Electronic Colloquium on Computational Complexity, Report No. 12 (2023)

Linear threshold functions in decision lists, decision trees, and depth-2 circuits

Yogesh Dahiya
The Institute of Mathematical Sciences, Chennai, India
Homi Bhabha National Institute, Training School Complex, Anushaktinagar, Mumbai, India

Vignesh K
Indian Institute of Technology Hyderabad, Kandi, Sangareddy, Hyderabad, 502285 Telangana, India

Meena Mahajan
The Institute of Mathematical Sciences, Chennai, India
Homi Bhabha National Institute, Training School Complex, Anushaktinagar, Mumbai, India

Karteek Sreenivasaiah
Indian Institute of Technology Hyderabad, Kandi, Sangareddy, Hyderabad, 502285 Telangana, India

Abstract

We show that polynomial-size constant-rank linear decision trees (LDTs) can be converted to polynomial-size depth-2 threshold circuits LTF ∘ LTF. An intermediate construct is polynomial-size decision lists that query a conjunction of a constant number of linear threshold functions (LTFs); we show that these are equivalent to polynomial-size exact linear decision lists (ELDLs) i.e. decision lists querying exact threshold functions (ELTFs).

2012 ACM Subject Classification Theory of computation → Oracles and decision trees; Theory of computation → Circuit complexity

Keywords and phrases linear threshold functions; decision lists; decision trees; depth-2 circuits; tree rank.

1 Introduction

Understanding the power of linear threshold functions LTFs as a primitive operation is a significant question in complexity theory. At the frontier of circuit lower bounds is the class TC^0: polynomial-size constant-depth circuits using LTF gates. Currently we do not know explicit lower bounds even for polynomial-size depth-2 threshold circuits. We denote this class LTF ∘ LTF. Restricting this circuit class further to polynomial-weight LTFs (MAJ gates) at either one or both of the two levels gives the classes MAJ ∘ LTF, LTF ∘ MAJ, and MAJ ∘ MAJ. The exact relationship among these classes, and non-trivial lower bounds, are known — MAJ ∘ MAJ equals MAJ ∘ LTF and is strictly weaker than LTF ∘ MAJ [5], which is strictly weaker than LTF ∘ LTF [3]. Another way to use LTFs as a primitive is as a query, in linear decision lists LDLs and linear decision trees LDTs, see [6, 9]. Here, the usual complexity measure is query complexity, but one can also consider size, which is a measure of the space required to store the function in this representation. It is known that the class LDL of functions with polynomial-size LDLs is contained in LTF ∘ LTF [9], but the same is not known for the class LDT of functions computable by polynomial-size LDTs. For the classes LTF ∘ LTF and LDT, a common upper bound is MAJ ∘ MAJ ∘ MAJ. However, unlike LTF ∘ LTF, we do have non-trivial lower bounds against LDTs, obtained in [9] by investigating the rank of decision trees. This makes LDT rank a very interesting measure from the lower bounds point of view.

In general, small rank in decision trees of various types points to a certain simplicity of the function, making it easy to learn in certain settings, see for instance [4, 8]. In the context of LTF primitives, the use of small rank has been particularly fruitful. It was shown in [9] that

ISSN 1433-8092

LDLs (which are just rank-1 LDTs) and constant-rank LDTs require large depth to compute the Inner Product function. In fact, a rank-depth tradeoff was established; and this implies (as observed in [10]) that LDTs of any rank computing the Inner Product function must be of exponential size[1]. Also in [9], a simple construction showed that polynomial-size LDLs can be transformed to polynomial-size depth-2 threshold circuits, and hence $\mathsf{LDL} \subseteq \mathsf{LTF} \circ \mathsf{LTF}$. In [10], this was pushed further a bit: polynomial-size constant-rank LDTs with MAJ queries were also transformed to polynomial-size depth-2 threshold circuits.

Our main result takes this one step further by removing the polynomial-weight restriction. We show the following.

