Tensors in Computations III

Lek-Heng Lim

University of Chicago

recap from lecture II

recap: three definitions

- tensors capture three great ideas:
  ① equivariance
  ② multilinearity
  ③ separability
- roughly correspond to three common definitions of a tensor
  ① a multi-indexed object that satisfies tensor transformation rules
  ② a multilinear map
  ③ an element of a tensor product of vector spaces

recap: definition ②

- recall: is this

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

  a tensor?

- makes no sense
- suppose it does represent a tensor, what kind of tensor is it?
- answer: can be
  - covariant 2-tensor $\beta : \mathbb{V} \times \mathbb{V} \to \mathbb{R}$
  - contravariant 2-tensor $\varphi : \mathbb{V}^* \times \mathbb{V}^* \to \mathbb{R}$
  - mixed 2-tensor $\Phi : \mathbb{V} \to \mathbb{V}$
  - contravariant 1-tensor $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3) \in \mathbb{V} \oplus \mathbb{V} \oplus \mathbb{V}$
  - covariant 1-tensor $(\varphi_1, \varphi_2, \varphi_3) \in \mathbb{V}^* \oplus \mathbb{V}^* \oplus \mathbb{V}^*$
  - or yet other possibilities

  here $\mathbb{V}$ is any vector space of dimension three

recap: definition ②

- say it is a mixed 2-tensor, which $\Phi : \mathbb{V} \to \mathbb{V}$ does it represent?
- answer: with probability one, any $\Phi : \mathbb{V} \to \mathbb{V}$ can be represented as

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

  with respect to some choice of basis on $\mathbb{V}$
- moral: knowing

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

  tells us almost completely nothing about the tensor

recap: why important

- saw two examples

  higher derivatives: functions defined on spaces other than $\mathbb{R}^n$ like

  $$f, g : \mathbb{S}^n_{++} \to \mathbb{R}, \quad f(X) = \log\det(X), \quad g(X) = \mathrm{tr}(X^{-1})$$

  self-concordance: essential for defining this important notion

  $$\left|\nabla^3 f(\mathbf{x})(\mathbf{h},\mathbf{h},\mathbf{h})\right| \le 2\sigma \left|\nabla^2 f(\mathbf{x})(\mathbf{h},\mathbf{h})\right|^{3/2}$$

  $\nabla^3 f(\mathbf{x})$ is trilinear, $\nabla^2 f(\mathbf{x})$ is bilinear functional

- will say a bit more about these today

example: self-concordance

example: self-concordance

- log barrier for semidefinite programming

$$f : \mathbb{S}^n_{++} \to \mathbb{R}, \quad f(X) = -\log\det(X)$$

- inverse barrier for semidefinite programming

$$g : \mathbb{S}^n_{++} \to \mathbb{R}, \quad g(X) = \mathrm{tr}(X^{-1})$$

- why don't we ever see the latter?

example: self-concordance

- log barrier f

$$D^2 f(X)(H,H) = \mathrm{tr}(H^\top[\nabla^2 f(X)](H)) = \mathrm{tr}(HX^{-1}HX^{-1})$$
$$D^3 f(X)(H,H,H) = \mathrm{tr}(H^\top[\nabla^3 f(X)](H,H)) = -2\,\mathrm{tr}(HX^{-1}HX^{-1}HX^{-1})$$

- self-concordant by Cauchy–Schwarz

$$|D^3 f(X)(H,H,H)| \le 2\|HX^{-1}\|^3 = 2[D^2 f(X)(H,H)]^{3/2}$$

- inverse barrier g

$$D^2 g(X)(H,H) = 2\,\mathrm{tr}(HX^{-1}HX^{-2})$$
$$D^3 g(X)(H,H,H) = -6\,\mathrm{tr}(HX^{-1}HX^{-1}HX^{-2})$$

- set $H = hI$, $X = xI$, then $6|h|^3/x^4 > 2(2h^2/x^3)^{3/2}$ as $x \to 0^+$, self-concordant condition fails when X is near singular

example: bilinear complexity

example: bilinear complexity

- given $\mathbb{U}$, $\mathbb{V}$, $\mathbb{W}$, how to construct a bilinear operator

$$\mathrm{B} : \mathbb{U} \times \mathbb{V} \to \mathbb{W}?$$

- take linear functional $\varphi : \mathbb{U} \to \mathbb{R}$, linear functional $\psi : \mathbb{V} \to \mathbb{R}$, vector $\mathbf{w} \in \mathbb{W}$, define

