Some Complexity Measures for Boolean Functions

Xinyu Mao
Shanghai Jiao Tong University

January 7, 2021

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is a function that takes an input of n bits, and outputs a single bit. For input $x \in \{0,1\}^n$, we use x_i ot denote its ith bit. We use $|x|$ to denote the Hamming distance of x, i.e, number of ones in x. Here are some examples:

- $\texttt{zero}_n(x) = 0$ for all $x \in \{0,1\}^n$;
- $\texttt{or}_n(x) = 1$ iff $|x| \geq 1$;
- $\texttt{maj}_n(x) = 1$ iff $|x| \geq n/2$.

At first glance, one might say: "The function $\texttt{or}_n$ is more *complicated* than $\texttt{zero}_n$." However, what does 'more complicated' mean here? In other words, *how to measure the* ***complexity*** of Boolean functions? Also, inspired by the study of Turing machines, we may ask: can we use a more concise computational model when we focus only on Boolean functions? Perhaps the simplest one is the model of *decision trees*, which is investigated in section section 1.

Moreover, we shall discuss several complexity measures for Boolean functions, including sensitivity, block sensitivity, and the degree of a representing polynomial. These measures have close connection to the desition tree complexity, thus providing various techniques for studying the latter; these are elaborated in section 2 and section 3.

This essay serves as an assignment for CS087-1. I learned a lot from [4] and Chapter 12 of [2] in the midst of writing.

Acknowledegement I would like to warmly thank Jiapeng Zhang, who is now an assistant professor at USC, for his enlightening lecture about Boolean functions on Oct.26, 2020. The lecture also inspires this essay's topic selection.

1 Decision tree complexity

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function. If we can access one bit (not necessarily in order) of the input x at a time, the following question is natural: in order to compute f, *how many bits we need to examine*?

We may describe a deterministic algorithm for f as a tree.

Decition tree. A *decision tree* for f is a binary tree such that

1. each internal node is labeled with some $i \in [n]$;
2. each leaf is labeled with an output value 0 or 1;
3. for $\sigma \in \{0,1\}$, exactly one of the two outgoing edge of an node is labeled with σ.

The computation on x starts at the root and proceeds as follows: at each node, say the label is i, x_i is examined. The computation continues by moving to the node reached by the edge with label x_i from the current node. If a leaf is reached, the output value at the leaf is $f(x)$. See fig. 1 for an example.

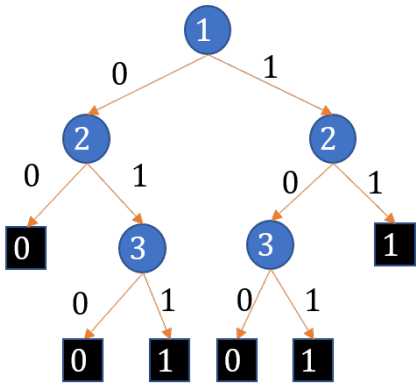

Figure 1: A decision tree that computes $\mathtt{maj}_3$.

Let $\mathcal{T}_f$ be the set of decision trees that computes f. The height of a decision tree $T \in \mathcal{T}_f$, denoted by $h(T)$, is the depth of the deepest node. For instance, the

height of the tree in fig. 1 is 3. That is, $h(T)$ is the number of bits examined by T on the worst input. Then the following definition is quite straightforward.

Definition 1. *The decision tree complexity of f is defined as*

$$D(f) := \min_{T \in \mathcal{T}_f} h(T).$$

Intuitively, $D(f)$ is the number of bits we need to examine using the optimal algorithm to compute f. Clearly, $D(f) \leq n$. It is interesting that for some function, this trivial bound is tight, i.e., $D(f) = n$. If $D(f) = n$, we say f is *evasive*.

Proposition 2. or_n *is evasive.*

Proof. To do this, we use an *adversary argument*. Let $T \in \mathcal{T}_{\mathsf{or}_n}$. We think of an execution of T, where each query on input is answered by some adversary $\mathcal{A}$. $\mathcal{A}$ always answered 0 for the first $n-1$ queries so that T has to continue its computation until the last bit is revealed. Hence, there exists an input x that consists of the answers made by $\mathcal{A}$, such that x has depth n in T. Since T is arbitrary, we conclude that $D(\mathsf{or}_n) = n$. □

A *graph property* is a Boolean function $P : \{0,1\}^{\binom{m}{2}} \to \{0,1\}$, where $m := |V|$ is the number of vertices. It is easy to see that an input $x \in \{0,1\}^{\binom{m}{2}}$ can be interpreted as a undirected graph: for each $e \in \binom{V}{2}$, e is in G iff $x_e = 1$, assuming that the bits in x is indexed by $\binom{V}{2}$.

