Deterministic Factorization of Sparse Polynomials with Bounded Individual Degree

Part 1: Sparsity bound for the factors of sparse polynomials using Newton polytopes

Mohith Raju Nagaraju

March 2022

Introduction

Polynomial Factorization

Given $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ a multivariate polynomial over a field $\mathbb{F}$, compute each of the irreducible factors of f.

- For general polynomials, factorization is a hard problem
- So we try to solve the problem for a specific class of polynomials
- Specifically, the class of sparse polynomials with bounded individual degree

Sparsity of a polynomial

The sparsity of f, denoted $||f||$, is the number of monomials (with non-zero coefficients) appearing in f.

- Ex: let $f(\underline{\boldsymbol{x}}) = x_1x_5 + x_3^2 + x_3^2x_4^2x_5^2 + 20$, what is $||f||$?
- Caution: throughout the lecture the variable s will denote sparsity and not size of circuit

Introduction

Bounded individual degree means the individual degrees of each variable is bounded by d

- Ex 1: let $f(\underline{\boldsymbol{x}}) = x_1x_5 + x_3^2 + x_3^2x_4^2x_5^2 + 20$, then the individual degrees of each variable of f is bounded by 2
- Ex 2: If $f(\underline{\boldsymbol{x}})$ is a multilinear polynomial, what can we say?
- Caution: throughout the lecture the variable d will denote the maximum individual degree and not the total degree

What we shall prove

1. A quasi-polynomial* size sparsity bound for factors of sparse polynomials of bounded individual degree
2. A deterministic quasi-polynomial* time algorithm for factoring sparse polynomials of bounded individual degree

Introduction

Example

- Let $f(\underline{\boldsymbol{x}}) = 16x_1^2 - 25x_2^2$
- Want: $g(\underline{\boldsymbol{x}})$ such that $f(\underline{\boldsymbol{x}}) = g(\underline{\boldsymbol{x}}) \cdot h(\underline{\boldsymbol{x}})$
- $g(\underline{\boldsymbol{x}}) = 4x_1 + 5x_2$ works because
- $16x_1^2 - 25x_2^2 = (4x_1 + 5x_2) \cdot (4x_1 - 5x_2)$
- Observe the sparsity of $g(\underline{\boldsymbol{x}})$ is comparable to the sparsity of $f(\underline{\boldsymbol{x}})$
- In general we expect the sparsity of $g(\underline{\boldsymbol{x}})$ to be *not too much bigger than* than the sparsity of $f(\underline{\boldsymbol{x}})$

What we shall prove

1. A quasi-polynomial* size sparsity bound for factors of sparse polynomials of bounded individual degree
2. A deterministic quasi-polynomial* time algorithm for factoring sparse polynomials of bounded individual degree

Main results

Factor sparsity bound

Theorem 1. Let $\mathbb{F}$ be an arbitrary field (finite or otherwise) and let $f(\underline{\boldsymbol{x}}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial of sparsity s and individual degrees at most d, then the sparsity of every factor of $f(\underline{\boldsymbol{x}})$ is bounded by $s^{\mathcal{O}(d^2 \log n)}$.

Factorization sparse polynomials of bounded individual degree

Theorem 2. There exists a deterministic algorithm that given a polynomial $f(\underline{\boldsymbol{x}}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of sparsity s and individual degrees at most d, computes the complete factorization of $f(\underline{\boldsymbol{x}})$, using $s^{\mathcal{O}(d^7 \log n)} \cdot \mathrm{poly}(c_{\mathbb{F}}(d^2))$ field operations.

Main results

Factor sparsity bound

Theorem 1. Let $\mathbb{F}$ be an arbitrary field (finite or otherwise) and let $f(\underline{\boldsymbol{x}}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial of sparsity s and individual degrees at most d, then the sparsity of every factor of $f(\underline{\boldsymbol{x}})$ is bounded by $s^{\mathcal{O}(d^2 \log n)}$.

Factorization sparse polynomials of bounded individual degree

Theorem 2. There exists a deterministic algorithm that given a polynomial $f(\underline{\boldsymbol{x}}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of sparsity s and individual degrees at most d, computes the complete factorization of $f(\underline{\boldsymbol{x}})$, using $s^{\mathcal{O}(d^7 \log n)} \cdot \mathrm{poly}(c_{\mathbb{F}}(d^2))$ field operations.

Where $c_{\mathbb{F}}(d)$ denotes the time of the best known algorithm that factors a univariate polynomial of degree d over $\mathbb{F}$.
(Think $c_{\mathbb{F}} = \mathrm{poly}(d)$)

Chapter 1: Factor sparsity bound

- If $f(\underline{x}) = \sum a_{\underline{i}} \underline{x}^{\underline{i}}$, let $\mathrm{Supp}(f) = \left\{ \underline{i} \;\; | a_{\underline{i}} \neq 0 \right\}$
- We want to compare the size of $\mathrm{Supp}(f)$ and $\mathrm{Supp}(g)$
- To do this we use tools from convex geometry such as the theory of Newton polytopes and an approximate version of Caratheodory's theorem

Polytopes

Convex Span

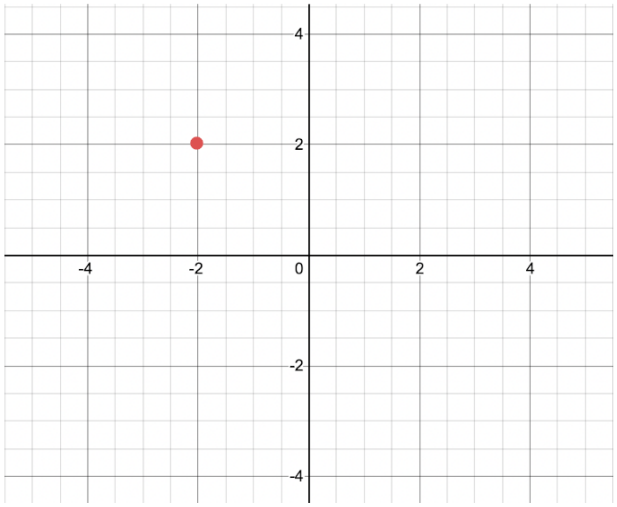
Given a finite set of points $u_1, u_2, \ldots, u_k \in \mathbb{R}^n$, their convex span, which we denote by $CS(u_1, \ldots, u_k)$ is the set defined by

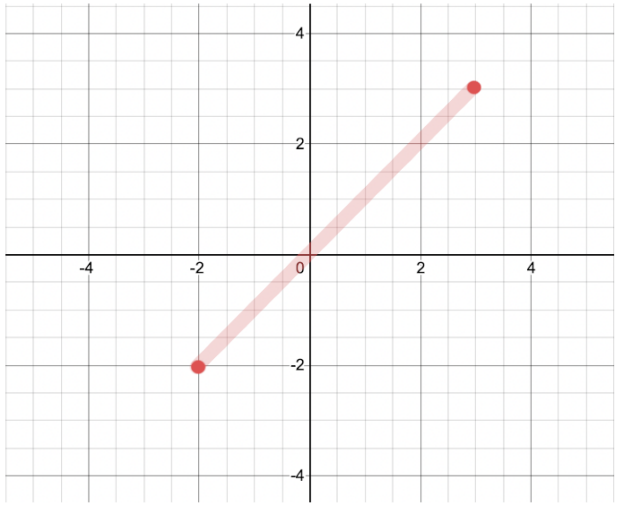
$$CS(u_1, \ldots, u_k) = \left\{ \sum_{i=1}^{k} \lambda_i u_i \;\middle|\; \lambda_i \geq 0 \text{ and } \lambda_1 + \lambda_2 + \cdots + \lambda_k = 1 \right\}$$

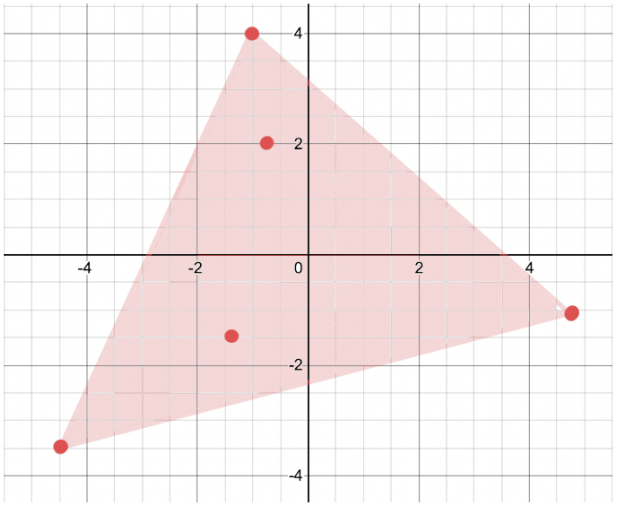
Polytopes

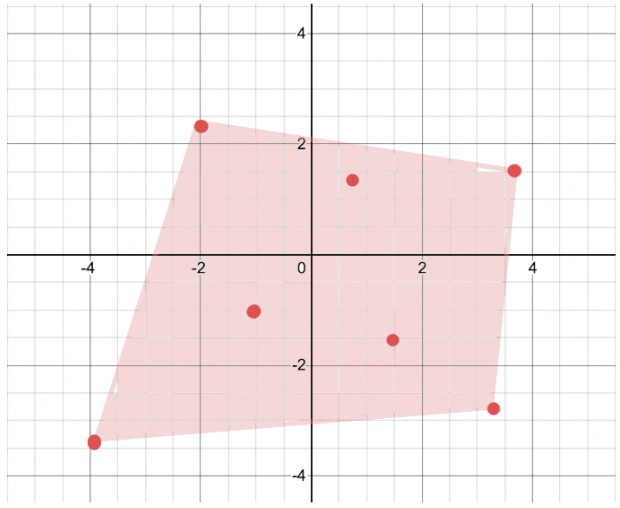
A set $P \subseteq \mathbb{R}^n$ is a called a polytope if there is a finite set of points $u_1, u_2, \ldots, u_k \in \mathbb{R}^n$ such that $P = CS(u_1, \ldots, u_k)$.

Polytopes in $\mathbb{R}^2$

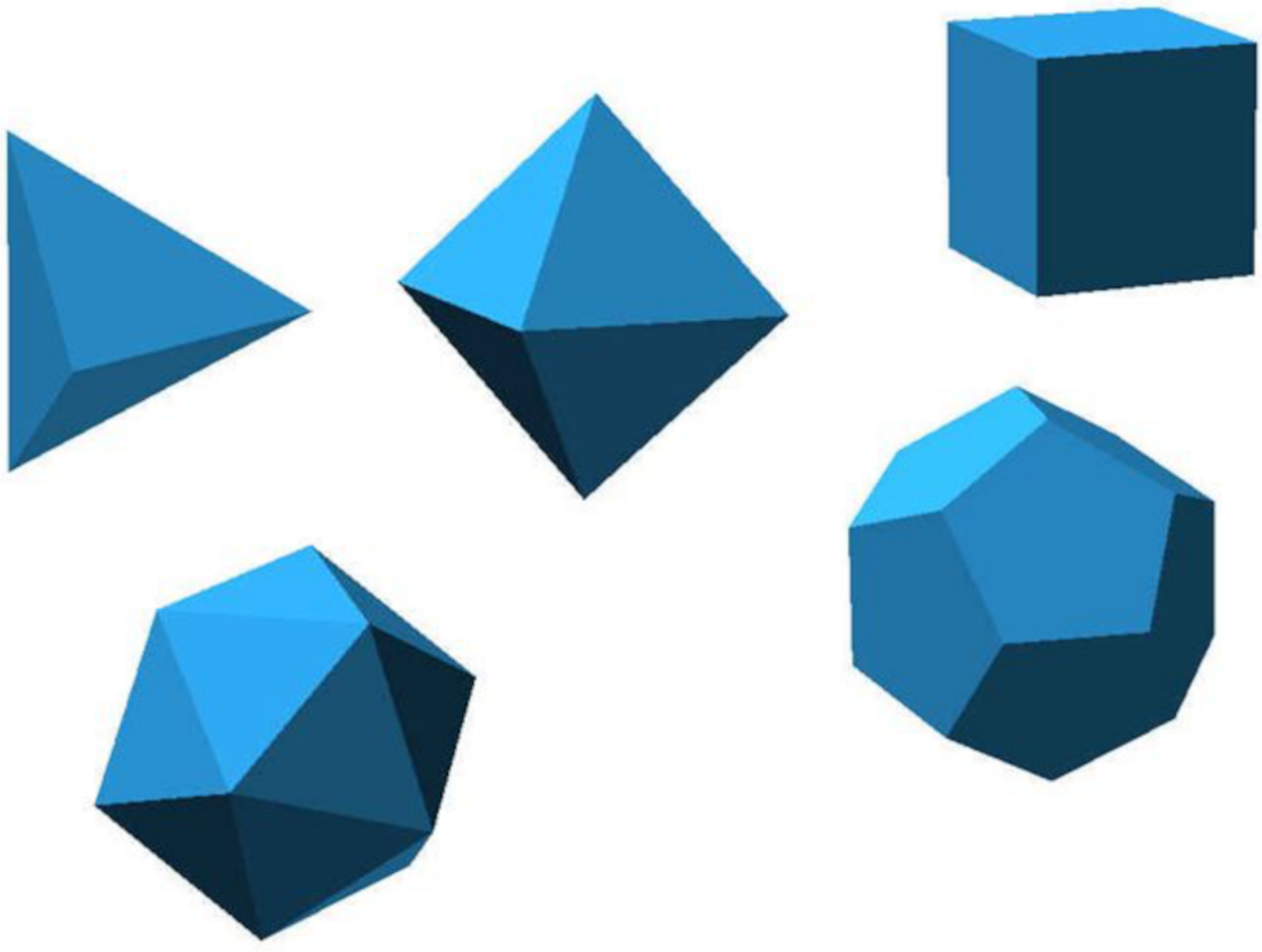








Polytopes in $\mathbb{R}^3$



Polytopes

Vertices of a polytope

- For a polytope P, and a point $v \in P$, we say that v is a vertex of P if it cannot be written as $v = \lambda u + (1-\lambda)w$ for any $u, w \in P \backslash \{v\}$ and $\lambda \in [0,1]$
- Equivalently, $v \in P$ is a vertex if there if a hyperplane H that passes through v such that $P \backslash \{v\}$ lies completely on one side of H
- i.e. there exists an affine function, $h(\underline{\boldsymbol{x}}) = c_0 + c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$, such that $h(v) = 0$ and $h(u) > 0$ for all u in $P \backslash \{v\}$
- We denote the set of vertices of a polytope P by $V(P)$
- Obs 1: If $P = CS(u_1, \ldots, u_k)$ then $V(P) \subseteq \{u_1, u_2, \ldots, u_k\}$
- Cor of obs 1: If S is a finite set of points, say $S = \{u_1, \ldots, u_k\}$, then $|V(CS(S))| \leq |S|$
- Obs 2: Any point in P can be written as a convex combination of the vertices of the polytope, i.e. $CS(S) = CS(V(S))$

