Depth-2 threshold circuits

Provable limitations

Meena Mahajan

Circuits with linear threshold functions as primitives are a natural model for computation in the brain. Small threshold circuits of depth two cannot compute most functions, but how do we prove such a statement? And how do we lay our hands on explicit functions that they cannot compute? This article gives an overview of the landscape.

Meena Mahajan works in the theoretical computer science group at the Institute of Mathematical Sciences (IMSc, HBNI) in Chennai. Her research is in the area of computational complexity. She is particularly interested in questions concerning circuit complexity - how much circuitry is *needed* to compute a function, and in proof systems - how hard is it to *prove* that a false statement is indeed false.

Introduction

In a world increasingly driven by technology, better (faster, or more efficient, or better in some other way) algorithms are often the key to opening up new possibilities. But when does one stop trying to devise a better algorithm for a task? When the algorithm at hand is already "optimal" and cannot be improved. A central goal of computational complexity theory is to provide these "stopping" criteria – to establish lower bounds on the amount of resource (time, space, circuit size, ...) required to solve problems. Thus we seek a better understanding of the capabilities, and the limitations, of various computation models. This article focuses on one specific model: circuits with linear threshold functions as the primitive operation.

1. The Computation Model

Linear threshold functions

The primitive operations we use are linear threshold functions, LTFs. Such a function evaluates to the value 1 if a certain linear combination of the input values exceeds a certain threshold, and otherwise evaluates to 0. (All input and output values are Boolean

Keywords
computation, circuits, threshold functions, complexity

– either 0 or 1.)

Here's an example. The GreaterThan function on $2n$ arguments, $\mathsf{GT}_n : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, is defined as follows:

$$\mathsf{GT}_n(x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_n) = \begin{cases} 1 & \text{if } X > Y \\ 0 & \text{otherwise} \end{cases}$$

where X and Y are the numbers whose representation in binary are the bit strings $\tilde{x} = x_1 x_2 \cdots x_n$ and $\tilde{y} = y_1 y_2 \cdots y_n$. This is an LTF, as can be seen from the identity

$$\mathsf{GT}_n(\tilde{x}, \tilde{y}) = 1 \iff \left(\sum_{i=1}^{n} 2^{n-i} x_i\right) - \left(\sum_{i=1}^{n} 2^{n-i} y_i\right) \geq 1.$$

Notice that the GT function is defined for each natural number n. This is typical in complexity theory; we are interested in quantifying how the amount of resource needed to solve a problem grows as the input size increases.

In general, a Boolean function $f : \{0,1\}^n \to \{0,1\}$ is an LTF if there exist real numbers $w_0, w_1, \cdots, w_n$ such that for every $x \in \{0,1\}^n$, $f(x) = 1$ if and only if $\sum_i w_i x_i \geq w_0$. The weights w_i are said to realise the function f. If we consider the discrete set of points $\{0,1\}^n$ in $\mathbb{R}^n$, then the hyperplane $\sum_i w_i x_i = w_0$ separates the 0s and 1s of f.

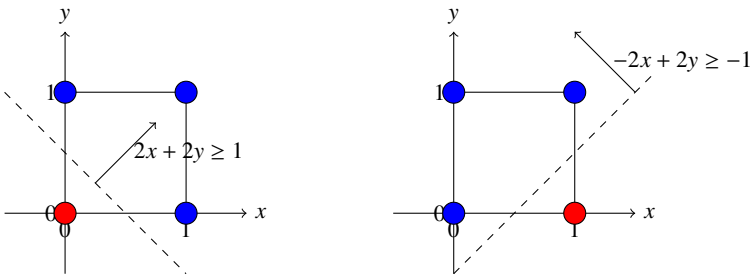
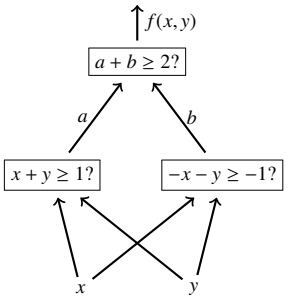


Figure 1. Some 2-variable LTFs. The functions are 1 at the blue points and 0 at the red points. The separating 2-dimensional hyperplanes are lines.

Threshold circuits

The basic model we consider is that of circuits. These are directed acyclic graphs, or networks, with Boolean (zero or one) values travelling along every edge or wire. At each node, the values along the incoming edges are combined according to the primitive operation at that node, and the resulting value travels along

all outgoing edges. In threshold circuits, each node computes an LTF, though the specific functions at different nodes could be different. A simple example is shown alongside. You can verify that $f(x, y) = 1$ exactly when $x \neq y$.