$$\mathrm{B}(\mathbf{u}, \mathbf{v}) = \varphi(\mathbf{u})\psi(\mathbf{v})\mathbf{w}$$

  for any $\mathbf{u} \in \mathbb{U}$, $\mathbf{v} \in \mathbb{V}$
- evaluating B requires exactly one multiplication of variables
- call such a bilinear operator rank-one
- every bilinear operator is a sum of rank-one bilinear operators

example: bilinear complexity

- for example $\mathbb{U} = \mathbb{V} = \mathbb{W} = \mathbb{R}^3$ with

$$\varphi(\mathbf{u}) = u_1 + 2u_2 + 3u_3$$
$$\psi(\mathbf{v}) = 2v_1 + 3v_2 + 4v_3$$
$$\mathbf{w} = (3, 4, 5)$$

  then

$$\mathrm{B}(\mathbf{u}, \mathbf{v}) = \begin{bmatrix} 3(u_1 + 2u_2 + 3u_3)(2v_1 + 3v_2 + 4v_3) \\ 4(u_1 + 2u_2 + 3u_3)(2v_1 + 3v_2 + 4v_3) \\ 5(u_1 + 2u_2 + 3u_3)(2v_1 + 3v_2 + 4v_3) \end{bmatrix}$$

- multiplications like $2u_2$ or $4v_3$ are all scalar multiplications, i.e., one of the factors is a constant
- only variable multiplication like $(u_1 + 2u_2 + 3u_3)(2v_1 + 3v_2 + 4v_3)$ counts

example: bilinear complexity

- this is the notion of bilinear complexity [Strassen, 1987]
- once we fixed φ, ψ, w, evaluation of these can be hardwired or hardcoded
- e.g., discrete Fourier transform

$$\begin{bmatrix} x'_0 \\ x'_1 \\ x'_2 \\ x'_3 \\ \vdots \\ x'_{n-1} \end{bmatrix} = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \cdots & \omega^{2(n-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \cdots & \omega^{3(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \omega^{3(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \end{bmatrix}$$

- may use FFT to evaluate DFT
- bilinear complexity of DFT or FFT all the same, namely, zero

example: bilinear complexity

- may often bound number of additions and scalar multiplications in terms of number of variable multiplications
- e.g., if an algorithm takes n^p variable multiplications, may show that it takes at most $10n^p$ additions and scalar multiplications
- so algorithm still $O(n^p)$ even if we count all arithmetic operations
- most importantly, bilinear complexity = tensor rank

$$\operatorname{rank}(\mathrm{B}) = \min\left\{ r : \mathrm{B}(\mathbf{u}, \mathbf{v}) = \sum_{i=1}^{r} \varphi_i(\mathbf{u})\psi_i(\mathbf{v})\mathbf{w}_i \right\}$$

- if only need $\mathrm{B}(\mathbf{u}, \mathbf{v})$ up to ε-accuracy, border rank

$$\overline{\operatorname{rank}}(\mathrm{B}) = \min\left\{ r : \mathrm{B}(\mathbf{u}, \mathbf{v}) = \lim_{\varepsilon \to 0^+} \sum_{i=1}^{r} \varphi_i^\varepsilon(\mathbf{u})\psi_i^\varepsilon(\mathbf{v})\mathbf{w}_i^\varepsilon \right\}$$

- due to [Strassen, 1973] and [Bini–Lotti–Romani, 1980] respectively

example: Gauss's algorithm

- complex multiplication with three real multiplications

$$\begin{aligned}(a+bi)(c+di) &= (ac-bd)+i(bc+ad) \\ &= (ac-bd)+i[(a+b)(c+d)-ac-bd]\end{aligned}$$

- $\mathrm{B}:\mathbb{C}\times\mathbb{C}\to\mathbb{C}$, $(z,w)\mapsto zw$ is $\mathbb{R}$-bilinear

$$\mathrm{B}:\mathbb{R}^2\times\mathbb{R}^2\to\mathbb{R}^2,\quad \mathrm{B}\left(\begin{bmatrix}a\\b\end{bmatrix},\begin{bmatrix}c\\d\end{bmatrix}\right)=\begin{bmatrix}ac-bd\\bc+ad\end{bmatrix}$$

- usual:

$$\mathrm{B}(z,w)=[\mathbf{e}_1^*(z)\mathbf{e}_1^*(w)-\mathbf{e}_2^*(z)\mathbf{e}_2^*(w)]\mathbf{e}_1+[\mathbf{e}_1^*(z)\mathbf{e}_2^*(w)+\mathbf{e}_2^*(z)\mathbf{e}_1^*(w)]\mathbf{e}_2$$

