ELSEVIER

Theoretical Computer Science 262 (2001) 37–68

Theoretical Computer Science

www.elsevier.com/locate/tcs

Approximate evaluations of characteristic polynomials of Boolean functions ☆

David Lee[a,b,*], Henryk Woźniakowski[c,d]

[a] *Bell Laboratories Research China, 15/F Aero Space Great Wall Building, No. 30, Hai Dian Nan Lu, Beijing, China*

[b] *Bell Laboratories, 600 Mountain Ave, RM 2C-424, Murray Hill, NJ 07974, USA*

[c] *Department of Computer Science, Columbia University, NY 10027, USA*

[d] *Institute of Applied Mathematics, University of Warsaw, Warsaw, Poland*

Received 15 December 1998; revised 27 August 1999; accepted 29 February 2000

Abstract

Motivated by combinational circuit verification and testing, we study the approximate evaluation of characteristic polynomials of Boolean functions. We consider an oracle model in which the values of the characteristic polynomials are approximated using the evaluations of the corresponding Boolean functions. The approximation error is defined in the worst case, average case and randomized settings. We derive lower bounds on the approximation errors in terms of the number of Boolean function evaluations. We design algorithms with an error that matches the lower bound. Let $k(\varepsilon, n)$ denote the minimal number of Boolean function evaluations needed to reduce the initial error by a factor of ε where n is the number of Boolean variables. We show that $k(\varepsilon, n)$ is exponential in n in the worst and average case settings, and that it is independent of n and of order ε^{-2} in the randomized setting.

Keywords: Boolean function; Characteristic polynomial; Complexity; Combinational circuit verification; Testing

1. Introduction

In combinational circuit verification we want to check if two given circuits are identical, or equivalently, if two given Boolean functions are identical. That is, given two Boolean functions $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$ where $\boldsymbol{x}$ is a Boolean vector, we want to verify if they are identical: $f \equiv g$. The problem is obviously NP-hard. In testing [10] we have a specification circuit, a Boolean function $f(\boldsymbol{x})$, for which we have complete information, and an implementation circuit, a Boolean function $g(\boldsymbol{x})$, which is given

☆ This work was done while consulting at Bell Laboratories, and is partially supported by the National Science Foundation.

* Corresponding author. Tel.: +1-908-582-5872; fax: +1-908-582-6632.

E-mail addresses: lee@research.bell-labs.com (D. Lee), henryk@cs.columbia.edu (H. Woźniakowski).

0304-3975/01/$ - see front matter
PII: S0304-3975(00)00151-1

as a "black-box", to which we can only supply inputs and observe outputs. We want to check if the implementation *conforms* to the specification, or equivalently, if their corresponding Boolean functions are identical: $f \equiv g$.

One approach for solving the circuit verification and testing problems is to use characteristic polynomials [14, 1]. For a Boolean function $f(\boldsymbol{x})$ of n Boolean variables there is a unique characteristic polynomial $F(\boldsymbol{X})$ of n real variables defined in $[0,1]^n$. Two Boolean functions f and g are identical if and only if their corresponding characteristic polynomials F and G are identical. This can be checked by the one evaluation of the characteristic polynomials at a randomly chosen real vector, and that gives a correct answer with probability one. Specifically, let $\boldsymbol{X}^*$ be a real vector chosen uniformly at random from $[0,1]^n$. If $F(\boldsymbol{X}^*) \neq G(\boldsymbol{X}^*)$ then obviously $F \not\equiv G$ and hence $f \not\equiv g$. Conversely, if $F(\boldsymbol{X}^*) = G(\boldsymbol{X}^*)$ then $F \equiv G$ and hence $f \equiv g$ with probability one. Therefore, we reduce the problem of checking Boolean function equivalence (or circuit equivalence) to an evaluation of characteristic polynomials at a real vector. Note that for verification we have complete information of both Boolean functions for an evaluation of their corresponding characteristic polynomials. However, for testing we only have complete information of the specification Boolean function f and we do not know the structure of the implementation Boolean function g; we can only observe the Boolean outputs from its Boolean inputs. Also note that characteristic polynomials are multi-linear extensions of Boolean functions, as is also used in probabilistically checkable-proof literature [13, 4, 21].

There are algorithms for the exact evaluation of the characteristic polynomials. One is the Shannon Expansion [1] which is effective if all Boolean function values are known, and the degree of the characteristic polynomial is relatively small or Boolean functions have certain properties [11]. However, it is, in general, impractical to exactly evaluate the characteristic polynomial of a given Boolean function due to an exponential "blow-up" of the number of terms in the characteristic polynomial. In fact, the exact evaluation of characteristic polynomials is equivalent to the satisfiability problem and is known to be NP-hard [9, 12].

Therefore, we study *approximate* evaluation of characteristic polynomials at real vectors. To cover the problem of circuit testing, we use an oracle model. That is, we do not know the structure of a Boolean function f; instead, we assume that the values $f(\boldsymbol{x})$ for any Boolean vector $\boldsymbol{x}$ can be obtained, and we want to approximate the values of its characteristic polynomial from the Boolean function evaluations. Our model is weaker than the commonly used model in the study of satisfiability and verification of Boolean functions where it is assumed that one knows the structure of the Boolean functions [12].

Suppose that we use k Boolean function values for an approximate evaluation of a characteristic polynomial of a Boolean function of n variables. We consider approximation errors in the worst case, average case, and randomized settings. We provide algorithms with errors that match the lower bounds of approximation errors.

Suppose that with k Boolean function evaluations the approximation error lower bound is e_k. We consider a *relative* error, e_k/e_0, which is the reduction of errors

with k Boolean function evaluations from the initial error without any Boolean values. We compute the smallest number $k = k(n, \varepsilon)$ of Boolean function values, which is needed to obtain a relative error $\varepsilon \in [0, 1)$. We prove that $k(n, \varepsilon)$ is exponential in n for all $\varepsilon \in [0, 1)$ in the worst and average case settings. Specifically, this holds for the L_p-norm, $p \in [1, \infty]$, in the worst case setting, and for the L_2-norm in the average case setting. Hence, the problem of approximate evaluations of characteristic polynomials in the oracle model is *intractable*. In the randomized setting, we consider the L_2-norm and prove that $k(n, \varepsilon)$ is independent of n and is of order ε^{-2}. Hence, $k(\varepsilon, n)$ is polynomial in $1/\varepsilon$ and the problem is *tractable* in $1/\varepsilon$.

The proof techniques in this paper are typical in information-based complexity [23] where the oracle model is studied for general computational problems. To make this paper self-contained we present complete proofs without shortcuts which are made possible by using information-based complexity results. Therefore, knowledge of information-based complexity is not needed, although it might be helpful to the readers.

We now summarize the contents of this paper. Section 2 deals with the definition and properties of characteristic polynomials $F(\boldsymbol{X})$ of Boolean functions $f : \{0, 1\}^n \to \{0, 1\}$. Here, $\boldsymbol{X} \in [0, 1]^n$. In particular, we show that the value of a characteristic polynomial is given by a multivariate integral over the unit cube $[0, 1]^n$. This representation permits us to use the classical Monte Carlo algorithm for the approximate evaluation of characteristic polynomials in the randomized setting. In Section 3, we introduce algorithms that use Boolean functions values at k sample points. We study two classes of such algorithms depending on whether the sample points depend on the vector $\boldsymbol{X}$. Section 3 deals with the worst case setting in which the approximation error is determined by the worst possible Boolean function and by taking the L_p-norm with respect to $\boldsymbol{X}$ for $p \in [1, \infty]$. We prove that the problem is intractable for all ε and all p. Section 4 deals with the average case setting. The approximation error is relaxed by averaging over the Boolean function space. We assume that Boolean functions are equi-probable and we choose the L_2-norm for vectors $\boldsymbol{X}$. Due to the averaging "artifacts", the initial error is exponentially small in n but the problem remains intractable. Section 5 deals with the fixed evaluation point problem where we consider the approximate evaluation of characteristic polynomials for a fixed vector $\boldsymbol{X}$. We prove that the measure of the set of vectors $\boldsymbol{X}$ for which the initial error is reduced by a factor β is exponentially small in n. This holds in both the worst and average case settings. Section 6 deals with the randomized setting in which the sample points and the algorithms are chosen randomly. If randomization is independent of the evaluation point $\boldsymbol{X}$ then the problem is still intractable. If, however, randomization depends on $\boldsymbol{X}$ then the problem becomes tractable. In the final Section 7, we comment on related works.

2. Characteristic polynomials

In this section we discuss characteristic polynomials of Boolean functions and derive their basic properties. We use lower case letters for Boolean variables or numbers with

values in $\{0,1\}$: x_i, $i=1,2,\ldots,n$, and Boolean vectors $\boldsymbol{x}=[x_1,x_2,\ldots,x_n]$. We use upper case letters for real variables or numbers with values in $\mathbb{R}$: X_i, $i=1,2,\ldots,n$, and real vectors $\boldsymbol{X}=[X_1,X_2,\ldots,X_n]$. Similarly, we use lower case letters for Boolean functions $f:\{0,1\}^n\to\{0,1\}$ and upper case letters for real functions $F:\mathbb{R}^n\to\mathbb{R}$.

We consider Boolean functions (expressions) of n variables $\boldsymbol{x}=[x_1,x_2,\ldots,x_n]$. A Boolean function f can be uniquely represented by a truth table. The standard *sum of products* form of f can be obtained from the given truth table by taking a minterm for each combination of variables which produces a value 1 (TRUE) in the function, and then taking the OR of all those minterms. Let K be the total number of TRUE's in the truth table, $K\in[0,2^n]$. Then

$$f=E_1\vee E_2\vee\cdots\vee E_K, \tag{1}$$

where each minterm is

$$E_j=\alpha_{j,1}\wedge\alpha_{j,2}\wedge\cdots\wedge\alpha_{j,n},\quad j=1,2,\ldots,K$$

and

$$\alpha_{j,i}=x_i\quad\text{or}\quad\alpha_{j,i}=x_i{}':=1-x_i.$$

If we replace each Boolean variable x_i by a real variable X_i, x_i' by $1-X_i$, AND operation by product, and OR by summation, then we obtain a real-valued polynomial of n variables. Specifically, we construct a real-valued polynomial by the following substitutions in (1):

$$x_i\leftarrow X_i$$

$$x_i'\leftarrow 1-X_i$$

$$\wedge\leftarrow\cdot$$

$$\vee\leftarrow+.$$

We obtain a real-valued polynomial of n variables: $F(\boldsymbol{X})=F(X_1,X_2,\ldots,X_n)$ [14]. For a Boolean function f, the corresponding polynomial F is unique, and is called the *characteristic polynomial* of the given Boolean function. We denote the transformation of a Boolean function to its characteristic polynomial by τ and we have

$$F(\boldsymbol{X})=\tau(f)(\boldsymbol{X})=\sum_{\boldsymbol{x}=[x_1,x_2,\ldots,x_n]\in\{0,1\}^n} f(\boldsymbol{x})E_{\boldsymbol{x}}(\boldsymbol{X}), \tag{2}$$

where

$$E_{\boldsymbol{x}}(\boldsymbol{X})=\prod_{i=1}^{n}X_i^{x_i}(1-X_i)^{1-x_i}\quad\text{with } 0^0=1.$$

Observe that $E_{\boldsymbol{x}}$ as well as F are polynomials of a total degree n which are linear in each variable X_i. Clearly, $E_{\boldsymbol{x}}(\boldsymbol{X}) \geqslant 0$ for $\boldsymbol{X} \in [0,1]^n$. It is easy to show by induction on n that

$$\sum_{\boldsymbol{x}\in\{0,1\}^n} E_{\boldsymbol{x}}^p(\boldsymbol{X}) = \prod_{i=1}^{n} (X_i^p + (1-X_i)^p), \quad \forall \boldsymbol{X} \in [0,1]^n, \ \forall p \geqslant 0. \tag{3}$$

Indeed, for $n=1$ it is trivial and

$$\begin{aligned}
&\sum_{\boldsymbol{x}\in\{0,1\}^n} E_{\boldsymbol{x}}^p(\boldsymbol{X}) \\
&= \sum_{\boldsymbol{x}=[x_1,\ldots,x_{n-1}]\in\{0,1\}^{n-1}} \left(\prod_{i=1}^{n-1} X_i^{x_i p} (1-X_i)^{(1-x_i)p} (X_n^p + (1-X_n)^p) \right) \\
&= (X_n^p + (1-X_n)^p) \sum_{\boldsymbol{x}\in\{0,1\}^{n-1}} E_{\boldsymbol{x}}^p(X_1, X_2, \ldots, X_{n-1}) \\
&= \prod_{i=1}^{n} (X_i^p + (1-X_i)^p).
\end{aligned}$$

In particular, for $p=1$ we have

$$\sum_{\boldsymbol{x}\in\{0,1\}^n} E_{\boldsymbol{x}}(\boldsymbol{X}) = 1, \quad \forall \boldsymbol{X} \in [0,1]^n.$$

From this we have

$$0 \leqslant F(\boldsymbol{X}) \leqslant 1, \quad \forall \boldsymbol{X} \in [0,1]^n.$$

These properties allow a probabilistic interpretation of the value of the characteristic polynomial $F(\boldsymbol{X})$. Namely, let $0 \leqslant X_i \leqslant 1$, $i = 1,2,\ldots,n$. Let us interpret X_i as the probability that the independent random Boolean variables x_i's take value 1. Then for each minterm E_j in (1), $E_{\boldsymbol{x}}(\boldsymbol{X})$ is the probability that this minterm takes Boolean value 1. Taking the summation, $F(\boldsymbol{X}) = \tau(f)(\boldsymbol{X})$ is the probability that the Boolean function f takes value 1. We summarize this interpretation in

Proposition 1. *Let $0 \leqslant X_i \leqslant 1$ be the probability that the independent random Boolean variables x_i take value 1, $i = 1,\ldots,n$. Then the value of the characteristic polynomial at $\boldsymbol{X}$, $\tau(f)(\boldsymbol{X})$, is the probability that the Boolean function f takes value 1.*

We now derive Shannon's expansion for characteristic polynomials. For a Boolean function $f : \{0,1\}^n \to \{0,1\}$ by $f_{x_1=a} : \{0,1\}^{n-1} \to \{0,1\}$ we mean a Boolean function such that $f_{x_1=a}(x_2,x_3,\ldots,x_n) = f(a,x_2,x_3,\ldots,x_n)$. Here, obviously, $a \in \{0,1\}$.

