Block Multilinear Degree

Akash Kumar Singh, Siddhant Kar, Snehal Raj

Supervisor: Rajat Mittal

June 2020

Introduction

One of the most useful models used 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 quantum algorithm based on the quantum query model involves querying a boolean function x repeatedly so that some property of x can be checked at the end. This property can be thought of as calculating another function f on input x. The minimum number of queries that such any algorithm must make 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. The minimum degree that any such polynomial approximating f up to ϵ can attain is called the ϵ-approximate degree of f. Thus, $2Q_f$ is at least equal to the approximate degree of f.

A new notion was proposed by Aaronson et al. [**?**], where they approximated f using a very specific kind of polynomial of degree $2Q_f$, called a block-multilinear polynomial. The minimum degree attained by any such polynomial approximating f up to ϵ is called the ϵ-block-multilinear degree. It is easy to see that the ϵ-approximate block-multilinear degree is at least equal to the ϵ-approximate degree. This notion of block-multilinear degree is thus closer to query complexity than degree. Aaronson et al. [**?**] showed an equivalence between polynomials of degree 2 that approximate f (block-multilinear or not) and 1-query quantum algorithms for f, by constructing a 1-query algorithm from any degree 2 polynomial. However, there is a gap between Q_f and the block-multilinear degree of f for higher degrees [**?**].

We are mainly interested in the gap between the approximate degree and block-multiliear degree, as well as the exact degree and block-multilinear degree. The exact (block multilinear) degree is simply the 0-approximate (block multilinear) degree. The exact approximation of f is nothing but the representation of f in the Fourier basis.

The notion of block-multilinear polynomials was also used in [**?**] to solve a problem called Forrelation classically. Forrelation has achieved the largest gap between quantum and classical query complexities known yet among promise problems. A *promise* problem is a problem where we are promised some condition on the input function x and thus calculate $f(x)$ accordingly. For example, Simon's problem is a promise problem whereas Grover's total search is a *total* problem.

Chapter 1

Preliminaries

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

1.1 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 independent 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 form quantum *circuits*. With this, we can now talk about the quantum query model.

1.2 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.

$$\begin{array}{cc} [\text{wires=2}]i & [\text{wires=3}]\text{O}_x[wires = 2]\text{i} \\ b \quad b \oplus x_i & [\text{wires=2}]i \quad [\text{wires=3}]\text{O}_x[wires = 3]\text{x}_i ib \\ & b \end{array}$$

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.

$$\begin{array}{c} [\text{wires=2}]i \quad [\text{wires=4}]\text{U}_0[wires = 4]O_x[wires = 4]U_1[wires = 4]O_x \ \ldots \ [wires = \\ 4]U_{t-1}[wires = 4]O_x[wires = 4]U_t \\ \ldots \\ [\text{wires=2}]w \quad \ldots \\ \ldots \end{array}$$

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.

1.3 Block-multilinear polynomial

Here, we formally define the notions of 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 into* 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* $\mathrm{bmdeg}_\epsilon(f)$*, 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*

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

The exact block-multilinear degree of f *is defined to be* $\mathrm{bmdeg}_0(f)$*.*

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 $bmdeg_\epsilon(f)$. The following theorem, with proof, is a rephrased version of the result in [**?**].

Theorem 1. *The probability that a quantum query algorithm of complexity* t *accepts a function* x *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, $p(\tilde{x})$ being an inner product lies in $[-1,1]$. □

1.3.1 Forrelation

Forrelation is a promise problem which measures the correlation between a function and the fourier transform of a second function. Given oracle access to two boolean functions $f, g\colon \{0,1\}^n \to \{-1,1\}$, let

$$\Phi_{f,g} := \frac{1}{2^{3n/2}} \sum_{x,y\in\{0,1\}^n} f(x)(-1)^{x\cdot y} g(y). \tag{1.1}$$

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

1.3.2 k-fold Forrelation

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

$$\Phi_{f_1,\ldots,f_k} := \frac{1}{2^{(k+1)n/2}} \sum_{x_1,\ldots,x_k\in\{0,1\}^n} f_1(x_1)(-1)^{x_1\cdot x_2} f_2(x_2)\ldots(-1)^{x_{k-1}\cdot x_k} f_k(x_k) \tag{1.2}$$

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.

