gct2022: School and Conference on Geometric Complexity Theory

Arithmetic circuit reconstruction: Part 2

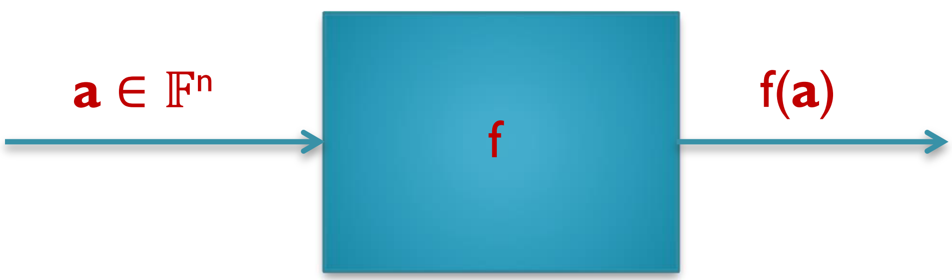
Chandan Saha

Indian Institute of Science

Recap: The reconstruction problem

- Let $f(\mathbf{x})$ be a n-variate degree-d polynomial computed by a circuit of size s from a class C.

- Reconstruction problem for C. Given black-box access to f, output a *small* circuit computing f.



Black-box access to f
(membership query access to f)

Recap: The reconstruction problem

- Let $f(\mathbf{x})$ be a n-variate degree-d polynomial computed by a circuit of size s from a class $\mathcal{C}$.

- Reconstruction problem for $\mathcal{C}$. Given black-box access to f, output a *small* circuit computing f.

- Size of the output circuit. Ideally, $\mathrm{poly}(s)$.
- Proper learning. Output circuit belongs to $\mathcal{C}$.
- Efficiency. Ideally, $\mathrm{poly}(d,s)$.

Recap: Part I summary

- Hardness of ***worst-case*** reconstruction.
- A survey of known results on ***worst-case*** reconstruction.

- Depth-2 circuit reconstruction.
- $\sum\wedge\sum$ circuit reconstruction
  - Improper: ROABP reconstruction
  - Proper: Waring decomposition for $\sum\wedge\sum(k)$ circuits.

Recap: Part I summary

- Hardness of *worst-case* reconstruction.
- A survey of known results on *worst-case* reconstruction.

- Depth-2 circuit reconstruction.
- $\sum\wedge\sum$ circuit reconstruction
  - Improper: ROABP reconstruction
  - Proper: Waring decomposition for $\sum\wedge\sum(k)$ circuits.

- This talk: We will discuss *average-case* reconstruction.

a.k.a. learning in the *non-degenerate* case

Recap: Depth-3 powering circuits

- A depth-3 powering circuit (a.k.a $\sum\wedge\sum$ circuit) computes a sum of powers of linear polynomials, i.e.,
$$f = \ell_1^{d_1} + \ldots + \ell_s^{d_s} ,$$
where ℓ_i has degree 1.

- The reconstruction problem. Given black-box access to a $\sum\wedge\sum$ circuit computing f, output a small circuit for f.

- Proper learning seems hard in the *worst-case* as computing Waring rank is NP-hard [Shitov'16].

Recap: Depth-3 powering circuits

- A depth-3 powering circuit (a.k.a $\sum\wedge\sum$ circuit) computes a sum of powers of linear polynomials, i.e.,
$$f = \ell_1^{d_1} + \ldots + \ell_s^{d_s} ,$$
where ℓ_i has degree 1.

- The reconstruction problem. Given black-box access to a $\sum\wedge\sum$ circuit computing f, output a small circuit for f.

- What if the coefficients of $\ell_1, \ldots, \ell_s$ are chosen <u>randomly</u>?

Learning random $\sum\wedge\sum$ circuits

- A *random* $\sum\wedge\sum$ circuit computes
$$f = \ell_1^{d_1} + \ldots + \ell_s^{d_s} \;,$$
where the coefficients of $\ell_1, \ldots, \ell_s$ are chosen uniformly and independently at random from a sufficiently large finite subset of $\mathbb{F}$.
- The average-case learning problem. Given black-box access to a random $\sum\wedge\sum$ circuit computing f, output a small $\sum\wedge\sum$ circuit for f.

Learning random $\sum\wedge\sum$ circuits

- A *random* $\sum\wedge\sum$ circuit computes

$$f = \ell_1^{d_1} + \ldots + \ell_s^{d_s} \;,$$

where the coefficients of $\ell_1, \ldots, \ell_s$ are chosen uniformly and independently at random from a sufficiently large finite subset of $\mathbb{F}$.

- The average-case learning problem. Given black-box access to a random $\sum\wedge\sum$ circuit computing f, output a small $\sum\wedge\sum$ circuit for f.

- For simplicity, assume that $\ell_1, \ldots, \ell_s$ are linear forms and $d_1 = \ldots = d_s = d$.

Learning random $\Sigma\wedge\Sigma$ circuits

- A *random* $\Sigma\wedge\Sigma$ circuit computes
$$f = \ell_1^d + \ldots + \ell_s^d \quad ,$$
where the coefficients of $\ell_1, \ldots, \ell_s$ are chosen uniformly and independently at random from a sufficiently large finite subset of $\mathbb{F}$.
- The average-case learning problem. Given black-box access to a random $\Sigma\wedge\Sigma$ circuit computing f, output a small $\Sigma\wedge\Sigma$ circuit for f.

- What is the complexity of the above problem?

Learning random $\sum\wedge\sum$ circuits

- A *random* $\sum\wedge\sum$ circuit computes
$$f = \ell_1^d + \ldots + \ell_s^d \ ,$$
where the coefficients of $\ell_1, \ldots, \ell_s$ are chosen <u>uniformly and independently at random</u> from a sufficiently large finite subset of $\mathbb{F}$.
- An easier case. Suppose $s \leq n$. Then, $\ell_1, \ldots, \ell_s$ are $\mathbb{F}$-linearly independent w.h.p.

- In other words, f is <u>*equivalent*</u> to the d-th power symmetric polynomial PSym in s variables w.h.p.

Learning random $\sum\wedge\sum$ circuits

- A *random* $\sum\wedge\sum$ circuit computes
$$f = \ell_1^d + \ldots + \ell_s^d \;,$$
where the coefficients of $\ell_1, \ldots, \ell_s$ are chosen uniformly and independently at random from a sufficiently large finite subset of $\mathbb{F}$.
- An easier case. Suppose $s \leq n$. Then, $\ell_1, \ldots, \ell_s$ are $\mathbb{F}$-linearly independent w.h.p.
- In other words, f is *equivalent* to the d-th power symmetric polynomial PSym in s variables w.h.p.

Let's take a detour into the *polynomial equivalence problem*...

The polynomial equivalence problem

Polynomial equivalence problem

- Orbit. The *orbit* of an n-variate polynomial g over $\mathbb{F}$ is the set $\mathrm{orb}(g) := \{g(A\mathbf{x}) : A \in GL(n, \mathbb{F})\}$.

- Equivalent polynomials. Two n-variate polynomials $f, g \in \mathbb{F}[\mathbf{x}]$ are *equivalent*, denoted as $f \sim g$, if there's a $A \in GL(n, \mathbb{F})$ s.t. $f = g(A\mathbf{x})$ (i.e., $f \in \mathrm{orb}(g)$).

- For example, $f = \ell_1^d + \ldots + \ell_n^d$, where $\ell_1, \ldots, \ell_n$ are $\mathbb{F}$–linearly independent, is equivalent to the power symmetric polynomial $\mathrm{PSym}_{n,d} = x_1^d + \ldots + x_n^d$.

Polynomial equivalence problem

- Orbit. The *orbit* of an n-variate polynomial g over $\mathbb{F}$ is the set $\mathrm{orb}(g) := \{g(A\mathbf{x}) : A \in GL(n, \mathbb{F})\}$.
- Equivalent polynomials. Two n-variate polynomials $f, g \in \mathbb{F}[\mathbf{x}]$ are *equivalent*, denoted as $f \sim g$, if there's a $A \in GL(n, \mathbb{F})$ s.t. $f = g(A\mathbf{x})$ (i.e., $f \in \mathrm{orb}(g)$).
- The equivalence problem. Given f and g as <u>lists of coefficients</u>, check if $f \sim g$. If equivalent, then find a certificate $A \in GL(n, \mathbb{F})$.
- Polynomial equivalence (PE) has been studied intensely.

PE: Known results

- Thierauf (1998); Saxena (2006): PE is in NP $\cap$ coAM over $\mathbb{F}_q$. Hence, unlikely to be NP-complete.

- Not known to be decidable over $\mathbb{Q}$.

- Over arbitrary fields, the best known complexity is the same as that of polynomial solvability.

PE: Known results

- Thierauf (1998); Saxena (2006): PE is in NP $\cap$ coAM over $\mathbb{F}_q$. Hence, unlikely to be NP-complete.

- Not known to be decidable over $\mathbb{Q}$.

- Over arbitrary fields, the best known complexity is the same as that of polynomial solvability.

- What if f and g belong to restricted classes/families of polynomials?

PE: Known results

- Minkowski (1885); Hasse (1921); Serre (1973); Witt (1998); Wallenborn (2013): Quadratic form equivalence can be solved in randomized polynomial time over $\mathbb{F}_q$, $\mathbb{C}$, $\mathbb{R}$, and over $\mathbb{Q}$ (with access to Integer Factoring oracle).

