[n^{g(k)}]$ without $n^k$-size $\sum \circ LTF$

Can we prove $SAT$ requires $n^k$-size $\sum \circ LTF$, for all $k$?

Current algorithms-to-lower bounds connections don’t seem to point a way

Constant Degree Hypothesis [Barrington-Straubing-Therien’90]:
For each fixed $d$, $AND$ does not have $MOD_p \circ MOD_q \circ AND_d$ circuits of $2^{o(n)}$ size

Open even when $d = 2$. Could our approaches say anything?

New ways of deriving strong lower bounds from “old” approaches

The (old) algorithm for #Subset-Sum splits the instance of $n$ items into two parts of size $n/2$ each, lists all $2^{n/2}$ subsums separately, sorts the two lists and binary searches for the overall subset sum.

Reviewer asked: Can “split-and-list” be viewed as a lower bound method? The alg. for polys uses Razborov-Smolensky, which is used in lower bounds!
Thank you!