Reconstruction of non-degenerate homogenous $\Sigma\Pi\Sigma\Lambda$ circuits

Omkar Bhalchandra Baraskar
Under the guidance of Prof. Chandan Saha and Prof. R. Venkatesh

Abstract

We develop a learning algorithm for non-degenerate n-variate degree d polynomial f which is compuatble by a $\Sigma\Pi^{[m]}\Sigma\Lambda^{[t]}$ circuit. The non-degeneracy conditions are stated in Section 1.4. Now f can be expressed as

$$f = \sum_{i=1}^{s} \prod_{j=1}^{m} Q_{ij}$$

where $Q_{ij} = c_{ij1}x_1^t + \cdots + c_{ijn}x_n^t$. The learning algorithm takes black box access to f as inputs and outputs black box access to $c_{ij'}Q_{ij'}$ where $j' = \sigma(j)$ where σ is some permutation, and $c_{ij'} \in \mathbb{F}^{\times}$. We also discuss a "weaker" notion of learning for the case of non-degenerate homogenous depth-4 circuit. A polynomial f computed by a homogenous depth-4 circuit can be expressed as

$$f = T_1 + \cdots + T_s, \quad T_i = \prod_{i=1}^{m} Q_{ij}$$

where Q_{ij} is degree-t homogeneous polynomial. In this "weaker" notion instead of black-box access to a polynomial f we are given black box access to $\langle T_1, \ldots, T_s \rangle$ and now the aim is to learn Q_{ij}'s.

1 Introduction

The following are the important problems in the field of Arithmetic circuits:

1. *Lower Bound:* find "hard" polynomials (i.e. polynomials which require super-polynomial sized circuits for computation) in some family of polynomials or circuit classes.

2. *PIT:* Given an arithmetic circuit computing a polynomial f, determine whether $f = 0$ or not?

3. *Reconstruction:* Given black box access to a polynomial f computable by a arithmetic circuit by circuit of size of s, we aim to find a poly(s) size circuit computing the polynomial f.[1]

1.1 Reconstruction Of Arithmetic Circuits

The problem of reconstruction is deeply interconnected to the problem of lower bounds and PIT (see section 1 of [2]), and these interconnections could help us design the learning algorithm. The design of all the learning algorithms[2] mentioned in this report is inspired by the deep connection between the lower bound and reconstruction problem. Reconstruction of the arithmetic circuit is believed to be a "hard" problem to solve, let alone the case of general polynomials; there are results which tell us that even reconstruction of depth 3 circuits is a hard problem (see Section 1.2 of [1]). Due to this, we resort to "average case reconstruction" which is described below.

1.2 Average Case Reconstruction

In *average case reconstruction*, we aim at learning a polynomial f that satisfies some conditions called the "non-degeneracy" conditions. Say f belongs to a circuit class C, now the key point is that we choose the non-degeneracy conditions such that if we randomly sample a polynomial from C (i.e., we sample from a large finite subset of C), then with a very high probability f is non-degenerate (i.e., it satisfies the non-degeneracy conditions). This report will mainly focus on average case learning of various circuit classes.

1.3 Previous Work

In this section, I will state the following two results and discuss the techniques used in them,

1. *Reconstruction of non-degenerate homogeneous depth 3 circuit*: Let $n, d, s \in \mathbb{N}$ and $s \leq (\frac{n}{d})^{\frac{d}{3}}$. Given black box access to n-variate degree d polynomial $f = T_1 + \cdots + T_s$, $T_i = l_{i1} \ldots l_{id}$ and l_{ij} is a linear form in n variables, and f satisfies the non-degeneracy conditions (stated in section 1.4) then in randomised

[1]There may be other notions of "reconstruction" in the literature, we will only stick with this here.
[2]Even though these are average case learning algorithms but still we can see how we can exploit these connections to get a learning algorithm

poly(n, d, s) time we get (with high probability) black box access to $c_{ij'}l_{ij'}$ where $j' = \sigma(j)$ for some permutation σ and $c_{ij'} \in \mathbb{F}^{\times}$, for every $i \in [s]$.[3]

2. *Learning sums of powers of low-degree polynomials in the non-degenerate*: Let $n, s, d, t \in \mathbb{N}$ s.t it satisfies relations in Corollary 1.1 of [2]. Given black box access to $f = \sum_{i=1}^{s} Q_i^m$ where Q_i's are homogeneous t degree polynomials and $mt = d$ and coefficients of Q_i's are selected at random from a set $S \subset \mathbb{F}$ s.t $|S| \geq (ns)^{150t}$, then in randomised poly$((ns)^t)$ time we get (with high probability) black box access to $c_{ij'}Q_{ij'}$'s where $i' = \sigma(i)$ for some permutation σ and $c_{ij'} \in \mathbb{F}^{\times}$.