- Uses well-known classification results for quadratic forms over $\mathbb{F}_q$, $\mathbb{C}$, $\mathbb{R}$, and $\mathbb{Q}$.

- For e.g., a quadratic form over $\mathbb{C}$ having n essential variables is equivalent to $x_1^2 + \ldots + x_n^2$.

PE: Known results

- Minkowski (1885); Hasse (1921); Serre (1973); Witt (1998); Wallenborn (2013): Quadratic form equivalence can be solved in randomized polynomial time over $\mathbb{F}_q$, $\mathbb{C}$, $\mathbb{R}$, and over $\mathbb{Q}$ (with access to Integer Factoring oracle).

- Agrawal & Saxena (2005): Cubic form equivalence is graph isomorphism hard.

- Grochow & Qiao (2019): Tensor isomorphism, matrix space isometry & conjugacy, algebra isomorphism and cubic form equivalence are poly-time equivalent.

PE: Known results

- Kayal (2011): Initiated the study of a natural variant of PE for <u>well-known polynomial families</u>.
- Let $G = \{g_m : m \in \mathbb{N}\}$ be a polynomial family, say Det .

- PE for G. Given <u>b.b.a</u> to f, check if $f \in orb(g_m)$ for some $m \in \mathbb{N}$. If yes, then find a certificate $A \in GL(n, \mathbb{F})$.

PE: Known results

- Kayal (2011): Initiated the study of a natural variant of PE for well-known polynomial families.
- Let $G = \{g_m : m \in \mathbb{N}\}$ be a polynomial family, say Det .

- PE for G. Given b.b.a to f, check if $f \in orb(g_m)$ for some $m \in \mathbb{N}$. If yes, then find a certificate $A \in GL(n, \mathbb{F})$.

- Why is this version of PE interesting?

PE: Known results

- Kayal (2011): Initiated the study of a natural variant of PE for <u>well-known polynomial families</u>.
- Let $G = \{g_m : m \in \mathbb{N}\}$ be a polynomial family, say Det .

- Affine projections. The set of *affine projections* of a n-variate g is $\mathrm{aproj}(g) := \{g(A\mathbf{x} + \mathbf{b}) : A \in \mathbb{F}^{n \times n}, \mathbf{b} \in \mathbb{F}^n\}$.

PE: Known results

- Kayal (2011): Initiated the study of a natural variant of PE for <u>well-known polynomial families</u>.
- Let $G = \{g_m : m \in \mathbb{N}\}$ be a polynomial family, say Det .

- Affine projections. The set of *affine projections* of a n-variate g is $\text{aproj}(g) := \{g(A\mathbf{x} + \mathbf{b}) : A \in \mathbb{F}^{n \times n}, \mathbf{b} \in \mathbb{F}^n\}$.

- Affine projections of a "simple" G can be very powerful.
  - aproj(PSym) captures $\sum\wedge\sum$ circuits,
  - aproj(SumProd) captures $\sum\prod\sum$ circuits, (SumProd has a depth-2 ROF)
  - aproj(ANF) captures formulas, (ANF has a ROF)
  - aproj(Det) & aproj(IMM) capture ABPs.

PE: Known results

- Kayal (2011): Initiated the study of a natural variant of PE for <u>well-known polynomial families</u>.
- Let $G = \{g_m : m \in \mathbb{N}\}$ be a polynomial family, say Det .
- Orbit closure. The *orbit closure* of g over $\mathbb{F}$, denoted as $\overline{\mathrm{orb}(g)}$, is the Zariski closure of $\mathrm{orb}(g)$.
- Fact. $\mathrm{orb}(g) \subseteq \mathrm{aproj}(g) \subseteq \overline{\mathrm{orb}(g)}$. $(\mathrm{char}(\mathbb{F}) = 0)$

PE: Known results

- Kayal (2011): Initiated the study of a natural variant of PE for <u>well-known polynomial families</u>.
- Let $G = \{g_m : m \in \mathbb{N}\}$ be a polynomial family, say Det .
- Orbit closure. The *orbit closure* of g over $\mathbb{F}$, denoted as $\overline{\mathrm{orb}(g)}$, is the Zariski closure of $\mathrm{orb}(g)$.
- Fact. $\mathrm{orb}(g) \subseteq \mathrm{aproj}(g) \subseteq \overline{\mathrm{orb}(g)}$. $(\mathrm{char}(\mathbb{F}) = 0)$
- Natural to study the learning problem for orbits of well-known polynomial families.

PE: Known results

- Kayal (2011): Initiated the study of a natural variant of PE for well-known polynomial families.
- Let $G = \{g_m : m \in \mathbb{N}\}$ be a polynomial family, say Det .

- PE for G. Given b.b.a to f, check if $f \in \mathrm{orb}(g_m)$ for some $m \in \mathbb{N}$. If yes, then find a certificate $A \in GL(n, \mathbb{F})$.

- Kayal (2011, 2012); Kayal, Nair, S., Tavenas (2017); Garg, Gupta, Kayal, S. (2019); Murthy, Nair, S. (2020): Randomized poly-time PE are known for Det, Perm, IMM, tr-IMM, ESym, PSym, SumProduct etc.

PE for Power Symmetric Polynomials

- Let PSym $= x_1^d + \ldots + x_s^d$. Assume $d \geq 3$, $\mathrm{char}(\mathbb{F}) = 0$.
- Let $f = \ell_1^d + \ldots + \ell_s^d$, where $\ell_1, \ldots, \ell_s$ are $\mathbb{F}$-l.i.

- Recall, a quadratic form over $\mathbb{C}$ having n essential variables is equivalent to $x_1^2 + \ldots + x_n^2$.

PE for Power Symmetric Polynomials

- Let $\mathrm{PSym} = x_1^d + \ldots + x_s^d$. Assume $d \geq 3$, $\mathrm{char}(\mathbb{F}) = 0$.
- Let $f = \ell_1^d + \ldots + \ell_s^d$, where $\ell_1, \ldots, \ell_s$ are $\mathbb{F}$-l.i.

- Given b.b.a. to f, can we recover $\ell_1, \ldots, \ell_s$? (up to d^{th} roots of 1)

PE for Power Symmetric Polynomials

- Let PSym $= x_1^d + \ldots + x_s^d$. Assume $d \geq 3$, $\mathrm{char}(\mathbb{F}) = 0$.
- Let $f = \ell_1^d + \ldots + \ell_s^d$, where $\ell_1, \ldots, \ell_s$ are $\mathbb{F}$-l.i.

- Given b.b.a. to f, can we recover $\ell_1, \ldots, \ell_s$? (up to d^{th} roots of 1)

- Obs. The number of essential variables of f is s.

- Apply the Carlini-Kayal algorithm to remove redundant variables. So, we can assume w.l.o.g that $s = n$.

PE for Power Symmetric Polynomials

- Let $\mathrm{PSym} = x_1^d + \ldots + x_n^d$. Assume $d \geq 3$, $\mathrm{char}(\mathbb{F}) = 0$.

- Input: Black-box access to $f \in \mathrm{orb}(\mathrm{PSym})$.

  Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f = \mathrm{PSym}(A\mathbf{x})$.

- Algorithm: Uses the <u>Hessian matrix</u> associated with f.

The Hessian of a polynomial

- Let f be an n-variate polynomial and $\partial_{i,j}f$ the derivative of f w.r.t. x_i and x_j.

- The Hessian of f. It is the matrix $Hes_f(\mathbf{x}) := (\ \partial_{i,j}f\ )_{i,j\in[n]}$.

- The Hessian matrix appears naturally in the Taylor expansion of a polynomial and has important applications in optimization, second derivative tests, etc.

The Hessian of a polynomial

- Let f be an n-variate polynomial and $\partial_{i,j}f$ the derivative of f w.r.t. x_i and x_j.

- The Hessian of f. It is the matrix $\mathrm{Hes}_f(\mathbf{x}) := (\partial_{i,j}f)_{i,j\in[n]}$.

- Obs. If $f = g(A\mathbf{x})$ for some $A \in \mathbb{F}^{n\times n}$, then
$$\mathrm{Hes}_f(\mathbf{x}) = A^T \cdot \mathrm{Hes}_g(A\mathbf{x}) \cdot A .$$

The Hessian of a polynomial

- Let f be an n-variate polynomial and $\partial_{i,j}f$ the derivative of f w.r.t. x_i and x_j.

- The Hessian of f. It is the matrix $\mathrm{Hes}_f(\mathbf{x}) := (\ \partial_{i,j}f\ )_{i,j\in[n]}$.

- Obs. If $f = g(A\mathbf{x})$ for some $A \in \mathbb{F}^{n \times n}$, then
$$\mathrm{Hes}_f(\mathbf{x}) \ = \ A^T \cdot \mathrm{Hes}_g(A\mathbf{x}) \cdot A\ .$$
- Proof~. Uses chain rule. Let $\nabla f := (\partial_1 f, \partial_2 f, \dots \partial_n f)^T$. Then,
$$\nabla f \ = A^T \cdot [\nabla g](A\mathbf{x}) \ \ .$$

The Hessian of a polynomial

- Let f be an n-variate polynomial and $\partial_{i,j}f$ the derivative of f w.r.t. x_i and x_j.

- The Hessian of f. It is the matrix $\mathrm{Hes}_f(\mathbf{x}) := (\ \partial_{i,j}f\ )_{i,j\in[n]}$.

