Block Multilinear Degree

Akash Kumar Singh, Siddhant Kar, Snehal Raj

Supervisor: Dr. Rajat Mittal

Contents

Chapter 1

Introduction

One of the most widely used models to study quantum algorithms is the quantum query model. Many algorithms, ranging from Grover's total search to Shor's famous factoring algorithm, use the quantum query model.

Any algorithm based on the quantum query model queries a Boolean function x repeatedly so that some property of x can be checked at the end. Checking this property can be thought of as calculating another function f on input x. The minimum number of queries that any such algorithm must make in order to determine f is called the quantum query complexity Q_f of f. The well-known polynomial method puts a lower bound on Q_f, by constructing a degree $2Q_f$ polynomial that approximates f up to some ϵ. The minimum degree that any such polynomial can attain is called the ϵ-approximate degree of f. Thus, $2Q_f$ is at least equal to the approximate degree of f.

Aaronson et al. [1] introduced a new notion, where they approximated f up to ϵ using a very specific kind of polynomial of degree $2Q_f$, called a block-multilinear polynomial. The minimum degree attainable by such a polynomial is called the ϵ-approximate block-multilinear degree. This notion of block-multilinear polynomials was used in [1] to solve a problem called Forrelation. Forrelation has achieved the largest gap between quantum and classical query complexities known yet among promise problems. A *promise* problem is a problem where there is some condition on the input function x and so we calculate $f(x)$ only for a limited number of x, for example, Simon's problem.

It is easy to see that the ϵ-approximate block-multilinear degree is at least equal to the ϵ-approximate degree. The former may hence be closer to query complexity. Aaronson et al. [2] showed an equivalence between degree 2 block-multilinear polynomials that approximate f and 1-query quantum algorithms for f. This, however, does not hold for higher degrees. They also showed an equivalence between general and block-multilinear degree up to some loss in the quality of approximation.

We mainly worked on the gap between the exact degree and exact block-multiliear degree of f. The exact block-multilinear degree is simply the 0-approximate block-multilinear degree. This is well-defined since f is a Boolean function, that is, we can always write f as a polynomial using indicator functions, which can then easily be converted to a block-multilinear polynomial of larger degree. We tried separating these two degrees by using two different approaches, symmetrization and dual witness. The details can be found in Chapter 3. We also present the main results of [1] in Chapter 4.

Chapter 2

Preliminaries

In this chapter, we give an overview of the basics of quantum mechanics and the quantum query model, and then we proceed to formally define block-multilinear degree and the Forrelation problem.

2.1 Quantum Mechanics

The postulates of quantum mechanics state that any quantum mechanical system can be quantized to the set of eigenstates of its observables. One of the most basic quantum mechanical systems is a simple 2-state system called the qubit. The qubit is a linear superposition of two states, $|0\rangle$ and $|1\rangle$, which form an orthonormal basis of the qubit space. We thus write the state of the qubit as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

where α and β are called the amplitudes and can be complex numbers in general. Like any observable, measuring this qubit must result in one of the basis states, with probability equal to the square of the corresponding amplitude. Thus, we have

$$\alpha^2 + \beta^2 = 1.$$

A quantum system can be thought of as having several of such qubits, and its state can be expressed as the tensor product of all of its qubit states. The postulates also state that the evolution of any quantum state must be linear and unitary, and thus any operation on a set of n qubits must be of the form

$$U|\psi\rangle = \alpha_0 U|0^{\otimes n}\rangle + \ldots + \alpha_{2^n-1} U|1^{\otimes n}\rangle$$

where U is a unitary $n \times n$ matrix. These operations are performed using quantum *gates*, similar to logic gates in classical computing, and they can be used to build quantum *circuits*. With this, we can now talk about the quantum query model.

2.2 Quantum Query Model

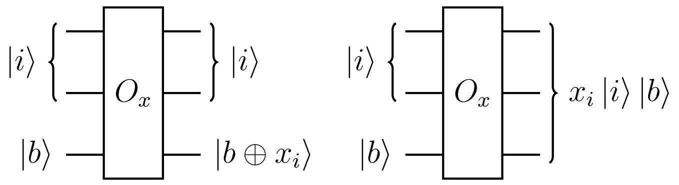
The quantum query model is the most widely used model to study quantum algorithms. Given a boolean function x, we are required to calculate a property $f(x)$ by querying x.

The function x may either be partial (promise problem) or total (total problem). A partial function has some condition on its outputs and hence can be constructed using fewer inputs, whereas a total function has no such restriction.