- Gauss:

$$\begin{aligned}\mathrm{B}(z,w)=[(\mathbf{e}_1^*+\mathbf{e}_2^*)(z)(\mathbf{e}_1^*+\mathbf{e}_2^*)(w)]\mathbf{e}_2 \\ +[\mathbf{e}_1^*(z)\mathbf{e}_1^*(w)](\mathbf{e}_1-\mathbf{e}_2)-[\mathbf{e}_2^*(z)\mathbf{e}_2^*(w)](\mathbf{e}_1+\mathbf{e}_2)\end{aligned}$$

example: Gauss's algorithm

- Gauss optimal in both exact and approximate sense:

$$\text{rank}(\mathrm{B}) = 3 = \overline{\text{rank}}(\mathrm{B})$$

- why useful?
- complex matrix multiplication:

$$(A + iB)(C + iD) = (AC - BD) + i[(A + B)(C + D) - AC - BD]$$

for $A + iB, C + iD \in \mathbb{C}^{n\times n}$ with $A, B, C, D \in \mathbb{R}^{n\times n}$

- which is why we should allow for modules
    - $\mathbb{C}$ two-dimensional vector space over $\mathbb{R}$
    - $\mathbb{C}^{n\times n}$ two-dimensional free module over $\mathbb{R}^{n\times n}$

other simple example?

- Gauss essentially the only one in two dimensions
- parallel evaluation of standard inner product and standard symplectic form on $\mathbb{R}^2$

$$g(x, y) = x_1 y_1 + x_2 y_2, \qquad \omega(x, y) = x_1 y_2 - x_2 y_1$$

- algorithm similar to Gauss's gives result with $\operatorname{rank}(\mathrm{B}) = 3 = \overline{\operatorname{rank}}(\mathrm{B})$
- three dimensions: skew-symmetric matrix-vector product

$$\begin{bmatrix} 0 & a & b \\ -a & 0 & c \\ -b & -c & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} ay + bz \\ -ax + cz \\ -bx - cy \end{bmatrix}$$

- in this case[1] $\operatorname{rank}(\mathrm{B}) = 5 = \overline{\operatorname{rank}}(\mathrm{B})$

[1]thanks to J. M. Landsberg (for $\mathbb{C}$) and Visu Makam (for $\mathbb{R}$)

example: Strassen's algorithm

- 2×2 matrix multiplication with seven multiplications

$$\begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \begin{bmatrix} b_1 & b_2 \\ b_3 & b_4 \end{bmatrix} = \begin{bmatrix} a_1 b_1 + a_2 b_2 & \beta + \gamma + (a_1 + a_2 - a_3 - a_4) b_4 \\ \alpha + \gamma + a_4 (b_2 + b_3 - b_1 - b_4) & \alpha + \beta + \gamma \end{bmatrix}$$

  with

$$\alpha = (a_3 - a_1)(b_3 - b_4),\ \beta = (a_3 + a_4)(b_3 - b_1),\ \gamma = a_1 b_1 + (a_3 + a_4 - a_1)(b_1 + b_4 - b_3)$$

- consequence: inverting $n \times n$ matrix in $5.64 n^{\log_2 7}$ arithmetic operations (both additions and multiplications) [Strassen, 1969]
- huge surprise as there were results showing $n^3/3$ required by Gaussian elimination cannot be improved
- such results assume row and column operations, Strassen used block operations
- $\operatorname{rank}(\mathrm{B}) = 7 = \overline{\operatorname{rank}}(\mathrm{B})$ [Landsberg, 2006]

example: exponent of matrix multiplication

- bilinear operator

$$\mu_{m,n,p} : \mathbb{R}^{m\times n} \times \mathbb{R}^{n\times p} \to \mathbb{R}^{n\times p}, \quad (A, B) \mapsto AB$$

  called matrix multiplication tensor
- exponent of matrix multiplication is

$$\omega := \inf\{p \in \mathbb{R} : \operatorname{rank}(\mu_{n,n,n}) = O(n^p)\}$$