- Obs. If $f = g(A\mathbf{x})$ for some $A \in \mathbb{F}^{n \times n}$, then
$$\mathrm{Hes}_f(\mathbf{x}) = A^T \cdot \mathrm{Hes}_g(A\mathbf{x}) \cdot A .$$

- Cor. $\det(\mathrm{Hes}_f) = c \cdot \det(\mathrm{Hes}_g)(A\mathbf{x})$, where $c \in \mathbb{F}$.

PE for Power Symmetric Polynomials

- Let $\mathrm{PSym} = x_1^d + \ldots + x_n^d$. Assume $d \geq 3$, $\mathrm{char}(\mathbb{F}) = 0$.

- **Input:** Black-box access to $f \in \mathrm{orb}(\mathrm{PSym})$.

  Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f = \mathrm{PSym}(A\mathbf{x})$.

$$\boxed{\mathrm{Hes}_f(\mathbf{x}) = A^T \cdot \mathrm{Hes}_{\mathrm{PSym}}(A\mathbf{x}) \cdot A}$$

- **Algorithm:** (high-level)
  - **Step 1.** Compute b.b.a to $H := \det(\mathrm{Hes}_f)$.

  Obs. B.b.a. to $\partial_{i,j} f$ can be computed efficiently from b.b.a. to f.

PE for Power Symmetric Polynomials

- Let $\mathrm{PSym} = x_1^d + \ldots + x_n^d$. Assume $d \geq 3$, $\mathrm{char}(\mathbb{F}) = 0$.

- Input: Black-box access to $f \in \mathrm{orb}(\mathrm{PSym})$.
  Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f = \mathrm{PSym}(A\mathbf{x})$.

$$\boxed{\mathrm{Hes}_f(\mathbf{x}) = A^T \cdot \mathrm{Hes}_{\mathrm{PSym}}(A\mathbf{x}) \cdot A}$$

- Algorithm: (high-level)
  - Step 1. Compute b.b.a to $H := \det(\mathrm{Hes}_f)$.
  - Step 2. Compute b.b.a to the factors of H.
    (using b.b. polynomial factorization)

PE for Power Symmetric Polynomials

- Let $\mathrm{PSym} = x_1^d + \ldots + x_n^d$. Assume $d \geq 3$, $\mathrm{char}(\mathbb{F}) = 0$.

- Input: Black-box access to $f \in \mathrm{orb}(\mathrm{PSym})$.

  Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f = \mathrm{PSym}(A\mathbf{x})$.

$$\mathrm{Hes}_f(\mathbf{x}) = A^T \cdot \mathrm{Hes}_{\mathrm{PSym}}(A\mathbf{x}) \cdot A$$

- Algorithm: (high-level)
  - Step 1. Compute b.b.a to $H := \det(\mathrm{Hes}_f)$.
  - Step 2. Compute b.b.a to the factors of H.
  - Step 3. Recover A from the *linear* factors of H.

PE for Power Symmetric Polynomials

- Let PSym $= x_1^d + \ldots + x_n^d$. Assume $d \geq 3$, $\mathrm{char}(\mathbb{F}) = 0$.

- Input: Black-box access to $f \in \mathrm{orb}(\mathrm{PSym})$.
  Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f = \mathrm{PSym}(A\mathbf{x})$.

- Algorithm: (correctness) $\boxed{\mathrm{Hes}_f(\mathbf{x}) = A^T \cdot \mathrm{Hes}_{\mathrm{PSym}}(A\mathbf{x}) \cdot A}$
  - Step 3. Recover A from the linear factors of H.

  Proof~. $\mathrm{Hes}_{\mathrm{PSym}}(\mathbf{x}) = \mathrm{diag}\,(d' x_1^{d-2}, \ldots, d' x_n^{d-2})$, where $d' = d(d-1)$.

PE for Power Symmetric Polynomials

- Let PSym $= x_1^d + \ldots + x_n^d$. Assume $d \geq 3$, $\mathrm{char}(\mathbb{F}) = 0$.

- Input: Black-box access to $f \in \mathrm{orb}(\mathrm{PSym})$.
  Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f = \mathrm{PSym}(A\mathbf{x})$.

- Algorithm: (correctness) $\boxed{\mathrm{Hes}_f(\mathbf{x}) = A^T \cdot \mathrm{Hes}_{\mathrm{PSym}}(A\mathbf{x}) \cdot A}$
  - Step 3. Recover A from the linear factors of H.

  Proof~.
  $$\mathrm{Hes}_{\mathrm{PSym}}(\mathbf{x}) = \mathrm{diag}\,(d' x_1^{d-2}, \ldots, d' x_n^{d-2})$$
  $$\mathrm{Hes}_{\mathrm{PSym}}(A\mathbf{x}) = \mathrm{diag}\,(d'\ell_1^{d-2}, \ldots, d'\ell_n^{d-2}),\text{ where}$$
  $$A\mathbf{x} = (\ell_1, \ldots, \ell_n)^T$$

PE for Power Symmetric Polynomials

- Let PSym $= x_1^d + \ldots + x_n^d$. Assume $d \geq 3$, $\mathrm{char}(\mathbb{F}) = 0$.

- Input: Black-box access to $f \in \mathrm{orb}(\mathrm{PSym})$.

  Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f = \mathrm{PSym}(A\mathbf{x})$.

- Algorithm: (correctness) $\boxed{\mathrm{Hes}_f(\mathbf{x}) = A^T \cdot \mathrm{Hes}_{\mathrm{PSym}}(A\mathbf{x}) \cdot A}$

  - Step 3. Recover A from the linear factors of H.

  Proof~. $\mathrm{Hes}_{\mathrm{PSym}}(\mathbf{x}) = \mathrm{diag}\,(d' x_1^{d-2}, \ldots, d' x_n^{d-2})$

  $\mathrm{Hes}_{\mathrm{PSym}}(A\mathbf{x}) = \mathrm{diag}\,(d' \ell_1^{d-2}, \ldots, d' \ell_n^{d-2})$

  $H = c \cdot \ell_1^{d-2} \cdot \ldots \cdot \ell_n^{d-2}, \quad c \in \mathbb{F}$.

PE for Power Symmetric Polynomials

- Let $\mathrm{PSym} = x_1^d + \ldots + x_n^d$. Assume $d \geq 3$, $\mathrm{char}(\mathbb{F}) = 0$.

- Input: Black-box access to $f \in \mathrm{orb}(\mathrm{PSym})$.

  Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f = \mathrm{PSym}(A\mathbf{x})$.

- Algorithm: (correctness) $\boxed{\mathrm{Hes}_f(\mathbf{x}) = A^T \cdot \mathrm{Hes}_{\mathrm{PSym}}(A\mathbf{x}) \cdot A}$

  - Step 3. Recover A from the linear factors of H.

  Proof~.
$$\mathrm{Hes}_{\mathrm{PSym}}(\mathbf{x}) = \mathrm{diag}\,(d' x_1^{d-2}, \ldots, d' x_n^{d-2})$$
$$\mathrm{Hes}_{\mathrm{PSym}}(A\mathbf{x}) = \mathrm{diag}\,(d' \ell_1^{d-2}, \ldots, d' \ell_n^{d-2})$$
$$H = c \cdot \ell_1^{d-2} \cdot \ldots \cdot \ell_n^{d-2}, \quad c \in \mathbb{F}.$$

Recover $\ell_1, \ldots, \ell_n$ (up to $\mathbb{F}$-multiples)

PE for Power Symmetric Polynomials

- Let $\mathrm{PSym} = x_1^d + \ldots + x_n^d$. Assume $d \geq 3$, $\mathrm{char}(\mathbb{F}) = 0$.

- Input: Black-box access to $f \in \mathrm{orb}(\mathrm{PSym})$.

  Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f = \mathrm{PSym}(A\mathbf{x})$.

- Algorithm: (correctness) $\boxed{\mathrm{Hes}_f(\mathbf{x}) = A^T \cdot \mathrm{Hes}_{\mathrm{PSym}}(A\mathbf{x}) \cdot A}$

  - Step 3. Recover A from the linear factors of H.

  Proof~. $\mathrm{Hes}_{\mathrm{PSym}}(\mathbf{x}) = \mathrm{diag}\,(d' x_1^{d-2}, \ldots, d' x_n^{d-2})$

  $\mathrm{Hes}_{\mathrm{PSym}}(A\mathbf{x}) = \mathrm{diag}\,(d' \ell_1^{d-2}, \ldots, d' \ell_n^{d-2})$

  $$H = c \cdot \underbrace{\ell_1^{d-2} \cdot \ldots \cdot \ell_n^{d-2}}_{\text{Recover } \ell'_1, \ldots, \ell'_n}, \quad c \in \mathbb{F} .$$

PE for Power Symmetric Polynomials

- Let $\mathrm{PSym} = x_1^d + \ldots + x_n^d$. Assume $d \geq 3$, $\mathrm{char}(\mathbb{F}) = 0$.

- Input: Black-box access to $f \in \mathrm{orb}(\mathrm{PSym})$.

  Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f = \mathrm{PSym}(A\mathbf{x})$.

- Algorithm: (correctness)
  - Step 3. Recover A from the linear factors of H.

  Proof~. Observe, $f = z_1 \cdot \ell'^{\,d}_1 + \ldots + z_n \cdot \ell'^{\,d}_n$ for some unknown $z_1, \ldots, z_n$. Set up a linear system in $z_1, \ldots, z_n$ by evaluating f and $\ell'_1, \ldots, \ell'_n$ at n random points. Solve for $z_1, \ldots, z_n$ and take d-th roots.