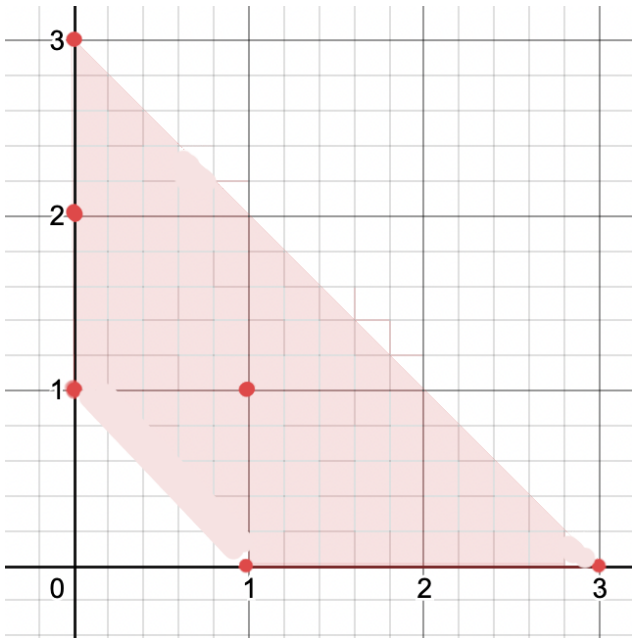
Polytopes and sparsity

Newton polytope

Given $f(\underline{\boldsymbol{x}}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ associate a polytope $P_f \subseteq \mathbb{R}^n$, called the Newton polytope of f, which is the convex hull of points in $\mathrm{Supp}(f)$, i.e. $P_f = CS(\mathrm{Supp}(f))$

Example

If $f(\underline{\boldsymbol{x}}) = x_1^3 + 2x_2^3 + 6x_2^2 - 5x_1x_2 + x_1 - x_2$, then P_f is



Polytopes and sparsity

Obs 3: $|V(P_f)| \leq |\text{Supp}(f)|$

Proof: $|V(P_f)| = |V(CS(\text{Supp}(f)))| \leq |\text{Supp}(f)|$ (inequality follows from Obs 1)

Question: Is there a "not too big function" ϕ such that $\phi(|V(P_f)|) \geq |\text{Supp}(f)|$?

Answer: Yes! $\phi(t) = t^{\mathcal{O}(d^2 \log n)}$

Theorem 4.2: Let $E \subseteq \{0, 1, \ldots, d\}^n$. There is an absolute constant C such that $|V(CS(E))|^{Cd^2 \log n} \geq |E|$

Moral: $||f|| = |\text{Supp}(f)|$ is roughly comparable to $|V(P_f)|$ i.e. we have $|V(P_f)|^{\mathcal{O}(d^2 \log n)} \geq ||f|| \geq |V(P_f)|$

Polytopes and sparsity

Recall, if $f(\underline{x}) = g(\underline{x}) \cdot h(\underline{x})$, our aim was to upperbound $||g||$ using $||f||$, i.e. upperbound $|\text{Supp}(g)|$ with $|\text{Supp}(f)|$. We do this by upperbounding $|V(P_g)|$ by $|V(P_f)|$.

Example

- Let $g(\underline{x}) = (x_1^2 - x_2^2)$, $h(\underline{x}) = (1 + x_1 + x_2)$ and $f(\underline{x}) = g(\underline{x}) \cdot h(\underline{x})$
- Observe:

$$\begin{aligned} f(\underline{x}) &= (x_1^2 x_2^0 - x_1^0 x_2^2) \cdot (x_1^0 x_2^0 + x_1^1 x_2^0 + x_1^0 x_2^1) \\ &= x_1^{2+0} x_2^{0+0} + x_1^{2+1} x_2^{0+0} + \cdots \end{aligned}$$

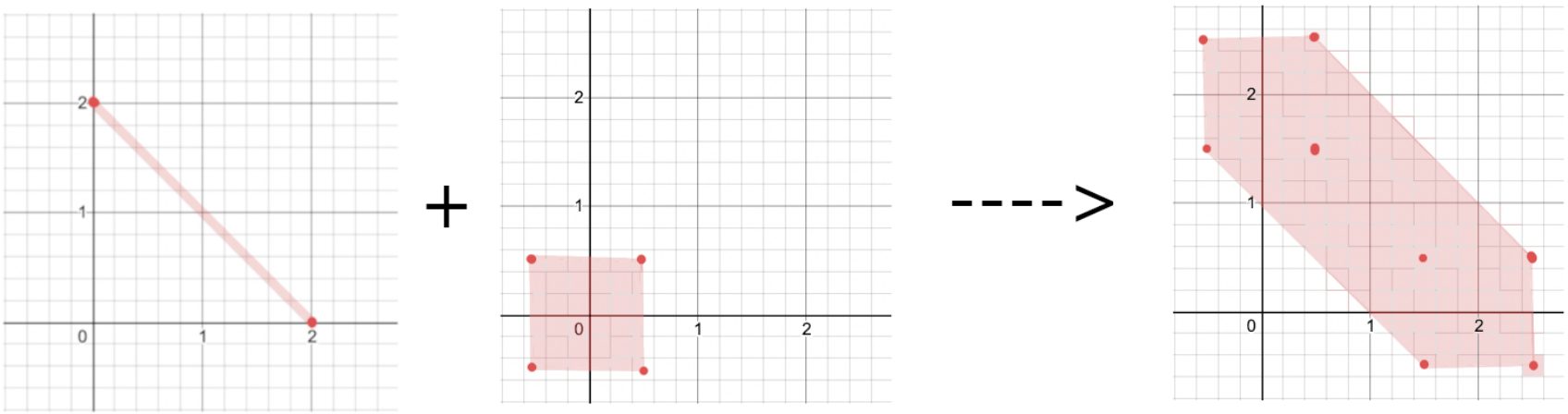
- $\text{Supp}(g) = \{ (2,0), (0,2) \}$, $\text{Supp}(h) = \{ (0,0), (1,0), (0,1) \}$ and $\text{Supp}(f) = \{ (2,0), (3,0), (2,1), (0,2), (1,2), (0,3) \}$
- Obs 4: $\text{Supp}(f) \subseteq \text{Supp}(g) + \text{Supp}(h)$
- This motivates us to look at $P_g + P_h$, i.e. the Minkowski sum of the Newton polytopes of f and g

Polytopes and sparsity

Minkowski sum of polytopes

Given two polytopes P_1 and P_2 in $\mathbb{R}^n$, we define their Minkowski Sum $P_1 + P_2$ to be the set of points given by

$$P_1 + P_2 = \{ u_1 + u_2 \mid u_1 \in P_1 \text{ and } u_2 \in P_2 \}$$



Minkowski sum of polytopes

Proposition 3.2. Let P_1 and P_2 be polytopes in $\mathbb{R}^n$. Then their Minkowski sum $P_1 + P_2$ is a polytope and

$$|V(P_1 + P_2)| \geq \max\{|V(P_1)|, |V(P_2)|\}.$$

Proposition 3.3. Let $f, g, h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be polynomials such that $f = g \cdot h$. Then $P_f = P_g + P_h$.

Corollary 3.5. Let $f, g, h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be polynomials such that $f = g \cdot h$. Then

$$||f|| \geq |V(P_f)| = |V(P_g + P_h)| \geq |V(P_g)|.$$

Note: Cor 3.5 follows from Obs 3, Prop 3.3, and Prop 3.2.

Minkowski sum of polytopes

Proposition 3.2. Let P_1 and P_2 be polytopes in $\mathbb{R}^n$. Then their Minkowski sum $P_1 + P_2$ is a polytope and

$$|V(P_1 + P_2)| \geq \max\{|V(P_1)|, |V(P_2)|\}.$$

Proposition 3.3. Let $f, g, h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be polynomials such that $f = g \cdot h$. Then $P_f = P_g + P_h$.

Corollary 3.5. Let $f, g, h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be polynomials such that $f = g \cdot h$. Then

$$||f|| \geq |V(P_f)| = |V(P_g + P_h)| \geq |V(P_g)|.$$

Observe: The sparsity bound follows from Cor 3.5 and Thm 4.2:

$$||f|| \geq |V(P_g)| \geq ||g||^{\frac{1}{\mathcal{O}(d^2 \log n)}}$$

$$||f||^{\mathcal{O}(d^2 \log n)} \geq ||g|| \qquad \blacksquare$$

$P_1 + P_2$ is a polytope

Proposition 3.2. Let P_1 and P_2 be polytopes in $\mathbb{R}^n$. Then their Minkowski sum $P_1 + P_2$ is a polytope and

$$|V(P1 + P2)| \geq \max\{|V(P_1)|, |V(P_2)|\}.$$

Proof: Recall $P_1 + P_2 = \{\, u + u' \mid u \in P_1 \text{ and } u' \in P_2 \,\}$. To show $P_1 + P_2$ is a polytope we need to show there exists a finite set of points $u_1, u_2, \ldots, u_k$ such that $P_1 + P_2 = CS(u_1, u_2, \ldots, u_k)$.

$P_1 + P_2$ is a polytope

Proposition 3.2. Let P_1 and P_2 be polytopes in $\mathbb{R}^n$. Then their Minkowski sum $P_1 + P_2$ is a polytope and

$$|V(P1 + P2)| \geq \max\{|V(P_1)|, |V(P_2)|\}.$$

Proof: Recall $P_1 + P_2 = \{\, u + u' \mid u \in P_1 \text{ and } u' \in P_2 \,\}$. To show $P_1 + P_2$ is a polytope we need to show there exists a finite set of points $u_1, u_2, \ldots, u_k$ such that $P_1 + P_2 = CS(u_1, u_2, \ldots, u_k)$.

Claim 1: $P_1 + P_2 = CS(V(P_1) + V(P_2))$

Pf: Let $V(P_1) = \{v_1, v_2, \ldots, v_n\}$ and $V(P_2) = \{v'_1, v'_2, \ldots, v'_m\}$
Given $u + u' \in P_1 + P_2$, where $u = \sum_i \lambda_i v_i$ and $u' = \sum_j \lambda'_j v'_j$

$$u + u' = \sum_{i,j} \lambda_i \lambda'_j (v_i + v'_j)$$

which belongs to $CS(V(P_1) + V(P_2))$

$P_1 + P_2$ is a polytope

Proposition 3.2. Let P_1 and P_2 be polytopes in $\mathbb{R}^n$. Then their Minkowski sum $P_1 + P_2$ is a polytope and

$$|V(P1 + P2)| \geq \max\{|V(P_1)|, |V(P_2)|\}.$$

Proof: Recall $P_1 + P_2 = \{ u + u' \mid u \in P_1 \text{ and } u' \in P_2 \}$. To show $P_1 + P_2$ is a polytope we need to show there exists a finite set of points $u_1, u_2, \ldots, u_k$ such that $P_1 + P_2 = CS(u_1, u_2, \ldots, u_k)$.