▶ **Theorem 1.** *If a function $f : \{0,1\}^n \longrightarrow \{0,1\}$ is computed by an LDT of rank r and size s, then f can be computed by depth-2 threshold circuits of size $O(s \cdot n^{3r} \log^r n)$.*

In particular, at polynomial size, $O(1)$-rank-LDT is contained in $\mathsf{LTF} \circ \mathsf{LTF}$.

In proving the above, an intermediate (between the small-rank LDTs and the depth-2 circuits) computation model used is decision lists where each query is a conjunction of LTFs. The arity of the conjunctions is crucial; in our construction, this is the rank of the LDT we start out with. Our method also relates such lists, with queries that are constant-arity conjunctions of LTFs, to a related model studied in [7] of linear decision lists with equality queries ELDLs; these are decision lists where each query is an exact linear threshold function ELTF. Namely, we show the following:

▶ **Theorem 2.** *The following are equivalent:*

1. *ELDLs of polynomial length.*
2. *Decision lists of polynomial length, querying functions in $\mathsf{AND}_r \circ \mathsf{LTF}$ for some fixed $r \geq 2$.*

That r is at least 2 is crucial; at $r = 1$ the decision lists are LDLs, which, by this theorem, are contained in the class ELDL of polynomial length ELDLs, but are known to be strictly weaker (the $\mathsf{OR}_n \circ \mathrm{EQ}_n$ function requires size $2^{\Omega(n)}$ in LDLs and super-polynomial size even in LDTs, but can easily be computed by an ELDL of length n).

Note that ELDL is another frontier class within $\mathsf{LTF} \circ \mathsf{LTF}$ for which no lower bounds are currently known. By Theorem 2, obtaining lower bounds for ELDLs is no easier than, and in fact equivalent to, showing lower bounds for decision lists with $\mathsf{AND}_2 \circ \mathsf{LTF}$ queries.

2 Some definitions, notation, and known results

We include here some basic definitions and notation; for more details, we follow standard notation as, for instance, in [2].

A linear threshold function, denoted LTF, is a Boolean function $f : \{0,1\}^n \to \{0,1\}$ expressible as $f(x) = 1 \Leftrightarrow \sum_i w_i x_i \geq w_0$ for some $w_0, w_1, \ldots, w_n \in \mathbb{R}$. If f is an LTF, then so is $\neg f$. An exact linear threshold function, denoted ELTF, is a Boolean function $f : \{0,1\}^n \to \{0,1\}$ expressible as $f(x) = 1 \Leftrightarrow \sum_i w_i x_i = w_0$ for some $w_0, w_1, \ldots, w_n \in \mathbb{R}$. From a geometric point of view, LTFs are halfspaces, and ELTFs are hyperplanes. The class of ELTF functions is not closed under negations[2]. For every ELTF f, there are LTFs g, h

[1] It is not hard to see that these lower bounds, presented in [9] for the Inner Product, hold for any function with no large monochromatic squares; for any such function, the depth-rank tradeoffs and size lower bounds hold. In particular, the functions $\mathsf{MAJ} \circ \mathsf{XOR}$, $\mathsf{OR} \circ \mathrm{EQ}$, $\mathrm{SINK} \circ \mathsf{XOR}$, are all hard for LDT.

[2] It is easy to show that the function that is 1 on precisely the unit vectors is an ELTF, but the negation is not.

such that $h \implies g$ and $f = g \wedge \neg h = g - h$. We denote the class of functions that can be written as an LTF (ELTF) as LTF (ELTF respectively).

For any function classes C_1, C_2, the function class $\mathcal{C}_1 \circ \mathcal{C}_2$ is defined as :
$\mathcal{C}_1 \circ \mathcal{C}_2 = \{f(g_1, g_2, \ldots, g_m) \mid f \in \mathcal{C}_1; g_1, \ldots g_m \in \mathcal{C}_2\}$. That is, these are functions computable by depth-2 circuits with a $\mathcal{C}_1$ gate on top and $\mathcal{C}_2$ gates below it. The class $\mathcal{C}_1 \circ \mathcal{C}_2 \circ \mathcal{C}_3$ is analagous to the above — depth-3 circuits with a $\mathcal{C}_1$ gate in top, $\mathcal{C}_2$ gates at the middle layer, and $\mathcal{C}_3$ gates at the bottom layer.