PE for Sum-Product polynomials

- Let SumProd $= x_{11}x_{12}\cdots x_{1d} + \ldots + x_{s1}x_{s2}\cdots x_{sd}$.
- Observe, aproj(SumProd) captures $\Sigma\Pi\Sigma$ circuits.
- A quadratic form over $\mathbb{C}$ having n essential variables (where n is even) is equivalent to $x_1x_2 + \ldots + x_{n-1}x_n$.

PE for Sum-Product polynomials

- Let SumProd $= x_{11}x_{12}\cdots x_{1d} + \ldots + x_{s1}x_{s2}\cdots x_{sd}$.
- Observe, aproj(SumProd) captures $\sum\prod\sum$ circuits.

- Input: Black-box access to $f \in$ orb(SumProd).

  Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f =$ SumProd($A\mathbf{x}$).

- Here, $n = sd$.

PE for Sum-Product polynomials

- Let SumProd $= x_{11}x_{12}\cdots x_{1d} + \ldots + x_{s1}x_{s2}\cdots x_{sd}$.
- Observe, aproj(SumProd) captures $\Sigma\Pi\Sigma$ circuits.

- Input: Black-box access to $f \in \mathrm{orb}(\mathrm{SumProd})$.

  Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f = \mathrm{SumProd}(A\mathbf{x})$.

$$\boxed{\mathrm{Hes}_f(\mathbf{x}) = A^T \cdot \mathrm{Hes}_{\mathrm{SumProd}}(A\mathbf{x}) \cdot A}$$

- Algorithm: (high-level)
  - Step 1. Compute b.b.a to $H := \det(\mathrm{Hes}_f)$.
  - Step 2. Compute b.b.a to the factors of H.
  - Step 3. Recover A from the linear factors of H.

PE for Sum-Product polynomials

- Let SumProd $= x_{11}x_{12}\cdots x_{1d} + \ldots + x_{s1}x_{s2}\cdots x_{sd}$.
- Observe, aproj(SumProd) captures $\Sigma\Pi\Sigma$ circuits.

- Input: Black-box access to $f \in$ orb(SumProd).

Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f = \text{SumProd}(A\mathbf{x})$.

- Algorithm: (correctness) $\boxed{\text{Hes}_f(\mathbf{x}) = A^T \cdot \text{Hes}_{\text{SumProd}}(A\mathbf{x}) \cdot A}$
  - Step 3. Recover A from the linear factors of H.

Proof~. $\text{Hes}_{\text{SumProd}}(\mathbf{x}) = \text{blocdiag}(\text{Hes}_{h_1}(\mathbf{x}), \ldots, \text{Hes}_{h_s}(\mathbf{x}))$, where $h_i = x_{i1}x_{i2}\cdots x_{id}$ is a monomial.

PE for Sum-Product polynomials

- Let SumProd $= x_{11}x_{12}\cdots x_{1d} + \ldots + x_{s1}x_{s2}\cdots x_{sd}$.
- Observe, aproj(SumProd) captures $\Sigma\Pi\Sigma$ circuits.

- Input: Black-box access to $f \in \mathrm{orb}(\mathrm{SumProd})$.

  Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f = \mathrm{SumProd}(A\mathbf{x})$.

- Algorithm: (correctness) $\boxed{\mathrm{Hes}_f(\mathbf{x}) = A^T \cdot \mathrm{Hes}_{\mathrm{SumProd}}(A\mathbf{x}) \cdot A}$
  - ➢ Step 3. Recover A from the linear factors of H.

  Proof~.

  Clm. $\det(\mathrm{Hes}_{h_i}(\mathbf{x})) = (-1)^{d-1}(d-1) \cdot x_{i1}{}^{d-2} \cdots x_{id}{}^{d-2}$.

  Hessian determinant of a monomial is a monomial.

PE for Sum-Product polynomials

- Let SumProd $= x_{11}x_{12}\cdots x_{1d} + \ldots + x_{s1}x_{s2}\cdots x_{sd}$.
- Observe, aproj(SumProd) captures $\Sigma\Pi\Sigma$ circuits.

- Input: Black-box access to $f \in \mathrm{orb}(\mathrm{SumProd})$.

  Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f = \mathrm{SumProd}(A\mathbf{x})$.

- Algorithm: (correctness) $\boxed{\mathrm{Hes}_f(\mathbf{x}) = A^T \cdot \mathrm{Hes}_{\mathrm{SumProd}}(A\mathbf{x}) \cdot A}$
  - Step 3. Recover A from the linear factors of H.

  Proof~. Hence, $H = c \cdot \ell_{11}^{d-2} \cdot \ldots \cdot \ell_{sd}^{d-2}$, $c \in \mathbb{F}$,
  where $A\mathbf{x} = (\ell_{11}, \ldots, \ell_{sd})^T$.

PE for Sum-Product polynomials

- Let SumProd $= x_{11}x_{12}\cdots x_{1d} + \ldots + x_{s1}x_{s2}\cdots x_{sd}$.
- Observe, aproj(SumProd) captures $\sum\prod\sum$ circuits.

- Input: Black-box access to $f \in$ orb(SumProd).

  Output: A matrix $A \in GL(n, \mathbb{F})$ s.t. $f = \mathrm{SumProd}(A\mathbf{x})$.

- Algorithm: (correctness) $\boxed{\mathrm{Hes}_f(\mathbf{x}) = A^T \cdot \mathrm{Hes}_{\mathrm{SumProd}}(A\mathbf{x}) \cdot A}$
  - ➢ Step 3. Recover A from the linear factors of H.

  Proof~. Hence, $H = c \cdot \ell_{11}^{d-2} \cdot \ldots \cdot \ell_{sd}^{d-2}$, $c \in \mathbb{F}$, where $A\mathbf{x} = (\ell_{11}, \ldots, \ell_{sd})^T$. The rest of the argument is similar to PE for PSym.

PE for Read-once formulas

- Let SumProd $= x_{11}x_{12}\cdots x_{1d} + \ldots + x_{s1}x_{s2}\cdots x_{sd}$.
- Observe, SumProd is a ROF. Recall, affine projections of ROFs capture formulas.
- Can PE for ROFs be solved efficiently?

PE for Read-once formulas

- Let SumProd $= x_{11}x_{12}\cdots x_{1d} + \ldots + x_{s1}x_{s2}\cdots x_{sd}$.
- Observe, SumProd is a ROF. Recall, affine projections of ROFs capture formulas.
- Can PE for ROFs be solved efficiently?

- Gupta, S., Thankey (*ongoing*). Given b.b.a. to f, g in the orbits of ROFs*, the problem of checking if $f \sim g$ and finding a witness A can be solved in randomized poly-time.

- Analyzes the Hessian of a general ROF.

*mild conditions apply

PE for Read-once formulas

- Let SumProd $= x_{11}x_{12}\cdots x_{1d} + \ldots + x_{s1}x_{s2}\cdots x_{sd}$.
- Observe, SumProd is a ROF. Recall, affine projections of ROFs capture formulas.
- Can PE for ROFs be solved efficiently?

- Gupta, S., Thankey (*ongoing*). Given b.b.a. to f, g in the orbits of ROFs*, the problem of checking if $f \sim g$ and finding a witness A can be solved in randomized poly-time.

- A generalization of quadratic form equivalence:

  A quadratic form over $\mathbb{C}$ is equivalent to $x_1x_2+\ldots+x_{n-1}x_n$ (n even).

PE for PSym w/o poly factoring

- Koiran & Saha (2021); Koiran & Skomra (2020). Solves the decision version of PE for PSym over $\mathbb{C}$ in randomized poly-time without appealing to polynomial factorization.

- Involves *only* arithmetic operations and equality tests.
- If f has rational coefficients, then the algorithm requires polynomial number of bit operations.

Back to learning random $\sum\wedge\sum$ circuits

- A *random* $\sum\wedge\sum$ circuit computes
$$f = \ell_1^d + \ldots + \ell_s^d \quad ,$$
where the coefficients of $\ell_1, \ldots, \ell_s$ are chosen uniformly and independently at random from a sufficiently large subset of $\mathbb{F}$.
- The average-case learning problem. Given black-box access to a random $\sum\wedge\sum$ circuit computing f, output a small $\sum\wedge\sum$ circuit for f.

- Can we handle $s > n$? Does Hessian help?

Back to learning random $\sum\wedge\sum$ circuits

- Garcia-Marco, Koiran, Pecatte (2018). Random $\sum\wedge\sum$ circuits can be reconstructed in randomized poly-time provided $s \approx n^2/2$ and $d \geq 5$.

- Uses <u>*4-th order* Hessian</u> and shows that the determinant is nonzero (w.h.p) and factorizes into linear factors.

Back to learning random $\sum\wedge\sum$ circuits

- Garcia-Marco, Koiran, Pecatte (2018). Random $\sum\wedge\sum$ circuits can be reconstructed in randomized poly-time provided $s \approx n^2/2$ and $d \geq 5$.

- Uses *4-th order* Hessian and shows that the determinant is nonzero (w.h.p) and factorizes into linear factors.

- Unclear if the strategy scales to higher s. More importantly, it is not clear how effective Hessian is in learning other − more powerful − models.
- It seems we need a different strategy…

Learning from lower bounds:
A paradigm

Learning from lower bounds?