- current bound $\omega < 2.3728596$ [Alman–Vassilevska Williams, 2021]
- ω underlies nearly every problem in numerical linear algebra

example: exponent of matrix multiplication

- inversion: given $A \in \mathrm{GL}(n)$, find $A^{-1} \in \mathrm{GL}(n)$
- determinant: given $A \in \mathrm{GL}(n)$, find $\det(A) \in \mathbb{R}$
- null basis: given $A \in \mathbb{R}^{n \times n}$, find a basis $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{R}^n$ of $\ker(A)$
- linear system: given $A \in \mathrm{GL}(n)$ and $\mathbf{b} \in \mathbb{R}^n$, find $\mathbf{v} \in \mathbb{R}^n$ so that $A\mathbf{v} = \mathbf{b}$
- LU decomposition: given $A \in \mathbb{R}^{m \times n}$ of full rank, find permutation P, unit lower triangular $L \in \mathbb{R}^{m \times m}$, upper triangular $U \in \mathbb{R}^{m \times n}$ so that $PA = LU$
- QR decomposition: given $A \in \mathbb{R}^{n \times n}$, find orthogonal $Q \in \mathrm{O}(n)$, upper triangular $U \in \mathbb{R}^{n \times n}$ so that $A = QR$

example: exponent of matrix multiplication

- eigenvalue decomposition: given $A \in \mathbb{S}^n$, find $Q \in \mathrm{O}(n)$ and diagonal $\Lambda \in \mathbb{R}^{n \times n}$ so that $A = Q\Lambda Q^{\mathsf{T}}$
- Hessenberg decomposition: given $A \in \mathbb{R}^{n \times n}$, find $Q \in \mathrm{O}(n)$ and upper Hessenberg $H \in \mathbb{R}^{n \times n}$ so that $A = QHQ^{\mathsf{T}}$
- characteristic polynomial: given $A \in \mathbb{R}^{n \times n}$, find $(a_0, \dots, a_{n-1}) \in \mathbb{R}^n$ so that $\det(xI - A) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$
- sparsification: given $A \in \mathbb{R}^{n \times n}$ and $c \in [1, \infty)$, find $X, Y \in \mathrm{GL}(n)$ so that $\mathrm{nnz}(XAY^{-1}) \le cn$

exponent of nearly all matrix computations

any $\varepsilon > 0$, there is an algorithm for each of these problems in $O(n^{\omega+\varepsilon})$ arithmetic operations (including additions and scalar multiplications)

example: integer multiplication

example: integer multiplication

- need to consider tensors over modules, i.e., replace field of scalars like $\mathbb{R}$ or $\mathbb{C}$ by a ring like $\mathbb{Z}$
- integer multiplication

$$\mathrm{B}\colon \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}, \qquad (a, b) \mapsto ab$$

  bilinear map over the $\mathbb{Z}$-module $\mathbb{Z}$
- but this is not the relevant module structure in fast integer multiplication algorithms

example: integer multiplication

- unsigned integers represented as polynomials

$$a = \sum_{i=0}^{p-1} a_i \theta^i =: a(\theta), \qquad b = \sum_{j=0}^{p-1} b_j \theta^j =: b(\theta)$$

  for some number base θ

- product has coefficients given by convolutions

$$ab = \sum_{k=0}^{2p-2} c_k \theta^k =: c(\theta), \qquad c_k = \sum_{i=0}^{k} a_i b_{k-i}$$

- set $n = 2p - 1$ and pad vectors of coefficients with enough zeros

$$(a_0, \ldots, a_{n-1}), \quad (b_0, \ldots, b_{n-1}), \quad (c_0, \ldots, c_{n-1})$$

example: integer multiplication

- use DFT for some root of unity ω to perform convolution

$$\begin{bmatrix} a'_0 \\ a'_1 \\ a'_2 \\ \vdots \\ a'_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{bmatrix}$$

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ \vdots \\ b'_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{bmatrix}$$

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{bmatrix} = \frac{1}{n} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \cdots & \omega^{-(n-1)} \\ 1 & \omega^{-2} & \omega^{-4} & \cdots & \omega^{-2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-(n-1)} & \omega^{-2(n-1)} & \cdots & \omega^{-(n-1)(n-1)} \end{bmatrix} \begin{bmatrix} a'_0 b'_0 \\ a'_1 b'_1 \\ a'_2 b'_2 \\ \vdots \\ a'_{n-1} b'_{n-1} \end{bmatrix}$$

example: integer multiplication

- Fourier transform turns convolution $*$ into pointwise product $\cdot$

$$\mathbf{a} * \mathbf{b} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{a}) \cdot \mathcal{F}(\mathbf{b}))$$