Of special interest in this note are the classes $\mathsf{LTF}_s \circ \mathsf{LTF}$, $\mathsf{AND}_r \circ \mathsf{LTF}$, $\mathsf{OR}_k \circ \mathsf{ELTF}$, and $\mathsf{LTF}_m \circ \mathsf{AND}_r \circ \mathsf{LTF}$, where the subscript denotes the arity/in-degree of the functions/gates. (We drop the subscript where the arity is not important.)

For function class $\mathcal{C}$, a $\mathcal{C}$-decision tree is a decision tree where each query computes some function from $\mathcal{C}$ on the inputs. The size of such a decision tree is the number of query nodes (or the number of leaves, which is just one more), the depth of the tree is the maximum number of query nodes in any root-to-leaf path, and the rank of the tree is the largest d such that a complete binary tree of depth d can be embedded in it. (Formally, the rank of a leaf is 0, and the rank of an internal node is the maximum rank of its children if they have unequal rank, and is one more than the rank of its children if they have the same rank.) Of special interest to us are LTF-decision trees, also referred to as Linear Decision Trees (LDTs).

A $\mathcal{C}$-decision tree of rank one is a $\mathcal{C}$-decision list, and the depth of the tree in this case is called the length of the list. Thus a $\mathcal{C}$-decision list of length ℓ has the form
If f_1 then b_1; elseif f_2 then b_2; $\ldots$; elseif f_ℓ then b_ℓ; else $\neg b_\ell$,
where each f_i belongs to the class $\mathcal{C}$, and each $b_i \in \{0, 1\}$. For brevity, we shall often write such a decision list as simply a tuple $((f_1, b_1), \ldots, (f_\ell, b_\ell))$. Of special interest to us are (all of polynomial size) LTF-decision lists denoted LDL, ELTF-decision lists denoted ELDL, and $(\mathsf{AND}_r \circ \mathsf{LTF})$-decision lists for some fixed constant r.

(Throughout this article, we shall use the standard convention, see for example [7], of denoting the complexity class of functions computable *efficiently* by some computation model using bold-face font. For example, the class of functions computable by polynomial size LDLs is denoted LDL.)

We collect some known facts that will be used in proving our results.

▶ **Proposition 3.** **1.** *Any $\mathcal{C}$-decision list of length s can be converted to a depth-2 circuit of the form* $\mathsf{LTF}_s \circ \mathcal{C}$.

2. *An LDT of size s and rank r can be converted to an (*$\mathsf{AND}_r \circ \mathsf{LTF}$*)-decision list of length s. ([1]; see also Fact 1 in [11].)*

3. *Any linear threshold function on n variables can be computed as a disjoint OR of $O(n^3 \log n)$ exact threshold functions. Thus* $\mathsf{LTF} \subseteq \mathsf{OR}_{O(n^3 \log n)} \circ \mathsf{ELTF}$. *([7](Theorem 7)].)*

4. $\mathsf{AND} \circ \mathsf{ELTF} = \mathsf{ELTF}$*; any conjunction of* ELTF*s is also an* ELTF *([7](Proposition 6 item 2)].)*

5. *An LDT of size s and depth d can be converted to an (*$\mathsf{OR}_s \circ \mathsf{AND}_d \circ \mathsf{MAJ}$*) circuit. In particular,* $\mathsf{LDT} \subseteq \mathsf{MAJ} \circ \mathsf{MAJ} \circ \mathsf{MAJ}$.

Proof. Items 2,3,4 are shown in the references cited.

Item 1 is the folklore binary coding technique; $((f_1, b_1), \ldots, (f_\ell, b_\ell))$ evaluates to 1 on input x if and only if $\sum_{i=1}^{\ell+1} (-1)^{1-b_i} 2^{\ell+1-i} [f_i(x) = 1?] \geq 0$, where $f_{\ell+1}$ is the constant 1 function and $b_{\ell+1} = 1 - b_\ell$.