For (ii) one can equivalently state the result in terms of "non-degeneracy conditions" (see Theorem 1 of [2]).
Now we will discuss a meta frame-work which helps us design learning algorithm from lower bound results. The design of meta-framework in itself is inspired from lower bound techniques (See section 1.2 of [2] for more details). Here we will discussing meta framework for polynomials f which can be expressed as $f = T_1 + \cdots + T_s$ where T_i's are "simple" polynomials in some sense [4].The meta-framework is divided into three parts:-

1. Choose family of linear operator $\mathcal{L}_1$, $\mathcal{L}_2$ s.t
   - $U = U_1 \oplus \cdots \oplus U_s$ where $U = \langle \mathcal{L}_1 \circ f \rangle$ and $U_i = \langle \mathcal{L}_1 \circ T_i \rangle$.
   - $V = V_1 \oplus \cdots \oplus V_s$ where $V = \langle \mathcal{L}_2 \circ U \rangle$ and $V_i = \langle \mathcal{L}_2 \circ U_i \rangle$.
2. We want to choose $\mathcal{L}_2$ s.t the above decomposition of U and V w.r.t $\mathcal{L}_2$ is unique (upto some permutation) and indecomposable (defined in section 2.4).
3. Recovery of T_i from $\langle \mathcal{L}_1 \circ T_i \rangle$

Note that, in the general setting, it is unlikely to find $\mathcal{L}_1, \mathcal{L}_2$ satisfying the conditions mentioned in the framework for all f belonging to some circuit class; we aim to choose them such that for a "random" f these conditions are satisfied with high probability. Once we are ready with the framework, i.e. we have found $\mathcal{L}_1$ and $\mathcal{L}_2$, which satisfy the above conditions. It is pretty easy to design a learning algorithm (see algorithm 1 of [2] for more details) for f. This meta-framework is used in [2] to design a learning algorithm for *sums of powers of low-degree polynomials in the non-degenerate case*; also, it can be used to learn *non-degenerate homogeneous depth three circuits.*

[3]After getting black box access to l_{ij}'s we can get a circuit computing f in poly(n, d, s) time
[4]For example: In case of homogeneous depth three circuits T_i's are just product of some linear forms

1.4 Our Results

A *powered homogeneous depth three circuits* is a circuit of form $\Sigma\Pi^{[m]}\Sigma\Lambda^{[t]}$ computing the polynomial of form $f = \sum_{i=1}^{s} \prod_{j=1}^{m} (c_{ij1}x_1^t + \cdots + c_{ijn}x_n^t)$ where $c_{ijk} \in \mathbb{F}$. We provide a learning algorithm for non-degenerate *powered homogeneous depth three circuits.* Let us first state the non-degeneracy conditions:-

Non-Degeneracy Conditions: We say a polynomial $f = \sum_{i=1}^{s} \prod_{j=1}^{m} (c_{ij1}x_1^t + \cdots + c_{ijn}x_n^t)$ computed by a $\Sigma\Pi^{[m]}\Sigma\Lambda^{[t]}$ circuit to be non-degenerate if the polynomial $g = \sum_{i=1}^{s} \prod_{j=1}^{m} (c_{ij1}x_1 + \cdots + c_{ijn}x_n)$ satisfies the following

1. $\dim(\mathcal{U}) = s\binom{m}{k}$ where $\mathcal{U} = \langle \partial^{=k}(g) \rangle$ and $k = \lfloor \frac{log(s)}{log(\frac{n}{ed})} \rfloor$.
2. For every i there exists $2k + 1$ linear forms $l_{ir_1}, \ldots, l_{ir_{2k+1}}$ s.t $\dim(\mathcal{U} \bmod \langle l_{ir_1}, \ldots, l_{ir_{2k+1}} \rangle) = (s - 1)\binom{m}{k}$ where $\langle l_{ir_1}, \ldots, l_{ir_{2k+1}} \rangle$ is the ideal generated by $l_{i_1}, \ldots, l_{i_{2k+1}}$, $l_{ij} = (c_{ij1}x_1 + \cdots + c_{ijn}x_n)$.