Chapter 2

Block-multilinear Degree

In this chapter, we try comparing the block-multilinear degree of a boolean function, definied below, to its standard degree. We will first present a result related to approximate bmdeg and then analyse exact bmdeg.

2.1 bmdeg v/s deg

It is easy to see that for any boolean function f, $\deg_\epsilon(f) \leq \text{bmdeg}_\epsilon(f)$. We wish to get an idea of the other way around, i.e., how much larger is bmdeg than deg. To do this, we can try constructing a block-multilinear polynomial from a given general polynomial.

2.1.1 Approximate bmdeg

If we are given a polynomial approximation of f, we can construct a block-multilinear approximation as follows [**?**].

Theorem 2. *Let $p(x)$ be a polynomial of degree d such that $|p(x)| \leq 1$ for all $x \in \{-1,1\}^n$. Then there is a block-multilinear polynomial $\tilde{p}\colon R^{(n+1)d} \to R$ such that*

1. *$\tilde{p}(x,\ldots,x) = p(x)$ for any $x \in \{-1,1\}^n$.*
2. *$|\tilde{p}(\bar{x})| \leq C_d$ for any $\bar{x} \in \{-1,1\}^{(n+1)d}$ where C_d is a constant that depends only on d.*

Corollary. *Let $0 \leq \epsilon \leq \frac{1}{2}$ and $\epsilon' = \frac{1}{2} - \frac{1}{2C_d}\left(\frac{1}{2} - \epsilon\right)$. Then, $\deg_\epsilon(f) \leq d \implies \text{bmdeg}_{\epsilon'}(f) \leq d$.*

Proof. Suppose the degree d polynomial q approximates f up to ϵ. Let $p(x) = q(x) - \frac{1}{2}$ so that $|p(x)| \leq 1$ for all x. We then construct $\tilde{p}$ as shown above. Now, the polynomial $\frac{1}{2}\left(1 + \frac{\tilde{p}}{C_d}\right)$ approximates f up to the given ϵ', and is block-multilinear with degree d. □

2.1.2 Exact bmdeg

This is a construction we tried from the exact polynomial of $f\colon \{-1,1\}^n \to \{0,1\}$, of degree d. In the fourier basis representation,

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

We simply split the coefficients of each χ_S equally into coefficients of the corresponding monomials of block-multilinear form. Define

$$I_S := \{m \in \{0, 1, \ldots, n\}^d \colon S_m = S\} \tag{2.1}$$

where $S_m = \{i \colon i \text{ occurs in } m \text{ an odd number of times}\} \setminus \{0\}$. Our construction is

$$f_{sym}(\bar{x}) := \sum_{S \subseteq [n]} \sum_{m \in I_S} \frac{\hat{f}(S)}{|I_S|} \chi_m(\bar{x}) \tag{2.2}$$

where $\bar{x} \in \{-1, 1\}^{(n+1)d}$. This construction, unfortunately, is not bounded in $[-1, 1]$.

2.2 Dual Witness

It is easy to see that $\text{bmdeg}_\epsilon(f) \leq \text{bmdeg}_{\epsilon'}(f)$ if and only if $\epsilon \geq \epsilon'$. Consider the linear program given below, where we find the best possible approximation of f using a block-multilinear polynomial g of degree d. If the value of its dual is strictly greater than ϵ_0, then so is the value of the primal, and thus $\text{bmdeg}_{\epsilon_0}(f) > d$.

$$\begin{array}{ll} \min & \epsilon \\ \text{s.t.} & f(x) - g(x, \ldots, x) \leq \epsilon \ \forall x \\ & g(x, \ldots, x) - f(x) \leq \epsilon \ \forall x \\ & g(\bar{x}) \leq 1 \ \forall \bar{x} \\ & -1 \leq g(\bar{x}) \ \forall \bar{x} \end{array}$$

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

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

where the vectors ϕ, ψ_1 and ψ_2 are the variables.