To see item 5, observe that we can check if an input reaches any specific leaf of an LDT by a conjunction of the queried LTFs, or their negations, that appear on the path from root to that leaf. Computing an OR over all leaves labelled 1 gives an $\mathsf{OR}_s \circ \mathsf{AND}_d \circ \mathsf{LTF}$ circuit. We know that $\mathsf{OR} \circ \mathsf{AND} \circ \mathsf{LTF} \subseteq \mathsf{MAJ} \circ \mathsf{MAJ} \circ \mathsf{LTF} \subseteq \mathsf{MAJ} \circ \mathsf{MAJ} \circ \mathsf{MAJ}$ by [5]. ◀

3 Proving Theorem 1

At a high level, the conversion of a small-rank LDT to a depth-2 threshold circuit proceeds in stages. We first convert the tree to a decision list using Proposition 3(2), then the list to a depth-3 circuit using Proposition 3(1). The next crucial and new step, which we describe below, is to replace each sub-circuit feeding into the top gate by a disjoint OR of ELTFs. This generalisation of Proposition 3(3), that we show below, allows us to obtain an equivalent $\mathsf{LTF} \circ \mathsf{ELTF}$ circuit. Expressing the ELTFs as the difference of LTFs completes the construction.

We now describe the details of the conversion.

Proposition 3(3) tells us that a function described by a halfspace can also be seen as a union of disjoint hyperplanes. The lemma below extends this to the intersection of half-spaces $\mathsf{AND}_r \circ \mathsf{LTF}$:

▶ **Lemma 4.** *Let $f : \{0,1\}^n \to \{0,1\}$ be computed by a circuit of the form $\mathsf{AND}_r \circ \mathsf{LTF}_n$. Then there exists a set $\mathcal{A}_f = \{f^1, f^2, \ldots, f^\ell\}$ where $\forall i, f^i \in \mathsf{ELTF}$ and $\ell \in O(n^{3r} \log^r n)$ such that $\forall x \in \{0,1\}^n$:*

- *If $f(x) = 0$ then $\forall i \in [\ell]$ we have $f^i(x) = 0$.*
- *If $f(x) = 1$ then $\exists i \in [\ell]$ such that $f^i(x) = 1$ and $\forall j \neq i$, $f^j(x) = 0$.*

Proof. We first describe the proof for $r = 2$. Let $f = g \wedge h$ where $g, h \in \mathsf{LTF}$. Using Proposition 3(3) on g, we obtain a set $S = \{g_1, g_2, \ldots, g_s\}$ of ELTF functions with $|S| \in O(n^3 \log n)$ such that g is the disjoint-OR of all the g_i. Similarly, we obtain a set T of ELTF functions with $|T| \in O(n^3 \log n)$ for the function h. Define the following set:

$$\mathcal{A} = \{g_i \wedge h_j \mid g_i \in S, h_j \in T\}$$

Note that since g (h) is a disjoint-OR of the functions in S (T respectively), if $(g \wedge h)(x) = 1$, then exactly one of the functions in S and exactly one of the functions in T will evaluate to 1 on x. Hence if $f(x) = 1$, then exactly one of the functions in $\mathcal{A}$ evaluates to 1 on x. On the other hand if $f(x) = 0$, then every function in $\mathcal{A}$ evaluates to 0. Thus f is the disjoint-OR of the functions in $\mathcal{A}$. As described above, $\mathcal{A}$ consists of $\mathsf{AND}_2 \circ \mathsf{ELTF}$ functions. By Proposition 3(4), each such function is in fact an ELTF function, and hence the set $\mathcal{A}$ has all the properties stated in the lemma. Note that $|\mathcal{A}| \in O(n^6 \log^2 n)$.