Note that f is non-degenerate w.r.t the above mentioned non-degeneracy conditions if and only if g is non-degenerate w.r.t non-degeneracy conditions mentioned in section 1.1 of [1]. Also note the notion of random choice is same for both f and g.

Lemma 1. Let f be as defined above. Now if we randomly choose c_{ijl}'s from a set $S \subseteq \mathbb{F}$ then f is non-degenerate with a probablity of at least $1 - \frac{2s^2d^3}{|S|}$.

Proof. By combining the above observation with result in appendix A of [1] we are done. $\square$

Theorem 1. Let $n, d, s, m \in \mathbb{N}$ and $\mathbb{F}$ be the underlying field, satisfying $s \leq (\frac{n}{m})^{\frac{m}{3}}$ and char$(\mathbb{F}) >$ max$(ms^2, t(m + 1))$ or 0. Given black box access to n-variate degree d polynomial $f = T_1 + \cdots + T_s$, where each $T_i = \prod_{j=1}^{m} Q_{ij}$ and $Q_{ij} = (c_{ij1}x_1^t + \cdots + c_{ijn}x_n^t)$ and f satisfies the non-degeneracy conditions,then in randomised poly(n, d, s) time we get black box access to $c_{ij'}Q_{ij'}$ where $j' = \sigma(j)$ for some permutation σ and $c_{ij'} \in \mathbb{F}^{\times}$, for every $i \in [s]$.

2 Preliminaries

In this section, we will be working with an underlying field $\mathbb{F}$ unless specified.

2.1 Notation

Let $\alpha = (a_1, \dots, a_n)$, $\mathbf{x} = \{x_1, \dots, x_n\}$ and $f(x_1, \dots, x_n)$ be a polynomial. Now we define

$$\mathbf{x}^\alpha = x_1^{a_1} \dots x_n^{a_n}$$
$$|\alpha| = a_1 + \dots + a_n$$
$$\frac{\partial^k f}{\partial \mathbf{x}^\alpha} = \frac{\partial^k f}{\partial x_1^{a_1} \partial x_2^{a_2} \dots \partial x_n^{a_n}}$$
$$\partial_{\mathbf{x}}^{=k} = \left\{ \frac{\partial^k f}{\partial \mathbf{x}^\alpha} \middle| \, |\alpha| = k \right\}$$

2.2 Arithemetic Circuits

An *Arithmetic Circuit* is a directed acyclic graph with nodes of in-degree 0 or more, and there is precisely one node with out-degree 0. The nodes with in-degree are called input nodes and are labelled by variables or constants (from $\mathbb{F}$); the node with out-degree 0 is called the output node. All nodes except the input nodes are labelled by $\{+, \times\}$. *Size* or an arithmetic circuit is defined as the number of edges in the graph. *Depth* of a node is defined as the length of the shortest path from that node to an input node, *depth* of an arithmetic circuit is defined as the depth of the output node.

An arithmetic circuit captures the computation of a polynomial. Σ, Π, Λ denote addition, multiplication and powering gates. A $\Sigma\Lambda^{[t]}$ denotes an arithmetic circuit with two layers where the first layer has powering gates of in-degree t and the second layer consist of addition gates of arbitrary in-degree; now, one can understand similar notations accordingly.

2.3 Black-Box

A *black box* of a polynomial $f(x_1, \dots, x_n)$ takes a n-tuple $(a_1, \dots, a_n)$ as input and outputs $f(a_1, \dots, a_n)$. We do not consider the computation time of a black box while analysing the time complexity of a algorithm, i.e we treat the computation as a instantaneous one.

Lemma 2. Given a black box to a n-variate degree d polynomial $f(x_1, \dots, x_n)$ we can black box access to all homogeneous components of f in $\mathcal{O}(d^3)$ time.