- key idea 1: convert integer multiplication to a bilinear operator

$$\begin{aligned} \mathrm{B}_1 \colon (\mathbb{Z}/2^s\mathbb{Z})[\theta] \times (\mathbb{Z}/2^s\mathbb{Z})[\theta] &\to (\mathbb{Z}/2^s\mathbb{Z})[\theta] \\ (a(\theta), b(\theta)) &\mapsto a(\theta)b(\theta) \end{aligned}$$

- key idea 2: a Fourier conversion into another bilinear operator

$$\begin{aligned} \mathrm{B}_2 \colon (\mathbb{Z}/m\mathbb{Z})^n \times (\mathbb{Z}/m\mathbb{Z})^n &\to (\mathbb{Z}/m\mathbb{Z})^n \\ ((a_0, \ldots, a_{n-1}), (b_0, \ldots, b_{n-1})) &\mapsto (a_0 b_0, \ldots, a_{n-1} b_{n-1}) \end{aligned}$$

- $(\mathbb{Z}/2^s\mathbb{Z})[\theta]$ is a $\mathbb{Z}/2^s\mathbb{Z}$-module
- $(\mathbb{Z}/m\mathbb{Z})^n$ is a $\mathbb{Z}/m\mathbb{Z}$-module

example: integer multiplication

- all fast integer multiplication algorithms based on variation of these ideas: [Karatsuba–Ofman, 1962], [Cook–Aanderaa, 1969], [Toom, 1963], [Schönhage–Strassen, 1971], [Fürer, 2009]
- sensational breakthrough by [Harvey–van der Hoeven, 2021]: $O(n \log n)$-algorithm for n-bit integer multiplication
- clever idea: use multidimensional DFT

$$a'(\phi_1, \phi_2, \ldots, \phi_d) = \sum_{\theta_1=0}^{n_1} \cdots \sum_{\theta_d=0}^{n_d} \omega_1^{\phi_1\theta_1} \omega_2^{\phi_2\theta_2} \cdots \omega_d^{\phi_d\theta_d} a(\theta_1, \theta_2, \ldots, \theta_d),$$

- replace bilinear operator with d-linear operator

example: cryptography

example: Diffie–Hellman key exchange

- Alice and Bob want to generate (secure) common password over (insecure) internet
- pick large prime p and primitive root of unity $g \in (\mathbb{Z}/p\mathbb{Z})^\times$
- any non-zero $x \in \mathbb{Z}/p\mathbb{Z}$ may be expressed as

$$x \equiv g^a \pmod{p}$$

  henceforth write $x = g^a$
- Alice picks secret $a \in \mathbb{Z}$ and sends g^a publicly to Bob
- Bob picks secret $b \in \mathbb{Z}$ and sends g^b publicly to Alice
- Alice computes $g^{ab} = (g^b)^a$ from the g^b she received from Bob
- Bob computes $g^{ab} = (g^a)^b$ from the g^a he received from Alice
- they now share the secure password g^{ab}

example: multilinear cryptography

- security based on intractability of computing $a = \log_g(g^a)$
- observation 1: $(\mathbb{Z}/p\mathbb{Z})^\times$ is $\mathbb{Z}$-module
- observation 2: Diffie–Hellman is $\mathbb{Z}$-bilinear map

$$\mathrm{B}\colon (\mathbb{Z}/p\mathbb{Z})^\times \times (\mathbb{Z}/p\mathbb{Z})^\times \to (\mathbb{Z}/p\mathbb{Z})^\times, \quad (g^a, g^b) \mapsto g^{ab}$$

- for any $\lambda, \lambda' \in \mathbb{Z}$ and $g^a, g^b \in (\mathbb{Z}/p\mathbb{Z})^\times$

$$\mathrm{B}(g^{\lambda a + \lambda' a'}, g^b) = \mathrm{B}(g^a, g^b)^\lambda \mathrm{B}(g^{a'}, g^b)^{\lambda'}$$

- what if not two parties but 1000 parties, e.g., on Zoom or Teams?
- $d+1$ parties require each party doing $d+1$ exponentiations
- solution: cryptographic multilinear map [Boneh–Silverberg, 2003]

example: multilinear cryptography

- cryptographic d-linear map $\Phi : G \times \cdots \times G \to G$
- assumptions: discrete log in G hard, evaluating Φ easy
- ith party pick password a_i, perform one exponentiation to get g^{a_i}
- broadcast g^{a_i} to other parties, who will each do likewise
- every party now has $g^{a_1}, \dots, g^{a_{d+1}}$
- ith party will now compute

$$\Phi(g^{a_1}, \dots, g^{a_{i-1}}, g^{a_{i+1}}, \dots, g^{a_{d+1}})^{a_i} = \Phi(g, \dots, g)^{a_1 \cdots a_{d+1}}$$

