On the Distributional Complexity of Boolean Decision Trees

ChenGuang Liu[⋆] and Kazuyuki Tanaka

Mathematical Institute, Tohoku University, Sendai 980-8578, Japan
liu@mail.tains.tohoku.ac.jp, tanaka@math.tohoku.ac.jp

Abstract. Let $P(f)$ be the Las Vegas distributional complexity of function f, and $P^{\epsilon}(f)$ the Monte Carlo distributional complexity with error $\epsilon \in [0, 0.5]$. Yao [7] proved that $R(f) \geq P(f)$ and $2R^{\epsilon}(f) \geq P^{2\epsilon}(f)$, where $R(f)$ (resp. $R^{\epsilon}(f)$) is the Las Vegas (resp. Monte Carlo) randomized complexity of function f. In this paper, we prove that for any uniform alternating Boolean function f on n variables, the Las Vegas distributional complexity is given as $P(f) = \Omega(n^{log_2 \frac{1+\sqrt{5}}{2}})$, and the Monte Carlo distributional complexity $P^{\epsilon}(f) \leq n^{log_2 \frac{1+\sqrt{5}}{2}}$ for $\epsilon \in [0.0246, 0.5]$. Moreover, we show that $P^{\epsilon}(f) = (1 - 2\epsilon)P(f)$, and the distributional probability ρ belongs to $[\frac{\sqrt{7}-1}{3}, \frac{\sqrt{5}-1}{2}]$.

1 Introduction

The *Boolean decision tree* is a natural model for computing a Boolean function $f : \{0,1\}^n \rightarrow \{0,1\}$. In this paper, we focus on a class of Boolean decision trees of read-once formulae, which are also called uniform alternating Boolean decision trees (functions) in other papers. Such a tree may be defined as a full binary AND-OR tree T_2^k, $k \geq 0$, where the subscript 2 means "binary", and k is the number of rounds (one round: one level AND node followed by one level OR node). The root (at depth 0) is labeled $\wedge$ (AND node) as are all the internal nodes with even depth, while all the internal nodes with odd depth are labeled $\vee$ (OR node). The external nodes are labeled either 0 or 1, and our objective is to calculate the value of the root, following the operations labeled at each internal node.

Let A be a deterministic algorithm, and I_p a random input under the distribution p. By $C(A, I_p)$, we denote the expected number of input variables queried by A on I_p. The *distributional complexity* of function f, is defined as the cost of a best deterministic algorithm computing f for the worst distribution on input:

$$\max_p \min_A C(A, I_p).$$

In this paper, two different kinds of deterministic algorithm A will be considered:

[⋆] Corresponding author.

(1) *Las Vegas algorithm*, which computes the function f always correctly;
(2) *Monte Carlo algorithm*, which is allowed to output a wrong answer with some probability $\epsilon \in [0, \frac{1}{2}]$.

The distributional complexity of function f according to the Las Vegas algorithm is called the *Las Vegas distributional complexity*, denoted by $P(f)$, and to the Monte Carlo algorithm is called the *Monte Carlo distributional complexity*, by $P^{\epsilon}(f)$. The Las Vegas distributional complexity can be seen as the Monte Carlo distributional complexity with zero-error.

The *distributional probability* of function f is defined as the probability ρ such that

$$\min_{A} C(A, I_{\rho}) = P(f).$$

Boolean decision trees have been investigated thoroughly in the literature (See also Yao [7], Karp and Zhang [2], Saks and Wigderson [6], Liu and Tanaka [3]). For any Boolean decision tree, the distributional complexity is not greater than the deterministic complexity. Considering two roles of randomness in algorithms, randomness inside the algorithm itself, and randomness on the inputs, Yao [7] proved that, for any Boolean function f,

$$R(f) \geq P(f), \tag{1}$$

$$2R^{\epsilon}(f) \geq P^{2\epsilon}(f) \text{ for } \epsilon \in [0, \frac{1}{2}]. \tag{2}$$

where $R(f)$ is the Las Vegas randomized complexity of f, and $R^{\epsilon}(f)$ is the Monte Carlo randomized complexity with error ϵ. This is often referred as Yao's Principle in the literature. Moreover, Graaf and Wolf [1] advanced quantum versions of Yao principle.