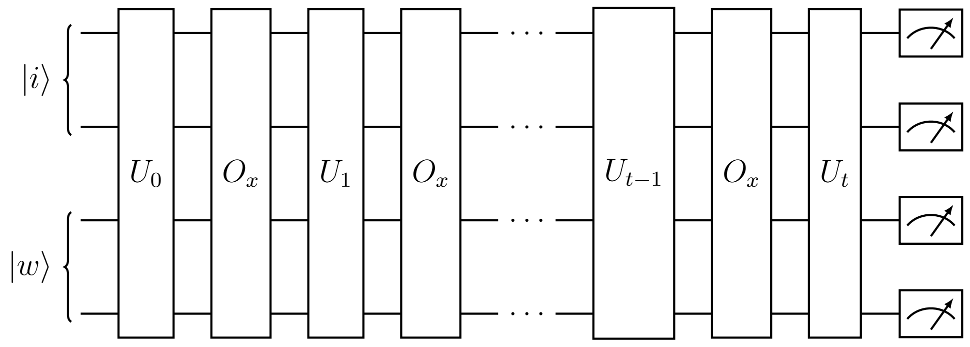
For our convenience, we assume x is a function from $\{-1,1\}^n$ to $\{-1,1\}$. Let us represent it in truth table form as $x = (x_1, \ldots, x_N)$, where $N = 2^n$ and each $x_i \in \{-1,1\}$ corresponds to the ith output. We assume that we only want to answer a yes/no question about x. Thus, we wish to calculate

$$f\colon \{-1,1\}^N \longrightarrow \{0,1\}.$$

We are given access to an oracle or black box O_x which is a quantum gate that, on input $|i\rangle$ and control qubit b, does either of the following.



Both the oracles shown above are equivalent. For example, to convert the first to the second, we set the control bit $|b\rangle$ to $|-\rangle = \frac{1}{\sqrt{2}}(|1\rangle - |-1\rangle)$. We will only use the latter oracle that takes $|i\rangle$ and returns $x_i|i\rangle$. Our quantum query algorithm can call the oracle several times and also perform other linear operations in between as shown below.



The gates in between the oracles are fixed and do not vary with x. Out of all the possible combinated states of the qubits at the end of the algorithm, we call only a few of these states *accepting* states. If the final measurement gives one of these states, the algorithm outputs 1, else it outputs 0. The algorithm should output 1 with high probability if $f(x) = 1$ and with 0 with high probability if $f(x) = 0$. More formally, for $\epsilon \geq 0$,

$$\Pr[\text{Algorithm outputs 1}] \in \begin{cases} [1-\epsilon, 1] & \text{if } f(x) = 1 \\ [0, \epsilon] & \text{if } f(x) = 0. \end{cases}$$

The number of calls t to the oracle O_x is called the query complexity of the algorithm. The query complexity Q_f of f, as discussed eariler, is the minimum possible query complexity of any such algorithm that calculates f.

2.3 Block-multilinear degree

We now formally define block-multilinear polynomials and block-multilinear degree.

Definition 1. *A block multilinear polynomial on* $\{-1,1\}^n$ *is a polynomial of the form*

$$p(x) = p(x_{1,1}, x_{1,2}, \ldots) = \sum_{(i_1,\ldots,i_k)} a_{i_1 \ldots i_k} x_{1,i_1} \ldots x_{k,i_k}$$

where its n *variables can be partitioned insto* k *disjoint blocks* $B_i, i \in [k]$*, such that* $x_{i,j} \in B_i \; \forall j$.

Definition 2. *Let* $\epsilon \geq 0$*. A boolean function* $f \colon \{-1,1\}^N \to \{0,1\}$ *is said to have* ϵ*-approximate block-multilinear degree, denoted* $\operatorname{bmdeg}_\epsilon(f)$*, equal to* d *if* d *is the minimum degree a block-multilinear polynomial* $p \colon \{-1,1\}^{Nk} \to [-1,1]$ *can have so that*

$$|p(x, \ldots x) - f(x)| \leq \epsilon \; \forall x \in \{-1,1\}^N.$$

Given any quantum query algorithm for f with complexity t, we can construct a block-multilinear polynomial of degree $2t$ that approximates f. This implies that we can have a degree $2Q_f$ block-multilinear approximation for f up to some ϵ, and hence $2Q_f$ must be at least equal to $\operatorname{bmdeg}_\epsilon(f)$. The following theorem, with proof, is a rephrased version of the result in [1].

Theorem 1. *The probability that a quantum query algorithm of complexity* t *accepts a function* $x \in -1, 1^N$ *is given by* $p(x, x, \ldots)$*, where* p *is a block multilinear polynomial with* $2t$ *blocks of size* N *each, and* $p(\bar{x})$ *is bounded in* $[-1,1]$.

Proof. Consider the query model discussed above. Let $x_{j,i}$ denote the value of x_i returned on the jth query to the oracle. We first prove that the amplitudes of the final state after t queries are block-multilinear in $x_{1,1}, \ldots, x_{t,N}$. We prove this using induction on t.