Proof. Multiply each variable by the variable z. Now the polynomial looks like $f = f^{[0]} + \dots + f^{[i]} z^i + \dots + f^{[d]} z^d$ where $f^{[i]}$ is the homogeneous polynomial with degree i. Now we evaluate f at d distinct values of z, so we get a d-tuple of polynomial from the evaluations, which can be expressed as the product of the vandermonde matrix on the d distinct evaluations times the row vector of the homogeneous components. By choice of the evaluations, we know that the matrix is invertible, so we can multiply it by its inverse[5] and get all the homogenous components. $\square$

Lemma 3. Given a n-variate degree d polynomial $f(x_1, \dots, x_n)$, we get black box to $\frac{\partial^k f}{\partial x^\alpha}$ in $\text{poly}(n, d^k)$ time.

Proof. We will first give the algorithm for $k = 1$, so let's say we are taking the derivative w.r.t x_1, then $f(x_1, x_2, \dots, x_n) = f_0 + f_1 x_1 + \dots + f_d x_1^d$ where f_i are polynomial in $x_2, \dots, x_n$. Again as we did in lemma 2 we can evaluate x_1 at d distinct and get all f_i's. As f_1 is the partial derivative of f w.r.t x_1 we are done. Now for $k > 1$ we can continue recursively. $\square$

2.4 Vector Space Decomposition

Given vector space U , V and a set of linear operator $\mathcal{L}$. Then a decomposition s.t $U = U_1 \oplus \dots \oplus U_s$,$V = V_1 \oplus \dots \oplus V_n$ and $\langle \mathcal{L} \circ U_i \rangle \subseteq V_i$ is called a *Vector Space Decomposition.* We say that a *Vector Space Decomposition* is indecomposable w.r.t to $\mathcal{L}$ if there do not exists $U_{i1}, U_{i2}, V_{i1}, V_{i2}$ satisfying $U_i = U_{i1} \oplus U_{i2}$, $V_i = V_{i1} \oplus V_{i2}$ and $\langle \mathcal{L} \circ U_{ij} \rangle \subseteq V_{ij}$ $\forall j \in [2]$.

2.5 Affine projections on partials(APP)

We have n-variate polynomial $f(x_1, x_2, \dots, x_n)$ and a n-tuple of linear forms $L = (l_i(z_1, \dots, z_{n_0}))_{i \in [n]}$. Say $\mathbf{x} = (x_1, \dots, x_n)$ and $\mathbf{z} = (z_1, \dots, z_{n_0})$. We define a projection map $\pi_L : \mathbb{F}[x] \to \mathbb{F}[z]$ as $\pi_L(f) = f(l_1(z), \dots, l_n(z))$, clearly this map is linear. If $S \subset \mathbb{F}[x]$ then we define $\pi_L(S) = \{\pi_L(f) : f \in S\}$. Then,

$$\text{APP}_{k,n_0}(f) = \max_L \ \dim \langle \pi_L(\partial_x^{=k} f) \rangle$$

APP measure was first introduced in [2] to tackle the following problem: We get very high dimension spaces when we use shifted partial measure, due to which we do not get direct sum decomposition(For more details, check appendix C and section 1.2.3 of [2]).

Lemma 4. If $|\mathbb{F}| \geq \alpha \, (d-k) \binom{n+k-1}{k}$ and every coefficient of every linear form in $L = (l_1(z), \dots, l_n(z))$ is selected at random from a $S \subset \mathbb{F}$ s.t $|S| = \alpha \, (d-k) \binom{n+k-1}{k}$. Then with a probability of atleast $1 - \frac{1}{\alpha}$ we get

$$\text{APP}_{k,n_0}(f) = \dim \langle \pi_L(\partial_x^{=k} f) \rangle$$

Proof. Let $r = \text{APP}_{k,n_0}$ We construct a matrix M where the rows are indexed by all k degree monomials and the columns are indexed by all $d-k$ degree monomials, so the (α, β)-th entry of M is the coefficient of β in $\frac{\partial^k f}{\partial \mathbf{x}^\alpha}$. Now, $\dim \langle \pi_L(\partial_x^{=k} f) \rangle = \text{rank}(M)$ so the question boils done to existence a $r \times r$ submatrix of M, and now by using *Schwartz-Zippel* lemma we done. $\square$

[5]We can fix the d evaluations before the computation so the inverse can be pre-calculated

3 Learning algorithm for non-degenerate $\Sigma\Pi^{[m]}\Sigma\Lambda^{[t]}$ circuits

In this section, I will state the algorithm to learn *powered homogeneous depth three circuits* and its correctness. However, I will first discuss the most crucial idea around which the algorithm is developed.

