Tensor decompositions and Theoretical computer science

Lecture 1: Tensors and complexity of bilinear maps

Vladimir Lysikov

12.09.2022

Computational model

To study complexity of problems, we need to formally define:

- What is a computational problem?
- What is an algorithm?
- How to check if an algorithm solves a problem?
- How to measure complexity of a computation?

- How to compose and modify algorithms?
- How to use modification of algorithms to reduce one problems to others?

Evaluation of polynomials

We focus on one of the simplest algebraic problems:

$F \in k[x_1, \ldots, x_m]$ is a polynomial.

Given $x_1, \ldots, x_m$, compute the value of F

Evaluation of polynomials

We focus on one of the simplest algebraic problems:

> $F \in k[x_1, \dots, x_m]$ is a polynomial.
>
> Given $x_1, \dots, x_m$, compute the value of F

Usually polynomials we want to compute come in families

Evaluation of polynomials

We focus on one of the simplest algebraic problems:

$F \in k[x_1, \ldots, x_m]$ is a polynomial.

Given $x_1, \ldots, x_m$, compute the value of F

Usually polynomials we want to compute come in families

$$\det_n(x_{11}, \ldots, x_{nn}) = \sum_{\sigma \in S_n} (-1)^{\operatorname{sign} \sigma} \prod_{i=1}^{n} x_{i,\sigma i}$$

$$e_{n,d}(x_1, \ldots, x_n) = \sum_{\substack{S \subset [n] \\ \sharp S = d}} \prod_{i \in S} x_i$$

$$p_n(x_1, \ldots, x_n) = \sum_{i=1}^{n} x_i^n$$

Computational model of formulas

- Problems: polynomials or sequences of polynomials
- Algorithms: formulas
- We consider formulas which use variables, constants and operations $+$, $-$, $\times$, but not powers, complicated sum notations etc.
- Complexity measure: $\mathrm{fs}(\Phi) =$ number of operations in a formula Φ

$$3 \cdot x \cdot y \cdot (x + y) + x \cdot x \cdot x \qquad \mathrm{fs} = 7$$
$$(x + y) \cdot (x + y) \cdot (x + y) - y \cdot y \cdot y \qquad \mathrm{fs} = 8$$

$$\mathrm{fs}(F) = \min\{\mathrm{fs}(\Phi) \mid \Phi \text{ is a formula computing } F\}$$

An issue with formulas

The computational model of formulas does not allow reuse of intermediate computations

$$(x + y) \cdot (x + y) \cdot (x + y) - y \cdot y \cdot y \qquad \mathrm{fs} = 8$$

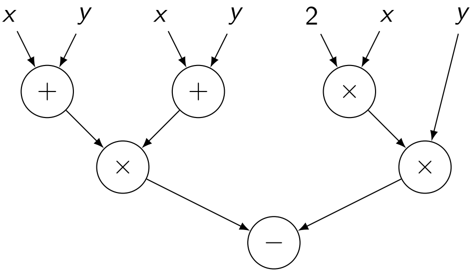
Can be computed as follows:

$$u \leftarrow x + y$$
$$u \cdot u \cdot u - y \cdot y \cdot y$$

This computation uses 6 operations.

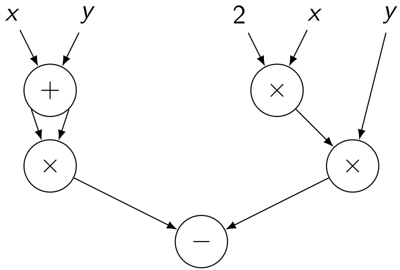
Formulas as trees

$$(x + y) \cdot (x + y) - 2 \cdot x \cdot y$$



From formulas to circuits

$$(x + y) \cdot (x + y) - 2 \cdot x \cdot y$$



Definition of a circuit

A *circuit* is a labeled directed acyclic graph with vertices of two types:

- vertices with indegree 0 (sources) are labeled by constants or variables,
- vertices with indegree 2 (gates) are labeled by $\times$, $+$, $-$.

One vertex is labeled as an *output node*

Each vertex γ of a circuit computes a polynomial $\hat{\gamma}$

- if γ is a source, then $\hat{\gamma}$ is the corresponding constant or variable
- if γ is a gate of type $\circ$ with arrows from α and β, then $\hat{\gamma} = \hat{\alpha} \circ \hat{\beta}$

The polynomial computed at the output gate is the polynomial computed by the circuit

A tree circuit is called a *formula*

Definition of a circuit

A *circuit* is a labeled directed acyclic graph with vertices of two types:

- vertices with indegree 0 (sources) are labeled by constants or variables,
- vertices with indegree 2 (gates) are labeled by $\times$, $+$, $-$.