Claim 1: $P_1 + P_2 = CS(V(P_1) + V(P_2))$

Pf: Let $V(P_1) = \{v_1, v_2, \ldots, v_n\}$ and $V(P_2) = \{v'_1, v'_2, \ldots, v'_m\}$
Given $w \in CS(V(P_1) + V(P_2))$, $w = \sum_i \lambda_{i,j} \tilde{\lambda}_{ij} (v_i + v'_j)$
Put $\lambda_i = \sum_j \tilde{\lambda}_{ij}$ and $\lambda'_j = \sum_i \tilde{\lambda}_{ij}$
Observe $w = \sum_i \lambda_i v_i + \sum_j \lambda'_j v'_j$
which belongs to $P_1 + P_2$ $\square$

Proof of $|V(P1 + P2)| \geq \max\{|V(P_1)|, |V(P_2)|\}$

Proposition 3.2. Let P_1 and P_2 be polytopes in $\mathbb{R}^n$. Then their Minkowski sum $P_1 + P_2$ is a polytope and

$$|V(P1 + P2)| \geq \max\{|V(P_1)|, |V(P_2)|\}.$$

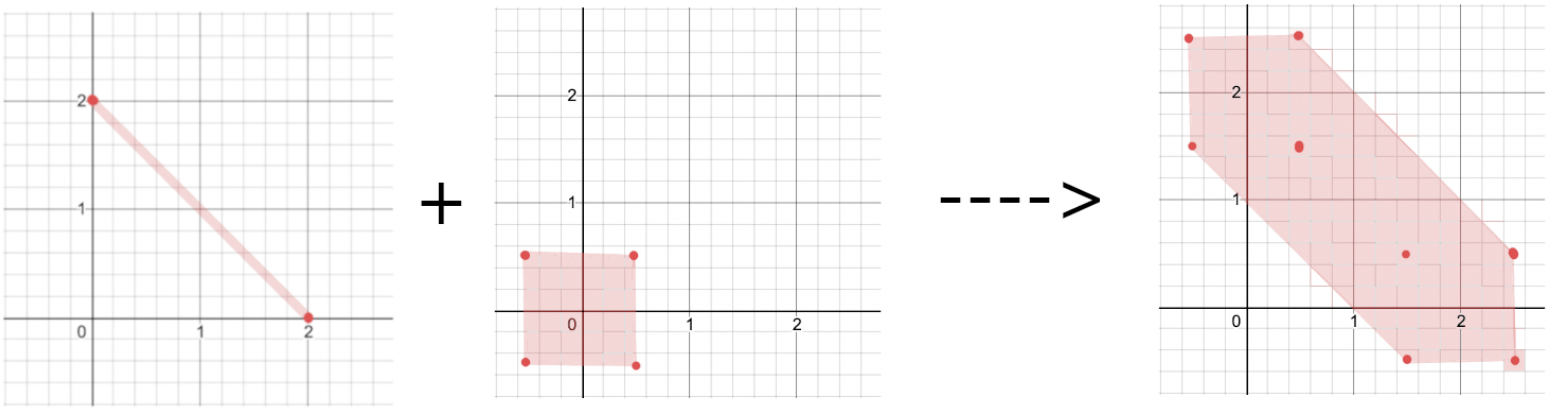
Proof: Observe that it suffices to show $|V(P1 + P2)| \geq |V(P_1)|$

As $P_1 + P_2 = CS(V(P_1) + V(P_2))$, by Obs 1

$V(P_1 + P_2) \subseteq V(P_1) + V(P_2)$ . . . . . (Call this Obs 5)

Observe that $V(P_1 + P_2) \subseteq \bigcup_{v \in V(P_1)} (v + V(P_2))$

Claim 2: For each v in $V(P_1)$, $v + V(P_2)$ contains a vertex of $P_1 + P_2$



Proof of $|V(P1 + P2)| \geq \max\{|V(P_1)|, |V(P_2)|\}$

Proposition 3.2. Let P_1 and P_2 be polytopes in $\mathbb{R}^n$. Then their Minkowski sum $P_1 + P_2$ is a polytope and

$$|V(P1 + P2)| \geq \max\{|V(P_1)|, |V(P_2)|\}.$$

Proof: Observe that it suffices to show $|V(P1 + P2)| \geq |V(P_1)|$

Claim 2: For each v in $V(P_1)$, $v + V(P_2)$ contains a vertex of $P_1 + P_2$

Claim 3: If $v_1, \tilde{v}_1 \in V(P_1)$, and $v_1 \neq \tilde{v}_1$, then the vertices of $P_1 + P_2$ inside $v_1 + V(P_2)$ and $\tilde{v}_1 + V(P_2)$ are distinct, i.e. if $v_1 + v_2 \in v_1 + V(P_2)$ and $\tilde{v}_1 + \tilde{v}_2 \in \tilde{v}_1 + V(P_2)$ are vertices of $P_1 + P_2$, then $v_1 + v_2 \neq \tilde{v}_1 + \tilde{v}_2$ if $v_1 \neq \tilde{v}_1$. (HW)

Proof of $|V(P1 + P2)| \geq \max\{|V(P_1)|, |V(P_2)|\}$

Proposition 3.2. Let P_1 and P_2 be polytopes in $\mathbb{R}^n$. Then their Minkowski sum $P_1 + P_2$ is a polytope and

$$|V(P1 + P2)| \geq \max\{|V(P_1)|, |V(P_2)|\}.$$

Proof: Observe that it suffices to show $|V(P1 + P2)| \geq |V(P_1)|$

Claim 2: for each v in $V(P_1)$, $v + V(P_2)$ contains a vertex of $P_1 + P_2$

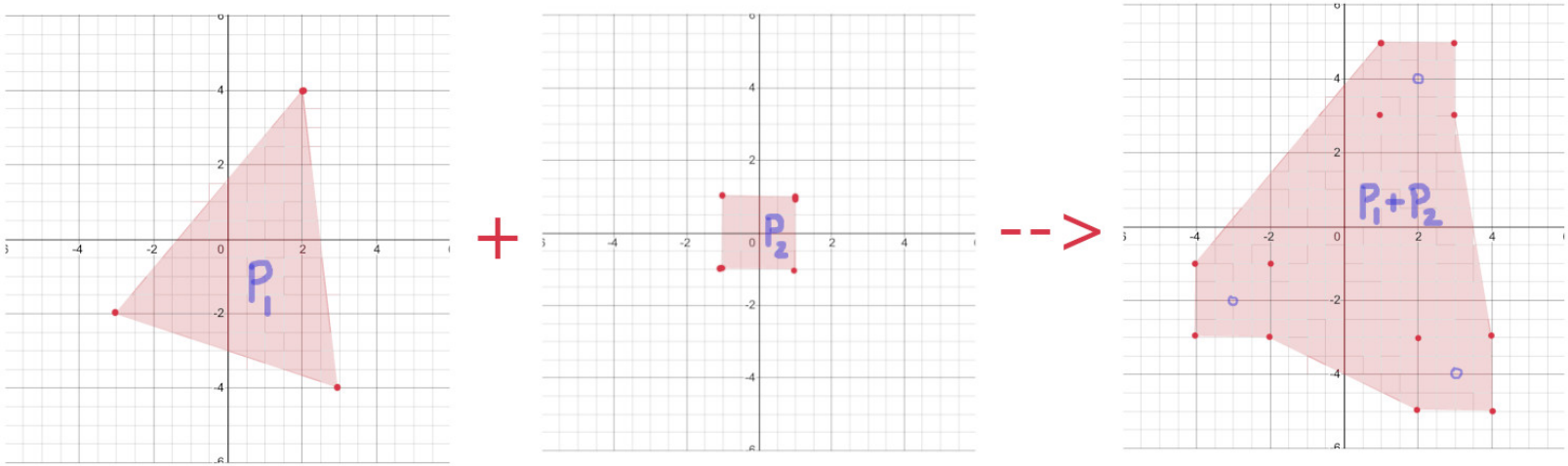
Claim 3: If $v_1, \tilde{v}_1 \in V(P_1)$, and $v_1 \neq \tilde{v}_1$, then the vertices of $P_1 + P_2$ inside $v_1 + V(P_2)$ and $\tilde{v}_1 + V(P_2)$ are distinct, i.e. if $v_1 + v_2 \in v_1 + V(P_2)$ and $\tilde{v}_1 + \tilde{v}_2 \in \tilde{v}_1 + V(P_2)$ are vertices of $P_1 + P_2$, then $v_1 + v_2 \neq \tilde{v}_1 + \tilde{v}_2$ if $v_1 \neq \tilde{v}_1$. (HW)

Claim 2 and 3 gives the result because, for each v in $V(P_1)$, $v + V(P_2)$ contains a distinct vertex of $P_1 + P_2$ and hence $P_1 + P_2$ has at least $|V(P_1)|$ vertices ■

$v + V(P_2)$ contains a vertex

Claim 2: for each v in $V(P_1)$, $v + V(P_2)$ contains a vertex of $P_1 + P_2$

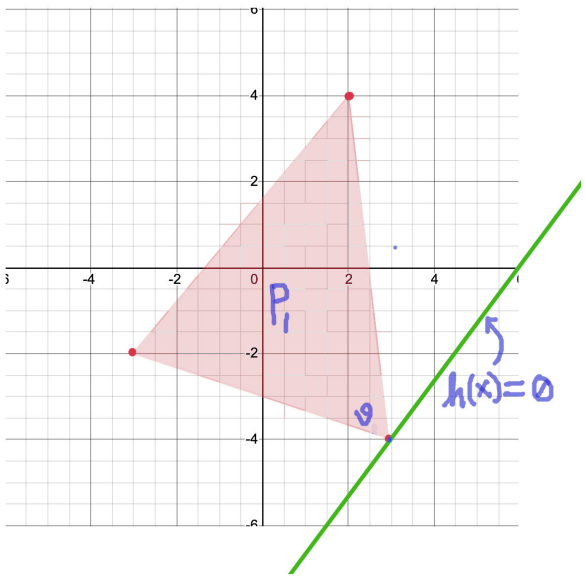
- Recall, v is a vertex of P_1 if and only if there is an hyperplane H passing through v such that $P_1 \backslash \{v\}$ lies on one side of H, i.e. there is an affine map h such that $h(v) = 0$ and $h(u) < 0$ for all $u \in P_1 \backslash \{v\}$ (Note, H is the zero set of h)



$v + V(P_2)$ contains a vertex

Claim 2: for each v in $V(P_1)$, $v + V(P_2)$ contains a vertex of $P_1 + P_2$

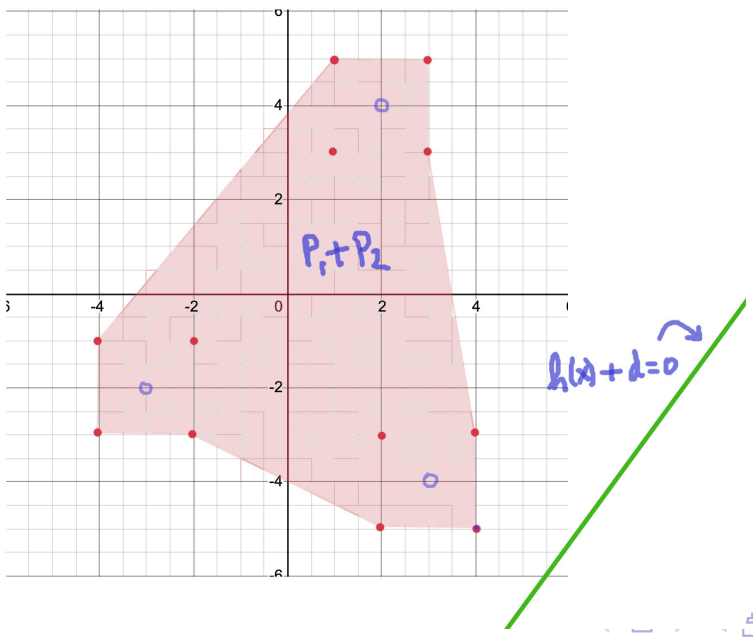
- Recall, v is a vertex of P_1 if and only if there is an hyperplane H passing through v such that $P_1 \backslash \{v\}$ lies on one side of H, i.e. there is an affine map h such that $h(v) = 0$ and $h(u) < 0$ for all $u \in P_1 \backslash \{v\}$ (Note, H is the zero set of h)



$v + V(P_2)$ contains a vertex

Claim 2: for each v in $V(P_1)$, $v + V(P_2)$ contains a vertex of $P_1 + P_2$

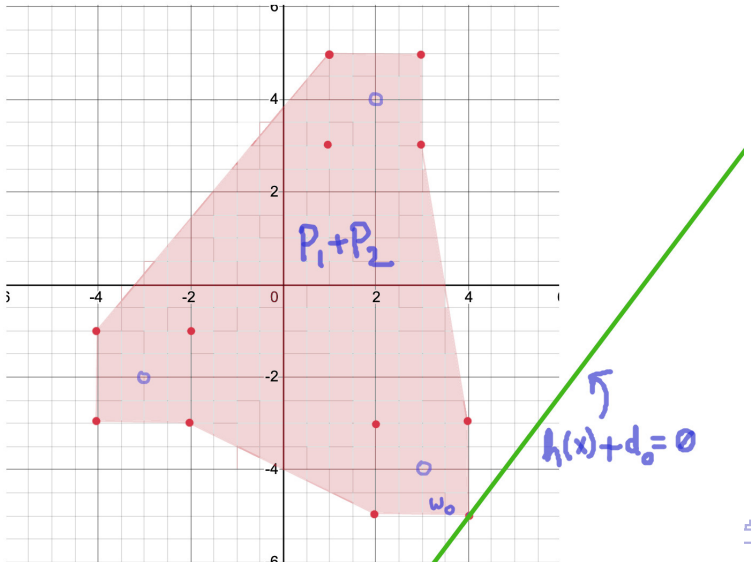
- Recall, $v \in V(P_1)$ iff there is an affine map h such that $h(v) = 0$ and $h(u) < 0$ for all $u \in P_1 \backslash \{v\}$
- As the polytope $P_1 + P_2$ is bounded, there is a large d such that $h(w) + d < 0$ for all $w \in P_1 + P_2$, i.e. we can translate the hyperplane H so that $P_1 + P_2$ lies completely on one side of H



$v + V(P_2)$ contains a vertex

Claim 2: for each v in $V(P_1)$, $v + V(P_2)$ contains a vertex of $P_1 + P_2$

- Recall, $v \in V(P_1)$ iff there is an affine map h such that $h(v) = 0$ and $h(u) < 0$ for all $u \in P_1 \backslash \{v\}$
- As the polytope $P_1 + P_2$ is bounded, there is a large d such that $h(w) + d < 0$ for all $w \in P_1 + P_2$
- Now slowly decrease the value of d. By slowly decreasing we will arrive at a d_0 such that $h(w_0) + d_0 = 0$ for some w_0 in $P_1 + P_2$ and $h(w) + d_0 < 0$ for all $w \in P_1 + P_2$



$v + V(P_2)$ contains a vertex

Claim 2: for each v in $V(P_1)$, $v + V(P_2)$ contains a vertex of $P_1 + P_2$

- Recall, $v \in V(P_1)$ if there is an affine map h such that $h(v) = 0$ and $h(u) < 0$ for all $u \in P_1 \backslash \{v\}$
- As the polytope $P_1 + P_2$ is bounded, there is a large d such that $h(w) + d < 0$ for all $w \in P_1 + P_2$
- Arrive at a d_0 such that $h(w_0) + d_0 = 0$ for some w_0 in $P_1 + P_2$ and $h(w) + d_0 < 0$ for all $w \in P_1 + P_2$
- By definition, w_0 is a vertex of $P_1 + P_2$.
  As $V(P_1 + P_2) \subseteq V(P_1) + V(P_2)$, $w_0 = v_1 + v_2$ for some $v_1 \in V(P_1)$ and $v_2 \in V(P_2)$
- Sub claim: $v_1 = v$

  Pf(HW): Follows beacuse
  - $h(v) > h(v_1)$ for all $v_1 \in V(P_1) \backslash \{v\}$
  - $h(r + s) = h(r) + h(s) - c_0$ as h is affine $\square$

$P_f = P_g + P_h$

Proposition 3.3. Let $f, g, h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be polynomials such that $f = g \cdot h$. Then $P_f = P_g + P_h$.

Proof: Recall Obs 4

$$\begin{aligned} \text{Supp}(f) &\subseteq \text{Supp}(g) + \text{Supp}(h) \\ &\subseteq P_g + P_h \end{aligned}$$

Taking Convex hull on both sides,

$$\begin{aligned} CS(\text{Supp}(f)) &\subseteq CS(P_g + P_h) \\ P_f &\subseteq CS(P_g + P_h) \end{aligned}$$

$CS(P_g + P_h) = P_g + P_h$ as $P_g + P_h$ is a polytope. Hence,

$$P_f \subseteq P_g + P_h$$

$P_f = P_g + P_h$

Proposition 3.3. Let $f, g, h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be polynomials such that $f = g \cdot h$. Then $P_f = P_g + P_h$.

Proof: Recall Obs 4

$$\begin{aligned} \text{Supp}(f) &\subseteq \text{Supp}(g) + \text{Supp}(h) \\ &\subseteq P_g + P_h \end{aligned}$$

Taking Convex hull on both sides,

$$\begin{aligned} CS(\text{Supp}(f)) &\subseteq CS(P_g + P_h) \\ P_f &\subseteq CS(P_g + P_h) \end{aligned}$$

$CS(P_g + P_h) = P_g + P_h$ as $P_g + P_h$ is a polytope. Hence,

$$P_f \subseteq P_g + P_h$$

To show $P_g + P_h \subseteq P_f$ it suffices to show $V(P_g + P_h)$ lies in P_f.

Claim: $V(P_g + P_h) \subseteq \text{Supp}(f) \subseteq P_f$

$P_f = P_g + P_h$

Claim: $V(P_g + P_h) \subseteq \text{Supp}(f) \subseteq P_f$

Proof: Let $g(\underline{x}) = \sum b_{\underline{i}} \underline{x}^{\underline{i}}$ and $h(\underline{x}) = \sum c_{\underline{i}} \underline{x}^{\underline{i}}$.
Let $v_1 + v_2 \in V(P_g + P_h)$.
Obs 5 says $V(P_g + P_h) \subseteq V(P_g) + V(P_h)$.
$\Rightarrow v_1 \in V(P_g)$ and $v_2 \in V(P_h)$.
$\Rightarrow b_{v_1} \underline{x}^{v_1}$ and $c_{v_2} \underline{x}^{v_2}$ are non-zero terms in g and h.
$\Rightarrow$ the term $b_{v_1} c_{v_2} \underline{x}^{v_1+v_2}$ is a non-zero term in $g \cdot h$.
We can safely say $v_1 + v_2$ is in $\text{Supp}(f)$ if there is no other non-zero term $d\underline{x}^{v_1+v_2}$ in $g \cdot h$.

$P_f = P_g + P_h$

Claim: $V(P_g + P_h) \subseteq \text{Supp}(f) \subseteq P_f$

Proof: Let $g(\underline{x}) = \sum b_{\underline{i}} \underline{x}^{\underline{i}}$ and $h(\underline{x}) = \sum c_{\underline{i}} \underline{x}^{\underline{i}}$.
Let $v_1 + v_2 \in V(P_g + P_h)$.
Obs 5 says $V(P_g + P_h) \subseteq V(P_g) + V(P_h)$.
$\Rightarrow v_1 \in V(P_g)$ and $v_2 \in V(P_h)$.
$\Rightarrow b_{v_1} \underline{x}^{v_1}$ and $c_{v_2} \underline{x}^{v_2}$ are non-zero terms in g and h.
$\Rightarrow$ the term $b_{v_1} c_{v_2} \underline{x}^{v_1+v_2}$ is a non-zero term in $g \cdot h$.
We can safely say $v_1 + v_2$ is in $\text{Supp}(f)$ if there is no other non-zero term $d\underline{x}^{v_1+v_2}$ in $g \cdot h$.
Assume there is a non-zero term $d\underline{x}^{v_1+v_2}$ in $g \cdot h$.

- $\Rightarrow d\underline{x}^{v_1+v_2} = b_{u_1} \underline{x}^{u_1} c_{u_2} \underline{x}^{u_2}$ such that $v_1 \neq u_1$ and $v_2 \neq u_2$
- $\Rightarrow u_1 \in \text{Supp}(g) \subseteq P_g$ and $u_2 \in \text{Supp}(h) \subseteq P_h$
- Observe $u_1 + v_2$ and $v_1 + u_2$ are distinct points in $P_g + P_h$
- $\Rightarrow u_1 + u_2 = v_1 + v_2$ lies on the line joining $u_1 + v_2$ and $v_1 + u_2$
- $\Rightarrow u_1 + u_2 = v_1 + v_2$ is not a vertex of $P_g + P_h$ which is a contradiction.

Thus there is no other non-zero term $d\underline{x}^{v_1+v_2}$ in $g \cdot h$ and we can safely say $v_1 + v_2$ is in $\text{Supp}(f)$ ■

$|V(P_f)|^{\mathcal{O}(d^2 \log n)} \geq ||f||$

Theorem 4.2: Let $E \subseteq \{0, 1, \ldots, d\}^n$. There is an absolute constant C such that $|V(CS(E))|^{Cd^2 \log n} \geq |E|$

Why would we expect this theorem to be true?

If the number of vertices is too low, there cannot be too many lattice points inside the convex span of the vertices, i.e. $\{0, 1, \ldots, d\}^n \cap CS(V)$ is bounded above by some function of $|V|$.

Let $U \subseteq \mathbb{R}^n$. A vector $\mu \in CS(U)$ is said to k-uniform wrt U if

$$\mu = \sum_i \frac{N_i}{k} u_i \quad \text{where } u_i \in U$$

Obs 6: The number of distinct k-uniform vectors wrt U is at most $|U|^k$

Pf: Observe that a vector μ is k-uniform if there is a *multiset* of $\{1, 2, \ldots, |U|\}$ of size k such that $\mu = \frac{1}{k} \sum_{i \in S} u_i$

Approximate Caratheodory theorem

Theorem 3.6: Let $U = \{u_1, u_2, \ldots, u_m\} \subseteq \mathbb{R}^n$ with $||u_i||_\infty \leq 1$ for all $1 \leq i \leq m$. Given $\epsilon > 0$ for every $w \in CS(U)$ there exists an $\mathcal{O}\left(\frac{\log n}{\epsilon^2}\right)$-uniform vector $\mu \in CS(U)$ such that $||w - \mu||_\infty \leq \epsilon$.

Proof: (probabilistic method)

- Fix $w \in CS(U)$. Let $w = \sum_{i=1}^{m} a_i u_i$ where $a_i \geq 0$ and $\sum a_i = 1$
- Consider the probability distribution on U, where the probability of sampling u_i is a_i
- Pick $k = \left(\frac{\log n}{\epsilon^2}\right)$ samples independently from the distribution and let the vectors be $v_1, v_2, \ldots, v_k$
- Let $\mu = \frac{\sum_{i=1}^{k} v_i}{k}$
- Let $\mu^{(j)}$, $u_i^{(j)}, w^{(j)}$ denote the $j-$th component of μ, u_i, w respectively

Claim For any coordinate $j \in [n]$, $\Pr\left[\ |w^{(j)} - \mu^{(j)}| > \epsilon\right] \leq 1/n$

Approximate Caratheodory theorem

Claim For any coordinate $j \in [n]$, $\Pr\left[\ \left|w^{(j)} - \mu^{(j)}\right| > \epsilon\right] < 1/n$

Proof:

- Fix a coordinate j
- Let Y be a random variable such that $Y = u_i^{(j)}$ with probability a_i
- Let $Y_1, Y_2, \ldots, Y_k$ be k independent samples of the random variable Y
- Observe $\mathbb{E}[Y] = w^{(j)}$ and $\mu^{(j)} = \sum \frac{Y_i}{k}$
- By Chernoff-Hoeffding inequality,

$$\begin{aligned}\Pr\left[\ \left|w^{(j)} - \mu^{(j)}\right| > \epsilon\right] &= \Pr\left[\ \left|\mathbb{E}[Y] - \sum Y_i/n\right| > \epsilon\right] \\ &< 2\exp(-2\epsilon^2 k) \\ &< 1/n \qquad \square\end{aligned}$$

Approximate Caratheodory theorem

Theorem 3.6: Let $U = \{u_1, u_2, \ldots, u_m\} \subseteq \mathbb{R}^n$ with $||u_i||_\infty \leq 1$ for all $1 \leq i \leq m$ Given $\epsilon > 0$ for every $w \in CS(U)$ there exists an $\mathcal{O}\left(\frac{\log n}{\epsilon^2}\right)$-uniform vector $\mu \in CS(U)$ such that $||w - \mu||_\infty \leq \epsilon$.

Proof:

Claim For any coordinate $j \in [n]$, $\Pr\left[\ |w^{(j)} - \mu^{(j)}|\ \right] \leq 1/n$ $\square$

To finish the proof, we take a union bound.

$$\begin{aligned}
\Pr\left[||w - \mu||_\infty \leq \epsilon\right] &= \Pr\left[\cap_{j=1}^n |w^{(j)} - \mu^{(j)}| \leq \epsilon\right] \\
&= 1 - \Pr\left[\cup_{j=1}^n |w^{(j)} - \mu^{(j)}| > \epsilon\right] \\
&\geq 1 - \sum_{j=1}^n \Pr\left[|w^{(j)} - \mu^{(j)}| > \epsilon\right] \\
&> 1 - \sum_{j=1}^n 1/n \quad \blacksquare
\end{aligned}$$

$|V(P_f)|^{\mathcal{O}(d^2 \log n)} \geq ||f||$

Theorem 4.2: Let $E \subseteq \{0, 1, \ldots, d\}^n$. There is an absolute constant C such that $|V(CS(E))|^{Cd^2 \log n} \geq |E|$

Proof:

- Given $E \subseteq \{0, 1, \ldots, d\}^n$, we scale it down by $1/d$ to get $E_d \subseteq [0, 1]^n$
- Let U_d be the vertex set of $CS(E_d)$. Note: $|U_d| = |V(CS(E))|$
- Let $\epsilon = 1/3d$
- We choose such an epsilon because if u_1, u_2 are distinct elements of $U_d \subseteq E_d$, and if μ_1, μ_2 are uniform vectors that are $\epsilon = 1/3d$ close to u_1, u_2 respectively, then $\mu_1 \neq \mu_2$ (This is because of triangle inequality)
- Thm 3.6 tells us for every $u \in E_d$, there is $\mathcal{O}\left(\frac{\log n}{\epsilon^2}\right) = \mathcal{O}(d^2 \log n)$ uniform vector $\mu \in CS(E_d)$ such that $||u - \mu||_\infty \leq 1/3d$
- By choice of ϵ, two distinct $u_1, u_2 \in U_d$ will have distinct uniform vectors μ_1, μ_2 such that $||u_i - \mu_i||_\infty \leq 1/3d$

$|V(P_f)|^{\mathcal{O}(d^2 \log n)} \geq ||f||$

Theorem 4.2: Let $E \subseteq \{0, 1, \ldots, d\}^n$. There is an absolute constant C such that $|V(CS(E))|^{Cd^2 \log n} \geq |E|$

Proof:

- By choice of ϵ, two distinct $u_1, u_2 \in U_d$ will have distinct uniform vectors μ_1, μ_2 such that $||u_i - \mu_i||_\infty \leq 1/3d$
- $|E| = |E_d|$ is bounded above by the number of distinct $\mathcal{O}(d^2 \log n)$ uniform vectors wrt U_d
- By Obs 6, the number of distinct $\mathcal{O}(d^2 \log n)$-uniform vector wrt U_d is bounded above by $|U_d|^{\mathcal{O}(d^2 \log n)}$

$$
\begin{aligned}
|E| &\leq \text{Number of } \mathcal{O}(d^2 \log n)\text{-uniform vectors} \\
&\leq |U_d|^{\mathcal{O}(d^2 \log n)} \\
&\leq |V(CS(E))|^{\mathcal{O}(d^2 \log n)} \qquad \blacksquare
\end{aligned}
$$

Theorem 4.2: Let $E \subseteq \{0, 1, \ldots, d\}^n$. There is an absolute constant C such that $|V(CS(E))|^{Cd^2 \log n} \geq |E|$

Applying the theorem to $E = \text{Supp}(f)$, gives $|V(P_f)|^{\mathcal{O}(d^2 \log n)} \geq ||f||$

References

- V. Bhargava, S. Saraf, I. Volkovich. Deterministic Factorization of Sparse Polynomials with Bounded Individual Degree.
- A. Nowicki. Convex polytopes and Newton polytopes. Teaching Materials, 2016. (web)

Deterministic Factorization of Sparse Polynomials with Bounded Individual Degree

Part 2: Factorization using Sparsity bound

Mohith Raju Nagaraju

March 2022

Main results

Factor sparsity bound

Theorem 1. Let $\mathbb{F}$ be an arbitrary field (finite or otherwise) and let $f(\underline{\boldsymbol{x}}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial of sparsity s and individual degrees at most d, then the sparsity of every factor of $f(\underline{\boldsymbol{x}})$ is bounded by $s^{\mathcal{O}(d^2 \log n)}$.

Factorization sparse polynomials of bounded individual degree

Theorem 2. There exists a deterministic algorithm that given a polynomial $f(\underline{\boldsymbol{x}}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of sparsity s and individual degrees at most d, computes the complete factorization of $f(\underline{\boldsymbol{x}})$, using $s^{\mathcal{O}(d^7 \log n)} \cdot \mathrm{poly}(c_{\mathbb{F}}(d^2))$ field operations.

Where $c_{\mathbb{F}}(d)$ denotes the time of the best known algorithm that factors a univariate polynomial of degree d over $\mathbb{F}$.
(Think $c_{\mathbb{F}} = \mathrm{poly}(d)$)

Main results

Factor sparsity bound

Theorem 1. Let $\mathbb{F}$ be an arbitrary field (finite or otherwise) and let $f(\underline{\boldsymbol{x}}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial of sparsity s and individual degrees at most d, then the sparsity of every factor of $f(\underline{\boldsymbol{x}})$ is bounded by $s^{\mathcal{O}(d^2 \log n)}$.

Factorization sparse polynomials of bounded individual degree

Theorem 2. There exists a deterministic algorithm that given a polynomial $f(\underline{\boldsymbol{x}}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of sparsity s and individual degrees at most d, computes the complete factorization of $f(\underline{\boldsymbol{x}})$, using $s^{\mathcal{O}(d^7 \log n)} \cdot \mathrm{poly}(c_{\mathbb{F}}(d^2))$ field operations.

Introduction

- Given $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ the task is to factor f into irreducibles
- First we solve the problem assuming f is monic, i.e. $f \in \mathbb{F}[y, x_1, x_2, \ldots, x_n]$ is monic in y and of sparsity s and individual degree bounded by d
- Obs 1: f has at most d factors
  Proof: let $f(y, \underline{x}) = h_1(y, \underline{x}) \cdots h_m(y, \underline{x})$. Now WLOG h_i is monic in y. This gives $m \leq \deg_y(f) \leq d$

Broad idea of factorization algorithms

- Project $f(y, \underline{x})$ down to a polynomial in two variables $f(y, g(t))$ where t is a new variable and g is some function
- Factor $f(y, g(t)) = \tilde{h_1}(y, t) \cdots \tilde{h_m}(y, t)$ using a bivariate factoring algorithm
- Lift the factorization to $f(y, \underline{x})$

Introduction

Idea of our factorization algorithm

Step 1. Construct a black-box/oracle access to the factors of $f(y, \underline{x})$, i.e. given $\underline{b} \in \mathbb{F}^n$ we return $h_1(y, \underline{b}), h_2(y, \underline{b}), \ldots, h_m(y, \underline{b})$ where h_i are the irreducible factors of f

- Calculate n, d, s and construct a good set $S_{n,d,s}$
- Pick $\underline{a} \in S_{n,d,s}$ and substitute $\underline{x} = (1-t)\underline{a} + t\underline{b} = l_{\underline{a},\underline{b}}(t)$
- Project $f(y, \underline{x})$ to a bivariate polynomial $f(y, l_{\underline{a},\underline{b}}(t)) \in \mathbb{F}[y, t]$
- Factor f into irreducibles $f(y, l_{\underline{a},\underline{b}}(t)) = f_1^{v_1}(y, t) f_2^{v_2}(y, t) \cdots f_{r'}^{v'_r}(y, t)$ by using a bivariate factoring algorithm

Idea of our factorization algorithm

Step 1. Construct a black-box/oracle access to the factors of $f(y, \underline{x})$, i.e. given $\underline{b} \in \mathbb{F}^n$ we return $h_1(y, \underline{b}), h_2(y, \underline{b}), \ldots, h_m(y, \underline{b})$ where h_i are the irreducible factors of f

- Calculate n, d, s and construct a good set $S_{n,d,s}$
- Pick $\underline{a} \in S_{n,d,s}$ and substitute $\underline{x} = (1-t)\underline{a} + t\underline{b} = l_{\underline{a},\underline{b}}(t)$
- Project $f(y, \underline{x})$ to a bivariate polynomial $f(y, l_{\underline{a},\underline{b}}(t)) \in \mathbb{F}[y, t]$
- Factor f into irreducibles $f(y, l_{\underline{a},\underline{b}}(t)) = f_1^{\nu_1}(y,t) f_2^{\nu_2}(y,t) \cdots f_{r'}^{\nu'_r}(y,t)$ by using a bivariate factoring algorithm
- Guess $h_1(y, l_{\underline{a},\underline{b}}(t)) = f_{i_1}^{\alpha_1}(y,t) \cdot f_{i_2}^{\alpha_2}(y,t) \cdots$ and $h_2(y, l_{\underline{a},\underline{b}}(t)) = f_{j_1}^{\beta_1}(y,t) \cdot f_{j_2}^{\beta_2}(y,t) \cdots$ and so on
- Return $h_1(y, \underline{b}) = f_{i_1}^{\alpha_1}(y,1) \cdot f_{i_2}^{\alpha_2}(y,1) \cdots$ and $h_2(y, l_{\underline{a},\underline{b}}(t)) = f_{j_1}^{\beta_1}(y,1) \cdot f_{j_2}^{\beta_2}(y,1) \cdots$ and so on

Idea of our factorization algorithm

Step 2.

- Use a reconstruction algorithm construct the monomial representations of $h_1(y, \underline{\boldsymbol{x}}), h_2(y, \underline{\boldsymbol{x}}), \ldots, h_m(y, \underline{\boldsymbol{x}})$ from the black-box/oracle representations of $h_1, h_2, \ldots, h_m$
- Check if $f(y, \underline{\boldsymbol{x}}) = h_1(y, \underline{\boldsymbol{x}}) \cdot h_2(y, \underline{\boldsymbol{x}}) \cdots h_m(y, \underline{\boldsymbol{x}})$ by multiplication
- Check if $h_1, h_2, \ldots, h_m$ are irreducibles by using a certain criterion

Good set $S_{n,d,s}$

Def (good set $S_{n,d,s}$):
Given n, d, s, a good set $S_{n,d,s}$ is such that given any monic polynomial $f(y, \underline{x}) \in \mathbb{F}[y, x_1, x_2, \ldots, x_n]$ of sparsity s and individual degree bounded by d, there is an $\underline{a} \in S_{n,d,s}$ such that

$$\gcd_y(h_i(y, \underline{a}), h_j(y, \underline{a})) = 1 \quad \forall i \neq j$$

where $f(y, \underline{x}) = h_1^{e_1}(y, \underline{x}) \cdot h_2^{e_2}(y, \underline{x}) \cdots h_m^{e_m}(y, \underline{x})$ is the complete factorization of f.

- i.e. there is an element $\underline{a}$ in $S_{n,d,s}$ such that the irreducible factors $h_i(y, \underline{x})$ will remain coprime even after setting $\underline{x} = \underline{a}$

Results we shall use without proof

Lemma 5.2: Existence of "small" good set

There is a deterministic algorithm that, given n, d, s, computes a good set $S_{n,d,s}$ of size $|S_{n,d,s}| = (n \cdot \xi(n,d,s))^{\mathcal{O}(d)}$ in time at most polynomial in size of the good set.

Lemma 2.11: Bivariate and univariate factoring [2]

There exists a deterministic algorithm that given a r-variate, degree d polynomial f over $\mathbb{F}$ outputs its irreducible factors. The runtime of the algorithm is $(c_{\mathbb{F}}(d))^{\mathcal{O}(r)}$.

Lemma 2.5: Reconstruction from black-box access of sparse polynomial [3]

Let $n, s, d \in \mathbb{N}$. There exists a deterministic algorithm that given n, s, d and an oracle access to an s-sparse polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ of degree d, uses $\text{poly}(n, s, d, \log |\mathbb{F}|)$ field operations and outputs f (in its monomial representation).

Overview

The factorization algorithm for sparse monic polynomials of bounded individual degree is constructed in two phases.

Phase 1: We construct algorithm 1 which takes as input $f(y, \underline{\boldsymbol{x}})$ and additional information (advice) about the factorization pattern of f and outpus (in some sense) black-box access to the factors of f.

Phase 2: We construct algorithm 2 which takes as input only $f(y, \underline{\boldsymbol{x}})$ and tries out different possibilities for the value of the advice and runs algorithm 1 on them. While trying out different possibilities it will hit the correct advice and hence obtain the correct factorization of f.

Remark: As algorithm 2 has to try out different possible values for the advice, it is only efficient if f is a sparse polynomial of bounded individual degree.

Algo 1: Black-box Factoring of Monic Polynomials (given some advice)

Algorithm 1

Input

- $f(y, \underline{\boldsymbol{x}}) \in \mathbb{F}[y, x_1, \ldots, x_n]$ monic in y
- Point $\underline{b} \in \mathbb{F}^n$
- Exponent vector $\underline{\boldsymbol{e}} = (e_1, e_2, \ldots, e_m) \in [d]^m$
- Point $\underline{a} \in \mathbb{F}^n$
- Univariate polynomials $g_1(y), g_2(y), \ldots, g_r(y)$
- Partition $A_1 \sqcup A_2 \sqcup \cdots \sqcup A_m = [r]$

Output

- $h_1(y, \underline{b}), h_2(y, \underline{b}), \ldots, h_m(y, \underline{b})$

Time

- $\text{poly}(n, c_{\mathbb{F}}(d))$ field operations and steps

Algo 1: Black-box Factoring of Monic Polynomials (given some advice)

Algorithm 1

Input

- $f(y, \underline{\boldsymbol{x}}) \in \mathbb{F}[y, x_1, \ldots, x_n]$ monic in y
  where $f(y, \underline{\boldsymbol{x}}) = h_1(y, \underline{\boldsymbol{x}})^{e_1} \cdot h_2(y, \underline{\boldsymbol{x}})^{e_2} \cdots h_m(y, \underline{\boldsymbol{x}})^{e_m}$
- Point $\underline{b} \in \mathbb{F}^n$
- Exponent vector $\underline{\boldsymbol{e}} = (e_1, e_2, \ldots, e_m) \in [d]^m$
  where e_i is the multiplicity of irreducible factor h_i in f
- Point $\underline{a} \in \mathbb{F}^n$
  where $\underline{a}$ is such that $\gcd_y(h_i(y, \underline{a}), h_j(y, \underline{a})) = 1 \quad \forall i \neq j$
- Univariate polynomials $g_1(y), g_2(y), \ldots, g_r(y)$
  where $g_1(y), g_2(y), \ldots, g_r(y)$ are (not necessarily distinct) irreducible factors of $f(y, \underline{a})$, i.e.
  $f(y, \underline{a}) = g_1(y) g_2(y) \cdots g_r(y)$
- Partition $A_1 \sqcup A_2 \sqcup \cdots \sqcup A_m = [r]$
  where $h_i(y, \underline{a})^{e_i} = \prod_{j \in A_i} g_j(y)$

Algo 1: Black-box Factoring of Monic Polynomials (given some advice)

Developing algorithm 1

- Substitute $\underline{\boldsymbol{x}} = (1-t)\underline{a} + t\underline{b} = l_{\underline{a},\underline{b}}(t)$
- $\tilde{f}(y,t) := f(y, l_{\underline{a},\underline{b}}(t))$
- Factorize $\tilde{f}(y,t) = f_1^{v_1}(y,t) \cdot f_2^{v_2}(y,t) \cdots f_{r'}^{v_{r'}}(y,t)$

Algo 1: Black-box Factoring of Monic Polynomials (given some advice)

Developing algorithm 1

- Substitute $\underline{x} = (1-t)\underline{a} + t\underline{b} = l_{\underline{a},\underline{b}}(t)$
- $\tilde{f}(y,t) := f(y, l_{\underline{a},\underline{b}}(t))$
- Factorize $\tilde{f}(y,t) = f_1^{v_1}(y,t) \cdot f_2^{v_2}(y,t) \cdots f_{r'}^{v_{r'}}(y,t)$
- If we knew which $f_k(y,t)$-s divide $h_1(y, l_{\underline{a},\underline{b}}(t))$ and the multiplicity of $f_k(y,t)$ dividing $h_1(y, l_{\underline{a},\underline{b}}(t))$ we would be able to compute $h_1(y, l_{\underline{a},\underline{b}}(t))$

Algo 1: Black-box Factoring of Monic Polynomials (given some advice)

