CS 598 EVS: Tensor Computations

Basics of Tensor Computations

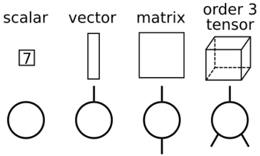
Edgar Solomonik

University of Illinois at Urbana-Champaign

Tensors

A *tensor* is a collection of elements

- *its dimensions define the size of the collection*
- *its order is the number of different dimensions*
- *specifying an index along each tensor mode defines an element of the tensor*



A few examples of tensors are

- *Order* 0 *tensors are scalars, e.g.,* $s \in \mathbb{R}$
- *Order* 1 *tensors are vectors, e.g.,* $\boldsymbol{v} \in \mathbb{R}^n$
- *Order* 2 *tensors are matrices, e.g.,* $\boldsymbol{A} \in \mathbb{R}^{m \times n}$
- *An order* 3 *tensor with dimensions* $s_1 \times s_2 \times s_3$ *is denoted as* $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{s_1 \times s_2 \times s_3}$ *with elements* t_{ijk} *for* $i \in \{1, \ldots, s_1\}, j \in \{1, \ldots, s_2\}, k \in \{1, \ldots, s_3\}$

Reshaping Tensors

Its often helpful to use alternative views of the same collection of elements

- *Folding* a tensor yields a higher-order tensor with the same elements
- *Unfolding* a tensor yields a lower-order tensor with the same elements
- *In linear algebra, we have the unfolding* $\boldsymbol{v} = \textit{vec}(\boldsymbol{A})$*, which stacks the columns of* $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ *to produce* $\boldsymbol{v} \in \mathbb{R}^{mn}$
- *For a tensor* $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{s_1 \times s_2 \times s_3}$, $\boldsymbol{v} = \textit{vec}(\boldsymbol{\mathcal{T}})$ *gives* $\boldsymbol{v} \in \mathbb{R}^{s_1 s_2 s_3}$ *with*

$$v_{i+(j-1)s_1+(k-1)s_1 s_2} = t_{ijk}$$

- *A common set of unfoldings is given by matricizations of a tensor, e.g., for order 3,*

$$\boldsymbol{T}_{(1)} \in \mathbb{R}^{s_1 \times s_2 s_3}, \boldsymbol{T}_{(2)} \in \mathbb{R}^{s_2 \times s_1 s_3}, \text{ and } \boldsymbol{T}_{(3)} \in \mathbb{R}^{s_3 \times s_1 s_2}$$

Matrices and Tensors as Operators and Multilinear Forms

- What is a matrix?
  - *A collection of numbers arranged into an array of dimensions* $m \times n$*, e.g.,* $\boldsymbol{M} \in \mathbb{R}^{m \times n}$
  - *A linear operator* $\boldsymbol{f_M}(\boldsymbol{x}) = \boldsymbol{M}\boldsymbol{x}$
  - *A bilinear form* $\boldsymbol{x}^T \boldsymbol{M}\boldsymbol{y}$
- What is a tensor?
  - *A collection of numbers arranged into an array of a particular order, with dimensions* $l \times m \times n \times \cdots$*, e.g.,* $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{l \times m \times n}$ *is order 3*
  - *A multilinear operator* $\boldsymbol{z} = \boldsymbol{f_M}(\boldsymbol{x}, \boldsymbol{y})$

$$z_i = \sum_{j,k} t_{ijk} x_j y_k$$

  - *A multilinear form* $\sum_{i,j,k} t_{ijk} x_i y_j z_k$

Tensor Transposition

For tensors of order $\geqslant 3$, there is more than one way to transpose modes

- *A* *tensor transposition* *is defined by a permutation* $\boldsymbol{p}$ *containing elements* $\{1, \dots, d\}$

$$y_{i_{p_1}, \dots, i_{p_d}} = x_{i_1, \dots, i_d}$$

- *In this notation, a transposition* $\boldsymbol{A}^T$ *of matrix* $\boldsymbol{A}$ *is defined by* $\boldsymbol{p} = [2, 1]$ *so that*

$$b_{i_2 i_1} = a_{i_1 i_2}$$

- *Tensor transposition is a convenient primitive for manipulating multidimensional arrays and mapping tensor computations to linear algebra*
- *When elementwise expressions are used in tensor algebra, indices are often carried through to avoid transpositions*