Proposition 2. *Shannon's expansion*

$$\tau(f)(\boldsymbol{X}) = X_1\tau(f_{x_1=1})(X_2,X_3,\ldots,X_n) + (1-X_1)\tau(f_{x_1=0})(X_2,X_3,\ldots,X_n).$$

Proof. Indeed, for $n=1$ it is trivial and for $n\geqslant 2$ we use induction. From (2) we have

$$\begin{aligned}\tau(f)(\boldsymbol{X}) &= X_1 \sum_{\boldsymbol{x}=[x_2,x_3,\ldots,x_n]\in\{0,1\}^{n-1}} f(1,\boldsymbol{x})E_{\boldsymbol{x}}(X_2,X_3,\ldots,X_n)\\ &\quad + (1-X_1)\sum_{\boldsymbol{x}=[x_2,x_3,\ldots,x_n]\in\{0,1\}^{n-1}} f(0,\boldsymbol{x})E_{\boldsymbol{x}}(X_2,X_3,\ldots,X_n)\\ &= X_1\tau(f_{x_1=1})(X_2,X_3,\ldots,X_n) + (1-X_1)\tau(f_{x_1=0})(X_2,X_3,\ldots,X_n),\end{aligned}$$

as claimed. □

We now show that characteristic polynomials of Boolean functions are related to multivariate integrals over the unit cube. To do this let us define a step function $S:[0,1]^2\to\{0,1\}$ by

$$S(Y,X) = \begin{cases} 1 & 0\leqslant Y\leqslant X\leqslant 1,\\ 0 & 0\leqslant X< Y\leqslant 1\end{cases}$$

and its multivariate analog

$$\boldsymbol{S}(\boldsymbol{Y},\boldsymbol{X}) = [S(Y_1,X_1),S(Y_2,X_2),\ldots,S(Y_n,X_n)].$$

Proposition 3. *Given a Boolean function f of n variables and a real vector $\boldsymbol{X}\in[0,1]^n$, the value of the characteristic polynomial at $\boldsymbol{X}$, $F(\boldsymbol{X})=\tau(f)(\boldsymbol{X})$, is equal to the mean of $f(\boldsymbol{S}(\boldsymbol{Y},\boldsymbol{X}))$ for a uniformly distributed $\boldsymbol{Y}$ over $[0,1]^n$. That is,*

$$F(\boldsymbol{X}) = \tau(f)(\boldsymbol{X}) = \int_{[0,1]^n} f(\boldsymbol{S}(\boldsymbol{Y},\boldsymbol{X})\,\mathrm{d}\boldsymbol{Y}. \tag{4}$$

Proof. We use induction on n. For $n=1$ we have

$$\begin{aligned}\int_0^1 f(S(Y_1,X_1)\,\mathrm{d}Y_1 &= \int_0^{X_1} f(1)\,\mathrm{d}Y_1 + \int_{X_1}^1 f(0)\,\mathrm{d}Y_1\\ &= f(1)X_1 + f(0)(1-X_1),\end{aligned}$$

which, due to (2) is equal to $F(X_1)$.

For $n\geqslant 2$ we have

$$\begin{aligned}&\int_{[0,1]^n} f(\boldsymbol{S}(\boldsymbol{Y},\boldsymbol{X})\,\mathrm{d}\boldsymbol{Y}\\ &\quad = \int_{[0,1]^{n-1}} \mathrm{d}Y_2\cdots\mathrm{d}Y_n\left(\int_0^{X_1} f(1,S(Y_2,X_2),\ldots,S(Y_n,X_n))\,\mathrm{d}Y_1\right.\end{aligned}$$

$$+\int_{X_1}^{1} f(0,S(Y_2,X_2),\ldots,S(Y_n,X_n))\,\mathrm{d}Y_1\Bigg)$$

$$=\int_{[0,1]^{n-1}} (X_1\cdot f_{x_1=1}(S(Y_2,X_2),\ldots,S(Y_n,X_n))$$

$$+(1-X_1)\cdot f_{x_1=0}(S(Y_2,X_2),\ldots,S(Y_n,X_n)))\,\mathrm{d}Y_2\cdots\mathrm{d}Y_n.$$

From the inductive hypothesis we conclude

$$\int_{[0,1]^n} f(\boldsymbol{S}(\boldsymbol{Y},\boldsymbol{X})\,\mathrm{d}\boldsymbol{Y} = X_1\tau(f_{x_1=1})(X_2,\ldots,X_n)+(1-X_1)\tau(f_{x_1=0})(X_2,\ldots,X_n)$$

and by Shannon's expansion, we have proved (4). □

Formula (4) will be used later for approximate evaluations of the characteristic polynomial. Observe that the integrand in (4) is piecewise constant and is, in general, a discontinuous function.

We now find more relations between the Boolean functions and their characteristic polynomials. Observe for any distinct $\boldsymbol{x},\boldsymbol{y}\in\{0,1\}^n$ we have

$$E_{\boldsymbol{x}}(\boldsymbol{x})=1 \quad \text{and} \quad E_{\boldsymbol{x}}(\boldsymbol{y})=0. \tag{5}$$

This yields

$$F(\boldsymbol{x})=f(\boldsymbol{x}), \quad \forall \boldsymbol{x}\in\{0,1\}^n. \tag{6}$$

Hence, two Boolean functions f and g are identical $f\equiv g$ if and only if their characteristic polynomials are identical: $\tau(f)\equiv\tau(g)$.

A Boolean function is satisfiable if it is not identically zero. The satisfiability problem is known to be NP-hard [12]. Since for any $\boldsymbol{x}\in\{0,1\}^n$ and for any $\boldsymbol{X}$ with $0<X_i<1$ for all i we have $E_{\boldsymbol{x}}(\boldsymbol{X})\neq 0$ it is obvious that

Proposition 4. *A Boolean function f is satisfiable if and only if for any $\boldsymbol{X}$ with $0<X_i<1$, $i=1,2,\ldots,n$, we have $\tau(f)(\boldsymbol{X})>0$.*

Therefore, the satisfiability problem can be reduced to one evaluation of the characteristic polynomial. This yields

Corollary 1. *Given a Boolean function f and an arbitrary rational vector $\boldsymbol{X}\in(0,1)^n$, an evaluation of the characteristic polynomial $\tau(f)$ at $\boldsymbol{X}$ is* NP-*hard.*

In view of Corollary 1, it is very unlikely that we can find an efficient algorithm for computing the values of characteristic polynomials. Therefore it is natural to discuss various approximation techniques for computing them. This will be done in the subsequent sections.

3. Worst case setting

In this section we deal with approximate computation of the characteristic polynomials in the worst case setting. More precisely, we approximate the values of the characteristic polynomial $F(\boldsymbol{X}) = \tau(f)(\boldsymbol{X})$ for an arbitrary Boolean function f and for an arbitrary vector $\boldsymbol{X}$ from the unit cube $[0,1]^n$.

Let $\mathscr{F}_n$ be the class of all Boolean functions $f : \{0,1\}^n \to \{0,1\}$. We assume that we can evaluate a Boolean function f at any point. Let

$$N(f,\boldsymbol{X}) = [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_k), \boldsymbol{X}], \quad f \in \mathscr{F}_n \tag{7}$$

be the information on the Boolean function f, and k is the cardinality of the information.

We consider two classes of information. The first class I_k^{res} is the class of restricted information which is defined as the set of all N of form (7) with the sample points $\boldsymbol{x}_i$ which are independent of $\boldsymbol{X}$. That is, the same points $\boldsymbol{x}_i$ are used for approximating the characteristic polynomial $\tau(f)(\boldsymbol{X})$ for *all* $\boldsymbol{X} \in [0,1]^n$. The second class I_k^{unr} is the class of unrestricted information which is defined as the set of all N of form (7) with the sample points $\boldsymbol{x}_i$ in (7) which *may* depend on $\boldsymbol{X}$, $\boldsymbol{x}_i = \boldsymbol{x}_i(\boldsymbol{X})$. This means that we can approximate $\tau(f)(\boldsymbol{X})$ by using different sample points for varying $\boldsymbol{X}$.

The information N is *nonadaptive* if and only if the sample points $\boldsymbol{x}_i$ are given a priori and they do not depend on f. For *adaptive* information, the choice of $\boldsymbol{x}_i$ may depend on the already computed $f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_{i-1})$. For the class I_k^{res} we write this as

$$\boldsymbol{x}_i = \boldsymbol{x}_i(f) = \boldsymbol{x}_i(f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_{i-1}))$$

and for the class I_k^{unr} as

$$\boldsymbol{x}_i = \boldsymbol{x}_i(f,\boldsymbol{X}) = \boldsymbol{x}_i(f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_{i-1}), \boldsymbol{X}).$$

In Sections 4 and 6 where the average case and randomized settings are considered, we will also allow the number k of Boolean evaluations to vary adaptively with f and $\boldsymbol{X}$.

By an algorithm we mean a mapping $\phi : N(\mathscr{F}_n \times [0,1]^n) \to \mathbb{R}$ and $\phi(N(f,\boldsymbol{X}))$ is regarded as an approximation to $\tau(f)(\boldsymbol{X})$. We have two classes of algorithms, depending on the information they used. The class Φ_k^{res} is the class of all algorithms that use restricted information from I_k^{res}, whereas the class Φ_k^{unr} is the class of all algorithms that use unrestricted information from I_k^{unr}. For notational convenience, we shall sometimes denote

$$\phi(f,\boldsymbol{X}) = \phi(N(f,\boldsymbol{X})),$$

when information N is clear from the context.

The parameter k is the total number of Boolean function values used by the algorithm. Since f is defined over the set of cardinality 2^n we always assume that

$k \in [0, 2^n]$. The extreme values of k are, however, not interesting. Indeed, $k = 0$ means that we do not use samples of f and algorithms are independent of f and may only depend on $\boldsymbol{X}$. For $k = 2^n$, we may sample f at *all* Boolean points and in this case $\phi(f, \cdot) = \tau(f)$ is well defined. Our emphasis will be on k which are essentially less than 2^n.

We now present examples of two algorithms. The first one is the constant algorithm $\phi^{\mathrm{con}}(f, \boldsymbol{X}) = 0.5$ for all Boolean functions f and vectors X from $[0,1]^n$. Obviously, $\phi^{\mathrm{con}} \in \Phi_k^{\mathrm{res}}$, $\forall k$, and ϕ^{con} does not use samples of f. As we shall see, the constant algorithm has minimal error in the class Φ_k^{res} as long as $k < 2^n$. The second algorithm is denoted by ϕ_k^{unr}, belongs to the class Φ_k^{unr}, and, as we shall see, has minimal error in this class. To define ϕ_k^{unr} we proceed as follows. For a given vector $\boldsymbol{X} \in [0,1]^n$ consider the set

$$A(\boldsymbol{X}) = \{E_{\boldsymbol{x}}(\boldsymbol{X}) : \boldsymbol{x} \in \{0,1\}^n\}.$$

The set $A(\boldsymbol{X})$ consists of 2^n numbers each of them from $[0,1]$. We order the elements of $A(\boldsymbol{X})$ from the largest to the smallest. That is, we define $\boldsymbol{x}_i = \boldsymbol{x}_i(\boldsymbol{X})$ such that

$$E_{\boldsymbol{x}_1}(\boldsymbol{X}) \geqslant E_{\boldsymbol{x}_2}(\boldsymbol{X}) \geqslant \cdots \geqslant E_{\boldsymbol{x}_{2^n}}(\boldsymbol{X}). \tag{8}$$

We stress that the ordering of $\boldsymbol{x}_i$ is, in general, not unique.

The algorithm ϕ_k^{unr} is defined by sampling the Boolean function f at the points $\boldsymbol{x}_i$ for $i = 1, 2, \ldots, k$ which correspond to the k largest $E_{\boldsymbol{x}}(\boldsymbol{X})$ and by assigning the values 0.5 for the value of f at unsampled points. That is,

$$\phi_k^{\mathrm{unr}}(f, \boldsymbol{X}) = \sum_{i=1}^{k} f(\boldsymbol{x}_i(\boldsymbol{X})) E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X}) + 0.5 \sum_{i=k+1}^{2^n} E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X}). \tag{9}$$

The second sum is of cardinality $2^n - k$, and from (3), we have

$$\phi_k^{\mathrm{unr}}(f, \boldsymbol{K}) = \sum_{i=1}^{k} f(\boldsymbol{x}_i(\boldsymbol{X})) E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X}) + 0.5 \sum_{i=1}^{k} (1 - E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})).$$

The last formula may be computationally more convenient especially when $k \ll 2^n$.

Note that the algorithm ϕ_k^{unr} has the same terms as the characteristic polynomial $\tau(f)$, and this is the case for terms which correspond to the sample points $\boldsymbol{x}_i$. The remaining terms correspond to the uncomputed values of f. Since now $f(\boldsymbol{x})$ is unknown it is replaced by 0.5 which is the mean of its two possible values 0 and 1.

We wish to find an algorithm ϕ from the class Φ_k^{res} or Φ_k^{unr} such that $\phi(N(f, \boldsymbol{X}))$ approximates $\tau(f)(\boldsymbol{X})$ with the smallest possible error. Obviously, the error depends on k and we would like to guarantee a small error even for a relatively small k.

There are many ways to define the error of approximation. In this section we consider the worst case setting in which the error is defined by a worst performance of the algorithm. In the next sections we relax the error criterion and consider the average case and randomized settings.