- Fortnow & Klivans (2009): A randomized poly-time reconstruction algorithm for *C* implies super-polynomial lower bound for *C*. (Learning –> lower bound)

- Does lower bound imply *worst-case* reconstruction? Unlikely. Reconstruction appears to be inherently hard.

Learning from lower bounds?

- Fortnow & Klivans (2009): A randomized poly-time reconstruction algorithm for *C* implies super-polynomial lower bound for *C*. (Learning –> lower bound)

- Does lower bound imply *worst-case* reconstruction? Unlikely. Reconstruction appears to be inherently hard.

- Does lower bound imply *average-case* reconstruction?

A typical lower bound proof

- Suppose that a circuit from C computes a polynomial

$$f = T_1 + \ldots + T_s ,$$

where each term T_i is "simple" is some sense.

- For example, T_i is a power of a linear polynomial for $\Sigma\wedge\Sigma$ circuits.

A typical lower bound proof

- Suppose that a circuit from $\mathcal{C}$ computes a polynomial

$$f = T_1 + \ldots + T_s ,$$

where each term T_i is "simple" is some sense.

- A typical lower bound proof for $\mathcal{C}$ proceeds by defining a *complexity measure* (map) $\mu : \mathbb{F}[\mathbf{x}] \rightarrow \mathbb{N}$ s.t.
  - $\mu(f + g) \leq \mu(f) + \mu(g)$ (subadditivity)
  - $\mu(T_i) \leq L$, where L is a "small" quantity,
  - $\mu(f) \geq H$, where H is a "large" quantity.

A typical lower bound proof

- Suppose that a circuit from C computes a polynomial

$$f = T_1 + \ldots + T_s ,$$

  where each term T_i is "simple" is some sense.

- A typical lower bound proof for C proceeds by defining a *complexity measure* (map) $\boldsymbol{\mu} : \mathbb{F}[\mathbf{x}] \to \mathbb{N}$ s.t.
  - $\mu(f + g) \leq \mu(f) + \mu(g)$ (subadditivity)
  - $\mu(T_i) \leq L$, where L is a "small" quantity,
  - $\mu(f) \geq H$, where H is a "large" quantity.
- Any C-circuit computing f must have $s \geq H/L$ terms.

A typical lower bound proof

- Suppose that a circuit from C computes a polynomial

$$f = T_1 + \ldots + T_s ,$$

where each term T_i is "simple" is some sense.

Typically, $\mu(f)$ is the *dimension of a vector space* U associated with f.

- A typical lower bound proof for C proceeds by defining a *complexity measure* (map) $\mu : \mathbb{F}[\mathbf{x}] \to \mathbb{N}$ s.t.
  - $\mu(f + g) \leq \mu(f) + \mu(g)$ (subadditivity)
  - $\mu(T_i) \leq L$, where L is a "small" quantity,
  - $\mu(f) \geq H$, where H is a "large" quantity.
- Any C-circuit computing f must have $s \geq H/L$ terms.

Lower bound for $\sum\wedge\sum$ circuits

- A $\sum\wedge\sum$ circuit computes
$$f = \ell_1^d + \ldots + \ell_s^d \quad ,$$
where a term $T_i = \ell_i^d$.
- Let $\partial^k f$ be the set of k-th order partials of f, $U := \langle \partial^k f \rangle$ and $U_i := \langle \partial^k T_i \rangle$. Define $\mu(f) := \dim U$, $\mu(T_i) := \dim U_i$.

Lower bound for $\sum\wedge\sum$ circuits

- A $\sum\wedge\sum$ circuit computes

$$f = \ell_1^d + \ldots + \ell_s^d ,$$

where a term $T_i = \ell_i^d$.

- Let $\partial^k f$ be the set of k-th order partials of f, $U := \langle \partial^k f \rangle$ and $U_i := \langle \partial^k T_i \rangle$. Define $\mu(f) := \dim U$, $\mu(T_i) := \dim U_i$.

- As ∂^k is a <u>*set of linear operators*</u> on $\mathbb{F}[\mathbf{x}]$,

$$U \subseteq U_1 + \ldots + U_s , \qquad \text{and so,}$$

$$\mu(f) \leq \mu(T_1) + \ldots + \mu(T_s) \quad \text{(subadditivity).}$$

Lower bound for $\sum\wedge\sum$ circuits

- A $\sum\wedge\sum$ circuit computes
$$f = \ell_1^d + \ldots + \ell_s^d ,$$
where a term $T_i = \ell_i^d$.

- Let $\partial^k f$ be the set of k-th order partials of f, $U := \langle \partial^k f \rangle$ and $U_i := \langle \partial^k T_i \rangle$. Define $\mu(f) := \dim U$, $\mu(T_i) := \dim U_i$.

- Obs. $\mu(T_i) = 1$ whereas $\mu(x_1 x_2 \cdots x_n) = \binom{n}{k}$.

- Choose $k = n/2$. This gives a $s = \tilde{\Omega}(2^n)$ lower bound for $\sum\wedge\sum$ circuits computing $x_1 x_2 \cdots x_n$.

A typical lower bound proof

- A C-circuit computes a polynomial

$$f = T_1 + \ldots + T_s ,$$

where each term T_i is "simple" is some sense.

- A typical lower bound proof for C involves a <u>set of linear operators</u> $\mathcal{L}$ on $\mathbb{F}[\mathbf{x}]$ s.t. $\dim \langle \mathcal{L} \circ T_i \rangle$ is "small".

- In the lower bound proof for $\sum\wedge\sum$ circuits, $\mathcal{L} = \partial^k$.

A typical lower bound proof

- A $\mathcal{C}$-circuit computes a polynomial

$$f = T_1 + \ldots + T_s ,$$

where each term T_i is "simple" is some sense.

- As $\mathcal{L}$ is linear, $\langle \mathcal{L} \circ f \rangle \subseteq \langle \mathcal{L} \circ T_1 \rangle + \ldots + \langle \mathcal{L} \circ T_s \rangle$.

Learning from LB: A framework

- A $\mathcal{C}$-circuit computes a polynomial

$$f = T_1 + \ldots + T_s ,$$

where each term T_i is "simple" is some sense.

- As $\mathcal{L}$ is linear, $\langle \mathcal{L} \circ f \rangle \subseteq \langle \mathcal{L} \circ T_1 \rangle + \ldots + \langle \mathcal{L} \circ T_s \rangle$.
- If $T_1, \ldots, T_s$ are <u>*random*</u>, we do expect (as $\dim\langle \mathcal{L} \circ T_i \rangle$ is "small")

1. $\langle \mathcal{L} \circ T_1 \rangle + \ldots + \langle \mathcal{L} \circ T_s \rangle = \langle \mathcal{L} \circ T_1 \rangle \oplus \ldots \oplus \langle \mathcal{L} \circ T_s \rangle$
2. $\langle \mathcal{L} \circ f \rangle = \langle \mathcal{L} \circ T_1 \rangle + \ldots + \langle \mathcal{L} \circ T_s \rangle$, implying

$$\boxed{\langle \mathcal{L} \circ f \rangle = \langle \mathcal{L} \circ T_1 \rangle \oplus \ldots \oplus \langle \mathcal{L} \circ T_s \rangle}$$

Learning from LB: A framework

$$f = T_1 + \ldots + T_s \,.$$

- A (crude) approach to learn the terms.
  - Compute a basis of $\langle \mathcal{L} \circ f \rangle$ from f.
  - Decompose $\langle \mathcal{L} \circ f \rangle = \langle \mathcal{L} \circ T_1 \rangle \oplus \ldots \oplus \langle \mathcal{L} \circ T_s \rangle$.
  - Obtain T_i from a basis of $\langle \mathcal{L} \circ T_i \rangle$.

Learning from LB: A framework

$$f = T_1 + \ldots + T_s .$$

- A (crude) approach to learn the terms.
  - Compute a basis of $\langle \mathcal{L} \circ f \rangle$ from f.
  - Decompose $\langle \mathcal{L} \circ f \rangle = \langle \mathcal{L} \circ T_1 \rangle \oplus \ldots \oplus \langle \mathcal{L} \circ T_s \rangle$.
    How?
  - Obtain T_i from a basis of $\langle \mathcal{L} \circ T_i \rangle$.
- What makes $\langle \mathcal{L} \circ T_1 \rangle, \ldots, \langle \mathcal{L} \circ T_s \rangle$ special subspaces of $\langle \mathcal{L} \circ f \rangle$?

Learning from LB: A framework

$$f = T_1 + \ldots + T_s .$$

- Turns out in a typical l.b. proof $\mathcal{L}$ can be expressed as $\mathcal{L} = \mathcal{L}_2 \circ \mathcal{L}_1$, where $\mathcal{L}_1$, $\mathcal{L}_2$ are sets of linear operators.
- For example, $\partial^{k+1} = \partial \circ \partial^k$.