The complexity measure

The size of a threshold circuit is the number of nodes (and edges) in it. The depth is the longest directed path in the circuit; equivalently, the maximum number of nodes an input value has to "flow through" before reaching the output wire. In the above example, the size is three (ten if we also count edges) and the depth is two.

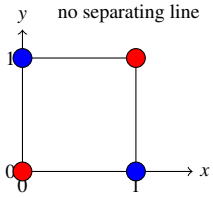
Threshold circuits are essentially the same as neural networks. A single LTF node i.e. a depth-1 circuit, is a neuron; a depth-2 threshold circuit is a shallow neural network (a perceptron); a depth-d threshold circuit is a neural network with $d - 1$ hidden layers.

Where we stand today

Every function can be computed by threshold circuits. In fact, every function can be computed by a threshold circuit of depth 2. (Can you figure out why?) But the size of the circuit may be very large; it may be exponential in n. For efficient computation, we'd like circuits of polynomial size. As we will see below, we do know, provably, that there are functions that cannot be computed by depth-2 threshold circuits of polynomial size. An example? We don't have one! We can prove that hard functions exist, but we do not yet know how to lay our hands on one explicitly described function that is hard! So depth-2 threshold circuits are at the current frontier of lower bounds. Let us see what it took even to reach this frontier.

2. A single gate

Recall the "separating hyperplane" description of LTFs. For 2-bit functions, we want a separating line. The 1s of the 2-bit function



$x \neq y$ cannot be separated from the 0s by any line in the plane $\mathbb{R}^2$. More generally, the PARITY function, defined as

$$\mathsf{PARITY}_n(x_1, x_2, \cdots, x_n) = 1 \iff x_1 + x_2 + \cdots x_n \text{ is an odd number}$$

is not an LTF. (The $x \neq y$ function is just PARITY_2.) So not all functions are LTFs.

Actually, the situation is much more extreme; **most** functions are not LTFs. To see this, we can use a useful fact about LTFs: every LTF bits can be realized by a weight vector of integers, and furthermore, for the n-bit function, the integers are no more than $2^{cn \log n}$ in magnitude, for some absolute constant c.

Why? Let f be an LTF realised by a real vector $\tilde{w}$. The system of 2^n linear inequalities in unknowns W_i, $i \in [n]$,

$$\begin{aligned} \textstyle\sum_i W_i x_i &\geq 1 \quad \text{for } \tilde{x} \in f^{-1}(1), \\ \textstyle\sum_i W_i x_i &\leq 0 \quad \text{for } \tilde{x} \in f^{-1}(0) \end{aligned}$$

certainly has a solution, obtained by suitably shifting and scaling $\tilde{w}$. But this solution is not unique. If we use linear programming techniques such as the Simplex algorithm, we will obtain a rational solution where the entries are not too large in magnitude, because all coefficients in the system are 0 or 1. (Recall, we consider only Boolean functions.) Clearing the denominators gives the desired integer weight vector realising f.

A counting argument: If set A is larger than set B, then there must be objects in A but not in B.

How does this fact help us? Well, now we know that the number of n-bit LTFs is bounded by the number of weight vectors of this magnitude. Each of the $n+1$ weights w_i can be any integer in the range $-2^{cn \log n}, \cdots, -1, 0, 1, \cdots, 2^{cn \log n}$, so there are at most $\left(2^{cn \log n + 1} + 1\right)^{n+1}$ such vectors. But the number of Boolean functions is much much more; 2^{2^n}; so most functions are not LTFs.

A natural question that would arise at this point is, why do we even allow integer weights as large as $2^{cn \log n}$? Shouldn't we also restrict weights to have polynomial magnitude? We certainly could, and then we would get a class of functions that we could denote $\widehat{\mathsf{LTF}}$. Small weights means we can duplicate wires and eliminate weights altogether, and use the most natural

threshold function, namely the Majority function Maj. For example, the LTF that checks $3x - 2y + z \geq 2$? can be expressed as $\mathsf{Maj}(x, x, x, \bar{y}, \bar{y}, z, 0)$. (Note, $\bar{y}$ denotes the negation of y; that is, $1 - y$. We will consider negating a Boolean value as a free operation.) If the weights are large, the duplications are large and we may end up with more than polynomially-many wires. That's a good reason to look for small-weight realisations. And that's why we will simply say MAJ when referring to the class $\widehat{\mathsf{LTF}}$.