Tensor Symmetry

We say a tensor is *symmetric* if $\forall j, k \in \{1, \ldots, d\}$

$$t_{i_1 \ldots i_j \ldots i_k \ldots i_d} = t_{i_1 \ldots i_k \ldots i_j \ldots i_d}$$

A tensor is *antisymmetric* (skew-symmetric) if $\forall j, k \in \{1, \ldots, d\}$

$$t_{i_1 \ldots i_j \ldots i_k \ldots i_d} = (-1) t_{i_1 \ldots i_k \ldots i_j \ldots i_d}$$

A tensor is *partially-symmetric* if such index interchanges are restricted to be within disjoint subsets of $\{1, \ldots, d\}$, e.g., if the subsets for $d = 4$ and $\{1, 2\}$ and $\{3, 4\}$, then

$$t_{ijkl} = t_{jikl} = t_{jilk} = t_{ijlk}$$

Tensor Sparsity

We say a tensor $\mathcal{T}$ is *diagonal* if for some $\boldsymbol{v}$,

$$t_{i_1,\dots,i_d} = \begin{cases} v_{i_1} & : i_1 = \cdots = i_d \\ 0 & : \textit{otherwise} \end{cases} = v_{i_1}\delta_{i_1 i_2}\delta_{i_2 i_3}\cdots\delta_{i_{d-1} i_d}$$

- *In the literature, such tensors are sometimes also referred to as 'superdiagonal'*
- *Generalizes diagonal matrix*
- *A diagonal tensor is symmetric (and not antisymmetric)*

If most of the tensor entries are zeros, the tensor is *sparse*

- *Generalizes notion of sparse matrices*
- *Sparsity enables computational and memory savings*
- *We will consider data structures and algorithms for sparse tensor operations later in the course*

Tensor Products and Kronecker Products

Tensor products can be defined with respect to maps $f : V_f \to W_f$ and $g : V_g \to W_g$

$$h = f \times g \quad \Rightarrow \quad g : (V_f \times V_g) \to (W_f \times W_g), \quad h(x, y) = f(x)g(y)$$

Tensors can be used to represent multilinear maps and have a corresponding definition for a tensor product

$$\boldsymbol{T} = \boldsymbol{X} \times \boldsymbol{Y} \quad \Rightarrow \quad t_{i_1,\ldots,i_m,j_1,\ldots,j_n} = x_{i_1,\ldots,i_m} y_{j_1,\ldots,j_n}$$

The *Kronecker product* between two matrices $\boldsymbol{A} \in \mathbb{R}^{m_1 \times m_2}$, $\boldsymbol{B} \in \mathbb{R}^{n_1 \times n_2}$

$$\boldsymbol{C} = \boldsymbol{A} \otimes \boldsymbol{B} \quad \Rightarrow \quad c_{i_2+(i_1-1)m_2, j_2+(j_1-1)n_2} = a_{i_1 j_1} b_{i_2 j_2}$$

corresponds to transposing and unfolding the tensor product

Tensor Contractions

A *tensor contraction* multiplies elements of two tensors and computes partial sums to produce a third, in a fashion expressible by pairing up modes of different tensors, defining *einsum* (term stems from Einstein's summation convention)

tensor contraction	*einsum*	*diagram*
inner product	$w = \sum_i u_i v_i$	
outer product	$w_{ij} = u_i v_{ij}$	
pointwise product	$w_i = u_i v_i$	
Hadamard product	$w_{ij} = u_{ij} v_{ij}$	
matrix multiplication	$w_{ij} = \sum_k u_{ik} v_{kj}$	
batched mat.-mul.	$w_{ijl} = \sum_k u_{ikl} v_{kjl}$	
tensor times matrix	$w_{ilk} = \sum_j u_{ijk} v_{lj}$	

The terms 'contraction' and 'einsum' are also often used when more than two operands are involved

General Tensor Contractions

Given tensor $\mathcal{U}$ of order $s + v$ and $\mathcal{V}$ of order $v + t$, a tensor contraction summing over v modes can be written as

$$w_{i_1 \dots i_s j_1 \dots j_t} = \sum_{k_1 \dots k_v} u_{i_1 \dots i_s k_1 \dots k_v} v_{k_1 \dots k_v j_1 \dots j_t}$$