When $t = 0$, the amplitudes are simply given by constant polynomials. Let the result hold for $t-1$ queries. So now, the state of the qubits after the gate U_{t-1} is given by

$$\sum_{i,w} a_{i,w}(x_{1,1}, \ldots, x_{t-1,N}) |i\rangle |w\rangle.$$

After applying O_x, the state is

$$\sum_{i,w} x_{t,i} a_{i,w}(x_{1,1}, \ldots, x_{t-1,N}) |i\rangle |w\rangle.$$

It is easy to see that the amplitudes are still block-multilinear, now with t blocks, even after the linear transformation U_t. The squares of the amplitudes are also block-multilinear, but with $2t$ blocks. We simply introduce a duplicate variable $x_{t+j,i}$ for each $x_{j,i}$ and then square amplitudes as shown below. The final polynomial $p(\bar{x})$ is given by

$$\sum_{i,w \in Acc} |a_{i,w}(x_{1,1}, \ldots, x_{t,N})|^2 = \sum_{i,w \in Acc} a^*_{i,w}(x_{1,1}, \ldots, x_{t,N}) a_{i,w}(x_{t+1,1}, \ldots, x_{2t,N})$$

which is clearly block multilinear. The probability that the algorithm accepts is simply $p(x, x, \ldots)$, which means that the norm of the vector of accepted state amplitudes is at most 1. Thus, $p(\bar{x})$ being an inner product lies in $[-1,1]$. $\square$

2.4 Forrelation

Finally, we define Forrelation. Intuitively, it is a measure of the correlation between a function f and the fourier transform of a second function g. Given oracle access to two boolean functions $f, g\colon \{0,1\}^n \to \{-1,1\}$, let

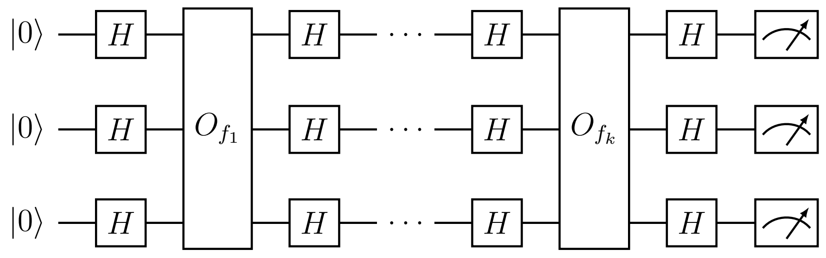
$$\Phi_{f,g} := \frac{1}{2^{3n/2}} \sum_{x,y \in \{0,1\}^n} f(x)(-1)^{x \cdot y} g(y). \tag{2.1}$$

The problem is to decide whether $\Phi_{f,g} \geq 0.6$ or $|\Phi_{\mathrm{f,g}}| \leq 0.01$, and we are promised that one of these is the case.

k-fold Forrelation is a more general problem, where we are given oracle access to k boolean functions $f_1, \ldots, f_k\colon \{0,1\}^n \to \{-1,1\}$ and

$$\Phi_{f_1,\ldots,f_k} := \frac{1}{2^{(k+1)n/2}} \sum_{x_1,\ldots,x_k \in \{0,1\}^n} f_1(x_1)(-1)^{x_1 \cdot x_2} f_2(x_2) \ldots (-1)^{x_{k-1} \cdot x_k} f_k(x_k) \tag{2.2}$$

is promised to be either at least 0.6 or at most 0.01 in magnitude. The problem is again to decide which is the case. This problem can also be defined using the following circuit, which solves it in k queries.



The Hadamard gate H is a linear operation that transforms $|0\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle$ to $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. It can be shown that $H^{\otimes n}$ corresponds to a Fourier transform. It can hence also be shown that the probability of finally measuring $|0\rangle^{\otimes n}$ in the above circuit is exactly $\Phi_{f_1,\ldots,f_k}$.

As shown in [1], the complexity can be further improved to $\lfloor k/2 \rfloor$ queries. It is also shown later that Forrelation takes at least $\Omega(\sqrt{N}/\log N)$ classical queries to solve, where $N = 2^n$. There is thus a gap of $\Omega(\sqrt{N}/\log N)$ between classical and quantum complexities for this problem. Interestingly, the block-multilinear polynomial corresponding to this problem can be used to solve it in $O(\sqrt{N})$ classical queries, meaning that the gap is indeed optimal. We will go through this proof in the last chapter.

Chapter 3

Block-multilinear Degree

In this chapter, we try comparing the exact block-multilinear degree of a boolean function to its degree. It is easy to see that the former is at least equal to the latter. We are interested in the gap between the two and want to find a function f for which the inequality is strict, or that $\mathrm{bmdeg}_\epsilon(f) > \deg_\epsilon(f)$.