- result is common password for the $d+1$ parties

no time for these

- tensor nuclear norm and numerical stability
- bilinear Hilbert transform and Calderon conjecture
- tensor fields: multilinear operators over $C^\infty(M)$-modules
- metric, Ricci, and Riemann curvature tensors
- Grothendieck inequality

classifying multilinear maps

problem with definition ②

- many more multilinear maps than there are types of tensors
- $d = 2$:
  - linear operators

$$\Phi : \mathbb{U}^* \to \mathbb{V}, \quad \Phi : \mathbb{U} \to \mathbb{V}^*, \quad \Phi : \mathbb{U}^* \to \mathbb{V}^*$$

  - bilinear functionals

$$\beta : \mathbb{U}^* \times \mathbb{V} \to \mathbb{R}, \quad \beta : \mathbb{U} \times \mathbb{V}^* \to \mathbb{R}, \quad \beta : \mathbb{U}^* \times \mathbb{V}^* \to \mathbb{R}$$

- $d = 3$:
  - bilinear operators

$$\mathrm{B} : \mathbb{U}^* \times \mathbb{V} \to \mathbb{W}, \ \mathrm{B} : \mathbb{U} \times \mathbb{V}^* \to \mathbb{W}, \dots, \mathrm{B} : \mathbb{U}^* \times \mathbb{V}^* \to \mathbb{W}^*$$

  - trilinear functionals

$$\tau : \mathbb{U}^* \times \mathbb{V} \times \mathbb{W} \to \mathbb{R}, \ \tau : \mathbb{U} \times \mathbb{V}^* \times \mathbb{W} \to \mathbb{R}, \dots, \tau : \mathbb{U}^* \times \mathbb{V}^* \times \mathbb{W}^* \to \mathbb{R}$$

  - more complicated maps

$$\Phi_1 : \mathbb{U} \to \mathsf{L}(\mathbb{V}; \mathbb{W}), \quad \Phi_2 : \mathsf{L}(\mathbb{U}; \mathbb{V}) \to \mathbb{W},$$

$$\beta_1 : \mathbb{U} \times \mathsf{L}(\mathbb{V}; \mathbb{W}) \to \mathbb{R}, \quad \beta_2 : \mathsf{L}(\mathbb{U}; \mathbb{V}) \times \mathbb{W} \to \mathbb{R}$$

problem with definition ②

- possibilities increase exponentially with order *d*
- ought to be only as many as types of transformation rules

covariant 2-tensor:	$\Phi : \mathbb{U} \to \mathbb{V}^*,$	$\beta : \mathbb{U} \times \mathbb{V} \to \mathbb{R}$
contravariant 2-tensor:	$\Phi : \mathbb{U}^* \to \mathbb{V},$	$\beta : \mathbb{U}^* \times \mathbb{V}^* \to \mathbb{R}$
mixed 2-tensor:	$\Phi : \mathbb{U} \to \mathbb{V},$	$\beta : \mathbb{U} \times \mathbb{V}^* \to \mathbb{R},$
	$\Phi : \mathbb{U}^* \to \mathbb{V}^*,$	$\beta : \mathbb{U}^* \times \mathbb{V} \to \mathbb{R}$

- definition ③ accomplishes this without reference to the transformation rules

imperfect fix

- only allow $\mathbb{W} = \mathbb{R}$
- d-tensor of contravariant order p and covariant order $d - p$ is multilinear functional

$$\varphi : \mathbb{V}_1^* \times \cdots \times \mathbb{V}_p^* \times \mathbb{V}_{p+1} \times \cdots \times \mathbb{V}_d \to \mathbb{R}$$

- excludes vectors, by far the most common 1-tensor
- excludes linear operators, by far the most common 2-tensor
- excludes bilinear operators, by far the most common 3-tensor
- e.g., instead of talking about $\mathbf{v} \in \mathbb{V}$, need to talk about linear functionals $f : \mathbb{V}^* \to \mathbb{R}$
- ultimately need definition ③

acknowledgment

- Simons Foundation Award no. 663281 granted to the Institute of Mathematics of the Polish Academy of Sciences for 2021–2023
- L.-H. Lim, "Tensors in computations," *Acta Numer.*, **30** (2021), pp. 555–764