Learning from LB: A framework

$$f = T_1 + \ldots + T_s .$$

- Turns out in a typical l.b. proof $\mathcal{L}$ can be expressed as $\mathcal{L} = \mathcal{L}_2 \circ \mathcal{L}_1$, where $\mathcal{L}_1$, $\mathcal{L}_2$ are sets of linear operators.
- If $T_1, \ldots, T_s$ are *<u>random</u>*, then we do expect

1. $\langle \mathcal{L}_1 \circ f \rangle = \langle \mathcal{L}_1 \circ T_1 \rangle \oplus \ldots \oplus \langle \mathcal{L}_1 \circ T_s \rangle$

$\mathcal{L}_2$

2. $\langle \mathcal{L} \circ f \rangle = \langle \mathcal{L} \circ T_1 \rangle \oplus \ldots \oplus \langle \mathcal{L} \circ T_s \rangle$

Learning from LB: A framework

$$f = T_1 + \ldots + T_s .$$

- Turns out in a typical l.b. proof $\mathcal{L}$ can be expressed as $\mathcal{L} = \mathcal{L}_2 \circ \mathcal{L}_1$, where $\mathcal{L}_1$, $\mathcal{L}_2$ are sets of linear operators.
- If $T_1, \ldots, T_s$ are <u>*random*</u>, then we do expect

1. $\langle \mathcal{L}_1 \circ f \rangle = \langle \mathcal{L}_1 \circ T_1 \rangle \oplus \ldots \oplus \langle \mathcal{L}_1 \circ T_s \rangle$
2. $\langle \mathcal{L} \circ f \rangle = \langle \mathcal{L} \circ T_1 \rangle \oplus \ldots \oplus \langle \mathcal{L} \circ T_s \rangle$

$\mathcal{L}_2$

- Let $U := \langle \mathcal{L}_1 \circ f \rangle$, $U_i := \langle \mathcal{L}_1 \circ T_i \rangle$, $V := \langle \mathcal{L} \circ f \rangle$, $V_i := \langle \mathcal{L} \circ T_i \rangle$.

Learning from LB: A framework

$$f = T_1 + \ldots + T_s .$$

- Turns out in a typical l.b. proof $\mathcal{L}$ can be expressed as $\mathcal{L} = \mathcal{L}_2 \circ \mathcal{L}_1$, where $\mathcal{L}_1$, $\mathcal{L}_2$ are sets of linear operators.
- If $T_1, \ldots, T_s$ are *random*, then we do expect
    1. $U = U_1 \oplus \ldots \oplus U_s$
    2. $V = V_1 \oplus \ldots \oplus V_s$

  $\mathcal{L}_2$

- Let $U := \langle \mathcal{L}_1 \circ f \rangle$, $U_i := \langle \mathcal{L}_1 \circ T_i \rangle$, $V := \langle \mathcal{L} \circ f \rangle$, $V_i := \langle \mathcal{L} \circ T_i \rangle$.

Learning from LB: A framework

$$f = T_1 + \ldots + T_s .$$

- Turns out in a typical l.b. proof $\mathcal{L}$ can be expressed as $\mathcal{L} = \mathcal{L}_2 \circ \mathcal{L}_1$, where $\mathcal{L}_1$, $\mathcal{L}_2$ are sets of linear operators.
- If $T_1, \ldots, T_s$ are *random*, then we do expect
  1. $U = U_1 \oplus \ldots \oplus U_s$
  2. $V = V_1 \oplus \ldots \oplus V_s$

  $\mathcal{L}_2$

- Observe, $V = \langle \mathcal{L}_2 \circ U \rangle$, $V_i = \langle \mathcal{L}_2 \circ U_i \rangle$.

Learning from LB: A framework

$$f = T_1 + \ldots + T_s\,.$$

- Turns out in a typical l.b. proof $\mathcal{L}$ can be expressed as $\mathcal{L} = \mathcal{L}_2 \circ \mathcal{L}_1$, where $\mathcal{L}_1$, $\mathcal{L}_2$ are sets of linear operators.
- If $T_1, \ldots, T_s$ are *random*, then we do expect
    1. $U = U_1 \oplus \ldots \oplus U_s$
    2. $V = V_1 \oplus \ldots \oplus V_s$ $\mathcal{L}_2$
    3. The above decomposition is the *unique decomposition* of U and V into *indecomposable subspaces* s.t. $V_i = \langle \mathcal{L}_2 \circ U_i \rangle$.

Learning from LB: A framework

$$f = T_1 + \ldots + T_s \,.$$

- Turns out in a typical l.b. proof $\mathcal{L}$ can be expressed as $\mathcal{L} = \mathcal{L}_2 \circ \mathcal{L}_1$, where $\mathcal{L}_1$, $\mathcal{L}_2$ are sets of linear operators.
- If $T_1, \ldots, T_s$ are *__random__*, then we do expect
  1. $U = U_1 \oplus \ldots \oplus U_s$ $\quad \mathcal{L}_2$
  2. $V = V_1 \oplus \ldots \oplus V_s$
  3. The above decomposition is the *__unique decomposition__* of U and V *under the action* of $\mathcal{L}_2$.

Learning from LB: A framework

$$f = T_1 + \ldots + T_s \,.$$

- A meta-algorithm to learn the terms. $\mathcal{L} = \mathcal{L}_2 \circ \mathcal{L}_1$
  - Compute bases of $U = \langle \mathcal{L}_1 \circ f \rangle$ and $V = \langle \mathcal{L} \circ f \rangle$.
  - Decompose U and V under the action of $\mathcal{L}_2$.
  - Obtain T_i from a basis of U_i.

Learning from LB: A framework

$$f = T_1 + \ldots + T_s \,.$$

- A meta-algorithm to learn the terms. $\mathcal{L} = \mathcal{L}_2 \circ \mathcal{L}_1$
  - Compute bases of $U = \langle \mathcal{L}_1 \circ f \rangle$ and $V = \langle \mathcal{L} \circ f \rangle$.
  - Decompose U and V under the action of $\mathcal{L}_2$.
  - Obtain T_i from a basis of U_i.
- Learning the terms –> <u>*vector space decomposition*</u>

Learning from LB: A framework

$$f = T_1 + \ldots + T_s .$$

- A meta-algorithm to learn the terms. $\mathcal{L} = \mathcal{L}_2 \circ \mathcal{L}_1$
  - Compute bases of $U = \langle \mathcal{L}_1 \circ f \rangle$ and $V = \langle \mathcal{L} \circ f \rangle$.
  - Decompose U and V under the action of $\mathcal{L}_2$.
  - Obtain T_i from a basis of U_i.

- Although easy-to-state, one needs to overcome a few <u>technical challenges</u> to make the meta-algorithm work.

Technical challenges

$$f = T_1 + \ldots + T_s .$$

- Task 1 (*Direct Sum*). Show that $U = U_1 \oplus \ldots \oplus U_s$ and $V = V_1 \oplus \ldots \oplus V_s$ w.h.p. if $T_1, \ldots, T_s$ are random.

Technical challenges

$$f = T_1 + \ldots + T_s .$$

- Task 1 (*Direct Sum*). Show that $U = U_1 \oplus \ldots \oplus U_s$ and $V = V_1 \oplus \ldots \oplus V_s$ w.h.p. if $T_1, \ldots, T_s$ are random.
- Task 2 (*Uniqueness*). Show that the above decomposition of U and V under the action of $\mathcal{L}_2$ is unique.

Technical challenges

$$f = T_1 + \ldots + T_s \,.$$

- Task 1 (*Direct Sum*). Show that $U = U_1 \oplus \ldots \oplus U_s$ and $V = V_1 \oplus \ldots \oplus V_s$ w.h.p. if $T_1, \ldots, T_s$ are <u>random</u>.

- Task 2 (*Uniqueness*). Show that the above decomposition of U and V under the action of $\mathcal{L}_2$ is unique.

- A $\mathcal{C}$-circuit satisfying the direct sum and the uniqueness criteria is called a <u>*non-degenerate $\mathcal{C}$-circuit*</u>.
- Task 1 & 2 ≡ Show that a random $\mathcal{C}$-circuit is non-degenerate w.h.p.

Technical challenges

$$f = T_1 + \ldots + T_s .$$

- Task 1 (*Direct Sum*). Show that $U = U_1 \oplus \ldots \oplus U_s$ and $V = V_1 \oplus \ldots \oplus V_s$ w.h.p. if $T_1, \ldots, T_s$ are <u>random</u>.
- Task 2 (*Uniqueness*). Show that the above decomposition of U and V under the action of $\mathcal{L}_2$ is unique.
- Task 3 (*Vector space decomposition*). Carry out the decomposition of U and V under the action of $\mathcal{L}_2$.

Technical challenges

$$f = T_1 + \ldots + T_s\ .$$

- Task 1 (*Direct Sum*). Show that $U = U_1 \oplus \ldots \oplus U_s$ and $V = V_1 \oplus \ldots \oplus V_s$ w.h.p. if $T_1, \ldots, T_s$ are <u>random</u>.
- Task 2 (*Uniqueness*). Show that the above decomposition of U and V under the action of $\mathcal{L}_2$ is unique.
- Task 3 (*Vector space decomposition*). Carry out the decomposition of U and V under the action of $\mathcal{L}_2$.
- Task 4 (*Terms from subspaces*). Recover T_i from U_i .

Known results

Known results that implement the framework:

- Kayal & S. (2019). Proper learns $\sum\wedge\sum$ ckts, tensors, and homogeneous $\sum\prod\sum$ ckts in the *non-degenerate* case.

  - Introduced the framework in a rudimentary form.
  - <u>Proper learns</u> random $\sum\wedge\sum$ circuits for $s \leq \binom{n + d/3}{n}$.

Known results

Known results that implement the framework:

- Kayal & S. (2019). Proper learns $\sum\wedge\sum$ ckts, tensors, and homogeneous $\sum\prod\sum$ ckts in the *non-degenerate* case.
- Garg, Kayal & S. (2020). Proper learns $\sum\wedge\sum\prod^{[t]}$ circuits in the *non-degenerate* case.