3.1 Reduction to homogeneous depth three circuits

Given a black box to $f = \sum_{i=1}^{s} T_i$ where $T_i = \prod_{j=1}^{m} Q_{ij}$ and $Q_{ij} = c_{ij1}x_1^t + \cdots + c_{ijn}x_n^t$, if we can get black box access to $g = \sum_{i=1}^{s} T_i'$ where $T_i' = \prod_{j=1}^{m} Q_{ij}'$ and $Q_{ij}' = c_{ij1}x_1 + \cdots + c_{ijn}x_n$ then by applying result 1 of section 1.3 we can learn Q_{ij}' hence learn Q_{ij}.

Naive Approach: What we want to do is treat x_i^t as a variable X_i, and then f w.r.t $X_1, \ldots, X_n$ can be computed by a homogeneous depth-3 circuit, but we do not necessarily have black-box access to $f(X_1, \ldots, X_n)$. To have black-box access to $f(X_1, \ldots, X_n)$ we must be able to evaluate it at any n-tuple of input. However, for that to happen, we need t-th radical of every element of $\mathbb{F}$, but this is not true for all fields; some important fields like $\mathbb{Q}$ do not satisfy these conditions.

Improved Approach: We will build now upon the "Naive Approach" by using formal power series expansion. We know that $P(x) = 1 + \sum_{i=0}^{\infty} \binom{\frac{1}{t}}{i} x^i = \sqrt[t]{1+x}$ [6], I will show that not all terms in the power series are relevant (The reader will understand what I mean by this in the next few arguments). Define $P_D(x)$ be the truncation of P at degree D now we have.

$$P(x) = P_D(x) + x^D(S) \quad S \in \mathbb{F}[[x]]$$
$$\implies P(x)^t = 1 + x = P_D(x)^t + x^D(S') \quad S' \in \mathbb{F}[[x]] \qquad (1)$$

Note that all the above computations were carried out in $\mathbb{F}[[x]]$. Now observe that as $1+x = P(x)^t$ and $P_D(x)^t$ are polynomials then S' has to be polynomial because it satisfies equation 1. Now fix $i \in [s]$, then

$$\begin{aligned} Q_{ij}(P_D(x_1), \ldots, P_D(x_n)) &= \sum_{l=1}^{n} c_{ijl}((1 + x_l) \\ &\quad - x_l^D(S')) \\ &= (\sum_{i=1}^{n} c_{ijl}x_l) + (\sum_{l=1}^{n} c_{ijl}) \\ &\quad + S_0 \end{aligned}$$

where S_0 has degree $\geq D$. If $D = m + 1$ then it can be easily observed that the degree m homogeneous polynomial of $T_i(P_D(x_1), \ldots, P_D(x_n)) = \prod_{j=1}^{m}(\sum_{l=1}^{n} c_{ijl}x_l) = \prod_{j=1}^{m} Q_{ij}'(x_1, \ldots, x_n) = T_i'(x_1, \ldots, x_n)$.Then the degree m homogeneous polynomial of $f(P_D(x_1), \ldots, P_D(x_n))$ is $g(x_1, \ldots, x_n)$.

As $P_D \in \mathbb{F}[x]$ thus every evaluation of P_D on $\mathbb{F}^n$ belongs to $\mathbb{F}$, hence if we have black box access to f then we get a black box access to $f \circ (P_D(x_1), \ldots, P_D(x_n))$ (here $\circ$ denotes composition) in $\mathcal{O}(n \cdot D \cdot log(D))$ time. Now by lemma 2 we can get degree m homogenous component of $f \circ (P_D(x_1), \ldots, P_D(x_n))$ in $\mathcal{O}((2 \cdot D \cdot m)^3)$. And as degree m homogenous component of $f \circ (P_D(x_1), \ldots, P_D(x_n))$ is $g(x_1, \ldots, x_n)$. Thus we can get black box of g given black box access to f in $\mathcal{O}(n \cdot D^4 \cdot log(D) \cdot m^3)$ time.