Developing algorithm 1

Finding the $f_k(y,t)$-s which divide $h_1(y, l_{\underline{a},\underline{b}}(t))$

- Observe

$$f(y,\underline{a}) = \prod_{i=1}^{m} h_i(y,\underline{a})^{e_i} = \prod_{i=1}^{m} \left(\prod_{j \in A_i} g_j(y) \right)$$

$$\tilde{f}(y,0) = f_1^{v_1}(y,0) \cdot f_2^{v_2}(y,0) \cdots f_{r'}^{v_{r'}}(y,0)$$

- Let $j \in A_1$ be such that $g_j(y)$ divides $f_k(y,0)$
- Then, $g_j(y)$ divides both $f_k(y,0)$ and $h_1(y,\underline{a})$
- We claim $f_k(y,t)$ divides $h_1(y, l_{\underline{a},\underline{b}}(t))$

Algo 1: Black-box Factoring of Monic Polynomials (given some advice)

Developing algorithm 1

Finding the $f_k(y,t)$-s which divide $h_1(y, l_{\underline{a},\underline{b}}(t))$

- Observe

$$f(y,\underline{a}) = \prod_{i=1}^{m} h_i(y,\underline{a})^{e_i} = \prod_{i=1}^{m} \left(\prod_{j \in A_i} g_j(y) \right)$$

$$\tilde{f}(y,0) = f_1^{v_1}(y,0) \cdot f_2^{v_2}(y,0) \cdots f_{r'}^{v_{r'}}(y,0)$$

- Let $j \in A_1$ be such that $g_j(y)$ divides $f_k(y,0)$
- Then, $g_j(y)$ divides both $f_k(y,0)$ and $h_1(y,\underline{a})$
- We claim $f_k(y,t)$ divides $h_1(y, l_{\underline{a},\underline{b}}(t))$
  - We know $f_k(y,t)$ divides some $h_w(y, l_{\underline{a},\underline{b}}(t))$ as $f_k(y,t)$ is irreducible
  - $\Rightarrow f_k(y,0)$ divides $h_w(y,\underline{a})$
  - As $g_j(y) \big| f_k(y,0)$, $g_j(y)$ divides both $h_1(y,\underline{a})$ and $h_w(y,\underline{a})$
  - $\Rightarrow \gcd_y(h_1(y,\underline{a}), h_w(y,\underline{a})) \neq 1$
  - $\Rightarrow 1 = w$ and $f_k(y,t)$ divides $h_1(y, l_{\underline{a},\underline{b}}(t))$

Algo 1: Black-box Factoring of Monic Polynomials (given some advice)

Developing algorithm 1

Finding the $f_k(y,t)$-s which divide $h_1(y, l_{\underline{a},\underline{b}}(t))$

- Observe

$$f(y,\underline{a}) = \prod_{i=1}^{m} h_i(y,\underline{a})^{e_i} = \prod_{i=1}^{m} \left(\prod_{j \in A_i} g_j(y) \right)$$

$$\tilde{f}(y,0) = f_1^{v_1}(y,0) \cdot f_2^{v_2}(y,0) \cdots f_{r'}^{v_{r'}}(y,0)$$

- Let $j \in A_1$ be such that $g_j(y)$ divides $f_k(y,0)$
- Then, $g_j(y)$ divides both $f_k(y,0)$ and $h_1(y,\underline{a})$
- We claim $f_k(y,t)$ divides $h_1(y, l_{\underline{a},\underline{b}}(t))$
- Takeaway: $f_k(y,t)$ divides $h_1(y, l_{\underline{a},\underline{b}}(t))$ iff there is a $j \in A_1$ such that $g_j(y)$ divides $f_k(y,0)$

Algo 1: Black-box Factoring of Monic Polynomials (given some advice)

Developing algorithm 1

Finding the multiplicity of $f_k(y,t)$ dividing $h_1(y, l_{\underline{a},\underline{b}}(t))$

- Let $f_k(y,t)$ divide $h_1(y, l_{\underline{a},\underline{b}}(t))$
- Assume $f_k(y,t)$ divides $h_1(y, l_{\underline{a},\underline{b}}(t))$ and $h_2(y, l_{\underline{a},\underline{b}}(t))$
- $\Rightarrow f_k(y,0)$ divides $h_1(y,\underline{a})$ and $h_2(y,\underline{a})$
- This contradicts the property of $\underline{a}$ which says $\gcd_y(h_{i_1}(y,\underline{a}), h_{i_2}(y,\underline{a})) = 1$ when $i_1 \neq i_2$
- Thus, $f_k(y,t)$ only divides $h_1(y, l_{\underline{a},\underline{b}}(t))$
- $\Rightarrow f_k^{v_k}$ divides $h_1^{e_1}$
- $\Rightarrow f_k^{v_k/e_1}$ divides h_1

Algo 1: Black-box Factoring of Monic Polynomials (given some advice)

Developing algorithm 1

Finding the multiplicity of $f_k(y,t)$ dividing $h_1(y, l_{\underline{a},\underline{b}}(t))$

- Let $f_k(y,t)$ divide $h_1(y, l_{\underline{a},\underline{b}}(t))$
- Assume $f_k(y,t)$ divides $h_1(y, l_{\underline{a},\underline{b}}(t))$ and $h_2(y, l_{\underline{a},\underline{b}}(t))$
- $\Rightarrow f_k(y,0)$ divides $h_1(y,\underline{a})$ and $h_2(y,\underline{a})$
- This contradicts the property of $\underline{a}$ which says $\gcd_y(h_{i_1}(y,\underline{a}), h_{i_2}(y,\underline{a})) = 1$ when $i_1 \neq i_2$
- Thus, $f_k(y,t)$ only divides $h_1(y, l_{\underline{a},\underline{b}}(t))$
- $\Rightarrow f_k^{v_k}$ divides $h_1^{e_1}$
- $\Rightarrow f_k^{v_k/e_1}$ divides h_1
- Takeaway: If f_k divides h_1, then $f_k^{v_k/e_1}$ divides h_1

Developing algorithm 1

Summary

- Substitute $\underline{x} = (1-t)\underline{a} + t\underline{b} = l_{\underline{a},\underline{b}}(t)$
- $\tilde{f}(y,t) \leftarrow f(y, l_{\underline{a},\underline{b}}(t))$
- Factorize $\tilde{f}(y,t) = f_1^{v_1}(y,t) \cdot f_2^{v_2}(y,t) \cdots f_{r'}^{v_{r'}}(y,t)$
- Hence,

$$f_1^{v_1}(y,t) \cdot f_2^{v_2}(y,t) \cdots f_{r'}^{v_{r'}}(y,t) = h_1(y, l_{\underline{a},\underline{b}}(t))^{e_1} h_2(y, l_{\underline{a},\underline{b}}(t))^{e_2} \cdots h_m(y, l_{\underline{a},\underline{b}}(t))^{e_m}$$

- So $h_1(y, l_{\underline{a},\underline{b}}(t)) = f_{i_1}^{\alpha_1}(y,t) \cdot f_{i_2}^{\alpha_2}(y,t) \cdots$
- We find f_i-s and α-s as follows:
- If there is a $j \in A_1$ such that $g_j(y)$ divides $f_k(y,0)$, then $f_k(y,t)$ divides $h_1(y, l_{\underline{a},\underline{b}}(t))$
- If f_k divides h_1, then $f_k^{v_k/e_1}$ divides h_1

Algo 1: Black-box Factoring of Monic Polynomials (given some advice)

Input: s-sparse monic (in y) polynomial $f \in \mathbb{F}[y, x_1, x_2, \ldots, x_n]$ of individual degree at most d.
Assignments: $\overline{a}, \overline{b} \in \mathbb{F}^n$
Univariate Polynomials: $g_1(y), g_2(y), \ldots, g_r(y)$
Partition: $A_1 \dot{\bigcup} A_2 \dot{\bigcup} \cdots \dot{\bigcup} A_m = [r]$
Exponent Vector: $\overline{e} = (e_1, e_2, \ldots, e_m) \in [d]^m$
Output: Univariate Polynomials: $\varphi_1(y), \varphi_2(y), \ldots, \varphi_m(y)$

$\tilde{f}(y,t) \leftarrow f(y, \ell_{\overline{a},\overline{b}}(t))$;
Compute the bi-variate factorization of $\tilde{f}(y,t) = f_1^{v_1}(y,t) \cdot f_2^{v_2}(y,t) \cdots f_{r'}^{v_{r'}}(y,t)$; `/* wlog the polynomials are monic in` y `*/`
for $i \leftarrow 1$ **to** m **do**
    $\tilde{h}_i(y,t) \leftarrow 1$;
    for $k \leftarrow 1$ **to** r' **do**
        if *there exists* $j \in A_i$ *s.t* $g_j(y) \mid f_k(y,0)$ **then** $\tilde{h}_i(y,t) \leftarrow \tilde{h}_i(y,t) \cdot f_k^{v_k/e_i}(y,t)$;
    end
end
return $\tilde{h}_1(y,1), \tilde{h}_2(y,1), \ldots, \tilde{h}_m(y,1)$;

Algorithm 1: Black-Box Evaluation of Factors

Algo 1: Black-box Factoring of Monic Polynomials (given some advice)

Time Complexity of algorithm 1

Input: s-sparse monic (in y) polynomial $f \in \mathbb{F}[y, x_1, x_2, \ldots, x_n]$ of individual degree at most d.
Assignments: $\overline{a}, \overline{b} \in \mathbb{F}^n$
Univariate Polynomials: $g_1(y), g_2(y), \ldots, g_r(y)$
Partition: $A_1 \dot{\bigcup} A_2 \dot{\bigcup} \cdots \dot{\bigcup} A_m = [r]$
Exponent Vector: $\overline{e} = (e_1, e_2, \ldots, e_m) \in [d]^m$
Output: Univariate Polynomials: $\varphi_1(y), \varphi_2(y), \ldots, \varphi_m(y)$

$(C_{\mathbb{F}}(d))^2$

1 $\tilde{f}(y,t) \leftarrow f(y, \ell_{\overline{a},\overline{b}}(t))$;
2 Compute the bi-variate factorization of $\tilde{f}(y,t) = f_1^{v_1}(y,t) \cdot f_2^{v_2}(y,t) \cdots f_{r'}^{v_{r'}}(y,t)$; /* wlog the polynomials are monic in y */
3 **for** $i \leftarrow 1$ **to** m **do**
4 $\quad \tilde{h}_i(y,t) \leftarrow 1$;
5 $\quad$ **for** $k \leftarrow 1$ **to** r' **do**
6 $\quad\quad$ **if** *there exists* $j \in A_i$ *s.t* $g_j(y) \mid f_k(y,0)$ **then** $\tilde{h}_i(y,t) \leftarrow \tilde{h}_i(y,t) \cdot f_k^{v_k/e_i}(y,t)$;
7 $\quad$ **end**
8 **end**
9 **return** $\tilde{h}_1(y,1), \tilde{h}_2(y,1), \ldots, \tilde{h}_m(y,1)$;

poly(n,d) as $m, r' \leq d$

Total : poly$(n, C_{\mathbb{F}}(d))$

Algorithm 1: Black-Box Evaluation of Factors

Algorithm 2: Factoring Sparse Monic Polynomials (without advice)

Algorithm 2

Input

- $f(y, \underline{x}) \in \mathbb{F}[y, x_1, \ldots, x_n]$ monic in y of sparsity s and individual degrees bounded by d

Output

- $h_1(y, \underline{x}), h_2(y, \underline{x}), \ldots, h_m(y, \underline{x})$ and $(e_1, e_2, \ldots, e_m)$ such that $f(y, \underline{x}) = \prod h_i^{e_i}(y, \underline{x})$ is the complete factorization of f

Time

- $(n \cdot \xi(n, d, s))^{\mathcal{O}(d)} \cdot \text{poly}(c_{\mathbb{F}}(d))$ field operations and steps

Algorithm 2: Factoring Sparse Monic Polynomials (without advice)

Algorithm 2

Input

- $f(y, \underline{x}) \in \mathbb{F}[y, x_1, \ldots, x_n]$ monic in y of sparsity s and individual degrees bounded by d

Output

- $h_1(y, \underline{x}), h_2(y, \underline{x}), \ldots, h_m(y, \underline{x})$ and $(e_1, e_2, \ldots, e_m)$ such that $f(y, \underline{x}) = \prod h_i^{e_i}(y, \underline{x})$ is the complete factorization of f

Time

- $n^{\mathcal{O}(d)} \cdot s^{\mathcal{O}(d^3 \log n)} \cdot \text{poly}(c_{\mathbb{F}}(d))$ field operations and steps

Algorithm 2: Factoring Sparse Monic Polynomials (without advice)

Developing algorithm 2

- For each $m \in [d]$
- For each vector $\underline{\boldsymbol{e}} = (e_1, e_2, \ldots, e_m)$ in $[d]^m$,
- For each point $\underline{a}$ in the good set $S_{n,d,s}$
- Factorize $f(y, \underline{a})$ into $f(y, \underline{a}) = g_1(y)g_2(y)\cdots g_r(y)$
- For each partition $A_1 \sqcup A_2 \sqcup \cdots \sqcup A_m = [r]$
- Call algorithm 1 using above values and get blackbox access to the irreducible factors
- Use a reconstruction algorithm to reconstruct the actual factors
- Check whether the obtained factorization is the correct complete factorization

Algorithm 2: Factoring Sparse Monic Polynomials (without advice)

Tool for checking whether a given factorization is the correct complete factorization

Lemma 2.4: Consider the function $\Phi : \mathbb{N}^* \to \mathbb{N}$ given $\underline{\boldsymbol{e}} = (e_1, \ldots, e_m)$, $\Phi(\underline{\boldsymbol{e}}) = 2 \cdot \sum_{i=1}^{m} e_i - m$.

Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial and let $f = h_1^{e_1} \cdots h_m^{e_m}$ a factorization of f (not necessarily irreducible or coprime), where h_i-s are non-constant and $e_i \geq 1$. Then all irreducible, pairwise coprime factorizations of f correspond to those that maximize $\Phi(\underline{\boldsymbol{e}})$.

Algorithm 2: Factoring Sparse Monic Polynomials (without advice)

Lemma 2.4: Consider the function $\Phi : \mathbb{N}^* \to \mathbb{N}$ given $\underline{e} = (e_1, \ldots, e_m)$, $\Phi(\underline{e}) = 2 \cdot \sum_{i=1}^{m} e_i - m$.

Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial and let $f = h_1^{e_1} \cdots h_m^{e_m}$ a factorization of f (not necessarily irreducible or coprime), where h_i-s are non-constant and $e_i \geq 1$. Then all irreducible, pairwise coprime factorizations of f correspond to those that maximize $\Phi(\underline{e})$.

Proof:

- Let $f = h_1^{e_1} \cdots h_{m-1}^{e_m} \cdot h_m^{e_m}$ be such that h_m is not irreducible, i.e. $h_m = u \cdot v$
- Write $f = h_1^{e_1} \cdots h_{m-1}^{e_m} \cdot u^{e_m} \cdot v^{e_m}$
- For this factorization the exponent vector is $\underline{e}_{new} = (e_1, \ldots, e_{m-1}, e_m, e_m)$
- Now $\Phi(\underline{e}_{new}) = \Phi(\underline{e}) + 2e_m - 1$
- $\Rightarrow \Phi(\underline{e}_{new}) > \Phi(\underline{e})$

Algorithm 2: Factoring Sparse Monic Polynomials (without advice)

Lemma 2.4: Consider the function $\Phi : \mathbb{N}^* \to \mathbb{N}$ given $\underline{e} = (e_1, \ldots, e_m)$, $\Phi(\underline{e}) = 2 \cdot \sum_{i=1}^{m} e_i - m$.

Let $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be a polynomial and let $f = h_1^{e_1} \cdots h_m^{e_m}$ a factorization of f (not necessarily irreducible or coprime), where h_i-s are non-constant and $e_i \geq 1$. Then all irreducible, pairwise coprime factorizations of f correspond to those that maximize $\Phi(\underline{e})$.

Proof:

- Let $f = h_1^{e_1} \cdots h_{m-1}^{e_m} \cdot h_m^{e_m}$ be such that h_i-s are irreducible
- Assume h_{m-1} and h_m are not coprime
- $\Rightarrow h_{m-1} = \alpha h_m$
- Write $f = h_1^{e_1} \cdots h_{m-1}^{e_{m-1}+e_m}$
- For this factorization the exponent vector is $\underline{e}_{new} = (e_1, \ldots, e_{m-2}, e_{m-1} + e_m)$
- Now $\Phi(\underline{e}_{new}) = \Phi(\underline{e}) + 1$
- $\Rightarrow \Phi(\underline{e}_{new}) > \Phi(\underline{e})$

Algorithm 2: Factoring Sparse Monic Polynomials (without advice)

Description of algorithm 2

1. 1.1 For each $m \in [d]$
   - 1.2 For each vector $\underline{\boldsymbol{e}} = (e_1, e_2, \ldots, e_m)$ in $[d]^m$
   - 1.3 For each point $\underline{a}$ in the good set $S_{n,d,s}$
   - 1.4 Factorize $f(y, \underline{a})$ into $f(y, \underline{a}) = g_1(y)g_2(y)\cdots g_r(y)$
   - 1.5 For each partition $A_1 \sqcup A_2 \sqcup \cdots \sqcup A_m = [r]$
   - 1.6 Call algorithm 1 using above values and get blackbox access to the irreducible factors
   - 1.7 Use a reconstruction algorithm to reconstruct the actual factors $h_1(y, \underline{\boldsymbol{x}}), \ldots, h_m(y, \underline{\boldsymbol{x}})$
   - 1.8 Check whether $f \equiv h_1(y, \underline{\boldsymbol{x}})^{e_1} h_2(y, \underline{\boldsymbol{x}})^{e_2} \cdots h_m(y, \underline{\boldsymbol{x}})^{e_m}$
2. Return a factorization that maximizes the expression $\Phi(\underline{\boldsymbol{e}}) = 2 \cdot \sum_{i=1}^{m} e_i - m$

Algorithm 2: Factoring Sparse Monic Polynomials (without advice)

Correctness of algorithm 2

- As we are iterating over all possible values, we will eventually arrive at the correct values of $\underline{\boldsymbol{e}}, \underline{a}, A_1 \sqcup \cdots \sqcup A_m$
- In this iteration we construct the correct irreducible factors $h_1, \ldots, h_m$ and exponent vector $\underline{\boldsymbol{e}} = (e_1, \ldots, e_m)$
- Observe that this factorization will pass the test $f \equiv h_1^{e_1} h_2^{e_2} \cdots h_m^{e_m}$
- By the previous lemm this factorization will have the maximum value for $\Phi(\underline{\boldsymbol{e}})$
- Thus algorithm 2 returns the correct irreducible factors $h_1, \ldots, h_m$ and exponent vector $\underline{\boldsymbol{e}} = (e_1, \ldots, e_m)$

Algorithm 2: Factoring Sparse Monic Polynomials (without advice)

Time Complexity of algorithm 2

1. 1.1 For each $m \in [d]$ → d choices
   1.2 For each vector $\underline{e} = (e_1, e_2, \ldots, e_m)$ in $[d]^m$ → $d^m \leq d^d$ choices
   1.3 For each point $\underline{a}$ in the good set $S_{n,d,s}$ → $|S_{n,d,s}|$ choices
   1.4 Factorize $f(y, \underline{a})$ into $f(y, \underline{a}) = g_1(y)g_2(y)\cdots g_r(y)$ → $c_{\mathbb{F}}(d)$ steps
   1.5 For each partition $A_1 \sqcup A_2 \sqcup \cdots \sqcup A_m = [r]$ → $m^r \leq d^d$ choices
   1.6 Call algorithm 1 using above values and get blackbox access to the irreducible factors → $\text{poly}(n, c_{\mathbb{F}}(d))$ steps
   1.7 Use a reconstruction algorithm to reconstruct the actual factors $h_1(y, \underline{x}), \ldots, h_m(y, \underline{x})$ → $\text{poly}(n, \xi(n,d,s), d, \log|\mathbb{F}|)$ steps
   1.8 Check whether $f \equiv h_1(y, \underline{x})^{e_1} h_2(y, \underline{x})^{e_2} \cdots h_m(y, \underline{x})^{e_m}$ → each multiplication takes $\xi(n,d,s)^2$ total multiplication takes $\xi(n,d,s)^{O(d)}$
2. Return a factorization that maximizes the expression $\Phi(\underline{e}) = 2 \cdot \sum_{i=1}^{m} e_i - m$

$$\text{TOTAL} = |S_{n,d,s}| \cdot d^{O(d)} \cdot \text{poly}(n, s, c_{\mathbb{F}}(d), \xi(n,d,s)^{O(d)})$$

Algorithm 2: Factoring Sparse Monic Polynomials (without advice)

Time Complexity of algorithm 2

1. 1.1 For each $m \in [d]$ → d choices
   1.2 For each vector $\underline{e} = (e_1, e_2, \ldots, e_m)$ in $[d]^m$ → $d^m \leq d^d$ choices
   1.3 For each point $\underline{a}$ in the good set $S_{n,d,s}$ → $|S_{n,d,s}|$ choices
   1.4 Factorize $f(y, \underline{a})$ into $f(y, \underline{a}) = g_1(y)g_2(y)\cdots g_r(y)$ → $c_{\mathbb{F}}(d)$ steps
   1.5 For each partition $A_1 \sqcup A_2 \sqcup \cdots \sqcup A_m = [r]$ → $m^r \leq d^d$ choices
   1.6 Call algorithm 1 using above values and get blackbox access to the irreducible factors → $\text{poly}(n, c_{\mathbb{F}}(d))$ steps
   1.7 Use a reconstruction algorithm to reconstruct the actual factors $h_1(y, \underline{x}), \ldots, h_m(y, \underline{x})$ → $\text{poly}(n, \xi(n,d,s), d, \log|\mathbb{F}|)$ steps
   1.8 Check whether $f \equiv h_1(y, \underline{x})^{e_1} h_2(y, \underline{x})^{e_2} \cdots h_m(y, \underline{x})^{e_m}$ → each multiplication takes $\xi(n,d,s)^2$ total multiplication takes $\xi(n,d,s)^{O(d)}$
2. Return a factorization that maximizes the expression $\Phi(\underline{e}) = 2 \cdot \sum_{i=1}^{m} e_i - m$

TOTAL = $n^{O(d)}\, \xi(n,s,d)^{O(d)}\, \text{poly}(c_{\mathbb{F}}(d))$

Constructing a good set $S_{n,d,s}$

Lemma 2.6: [3]
There exists a deterministic algorithm that given $n, s, d, k \in \mathbb{N}$ outputs a set $SP_{(n,s,d,k)}$ of size poly(n, s, d, k) such that any set of (at most) k non-zero s-sparse polynomials $f_1, \ldots, f_k \in \mathbb{F}[x_1, \ldots, x_n]$ with individual degrees at most d have a common non-zero in $SP_{(n,s,d,k)}$. In other words, there exists $\underline{a} \in SP_{(n,s,d,k)}$ such that $f_i(\underline{a}) \neq 0 \quad \forall i \in [k]$

Constructing a good set $S_{n,d,s}$

Lemma 2.6: [3]
There exists a deterministic algorithm that given $n, s, d, k \in \mathbb{N}$ outputs a set $SP_{(n,s,d,k)}$ of size poly(n, s, d, k) such that any set of (at most) k non-zero s-sparse polynomials $f_1, \ldots, f_k \in \mathbb{F}[x_1, \ldots, x_n]$ with individual degrees at most d have a common non-zero in $SP_{(n,s,d,k)}$. In other words, there exists $\underline{a} \in SP_{(n,s,d,k)}$ such that $f_i(\underline{a}) \neq 0 \quad \forall i \in [k]$

Can we use S-Z lemma to prove this?

Let $f_1, \ldots, f_k$ be given. Note they have total degree $\leq nd$.
Consider $g(\underline{\boldsymbol{x}}) = f_1(\underline{\boldsymbol{x}}) \cdots f_k(\underline{\boldsymbol{x}})$ of total degree ndk.

S-Z lemma gives,

$$\Pr_{\underline{\boldsymbol{x}} \in_r S^n} [g(\underline{\boldsymbol{x}}) \neq 0] > 1 - \frac{ndk}{|S|}$$

Take $|S| > ndk$.

But this tells $|S^n| > (ndk)^n$

Constructing a good set $S_{n,d,s}$

Lemma 2.8: (Resultant Properties). Let $f, g \in \mathbb{F}[y, x_1, x_2, ..., x_n]$ be monic in y, s-sparse polynomial with individual degrees at most d. Then:

1. $\gcd(f, g) \neq 1$ iff $\text{Res}_y(f, g) \equiv 0$
2. $\text{Res}_y(f, g)(\underline{\boldsymbol{x}})$ is an $(2ds)^{2d}$-sparse polynomial over $\mathbb{F}[x_1, x_2, ..., x_n]$ with individual degrees at most $2d^2$

Constructing a good set $S_{n,d,s}$

Lemma 2.8: (Resultant Properties). Let $f, g \in \mathbb{F}[y, x_1, x_2, ..., x_n]$ be monic in y, s-sparse polynomial with individual degrees at most d. Then:

1. $\gcd(f, g) \neq 1$ iff $\text{Res}_y(f, g) \equiv 0$
2. $\text{Res}_y(f, g)(\underline{\boldsymbol{x}})$ is an $(2ds)^{2d}$-sparse polynomial over $\mathbb{F}[x_1, x_2, ..., x_n]$ with individual degrees at most $2d^2$

Lemma 5.2: Existence of "small" good set

There is a deterministic algorithm that, given n, d, s, computes a good set $S_{n,d,s}$ of size $|S_{n,d,s}| = (n \cdot \xi(n, d, s))^{\mathcal{O}(d)}$ in time at most polynomial in size of the good set.

The good set is such that given any polynomial $f(y, \underline{\boldsymbol{x}}) \in \mathbb{F}[y, x_1, x_2, \ldots, x_n]$ of sparsity s and individual degree bounded by d, there is an $\underline{a} \in S_{n,d,s}$ satsifying

$$\gcd_y(h_i(y, \underline{a}), h_j(y, \underline{a})) = 1 \quad \forall i \neq j$$

Constructing a good set $S_{n,d,s}$

Lemma 2.8: (Resultant Properties). Let $f, g \in \mathbb{F}[y, x_1, x_2, ..., x_n]$ be monic in y, s-sparse polynomial with individual degrees at most d. Then:

1. $\gcd(f, g) \neq 1$ iff $\text{Res}_y(f, g) \equiv 0$
2. $\text{Res}_y(f, g)(\underline{\boldsymbol{x}})$ is an $(2ds)^{2d}$-sparse polynomial over $\mathbb{F}[x_1, x_2, ..., x_n]$ with individual degrees at most $2d^2$

Lemma 5.2: Existence of "small" good set

There is a deterministic algorithm that, given n, d, s, computes a good set $S_{n,d,s}$ of size $|S_{n,d,s}| = (n \cdot \xi(n, d, s))^{\mathcal{O}(d)}$ in time at most polynomial in size of the good set.