  - Laid down the framework completely.
  - The $t = 2$ case has a potential application in learning *mixtures of Gaussians*.

Known results

Known results that implement the framework:

- Kayal & S. (2019). Proper learns $\sum\wedge\sum$ ckts, tensors, and homogeneous $\sum\prod\sum$ ckts in the *non-degenerate* case.
- Garg, Kayal & S. (2020). Proper learns $\sum\wedge\sum\prod^{[t]}$ circuits in the *non-degenerate* case.
- Bhargava, Garg, Kayal & S. (2021). Proper learns generalized $\sum\prod\sum$ circuits in the *non-degenerate* case.

➢ $g_1(\ell_{11}\cdots\ell_{1d}) + \ldots + g_s(\ell_{s1}\cdots\ell_{sd})$; g_i = mono., Det, IMM, etc.

➢ Gives a reasonably general way to accomplish Task 1.

Elaboration on the technical challenges

$$f = T_1 + \ldots + T_s .$$

- Task 1 (*Direct Sum*). Show that $U = U_1 \oplus \ldots \oplus U_s$ and $V = V_1 \oplus \ldots \oplus V_s$ w.h.p. if $T_1, \ldots, T_s$ are random.

- As $\mathcal{L}_1$ and $\mathcal{L}_2$ are linear operators, this task essentially boils down to showing that certain matrices (whose entries are polynomials in the "coefficients" of the terms) have the maximum possible rank.
- The <u>"bad" coefficients lie in an algebraic variety</u>. So, random coefficients are "good".

Elaboration on the technical challenges

$$f = T_1 + \ldots + T_s .$$

- Task 1 (*Direct Sum*). Show that $U = U_1 \oplus \ldots \oplus U_s$ and $V = V_1 \oplus \ldots \oplus V_s$ w.h.p. if $T_1, \ldots, T_s$ are random.

- For a $\Sigma\wedge\Sigma$ circuit, it is fairly easy to show that $\langle \partial^k f \rangle = \langle \ell_1^{d-k} \rangle \oplus \ldots \oplus \langle \ell_s^{d-k} \rangle$ for random $\ell_1, \ldots, \ell_s$.
- Note. Although easy for $\Sigma\wedge\Sigma$ and homogeneous $\Sigma\Pi\Sigma$ circuits, this task is nontrivial for $\Sigma\wedge\Sigma\Pi^{[t]}$ circuits and generalized $\Sigma\Pi\Sigma$ circuits.

Elaboration on the technical challenges

$$f = T_1 + \ldots + T_s \,.$$

- Task 2 (*Uniqueness*). Show that the decomposition of U and V under the action of $\mathcal{L}_2$ is unique.

- Need to understand all possible valid decompositions of U and V under the action of $\mathcal{L}_2$.

- This understanding is provided by the *adjoint algebra*.

The adjoint algebra

- Definition. Let φ: U -> U and ψ: V -> V be linear maps. The *adjoint algebra* associated with $(U, V, \mathcal{L}_2)$ is

$$\mathrm{adj}(U, V, \mathcal{L}_2) := \{(\varphi, \psi) : \lambda \circ \varphi = \psi \circ \lambda, \ \forall \lambda \in \mathcal{L}_2\} .$$

The adjoint algebra

- Definition. Let $\varphi: U \rightarrow U$ and $\psi: V \rightarrow V$ be linear maps. The *adjoint algebra* associated with $(U,V,\mathcal{L}_2)$ is

$$\mathrm{adj}(U,V,\mathcal{L}_2) := \{(\varphi, \psi) : \lambda\circ\varphi = \psi\circ\lambda,\ \forall\lambda \in \mathcal{L}_2\}\ .$$

- Obs. $\mathrm{adj}(U,V,\mathcal{L}_2)$ is a vector space over $\mathbb{F}$.

- Obs. We can compute a basis of $\mathrm{adj}(U,V,\mathcal{L}_2)$ in polynomial time from bases of U and V, and the operators in $\mathcal{L}_2$, by solving a linear system.

The adjoint algebra

- Definition. Let φ: U -> U and ψ: V -> V be linear maps. The *adjoint algebra* associated with $(U,V,\mathcal{L}_2)$ is

$$\mathrm{adj}(U,V,\mathcal{L}_2) := \{(\varphi,\psi) : \lambda\circ\varphi = \psi\circ\lambda,\ \forall\lambda\in\mathcal{L}_2\}.$$

- Obs. If $(\varphi,\psi)\in\mathrm{adj}(U,V,\mathcal{L}_2)$ and φ, ψ are <u>*invertible*</u>, then $U = \varphi(U_1)\oplus\ldots\oplus\varphi(U_s)$ and $V = \psi(V_1)\oplus\ldots\oplus\psi(V_s)$, and $\psi(V_i) = \langle\mathcal{L}_2\circ\varphi(U_i)\rangle$.

The adjoint algebra

- Definition. Let $\varphi: U \to U$ and $\psi: V \to V$ be linear maps. The *adjoint algebra* associated with $(U, V, \mathcal{L}_2)$ is

$$\mathrm{adj}(U, V, \mathcal{L}_2) := \{(\varphi, \psi) : \lambda\circ\varphi = \psi\circ\lambda, \ \forall\lambda \in \mathcal{L}_2\} .$$

- Obs. If $(\varphi, \psi) \in \mathrm{adj}(U, V, \mathcal{L}_2)$ and φ, ψ are <u>*invertible*</u>, then $U = \varphi(U_1) \oplus \dots \oplus \varphi(U_s)$ and $V = \psi(V_1) \oplus \dots \oplus \psi(V_s)$, and $\psi(V_i) = \langle \mathcal{L}_2 \circ \varphi(U_i) \rangle$.
- Proof~. Direct sum follows from the fact that φ, ψ are invertible. For $\lambda \in \mathcal{L}_2$, $\lambda\circ\varphi(U_i) = \psi\circ\lambda(U_i) \subseteq \psi(V_i)$. Equality follows from $V = \langle \mathcal{L}_2 \circ U \rangle$.

The adjoint algebra

- Definition. Let $\varphi: U \to U$ and $\psi: V \to V$ be linear maps. The *adjoint algebra* associated with $(U, V, \mathcal{L}_2)$ is

$$\mathrm{adj}(U, V, \mathcal{L}_2) := \{(\varphi, \psi) : \lambda \circ \varphi = \psi \circ \lambda, \ \forall \lambda \in \mathcal{L}_2\}.$$

- Obs. If $(\varphi, \psi) \in \mathrm{adj}(U, V, \mathcal{L}_2)$ and φ, ψ are <u>*invertible*</u>, then $U = \varphi(U_1) \oplus \ldots \oplus \varphi(U_s)$ and $V = \psi(V_1) \oplus \ldots \oplus \psi(V_s)$, and $\psi(V_i) = \langle \mathcal{L}_2 \circ \varphi(U_i) \rangle$.

- Krull-Schmidt theorem. These are the <u>only</u> decompositions of U and V under the action of $\mathcal{L}_2$.

The adjoint algebra

- Definition. Let $\varphi: U \to U$ and $\psi: V \to V$ be linear maps. The *adjoint algebra* associated with $(U,V,\mathcal{L}_2)$ is

$$\mathrm{adj}(U,V,\mathcal{L}_2) := \{(\varphi,\psi) : \lambda\circ\varphi = \psi\circ\lambda,\ \forall\lambda \in \mathcal{L}_2\}.$$

- Obs. If $(\varphi,\psi) \in \mathrm{adj}(U,V,\mathcal{L}_2)$ and φ, ψ are <u>*invertible*</u>, then $U = \varphi(U_1) \oplus\ldots\oplus \varphi(U_s)$ and $V = \psi(V_1) \oplus\ldots\oplus \psi(V_s)$, and $\psi(V_i) = \langle \mathcal{L}_2 \circ \varphi(U_i)\rangle$.

- We <u>need</u> to understand $\mathrm{adj}(U,V,\mathcal{L}_2)$ to show uniqueness of decomposition. When is the decomposition unique?

The adjoint algebra

- Definition. Let φ: $U \to U$ and ψ: $V \to V$ be linear maps. The *adjoint algebra* associated with $(U, V, \mathcal{L}_2)$ is

$$\mathrm{adj}(U, V, \mathcal{L}_2) := \{(\varphi, \psi) : \lambda \circ \varphi = \psi \circ \lambda,\ \forall \lambda \in \mathcal{L}_2\}.$$

- Let φ_i be the *projection map* from U to U_i, and ψ_i the projection map from V to V_i .
- Obs. $(\varphi_i, \psi_i) \in \mathrm{adj}(U, V, \mathcal{L}_2)$ for all $i \in [s]$.

The adjoint algebra

- Definition. Let $\varphi: U \to U$ and $\psi: V \to V$ be linear maps. The *adjoint algebra* associated with $(U, V, \mathcal{L}_2)$ is
  $$\mathrm{adj}(U, V, \mathcal{L}_2) := \{(\varphi, \psi) : \lambda\circ\varphi = \psi\circ\lambda,\ \forall\lambda \in \mathcal{L}_2\}.$$

- Let φ_i be the *projection map* from U to U_i, and ψ_i the projection map from V to V_i .
- Obs. $(\varphi_i, \psi_i) \in \mathrm{adj}(U, V, \mathcal{L}_2)$ for all $i \in [s]$.
- Proof~. Let $\mathbf{u} = \mathbf{u}_1 + \ldots + \mathbf{u}_s$ for $\mathbf{u} \in U$ and $\mathbf{u}_i \in U_i$. Then, $\lambda\circ\varphi_i(\mathbf{u}) = \lambda(\mathbf{u}_i) \in V_i$.
  Also, $\psi_i\circ\lambda(\mathbf{u}) = \psi_i\circ\lambda(\mathbf{u}_1 + \ldots + \mathbf{u}_s) = \lambda(\mathbf{u}_i)$.