In the worst case setting we proceed as follows. By the worst case error of the algorithm ϕ we mean

$$e^{\mathrm{wor}}(\phi) = \sup_{f \in \mathcal{F}_n} \sup_{\boldsymbol{X} \in [0,1]^n} |\tau(f)(\boldsymbol{X}) - \phi(N(f,\boldsymbol{X}))|. \tag{10}$$

By

$$e^{\mathrm{wor}}(\Phi_k) = \inf_{\phi \in \Phi_k} e^{\mathrm{wor}}(\phi),$$

we denote the minimal worst case error of algorithms from the class $\Phi_k = \Phi_k^{\mathrm{res}}$ or from $\Phi_k = \Phi_k^{\mathrm{unr}}$. We are ready to prove

Theorem 1.

$$e^{\mathrm{wor}}(\Phi_k^{\mathrm{res}}) = e^{\mathrm{wor}}(\Phi_0^{\mathrm{res}}) = 0.5 \quad \textit{for } k \in [0, 2^n - 1],$$

$$e^{\mathrm{wor}}(\Phi_k^{\mathrm{unr}}) = e^{\mathrm{wor}}(\Phi_0^{\mathrm{unr}}) = e^{\mathrm{wor}}(\Phi_0^{\mathrm{unr}})(1 - k\,2^{-n}) = 0.5(1 - k\,2^{-n}).$$

Let us first comment on this theorem before proving it. For the class Φ_k^{res}, the minimal error is 0.5 and it is achieved without any sampling by the constant algorithm. This holds even for $k = 2^n - 1$ in which case we may know f except for only one Boolean point. This shows that k samples are useless for the class Φ_k^{res} as long as $k < 2^n$. This bad property may indicate that the class Φ_k^{res} is too restrictive.

For the class Φ_k^{unr}, the algorithm ϕ_k^{unr} has minimal error. In particular, this means that the information

$$N(f,\boldsymbol{X}) = [f(\boldsymbol{x}_1(\boldsymbol{X})), f(\boldsymbol{x}_2(\boldsymbol{X})), \ldots, f(\boldsymbol{x}_k(\boldsymbol{X})), \boldsymbol{X}]$$

is optimal. The sample points $\boldsymbol{x}_i(\boldsymbol{X})$ vary with $\boldsymbol{X}$. It is quite natural since for some $\boldsymbol{X}$ the problem of computing $\tau(f)(\boldsymbol{X})$ is easy. Indeed, take $\boldsymbol{X} = 0$. Then $\tau(f)(0) = f(0)$ can be computed even exactly with one Boolean function value if we select $\boldsymbol{x}_1 = 0$. On the other hand, if we take $\boldsymbol{X}^* = [0.5, 0.5, \ldots, 0.5]$ then all $E_{\boldsymbol{x}}(\boldsymbol{X}^*) = 2^{-n}$ and the selection of the sample points $\boldsymbol{x}_i$ is arbitrary. As we shall see in the proof the vector $\boldsymbol{X}^*$ is the most difficult one for evaluating the characteristic polynomials.

The most important part of Theorem 1 is the formula for the minimal error in the class Φ_k^{unr}. This formula states that as long as k is small relative to 2^n, the minimal error is large and roughly equal to 0.5. This is a very bad property. The error 0.5 can be obtained without computing function values since we know a priori that $\tau(f)(\boldsymbol{X}) \in [0,1]$. Hence, to improve the quality of the initial error we must take k of order 2^n, that is, we must take an exponentially large k. To illustrate this point further assume that we want to reduce the initial error by a factor of ε, that is, $e^{\mathrm{wor}}(\Phi_k^{\mathrm{unr}}) \leqslant \varepsilon e^{\mathrm{wor}}(\Phi_0^{\mathrm{unr}})$, where $\varepsilon \in [0,1)$. Then the minimal k for which we can achieve this is

$$k = \lceil (1 - \varepsilon) 2^n \rceil.$$

So even for a modest $\varepsilon = 0.5$ we must compute $k = 2^{n-1}$ Boolean function values.

The exponential dependence on n means that the problem of approximate evaluations of characteristic polynomials in the worst case setting is *intractable*. Hence, NP-hardness of this problem cannot be broken by approximation in the worst case setting.

Proof. Consider first the class Φ_k^{res}. Take an arbitrary algorithm ϕ from Φ_k^{res}. Assume that $\boldsymbol{x}_1$ is its first sample and that $f(\boldsymbol{x}_1)=0$. Based on this first value, the second sample $\boldsymbol{x}_2$ is chosen, and assume again that $f(\boldsymbol{x}_2)=0$. In general, assume that for all chosen $\boldsymbol{x}_i$ we obtain $f(\boldsymbol{x}_i)=0$ for $i=1,2,\ldots,k$. For all Boolean functions f for which $f(\boldsymbol{x}_i)=0$, $i\leqslant k$, the algorithm $\phi(f,\boldsymbol{X})$ must give the same approximation which may only depend on the vector $\boldsymbol{X}$, that is, $\phi(f,\boldsymbol{X})=a(\boldsymbol{X})$ for some function a. Then we have

$$\tau(f)-\phi(f,\boldsymbol{X})=\sum_{\boldsymbol{x}\neq\boldsymbol{x}_i} f(\boldsymbol{x})E_{\boldsymbol{x}}(\boldsymbol{X})-a(\boldsymbol{X}).$$

Note that the last sum has $2^n-k>0$ terms. Take now $\boldsymbol{X}=\boldsymbol{x}$ where $\boldsymbol{x}$ is any Boolean point different from all $\boldsymbol{x}_i$'s. Then (5) yields

$$\tau(f)-\phi(f,\boldsymbol{X})=f(\boldsymbol{x})-a(\boldsymbol{x}).$$

Since $f(\boldsymbol{x})$ could be zero or one, and $a(\boldsymbol{x})$ does not depend on the value of $f(\boldsymbol{x})$, we have

$$e^{\mathrm{wor}}(\phi)\geqslant \max(|a(\boldsymbol{x})|,|1-a(\boldsymbol{x})|)\geqslant 0.5.$$

For the constant algorithm ϕ^{con} we have $|\tau(f)(\boldsymbol{X})-0.5|\leqslant 0.5$ since $\tau(f)(\boldsymbol{X})\in[0,1]$. This proves that ϕ^{con} has minimal error in the class Φ_k^{res}.

We now consider the class Φ_k^{unr}. Let us first estimate from above the error of ϕ_k^{unr}. We have

$$\tau(f)(\boldsymbol{X})-\phi_k^{\mathrm{unr}}(f,\boldsymbol{X})=\sum_{i=k+1}^{2^n}(f(\boldsymbol{x}_i(\boldsymbol{X}))-0.5)E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X}).$$

Since $|f(\boldsymbol{x})-0.5|=0.5$ for all f and $\boldsymbol{x}$ we obtain

$$|\tau(f)(\boldsymbol{X})-\phi_k^{\mathrm{unr}}(f,\boldsymbol{X})|\leqslant 0.5\sum_{i=k+1}^{2^n}E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})=0.5\left(1-\sum_{i=1}^{k}E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})\right),\qquad(11)$$

due to (3) with $p=1$. Let $a_i=E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})$. We now show that

$$\sum_{i=1}^{k}a_i\geqslant k2^{-n}.\qquad(12)$$

Indeed, since $\sum_{i=1}^{2^n} a_i = 1$ and a_i are the k largest numbers among $E_{\boldsymbol{x}(\boldsymbol{X})}(\boldsymbol{X})$ we have $E_{\boldsymbol{x}(\boldsymbol{X})}(\boldsymbol{X}) \leqslant k^{-1} \sum_{i=1}^{k} a_i$ for all $\boldsymbol{x}$ distinct from $\boldsymbol{x}_i(\boldsymbol{X})$'s. Therefore

$$1 = \sum_{\boldsymbol{x}\in\{0,1\}^n} E_{\boldsymbol{x}}(\boldsymbol{X}) \leqslant \sum_{i=1}^{k} a_i + (2^n - k)k^{-1} \sum_{i=1}^{k} a_i = 2^n k^{-1} \sum_{i=1}^{k} a_i,$$

as claimed in (12). This also proves that

$$e^{\mathrm{wor}}(\phi_k^{\mathrm{unr}}) \leqslant 0.5(1 - k\,2^{-n}).$$

We now show a lower bound on the error of an arbitrary algorithm ϕ. We will do it for $\boldsymbol{X}^* = [0.5, 0.5, \ldots, 0.5]$. Then $E_{\boldsymbol{x}}(\boldsymbol{X}^*) = 2^{-n}$ for all $\boldsymbol{x}$. Suppose the algorithm ϕ uses $\boldsymbol{y}_1$ as its first sample point. Assume that the computed $f(\boldsymbol{y}_1) = 0$. Similarly, any time the algorithm ϕ uses $\boldsymbol{y}_i$ we assume that $f(\boldsymbol{y}_i) = 0$. After k Boolean function values we learn that f vanishes at these k points. In the rest of$2^n - k$ Boolean points the function can be zero or one. Therefore, the value $\tau(f)(\boldsymbol{X}^*)$ can be zero or $(2^n - k)2^{-n} = 1 - k\,2^{-n}$. The best the algorithm ϕ can do is to take the mid-point of zero and $1 - k\,2^{-n}$ as its approximation. This leads to the error at least $0.5(1 - k\,2^{-n})$, and completes the proof. □

Remark 1. The algorithm ϕ_k^{unr} enjoys even a stronger optimality property. Namely, ϕ_k^{unr} minimizes the worst case error for *each* vector $\boldsymbol{X}$, i.e.,

$$\min_{\phi} \max_{f\in\mathscr{F}_n} |\tau(f)(\boldsymbol{X}) - \phi(f,\boldsymbol{X})| = \max_{f\in\mathscr{F}_n} |\tau(f)(\boldsymbol{X}) - \phi_k^{\mathrm{unr}}(f,\boldsymbol{X})|$$
$$= 0.5\left(1 - \sum_{i=1}^{k} E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})\right).$$

Due to (11) it is enough to show that for any algorithm ϕ we have

$$\max_{f\in\mathscr{F}_n} |\tau(f)(\boldsymbol{X}) - \phi(f,\boldsymbol{X})| \geqslant 0.5\left(1 - \sum_{i=1}^{k} E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})\right).$$

Indeed, let $\boldsymbol{x}_i(f,\boldsymbol{X})$, $i = 1,\ldots,k$, be the sample points used by the algorithm ϕ. Here, the dependence on f is through the use of adaption, and the dependence on $\boldsymbol{X}$ is only present for the class Φ_k^{unr}. Define the operator $T : \mathscr{F}_n \to \mathscr{F}_n$ given by

$$Tf(\boldsymbol{x}) = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \{\boldsymbol{x}_1(f,\boldsymbol{X}),\ldots,\boldsymbol{x}_k(f,\boldsymbol{X})\},$$
$$Tf(\boldsymbol{x}) = 1 - f(\boldsymbol{x}) \quad \text{otherwise.}$$

It is easy to check that T is one-to-one. The algorithm ϕ gives the same approximation for f and Tf at the vector $\boldsymbol{X}$ since the two Boolean functions are indistinguishable at the sample points $\boldsymbol{x}_i(f,\boldsymbol{X}) = \boldsymbol{x}_i(Tf,\boldsymbol{X})$, $i = 1,\ldots,k$. Hence, we have

$$\max_{f\in\mathscr{F}_n} |\tau(f)(\boldsymbol{X}) - \phi(f,\boldsymbol{X})| \geqslant 0.5 \max_{f\in\mathscr{F}_n} (|\tau(f)(\boldsymbol{X}) - \phi(f,\boldsymbol{X})| + |\tau(Tf)(\boldsymbol{X})$$
$$-\phi(f,\boldsymbol{X})|)$$

$$\geqslant 0.5 \max_{f\in\mathscr{F}_n} |\tau(f)(\boldsymbol{X}) - \tau(Tf)(\boldsymbol{X})|$$

$$= \max_{f\in\mathscr{F}_n} \left| \sum_{\boldsymbol{x}\neq \boldsymbol{x}_i(f,\boldsymbol{X})} (f(\boldsymbol{x}) - 0.5) E_{\boldsymbol{x}}(\boldsymbol{X}) \right|.$$

Taking $f \equiv 1$ we have

$$\max_{f\in\mathscr{F}_n} |\tau(f)(\boldsymbol{X}) - \phi(f,\boldsymbol{X})| \geqslant 0.5 \sum_{\boldsymbol{x}\neq \boldsymbol{x}_i(1,\boldsymbol{X})} E_{\boldsymbol{x}}(\boldsymbol{X})$$

$$= 0.5 \left(1 - \sum_{i=1}^{k} E_{\boldsymbol{x}_i(1,\boldsymbol{X})}(\boldsymbol{X})\right)$$

$$\geqslant 0.5 \left(1 - \sum_{i=1}^{k} E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})\right),$$

as claimed. □

We now check whether we can obtain different results than those in Theorem 1 by using a different error norm in (10). Observe that in (10) we use the worst case error in the sense of the L_∞-norm. It is natural to relax the error criterion by switching to the L_p-norm. That is, define

$$e_p^{\mathrm{wor}}(\phi) = \sup_{f\in\mathscr{F}_n} \left(\int_{[0,1]^n} |\tau(f)(\boldsymbol{X}) - \phi(N(f,\boldsymbol{X}))|^p \,\mathrm{d}\boldsymbol{X} \right)^{1/p}, \quad 1 \leqslant p < \infty. \tag{13}$$

Since we are going to prove a negative result, we consider only the larger class Φ_k^{unr}. In this class, as we shall see, the algorithm ϕ_k^{unr} remains optimal for any p. Unfortunately, its L_p error is still close to 0.5 if k is not exponentially large in n. Thus, the choice of the L_p-norm does not help and we still have intractability.

Theorem 2.

$$\inf_{\phi\in\Phi_k^{\mathrm{unr}}} e_p^{\mathrm{wor}}(\phi) = e_p^{\mathrm{wor}}(\phi_k^{\mathrm{unr}}) \geqslant e_1^{\mathrm{wor}}(\phi_k^{\mathrm{unr}}) \geqslant 0.5 \left(1 - k \left(\frac{3}{4}\right)^n\right). \tag{14}$$

Proof. We first prove optimality of the algorithm ϕ_k^{unr}. Let $Z_i(\boldsymbol{X}) = E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})$. We have

$$e_p^{\mathrm{wor}}(\phi_k^{\mathrm{unr}})^p = \sup_{f\in\mathscr{F}_n} \int_{[0,1]^n} \left| \sum_{i=k+1}^{2^n} (f(\boldsymbol{X}_i(\boldsymbol{X})) - 0.5) Z_i(\boldsymbol{X}) \right|^p \,\mathrm{d}\boldsymbol{X}.$$

Since $|f(\boldsymbol{x}) - 0.5| = 0.5$, $\forall\, \boldsymbol{x} \in \{0,1\}^n$, we have

$$e_p^{\mathrm{wor}}(\phi_k^{\mathrm{unr}})^p \leqslant 2^{-p} \int_{[0,1]^n} \left| \sum_{i=k+1}^{2^n} Z_i(\boldsymbol{x}) \right|^p \mathrm{d}\boldsymbol{X} = 2^{-p} \int_{[0,1]^n} \left| 1 - \sum_{i=1}^{k} Z_i(\boldsymbol{X} \right|^p \mathrm{d}\boldsymbol{X}.$$

To conclude optimality of ϕ_k^{unr}, it is enough to show that for any algorithm ϕ from the class Φ_k^{unr} we have

$$e_p^{\mathrm{wor}}(\phi)^p \geqslant 2^{-p} \int_{[0,1]^n} \left| 1 - \sum_{i=1}^{k} Z_i(\boldsymbol{X}) \right|^p \mathrm{d}\boldsymbol{X}.$$

As in Theorem 1, assume that $f(\boldsymbol{y}_i) = 0$ or all the sample points $\boldsymbol{y}_i = \boldsymbol{y}_i(\boldsymbol{X})$, $i = 1, 2, \ldots, k$, used by ϕ. Then f can be equal to $f_1 \equiv 0$ or f can be equal to f_2 which takes 1 at all Boolean points different from $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_k$. For these two functions, the algorithm ϕ gives the same approximation, $\phi(f_1, \boldsymbol{X}) = \phi(f_2, \boldsymbol{X}) = a(\boldsymbol{X})$. Note that $\tau(f_1)(\boldsymbol{X}) = 0$ and $\tau(f_2)(\boldsymbol{X}) = \sum_{\boldsymbol{x} \neq \boldsymbol{y}_i} E_{\boldsymbol{x}}(\boldsymbol{X})$. Then

$$e_p^{\mathrm{wor}}(\phi)^p \geqslant 0.5 \int_{[0,1]^n} \left(|a(\boldsymbol{X})|^p + \left| \sum_{\boldsymbol{x} \neq \boldsymbol{y}_i} E_{\boldsymbol{x}}(\boldsymbol{X}) - a(\boldsymbol{X}) \right|^p \right) \mathrm{d}\boldsymbol{X}.$$

Since $|\alpha + \beta|^p \leqslant 2^{p-1}(|\alpha|^p + |\beta|^p)$ for any α and β, we obtain

$$\begin{aligned} e_p^{\mathrm{wor}}(\phi)^p &\geqslant 2^{-p} \int_{[0,1]^n} \left| \sum_{\boldsymbol{x} \neq \boldsymbol{y}_i} E_{\boldsymbol{x}}(\boldsymbol{X}) \right|^p \mathrm{d}\boldsymbol{X} \\ &= 2^{-p} \int_{[0,1]^n} \left| 1 - \sum_{i=1}^{k} E_{\boldsymbol{y}_i}(\boldsymbol{X}) \right|^p \mathrm{d}\boldsymbol{X} \\ &\geqslant 2^{-p} \int_{[0,1]^n} \left| 1 - \sum_{i=1}^{k} Z_i(\boldsymbol{X}) \right|^p \mathrm{d}\boldsymbol{X}, \end{aligned}$$

as claimed.

We now estimate the error of ϕ_k^{unr}. Clearly, the standard use of Hölder's inequality yields that $e_p^{\mathrm{wor}}(\phi_k^{\mathrm{unr}}) \geqslant e_1^{\mathrm{wor}}(\phi_k^{\mathrm{unr}})$ (obviously this holds for any algorithm). Hence, it is enough to take $p = 1$ and find a lower bound on $e_1^{\mathrm{wor}}(\phi_k^{\mathrm{unr}})$,

$$e_1^{\mathrm{wor}}(\phi_k^{\mathrm{unr}}) = 0.5 \int_{[0,1]^n} \left(1 - \sum_{i=1}^{k} E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X}) \right) \mathrm{d}\boldsymbol{X}.$$

Let $I_0 = [0, 0.5]$ and $I_1 = (0.5, 1]$. For any $\boldsymbol{x} = [x_1, x_2, \ldots, x_n] \in \{0,1\}^n$ define

$$I_{\boldsymbol{x}} = I_{x_1} \times I_{x_2} \times \cdots \times I_{x_n}.$$

Clearly, $I_{\boldsymbol{x}}$'s are disjoint subsets of $[0,1]^n$, whose union is $[0,1]^n$ and each of them has Lebesgue measure 2^{-n}. Hence,

$$\int_{[0,1]^n} |\tau(0)(\boldsymbol{X}) - \phi_k^{\mathrm{unr}}(0, \boldsymbol{X})| \, \mathrm{d}\boldsymbol{X} = 0.5 \sum_{\boldsymbol{x} \in \{0,1\}^n} \int_{I_{\boldsymbol{x}}} \left(1 - \sum_{i=1}^{k} E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X}) \right) \mathrm{d}\boldsymbol{X}.$$

It is easy to observe that for $\boldsymbol{X} \in I_{\boldsymbol{x}}$ we have

$$E_{\boldsymbol{x}}(\boldsymbol{X}) \geqslant E_{\boldsymbol{y}}(\boldsymbol{X}), \quad \forall \boldsymbol{y} \in \{0,1\}^n.$$

Indeed, if $x_i = 0$ then $X_i \in I_{x_i} = [0, 0.5]$ and

$$X_i^{x_i}(1-X_i)^{1-x_i} = 1 - X_i = \max(X_i, 1-X_i) \geqslant X_i^{y_i}(1-X_i)^{1-y_i}.$$

Similarly, if $x_i = 1$ then $X_i \in I_{x_i} = (0.5, 1]$ and

$$X_i^{x_i}(1-X_i)^{1-x_i} = X_i = \max(X_i, 1-X_i) \geqslant X_i^{y_i}(1-X_i)^{1-y_i}.$$

Hence, for any $\boldsymbol{X} \in I_{\boldsymbol{x}}$ and any $\boldsymbol{y} \in \{0,1\}^n$ we have

$$E_{\boldsymbol{x}}(\boldsymbol{X}) = \prod_{i=1}^{n} \max(X_i, 1-X_i) \geqslant \prod_{i=1}^{n} X_i^{y_i}(1-X_i)^{1-y_i} = E_{\boldsymbol{y}}(\boldsymbol{X}), \tag{15}$$

as claimed. This yields that

$$\int_{I_{\boldsymbol{x}}} E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X} \leqslant \int_{I_{\boldsymbol{x}}} E_{\boldsymbol{x}}(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X} = \prod_{i=1}^{n} \int_{I_{x_i}} \max(X, 1-X)\,\mathrm{d}X = \left(\frac{3}{8}\right)^n, \tag{16}$$

since $\int_0^{0.5} \max(X, 1-X)\,\mathrm{d}X = \int_{0.5}^1 \max(X, 1-X)\,\mathrm{d}X = \frac{3}{8}$.

From this we conclude

$$\int_{[0,1]^n} |\tau(0)(\boldsymbol{X}) - \phi_k^{\mathrm{unr}}(0, \boldsymbol{X})|\,\mathrm{d}\boldsymbol{X} = 0.5\left(1 - \sum_{\boldsymbol{x} \in \{0,1\}^n} \sum_{i=1}^{k} \int_{I_{\boldsymbol{x}}} E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X}\right)$$

$$\geqslant 0.5\left(1 - k\,2^n \left(\frac{3}{8}\right)^n\right) = 0.5\left(1 - k\left(\frac{3}{4}\right)^n\right),$$

as claimed. This completes the proof. □

4. Average case setting

In this section we study the average case setting in which the cardinality of information as well as the error of an algorithm is defined by its average behavior. To simplify further calculations we only analyze the average case cardinality of information in the L_1 sense and the average case error in the L_2 sense.

We need to define a measure on the space $\mathscr{F}_n$ of Boolean functions. The cardinality of $\mathscr{F}_n$ is 2^{2^n} since each Boolean function from $\mathscr{F}_n$ is defined on 2^n points and can take two different values at each of these points. It is natural to assume that all Boolean functions from $\mathscr{F}_n$ are equally probable and occur with probability 2^{-2^n}. Hence, we assign a uniform measure μ on the space $\mathscr{F}_n$, and for any $A \subset \mathscr{F}_n$ we have $\mu(A) = 2^{-2^n}|A|$, where $|A|$ denotes the cardinality of the set A.

In the average case setting we generalize the notion of information N in (7) by allowing k to vary adaptively with f and $\boldsymbol{X}$. More precisely, we assume that

$$N(f,\boldsymbol{X}) = [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_{k(f,\boldsymbol{X})}), \boldsymbol{X}],$$

where the sample points $\boldsymbol{x}_i$'s are given as in Section 3, and $k(f,\boldsymbol{X})$ is defined as in [24]. The idea is to capture the notion that we may terminate depending on the computed data and $k(f,\boldsymbol{X})$ should denote the local cardinality of information for the Boolean function f. Formally, this is defined by termination functions

$$\mathrm{ter}_i\colon \{0,1\}^i \times [0,1]^n \to \{0,1\} \quad \text{for } i = 1,2,\ldots,2^n.$$

We assume that $\mathrm{ter}_{2^n} \equiv 1$. Then

$$k(f,\boldsymbol{X}) = \min\{i\colon \mathrm{ter}_i(f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_i), \boldsymbol{X}) = 1\}.$$

That is, we terminate the computation of information for the minimal i for which the termination criterion $\mathrm{ter}_i(f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_i), \boldsymbol{X}) = 1$ holds. Clearly, $k(f,\boldsymbol{X})$ is well defined and we always have $k(f,\boldsymbol{X}) \leqslant 2^n$. Let

$$A_i(\boldsymbol{X}) = \{f \in \mathscr{F}_n\colon k(f,\boldsymbol{X}) = i\}. \tag{17}$$

Then $\mu(A_i(\boldsymbol{X})) = 2^{-2^n}|A_i(\boldsymbol{X})|$ is the measure of the set of functions for which we compute exactly i Boolean function values for the vector $\boldsymbol{X}$. We define the average cardinality of N by the expected value of $k(f,\boldsymbol{X})$ in the L_1 sense,

$$k(N) = 2^{-2^n} \sum_{f\in\mathscr{F}_n} \int_{[0,1]^n} k(f,\boldsymbol{X})\,\mathrm{d}\boldsymbol{X} = \sum_{j=0}^{2^n} \mathrm{j} \int_{[0,1]^n} \mu(A_j(\boldsymbol{X}))\,\mathrm{d}\boldsymbol{X}. \tag{18}$$

Note that $k(N)$ is not necessarily an integer. For a given integer $k \in [0,2^n]$, we define the classes of information I_k^{res} and I_k^{unr} as well as the classes of algorithms Φ_k^{res} and Φ_k^{unr} analogously as in Section 3. In particular, I_k^{res} is the class of all information of the average cardinality at most k and for which the sample points as well as $k(f,\boldsymbol{X})$ and $A_j(\boldsymbol{X})$ do not depend on $\boldsymbol{X}$. This means that

$$k(N) = \sum_{j=0}^{2^n} \mathrm{j}\mu(A_j) \leqslant k. \tag{19}$$

For the class I_k^{unr} we have the same bound on the average cardinality but the sample points as well as $k(f,\boldsymbol{X})$ and $A_j(\boldsymbol{X})$ may depend on $\boldsymbol{X}$.

The average case error of the algorithm ϕ is defined as

$$e^{\mathrm{avg}}(\phi) = \left(2^{-2^n} \sum_{f\in\mathscr{F}_n} \int_{[0,1]^n} |\tau(f)(\boldsymbol{X}) - \phi(N(f,\boldsymbol{X}))|^2\,\mathrm{d}\boldsymbol{X}\right)^{1/2}. \tag{20}$$

By

$$e^{\mathrm{avg}}(\Phi_k) = \inf_{\phi\in\Phi_k} e^{\mathrm{avg}}(\phi), \tag{21}$$

we denote the minimal average case error of algorithms from the class $\Phi_k = \Phi_k^{\mathrm{res}}$ or from the class $\Phi_k = \Phi_k^{\mathrm{unr}}$.

Before we present optimal algorithms and estimate the errors, we point out that the constant algorithm $\phi^{\mathrm{con}} \equiv 0.5$ has minimal error if we do not sample f, i.e., $k = 0$, and

$$e^{\mathrm{avg}}(\Phi_0^{\mathrm{res}}) = e^{\mathrm{avg}}(\Phi_0^{\mathrm{unr}}) = e^{\mathrm{avg}}(\phi^{\mathrm{con}}) = 0.5\left(\frac{2}{3}\right)^{n/2}, \tag{22}$$

which will follow from Theorem 3 below with $k = 0$. Hence, even without sampling we have an exponentially small (in n) average error. This is, of course, an artifact of the L_2-norm used in the definition of the average case error. The characteristic polynomials have values in $[0, 1]$ and their deviation from 0.5 in the L_2-norm is exponentially small as a function of n. Therefore, when we consider the minimal errors after k samples we compare them to the minimal error achieved for $k = 0$. We will be interested in how fast the ratio $e^{\mathrm{avg}}(\Phi_k)/e^{\mathrm{avg}}(\Phi_0)$ goes to zero as k increases. As in the worst case setting, we prove that this ratio decreases exponentially slowly and is virtually constant for any k which is polynomial in n.

The constant algorithm ϕ^{con} is not optimal in the class Φ_k^{res} for $k > 0$. We shall see that the following algorithm ϕ_k^{res} is optimal. Take k arbitrary Boolean points $\boldsymbol{x}_i$ (independent of the vectors $\boldsymbol{X}$) compute $f(\boldsymbol{x}_i)$, assign 0.5 for the unsample points and define

$$\phi_k^{\mathrm{res}}(f, \boldsymbol{X}) = \sum_{i=1}^{k} f(\boldsymbol{x}_i) E_{\boldsymbol{x}_i}(\boldsymbol{X}) + 0.5\left(1 - \sum_{i=1}^{k} E_{\boldsymbol{x}_i}(\boldsymbol{X})\right). \tag{23}$$

Hence, the only difference between the algorithms ϕ_k^{res} and ϕ_k^{unr} is in that the sample points of ϕ_k^{unr} depend on $\boldsymbol{X}$, see (9). We are ready to prove

Theorem 3.

$$e^{\mathrm{avg}}(\Phi_k^{\mathrm{res}}) = e^{\mathrm{avg}}(\phi_k^{\mathrm{res}}) = e^{\mathrm{avg}}(\Phi_0^{\mathrm{res}})\sqrt{1 - k\,2^{-n}},$$

$$e^{\mathrm{avg}}(\Phi_k^{\mathrm{unr}}) = e^{\mathrm{avg}}(\phi_k^{\mathrm{unr}}) = e^{\mathrm{avg}}(\Phi_0^{\mathrm{unr}})\sqrt{\max(0, 1 - k a_n^{-n})},$$

where $a_n \in [\frac{8}{7}, 2]$.

Theorem 3 states that for the class Φ_k^{res} the algorithm ϕ_k^{res} is optimal. Its average case error is almost the same as the minimal error without sampling as long as k is small relative to 2^n. For the class Φ_k^{unr}, the algorithm ϕ_k^{unr} which is optimal in the worst case setting remains optimal also in the average case setting. Its average case error is, as for the previous class, almost the same as the minimal error without sampling as long as k is small relative to $(8/7)^n$. Hence, if we want to reduce the initial error by a factor of ε, that is, $e^{\mathrm{avg}}(\Phi_k^{\mathrm{unr}}) \leqslant \varepsilon e^{\mathrm{avg}}(\Phi_0^{\mathrm{unr}})$, the number k of samples must be at least

$$k = \left\lceil \left(\frac{8}{7}\right)^n (1 - \varepsilon^2) \right\rceil.$$

Hence, k is exponential in n. This indicates that the problem remains intractable in the average case setting.

Proof. Take an arbitrary algorithm ϕ from the class Φ_k^{res} or Φ_k^{unr}. Let

$$\alpha_\phi(\boldsymbol{X}) = 2^{-2^n} \sum_{f \in \mathscr{F}_n} |\tau(f)(\boldsymbol{X}) - \phi(f, \boldsymbol{X})|^2. \tag{24}$$

Clearly,

$$e^{\mathrm{avg}}(\phi)^2 = \int_{[0,1]^n} \alpha_\phi(\boldsymbol{X})\, \mathrm{d}\boldsymbol{X}. \tag{25}$$

As in Remark 1, for a Boolean function f from $\mathscr{F}_n$ denote by $\boldsymbol{x}_i(f, \boldsymbol{X})$, $i = 1, 2, \ldots, k(f, \boldsymbol{X})$, the sample points used by the algorithm ϕ. Define the operator $T : \mathscr{F}_n \to \mathscr{F}_n$ given by

$$Tf(\boldsymbol{x}) = f(\boldsymbol{x}) \quad \boldsymbol{x} \in \{\boldsymbol{x}_1(f, \boldsymbol{X}), \ldots, \boldsymbol{x}_{k(f,\boldsymbol{X})}(f, \boldsymbol{X})\},$$
$$Tf(\boldsymbol{x}) = 1 - f(\boldsymbol{x}) \quad \text{otherwise.}$$

The operator T is one-to-one, and since we have a uniform measure the probability of Tf is the same as the probability of f and both are equal to 2^{-2^n}. Hence, we can rewrite (24) as

$$\alpha_\phi(\boldsymbol{X}) = 2^{-2^n} \sum_{f \in \mathscr{F}_n} (\tau(Tf)(\boldsymbol{X}) - \phi(Tf, \boldsymbol{X}))^2.$$

Since the algorithm ϕ gives the same approximation for f and Tf at the vector $\boldsymbol{X}$, we have

$$\alpha_\phi(\boldsymbol{X}) = 0.5 \cdot 2^{-2^n} \sum_{f \in \mathscr{F}_n} ((\tau(Tf)(\boldsymbol{X}) - \phi(f, \boldsymbol{X}))^2 + (\tau(f)(\boldsymbol{X}) - \phi(f, \boldsymbol{X}))^2).$$

Since $|a-b|^2 + |b-c|^2 \geqslant 0.5|a-c|^2$ for any real numbers a, b, c, by taking

$$a = \tau(Tf)(\boldsymbol{X}), \quad b = \phi(f, \boldsymbol{X}), \quad c = \tau(f)(\boldsymbol{X})$$

we obtain

$$|a - c| = |\tau(Tf)(\boldsymbol{X}) - \tau(f)(\boldsymbol{X})| = 2 \left| \sum_{\boldsymbol{x} \neq \boldsymbol{x}_i(f, \boldsymbol{X})} (f(\boldsymbol{x}) - 0.5) E_{\boldsymbol{x}}(\boldsymbol{X}) \right|.$$

Therefore

$$\alpha_\phi(\boldsymbol{X}) \geqslant 2^{-2^n} \sum_{f \in \mathscr{F}_n} \alpha_f(\boldsymbol{X}), \tag{26}$$

where

$$\alpha_f(\boldsymbol{X}) = \left(\sum_{\boldsymbol{x} \neq \boldsymbol{x}_i(f,\boldsymbol{X})} (f(\boldsymbol{x}) - 0.5) E_{\boldsymbol{x}}(\boldsymbol{X}) \right)^2$$

$$= \sum_{\boldsymbol{x},\boldsymbol{y} \neq \boldsymbol{x}_i(f,\boldsymbol{X})} (f(\boldsymbol{x}) - 0.5)(f(\boldsymbol{y}) - 0.5) E_{\boldsymbol{x}}(\boldsymbol{X}) E_{\boldsymbol{y}}(\boldsymbol{X}). \tag{27}$$

Observe that for the algorithms ϕ_k^{res} and ϕ_k^{unr} we have equality in (26).

The sets $A_j = A_j(\boldsymbol{X})$ given by (17) are disjoint with respect to different j's. Since $\mathscr{F}_n = \bigcup_{j=0}^{2^n} A_j(\boldsymbol{X})$, we have

$$\sum_{j}^{2^n} \mu(A_j(\boldsymbol{X})) = 1, \quad \forall \boldsymbol{X} \in [0,1]^n. \tag{28}$$

Let $N(A_j(\boldsymbol{X}), \boldsymbol{X}) = \{z_{j,1}, \ldots, z_{j,s_j}\}$ denote the distinct information values for which we perform j Boolean function evaluations. Clearly, $z_{j,m} = z_{j,m}(\boldsymbol{X})$. Let $A_{j,m} = A_{j,m}(\boldsymbol{X}) = \{f \in A_j \colon N(f, \boldsymbol{X}) = z_{j,m}\}$. Then we have

$$\sum_{f \in \mathscr{F}_n} \alpha_f(\boldsymbol{X}) = \sum_{j=0}^{2^n} \sum_{f \in A_j} \alpha_f(\boldsymbol{X}) = \sum_{j=0}^{2^n} \sum_{m=1}^{s_j} \sum_{f \in A_{j,m}} \alpha_f(\boldsymbol{X}).$$

Let $\boldsymbol{x}_{i,j,m} = \boldsymbol{x}_{i,j,m}(\boldsymbol{X})$, $i = 1, 2, \ldots, j$, be the sample points used by N with $N(f, \boldsymbol{X}) = z_{j,m}$, and let $B_{j,m} = \{\boldsymbol{x} \in \{0,1\}^n : \boldsymbol{x} \neq \boldsymbol{x}_{i,j,m}\}$. Then we have

$$\sum_{f \in \mathscr{F}_n} \alpha_f(\boldsymbol{X}) = \sum_{j=0}^{2^n} \sum_{m=1}^{s_j} \sum_{f \in A_{j,m}} \sum_{\boldsymbol{x},\boldsymbol{y} \in B_{j,m}} (f(\boldsymbol{x}) - 0.5)(f(\boldsymbol{y}) - 0.5) E_{\boldsymbol{x}}(\boldsymbol{X}) E_{\boldsymbol{y}}(\boldsymbol{X})$$

$$= \sum_{j=0}^{2^n} \sum_{m=1}^{s_j} \sum_{\boldsymbol{x},\boldsymbol{y} \in B_{j,m}} E_{\boldsymbol{x}}(\boldsymbol{X}) E_{\boldsymbol{y}}(\boldsymbol{X}) \sum_{f \in A_{j,m}} (f(\boldsymbol{x}) - 0.5)(f(\boldsymbol{y}) - 0.5). \tag{29}$$

We now compute the last sum with respect to f. For $\boldsymbol{x} = \boldsymbol{y}$ we have $(f(\boldsymbol{x}) - 0.5)^2 = 0.25$ for all f and therefore

$$\sum_{f \in A_{j,m}} (f(\boldsymbol{x}) - 0.5)^2 = 0.25 \cdot |A_{j,m}|.$$

For $\boldsymbol{x} \neq \boldsymbol{y}$ we have

$$\sum_{f \in A_{j,m}} (f(\boldsymbol{x}) - 0.5)(f(\boldsymbol{y}) - 0.5) = 0.$$

Indeed, $(f(\boldsymbol{x})-0.5)(f(\boldsymbol{y})-0.5)$ takes the values 0.25 and -0.25 both with the same probability, and therefore its mean is zero. From this we can rewrite (29) as

$$\sum_{f\in\mathscr{F}_n}\alpha_f(\boldsymbol{X})=0.25\sum_{j=0}^{2^n}\sum_{m=1}^{s_j}\left(\sum_{\boldsymbol{x}\in\{0,1\}^n}E^2_{\boldsymbol{x}}(\boldsymbol{X})-\sum_{i=1}^{j}E^2_{\boldsymbol{x}_{i,j,m}}(\boldsymbol{X})\right)\cdot|A_{j,m}|. \tag{30}$$

We now consider the class Φ_k^{res}. The sample points $\boldsymbol{x}_{i,j,m}$ as well as the sets A_j and $A_{j,m}$ do not now depend on $\boldsymbol{X}$. Note that

$$\int_{[0,1]^n}E^2_{\boldsymbol{x}}(\boldsymbol{X})=\prod_{i=1}^{n}\int_0^1 X^{2x_i}(1-X)^{2(1-x_i)}\,\mathrm{d}X=\left(\frac{1}{3}\right)^n,\quad\forall\boldsymbol{x}\in\{0,1\}^n.$$

We now integrate (30) with respect to $\boldsymbol{X}$ and obtain

$$\begin{aligned}\int_{[0,1]^n}\sum_{f\in\mathscr{F}_n}\alpha_f(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X}&=0.25\sum_{j=0}^{2^n}\sum_{m=1}^{s_j}(2^n3^{-n}-j3^{-n})|A_{j,m}|\\&=0.25\left(\frac{2}{3}\right)^n\left(\sum_{j=0}^{2^n}\sum_{m=1}^{s_j}|A_{j,m}|-2^{-n}\sum_{j=0}^{2^n}j\sum_{m=1}^{s_j}|A_{j,m}|\right).\end{aligned}$$

Clearly,

$$\sum_{m=1}^{s_j}|A_{j,m}|=|A_j|=\mu(A_j)2^{2^n}.$$

From this we get

$$\int_{[0,1]^n}\sum_{f\in\mathscr{F}_n}\alpha_f(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X}=0.25\left(\frac{2}{3}\right)^n2^{2^n}\left(\sum_{j=0}^{2^n}\mu(A_j)-2^{-n}\sum_{j=0}^{2^n}j\mu(A_j)\right).$$

Since $\sum_{j=0}^{2^n}\mu(A_j)=1$ and $\sum_{j=0}^{2^n}j\mu(A_j)\leqslant k$, see (19), we finally have

$$\int_{[0,1]^n}\sum_{f\in\mathscr{F}_n}\alpha_f(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X}\geqslant0.25\left(\frac{2}{3}\right)^n2^{2^n}(1-k\,2^{-n}).$$

This, (25), and (26) yield

$$e^{\mathrm{avg}}(\phi)^2\geqslant0.25\,(\tfrac{2}{3})^n(1-k\,2^{-n}) \tag{31}$$

for any algorithm ϕ from the class Φ_k^{res}. Observe that for the algorithm ϕ_k^{res} we have equality in (31). This proves that ϕ_k^{res} is optimal and for $k=0$ we have $e^{\mathrm{avg}}(\Phi_0^{\mathrm{unr}})=0.5(2/3)^{n/2}$. This proves the first part of Theorem 3.

We now consider the class Φ_k^{unr} and return to (30). From (8) we have

$$\sum_{i=1}^{j}E^2_{\boldsymbol{x}_{i,j,m}}(\boldsymbol{X})\leqslant\sum_{i=1}^{j}E^2_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X}).$$

Thus we obtain

$$\sum_{f\in\mathscr{F}_n}\alpha_f(\boldsymbol{X}) \geqslant 0.25\sum_{j=0}^{2^n}\sum_{m=1}^{s_j}\left(\sum_{\boldsymbol{x}\in\{0,1\}^n}E^2_{\boldsymbol{x}}(\boldsymbol{X})-\sum_{i=1}^{j}E^2_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})\right)\cdot|A_{j,m}|$$

$$= 0.25\cdot 2^{2^n}\sum_{i=1}^{2^n}\mu(A_j(\boldsymbol{X}))\left(\sum_{\boldsymbol{x}\in\{0,1\}^n}E^2_{\boldsymbol{x}}(\boldsymbol{X})-\sum_{i=1}^{j}E^2_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})\right). \tag{32}$$

For the algorithm ϕ_k^{unr} we have equality in (32). We now show that the right-hand side of (32) is minimized when we terminate always with k Boolean function evaluations, i.e., $A_j=\emptyset$ for $j\neq k$ and $A_k=\mathscr{F}_n$, or, equivalently, $\mu(A_j(\boldsymbol{X}))=0$ for $j\neq k$ and $\mu(A_k(\boldsymbol{X})=1$. Let $r_j=\sum_{\boldsymbol{x}\in\{0,1\}^n}E^2_{\boldsymbol{x}}(\boldsymbol{X})-\sum_{i=1}^{j}E^2_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})$. Then we need to minimize

$$\sum_{j=1}^{2^n}\mu(A_j(\boldsymbol{X}))r_j$$

subject to the following constraints:

$$\sum_{j=1}^{2^n}\mu(A_j(\boldsymbol{X}))=1 \quad\text{and}\quad \sum_{j=1}^{2^n}j\mu(A_j(\boldsymbol{X}))\leqslant k.$$

Observe that $\{r_j\}$ is convex, i.e., $r_j\leqslant\frac{1}{2}(r_{j-1}+r_{j+1})$, $\forall j$. From [24] we know that the minimum is attained for $\mu(A_j(\boldsymbol{X})=\delta_{j,k}$. Hence

$$\sum_{f\in\mathscr{F}_n}\alpha_f(\boldsymbol{X})\geqslant 0.25\cdot 2^{2^n}\left(\sum_{\boldsymbol{x}\in\{0,1\}^n}E^2_{\boldsymbol{x}}(\boldsymbol{X})-\sum_{i=1}^{k}E^2_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})\right).$$

From this we have

$$\alpha_\phi(\boldsymbol{X})\geqslant 0.25\left(\sum_{\boldsymbol{x}\in\{0,1\}^n}E^2_{\boldsymbol{x}}(\boldsymbol{X})-\sum_{i=1}^{k}E^2_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})\right). \tag{33}$$

Once more for the algorithm ϕ_k^{unr} we have equality in (33). This means that the algorithm ϕ_k^{unr} is optimal for each $\boldsymbol{X}$. Obviously,

$$\alpha_\phi(\boldsymbol{X})\geqslant 0.25\left(\sum_{\boldsymbol{x}\in\{0,1\}^n}E^2_{\boldsymbol{x}}(\boldsymbol{X})-kE^2_{\boldsymbol{x}_1(\boldsymbol{X})}(\boldsymbol{X})\right). \tag{34}$$

We are ready to integrate (34) with respect to $\boldsymbol{X}$. We have

$$\int_{[0,1]^n}E^2_{\boldsymbol{x}}(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X}=\left(\frac{1}{3}\right)^n,\quad\forall\boldsymbol{x}\in\{0,1\}^n$$

and due to (15) we have

$$\int_{[0,1]^n} E^2_{\boldsymbol{x}_1(\boldsymbol{X})}(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X} = 2^n \int_{I_x} E^2_{\boldsymbol{x}}(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X} = 2^n \prod_{i=1}^{n} \int_0^{0.5} (1-X)^2\,\mathrm{d}X = \left(\frac{7}{12}\right)^n. \tag{35}$$

This yields that

$$e^{\mathrm{avg}}(\phi) \geqslant 0.5\sqrt{(\tfrac{2}{3})^n - k(\tfrac{7}{12})^n} = 0.5\,(\tfrac{2}{3})^{n/2}\sqrt{1-k(7/8)^n}$$

holds for any algorithm ϕ from the class Φ_k^{unr}. This proves that $a_n \geqslant 7/8$ in Theorem 3. We obviously have $a_n \leqslant 2$ since $\Phi_k^{\mathrm{unr}} \subset \Phi_k^{\mathrm{res}}$. This completes the proof. □

5. Fixed evaluation point

In Sections 3 and 4 we studied the problem of approximating the characteristic polynomials for *all* vectors $\boldsymbol{X}$ from $[0,1]^n$. In this section we relax the problem by approximating the characteristic polynomials at a *fixed* vector $\boldsymbol{X}$ from $[0,1]^n$.

As before, we consider algorithms that use k Boolean function values. Note, however, that the difference between two classes of information now disappears since $\boldsymbol{X}$ is fixed. The error of the algorithm ϕ that uses the information N is now defined:

in the *worst case* setting,

$$e^{\mathrm{wor}}(\phi,\boldsymbol{X}) = \sup_{f\in\mathscr{F}_n} |\tau(f)(\boldsymbol{X}) - \phi(N(f,\boldsymbol{X}))|$$

and in the *average case* setting

$$e^{\mathrm{avg}}(\phi,\boldsymbol{X}) = \left(2^{-2^n} \sum_{f\in\mathscr{F}_n} |\tau(f)(\boldsymbol{X}) - \phi(N(f,\boldsymbol{X}))|^2\right)^{1/2}.$$

Obviously, for some vectors $\boldsymbol{X}$, the worst case error can be small or even zero. Indeed, this holds for all boundary points of $[0,1]^n$ and for the algorithm $\phi = \phi_1^{\mathrm{unr}}$ defined by (9).

It is then natural to ask what is the Lebesgue measure λ of vectors $\boldsymbol{X}$ from $[0,1]^n$ for which the minimal error of algorithms using k Boolean functions reduces the initial error by β. Here $\beta \in (0,1)$, and the initial error is equal to the minimal error with $k=0$, i.e., without sampling of Boolean functions.

More precisely, let Φ_k be the class of algorithms that use at most k Boolean function values. In the average case setting, this means that algorithms use information with average cardinality at most k where the average cardinality is defined by (19).

Let

$$e_k^{\mathrm{wor}}(\boldsymbol{X}) = \min_{\phi \in \Phi_k} e^{\mathrm{wor}}(\phi, \boldsymbol{X}) \quad \text{and} \quad e_k^{\mathrm{avg}}(\boldsymbol{X}) = \min_{\phi \in \Phi_k} e^{\mathrm{avg}}(\phi, \boldsymbol{X})$$

be the minimal errors in the worst and average case settings.

Assume that $k = 0$. In the worst case setting we have $e_0^{\mathrm{wor}}(\boldsymbol{X}) = 0.5$ for all $\boldsymbol{X}$. In the average case setting, (33) and (3) yield

$$e_0^{\mathrm{avg}}(\boldsymbol{X}) = 0.5 \left(\sum_{\boldsymbol{x} \in \{0,1\}^n} E_{\boldsymbol{x}}^2(\boldsymbol{X}) \right)^{1/2} = 0.5 \prod_{i=1}^{n} (X_i^2 + (1 - X_i)^2)^{1/2}. \tag{36}$$

As already indicated in Remark 1 and in the proof of Theorem 3, the algorithm ϕ_k^{unr} minimizes the worst and average case error for each $\boldsymbol{X}$,

$$e_k^{\mathrm{wor}}(\boldsymbol{X}) = e^{\mathrm{wor}}(\phi_k^{\mathrm{unr}}, \boldsymbol{X}) \quad \text{and} \quad e_k^{\mathrm{avg}}(\boldsymbol{X}) = e^{\mathrm{avg}}(\phi_k^{\mathrm{unr}}, \boldsymbol{X}).$$

We are ready to prove

Theorem 4.

$$\lambda\{\boldsymbol{X} \in [0,1]^n\colon\ e_k^{\mathrm{wor}}(\boldsymbol{X}) \leqslant \beta e_0^{\mathrm{wor}}(\boldsymbol{X})\} \leqslant \frac{k}{1-\beta} \left(\frac{3}{4}\right)^n,$$

$$\lambda\{\boldsymbol{X} \in [0,1]^n\colon\ e_k^{\mathrm{avg}}(\boldsymbol{X}) \leqslant \beta e_0^{\mathrm{avg}}(\boldsymbol{X})\} \leqslant \frac{k}{1-\beta^2} \left(\frac{1+\ln 2}{2}\right)^n,$$

and $(1 + \ln 2)/2 = 0.8466\ldots$.

Theorem 4 states that the minimal kth error is large for most $\boldsymbol{X}$ since the measure of the set of $\boldsymbol{X}$ for which we reduce the initial error by β is exponentially small in n. Hence, k must be exponentially large in n if we want to guarantee that the measure of the set is relatively large.

Proof. As before, let $Z_i(\boldsymbol{X}) = E_{\boldsymbol{x}_i(\boldsymbol{X})}(\boldsymbol{X})$. Due to (3), the sum of $Z_i(\boldsymbol{X})$ is 1 and $Z_i(\boldsymbol{X})'$s are ordered: $Z_1(\boldsymbol{X}) \geqslant Z_2(\boldsymbol{X}) \geqslant \cdots$. In the worst case setting, the error of ϕ_k^{unr} is

$$e^{\mathrm{wor}}(\phi_k^{\mathrm{unr}}, \boldsymbol{X}) = 0.5 \left(1 - \sum_{i=1}^{k} Z_i(\boldsymbol{X})\right) \geqslant 0.5(1 - kZ_1(\boldsymbol{X})).$$

Let

$$\alpha := \lambda\{\boldsymbol{X} \in [0,1]^n\colon\ e_k^{\mathrm{wor}}(\boldsymbol{X}) \leqslant \beta e_0^{\mathrm{wor}}(\boldsymbol{X})\}.$$

Then $e_k^{\mathrm{wor}}(\boldsymbol{X}) = e^{\mathrm{wor}}(\phi_k^{\mathrm{unr}}, \boldsymbol{X})$ yields

$$\alpha = \lambda\left\{\boldsymbol{X} \in [0,1]^n\colon\ \sum_{i=1}^{k} Z_i(\boldsymbol{X}) \geqslant 1 - \beta\right\} \leqslant \lambda\{\boldsymbol{X} \in [0,1]^n\colon\ Z_1(\boldsymbol{X}) \geqslant (1-\beta)/k\}.$$

Applying Chebyshev's inequality [1] we conclude

$$\alpha \leqslant \frac{k}{1-\beta} \int_{[0,1]^n} Z_1(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X} \leqslant \frac{2^n k}{1-\beta} \int_{[0,0.5]^n} Z_1(\boldsymbol{X})\,\mathrm{d}\boldsymbol{X} = \frac{k}{1-\beta}\left(\frac{3}{4}\right)^n,$$

due to (16).

For the average case setting, let

$$\alpha := \lambda\{\boldsymbol{X} \in [0,1]^n:\ e_k^{\mathrm{avg}}(\boldsymbol{X}) \leqslant \beta e_0^{\mathrm{avg}}(\boldsymbol{X})\}.$$

Optimality of ϕ_k^{unr} and (36) give

$$\begin{aligned}\alpha &= \lambda\left\{\boldsymbol{X} \in [0,1]^n:\ \sum_{i=k+1}^{2^n} Z_i^2(\boldsymbol{X}) \leqslant \beta^2 e_0^{\mathrm{avg}}(\boldsymbol{X})^2\right\} \\ &\leqslant \lambda\left\{\boldsymbol{X} \in [0,1]^n:\ \frac{Z_1^2(\boldsymbol{X})}{\prod_{i=1}^n (X_i^2 + (1-X_i)^2)} \geqslant \frac{1-\beta^2}{k}\right\} \leqslant \frac{k}{1-\beta^2} c,\end{aligned}$$

where c is given by

$$c = \int_{[0,1]^n} \frac{Z_1^2(\boldsymbol{X})}{\prod_{i=1}^n (X_i^2 + (1-X_i)^2)}\,\mathrm{d}\boldsymbol{X}.$$

Due to (15) and (16) we have

$$\begin{aligned}c &= 2^n \int_{[0,0.5]^n} \prod_{i=1}^n \frac{(1-X_i)^2}{X_i^2 + (1-X_i)^2}\,\mathrm{d}\boldsymbol{X} \\ &= \left(2\int_0^{0.5} \frac{(1-t)^2}{t^2 + (1-t)^2}\,\mathrm{d}t\right)^n = \left(\frac{1+\ln 2}{2}\right)^n,\end{aligned}$$

as claimed. This completes the proof. □

6. Randomized setting

In this section we study the randomized setting. In this setting, the sample points of information as well as the choice of an algorithm may depend on a random element t from a set T. The elements t's are distributed according to a probability measure ρ defined on measurable subsets of T. That is, $\phi_t(N_t(f,\boldsymbol{X})$ is now our approximation to

[1] Chebyshev's inequality states that

$$\lambda\{\boldsymbol{X}:\ f(\boldsymbol{X}) \geqslant \gamma\} \leqslant E(f)/\gamma,$$

where $E(f)$ is the expectation of the measurable function f.

$\tau(f)(\boldsymbol{X})$ and the information N has the *randomized* cardinality defined as the expected value with respect to t, i.e.,

$$k(N) = \sup_{f\in\mathscr{F}_n} \sup_{\boldsymbol{X}\in[0,1]^n} \int_T k_t(f,\boldsymbol{X})\,\rho(\mathrm{d}t),$$

where $k_t(f,\boldsymbol{X})$ is defined as in Section 4 for a fixed random parameter t.

The *randomized* error of the algorithm ϕ is now given by

$$e^{\mathrm{ran}}(\phi) = \sup_{f\in\mathscr{F}_n} \sup_{\boldsymbol{X}\in[0,1]^n} \left(\int_T |\tau(f)(\boldsymbol{X}) - \phi_t(N_t(f,\boldsymbol{X}))|^2\,\rho(\mathrm{d}t) \right)^{1/2}. \tag{37}$$

Similarly as in the previous sections, let Φ_k^{res} and Φ_k^{unr} denote the classes of randomized algorithms which use information with the randomized cardinality at most k. That is, algorithms from the class Φ_k^{res} use randomized sample points that do not depend on $\boldsymbol{X}$, whereas algorithms from the class Φ_k^{unr} use randomized points that may depend on $\boldsymbol{X}$.

We now define two algorithms that belong to the classes Φ_k^{res} and Φ_k^{unr}. For the class Φ_k^{res}, take $T=[0,1]$ and $A=[0,k\,2^{-n}]$ for an integer $k\in[0,2^n]$. Let ρ be the Lebesgue measure. Then define $k_t(f,\boldsymbol{X})=k_t(f)=2^n$ for $t\in A$ and $k_f(f,\boldsymbol{X})=k_t(f)=0$ otherwise, as well as the information $N_t(f,\boldsymbol{X})=N_t(f)=[f(\boldsymbol{x}_1),\ldots,f(\boldsymbol{x}_{2^n})]$ for $t\in A$, and $N_t(f,\boldsymbol{X})=N_t(f)=0$ otherwise. Here, $\boldsymbol{x}_i$ for $i=1,2,\ldots,2^n$ are all Boolean sample points from $\{0,1\}^n$. Clearly, the information N has the randomized cardinality k. The algorithm ϕ_k^{ran} is defined by

$$\phi_{k,t}^{\mathrm{ran}}(f,\boldsymbol{X}) = \begin{cases} \tau(f)(\boldsymbol{X}) & \text{if } t\in A,\\ 0.5 & \text{otherwise.}\end{cases} \tag{38}$$

Observe that ϕ_k^{ran} is well defined and indeed uses the information N. The sample points used by ϕ_k^{ran} are either all Boolean sample points or none of them. In either case, they do not depend on $\boldsymbol{X}$ and therefore $\phi_k^{\mathrm{ran}}\in\Phi_k^{\mathrm{res}}$. As we shall see, the algorithm ϕ_k^{ran} is optimal in the class Φ_k^{res}.

For the class Φ_k^{unr}, we define an algorithm ϕ_k^{mc} which is closely related to the classical Monte Carlo algorithm and then we prove its optimality properties. Let us recall that the classical Monte Carlo algorithm approximates the integrand $\int_{[0,1]^n} g(\boldsymbol{Y})\,\mathrm{d}\boldsymbol{Y}$ of a L_2-integrable function g by $k^{-1}\sum_{i=1}^k g(\boldsymbol{Y}_i)$, where the randomized points $\boldsymbol{Y}_i$ are independent and uniformly distributed over $[0,1]^n$. It is well known that the square randomized error of the classical Monte Carlo algorithm is

$$\int_{[0,1]^{nk}} \left(\int_{[0,1]^n} g(\boldsymbol{Y})\,\mathrm{d}\boldsymbol{Y} - \frac{1}{k}\sum_{i=1}^k g(\boldsymbol{Y}_i) \right)^2 \mathrm{d}\boldsymbol{Y}_1\cdots\mathrm{d}\boldsymbol{Y}_k = \frac{1}{k}\,V(g), \tag{39}$$

where $V(g)$ is the variance of the function g,

$$V(g) = \int_{[0,1]^n} g^2(\boldsymbol{Y})\,\mathrm{d}\boldsymbol{Y} - \left(\int_{[0,1]^n} g(\boldsymbol{Y})\,\mathrm{d}\boldsymbol{Y} \right)^2.$$

In our case, we know that the characteristic polynomials are related to multivariate integration by (4) of Proposition 3. It would be tempting to set $g(\boldsymbol{Y}) = f(\boldsymbol{S}(\boldsymbol{Y}, \boldsymbol{X}))$. However, we can do a little better by remembering that the values of the characteristic polynomials are always from $[0,1]$. Hence, we can make their values more symmetric if we use the Monte Carlo algorithm for the function $f - 0.5$ and then add 0.5. Also, as in [17], we divide the sum of the computed function values by $k + \sqrt{k}$ instead of by k. That is, the algorithm ϕ_k^{mc} for $k \geqslant 1$ is defined as

$$\begin{aligned}\phi_{k,t}^{\mathrm{mc}}(f, \boldsymbol{X}) &= \frac{1}{k+\sqrt{k}} \sum_{i=1}^{k} (f(\boldsymbol{S}(\boldsymbol{Y}_i, \boldsymbol{X})) - 0.5) + \frac{1}{2} \\ &= \frac{1}{k+\sqrt{k}} \sum_{i=1}^{k} f(\boldsymbol{S}(\boldsymbol{Y}_i, \boldsymbol{X})) + \frac{\sqrt{k}}{2(k+\sqrt{k})}, \end{aligned} \tag{40}$$

where $t = [\boldsymbol{Y}_1, \ldots, \boldsymbol{Y}_k] \in T = [0,1]^{nk}$ and $\boldsymbol{Y}_i$ are independent and uniformly distributed over $[0,1]^n$.

Let $\boldsymbol{x}_i = \boldsymbol{S}(\boldsymbol{Y}, \boldsymbol{X})$. Then $\boldsymbol{x}_i$'s are independent random Boolean vectors whose components have binomial distribution with the probability of head equal to the successive components of $\boldsymbol{X}$. This is consistent with Proposition 1. With this interpretation, the algorithm (40) is the Rubin estimator, see [16].

Observe that the number of the sample points used in the algorithm ϕ_k^{mc} is always the same, $k(f, \boldsymbol{X}) \equiv k$. For $k = 0$ we set $\phi_{0,t}^{\mathrm{mc}}(f, \boldsymbol{X}) \equiv 0.5$ which, as we will see, is the optimal algorithm.

Assume now that $k \geqslant 1$. Although the random point t does not depend on $\boldsymbol{X}$, the sample points used by the algorithm ϕ_k^{mc} *do* depend on $\boldsymbol{X}$. This is because the sample points are $\boldsymbol{S}(\boldsymbol{Y}_i, \boldsymbol{X})$ and the function $\boldsymbol{S}$ depends on $\boldsymbol{X}$. For example, take $\boldsymbol{X} = [1, 1, \ldots, 1]$. Then $\boldsymbol{S}(\boldsymbol{Y}_i, \boldsymbol{X}) = \boldsymbol{X}$ for all $\boldsymbol{Y}_i$. In fact, for all boundary points $\boldsymbol{X} \in \{0,1\}^n$ we have $\boldsymbol{S}(\boldsymbol{Y}_i, \boldsymbol{X}) = \boldsymbol{X}$ with probability 1. Therefore, $\phi_k^{\mathrm{mc}} \in \Phi_k^{\mathrm{unr}}$ and $\phi_k^{\mathrm{mc}} \notin \Phi_k^{\mathrm{res}}$.

By

$$e^{\mathrm{ran}}(\Phi_k) = \inf_{\phi \in \Phi_k} e^{\mathrm{ran}}(\phi), \tag{41}$$

we denote the minimal randomized error of algorithms from the class $\Phi_k = \Phi_k^{\mathrm{res}}$ or from the class $\Phi_k = \Phi_k^{\mathrm{unr}}$.

In the randomized setting, we are looking, in particular, for the best probability measure ρ which minimizes the randomized error of an algorithm. Assume that ρ is atomic, i.e., there exists a point t^* such that $\rho(A) = 1$ iff $t^* \in A$. Such a choice of the measure ρ corresponds to deterministic algorithms ϕ_{t^*} that uses the deterministic information N_{t^*}. In this case, $e^{\mathrm{ran}}(\phi) = e^{\mathrm{wor}}(\phi_{t^*})$. Therefore

$$e^{\mathrm{ran}}(\Phi_k) \leqslant e^{\mathrm{wor}}(\Phi_k) \tag{42}$$

for both classes $\Phi = \Phi_k^{\mathrm{res}}$ and Φ_k^{unr}.

We now show that for $k = 0$ we have

$$e^{\mathrm{ran}}(\Phi_k^{\mathrm{unr}}) = e^{\mathrm{ran}}(\Phi_k^{\mathrm{res}}) = 0.5. \tag{43}$$

Indeed, due to (42) it is enough to show that $e^{\mathrm{ran}}(\Phi_k^{\mathrm{unr}}) \geqslant 0.5$ since $e^{\mathrm{wor}}(\Phi_k^{\mathrm{res}}) = 0.5$ due to Theorem 1. For $k = 0$, the algorithm $\phi_t(N_t(f, \boldsymbol{X}) = a_t(\boldsymbol{X})$ does not depend on f. It is easy to show that

$$\int_T |\tau(f)(\boldsymbol{X}) - a_t(\boldsymbol{X})|^2 \rho(\mathrm{d}t) \geqslant \left(\tau(f)(\boldsymbol{X}) - \int_T a_t(\boldsymbol{X})\rho(\mathrm{d}t)\right)^2.$$

By taking $f \equiv 0$ and 1 we see that

$$\sup_{f \in \mathcal{F}_n} \int_T |\tau(f)(\boldsymbol{X}) - a_t(\boldsymbol{X})|^2 \rho(\mathrm{d}t) \geqslant 0.25.$$

This shows that the randomized error must be at least 0.5. Furthermore the randomized error 0.5 is achieved for $\phi \equiv 0.5$.

We now prove optimality properties of ϕ^{ran} and ϕ^{mc} in the classes Φ_k^{res} and Φ_k^{unr}, respectively.

Theorem 5. *For the class* Φ_k^{res}, *we have*

$$e^{\mathrm{ran}}(\Phi_k^{\mathrm{res}}) = e^{\mathrm{ran}}(\phi_k^{\mathrm{ran}}) = e^{\mathrm{ran}}(\Phi_0^{\mathrm{res}})\sqrt{\max(0, 1 - k\,2^{-n})}. \tag{44}$$

For the class Φ_k^{unr}, *we have*

$$e^{\mathrm{ran}}(\Phi_k^{\mathrm{unr}}) \leqslant e^{\mathrm{ran}}(\phi_k^{\mathrm{mc}}) = \frac{e^{\mathrm{ran}}(\Phi_0^{\mathrm{unr}})}{1 + \sqrt{k}}, \quad \forall k, \tag{45}$$

and for $k = 2^s$ *with* $s < n - 1$ *we have*

$$e^{\mathrm{ran}}(\Phi_k^{\mathrm{unr}}) \geqslant \frac{e^{\mathrm{ran}}(\Phi_0^{\mathrm{unr}})}{4\sqrt{k}}. \tag{46}$$

Theorem 5 states that the algorithm ϕ^{ran} is optimal in the class Φ_k^{res}. Its randomized error is, however, very close to the initial error 0.5 as long as k is small relative to 2^n. This means that the approximate evaluation of characteristic polynomials is *intractable* in the randomized setting in the class Φ_k^{res}.

For the class Φ_k^{unr} with $k = 2^s$, $s < n - 1$, the algorithm ϕ^{mc} minimizes the randomized error up to a factor of 4. As we shall see in the proof, the factor 2 is lost due to the use of varying cardinality, see [19], and the other factor 2 is lost due to the switch to an average case setting.

We also remark that Mathé [17], proved that the error $(1 + \sqrt{k})^{-1}$ is the exact minimal error of randomized algorithms with fixed cardinality for the integration problem for the class of continuous functions bounded by one in the sup norm. In our case, the class consists of finitely many discontinuous functions, and this is why we cannot apply the result of Mathé.

The most important part of Theorem 5 states that the minimal randomized error now goes to zero with rate $(1+\sqrt{k})^{-1}$. If we want to reduce the initial error by ε, that is, $e^{\mathrm{ran}}(\Phi_k^{\mathrm{unr}}) \leqslant \varepsilon e^{\mathrm{ran}}(\Phi_0^{\mathrm{unr}})$, then

$$k = \min\left\{2^n, \left\lceil \left(\frac{1}{a\varepsilon} - 1\right)^2 \right\rceil\right\}, \quad a \in [0.125, 1].$$

This means that k depends polynomially on ε^{-1} and is independent of n. This is in sharp contrast to the worst and average case settings where k depends exponentially on n. Hence, the approximate evaluation of characteristic polynomials is *tractable* in $1/\varepsilon$ in the randomized setting for the class Φ_k^{unr}.

Proof. We first consider the class Φ_k^{res}. For the algorithm ϕ_k^{ran} we have

$$\int_0^1 (\tau(f)(\boldsymbol{X}) - \phi_{k,t}^{\mathrm{ran}}(N_t(f,\boldsymbol{X})))^2 \rho(\mathrm{d}t) = \int_{k\,2^{-n}}^1 (\tau(f)(\boldsymbol{X}) - 0.5)^2 \rho(\mathrm{d}t)$$
$$\leqslant 0.25(1 - k\,2^{-n}).$$

Thus $e^{\mathrm{ran}}(\phi_k^{\mathrm{ran}}) \leqslant 0.5\sqrt{1 - k\,2^{-n}}$.

We now show that for an arbitrary algorithm ϕ from the class Φ_k^{res} we have $e^{\mathrm{ran}}(\phi) \geqslant 0.5\sqrt{1 - k\,2^{-n}}$. Let N_t be information used by ϕ_t and let $k_t(f,\boldsymbol{X}) = k_t(f)$ be the cardinality of information for f. Define

$$A_f = \{t \in T : k_t(f) \leqslant 2^n - 1\}.$$

Since

$$k \geqslant \int_T k_t(f)\rho(\mathrm{d}t) \geqslant \int_{T - A_f} 2^n \rho(\mathrm{d}t) = 2^n(1 - \rho(A_f))$$

we have

$$\rho(A_f) \geqslant 1 - k2^{-n}, \quad \forall f \in \mathscr{F}_n. \tag{47}$$

For a Boolean function $f \in \mathscr{F}_n$ and a point $t \in A_f$, let $\boldsymbol{x}_i = \boldsymbol{x}_i(f,t)$, $i = 1,2,\ldots,k_t(f) \leqslant 2^n - 1$, denote the sample points used by the information N_t for f. Choose a Boolean point $\boldsymbol{y}$ which is different from all $\boldsymbol{x}_i$'s.

Take the Boolean function f^*, which depends on f and t, such that $f^*(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)$, $i = 1,2,\ldots,k_t(f)$, and $f^*(\boldsymbol{y}) = 1 - f(\boldsymbol{y})$. Observe that $A_f = A_{f^*}$ and the algorithm ϕ_t gives the same approximation for f and f^*.