Intuitively, a graph property does not depend on the indices of vertices. For example, conditions such as connectivity, being bipartite are graph properties, while things like 'vertex 1 and vertex 2' are connected' are not.

Proposition 3. *Graph connectivity is evasive.*

Proof Sketch. Let $P :=$ 'G is connected'. We again draw on adversary argument. Suppose that decision tree T computes P. The adversary $\mathcal{A}$ answers 0 unless answering 0 would imply that the graph was disconnected. This way, the known existing edges form a spanning forest, and the forest does not turn into a spanning tree until the last edge is revealed. □

Aanderaa-Karp-Rosenberg Conjecture Not all graph properties are evasive [3]. A graph property is *monotone* if adding edges to the graph preserves the property, e.g., connectivity. Stål Aanderaa, Richard M. Karp, and Arnold L. Rosenberg made the following conjecture:

Conjecture 4. *Every non-constant monotone graph property is evasive.*

This conjecture is still open so far. Rivest and Vuillemin proved that if m is a prime power, conjecture 4 is true [7].

2 Sensitivity

In this section, we introduce *sensitivity* of a function $f : \{0,1\}^n \to \{0,1\}$, which leads to lower bounds on decision tree complexity.

For $S \subseteq [n]$, let x^S be the input obtained from x by flipping each x_i for $i \in [S]$. Write $x^i := x^{\{i\}}$(flipping the ith bit of x) for simplicity.

Define *the sensitivity of f at x* as

$$s_x(f) := \#\{i \in [n] : f(x) \neq f(x^i)\}.$$

Let $Q_n = (V, E)$ be the n-dimensional hypercube, where $V = \{0,1\}^n, E := \{\{x, y\} : y = x^i \text{ for some } i\}$. View f as a 2-coloring on Q_n, then $s_x(f)$ is the number of adjacent vertices whose color is different with x.

Definition 5. *The quantity $s(f) := \max_{x\in\{0,1\}^n} s_x(f)$ is called the sensitivity of f.*

An extension of sensitivity is *block sensitivity*. The *block sensitivity of f at x* is the maximum number of k, such that there exists a partition of $[n]$, say $B_1, B_2, \cdots, B_k$, satisfying $f(x) \neq f(x^{B_i}), \forall i \in [k]$.

Definition 6. *The block sensitivity of f is defined via $bs(f) := \max_{x\in\{0,1\}^n} bs_x(f)$.*

Clearly, $s(f) \leq bs(f)$. It is conjectured that $bs(f) = O(s(f)^c)$ for some constant c. This is known as *sensitivity conjecture*, and is proven to be true by Hao Huang in recent years:

Theorem 7 (Hao Huang, [5]). $bs(f) \leq s(f)^4$.

Proposition 8. $s(f) \leq bs(f) \leq D(f)$.

Proof. Let x be such that $bs(f) = bs_x(f) = s$ with $B_1, B_2, \ldots, B_s$ be the corresponding partition of $[n]$. Let $T \in \mathcal{T}_f$. When given x as input, T has to query at least one bit in each block B_i for every $i \in [s]$ in order to distinguish x from x^{B_i}. □

Also, sensitivity can also give an upper bound on $D(f)$:

Theorem 9 ([1]). $D(f) \leq bs(f)^3$.

Together with theorem 7, we have $s(f) \leq D(f) \leq s(f)^{12}$.

3 Polynomial representation of Boolean functions

Another way to give lower bounds on $D(f)$ involves polynomial representation of f.

A polynomial $p \in \mathbb{C}[x_1, x_2, \ldots, x_n]$ is *multilinear* if for all $i \in [n]$ and $\alpha, \beta \in \mathbb{C}$,

$$p(x_1, \ldots, x_{i-1}, \alpha x_i + \beta x_i', \ldots, x_n) = \alpha p(x_1, \ldots, x_n) + \beta p(x_1, \ldots, x_{i-1}, x_i', \ldots, x_n).$$

That is, the degree of each variable in p is 1. Or equivalently, p must be of the form $p(x) = \sum_{S \subseteq [n]} (c_S \prod_{i \in S} x_i)$ for some complex numbers c_S.