3.2 Learning Algorithm

I will now state the learning algorithm discussed in the previous section.

Algorithm: The algorithm takes a black box access to non-degenrate n-variate degree d polynomial $f = \sum_{i=1}^{s} T_i$ where $T_i = \prod_{j=1}^{m} Q_{ij}$ and $Q_{ij} = c_{ij1}x_1^t + \cdots + c_{ijn}x_n^t$ as input and outputs black box access to $Q_{ij'}$ where $j' = \sigma(j)$ for some permutation σ for every $i \in [s]$.

1. Get black box access to $g = \sum_{i=1}^{s} T_i'$ where $T_i' = \prod_{j=1}^{m} Q_{ij}'$ and $Q_{ij}' = c_{ij1}x_1 + \cdots + c_{ijn}x_n$.
2. By applying result 1 to $g(x_1, \ldots, x_n)$ we get black box access to $c_{ij'}Q_{ij'}'$ where $j' = \sigma(j)$ for some permutation σ and $c_{ij'} \in \mathbb{F}^{\times}$, for every $i \in [s]$.
3. Output black box of $c_{ij'}Q_{ij'}' \circ (x_1^t, \ldots, x_n^t)$ where $j' = \sigma(j)$, for every $i \in [s]$.

Step 1 of the algorithm can be done by discussion in the previous section. In step 2 one has take into account a minor caveat, to apply result 1 we need g to be a non-degenerate polynomial, one can easily observe that by definition of non-degeneracy condition of f (stated in section 1.4) it requires g to be non-degenerate. Now in step 3 we have black-box access to Q_{ij}' and we are composing it with $(x_1^t, \ldots, x_n^t)$ and it is easy to see that we can get the black-box access to $Q_{ij}' \circ (x_1^t, \ldots, x_n^t)$ in $\mathcal{O}(n \cdot log(D))$ time. So overall, the running time of the algorithm is randomised $poly(n, d, s)$. Also, note that all the relations on the parameters except the characteristic of the field are imposed by the learning algorithm of non-degenerate depth three circuits. An extra condition on the characteristic of the field is imposed, so the coefficients in the truncated formal power series are well defined.

[6] In $\mathbb{F}[[x]]$ this is defined as $P(x)^t = 1 + x$

The above learning algorithm can be easily modified to work for the following cases

- When t is unknown, we first set all the variables except x_1 to 0. Now we evaluate f at different values of x_1; since m is known to the learning algorithm, we get the value of t from these evaluations.

- When different variables have been raised to different powers i.e $Q_{ij} = (c_{ij1}x_1^{t_1} + \cdots + c_{ijn}x_n^{t_n})$ where $t_i \in \mathbb{N}$. One can independently substitute appropriate power series expansion depending on the power the variable is raised to.

4 Learning non-degenerate $\Sigma\Pi^{[m]}\Sigma\Pi^{[t]}$ circuits

APP measure gives us a lower bound for $\Sigma\Pi^{[m]}\Sigma\Pi^{[t]}$ (see theorem 2 of [2]). Hence we we want to use the meta-framework to design learning algorithm for it (stated in section 1.3). But, in this section we will working on a weaker notion of recontruction. So, let $f = \sum_{i=1}^{s} T_i$ where $T_i = \prod_{j=1}^{m} Q_{j1} \ldots Q_{jm}$, now instead of f we are given black box access to $U = \langle T_1, \ldots, T_s \rangle$ and our aim is to learn T_i. Now giving black box access to U reduces a lot of effort. So in step 1 of the meta-framework we just need to work one linear map $\mathcal{L}$ instead of two to get a vector space decomposition i.e we need to choose $\mathcal{L}$ s.t $\Sigma\langle \mathcal{L} \circ T_i \rangle = \bigoplus \langle \mathcal{L} \circ T_i \rangle$ and this will give us a vector space decomposition between U and $\langle \mathcal{L} \circ U \rangle$; also note that here $\langle \mathcal{L} \circ U \rangle = \Sigma\langle \mathcal{L} \circ T_i \rangle$ follows trivially. In step 2 we get that the the decomposition is trivially indecomposable as $\dim(T_i) = 1$, then only thing to work on is the uniqueness of the decomposition. Step 3 is totally scrapped out as we directly get access to T_i from vector space decomposition. So now to learn this model we need to

1. Choose[7] a $\mathcal{L}$ s.t $\Sigma\langle \mathcal{L} \circ T_i \rangle = \bigoplus \langle \mathcal{L} \circ T_i \rangle$.