- *This form omits 'Hadamard indices', i.e., indices that appear in both inputs and the output (as with pointwise product, Hadamard product, and batched mat–mul.)*
- *Other contractions can be mapped to this form after transposition*

Unfolding the tensors reduces the tensor contraction to matrix multiplication

- *Combine (unfold) consecutive indices in appropriate groups of size* s ,t, *or* v
- *If all tensor modes are of dimension* n, *obtain matrix–matrix product* $\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B}$ *where* $\boldsymbol{C} \in \mathbb{R}^{n^s \times n^t}$, $\boldsymbol{A} \in \mathbb{R}^{n^s \times n^v}$, *and* $\boldsymbol{B} \in \mathbb{R}^{n^v \times n^t}$
- *Assuming classical matrix multiplication, contraction requires* n^{s+t+v} *elementwise products and* $n^{s+t+v} - n^{s+t}$ *additions*

Properties of Einsums

Given an elementwise expression containing a product of tensors, the operands commute

- *For example $\boldsymbol{A}\boldsymbol{B} \neq \boldsymbol{B}\boldsymbol{A}$, but*

$$\sum_k a_{ik} b_{kj} = \sum_k b_{kj} a_{ik}$$

- *Similarly with multiple terms, we can bring summations out and reorder as needed, e.g., for $\boldsymbol{A}\boldsymbol{B}\boldsymbol{C}$*

$$\sum_k a_{ik} (\sum_l b_{kl} c_{lj}) = \sum_{kl} c_{lj} b_{kl} a_{ik}$$

A contraction can be succinctly described by a *tensor diagram*

- *Indices in contractions are only meaningful in so far as they are matched up*
- *A tensor diagram is defined by a graph with a vertex for each tensor and an edge/leg for each index/mode*
- *Indices that are not-summed are drawn by pointing the legs/edges into whitespace*

Matrix-style Notation for Tensor Contractions

The *tensor times matrix* contraction along the mth mode of $\boldsymbol{\mathcal{U}}$ to produce $\boldsymbol{\mathcal{V}}$ is expressed as follows

$$\boldsymbol{\mathcal{W}} = \boldsymbol{\mathcal{U}} \times_m \boldsymbol{V} \Rightarrow \boldsymbol{W}_{(m)} = \boldsymbol{V}\boldsymbol{U}_{(m)}$$

- *$\boldsymbol{W}_{(m)}$ and $\boldsymbol{U}_{(m)}$ are unfoldings where the mth mode is mapped to be an index into rows of the matrix*
- *To perform multiple tensor times matrix products, can write, e.g.,*

$$\boldsymbol{\mathcal{W}} = \boldsymbol{\mathcal{U}} \times_1 \boldsymbol{X} \times_2 \boldsymbol{Y} \times_3 \boldsymbol{Z} \Rightarrow w_{ijk} = \sum_{pqr} u_{pqr} x_{ip} y_{jq} z_{kr}$$

The *Khatri-Rao product* of two matrices $\boldsymbol{U} \in \mathbb{R}^{m \times k}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times k}$ products $\boldsymbol{W} \in \mathbb{R}^{mn \times k}$ so that

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{u}_1 \otimes \boldsymbol{v}_1 & \cdots & \boldsymbol{u}_k \otimes \boldsymbol{v}_k \end{bmatrix}$$

The Khatri-Rao product computes the einsum $\hat{w}_{ijk} = u_{ik} v_{jk}$ then unfolds $\hat{\boldsymbol{\mathcal{W}}}$ so that $w_{i+(j-1)n,k} = \hat{w}_{ijk}$

Identities with Kronecker and Khatri-Rao Products

- Matrix multiplication is distributive over the Kronecker product

$$(\boldsymbol{A} \otimes \boldsymbol{B})(\boldsymbol{C} \otimes \boldsymbol{D}) = \boldsymbol{A}\boldsymbol{C} \otimes \boldsymbol{B}\boldsymbol{D}$$

we can derive this from the einsum expression

$$\sum_{kl} a_{ik} b_{jl} c_{kp} d_{lq} = \Big(\sum_{k} a_{ik} c_{kp}\Big)\Big(\sum_{l} b_{jl} d_{lq}\Big)$$

- For the Khatri-Rao product a similar distributive identity is

$$(\boldsymbol{A} \odot \boldsymbol{B})^T(\boldsymbol{C} \odot \boldsymbol{D}) = \boldsymbol{A}^T\boldsymbol{C} * \boldsymbol{B}^T\boldsymbol{D}$$

where $*$ denotes that Hadamard product, which holds since

$$\sum_{kl} a_{ki} b_{li} c_{kj} d_{lj} = \Big(\sum_{k} a_{ki} c_{kj}\Big)\Big(\sum_{l} b_{li} d_{lj}\Big)$$

Multilinear Tensor Operations

Given an order d tensor $\boldsymbol{\mathcal{T}}$, define multilinear function $\boldsymbol{x}^{(1)} = \boldsymbol{f}^{(\boldsymbol{\mathcal{T}})}(\boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(d)})$

- *For an order 3 tensor,*

$$x_{i_1}^{(1)} = \sum_{i_2,i_3} t_{i_1 i_2 i_3} x_{i_2}^{(2)} x_{i_3}^{(3)} \Rightarrow \boldsymbol{f}^{(\boldsymbol{\mathcal{T}})}(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) = \boldsymbol{\mathcal{T}} \times_2 \boldsymbol{x}^{(2)} \times_3 \boldsymbol{x}^{(3)} = \boldsymbol{T}_1(\boldsymbol{x}^{(2)} \otimes \boldsymbol{x}^{(3)})$$

- *For an order 2 tensor, we simply have the matrix-vector product $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$*
- *For higher order tensors, we define the function as follows*

$$x_{i_1}^{(1)} = \sum_{i_2 \dots i_d} t_{i_1 \dots i_d} x_{i_2}^{(2)} \cdots x_{i_d}^{(d)} \Rightarrow \boldsymbol{f}^{(\boldsymbol{\mathcal{T}})}(\boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(d)}) = \boldsymbol{\mathcal{T}} \bigtimes_{j=2}^{d} \boldsymbol{x}^{(j)} = \boldsymbol{T}_1 \bigotimes_{j=2}^{d} \boldsymbol{x}^{(j)}$$

- *More generally, we can associate d functions with a $\boldsymbol{\mathcal{T}}$, one for each choice of output mode, for output mode m, we can compute*

$$\boldsymbol{x}^{(m)} = \boldsymbol{T}_{(m)} \bigotimes_{j=1, j \neq m}^{d} \boldsymbol{x}^{(j)}$$

which gives $\boldsymbol{f}_{\tilde{\boldsymbol{\mathcal{T}}}}$ where $\tilde{\boldsymbol{\mathcal{T}}}$ is a transposition of $\boldsymbol{\mathcal{T}}$ defined so that $\tilde{\boldsymbol{T}}_{(1)} = \boldsymbol{T}_{(m)}$

Batched Multilinear Operations

The multilinear map $\boldsymbol{f}^{(\boldsymbol{\mathcal{T}})}$ is frequently used in tensor computations

- *Two common primitives (MTTKRP and TTMc) correspond to sets (batches) of multilinear function evaluations*
- *Given a tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n\times\cdots\times n}$ and matrices $\boldsymbol{U}^{(1)}, \ldots, \boldsymbol{U}^{(d)} \in \mathbb{R}^{n\times R}$, the matricized tensor times Khatri-Rao product (MTTKRP) computes*

$$u^{(1)}_{i_1 r} = \sum_{i_2\ldots i_d} t_{i_1\ldots i_d} u^{(2)}_{i_2 r} \cdots u^{(d)}_{i_d r}$$

which we can express columnwise as

$$\boldsymbol{u}^{(1)}_r = \boldsymbol{f}^{(\boldsymbol{\mathcal{T}})}(\boldsymbol{u}^{(2)}_r, \ldots, \boldsymbol{u}^{(d)}_r) = \boldsymbol{\mathcal{T}} \times_2 \boldsymbol{u}^{(2)}_r \cdots \times_d \boldsymbol{u}^{(d)}_r = \boldsymbol{T}_{(1)}(\boldsymbol{u}^{(2)}_r \otimes \cdots \otimes \boldsymbol{u}^{(d)}_r)$$