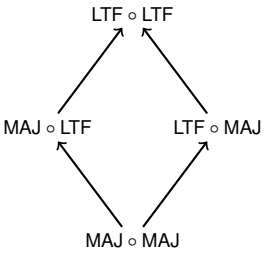
It turns out not all LTFs are in the class MAJ. A concrete example is the function GT we saw earlier. Why isn't it a small-weight LTF? (If you haven't seen this earlier, PAUSE reading this now! Spend a few minutes thinking of how you would prove this. If you succeed, great; in any case, then read on!) For any $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, we can set up a square matrix M_f of order 2^n with 0-1 entries as follows: there is a row for each assignment to the variables $\tilde{x}$, and a column for each assignment to the variables $\tilde{y}$. The entry $M_f[\tilde{a}, \tilde{b}]$ is just $f(\tilde{a}, \tilde{b})$. Analysing this matrix is crucial in many lower bound arguments. For a function in MAJ, say realized as $\langle \tilde{w}, \tilde{x} \rangle + \langle \tilde{w}', \tilde{y} \rangle \geq t$ with small integers, the number of distinct values taken by $\langle \tilde{w}, \tilde{x} \rangle$ is small, at most a polynomial in n (even though there are 2^n settings to $\tilde{x}$). So there are distinct $\tilde{x}, \tilde{x}'$ with the same weighted sum. The corresponding rows in M_f must then be the same. But for the GT function, all rows of the matrix are distinct; hence it is not an MAJ.

M_f is called the communication matrix of f. Imagine that Akbar and Birbal are trying to compute f, but Akbar only knows x, and Birbal only knows y, and communication is costly. They study M_f to figure out the minimum communication that can let them compute f. See [KN97] for an introduction to communication complexity.

3. Depth-2 circuits

Let's move on to depth-2 circuits. Recall, we counted Boolean functions and showed that this number is much more than the number of LTFs. More or less the same argument shows that this number is also larger than the number of depth-2 threshold circuits of polynomial size. Not just larger, but much larger. In fact, if we pull out an n-bit function uniformly at random from all n-bit functions, then with overwhelmingly high probability it has no polynomial-size depth-2 threshold circuit. So finding one such function is like searching for not the proverbial needle in the

haystack but for the hay itself! Why would this be hard? Well, it is hard because we don't yet have enough tools to tell apart the needle from the hay. We need to develop mathematical techniques that will let us conclude that a function is not easy in this model.



To begin with, let us carefully describe the types of depth-2 threshold circuits. Let $A \circ B$ denote functions computable by polynomial-size depth-2 circuits where the top node computes a function from the class A, and nodes at the bottom layer compute functions from the class B. We now have four types of circuits: $\mathsf{LTF} \circ \mathsf{LTF}$, $\mathsf{LTF} \circ \mathsf{MAJ}$, $\mathsf{MAJ} \circ \mathsf{LTF}$, $\mathsf{MAJ} \circ \mathsf{MAJ}$, and the figure alongside shows which class is contained in which by definition. The good news is that we do know quite a bit about three of these types. Let's see these one by one.

It is known (Theorem 24 in [GHR92]) that every LTF can be computed by a depth-2 threshold circuit with no large weight functions, that is, in $\mathsf{MAJ} \circ \mathsf{MAJ}$. This is quite surprising. (If you don't find it surprising, PAUSE, and try to construct a polynomial-size small-weight depth-2 threshold circuit for the GT function!) This containment is actually strict; we have already seen that $x \neq y$ function is a separating function; it is not an LTF, and we saw a depth-2 circuit computing it. More generally, let f be any "symmetric" Boolean function (for instance, PARITY); then it has a depth-2 circuit. Here symmetry means that the function value does not depend on the order amongs the arguments; $f(x_1, \cdots x_n)$ equals $f(x_{\pi(1)}, \cdots, x_{\pi(n)})$ for any permutation n. Such a function in fact is determined entirely by the value of $\mathsf{SUM} = x_1 + \cdots + x_n$. So it has the form $f(x) = 1 \iff \mathsf{SUM} \in S$ for some $S \subseteq \{0, 1, \cdots, n\}$. Now here's the depth-2 circuit: At the bottom level, we have $2|S|$ nodes, computing the MAJ functions $[\mathsf{SUM} \geq i?]$, $[-\mathsf{SUM} \geq -i?]$ for each $i \in S$. The top node just checks if at least $|S| + 1$ of these $2|S|$ nodes output a 1. Since $|S|$ is at most n, no large weights are required. Neat, isn't this?