Aaronson et al. [2] made a construction from any given polynomial $p\colon \{-1,1\}^N \to [-1,1]$ of degree d to a d-block-multilinear polynomial $\tilde{p}\colon \{-1,1\}^{(N+1)d} \to [-c_d, c_d]$ such that we have $\tilde{p}(1,x,\ldots,1,x) = p(x)$ for all $x \in \{-1,1\}^N$ and c_d is a constant depending only on d. We take a similar approach and attempt to construct a block-multilinear polynomial by symmetrically splitting the coefficients of a given polynomial. We also try a second approach which involves finding a dual witness for a suitable linear program.

3.1 Symmetrization

Suppose we are given the exact polynomial representation of $f\colon \{-1,1\}^N \to \{-1,1\}$, of degree d. In the Fourier basis, this is given by

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S)\chi_S(x) \tag{3.1}$$

where $\chi_S(x) = \prod_{i\in S} x_i$ is the Sth Fourier function. Let us equally split the coefficients of each χ_S into coefficients of corresponding monomials of block-multilinear form. Let

$$S_{(i_1,\ldots,i_k)} := \{i\colon i = i_j \text{ for an odd number of } j\} \setminus \{0\}, \tag{3.2}$$

$$I_S := \{m \in \{0,1,\ldots,n\}^d \colon S_m = S\}. \tag{3.3}$$

Then $f_{sym}\colon \{-1,1\}^{(N+1)d} \to \mathbb{R}$ is defined as

$$f_{sym}(\bar{x}) := \sum_{S\subseteq[n]} \sum_{m\in I_S} \frac{\hat{f}(S)}{|I_S|}\chi_m(\bar{x}) \tag{3.4}$$

where $\bar{x} \in \{-1,1\}^{(N+1)d}$. We can alternatively define $\tilde{f}_{sym}$ using

$$\tilde{S}_{(i_1,\ldots,i_k)} := \{i\colon i = i_j \text{ for exactly one } j\} \setminus \{0\}. \tag{3.5}$$

Notice that for any $x \in \{-1,1\}^N$, we have

$$\chi_m(1,x,\ldots,1,x) = \chi_{S_m}(x) \tag{3.6}$$

since S_m consists of only those i which occur an odd number of times in m. If i occurs an even number of times in m, then x_i gets squared up in $\chi_m(1,x,\ldots,1,x)$ and so does not appear in $\chi_{S_m}(x)$. Hence, both the constructions satisfy

$$f_{sym}(1,x,\ldots,1,x) = f(x) \tag{3.7}$$

for all $x \in \{-1,1\}^N$. The only thing that is left to check is whether this polynomial is bounded in $[-1,1]$ on all $\bar{x} \in \{-1,1\}^{(N+1)d}$. To do this, it is enough to find the minimum value attained by f_{sym} since it takes identical values on both sides of 0. If this value is at least -1, then f_{sym} is bounded in [-1,1].

3.1.1 A Counter-example

Given a function $f\colon \{-1,1\}^n \to \mathbb{R}$

$$f(x) = \sum_{S\subseteq[n]} a_S \prod_{i\in S} x_i$$

where $x \in \{-1,1\}^n$, the minimum value it attains is given by the following Integer Linear Program:

$$\begin{aligned}
\min\ & \sum_{S\subseteq[n]} a_S(1-2y_S) \\
\text{s.t. } & y_S + 2z_S = \sum_{i\in S} x_i \ \forall S \subseteq [n] \\
& y_S \in \{0,1\} \ \forall S \subseteq [n] \\
& x_i \in \{0,1\} \ \forall i \in [n] \\
& z_S \in \mathbb{Z}.
\end{aligned}$$

Using this ILP to check f_{sym} for all $f\colon \{-1,1\}^N \to \{-1,1\}$ of degree 2 and 3, we found the following counter-example:

$$f(x_1,x_2,x_3) = -0.5 + 0.5x_1x_2 - 0.5x_1x_3 - 0.5x_2x_3.$$

Here, $d = 2$ and $N = 3$. After symmetrization, we get

$$\begin{aligned}
f_{sym}(\bar{x}) = & -0.125x_{10}x_{20} - 0.125x_{11}x_{21} - 0.125x_{12}x_{22} - 0.125x_{13}x_{23} \\
& + 0.25x_{11}x_{22} + 0.25x_{12}x_{21} \\
& - 0.25x_{11}x_{23} - 0.25x_{13}x_{21} - 0.25x_{12}x_{23} - 0.25x_{13}x_{22}
\end{aligned}$$

where $\bar{x} \in \{-1,1\}^8$. This function takes value -1.25 at the following valuation:

$$\begin{aligned}
x_{10} &= -1.0 & x_{20} &= -1.0 \\
x_{11} &= 1.0 & x_{21} &= -1.0 \\
x_{12} &= 1.0 & x_{22} &= -1.0 \\
x_{13} &= -1.0 & x_{23} &= 1.0.
\end{aligned}$$