In general, for $r > 2$, we take $\mathcal{A}$ to be the cartesian product with operator $\wedge$ of all the r sets. This will give $|\mathcal{A}| \in O(n^{3r} \log^r n)$. ◀

Using Lemma 4, we can now transform depth-3 circuits

▶ **Theorem 5.** *For all r, s, we have*

$$\mathsf{LTF}_s \circ \mathsf{AND}_r \circ \mathsf{LTF}_n \subseteq \mathsf{LTF}_m \circ \mathsf{LTF}_n$$

where $m \in O(s \cdot n^{3r} \log^r n)$.

Proof. Let C be an $\mathsf{LTF}_s \circ \mathsf{AND}_r \circ \mathsf{LTF}_n$ circuit computing a function f. Let the top LTF gate in C be $\sum_{i=1}^{s} w_i g_i \geq t$ where the g_i functions are computed by $\mathsf{AND}_r \circ \mathsf{LTF}_n$ circuits.

For each g_i, we use Lemma 4 to obtain a set $\mathcal{A}_{g_i} = \{g_i^1, g_i^2, \ldots, g_i^{\ell_i}\}$ of $\ell_i \in O(n^{3r} \log^r n)$ hyperplanes.

In the top gate of C, we replace each g_i with $\sum_{j=1}^{\ell_i} g_i^j$ to obtain a $\mathsf{LTF}_{s'} \circ \mathsf{ELTF}$ circuit C', where $s' = \sum \ell_i = O(s \cdot n^{3r} \log^r n)$. Circuit C' computes the same Boolean function as C since Lemma 4 guarantees that $\forall x, g_i(x) = \sum_j g_i^j(x)$. Hence $f \in \mathsf{LTF}_{s'} \circ \mathsf{ELTF}_n$ via C'.

The top gate of C' is $\sum_{j=1}^{s'} w'_j f_j \geq t$ where the f_j are the ELTFs described above. Replace each f_j in this expression with $f_{j,1} - f_{j,2}$ where $f_{j,1}$, $f_{j,2}$ are LTFs whose difference is f_j, to get circuit C''. Circuit C'' also computes f, and has the form $\mathsf{LTF}_m \circ \mathsf{LTF}_n$, where $m = 2s' \in O(s \cdot n^{3r} \log^r n)$. ◀

Now we can complete the proof of our main result.

Proof. (of Theorem 1.) Let f be an n-variate Boolean function computed by an LDT T with rank r, size s. By Proposition 3(2), f is computed by a decision tree of length s with queries in $\mathsf{AND}_r \circ \mathsf{LTF}_n$. By Proposition 3(1), f is computed by a circuit of the form $\mathsf{LTF}_s \circ \mathsf{AND}_r \circ \mathsf{LTF}_n$. By Theorem 5, f is computed by an $\mathsf{LTF}_m \circ \mathsf{LTF}_n$ circuit, where $m \in O(s \cdot n^{3r} \log^r n)$. ◀

4 Proving Theorem 2

We first observe an easy manipulation of decision lists of a particular form.

▶ **Lemma 6.** *A Boolean function f computable by a $(\mathsf{OR}_\ell \circ \mathcal{C})$-decision list L of length s can also be computed by a $\mathcal{C}$-decision list of length $s\ell$.*

Proof. Let f and L satisfy the premise. Then L is an $(\mathsf{OR}_\ell \circ \mathcal{C})$-decision list that has the form: $((f_1, b_1), (f_2, b_2), \ldots, (f_s, b_s))$. For all $i \in [s]$, let $f_i = g_i^1 \vee \cdots \vee g_i^{\ell_i}$ for some $\ell_i \leq \ell$, where each g_i^j is in $\mathcal{C}$. Replacing each (f_i, b_i) in L with the sub-list $((g_i^1, b_i), (g_i^2, b_i), \ldots, (g_i^{\ell_i}, b_i))$ gives a $\mathcal{C}$-decision list computing f. ◀

Now the proof of Theorem 2 is straightforward.

Proof. (of Theorem 2) Since an ELTF is expressible as the conjunction of two LTFs, ELDLs of length s can be expressed as decision lists of length s with queries in $\mathsf{AND}_r \circ \mathsf{LTF}$ for any $r \geq 2$.