- *With the same inputs, the tensor-times-matrix chain (TTMc) computes*

$$u^{(1)}_{i_1 r_2\ldots r_d} = \sum_{i_2\ldots i_d} t_{i_1\ldots i_d} u^{(2)}_{i_2 r_2} \cdots u^{(d)}_{i_d r_d}$$

which we can express columnwise as

$$\boldsymbol{u}^{(1)}_{r_2\ldots r_d} = \boldsymbol{f}^{(\boldsymbol{\mathcal{T}})}(\boldsymbol{u}^{(2)}_{r_1}, \ldots, \boldsymbol{u}^{(d)}_{r_d})$$

Tensor Norm and Conditioning of Multilinear Functions

We can define elementwise and operator norms for a tensor $\mathcal{T}$

- *The tensor Frobenius norm generalizes the matrix Frobenius norm*

$$\|\mathcal{T}\|_F = \Big(\sum_{i_1 \dots i_d} |t_{i_1 \dots i_d}|^2\Big)^{1/2} = \|vec(\mathcal{T})\|_2 = \|\boldsymbol{T}_{(m)}\|_F$$

- *Denoting $\mathbb{S}^{n-1} \subset \mathbb{R}^n$ as the unit sphere (set of vectors with norm one), we define the tensor operator (spectral) norm to generalize the matrix 2-norm as*

$$\begin{aligned}
\|\mathcal{T}\|_2^2 &= \sup_{\boldsymbol{x}^{(1)},\dots,\boldsymbol{x}^{(d)} \in \mathbb{S}^{n-1}} \sum_{i_1 \dots i_d} t_{i_1 \dots i_d} x_{i_1}^{(1)} \cdots x_{i_d}^{(d)} \\
&= \sup_{\boldsymbol{x}^{(1)},\dots,\boldsymbol{x}^{(d)} \in \mathbb{S}^{n-1}} \langle \boldsymbol{x}^{(1)}, \boldsymbol{f}^{(\mathcal{T})}(\boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(d)}) \rangle \\
&= \sup_{\boldsymbol{x}^{(2)},\dots,\boldsymbol{x}^{(d)} \in \mathbb{S}^{n-1}} \|\boldsymbol{f}^{(\mathcal{T})}(\boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(d)})\|_2^2
\end{aligned}$$

- *These norms satisfy the following inequalities*

$$\max_{i_1 \dots i_d} |t_{i_1 \dots i_d}| \leqslant \|\mathcal{T}\|_2 \leqslant \|\mathcal{T}\|_F \quad \textit{and} \quad \|\mathcal{T} \times_m \boldsymbol{M}\|_2 \leqslant \|\mathcal{T}\|_2 \|\boldsymbol{M}\|_2$$

Conditioning of Multilinear Functions

Evaluation of the multilinear map is typically ill-posed for worst case inputs

- *The conditioning of evaluating* $\boldsymbol{f}^{(\boldsymbol{\mathcal{T}})}(\boldsymbol{x}^{(2)}, \dots \boldsymbol{x}^{(d)})$ *with* $\boldsymbol{x}^{(2)}, \dots \boldsymbol{x}^{(d)} \in \mathbb{S}^{n-1}$ *with respect to perturbation in a variable* $\boldsymbol{x}^{(m)}$ *for any* $m \geqslant 2$ *is*

$$\kappa_{\boldsymbol{f}^{(\boldsymbol{\mathcal{T}})}}(\boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(d)}) = \frac{\|\boldsymbol{J}^{(m)}_{\boldsymbol{f}^{(\boldsymbol{\mathcal{T}})}}(\boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(d)})\|_2}{\|\boldsymbol{f}^{(\boldsymbol{\mathcal{T}})}(\boldsymbol{x}^{(2)}, \dots \boldsymbol{x}^{(d)})\|_2}$$

where $\boldsymbol{G} = \boldsymbol{J}^{(m)}_{\boldsymbol{f}^{(\boldsymbol{\mathcal{T}})}}(\boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(d)})$ *is given by* $g_{ij} = df_i^{(\boldsymbol{\mathcal{T}})}(\boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(d)})/dx_j^{(m)}$

- *If we wish to associate a single condition number with a tensor, can tightly bound numerator*

$$\|\boldsymbol{J}^{(m)}_{\boldsymbol{f}^{(\boldsymbol{\mathcal{T}})}}(\boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(d)})\|_2 \leqslant \|\boldsymbol{\mathcal{T}}\|_2$$