The alternative construction gives

$$\begin{aligned}\tilde{f}_{sym}(\bar{x}) = &- 0.5x_{10}x_{20} + 0.25x_{11}x_{22} + 0.25x_{12}x_{21} - 0.25x_{11}x_{23}\\ &- 0.25x_{13}x_{21} - 0.25x_{12}x_{23} - 0.25x_{13}x_{22}\end{aligned}$$

where $\bar{x} \in \{-1,1\}^8$. This function takes value -2 at the following valuation:

$$\begin{aligned} x_{10} &= -1.0 & x_{20} &= -1.0\\ x_{11} &= 1.0 & x_{21} &= -1.0\\ x_{12} &= 1.0 & x_{22} &= -1.0\\ x_{13} &= -1.0 & x_{23} &= 1.0. \end{aligned}$$

Thus, unfortunately, our construction is incorrect. The program we used to find the counter-example can be accessed here, and for the alternative construction can be accessed here.

3.2 Dual Witness

We now try another approach. Consider the linear program given below, where we find the best possible approximation of a function $f\colon \{-1,1\}^N \to \{0,1\}$ using a block-multilinear polynomial g of degree $d = \deg(f)$. If the value of its dual is strictly greater than ϵ_0, then by weak duality, the value of the primal is greater than ϵ_0 as well, and thus $\mathrm{bmdeg}_{\epsilon_0}(f) > d$.

$$\begin{aligned} &\min \epsilon\\ &\text{s.t. } f(x) - g(x,\dots,x) \le \epsilon \ \forall x\\ &\qquad g(x,\dots,x) - f(x) \le \epsilon \ \forall x\\ &\qquad g(\bar{x}) \le 1 \ \forall \bar{x}\\ &\qquad -1 \le g(\bar{x}) \ \forall \bar{x}\end{aligned}$$

The variables here are ϵ and the coefficients of g. The dual for this program is

$$\begin{aligned} &\max F(\phi,\psi_1,\psi_2) := \sum_x \phi(x)f(x) - \sum_{\bar{x}}(\psi_1(\bar{x}) + \psi_2(\bar{x}))\\ &\text{s.t. } \left|\sum_x \phi(x)\right| = 1\\ &\qquad \psi_1(\bar{x}), \psi_2(\bar{x}) \ge 0 \ \forall \bar{x}\\ &\qquad \hat{\psi}_1(m) - \hat{\psi}_2(m) = \frac{N}{2^{(N+1)d}}\hat{\phi}(S_m) \ \forall m \in \{0,\dots,N\}^d\end{aligned}$$

where the vectors ϕ, ψ_1 and ψ_2 are the variables. If there exists a feasible solution (ϕ, ψ_1, ψ_2) such that $F(\phi,\psi_1,\psi_2) > \epsilon_0$, then the value of the dual is greater than ϵ_0 and hence $\mathrm{bmdeg}_{\epsilon_0}(f) > d$. The converse holds as well by strong duality. The next result follows.

Theorem 2. *Let $f\colon \{-1,1\}^N \to \{0,1\}$ have degree d. Then $\mathrm{bmdeg}_0(f) > d$ if and only if there exist $\phi\colon \{-1,1\}^N \to \mathbb{R}$ and $\psi_1,\psi_2\colon \{-1,1\}^{(N+1)d} \to \mathbb{R}^+$ such that*

1. $\sum_x \phi(x)f(x) \ge \sum_{\bar{x}}(\psi_1(\bar{x}) + \psi_2(\bar{x}))$.
2. $\hat{\psi}_1(m) - \hat{\psi}_2(m) = \frac{N}{2^{(N+1)d}}\hat{\phi}(S_m) \ \forall m \in \{0,\dots,N\}^d$.

Chapter 4

Classical-Quantum Gap

In this chapter, we give an overview of the two results in [1]. The results are stated below.

Theorem 3. *Any classical randomized algorithm for Forrelation must make* $\Omega(\frac{\sqrt{N}}{\log N})$ *queries.*

Theorem 4. *Let* Q *be any quantum algorithm that makes* $t = O(1)$ *queries to an* N*-bit string* $X \in \{0,1\}^N$*. Then we can estimate* $\Pr[Q \text{ accepts } x]$*, to constant additive error and with high probability, by making only* $O(N^{1-1/2t})$ *classical randomized queries to* X*. Moreover, the randomized queries are nonadaptive.*

4.1 Randomized Lower Bound

We saw earlier that Forrelation can be solved using $O(1)$ queries. The result states that any classical randomized algorithm for must make $\Omega(\frac{\sqrt{N}}{\log N})$ queries, therefore implying a separation of order $\Omega(\frac{\sqrt{N}}{\log N})$ between quantum and classical complexities. In this section, we give a brief overview of this proof.

4.1.1 Real Forrelation

The first step to prove the randomized lower bound is to convert the Forrelation problem into a Real Forrelation problem. In Real Forrelation, we are given oracle access to two real functions $f, g\colon \{0,1\}^n \to \mathbb{R}$ and are promised that either

1. every f(x) and g(y) value is an independent $\mathcal{N}(0,1)$ Gaussian, or else
2. every f(x) value is an independent $\mathcal{N}(0,1)$ Gaussian and every g(y) value equals $\hat{f}(y)$ (i.e. the Fourier transform of f evaluated at y).

The problem is to decide which holds.

It can be proved that if there exists a T-query algorithm that solves Forrelation with bounded error, then there also exists an $O(T)$-query algorithm that solves Real Forrelation with bounded error. The details of this reduction can be found in [1]. Thus, we only need to prove a lower bound on Real Forrelation.

4.1.2 Gaussian Distinguishing

In the Gaussian Distinguishing problem, we are given oracle access to a collection of $\mathcal{N}(0,1)$ real Gaussian variables $x_1, x_2..., x_m$ and are asked to decide whether

1. the variables are all independent, or

2. the variables lie in a known low dimensional subspace $S \leq R^m$ such that there is a covariance of at most ϵ between each pair of variables, i.e., $\mid Cov(x_i, x_j) \mid \leq \epsilon \ \forall i, j$.

We finally have the following result for Gaussian Distinguishing in [1].

Theorem 5. *Gaussian distinguishing requires $\Omega\left(\frac{1/\varepsilon}{\log(M/\varepsilon)}\right)$ classical randomized queries.*

The proof depends on the intuition that because the real gaussian variables have restricted covariances and thus are nearly orthogonal. So, if we restrict our attention to any subset S of R^m, their correlations are weak i.e. the variables satisfy an "orthogonal approximation". Now, to solve the Gaussian Distinguishing problem, we need to assess the number of queries required till the "orthogonal approximation" breaks down.

Using Gram-Schmidt orthogonalization and Gaussian Azuma's inequality, it can be shown that that the first $\Omega\left(\frac{1/\varepsilon}{\log(M/\varepsilon)}\right)$ query responses are close to independent Gaussians and thus Gaussian distinguishing requires at least that many queries. Thus, using this bound on Gaussian Distinguishing, and setting M to $2N$ and ϵ to $1/\sqrt{N}$, we get the lower bound on Real Forrelation.

4.2 Optimal Randomized Algorithm

Previously, we stated that solving Forrelation requires just one quantum query. This single-query quantum algorithm can in fact be converted to a $\sqrt{N}$-query randomized algorithm, by approximating the block-multilinear polynomial corresponding to the algorithm. In this section, we present a rephrased proof of this result from [1].

Theorem 6. *Every degree-k polynomial $p\colon \{-1,1\}^N \to \mathbb{R}$ that is*

1. *bounded in $[-1,1]$ at every Boolean point, and*

2. *"block-multilinear" (that is, the variables can be partitioned into k blocks, such that every monomial is the product of one variable from each block),*

can be approximated to within $\pm\epsilon$, with high probability, by nonadaptively querying only $O\left(N^{1-1/2t}\right)$ of the variables.

4.2.1 The Estimator

Given $x \in \{-1,1\}^{(n+1)k}$, we need to estimate the block-multilinear polynomial

$$p(x) := \sum_{i_1,\ldots,i_k} a_{i_1,\ldots,i_k} x_{1,i_1} \ldots x_{k,i_k}. \tag{4.1}$$

We define the estimator as

$$P := n \sum_{i_1,\dots,i_k} y_{1,i_1} \dots y_{k,i_k} b_{i_1,\dots,i_k}(x) \tag{4.2}$$

where

$$b_{i_1,\dots,i_k}(x) := a_{i_1,\dots,i_k} x_{1,i_1} \dots x_{k,i_k} \tag{4.3}$$

and the random variables $y_{j,i}$ are independent and take values $\{0, 1\}$ with $\Pr[y_{j,i} = 1] = \frac{1}{n^{1/k}}$. By linearity of expectation, we have

$$\mathrm{E}[P] = p(x). \tag{4.4}$$

Now let us say we sample P m times. We want m to be $O(1)$. Let $\tilde{P}$ be the mean of this sample. By the strong law of large numbers, $\tilde{P}$ converges to $p(x)$ with variance $\mathrm{Var}[P]/m$. By Chebyshev's inequality, we have

$$\Pr[\ |\tilde{P} - p(x)| \le \epsilon\ ] \le \frac{\mathrm{Var}[P]}{m\epsilon^2}. \tag{4.5}$$

Hence, for convergence with high probability in $m = O(1)$ steps, $\mathrm{Var}[P]$ must be $O(1)$ as well. This is what we will prove next. Each step requires us to query only those $x_{j.i}$ for which $y_{j,i} = 1$. This is on expectation $n/n^{1/k}$ (by linearity of expectation), that too with high probability (by the Chernoff bound). Thus, the algorithm requires $O(n^{1-1/k})$ queries.

4.2.2 Bound on Variance

We give a simplified version of the proof in [1].

$$\begin{aligned}
\mathrm{Var}[P] &= \sum_{i_1,\dots,i_k,i'_1,\dots,i'_k \in [n]} b_{i_1,\dots,i_k}(x) \cdot b_{i'_1,\dots,i'_k}(x) \cdot \mathrm{Cov}[y_{1i_1} \dots y_{k,i_k}, y_{1,i'_1} \dots y_{k,i'_k}] \cdot n^2 \\
&= \sum_{S \subseteq [k]} \left(\sum_{i,i' :\, i_j = i'_j \iff j \in S} b_{i_1,\dots,i_k}(x) \cdot b_{i'_1,\dots,i'_k}(x) \left(\frac{n^{|S|/k}}{n^2} - \frac{1}{n^2} \right) n^2 \right) \\
&= \sum_{S \subseteq [k]} \left((n^{|S|/k} - 1) \sum_{i,i' :\, i_j = i'_j \iff j \in S} b_{i_1,\dots,i_k}(x) \cdot b_{i'_1,\dots,i'_k}(x) \right) \\
&= \sum_{S \subseteq [k]} \left(\sum_{T \subseteq S} (-1)^{|S|-|T|} (n^{|T|/k} - 1) \right) \sum_{i,i' :\, j \in S \implies i_j = i'_j} b_{i_1,\dots,i_k}(x) \cdot b_{i'_1,\dots,i'_k}(x)
\end{aligned}$$

The last two steps follow from the inclusion-exclusion principle. Let I_S be the set of all $i_1, \dots, i_k, i'_1, \dots i'_k$ such that $i_j = i'_j$ for at least $j \in S$, and J_S be such that $i_j = i'_j$ if and only if $j \in S$. By inclusion-exclusion, we have

$$|J_S| = |I_S| - \left(\sum_{T \supset S :\, |T| = |S|+1} |I_T| \right) + \left(\sum_{T \supset S :\, |T| = |S|+2} |I_T| \right) \dots,$$

so multiplying each equation on both sides by $(n^{|S|/k}-1)$ and adding them together gives us the desired result. The coefficient of $|I_T|$ in the RHS is equal to $\sum_{S\subseteq T}(-1)^{|T|-|S|}(n^{|S|/k}-1)$. Let us now define the quantity

$$\Lambda_S(x) := \sum_{(i_j)_{j\in S}} \left(\sum_{(i_j)_{j\notin S}} b_{i_1,\dots,i_k}(x) \right)^2 \tag{4.6}$$

and let $\Lambda_S \leq \delta$ for each S. Then, we have

$$\mathrm{Var}[P] = \sum_{S\subseteq[k]} \left(\sum_{T\subseteq S} O(n^{|T|/k}) \right) \cdot O(\delta) = O(\delta n).$$

Hence, δ must be $O(1/n)$, say ϵ^2/n, for some ϵ. So for any x, $\Lambda_S(x) \leq \epsilon^2/n$ for all S. The initial polynomial should be preprocessed so that this holds.

4.2.3 Preprocessing

The approach is to prove that for any $S \subseteq [k]$,

$$\sum_{(i_j)_{j\in S}} \left(\sum_{(i_j)_{j\notin S}} a_{i_1,\dots,i_k} \right)^2 \leq \delta.$$

The method used to prove this is to split each variable $x_{j,i}$ into some m new variables and replace it with $(x_{j,i'_1} + \ldots + x_{j,i'_m})/m$. We initially have the k-block-multilinear polynomial

$$p(x) := \sum_{i_1,\dots,i_k} a_{i_1,\dots,i_k} x_{1,i_1} \dots x_{k,i_k} \tag{4.7}$$

where $i_j \in [N]\ \forall j \in [k]$. First, we want to show that we can split the variables so that

$$\Lambda_{[k]} = \sum_{(i_j)_{j\in[k]}} a^2_{i_1,\dots,i_k} \leq \delta.$$

Next, if $S = \{j_1, \dots, j_l\} \neq [k]$, we put $x_{j,i} = 1$ for $j \notin S$ in the polynomial. Then we have

$$\tilde{p}(x_{j_1,1}, \dots, x_{j_l,N}) = \sum_{i_{j_1},\dots,i_{j_l}} a'_{i_{j_1},\dots,i_{j_l}} x_{j_1,i_{j_1}} \dots x_{j_l,i_{j_l}}$$

where $a'_{i_{j_1},\dots,i_{j_l}} = \sum_{(i_j)_{j\notin S}} a_{i_1,\dots,i_k}$. Hence, after splitting the new polynomial $\tilde{p}$,

$$\Lambda_S = \sum_{(i_j)_{j\in S}} \left(\sum_{(i_j)_{j\notin S}} a_{i_1,\dots,i_k} \right)^2 = \sum_{i_{j_1},\dots,i_{j_l}} (a'_{i_{j_1},\dots,i_{j_l}})^2 \leq \delta.$$

Since each splitting only either decreases Λ_S for each S or keeps it constant, we can repeat it for each subset S. If each splitting introduces m new variables, we eventually have $2^k m$ new variables. Since the total number of variables n per block should stay $O(N)$, m should be $O(N)$ or $O(1/\delta)$.

Lemma 1. *We can introduce $O(1/\delta)$ new variables by variable splitting, such that for the resulting polynomial, we have*

$$\sum_{(i_j)_{j\in[k]}} a^2_{i_1,\ldots,i_k} \le \delta.$$

The above lemma from [1] completes the proof. However, we found a small inaccuracy. Suppose the lemma is correct. We split each $x_{j,i}$ into $1+m_{j,i}$ variables. Let $m_{j,i} = \lfloor f_j(i)/\delta \rfloor$. Then we have

$$\delta m_{j,i} \le f_j(i) \le \delta(1+m_{j,i}).$$

The new sum of coefficients squared is

$$\sum_{(i_j)_{j\in[k]}} \frac{a^2_{i_1,\ldots,i_k}}{(1+m_{1,i_1})\ldots(1+m_{k,i_k})} \le \delta^k \sum_{(i_j)_{j\in[k]}} \frac{a^2_{i_1,\ldots,i_k}}{f_1(i_1)\ldots f_k(i_k)}$$

and it must be at most δ. Hence, we must have

$$\frac{1}{N^{k-1}} \sum_{(i_j)_{j\in[k]}} \frac{a^2_{i_1,\ldots,i_k}}{f_1(i_1)\ldots f_k(i_k)} \le \frac{1}{(\delta N)^{k-1}} = O(1).$$

Also, the total number of new variables is $\sum_{j,i} m_{j,i} \le (1/\delta)\sum_{j,i} f_j(i)$. For this to be $O(1/\delta)$, we must also have $\sum_{j,i} f_j(i) = O(1)$. Then

$$\frac{\frac{1}{N^{k-1}}\left(\sum_{(i_j)_{j\in[k]}} \frac{a^2_{i_1,\ldots,i_k}}{f_1(i_1)\ldots f_k(i_k)} + \sum_{j,i} N^{k-1} f_j(i)\right)}{(1+k)\cdot N^k} \ge \frac{1}{N^{k-1}}\left(\prod_{(i_j)_{j\in[k]}} a^2_{i_1,\ldots,i_k}\right)^{\frac{1}{(1+k)\cdot N^k}}$$

using the AM-GM inequality. If $a^2_{i_1,\ldots,i_k} = \frac{1}{N^k}$ for each $(i_1,\ldots,i_k)$, then we have

$$O(1) \ge (1+k)\cdot N^{\frac{1}{1+k}}$$

which is a contradiction. We conjecture that this is possible even though $p(x)$ has to be bounded in [-1,1] on all x.

Chapter 5

Conclusion

We discussed the notion of block-multilinear degree, and compared it to the general degree. We tried to check if deg and $\text{bmdeg}_0(f)$ are actually equivalent, by trying to construct a block-multilinear f_{sym} from any given f. However, we found that the construction was incorrect since it was not bounded in [-1,1]. It is still not clear if $\deg(f)$ and $\text{bmdeg}_0(f)$ are separate, since we only showed unboundedness for symmetric block-multilinear polynomials. However, we now also have a condition for separating $\deg(f)$ and $\text{bmdeg}_0(f)$ using the dual witness.

5.1 Future Work

There is a lot of scope for future work related to block-multilinear degree. We can try constructing block-multilinear polynomials that are not-symmetric and more general to see if they are bounded. If we can find an unbounding input for every such polynomial, this will imply that $\text{bmdeg}_0(f) > \deg(f)$. We could also try computing a dual witness as discussed in Chapter 3 to show the same.

Bibliography

[1] S. Aaronson and A. Ambainis, “Forrelation: A problem that optimally separates quantum from classical computing,” 2014.

[2] S. Aaronson, A. Ambainis, J. Iraids, M. Kokainis, and J. Smotrovs, “Polynomials, quantum query complexity, and grothendieck's inequality,” 2015.