A notable feature of the proof from [GHR92] that $\mathsf{LTF} \subseteq \mathsf{MAJ} \circ \mathsf{MAJ}$ is that it is not constructive. It does not give us an algorithm that can convert an LTF to a depth-2 small-weight threshold circuit; it only proves that such a circuit exists. Such proofs seem

useless to the algorithms designer, but are quite valuable for proving lower bounds. They typically proceed as follows: construct an object of the desired type (in this case, a depth-2 small-weight threshold circuit) "randomly" following some process that uses randomness - coin flips - in the construction. Carefully analyse the probability that this random circuit does not compute the desired function. Show that this probability is strictly less than 1. Ergo, there must be some choice in the random process where the resulting circuit does compute the desired function. (The analysis can get quite tricky and intricate, so you need the probability theory toolkit even if you are not specifically looking at randomized computation!) It is a different matter that subsequently, explicit constructions have been devised, for instance, [Hof96].

A probabilistic argument.

The proof of the above result can be extended to show something even stronger: not just LTF, but even MAJ $\circ$ LTF is contained in MAJ $\circ$ MAJ (Theorem 26 in [GHR92]). That is, if weights at the top node are small, then the bottom weights don't matter.

The above two results are "simulation" results; an LTF node or circuit can be simulated by a small-weight threshold circuit with some restrictions. Coming to lower bounds, let's describe a function not in MAJ $\circ$ MAJ. The OddMaxBit function is defined as follows: $\mathsf{OMB}_n(\tilde{z}) = 1$ if the last 1 in $\tilde{z}$ is in an odd-numbered position. This is an LTF; check if $2^1z_1 - 2^2z_2 + 2^3z_3 - \cdots + (-1)^{n-1}2^nz_n$ is positive. Now we use a composition trick often used to produce hard functions: instead of giving the OMB function its n bits of input, give $2n$ input bits $\tilde{x}, \tilde{y}$ and evaluate OMB on the bit string where each z_i is 1 exactly when both x_i and y_i are 1. We denote this composed function by $\mathsf{OMB}_n \circ \mathsf{AND}_2$. It is clearly in LTF $\circ$ MAJ; the bottom layer nodes can compute the z_is as $[x_i + y_i \geq 2?]$, and the top node is the OMB function. The large weights at the top are essential; this function is provably not in MAJ $\circ$ MAJ. The proof of this proceeds as follows: if a function f has a MAJ $\circ$ MAJ circuit, then at least one of the LTF functions from the bottom layer must do a pretty good job of "discriminating" between the zeros and the ones of f ([HMP$^+$93]). To make this more precise, let $A = \{\tilde{z} \mid f(z) = 1\}$ and $B = \{\tilde{z} \mid f(z) = 0\}$.

For an arbitrary function g, consider the quantity

$$\left| \frac{|A \cap g^{-1}(1)|}{|A|} - \frac{|B \cap g^{-1}(1)|}{|B|} \right|.$$

If $g = f$, then the first term is 1 and the second is 0, so this quantity is 1. Otherwise, the first quantity could be less than 1; the second could be more than 0. We say that g is a β-discriminator if the difference is at least β. A $\mathsf{MAJ} \circ \mathsf{LTF}$ circuit for f gives a reasonably good discriminator (good means β is not tiny) that is itself an LTF. (This will be the case even if we give differing importance to different inputs – imagine a probability distribution μ on the inputs of f, usually it is uniform, all inputs are equally likely; then as μ changes, different LTFs from the bottom layer may be the appropriate good discriminator.) This means that in the matrix M_f, there must be a large submatrix with large "discrepancy" – large imbalance between 1s and 0s. But for the function $\mathsf{OMB}_n \circ \mathsf{AND}_2$, we know that the corresponding matrix has exponentially small discrepancy – every submatrix is almost balanced ([BVdW07]).

the discrepancy technique; very useful in showing that some functions have large communication complexity.

Another very recent result is that $\mathsf{LTF} \circ \mathsf{MAJ} \subsetneq \mathsf{LTF} \circ \mathsf{LTF}$ ([CM18]). That is, if weights at the top node are large, then the weights at the bottom do matter. The separating function is again a composed function: $\mathsf{OMB}_n \circ \mathsf{EQ}_n$. Here, EQ_n is the function with $2n$ arguments that outputs 1 exactly when $\tilde{x} = \tilde{y}$. So the composed function can be thought of as follows: There is a set of n^2 variables arranged in a square matrix X, and a set of n^2 variables arranged in a square matrix Y. Find the largest index $i \in [n]$, such that the ith rows of X and Y are identical. (If there is no such row, set $i = 0$.) Now output 1 exactly when i is odd. (Stop for a moment to figure out why this function is in $\mathsf{LTF} \circ \mathsf{LTF}$!) Now for the lower bound: let's re-encode the matrix M_f with $-1, 1$ instead of $0, 1$. For any matrix M with real entries, its "sign rank" is the minimum rank of a real-valued matrix each of whose entries has the same sign as the corresponding entry of M. In [FKL$^+$01], it is shown that if a function has large sign-rank, then it cannot have small $\mathsf{LTF} \circ \mathsf{MAJ}$ circuits. And in [CM18] it is shown that the $\mathsf{OMB} \circ \mathsf{EQ}$ function indeed has exponentially large sign-rank.

the sign-rank technique; also invaluable in communication complexity.