The good set is such that given any polynomial $f(y, \underline{\boldsymbol{x}}) \in \mathbb{F}[y, x_1, x_2, \ldots, x_n]$ of sparsity s and individual degree bounded by d, there is an $\underline{a} \in S_{n,d,s}$ satsifying

$$\text{Res}_y(h_i, h_j)(\underline{a}) \neq 0 \quad \forall i \neq j$$

Constructing a good set $S_{n,d,s}$

Lemma 5.2: Existence of "small" good set

There is a deterministic algorithm that, given n, d, s, computes a good set $S_{n,d,s}$ of size $|S_{n,d,s}| = (n \cdot \xi(n,d,s))^{\mathcal{O}(d)}$ in time at most polynomial in size of the good set.

The good set is such that given any polynomial $f(y, \underline{\boldsymbol{x}}) \in \mathbb{F}[y, x_1, x_2, \ldots, x_n]$ of sparsity s and individual degree bounded by d, there is an $\underline{a} \in S_{n,d,s}$ satsifying

$$\mathrm{Res}_y(h_i, h_j)(\underline{a}) \neq 0 \quad \forall i \neq j$$

Proof:

- Consider the set of polynomials $\mathrm{Res}_y(h_i, h_j)(\underline{\boldsymbol{x}})$ for all $1 \leq i < j \leq d$
- From Lemma 2.6 get $SP_{(n,(2d\cdot\xi(n,d,s))^{2d},2d^2,d^2)}$ of size $(d \cdot \xi(n,d,s))^{\mathcal{O}(d)}$
- By lemma 2.6, the above set will contain an $\underline{a}$ such that $\mathrm{Res}_y(h_i, h_j)(\underline{a}) \neq 0$ for all $1 \leq i < j \leq d$ $\square$

Algorithm 3

Input

- $f(\underline{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of sparsity s and individual degrees bounded by d

Output

- $h_1(\underline{x}), h_2(\underline{x}), \ldots, h_m(\underline{x})$ and $(e_1, e_2, \ldots, e_m)$ such that $f(\underline{x}) = \prod h_i^{e_i}(\underline{x})$ is the complete factorization of f

Time

- $(n \cdot \xi(n, d^2, s^d))^{\mathcal{O}(d^2)} \cdot \text{poly}(c_{\mathbb{F}}(d))$ field operations and steps

Algorithm 3

Input

- $f(\underline{x}) \in \mathbb{F}[x_1, \ldots, x_n]$ of sparsity s and individual degrees bounded by d

Output

- $h_1(\underline{x}), h_2(\underline{x}), \ldots, h_m(\underline{x})$ and $(e_1, e_2, \ldots, e_m)$ such that $f(\underline{x}) = \prod h_i^{e_i}(\underline{x})$ is the complete factorization of f

Time

- $n^{\mathcal{O}(d)} \cdot s^{\mathcal{O}(d^7 \log n)} \cdot \text{poly}(c_{\mathbb{F}}(d))$ field operations and steps

Dropping the assumption of f being monic

Converting f to a monic

Given $f(x_1, \ldots, x_n, x_{n+1})$ consider

$$\hat{f}(x_1, \ldots, x_n) = f(x_1, \ldots, x_n, \frac{y}{f_k}) \cdot f_k^{k-1}$$

where f_k is the leading coefficent of f when viewed as polynomial in x_{n+1}

Dropping the assumption of f being monic

Converting f to a monic

Given $f(x_1, \ldots, x_n, x_{n+1})$ consider

$$\hat{f}(x_1, \ldots, x_n) = f(x_1, \ldots, x_n, \frac{y}{f_k}) \cdot f_k^{k-1}$$

where f_k is the leading coefficent of f when viewed as polynomial in x_{n+1}

Lemma 5.5: Suppose f is an s-sparse polynomial with individual degrees at most d. Then function $\hat{f}$ is an (s^d)-sparse polynomial in $\mathbb{F}[y, x_1, x_2, \ldots, x_n]$ monic in y with individual degrees at most d^2.

Proof.

$$||\hat{f}|| = 1 + \sum_{j=0}^{k-1} ||f_j \cdot f_k^{k-1-j}|| \leq \sum_{j=0}^{k} ||f_j|| \cdot ||f_k||^{k-1}$$

$$\leq \sum_{j=0}^{k} ||f_j|| \cdot s^{k-1} \leq s^{k-1} \cdot ||f||$$

$$\leq s^k$$

Dropping the assumption of f being monic

Questions

1. How are the factors of $\hat{f}$ related the original factors of f ?
2. As the degree of y in $\hat{f}$ is at most d, we can recover at most d factors, while f could potentially have dn factors! How can we recover the remaining factors?

Lemma 5.6 Let $f(\underline{x}, x_{n+a}) = \prod_{i=1}^{m'} h_i(\underline{x}, x_{n+1})^{e_i} \cdot \prod_{l=m'+1}^{m} h_l(\underline{x})^{e_l}$ and $f_k(\underline{x}) = \prod_{j=1}^{r} w_j(\underline{x})^{\beta_j}$ and $\hat{f}(\underline{x}, y) = \prod_{j=1}^{\hat{m}} \hat{h}_j(\underline{x}, y)^{\hat{e}_j}$ then,

1. $\hat{m} = m'$ and there is a permutation $\sigma : [\hat{m}] \to [m']$ s.t. $\hat{h}_i(\underline{x}, f_k x_{n+1}) = h_{\sigma_i}(\underline{x}, x_n + 1) \cdot u_i(x)$ and $\hat{e}_i = e_{\sigma_i}$
2. $m - m' \leq r$ and there is an injective map $\tau : \{m' + 1, \ldots, m\} \to [r]$ s.t $h_l = h_{\tau_l}$
3. Each u_i is a product of w_j-s

Dropping the assumption of f being monic

Using lemma 5.6 to factor f

1. We know $e_i = \hat{e}_i$ for all $1 \leq i \leq \hat{m}$
2. Take $\hat{h}_i(\underline{\boldsymbol{x}}, f_k x_{n+1})$ and divide out all possible w_j-s to get $h_i(\underline{\boldsymbol{x}}, x_{n+1})$
3. By tallying everything we removed, we know what is $\prod_{j=1}^{\hat{m}} u_j(\underline{\boldsymbol{x}})$
4. By looking at 4, we know $h_l(\underline{\boldsymbol{x}})$ and e_l
5. We thus know $f(\underline{\boldsymbol{x}}, x_{n+1}) = \prod_{i=1}^{\hat{m}} h_i(\underline{\boldsymbol{x}}, x_{n+1})^{\hat{e}_i} \cdot \prod_{j=1}^{r} w_j^{\alpha_j}$

Input: s-sparse polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_{n+1}]$ with individual degrees at most d.
Output: irreducible factors $h_1, h_2, \ldots, h_m$, and $e_1, e_2, \ldots, e_m$ such that $f = h_1^{e_1} \cdots h_m^{e_m}$

1. **if** $n \leq 1$ **then** Return the bi-variate factorization of f;
2. $k = \deg_{x_{n+1}}(f)$; $f_k \leftarrow \mathrm{lc}_{x_{n+1}}(f)$;
3. Compute $\hat{f}(y, \overline{x})$ (Using Definition 5.4)
4. Compute the unique monic factorization $\hat{f}(y, \overline{x}) = \prod_{i=1}^{\hat{m}} \hat{h}_i^{e_i}(y, \overline{x})$ (Using Algorithm 2)
5. **foreach** $i \in [\hat{m}]$ **do** $h_i(x_1, \ldots, x_n, x_{n+1}) \leftarrow \hat{h}_i(f_k \cdot x_{n+1}, \overline{x})$;
6. Recursively compute a factorization of $f_k(x_1, \ldots, x_n) = \prod_{j=1}^{r} w_j^{\beta_j}(x_1, \ldots, x_n)$
7. **for** $j \leftarrow 1$ **to** r **do**
   $\alpha_j \leftarrow -\beta_j \cdot (k-1)$;
   for $i \leftarrow 1$ **to** $\hat{m}$ **do**
      Find the maximal d_{ij} such that $w_j^{d_{ij}} \mid h_i$; /* By iteratively applying Lemma 2.7 with $t = \xi(n, d^2, s^d)$ */
      $\alpha_j \leftarrow \alpha_j + d_{ij} \cdot e_i$; $h_i \leftarrow h_i / w_j^{d_{ij}}$;
   end
   end
8. **return** $h_1, \ldots, h_{\hat{m}}, w_1, \ldots, w_r$ and $e_1, \ldots, e_{\hat{m}}, \alpha_1, \ldots, \alpha_r$; /* Return only those where $\alpha_j > 0$ */

Algorithm 3: Main Algorithm: overview

Input: s-sparse polynomial $f \in \mathbb{F}[x_1, x_2, \ldots, x_{n+1}]$ with individual degrees at most d.
Output: irreducible factors $h_1, h_2, \ldots, h_m$, and $e_1, e_2, \ldots, e_m$ such that $f = h_1^{e_1} \cdots h_m^{e_m}$

1. **if** $n \leq 1$ **then** Return the bi-variate factorization of f;
2. $k = \deg_{x_{n+1}}(f)$; $f_k \leftarrow \mathrm{lc}_{x_{n+1}}(f)$;
3. Compute $\hat{f}(y, \overline{x})$ (Using Definition 5.4)
4. Compute the unique monic factorization $\hat{f}(y, \overline{x}) = \prod_{i=1}^{\hat{m}} \hat{h}_i^{e_i}(y, \overline{x})$ (Using Algorithm 2)
5. **foreach** $i \in [\hat{m}]$ **do** $h_i(x_1, \ldots, x_n, x_{n+1}) \leftarrow \hat{h}_i(f_k \cdot x_{n+1}, \overline{x})$;
6. Recursively compute a factorization of $f_k(x_1, \ldots, x_n) = \prod_{j=1}^{r} w_j^{\beta_j}(x_1, \ldots, x_n)$
7. **for** $j \leftarrow 1$ **to** r **do**
   $\alpha_j \leftarrow -\beta_j \cdot (k-1)$;
   for $i \leftarrow 1$ **to** $\hat{m}$ **do**
      Find the maximal d_{ij} such that $w_j^{d_{ij}} \mid h_i$; /* By iteratively applying Lemma 2.7 with $t = \xi(n, d^2, s^d)$ */
      $\alpha_j \leftarrow \alpha_j + d_{ij} \cdot e_i$; $h_i \leftarrow h_i / w_j^{d_{ij}}$;
   end
   end
8. **return** $h_1, \ldots, h_{\hat{m}}, w_1, \ldots, w_r$ and $e_1, \ldots, e_{\hat{m}}, \alpha_1, \ldots, \alpha_r$; /* Return only those where $\alpha_j > 0$ */

Algorithm 3: Main Algorithm: overview

Time analysis of the division step

Lemma 2.7 Let $f, g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be s-sparse polynomials of degree at most d. Then there exists an algorithm that given f, g and t uses poly(n, d, s, t) field operations and computes the quotient polynomial of f and g, if it is a t-sparse polynomial. That is, if $f = gh$ for some $h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, $||h|| \leq t$, then the algorithm outputs h. Otherwise, the algorithm rejects.

Time analysis of the division step

Lemma 2.7 Let $f, g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be s-sparse polynomials of degree at most d. Then there exists an algorithm that given f, g and t uses poly(n, d, s, t) field operations and computes the quotient polynomial of f and g, if it is a t-sparse polynomial. That is, if $f = gh$ for some $h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, $||h|| \leq t$, then the algorithm outputs h. Otherwise, the algorithm rejects.

Proof.

- Assume there is an $h(\underline{\boldsymbol{x}})$ such that $f = gh$. Observe that h has individual degree bounded by d
- We can construct an "almost" black-box access for $h(\underline{\boldsymbol{x}})$: Given $\underline{a}$, $h(\underline{a}) = f(\underline{a})/g(\underline{a})$
- If h is a t-sparse polynomial, we can use our reconstruction algorithm to construct the monomial representation of $h(\underline{\boldsymbol{x}})$
- The reconstruction algorithm will use poly(n, d, s, t) steps

Time analysis of the division step

Lemma 2.7 Let $f, g \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ be s-sparse polynomials of degree at most d. Then there exists an algorithm that given f, g and t uses poly(n, d, s, t) field operations and computes the quotient polynomial of f and g, if it is a t-sparse polynomial. That is, if $f = gh$ for some $h \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, $||h|| \leq t$, then the algorithm outputs h. Otherwise, the algorithm rejects.

Proof.

Algorithm:

1. Think of the map $\underline{a} \mapsto f(\underline{a})/g(\underline{a})$ as a blackbox access $\mathcal{B}$
2. Giving $n, d, t, \mathcal{B}$ as input to reconstruction algorithm, obtain $h(\underline{\boldsymbol{x}})$
3. If $f = gh$ return h; Else reject. $\square$

Think: What will happen if g doesn't divide f

References

1. V. Bhargava, S. Saraf, I. Volkovich. Deterministic Factorization of Sparse Polynomials with Bounded Individual Degree.
2. E. Kaltofen. Factorization of polynomials given by straight-line programs. In S. Micali, editor, *Randomness in Computation*, volume 5 of *Advances in Computing Research*, pages 375–412. JAI Press Inc., Greenwhich, Connecticut, 1989.
3. A. Klivans and D. Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 216–223, 2001.