Hence, to seek the distributional complexity provides us with a useful technique to establish lower bounds of randomized complexity. This method is quite useful, because it is usually much easier to analyze deterministic algorithms than to analyze randomized algorithms. In this paper, we prove that for any uniform alternating Boolean function f on n variables, the Las Vegas distributional complexity $P(f) = \Omega(n^{log_2 \frac{1+\sqrt{5}}{2}})$, and the Monte Carlo distributional complexity $P^{\epsilon}(f) \leq n^{log_2 \frac{1+\sqrt{5}}{2}}$ for $\epsilon \in [0.0246, 0.5]$. Moreover, we show that $P^{\epsilon}(f) = (1-2\epsilon)P(f)$ for $\epsilon \in [0, 0.5]$, and the distributional complexity $\rho \in [\frac{\sqrt{7}-1}{3}, \frac{\sqrt{5}-1}{2}]$.

2 Las Vegas distributional complexity

We assume that the leaves may receive a 0 with probability p (and a 1 with probability $1-p$) independently from one another. For any Boolean decision tree T_2^k, we are interested in two quantities. The first quantity is p_k, the probability of returning a 0 at the root. The second quantity is $\mu_k^p(f)$, the querying complexity to evaluate this Boolean decision tree for $p \in [0, 1]$. By the definition of the distributional complexity, it is clear that the Las Vegas distributional complexity $P_k(f)$ is given by $\max\limits_{p\in[0,1]} \mu_k^p(f)$.

By a node k (resp. $\underline{k}$), we denote a node labeled $\wedge$ (resp. $\vee$) at the k-th round of tree T_2^k. Node 0 is an external node. By p_k (resp. $p_{\underline{k}}$), we denote the probability of returning a 0 at the node k (resp. $\underline{k}$), and $\mu_k^p(f)$ (resp. $\mu_{\underline{k}}^p(f)$) the querying complexity to the node k (resp. $\underline{k}$) of the tree for function f. Then we obtain the recurrence

$$\begin{cases} p_0 &= p \\ \mu_0^p(f) &= 1 \end{cases}$$

$$\begin{cases} p_{\underline{k}} &= p_{k-1}^2 \\ \mu_{\underline{k}}^p(f) &= (1-p_{k-1}) \times \mu_{k-1}^p(f) + p_{k-1} \times 2 \times \mu_{k-1}^p(f) \\ &= p_{k-1} \times \mu_{k-1}^p(f) + \mu_{k-1}^p(f) \\ &= (p_{k-1}+1) \times \mu_{k-1}^p(f) \end{cases}$$

$$\begin{cases} p_k &= 1-(1-p_{\underline{k}})^2 \\ &= 1-(1-p_{k-1}^2)^2 \\ &= -p_{k-1}^4 + 2p_{k-1}^2 \\ \mu_k^p(f) &= p_{\underline{k}} \times \mu_{\underline{k}}^p(f) + (1-p_{\underline{k}}) \times 2 \times \mu_{\underline{k}}^p(f) \\ &= (2-p_{\underline{k}}) \times \mu_{\underline{k}}^p(f) \\ &= (2-p_{k-1}^2) \times (p_{k-1}+1) \times \mu_{k-1}^p(f) \\ &= (-p_{k-1}^3 - p_{k-1}^2 + 2p_{k-1} + 2) \times \mu_{k-1}^p(f) \end{cases}$$

To sum up, we obtain

$$\begin{cases} p_0 &= p \\ \mu_0^p(f) &= 1 \end{cases} \tag{3}$$

$$\begin{cases} p_k &= -p_{k-1}^4 + 2p_{k-1}^2 \\ \mu_k^p(f) &= (-p_{k-1}^3 - p_{k-1}^2 + 2p_{k-1} + 2) \times \mu_{k-1}^p(f) \end{cases} \tag{4}$$

Theorem 1 (Golden Section theorem). *For uniform alternating Boolean decision trees T_2^k, when $k \approx \infty$,*

$$p_k : \begin{cases} = 1 & p = 1 \\ \approx 1 & 1 > p > \frac{\sqrt{5}-1}{2} \\ = \frac{\sqrt{5}-1}{2} & p = \frac{\sqrt{5}-1}{2} \\ \approx 0 & 0 < p < \frac{\sqrt{5}-1}{2} \\ = 0 & p = 0 \end{cases}$$

Proof. By $p_k = -p_{k-1}^4 + 2p_{k-1}^2$, we have

$$\begin{aligned} p_k - p_{k-1} &= -p_{k-1}^4 + 2p_{k-1}^2 - p_{k-1} \\ &= -p_{k-1} \times (p_{k-1}-1) \times (p_{k-1} + \tfrac{1+\sqrt{5}}{2}) \times (p_{k-1} + \tfrac{1-\sqrt{5}}{2}) \end{aligned}$$

Then, we can easily observe that: