A quantum algorithm for approximating the influences of Boolean functions

Hong-Wei Li$^{1,2}$ and Li Yang$^1$*

1. State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
2. School of Mathematics and Statistics, Henan Institute of Education, Zhengzhou, 450046, Henan, China

(Dated: September 3, 2014)

We investigate the influences of variables on a Boolean function $f$ based on the quantum Bernstein-Vazirani algorithm. A previous paper has proved that if a $n$-variable Boolean function $f(x_1,\cdots,x_n)$ does not depend on an input variable $x_i$, using the Bernstein-Vazirani circuit to $f$ will always obtain an output $y$ that has a 0 in the $i$th position. We generalize this result and show that after several times running the algorithm, the number of ones in each position $i$ is relevant to the dependence degree of $f$ on the variable $x_i$, i.e. the influence of $x_i$ on $f$, and we give the relational expression between them. On this foundation, we give an approximation algorithm to evaluate the influence of any variable on a Boolean function. Next, as an application, we use it to study the Boolean functions with juntas, and construct probabilistic quantum algorithms to learn certain Boolean functions.

PACS number(s): 03.67.Ac

I. INTRODUCTION

In [1], the authors studied the juntas using the quantum Bernstein-Vazirani algorithm, and they proved that using the Bernstein-Vazirani circuit to a Boolean function $f(x_1,\cdots,x_n)$ that has nothing to do with a variable $x_i$, would always obtain the output $y = (y_1,\cdots,y_n) \in \{0,1\}^n$ with $y_i = 0$. In this paper, we will generalize the conclusion to the result that the number of ones in each position $i$ of the outputs relates to the influence of $x_i$ on $f$, and our conclusion will contain the one in [1].

During the last thirty years, there are a lot of researches about the influences of variables on Boolean functions. In [2], the authors transformed randomized algorithms to the processors flipping the coin. In order that a single processor doesn’t control the global bit, it must have a Boolean function that every variable has a little influence. Hatami[3] pointed out that the influence of a variable on a Boolean function appears in various contexts such as probability theory, computer science and statistical physics, and Boolean functions with small total influences are in close touch with threshold phenomenon. [4] introduced harmonic analysis methods on Boolean functions for the first time, proved a so called KKL inequality to give a lower bound on total influences. The KKL inequality now is usually used to estimate some bounds[5, 6].

In this paper, we begin with some preliminaries necessary. Next, we give a conclusion about the influence of a variable of a Boolean function and the gains after running the Bernstein-Vazirani algorithm, based on this, we propose a quantum approximation algorithm to evaluate the influence, and finally we exploit the above result to the learning of juntas.

II. PRELIMINARIES

II.1. Notations and definitions

Definition 1 Let $f(x_1,\cdots,x_n) : \{0,1\}^n \rightarrow \{0,1\}$ be a Boolean function, $i \in \{1,2,\cdots,n\} = [n]$, $\alpha^i \in \{0,1\}^n$, and all the coordinates of $\alpha^i$ are 0 except the $i$th one. The influence of a variable $x_i$ on the function $f$ is defined as

$$I_f(i) = \Pr[f(x \oplus \alpha^i) \neq f(x)],$$

where $\oplus$ denotes bitwise exclusive-or.

Definition 2 For any Boolean function $f$, define the Walsh transform of it by

$$S_f(y) = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} (-1)^{f(x) + y \cdot x},$$

where $y \in \{0,1\}^n$.

II.2. The Bernstein-Vazirani algorithm[1, 7–9]

For any Boolean function $f$, define the $U_f$ gate as

$$U_f|x\rangle \equiv |x\rangle |z + f(x)\rangle,$$

where $x \in \{0,1\}^n$, $z \in \{0,1\}$, and the addition is modulo 2.

Applying $n$ Hadamard gates to $|x\rangle$ obtains

$$H^\otimes n|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle,$$

Now, begin with the initial state $|0\rangle^\otimes n |1\rangle$, do the fol-
lowing $\sum x \in \{0,1\}^n \frac{|0\rangle \cdot |0\rangle - |1\rangle \cdot |1\rangle}{\sqrt{2}}$.

Discarding the last qubit. If $f(x) = a \cdot x$, $a \in \{0,1\}^n$, we just get $|a\rangle$, and if we measure in the computational basis, we will get $a$. If $f(x)$ is not a linear function, from (2) and (5), the output is actually
\[ \sum y \in F_2^n S_f(y) |y\rangle. \] (6)

This time if we measure in the computational basis, we will get $y$ with probability $S_f^2(y)$.

II.3. The Hoeffding inequality [10]

If $X_1, X_2, \cdots, X_n$ are independent random variables and $a_i \leq X_i \leq b_i (i = 1, 2, \cdots, n)$, then for $t > 0$
\[ \Pr\{\frac{1}{n} \sum_{i=1}^n X_i - \frac{1}{n} \sum_{i=1}^n X_i \geq t\} \leq 2e^{-2n^2t^2 / \sum_{i=1}^n (b_i - a_i)^2}, \] (7)
where $EX$ is the expected value of the random variable $X$.

III. THE MAIN RESULTS ABOUT THE INFLUENCES OF BOOLEAN FUNCTIONS

Theorem 1 For any Boolean function $f$,
\[ I_f(i) = \sum_{y_i=1} S_f^2(y). \] (8)

[11] has a similar result as Theorem 1, but there has no detailed proof.

proof (We’ll use a method similar to [12] to complete the proof.) First, let
\[ C_f(\gamma) = \sum_{x \in F_2^n} (-1)^{f(x)} \otimes f(x \otimes \gamma), \] (9)
\[ S_{(C_f)}(y) = 2^{-n} \sum_{\gamma \in F_2^n} C_f(\gamma)(-1)^{-y} \] (10)

Eq. (9) is substituted in Eq. (10),
\[ S_{(C_f)}(y) = 2^{-n} \sum_{\gamma \in F_2^n} \sum_{x \in F_2^n} (-1)^{f(x)} \otimes f(x \otimes \gamma)(-1)^{x \otimes y}. \] (11)

Therefore,
\[ C_f(\gamma) = \sum_{y \in F_2^n} S_{(C_f)}(y)(-1)^{-y} \] (12)
\[ = 2^n \sum_{y \in F_2^n} S_f^2(y)(-1)^{-y}. \]

This proves that
\[ C_f(\alpha^i) = 2^n (\sum_{y : \alpha^i=0} S_f^2(y) - \sum_{y : \alpha^i=1} S_f^2(y)). \] (13)

On the other hand,
\[ C_f(\alpha^i) = |\{x \in F_2^n | f(x \oplus \alpha^i) + f(x) = 0\}| - |\{x \in F_2^n | f(x \oplus \alpha^i) + f(x) = 1\}| \] (14)
\[ = |V_0| - |V_1|, \]
from (13) and (14), we have
\[ \sum_{y : \alpha^i=0} S_f^2(y) = \sum_{y : \alpha^i=1} S_f^2(y) = \frac{|V_0|}{2^n} - \frac{|V_1|}{2^n}. \] (15)

In addition, by Parseval’s relation
\[ \sum_{y : \alpha^i=0} S_f^2(y) + \sum_{y : \alpha^i=1} S_f^2(y) = 1 = \frac{|V_0|}{2^n} + \frac{|V_1|}{2^n}. \] (16)

Combining (15) and (16), we achieve
\[ \begin{cases} \sum_{y : \alpha^i=0} S_f^2(y) = \frac{|V_0|}{2^n}, \\ \sum_{y : \alpha^i=1} S_f^2(y) = \frac{|V_1|}{2^n}. \end{cases} \] (17)

Henceforth
\[ I_f(i) = \frac{|V_1|}{2^n} = \sum_{y_i=1} S_f^2(y), \]
for $\{y \in \{0,1\}^n | y \cdot \alpha^i = 1\} = \{y = (y_1, \cdots, y_n) \in \{0,1\}^n | y_i = 1\}$.

From Theorem 1, we immediately have the following.

Theorem 2 Using the Bernstein-Vazirani circuit once to a Boolean function $f$, the probability of finding the one in a position $i$ (i.e. $y_i = 1$) is identical to the influence of the function $x_i$ on $f$. Specially, if $f$ is independent of $x_i$, we will always find $y_i = 0$; if $I_f(i) = 1$, we will always find $y_i = 1$.

proof According to quantum mechanics and (6), if we measure in the computational basis, we will get $y$ with probability $S_f^2(y)$. So the probability we get $y_i = 1$ is
\[ \Pr(y_i = 1) = \sum_{y_i=1} S_f^2(y) = I_f(i), \] (18)
where the second equal comes from (8). Specially, if $f$ is independent of $x_i$, that is to say $I_f(i) = 0$, by (18), $\Pr(y_i = 1) = 0$, consequently, we can’t get $y$ with $y_i = 1$, we will always find $y_i = 0$. If $I_f(i) = 1$, by (18), the probability we get $y_i = 1$ is 1.

Remark Theorem 1, 2 concludes theorem 3.1, 3.2 in [1] as a special case.
IV. THE QUANTUM ALGORITHM FOR THE INFLUENCES OF BOOLEAN FUNCTIONS

IV.1. The quantum algorithm

Algorithm 1
1. Run the Bernstein-Vazirani circuit to the function \( f \) for \( m \) times to get \( y_1, \ldots, y_m \).
2. For any fixed \( i \in [n] \), count the total number \( l_i \) of ones in \( y_1, \ldots, y_m \).
3. Compute \( p_i = p_i(m) = \frac{l_i}{m} \).

Then by Theorem 2 (18),
\[
I_f(i) \approx p_i, \tag{19}
\]
and the total influence of all variables on \( f \) is
\[
\sum_{i=1}^{n} I_f(i) \approx \sum_{i=1}^{n} p_i = \frac{\sum_{i=1}^{n} l_i}{m}. \tag{20}
\]

From this we can know some properties about influence, such as whether every variable has a little influence or not, whether the total influence is small or not, and so on and so forth mentioned in the introduction.

IV.2. The analysis of the above quantum algorithm

What’s the error scope of \( I_f(i) \) that we just compute through the above method? In other words, what’s the distance of \( p_i \) and \( I_f(i) \)?

Theorem 3 \( \forall \epsilon > 0 \), we have
\[
\Pr(|I_f(i) - p_i| < \epsilon) > 1 - 2e^{-2m\epsilon^2}. \tag{21}
\]

**proof** Let \( Y_i, i \in [n] \) be random variables such that
\[
Y_i = \begin{cases} 1 & y_i = 1 \\ 0 & y_i = 0, \end{cases} \tag{22}
\]
where \( y_i \) is the \( i \)th coordinate of the \( y \) measured. Then by the Theorem 2, \( \Pr[Y_i = 1] = I_f(i) \).

\( m \) times running the algorithm corresponds to \( m \) independent identical distributed random variables \( Y_i^j \), \( j \in [m] \). By the Hoeffding’s inequality,
\[
\Pr(|I_f(i) - \frac{1}{m} \sum_{j=1}^{m} Y_i^j| < \epsilon) > 1 - 2e^{-2m\epsilon^2}. \tag{24}
\]

From the second step of the algorithm, we know \( \sum_{j=1}^{m} Y_i^j = l_i \), i.e. (21) holds.

IV.3. Contrast to the classical algorithm

In the classic probability turing model, if we want to evaluate a \( I_f(i) \) for \( \forall i \in [n] \), we should randomly choose some \( x \in \{0,1\}^n \), by means of compare \( f(x) \) with \( f(x \oplus a^i) \) to get a rude estimate. Through the following definition of a random variable \( Z_i \),
\[
Z_i = \begin{cases} 1 & f(x) \neq f(x \oplus a^i) \\ 0 & f(x) = f(x \oplus a^i), \end{cases} \tag{25}
\]
we can get an analysis of the error range similar to that of using the quantum algorithm. We obtain that For any fixed \( i \in [n] \), in order to compute \( I_f(i) \), it must compute the function \( f \) \( m \) times using the classical algorithm to achieve the degree of accuracy received by running the quantum algorithm \( m \) times. Even so, running the quantum algorithm once can get the influences of all variables on the function but the classical algorithm can only get one variable’s influence. So our quantum algorithm gains an \( O(n) \) times speedup over the classical one.

V. APPLICATIONS IN SOME SPECIAL CASES

Recall that a junta is a Boolean function that only depend on at most \( k \) out of \( n \) variables. From Theorem 2, the probability of find the one in the algorithm is just relevant to the influence of the variable, is entirely unrelated to \( k \) and \( n \). So we can exploit the above quantum algorithm to learn juntas. In [1], the authors researched the quadratic and cubic functions, and gave deterministic quantum algorithms for these functions. The authors hoped to devise probabilistic quantum algorithms. Now, we’ll complete this work based on the above results of Theorem 2 and Theorem 3. Before doing this, we need the following conclusion.

**Lemma 1** If \( f(x) \) is of the form
\[
f(x_1, x_2, \cdots x_n) = \prod_{i=1}^{r} x_i \ (r \in [n]), \tag{26}
\]
then
\[
I_f(i) = \begin{cases} \frac{1}{2^r} & i \in [r] \\ 0 & i \in [n] - [r]. \end{cases} \tag{27}
\]

**proof** If \( i \in [n] - [r] \), from (26), \( f(x) = f(x \oplus a^i) \) for \( \forall x \in \{0,1\}^n \), so by definition 1, \( I_f(i) = 0 \).

In the case \( i \in [r] \), from (26), if there is a \( j \in [r] \), such that \( x_j = 0 \), then \( f(x) = 0 \). If for all \( j \in [r], x_j = 1 \), then \( f(x) = 1 \). Therefore if and only if \( x_j = 1 \) for all the \( j \in [r] \) and \( j \neq i, f(x) \neq f(x \oplus a^i) \). The number of \( x \in \{0,1\}^n \) with \( x_j = 1, j \in [r] - \{i\} \) is \( 2^{n-r+1} \), the total number of \( x \in \{0,1\}^n \) is \( 2^n \), so by definition 1 the influence of \( x_i \) on the function \( f \) is
\[
I_f(i) = \frac{2^{n-r+1}}{2^n} = 2^{1-r}. \tag{28}
\]
From the proof we can see that the similar conclusion holds for any product of \( r \) variables. Specially, if a variable \( x_i \) only appears in linear term, then \( I_f(i) = 1 \). If a variable \( x_i \) only emerges in quadratic term, then \( I_f(i) = \frac{1}{2} \). If a variable \( x_i \) only arises in cubic term, then \( I_f(i) = \frac{1}{4} \).

Now we give our probabilistic quantum algorithms.

V.1. Quadratic functions

Suppose \( f \) is a Boolean function that is composed of linear and quadratic terms and each variable emerges in at most one term. Our assignment is to find the variables in linear terms and those in quadratic terms.

Algorithm 2

We apply the Bernstein-vazirani circuit to \( f \) \( \rho \) (\( \rho \) is an integer, and \( \rho \geq 2 \)) times, if we always get 1 in a position \( i \), then \( x_i \) is declared to be in linear term. If we get some 1 and some 0 in a position \( j \), then \( x_j \) is declared to be in quadratic term. If we always get 0 in a position \( k \), then \( x_k \) is declared to be not in the expression of \( f \).

Now let us see the success probability of above method. If \( x_i \) is in linear term, from Lemma 1, \( I_f(i) = 1 \), so by Theorem 2, the probability we get \( y_i = 1 \) is 1, we’ll always find \( y_i = 1 \). Conversely, if \( x_i \) is in quadratic term, from Lemma 1, \( I_f(i) = \frac{1}{2} \), running the algorithm once, the probability of we getting \( y_i = 1 \) is \( \frac{1}{2} \), so the probability that \( x_i \) is in quadratic term but we still get \( y_i = 1 \) every time is

\[
\left(\frac{1}{2}\right)^\rho = \frac{1}{2^\rho}.
\] (29)

From this we can see that if we declare \( x_i \) to be in linear term, the error probability is exponentially small.

If \( x_k \) is declared to be not in the expression of \( f \), but in fact \( x_k \) is probably in quadratic term, the probability that this happens is

\[
\left(\frac{1}{2}\right)^\rho = \frac{1}{2^\rho}.
\] (30)

If we declare \( x_j \) to be in quadratic term, \( x_j \) is indeed in it. But \( x_j \) is indeed in quadratic term, we can’t judge it is

\[
\left(\frac{1}{2}\right)^\rho + \left(\frac{1}{2}\right)^\rho = \frac{1}{2^{\rho-1}}.
\] (31)

From (29), (30), (31), we can see that all of these error probabilities are exponentially small.

V.2. Cubic functions

This time we still suppose \( f \) is a Boolean function and each variable arise in at most one term. The difference is that there are cubic terms in the expression of \( f \) besides some linear and quadratic terms. Our aim is to determine the variables in linear, quadratic and cubic terms.

Algorithm 3

We still just apply the Bernstein-vazirani circuit to \( f \) \( \lambda(\lambda \) is an integer dependent of \( n \) and \( \lambda \geq 4 \)) times, if we always get ones in a position \( i \), then \( x_i \) is declared to be in linear term. If we find \( \mu \lambda \) ones in a position \( j \) (where \( \mu \in (\frac{1}{2} - \epsilon, \frac{1}{2} + \epsilon) \), \( \epsilon \) is a real number and \( 0 < \epsilon < \frac{1}{3} \), we may set \( \epsilon = 0.1 \)), then \( x_j \) is declared to be quadratic term. If we find \( \nu \lambda \) ones in a position \( l \) (where \( \nu \in (\frac{1}{2} - \epsilon, \frac{1}{2} + \epsilon) \), \( \epsilon = 0.1 \)), then \( x_l \) is declared to be in cubic term. If we always get 0 in a position \( k \), then \( x_k \) is declared to be not in the expression of \( f \).

The analysis of Algorithm 3 will be more complicated than that of Algorithm 2. We give a less precise evaluation. The same as (24), by Hoeffding’s inequality, we have

\[
\Pr(|\mu - \frac{1}{2}| < \epsilon) > 1 - 2e^{-2\lambda^2},
\] (32)

\[
\Pr(|\nu - \frac{1}{4}| < \epsilon) > 1 - 2e^{-2\lambda^2}.
\] (33)

Therefore we will get the quadratic and cubic terms with a probability no less than \( 1 - 2e^{-2\lambda^2} \). The number approximate to 1 exponentially with the increase of \( \lambda \). The analysis of linear terms is more like that about Algorithm 2.

In conclusion, we can use the generalized Bernstein-vazirani quantum for learning some simple functions, such as quadratic, cubic, quartic and maybe higher degree functions.

VI. CONCLUSIONS

We have presented a quantum approximation algorithm to compute the influence of every variable on the Boolean functions. In general, for \( n \) variables function, our algorithm is \( O(n) \) times faster than the classical one. Moreover, based on this, we give a probabilistic quantum algorithm for learning some special functions with simple form. The running time of attaining some success probability relies on the form of the function \( f \), but don’t on the total variables of \( f \). To this end, we use a method presented in [12], but compare with that, this paper has no the solving equation procedure in [12]. I expect that the methods in this paper will be helpful for some other questions.

ACKNOWLEDGEMENT

This work was supported by the National Natural Science Foundation of China under Grant No.61173157.
[1] D.Floess, E.Andersson, and M. Hillery, Quantum algorithms for testing and learning Boolean functions, Math. Struct. Comp. Science (2013), vol.23, pp.386-398.
[2] M.Ben-Or and N.Linial, Collective coin flipping, Randomness and Computation (S.Micali ed.) Academic press, New York, 1989, pp.91-115.
[3] H.Hatami, A structure theorem for Boolean functions with small total influences, Annals of Mathematics, 176(1), pp.509-533,2012. also ArXiv:1008.1021 v3 [math.CO] 12 Nov 2011.
[4] J.Kahn, G.Kalai and N. Linial, The influence of variables on Boolean functions, Proc.29th FOCS,1988, pp.68-80.
[5] R.d.Wolf, A brief introduction to fourier analysis on the Boolean cube, TCGS 1(2008), pp.1-20 Http:theoryofcomputing.org
[6] D.Gavinsky, J.Kempe, I.Kerenidis, R.Raz and R.d.Wolf, Exponential separation for one-way quantum communication complexity, with applications to cryptograph, SIAM J.Comput. 38(5):1695-1708(2008). arXiv:quant-ph/0611209 v3
[7] E.bernstein and U.Vazirani, Quantum complexity theory. Proceedings of the 25th Annual ACM Symposium on theory of computing, ACM Press, New York, 1993, pp.11-20.
[8] R. cleve, A. Ekert, C. Macchiavello and M. Mosca, Quantum algorithms revisited, Proc. R. Soc. Lond. A 1998 454, pp.339-354.
[9] M. Hillery and E. Anderson, Quantum tests for the linearity and permutation invariance of Boolean functions, Phys. Rev. A 84, 062326 (2011).
[10] W.Hoeffding, Probability inequalities for sums of bounded random variables, American statistical association journal, March 1963, pp.13-30.
[11] R.O'Donnell, Some topics in analysis of Boolean functions, STOC’08, May 17-20,2008, Victoria, British Columbia, Canada, invited paper.
[12] H.W.Li and L.Yang, Quantum algorithm for the finding of Boolean function’s linear structures, arXiv:1404.0611[quant-ph] 2 Apr 2014.