Chapter 3

Classical-Quantum Gap

3.1 Quantum Upper Bound

We prove that k-fold Forrelation can be solved using a total of only $\lceil k/2 \rceil$ queries to the k corresponding oracles. Forrelation can hence be solved in a single query.

Theorem 3. *k-fold Forrelation can be solved using $\lceil k/2 \rceil$ queries to the corresponding oracles $O_{f_1}, \ldots, O_{f_k}$, and using $O(nk)$ quantum gates.*

Proof. The following circuit solves the problem in k queries. The probability of finally measuring $|0\rangle^{\otimes n}$ is exactly equal to $\Phi_{f_1,\ldots,f_k}$ from its definition.

$$\begin{array}{ccccccc} 0 & \mathrm{H} & [\text{wires=3}]\mathrm{O}_{f_1} H & \ldots & H[wires = 3]O_{f_k} H \\ & 0 & \mathrm{H} & \mathrm{H} & \ldots & \mathrm{H} & \mathrm{H} \\ & 0 & \mathrm{H} & \mathrm{H} & \ldots & \mathrm{H} & \mathrm{H} \end{array}$$

To improve the complexity to $\lceil k/2 \rceil$ queries, we introduce a control qubit c, which is set initially to $|+\rangle$, and also modify the oracles so that when $c = 0$, we get the sequence

$$0^{\otimes n} \quad \mathrm{H}^{\otimes n} O_{f_1} \quad \ldots \quad O_{f_{\lceil k/2 \rceil}} H^{\otimes n}$$

and when $c = 1$, we get the following sequence.

$$0^{\otimes n} \quad \mathrm{H}^{\otimes n} O_{f_k} \quad \ldots \quad H^{\otimes n} O_{f_{\lceil k/2 \rceil + 1}}$$

We finally measure in the basis $+, -$ and accept if the control qubit c is $+$. This ensures that the probability of accepting is

$$\frac{1 + \Phi_{f_1,\ldots,f_k}}{2}.$$

□

3.2 Randomized Lower Bound

As we showed eariler, Forrelation can be solved using $O(1)$ queries. We prove that any classical randomized algorithm 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 methods. This is also optimal, as we show later that Forrelation can be solved classically using $\sqrt{N}$ queries.

3.2.1 Real Forrelation

One of the first steps that the authors take 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.

Theorem 4. *Suppose $\langle f, g\rangle$ are drawn from the forrelated measure $\mathcal{F}$. Define boolean functions $F, G\colon \{0,1\}^n \to \{-1,1\}$ by $F(x) = \mathrm{sgn}(f(x))$ and $G(x) = \mathrm{sgn}(g(x))$. Then,*

$$\mathrm{E}_{f,g\sim\mathcal{F}}\left[\Phi_{F,G}\right] = \frac{2}{\pi} \pm O\left(\frac{\log N}{N}\right) \tag{3.1}$$

Corollary. *Suppose there exists a T-query algorithm that solves Forrelation with bounded error. Then there also exists and O(T)-query algorithm that solves real Forrelation with bounded error.*

3.2.2 Gaussian Distinguishing

Given Oracle access to a collection of $\mathcal{N}(0,1)$ real Gaussian variables $x_1, x_2..., x_m$, we are asked to decide whether

1. variables are all independent, or
2. 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. $\mid Cov(x_i, x_j) \mid \leq \epsilon \ \ \forall \ \ i, j$

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 they 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 atleast that many queries.

Now using the corollary of Theorem 1 and a more general result on *Gaussian Distinguishing* stated in Theorem 2 such that according to the given case M = 2N , $\epsilon = \frac{1}{\sqrt{N}}$, we prove the following theorem which gives us the Randomized lower bound on *Forrelation*

Theorem 6. *Any classical randomized algorithm for Forrelation must make* $\Omega\left(\frac{\sqrt{N}}{\log N}\right)$ *queries.*

3.3 Optimal Randomized Algorithm

In the previous sections, we showed that solving Forrelation requires at least $\Omega(\sqrt{N}/\log n)$ queries classically but just one quantum query. We now try to show that this 1-query quantum algorithm can be converted to a $\sqrt{N}$-query randomized algorithm. We do this by approximating the block-multilinear polynomial corresponding to the quantum algorithm. We have the following result [**?**].