The adjoint algebra

- Definition. Let $\varphi: U \to U$ and $\psi: V \to V$ be linear maps. The *adjoint algebra* associated with $(U, V, \mathcal{L}_2)$ is

$$\mathrm{adj}(U, V, \mathcal{L}_2) := \{(\varphi, \psi) : \lambda \circ \varphi = \psi \circ \lambda,\ \forall \lambda \in \mathcal{L}_2\}.$$

- Let φ_i be the *projection map* from U to U_i, and ψ_i the projection map from V to V_i .
- Obs. $(\varphi_i, \psi_i) \in \mathrm{adj}(U, V, \mathcal{L}_2)$ for all $i \in [s]$.

- The adjoint is *trivial* if it is generated as a vector space over $\mathbb{F}$ by $(\varphi_1, \psi_1), \ldots, (\varphi_s, \psi_s)$.

The adjoint algebra

- Definition. Let φ: U -> U and ψ: V -> V be linear maps. The *adjoint algebra* associated with $(U,V,\mathcal{L}_2)$ is
$$\mathrm{adj}(U,V,\mathcal{L}_2) := \{(\varphi,\psi) : \lambda\circ\varphi = \psi\circ\lambda,\ \forall\lambda \in \mathcal{L}_2\}.$$

- Let φ_i be the <u>*projection map*</u> from U to U_i, and ψ_i the projection map from V to V_i .
- Obs. $(\varphi_i, \psi_i) \in \mathrm{adj}(U,V,\mathcal{L}_2)$ for all $i \in [s]$.

- Clm. If $\mathrm{adj}(U,V,\mathcal{L}_2)$ is <u>trivial</u>, then $U = U_1 \oplus\ldots\oplus U_s$ and $V = V_1 \oplus\ldots\oplus V_s$ is the <u>unique</u> decomposition of U and V under the action of $\mathcal{L}_2$.

The adjoint algebra

- Clm. If $\mathrm{adj}(U,V,\mathcal{L}_2)$ is <u>trivial</u>, then $U = U_1 \oplus \ldots \oplus U_s$ and $V = V_1 \oplus \ldots \oplus V_s$ is the <u>unique</u> decomposition of U and V under the action of $\mathcal{L}_2$.
- Proof~. Let $U = \boldsymbol{\varphi}(U_1) \oplus \ldots \oplus \boldsymbol{\varphi}(U_s)$ and $V = \boldsymbol{\psi}(V_1) \oplus \ldots \oplus \boldsymbol{\psi}(V_s)$ be another decomposition for some $(\boldsymbol{\varphi}, \boldsymbol{\psi}) \in \mathrm{adj}(U,V,\mathcal{L}_2)$, where $\boldsymbol{\varphi}, \boldsymbol{\psi}$ are invertible.

The adjoint algebra

- Clm. If $adj(U,V,\mathcal{L}_2)$ is <u>trivial</u>, then $U = U_1 \oplus \ldots \oplus U_s$ and $V = V_1 \oplus \ldots \oplus V_s$ is the <u>unique</u> decomposition of U and V under the action of $\mathcal{L}_2$.
- Proof~. Let $U = \varphi(U_1) \oplus \ldots \oplus \varphi(U_s)$ and $V = \psi(V_1) \oplus \ldots \oplus \psi(V_s)$ be another decomposition for some $(\varphi, \psi) \in adj(U,V,\mathcal{L}_2)$, where φ, ψ are invertible.

- As $adj(U,V,\mathcal{L}_2)$ is trivial, $\varphi = a_1\varphi_1 + \ldots + a_s\varphi_s$ and $\psi = b_1\psi_1 + \ldots + b_s\psi_s$ for some <u>non-zero</u> $a_i, b_i \in \mathbb{F}$.

The adjoint algebra

- Clm. If $\mathrm{adj}(U,V,\mathcal{L}_2)$ is <u>trivial</u>, then $U = U_1 \oplus \ldots \oplus U_s$ and $V = V_1 \oplus \ldots \oplus V_s$ is the <u>unique</u> decomposition of U and V under the action of $\mathcal{L}_2$.
- Proof~. Let $U = \varphi(U_1) \oplus \ldots \oplus \varphi(U_s)$ and $V = \psi(V_1) \oplus \ldots \oplus \psi(V_s)$ be another decomposition for some $(\varphi, \psi) \in \mathrm{adj}(U,V,\mathcal{L}_2)$, where φ, ψ are invertible.

- As $\mathrm{adj}(U,V,\mathcal{L}_2)$ is trivial, $\varphi = a_1\varphi_1 + \ldots + a_s\varphi_s$ and $\psi = b_1\psi_1 + \ldots + b_s\psi_s$ for some <u>non-zero</u> $a_i, b_i \in \mathbb{F}$.

- Now observe that $\varphi(U_i) = U_i$ and $\psi(V_i) = V_i$.

Elaboration on the technical challenges

$$f = T_1 + \ldots + T_s .$$

- Task 2 (*Uniqueness*). Show that the decomposition of U and V under the action of $\mathcal{L}_2$ is unique.

- This task is accomplished in [GKS'20] and [BGKS'21] by showing that the adjoint algebra $\mathrm{adj}(U, V, \mathcal{L}_2)$ is trivial if $T_1, \ldots, T_s$ are randomly chosen.

Elaboration on the technical challenges

$$f = T_1 + \ldots + T_s \,.$$

- Task 3 (*Vector space decomposition*). Carry out the decomposition of U and V under the action of $\mathcal{L}_2$.

- Chistov, Ivanyos & Karpinski (1997); Eberly (1991); Ronyai (1990); Friedl & Ronyai (1985): There are known efficient vector space decomposition algorithms.

- Work over finite fields, $\mathbb{C}$ and $\mathbb{R}$. Over $\mathbb{Q}$, the output decomposition is over an extension field.

Elaboration on the technical challenges

$$f = T_1 + \ldots + T_s .$$

- Task 3 (*Vector space decomposition*). Carry out the decomposition of U and V under the action of $\mathcal{L}_2$.
- Turns out, if the adjoint is trivial, then the vector space decomposition problem can be reduced to *diagonalizing* a random element of the adj(U,V, $\mathcal{L}_2$).
- Vector space decomposition –> diagonalizing a matrix.

Elaboration on the technical challenges

$$f = T_1 + \ldots + T_s .$$

- Task 3 (*Vector space decomposition*). Carry out the decomposition of U and V under the action of $\mathcal{L}_2$.
- Turns out, if the adjoint is trivial, then the vector space decomposition problem can be reduced to *diagonalizing* a random element of the adj(U,V, $\mathcal{L}_2$).
- The results in [KS'19], [GKS'20] and [BGKS'21] hold over $\mathbb{Q}$.

Elaboration on the technical challenges

$$f = T_1 + \ldots + T_s \, .$$

- Task 4 (*Terms from subspaces*). Recover T_i from U_i .

- Mostly easy, if $\mathcal{L}$ is the set of all partial derivatives.

- Example. For a $\sum\wedge\sum$ circuit, $U_i = \langle \ell_i^{d-k} \rangle$. Obtain a $\mathbb{F}$-multiple of ℓ_i (say, ℓ_i') from U_i using b.b. polynomial factorization. Observe, $f = z_1 \cdot \ell'_1{}^d + \ldots + z_s \cdot \ell'_s{}^d$ for unknown $z_1, \ldots, z_s$. Set up a linear system in $z_1, \ldots, z_s$ as before. Solve it and take d-th roots.

Elaboration on the technical challenges

$$f = T_1 + \ldots + T_s \, .$$

- Task 4 (*Terms from subspaces*). Recover T_i from U_i .
- Mostly easy, if $\mathcal{L}$ is the set of all partial derivatives.
- But not necessarily trivial, if $\mathcal{L}$ is more complex (as is the case in [GKS'20]).

Other average-case learning results

- Gupta, Kayal & Lokam (2011). Proper learns random fanin-2 *multilinear* formulas.

- Gupta, Kayal & Qiao. (2013). Proper learns random fanin-2 *regular* formulas.

- Kayal, Nair & S. (2019). Proper learns random *ABPs* of low width.

Other average-case learning results

- Gupta, Kayal & Lokam (2011). Proper learns random fanin-2 *multilinear* formulas.

- Gupta, Kayal & Qiao. (2013). Proper learns random fanin-2 *regular* formulas.

- Kayal, Nair & S. (2019). Proper learns random *ABPs* of low width.

- These algorithms are implicitly connected to the corresponding lower bounds known for these models.

Learning other circuit models?

- Can we implement the learning from lower bound framework for other circuit models?

Summary

- A survey of known results on polynomial equivalence and average-case reconstruction.

- Polynomial equivalence problem
  - Hessian based equivalence tests.

- Average-case learning
  - A framework for designing learning algorithms from lower bounds based on vector space decomposition.

Thanks!