- *However, the condition number goes to infinity (problem becomes ill-posed) when* $\|\boldsymbol{f}^{(\boldsymbol{\mathcal{T}})}(\boldsymbol{x}^{(2)}, \dots \boldsymbol{x}^{(d)})\|_2 = 0$
- *Consequently, wish to lower bound the denominator in*

$$\kappa_{\boldsymbol{f}^{(\boldsymbol{\mathcal{T}})}} = \|\boldsymbol{\mathcal{T}}\|_2 / \inf_{\boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(d)} \in \mathbb{S}^{n-1}} \|\boldsymbol{f}^{(\boldsymbol{\mathcal{T}})}(\boldsymbol{x}^{(2)}, \dots \boldsymbol{x}^{(d)})\|_2$$

Well-conditioned Tensors

For equidimensional tensors (all modes of same size), some small ideally conditioned tensors exist

- *For order 2 tensors, for any dimension n, there exist $n \times n$ orthogonal matrices with unit condition number*
- *For order 3, there exist tensors $\mathcal{T} \in \mathbb{R}^{n \times n \times n}$ with $n \in \{2, 4, 8\}$, s.t.*

  $$\inf_{\boldsymbol{x}^{(2)},\ldots,\boldsymbol{x}^{(d)} \in \mathbb{S}^{n-1}} \|\boldsymbol{f}^{(\mathcal{T})}(\boldsymbol{x}^{(2)},\ldots \boldsymbol{x}^{(d)})\|_2 = \|\mathcal{T}\|_2 = 1$$

  which correspond to ideally conditioned multilinear maps (generalize orthogonal matrices)
- *For $n = 2$, an example of such a tensor is given by combining the two slices*

  $$\begin{bmatrix} 1 & \\ & 1 \end{bmatrix} \quad \textit{and} \quad \begin{bmatrix} & 1 \\ -1 & \end{bmatrix}$$

  while for $n = 4$, an example is given by combining the 4 slices

  $$\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & -1 \end{bmatrix} \begin{bmatrix} & 1 & & \\ -1 & & & \\ & & & 1 \\ & & 1 & \end{bmatrix} \begin{bmatrix} & & & 1 \\ & & 1 & \\ & -1 & & \\ 1 & & & \end{bmatrix} \begin{bmatrix} & & -1 & \\ & & & 1 \\ 1 & & & \\ & 1 & & \end{bmatrix}$$

Ill-conditioned Tensors

For $n \notin \{2,4,8\}$ given any $\mathcal{T} \in \mathbb{R}^{n\times n\times n}$, $\inf_{\boldsymbol{x},\boldsymbol{y}\in\mathbb{S}^{n-1}} \|\boldsymbol{f}^{(\mathcal{T})}(\boldsymbol{x},\boldsymbol{y})\|_2 = 0$

- *In 1889, Adolf Hurwitz posed the problem of finding bilinear (in $\boldsymbol{x}$ and $\boldsymbol{y}$) forms $z_1,\dots,z_n$, such that for all $(\boldsymbol{x},\boldsymbol{y})$,*

$$(x_1^2+\cdots+x_l^2)(y_1^2+\cdots+y_m^2) = z_1^2+\cdots+z_n^2.$$

- *In 1922, Johann Radon derived results that imply that over the reals, when $l=m=n$, solutions exist only if $n \in \{2,4,8\}$*
- *If for $\mathcal{T}$ and any vectors $\boldsymbol{x}$, $\boldsymbol{y}$,*

$$\frac{\|\mathcal{T}\times_2 \boldsymbol{x}\times_3 \boldsymbol{y}\|_2}{\|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2} = 1 \quad \Rightarrow \quad \|\mathcal{T}\times_2 \boldsymbol{x}\times_3 \boldsymbol{y}\|_2^2 = \|\boldsymbol{x}\|_2^2\|\boldsymbol{y}\|_2^2,$$

  we can define bilinear forms that provide a solution to the Hurwitz problem

$$z_i = \sum_j\sum_k t_{ijk}x_jy_k$$

- *Radon's result immediately implies $\kappa_{\boldsymbol{f}^{(\mathcal{T})}} > 1$ for $n \notin \{2,4,8\}$, while a 1962 result by Frank J. Adams gives $\kappa_{\boldsymbol{f}^{(\mathcal{T})}} = \infty$, as there exists a linear combination of any n real $n\times n$ matrices that is rank-deficient for $n \notin \{2,4,8\}$*