Theorem 7. *Every degree-k polynomial* $p\colon \{-1,1\}^N \to \mathbb{R}$ *in* x *that is block-multilinear and bounded in* $[-1,1]$ *can be approximated to within* $\pm\epsilon$ *with high probability by querying* x *classically* $O\left(N^{1-1/2t}\right)$ *times.*

Corollary. *Let* Q *be any quantum algorithm that makes* $t = O(1)$ *queries to an oracle* O_x *where* $x \in \{0,1\}^N$. *Then we can estimate* $\Pr[Q \text{ accepts } x]$, *to constant additive error with high probability, by making only* $O\left(N^{1-1/2t}\right)$ *classical randomized queries to* x.

3.3.1 The Estimator

We are required to estimate the k-block-multilinear polynomial

$$p(x) := \sum_{i_1,\ldots,i_k} a_{i_1,\ldots,i_k} x_{1,i_1} \ldots x_{k,i_k} \tag{3.2}$$

where $i_j \in [n]\ \forall j \in [k]$. We define our estimator as

$$P := n \sum_{i_1,\ldots,i_k} y_{1,i_1} \ldots y_{k,i_k} b_{i_1,\ldots,i_k}(x) \tag{3.3}$$

where

$$b_{i_1,\ldots,i_k}(x) := a_{i_1,\ldots,i_k} x_{1,i_1} \ldots x_{k,i_k} \tag{3.4}$$

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

$$\mathrm{E}[P] = p(x). \tag{3.5}$$

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

$$\Pr[|\tilde{P} - p(x)| \leq \epsilon] \leq \frac{\mathrm{Var}[P]}{m\epsilon^2}. \tag{3.6}$$

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 $\frac{n}{n^{1/k}}$ by linearity of expectation, that too with high probability by the Chernoff bound. Thus, the algorithm requires $O(n^{1-1/k})$ queries.

3.3.2 Bound on Variance

We give a simpler version of the proof in [**?**].

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

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

$$|I_{=S}| = |I_{\geq S}| - \left(\sum_{T \supset S \colon |T| = |S|+1} |I_{\geq T}| \right) + \left(\sum_{T \supset S \colon |T| = |S|+2} |I_{\geq T}| \right) \ldots .$$

We multiply the equation on both sides by $(n^{|S|/k} - 1)$ for each S and add them all together. The coefficient of $|I_{\geq T}|$ in the RHS is thus

$$\sum_{S \subseteq T} (-1)^{|T|-|S|} (n^{|S|/k} - 1).$$

Let us now define the quantity

$$\Lambda_S(x) := \sum_{(i_j)_{j \in S}} \left(\sum_{(i_j)_{j \notin S}} b_{i_1,\ldots,i_k}(x) \right)^2 \tag{3.7}$$

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.

3.3.3 Preprocessing

We attempt to prove that for any $S \subseteq [k]$,

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

which is a much weaker result than $\Lambda_S(x) \leq \delta$ since $a_{i_1,\ldots,i_n} = b_{i_1,\ldots,i_k}(1,\ldots,1)$. 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

$$p(x) := \sum_{i_1,\ldots,i_k} a_{i_1,\ldots,i_k} x_{1,i_1} \ldots x_{k,i_k} \tag{3.8}$$

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

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

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

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

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

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

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*

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

This lemma is incorrect. Suppose we can do so. We split each $x_{j,i}$ into $1 + m_{j,i}$ variables. Let $m_{j,i} = \lfloor f_j(i)/\delta \rfloor$. Then,

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

The new sum of coefficients squared is

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

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

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

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

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

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

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

which is a contradiction.