2. Showing the uniqueness of vector space decomposition between U and $\langle \mathcal{L} \circ U \rangle$

Attempt to solve the step 1: As *APP* measure gave us the lower bound so from our formulation we expect for $\mathcal{L} = \text{APP}_{k,n_0}$ to be the right choice. In order to get the direct sum structure we started by exploring $V_i = \langle \mathcal{L} \circ T_i \rangle$, and tried to understand the structure of its basis and its dimension.

Let $\mathcal{S}$ denote the collection of all $m-k$ size subsets of $[m]$ Define $G_{ij} = \langle \pi_L(\partial_{\mathbf{x}}^{=k} Q_{ij}) \rangle$, $G_{iS} = G_{is_1} \cdots G_{is_{m-k}}$ where $S = \{s_1, \ldots, s_{m-k}\} \in \mathcal{S}$ and $s_1 < s_2 < \cdots < s_{m-k}$. Let $M_{d,n_0,\mathbf{x}} = \{\mathbf{x}^\alpha \mid \alpha \in \mathbb{N}^{n_0} \ |\alpha| = d\}$.
As $V_i \subseteq \mathbb{F} - \langle p \cdot G_{iS} \rangle$ where $S \in \mathcal{S}$ and $p \in M_{k(t-1),n_0,\mathbf{x}}$, so if we show $\sum \mathbb{F} - \langle p \cdot G_{iS} \rangle = \oplus \mathbb{F} - \langle p \cdot G_{iS} \rangle$ then 1 follows. (Note that here we have just stated the spanning set it is does not say anything about linear independence).

One obvious kind of dependency in this spanning set are of the following form: $p_1 \cdot G_{iS} = p_2 \cdot G_{iS'}$ where $S, S' \in \mathcal{S}$. Now this equation has a solution for p_1, p_2 (we have fixed the L_i 's) if and only if $\gcd(G_{iS}, G_{iS'}) \geq (m-k)t - k(t-1)$. Also note that in the average case know that if either $i \neq i'$ or $j \neq j'$ then $\gcd(G_{ij}, G_{i'j'}) = 1$ with a very high probablity, so in average case each product G_{iS} can be identified by S and the $\gcd(G_{iS}, G_{iS'}) = |S \cap S'|t$.

Also, we could not find any other dependencies. Keeping all this in mind, we conjectured the following lemma. Let B' be the subset of $\{G_{iS} \mid S \in \mathcal{S}\}$ with highest cardinality s.t $\gcd(b, b') < (m-k)t - k(t-1)$ for all $b, b' \in B'$. We observed that in the average case, we could identify a product by a $m-k$ subset of $[m]$, and the gcd of any two products can be completely determined by the cardinality of the intersection of their corresponding sets.

Lemma 5. Let $B = \{p \cdot b\}$ where $b \in B'$ and $p \in M_{k(t-1),n_0,\mathbf{x}}$. Then in the average case B forms a basis with a very high probablity.

I ran experiments on *Mathematica* software to check whether it is a reasonable assumption. I was checking these for the average case, which means I was uniformly sampling the coefficients of L, Q_{ij} from a reasonably large set. In these experiments, I found several counter-examples to the lemma. If the result were true, it is improbable to find a single counter-example!

References

[1] Kayal, Neeraj and Saha, Chandan, *Reconstruction of Non-Degenerate Homogeneous Depth Three Circuits*, Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, 413?424, 2019.

[2] Garg, Ankit and Kayal, Neeraj and Saha, Chandan, *Learning sums of powers of low-degree polynomials in the non-degenerate case*, 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS), 889-899, 2020.

[7] As we are focusing on average case reconstruction the meaning of "choose" should be understood accordingly