It remains to prove that for any fixed r, decision lists of length s querying $\mathsf{AND}_r \circ \mathsf{LTF}$ functions can be converted to ELDLs of length polynomial in s.

Let L be an $\mathsf{AND}_r \circ \mathsf{LTF}$-decision list of length s. Using Lemma 4, we can reframe each query as an $\mathsf{OR}_\ell \circ \mathsf{ELTF}$ function, where $\ell \in O(n^{3r} \log^r n)$. Using Lemma 6, we can strip away the outer OR and obtain an ELDL with length $O(s\ell) = O(sn^{3r} \log^r n)$. ◀

Using Proposition 3(2) and Theorem 2, we can convert polynomial-size $O(1)$-rank LDTs to polynomial-size ELDLs. Using Theorem 2, Proposition 3(1), and Theorem 5, we can convert polynomial-size ELDLs to polynomial-size $\mathsf{LTF} \circ \mathsf{LTF}$ circuits. Thus we obtain the following corollary.

▶ **Corollary 7.** *The class of functions computed by polynomial-size ELDLs contains all functions computed by $O(1)$-rank polynomial-size LDTs and is contained in the class of functions computable by polynomial-size $\mathsf{LTF} \circ \mathsf{LTF}$ circuits. i.e., $O(1)$-rank-$\mathsf{LDT} \subseteq \mathsf{ELDL} \subseteq \mathsf{LTF} \circ \mathsf{LTF}$.*

References

1 Avrim Blum. Rank-r decision trees are a subclass of r-decision lists. *Information Processing Letters*, 42(4):183–185, 1992. doi:10.1016/0020-0190(92)90237-P.

2 Arkadev Chattopadhyay, Meena Mahajan, Nikhil S. Mande, and Nitin Saurabh. Lower bounds for linear decision lists. *Chic. J. Theor. Comput. Sci.*, 2020, 2020. URL: http://cjtcs.cs.uchicago.edu/articles/2020/1/contents.html.

3 Arkadev Chattopadhyay and Nikhil S. Mande. A short list of equalities induces large sign-rank. *SIAM J. Comput.*, 51(3):820–848, 2022. doi:10.1137/19m1271348.

4 Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Information and Computation*, 82(3):231 – 246, 1989. URL: http://www.sciencedirect.com/science/article/pii/0890540189900011, doi:https://doi.org/10.1016/0890-5401(89)90001-1.

5 Mikael Goldmann, Johan Håstad, and Alexander A. Razborov. Majority gates vs. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992.

6 Hans Dietmar Gröger and György Turán. On linear decision trees computing boolean functions. In *Automata, Languages and Programming, 18th International Colloquium, ICALP91, Madrid, Spain, July 8-12, 1991, Proceedings*, pages 707–718, 1991.

7 Kristoffer Arnsfelt Hansen and Vladimir V. Podolskii. Exact threshold circuits. In *Proceedings of the 25th Annual IEEE Conference on Computational Complexity, CCC 2010, Cambridge, Massachusetts, June 9-12, 2010*, pages 270–279, 2010.

8 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.

9 György Turán and Farrokh Vatan. Linear decision lists and partitioning algorithms for the construction of neural networks. In *Foundations of Computational Mathematics*, pages 414–423, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

10 Kei Uchizawa and Eiji Takimoto. Lower bounds for linear decision trees with bounded weights. In *SOFSEM 2015: Theory and Practice of Computer Science*, pages 412–422, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

11 Kei Uchizawa and Eiji Takimoto. Lower bounds for linear decision trees with bounded weights. In *SOFSEM 2015: 41st International Conference on Current Trends in Theory and Practice of Computer Science, Pec pod Sněžkou, Czech Republic, January 24-29, 2015. Proceedings*, volume 8939 of *Lecture Notes in Computer Science*, pages 412–422. Springer, 2015. doi:10.1007/978-3-662-46078-8_34.