Here's a nice challenging question. The Inner Product function IP is defined as follows:

$$\mathsf{IP}_n(\tilde{x},\tilde{y}) = \begin{cases} 1 & \text{if } x_1y_1 + \cdots + x_ny_n \text{ is odd} \\ 0 & \text{otherwise} \end{cases}$$

It was shown that this function is not in MAJ ∘ MAJ ([HMP+93]) using the discriminator-circuit technique. It was then shown to be not even in LTF ∘ MAJ ([For02]), by showing that it has large sign-rank. To this day, we do not know whether it is in LTF ∘ LTF. Can we prove this? Or can we design a polynomial-size LTF ∘ LTF circuit computing it?

4. Beyond depth-2

In the foregoing discussion we have only referred to results concerning depth-2 threshold circuits. However, the simulation results have analogues at larger depths as well. For instance, in [GHR92] it is shown that any depth-d threshold circuit with a small-weight LTF at the top has an equivalent depth-d threshold circuit, not much larger, where the weights at all nodes are small; that is, it uses only MAJ nodes. Also, it is shown that every depth-d threshold circuit with LTF nodes has an equivalent depth-$d+1$ threshold circuit, not much larger, with small-weights everywhere (only MAJ nodes).

Here's a summary of containments and separations.

MAJ	$\subsetneq$	LTF	separated by GT
	$\subsetneq$	MAJ ∘ LTF	separated by PARITY
	$=$	MAJ ∘ MAJ	
	$\subsetneq$	LTF ∘ MAJ	separated by OMB ∘ AND
	$\subsetneq$	LTF ∘ LTF	separated by OMB ∘ EQ
	$\subseteq$	MAJ ∘ MAJ ∘ MAJ	Is this strict? Not yet known

5. Conclusion

Recall that threshold circuits with depth $d > 2$ are exactly what are popularly called deep neural networks with $d - 1 \geq 2$ hidden layers. Current machine learning algorithms work with deep

neural networks, which seem to give more power, but until and unless we prove lower bounds for depth-2 circuits, we must keep in mind that "gaining additional computing power" through deep networks may be an illusion!

Suggested Reading

[BVdW07] **Harry Buhrman, Nikolay Vereshchagin, and Ronald de Wolf. On computation and communication with small bias. In *Proceedings of the Twenty-Second Annual IEEE Conference on Computational Complexity*, CCC '07, pages 24–32. IEEE Computer Society, 2007.**

[CM18] **Arkadev Chattopadhyay and Nikhil S. Mande. A short list of equalities induces large sign rank. In *Proc. 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS),*, 2018. preliminary version in ECCC TR 2017-083.**

[FKL+01] **Jürgen Forster, Matthias Krause, Satyanarayana V. Lokam, Rustam Mubarakzjanov, Niels Schmitt, and Hans Ulrich Simon. Relations between communication complexity, linear arrangements, and computational complexity. In *FST TCS 2001: Foundations of Software Technology and Theoretical Computer Science, 21st Conference, Bangalore, India, December 13-15, 2001, Proceedings*, pages 171–182, 2001.**

[For02] **Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *Journal of Computer and System Sciences*, 65(4):612–625, 2002.**

[GHR92] **Mikael Goldmann, Johan Håstad, and Alexander A. Razborov. Majority gates vs. general weighted threshold gates. *Computational Complexity*, 2:277–300, 1992.**

[HMP+93] **András Hajnal, Wolfgang Maass, Pavel Pudlák, Mario Szegedy, and György Turán. Threshold circuits of bounded depth. *Journal of Computer and System Sciences*, 46(2):129–154, 1993.**

[Hof96] **Thomas Hofmeister. A note on the simulation of exponential threshold weights. In *Computing and Combinatorics, Second Annual International Conference, COCOON '96, Hong Kong, June 17-19, 1996, Proceedings*, pages 136–141, 1996.**

[KN97] **Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.**