Some vertices are labeled as *output nodes*

Each vertex γ of a circuit computes a polynomial $\hat{\gamma}$

- if γ is a source, then $\hat{\gamma}$ is the corresponding constant or variable
- if γ is a gate of type $\circ$ with arrows from α and β, then $\hat{\gamma} = \hat{\alpha} \circ \hat{\beta}$

A circuit computes a polynomial map given by polynomials computed at the output nodes

A tree circuit is called a *formula*

Computational model of circuits

- Problems: polynomials or sequences of polynomials
- Algorithms: circuits
- Complexity measure: $\mathrm{cs}(\Phi) =$ number of gates in a circuit Φ

$$\mathrm{cs}(F) = \min\{\mathrm{cs}(\Phi) \mid \Phi \text{ is a circuit computing } F\}$$

Circuit lower bounds

It is not hard to prove that a random polynomial needs a reasonably high circuit size

- There is a finite number of underlying graphs for circuits of size s
- There is a finite number of gate labelings
- A circuit of size s contains at most s constants
- The set of all polynomials of degree at most d computed by size s circuits is a constructible set of dimension at most s

We don't want to compute random polynomials, but we cannot prove good lower bounds for explicitly written polynomials
The best lower bound: $\Omega(n \log d)$ (Baur & Strassen 83)

Multilinear maps

If a polynomial map is multilinear, then there are some circuits that make multlinearity apparent

Multilinear formula

$$x_1 y_1 + x_2 y_2$$

Non-multilinear formula

$$(x_1 + y_2)(x_2 + y_1) - x_1 x_2 - y_1 y_2$$

Multlinear circuits have more structure to exploit
But is it enough to only consider multilinear circuits?

Multidegree

- $V_1, V_2, \dots, V_d$ — vector spaces
- a vector in each space V_i are represented by $\dim V_i$ variables
- consider polynomials from $\mathbf{k}[V_1 \oplus V_2 \oplus \dots \oplus V_d]$
- a polynomial is *homogeneous of multidegree* $D \in \mathbb{Z}^d$ if each monomial has degree D_i in variables corresponding to V_i
- $\mathbf{k}[V_1 \oplus V_2 \oplus \dots \oplus V_d]$ is $\mathbb{Z}^d$-graded by multidegree
- multilinear forms $F\colon V_1 \times V_2 \times \dots \times V_k \to \mathbf{k}$ correspond to multidegree $(1, 1, \dots, 1)$ polynomials
- if F is a polynomial, denote by F_D its part of multidegree D

Multilinear circuits

A circuit computing polynomials in $\mathbf{k}[V_1 \oplus V_2 \oplus \cdots \oplus V_d]$ is *multilinear* if

- Each vertex is marked by a multidegree in $\{0, 1\}^d$
- Nonzero constants are marked with multidegree 0
- Variables for V_i are marked with multidegree e_i
- Arguments of an addition gate γ ($+$ or $-$) must have the same multidegree as γ
- Multidegree of a multiplication gate γ is the sum of multidegrees of its arguments

Claim

Each gate in a multilinear circuit computes a homogeneous polynomial of the corresponding multidegree

Multilinearization theorem

Theorem (Nisan & Wigderson 95)

If a multilinear map $F\colon V_1 \times V_2 \times \cdots \times V_d \to W$ can be computed by a circuit of size s, then it can be computed by a multilinear circuit of size at most $2 \cdot 3^d s$

Construction

Let Φ be a circuit computing F. We construct a multilinear circuit Φ'.

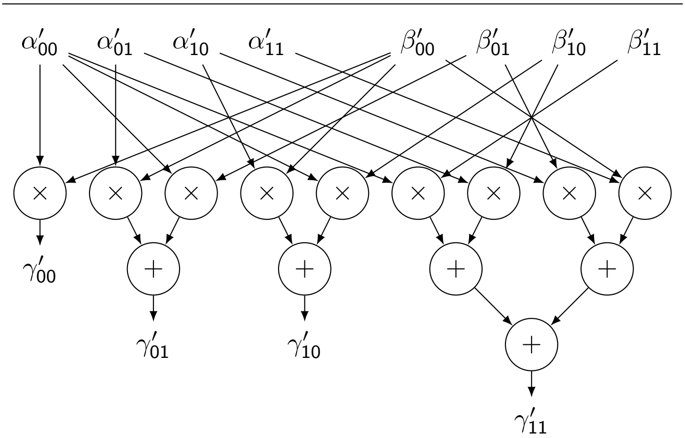
For each vertex γ in Φ, there will be 2^d vertices γ'_D in Φ' which compute multidegree D homogeneous parts of $\hat{\gamma}$ (for all $D \in \{0,1\}^d$).