We say a polynomial $p \in \mathbb{C}[x_1, x_2, \ldots, x_n]$ *represents* Boolean function $f : \{0,1\}^n \to \{0,1\}$ if $p(x) = f(x)$ for all $x \in \{0,1\}^n$. In fact, we have

Proposition 10. *Each Boolean function $f : \{0,1\}^n \to \{0,1\}$ has a unique multilinear polynomial p that represents f.*

Proof. It is easy to construct a such a polynomial from f. For each $a \in \{0,1\}^n$, define a polynomial

$$\chi_a(x_1, \ldots, x_n) = \prod_{i:a_i=1} x_i \prod_{i:a_i=0} (1 - x_i).$$

Clearly, χ_a is multilinear. Now we set

$$p_f := \sum_{a \in \{0,1\}^n} f(a) \chi_a.$$

One can easily check that p_f represents f (and it is of course multilinear).

Assume that a multilinear polynomial p' also represents f, we shall prove that $p' = p_f$. Note that the polynomial $p := p' - p_f$ is multilinear and $p(x) = 0$ for all $x \in \{0,1\}^n$. Write $p(x) = \sum_{S \subseteq [n]} (c_S \prod_{i \in S} x_i)$. An induction on $|S|$ show that $c_S = 0$ for all $S \subseteq [n]$, which means $p \equiv 0$. Hence, $p' = p_f$, that is, p_f is the unique multilinear polynomial that represents f. □

The degree of f, denoted by $\deg(f)$, is the degree of the multilinear polynomial that represents f. According to proposition 10, $\deg(f)$ is well-defined.

For example, or_n is represented by

$$p(x) := 1 - \prod_{i=1}^{n} (1 - x_i),$$

and hence $\deg(\mathsf{or}_n) = n$. In fact, we have

Theorem 11. $\deg(f) \leq D(f)$.

Proof. Let $T \in \mathcal{T}_f$ such that $h(T) = D(f)$. It suffices to construct a multilinear polynomial p that represents f with $\deg(p) \leq h(T)$.

For each leaf L, let $(b_1, e_1), (b_2, e_2), \ldots, (b_\ell, e_\ell) \in [n] \times \{0, 1\}$ be the path from root to L in T. where b_i is the label on the node, e_i is the label on the edge that starts from b_i. We study the polynomial

$$\chi_L(x) := \prod_{i:e_i=1} x_{b_i} \prod_{i:e_i=0} (1 - x_{b_i}).$$

Note that for $x \in \{0, 1\}^n$, if leaf L is reached on input x, $\chi_L(x) = 1$; otherwise, $\chi_L(x) = 0$. Set $p(x) := \sum_L \chi_L$, where the sum runs over all leaf L with output value 1. One can see that p has the properties we want. □

A rough upper bound of $D(f)$ is also known (proved by Nisan and Smolensky [6]):

Theorem 12. *For every Boolean function f,*

1. $bf(f) \leq 2 \deg(f)^2$;
2. $D(f) \leq \deg(f)^2 bs(f)$.

Therefore, $\deg(f) \leq D(f) \leq 2deg(f)^4$.

References

[1] A. Ambainis, *Polynomial degree vs. quantum query complexity*, Journal of Computer and System Sciences, 72 (2006), pp. 220 – 238. JCSS FOCS 2003 Special Issue. 5

[2] S. Arora and B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, 2009. 2

[3] M. Best, P. van Emde Boas, and H. Lenstra, *A Sharpened Version of the Aanderaa-Rosenberg Conjecture*, Mathematisch Centrum Amsterdam. Afdeling Zuivere Wiskunde: ZW, Stichting Mathematisch Centrum, 1974. 4

[4] H. Buhrman and R. de Wolf, *Complexity measures and decision tree complexity: a survey*, Theoretical Computer Science, 288 (2002), pp. 21 – 43. Complexity and Logic. 2

[5] H. Huang, *Induced subgraphs of hypercubes and a proof of the sensitivity conjecture*, 2019. 4

[6] N. Nisan and M. Szegedy, *On the degree of boolean functions as real polynomials*, Computational complexity, 4 (1994), pp. 301–313. 6

[7] R. L. Rivest and J. Vuillemin, *On recognizing graph properties from adjacency matrices*, Theoretical Computer Science, 3 (1976), pp. 371 – 384. 4