Take $\boldsymbol{X} = \boldsymbol{y}$. Due to (6), we have

$$\tau(f)(\boldsymbol{X}) = f(\boldsymbol{y}) \quad \text{and} \quad \tau(f^*)(\boldsymbol{X}) = f^*(\boldsymbol{y}) = 1 - f(\boldsymbol{y}).$$

Let $a_t(f)=\phi_t(N_t(f),\boldsymbol{X})=\phi_t(N_t(f^*),\boldsymbol{X})$. Then we have

$$\begin{aligned} e^{\mathrm{ran}}(\phi)^2 &\geqslant 0.5\left(\int_{A_f}(\tau(f)(\boldsymbol{X})-a_t(f))^2\rho(\mathrm{d}t)\right.\\ &\quad\left.+\int_{A_{f^*}}(\tau(f^*)(\boldsymbol{X})-a_t(f))^2\rho(\mathrm{d}t)\right)\\ &= 0.5\int_{A_f}((1-a_t(f))^2+a_t^2(f))\rho(\mathrm{d}t). \end{aligned}$$

Since $(1-x)^2+x^2\geqslant 0.5$, this and (47) yield

$$e^{\mathrm{ran}}(\phi)\geqslant\left(0.25\int_{A_f}\rho(\mathrm{d}t)\right)^{1/2}=0.5\sqrt{1-k\,2^{-n}},$$

as claimed.

We now consider the class Φ_k^{unr}. We first estimate the randomized error of the algorithm ϕ_k^{mc}. For a fixed $\boldsymbol{X}$, let $g(\boldsymbol{Y})=f(\boldsymbol{S}(\boldsymbol{Y},\boldsymbol{X}))-0.5$, and let $I(g^p)=\int_{[0,1]^n}g^p(\boldsymbol{Y})\,\mathrm{d}\boldsymbol{Y}$ for $p=1$ and 2. Note that $g^2\equiv 0.25$ and, therefore, $I(g^2)=0.25$. Then using the same proof as the classical proof for (39) we have

$$\begin{aligned} &\int_{[0,1]^{nk}}(\tau(f)(\boldsymbol{X})-\phi_{k,t}^{\mathrm{mc}}(f,\boldsymbol{X}))^2\,\mathrm{d}t\\ &\quad=\int_{[0,1]^{nk}}\left(I(g)-\frac{1}{k+\sqrt{k}}\sum_{i=1}^{k}g(\boldsymbol{Y}_i)\right)^2\mathrm{d}\boldsymbol{Y}_1\cdots\mathrm{d}\boldsymbol{Y}_k\\ &\quad=I^2(g)-\frac{2k}{k+\sqrt{k}}I^2(g)+\frac{1}{(k+\sqrt{k})^2}\sum_{i,j=1}^{k}\int_{[0,1]^{nk}}g(\boldsymbol{Y}_i)g(\boldsymbol{Y}_j)\,\mathrm{d}\boldsymbol{Y}_1\cdots\mathrm{d}\boldsymbol{Y}_k\\ &\quad=I^2(g)\frac{\sqrt{k}-k}{k+\sqrt{k}}+\frac{1}{(k+\sqrt{k})^2}(kI(g^2)+(k^2-k)I^2(g))\\ &\quad=I^2(g)\left(\frac{\sqrt{k}-k}{k+\sqrt{k}}+\frac{k^2-k}{(k+\sqrt{k})^2}\right)+\frac{k}{4(k+\sqrt{k})^2}\\ &\quad=I(g^2)\cdot 0+\frac{1}{4(1+\sqrt{k})^2}. \end{aligned}$$

This proves that $e^{\mathrm{ran}}(\phi_k^{\mathrm{mc}})=0.5/(1+\sqrt{k})=e^{\mathrm{ran}}(\Phi_0^{\mathrm{unr}})/(1+\sqrt{k})$, as claimed in the upper bound of Theorem 5.

We now prove that for $k=2^s$ with $s<n-1$, an arbitrary algorithm ϕ from the class Φ_k^{unr} has the randomized error at least equal to $1/(8\sqrt{k})$.

To obtain a lower bound on the randomized error of ϕ we apply the standard proof technique which relates the randomized error to the average case error, see [7, 19, 23, 25]. Let μ be an arbitrary probability measure on the class $\mathscr{F}_n$ of Boolean functions. Let N_t be the information used by ϕ_t, and let $k_t(f,\boldsymbol{X})$ be the cardinality of information for $(f,\boldsymbol{X})$. We know that

$$\sup_{f\in\mathscr{F}_n}\sup_{\boldsymbol{X}\in[0,1]^n}\int_T k_t(f,\boldsymbol{X})\rho(\mathrm{d}t)\leqslant k,$$

Replacing the supremum with respect to f by the expectation we have

$$\int_T\int_{\mathscr{F}_n} k_t(f,\boldsymbol{X})\mu(\mathrm{d}f)\rho(\mathrm{d}t)\leqslant k,\quad \forall\boldsymbol{X}\in[0,1]^n.$$

Define the set

$$T(\boldsymbol{X})=\left\{t\in T:\int_{\mathscr{F}_n} k_t(f,\boldsymbol{X})\mu(\mathrm{d}f)\leqslant 2k\right\}.$$

As in (47), it is easy to see that

$$\rho(T(\boldsymbol{X}))\geqslant 0.5,\quad \forall\boldsymbol{X}\in[0,1]^n.$$

Similarly, the randomized error of ϕ is bounded from below by

$$\begin{aligned} e^{\mathrm{ran}}(\phi)^2 &= \sup_{f\in\mathscr{F}_n}\sup_{\boldsymbol{X}\in[0,1]^n}\int_T |\tau(f)(\boldsymbol{X})-\phi_t(N_t(f,\boldsymbol{X}))|^2\,\rho(\mathrm{d}t)\\ &\geqslant \sup_{\boldsymbol{X}\in[0,1]^n}\int_{T(\boldsymbol{X})}\left(\int_{\mathscr{F}_n}|\tau(f)(\boldsymbol{X})-\phi_t(N_t(f,\boldsymbol{X}))|^2\,\mu(\mathrm{d}f)\right)\rho(\mathrm{d}t). \end{aligned}\tag{48}$$

Let

$$e^{\mathrm{avg}}(\boldsymbol{X},\mu,2k)=\left(\inf_{\phi\in\Phi_{2k}^{\mathrm{unr}}}\int_{\mathscr{F}_n}|\tau(f)(\boldsymbol{X})-\phi(N(f,\boldsymbol{X}))|^2\,\mu(\mathrm{d}f)\right)^{1/2}$$

denote the minimal *average* case error for approximating the characteristic polynomials at the point $\boldsymbol{X}$ by deterministic algorithms that use information of cardinality at most $2k$. This and (48) yields

$$e^{\mathrm{ran}}(\phi)^2\geqslant\sup_{\boldsymbol{X}\in[0,1]^n}e^{\mathrm{avg}}(\boldsymbol{X},\mu,2k)^2\rho(T(\boldsymbol{X})\geqslant 0.5\sup_{\boldsymbol{X}\in[0,1]^n}e^{\mathrm{avg}}(\boldsymbol{X},\mu,2k)^2.\tag{49}$$

We now select a special probability measure μ and a point $\boldsymbol{X}$ such that we can compute the minimal average case error $e^{\mathrm{avg}}(\boldsymbol{X},\mu,2k)$. Recall that $k=2^s$ with $s<n-1$. We choose μ as the probability measure over the Boolean functions f which only depend on the first $s+2\leqslant n$ components of $\boldsymbol{x}$. That is, let

$$\mathscr{F}_{n,s}=\{f\in\mathscr{F}_n:\ \exists g:\ \{0,1\}^{s+2}\to\{0,1\}\ \text{such that}\ f(\boldsymbol{x})=g(x_1,x_2,\ldots,x_{s+2})\}.$$

Clearly, the cardinality of the set $\mathscr{F}_{n,s}$ is $2^{2^{s+2}}$. We assume that μ is equally probable over $\mathscr{F}_{n,s}$, i.e., $\mu(\{f\})=2^{-2^{s+2}}$ for all $f\in\mathscr{F}_{n,s}$. As the points $\boldsymbol{X}$ we take $\boldsymbol{X}=[0.5,\ldots,0.5,0,\ldots,0]$, where 0.5 is taken $s+2$ times.

This choice of μ and $\boldsymbol{X}$ corresponds to the average case setting with n replaced by $s+2$. From (33) we know that

$$e^{\mathrm{avg}}(\boldsymbol{X},\mu,2k)^2 = 0.25\cdot\left(\sum_{\boldsymbol{x}\in\{0,1\}^{s+2}} E_{\boldsymbol{x}}^2(\boldsymbol{X}) - \sum_{i=1}^{2k} E_{\boldsymbol{x}_i(\boldsymbol{X})}^2(\boldsymbol{X})\right). \tag{50}$$

Observe that just now all $E_{\boldsymbol{x}}(\boldsymbol{X}) = 2^{-s-2}$ and therefore (50) can be rewritten as

$$e^{\mathrm{avg}}(\boldsymbol{X},\mu,2k)^2 = 0.25(2^{-s-2} - 2k\,2^{-2(s+2)}) = (32k)^{-1}.$$

From (49) we conclude that $e^{\mathrm{ran}}(\phi)\geqslant 1/(8\sqrt{k})$, as claimed. □

7. Related works

We briefly mention a few related works. Characteristic polynomial is one of the arithmetizations of Boolean functions, mostly used in combinational circuit analysis. Arithmetization was first studied in [6, 22]. A slightly different arithmetization than ours was used for the earlier PCP construction [3] where a multivariate polynomial is obtained to check for satisfiability. Arithmetization was also used for testing whether a "black-box" is computing a low degree polynomial on most points and for other PCP and IP works [13, 3, 21, 5, 2]. In all these studies, it was assumed that the structure of the Boolean function is known whereas in our model it is an oracle; it is a "black-box" that produces outputs on Boolean inputs. Boolean functions can also be represented by polynomials over the ring of integers modulo m so that they agree on all 0–1 assignments. Low degree polynomial representation is preferable and various bounds were established [8]. The degree of the representing polynomial is related to the complexity of the representation and is characterized by the combinatorial properties of the Boolean function [18, 20] and a tight lower bound was obtained in [18].

Acknowledgements

We are deeply indebted to Funda Ergun, Peter Mathé, Erich Novak, Piotr Pokarowski, Joseph Traub, and Mihalis Yannakakis for their constructive and valuable comments.

References

[1] V.D. Agrawal, D. Lee, H. Woźniakowski, Numerical computation of characteristic polynomials of Boolean functions and its applications, Numer. Algorithms 17 (1998) 261–278.
[2] S. Ar, R.J. Lipton, R, Rubinfeld, M. Sudan, Reconstructing algebraic functions from mixed data, SIAM J. Comput. 28 (1999) 487–496.
[3] S. Arora, C. Lund, R. Motwani, M. Sudan, M. Szegedy, Proof verification and hardness of approximation problems, 33rd Annu. Symp. on Foundations of Computer Science 1992 (pp. 14–23).
[4] S. Arora, M. Safra, Probabilistic checking of proofs: a new characterization of NP, 33rd Annu. Symp. on Foundations of Computer Science 1992 (pp. 2–13).

[5] S. Arora, M. Sudan, Improved low-degree testing and its aplications, 29th Annu. ACM Symp. on the Theory of Computing 1997 (pp. 485–495).
[6] L. Babai, L. Fortnow, Arithmetization: a new method in structural complexity theory, Comput. Complexity 1 (1) (1991) 41–66.
[7] N.S. Bakhvalov, On approximate calculation of integrals Vestnik MGU, Ser. Math. Mech. Astron. Phys. Chem. 4 (1959) 3–18 (in Russian).
[8] D.A.M. Barrington, R. Beigel, S. Rudich, Representing Boolean functions as polynomials modulo composite numbers, 24th Annu. ACM Symp. on the Theory of Computing 1992 (pp. 455–461).
[9] S.A. Cook, The complexity of theorem-proving procedures, 3rd Annu. ACM Symp. on the Theory of Computing 1971 (pp. 151–158).
[10] A.D. Friedman, P.R. Menon, Fault Detection in Digital Circuits, Prentice-Hall, Englewood Cliffs, NJ, 1971.
[11] H. Fujiwara, Logic Testing and Design for Testability, MIT Press, Cambridge, MA, 1985.
[12] M.R. Garey, D.S. Johnson, Computers and Intractability, Freeman, New York, 1979.
[13] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, A. Wigderson, Self-testing/correcting for polynomials and for approximate functions, 23rd Annu. ACM Symp. on the Theory of Computing 1991 (pp. 32–42).
[14] J. Jain, J. Bitner, D.S. Fussel, J.A. Abraham, Probabilistic verification of Boolean functions, Formal Methods System Design 1 (1992) 63–117.
[16] E.L. Lehmann, Theory of Point Estimation, Wiley, New York, 1983.
[17] P. Mathé, The optimal error of Monte Carlo integration, J. Complexity 11 (1995) 394–415.
[18] N. Nisan, M. Szegedy, On the degree of Boolean functions as real polynomials, 24th Annu. ACM Symp. on the Theory of Computing 1992 (pp. 455–461).
[19] E. Novak, Stochastic properties of quadrature formulas, Numer. Math. 53 (1988) 609–620.
[20] R. Paturi, ON the degree of polynomials that approximate symmetric Boolean functions, 24th Annu. ACM Symp. on the Theory of Computing 1992 (pp. 468–474).
[21] R. Rubinfeld, M. Sudan, Robust characterizations of polynomials with applications to program testing, Siam J. Comput. 25 (2) (1996) pp 252–271.
[22] A. Shamir, IP=PSPACE, JACM 39 (4) (1992) pp 869–877.
[23] J.F. Traub, G.W. Wasilkowski, H. Woźniakowski, Information Based-Complexity, Academic Press, New York, 1988.
[24] G.W. Wasilkowski, Information of varying cardinality, J. Complexity 2 (1986) 204–228.
[25] G.W. Wasilkowski, Randomization for continuous problems, J. Complexity 5 (1989) 195–218.