- Each constant source γ is replaced with 2^d constants: $\gamma'_0 = \gamma$, other $\gamma'_D = 0$
- Each variable source γ is replaced with 2^d sources: $\gamma'_{e_i} = \gamma$, other $\gamma'_D = 0$
- Each addition gate $\gamma = \alpha \pm \beta$ is replaced with 2^d addition gates $\gamma'_D = \alpha'_D \pm \beta'_D$

Construction: multiplication

- Each multiplication gate $\gamma = \alpha \times \beta$ is replaced with 2^d subcircuits implementing the relation

$$\gamma'_D = \sum_{E \leq D} \alpha'_E \cdot \beta'_{D-E}$$



Construction

- Each constant source γ is replaced with 2^d constants: $\gamma'_0 = \gamma$, other $\gamma'_D = 0$
- Each variable source γ is replaced with 2^d sources: $\gamma'_{e_i} = \gamma$, other $\gamma'_D = 0$
- Each addition gate $\gamma = \alpha \pm \beta$ is replaced with 2^d addition gates $\gamma'_D = \alpha'_D \pm \beta'_D$
- Each multiplication gate $\gamma = \alpha \times \beta$ is replaced with 2^d subcircuits implementing the relation

$$\gamma'_D = \sum_{E \le D} \alpha'_E \cdot \beta'_{D-E}$$

Suppose D contains k ones and $d - k$ zeros. The corresponding multiplication subcircuit will have 2^k multiplications and $2^k - 1$ additions.
The total number of new gates per multiplication: $< 2 \cdot \sum_{k=0}^{d} \binom{n}{k} 2^k = 2 \cdot 3^d$.

Multilinearization

Theorem (Nisan & Wigderson 95)

If a multilinear map $F\colon V_1 \times V_2 \times \cdots \times V_d \to W$ can be computed by a circuit of size s, then it can be computed by a multilinear circuit of size at most $2 \cdot 3^d s$

Consider a sequence of multilinear maps $F_1, F_2, \dots, F_n, \dots$.
Denote by $L(n)$ the circuit complexity of F_n, and by $L_{mlin}(n)$ the multilinear circuit complexity.

- If $\deg F_n$ is the same, then $L(n)$ and $L_{mlin}(n)$ differ only by a constant factor
- If the degree grows like $\deg F_n = O(\log n)$, then:
  - $\mathrm{poly}(n)$ upper bounds on $L(n)$ translate to $L_{mlin}(n)$
  - superpolynomial lower bounds on $L_{mlin}(n)$ translate to $L(n)$
- But if the degree is high, then the restriction to multilinear circuits is not that useful for $L(n)$

Bilinear circuits

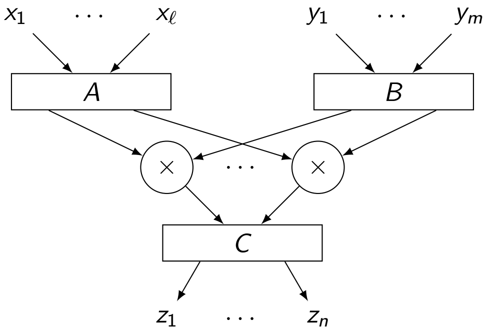
There are many important bilinear maps: matrix multiplication, polynomial multiplication, commutator of matrices etc.

What is the structure of bilinear circuits for a bilinear map $B\colon X \times Y \to Z$?

- There are 4 possible multidegrees: 00, 10, 01, 11
- Multidegree 00 — constants, can be replaced by a constant source vertex
- Multidegree 10 — linear forms in X
- Multidegree 01 — linear forms in Y
- Multidegree 11 can be obtained as a product of multidegrees 10 and 01
- All multidegree 11 gates, including output gates, are linear combinations of these products

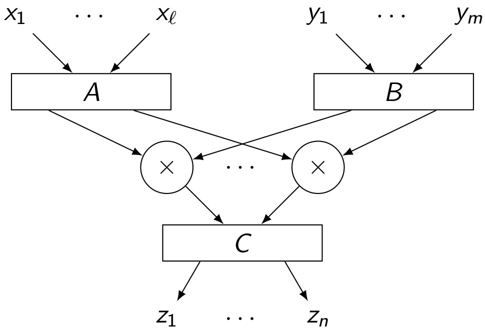
A multiplication gate is *proper* if it multiplies a linear form in X by a linear form in Y

Bilinear circuits



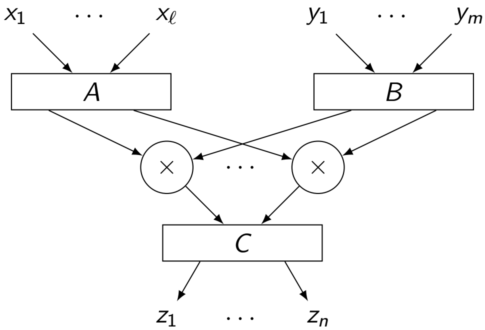
$$z_k = \sum_{a=1}^{r} w_{ak} f_a(x) g_b(y)$$

Bilinear circuits



$$B(x, y) = \sum_{a=1}^{r} f_a(x) g_a(y) w_k$$

Bilinear circuits



$$B = \sum_{a=1}^{r} f_a \otimes g_a \otimes w_a$$

Rank and bilinear complexity

Theorem (Strassen 73)

Minimal number of proper multiplication in a bilinear circuit computing a bilinear map B is equal to the rank of the corresponding tensor

More on algebraic complexity

R. Saptharishi. *A survey of lower bounds in arithmetic circuit complexity*

`https://github.com/dasarpmar/lowerbounds-survey`