Algebras as Tensors

A third order tensor can be used to describe an algebra

- *An algebra over a field is a* n*-dimensional vector space and a bilinear product* $f(\boldsymbol{u}, \boldsymbol{v})$
- *Any bilinear product defining an algebra corresponds to an* $n \times n \times n$ *tensor* $\boldsymbol{w} = f(\boldsymbol{u}, \boldsymbol{v}) \Rightarrow w_i = \sum_{jk} t_{ijk} u_j v_k$

The Hurwitz problem also implies a result for division algebras, for which the bilinear product is invertible

- *These include the complex numbers, quaternions, and octonions, corresponding to* $n = 2, 4, 8$ *respectively*
- *These algebras may are described by tensors with* $\kappa_{\boldsymbol{f}(\boldsymbol{\mathcal{T}})} = 1$

CP Decomposition

- The *canonical polyadic or CANDECOMP/PARAFAC (CP) decomposition* expresses an order d tensor in terms of d factor matrices
  - *For a tensor $\mathcal{T} \in \mathbb{R}^{n \times n \times n}$, the CP decomposition is defined by matrices $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{W}$ such that*
    $$t_{ijk} = \sum_{r=1}^{R} u_{ir} v_{jr} w_{kr}$$
    the columns of $\boldsymbol{U}$, $\boldsymbol{V}$, and $\boldsymbol{W}$ are generally not orthonormal, but may be normalized, so that
    $$t_{ijk} = \sum_{r=1}^{R} \sigma_r u_{ir} v_{jr} w_{kr}$$
    where each $\sigma_r \geqslant 0$ and $\|\boldsymbol{u}_r\|_2 = \|\boldsymbol{v}_r\|_2 = \|\boldsymbol{w}_r\|_2 = 1$
  - *For an order N tensor, the decomposition generalizes as follows,*
    $$t_{i_1 \dots i_d} = \sum_{r=1}^{R} \prod_{j=1}^{d} u_{i_j r}^{(j)}$$
  - *Its rank is generally bounded by $R \leqslant n^{d-1}$*

CP Decomposition Basics

- The CP decomposition is useful in a variety of contexts
  - *If an exact decomposition with* $R \ll n^{d-1}$ *is expected to exist*
  - *If an approximate decomposition with* $R \ll n^{d-1}$ *is expected to exist*
  - *If the factor matrices from an approximate decomposition with* $R = O(1)$ *are expected to contain information about the tensor data*
  - *CP a widely used tool, appearing in many domains of science and data analysis*
- Basic properties and methods
  - *Uniqueness (modulo normalization) is dependent on rank*
  - *Finding the CP rank of a tensor or computing the CP decomposition is NP-hard (even with* $R = 1$*)*
  - *Typical rank of tensors (likely rank of a random tensor) is generally less than the maximal possible rank*
  - *CP approximation as a nonlinear least squares (NLS) problem and NLS methods can be applied in a black-box fashion, but structure of decomposition motivates alternating least-squares (ALS) optimization*

Tucker Decomposition

- The *Tucker decomposition* expresses an order d tensor via a smaller order d core tensor and d factor matrices
  - *For a tensor* $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n\times n\times n}$*, the Tucker decomposition is defined by core tensor* $\boldsymbol{\mathcal{Z}} \in \mathbb{R}^{R_1\times R_2\times R_3}$ *and factor matrices* $\boldsymbol{U}$*,* $\boldsymbol{V}$*, and* $\boldsymbol{W}$ *with orthonormal columns, such that*

$$t_{ijk} = \sum_{p=1}^{R_1}\sum_{q=1}^{R_2}\sum_{r=1}^{R_3} z_{pqr}u_{ip}v_{jq}w_{kr}$$

  - *For general tensor order, the Tucker decomposition is defined as*

$$t_{i_1\dots i_d} = \sum_{r_1=1}^{R_1}\cdots\sum_{r_d=1}^{R_d} z_{r_1\dots r_d}\prod_{j=1}^{d} u^{(j)}_{i_j r_j}$$

    which can also be expressed as

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{Z}} \times_1 \boldsymbol{U}^{(1)} \cdots \times_d \boldsymbol{U}^{(d)}$$

  - *The Tucker ranks,* (R_1, R_2, R_3) *are each bounded by the respective tensor dimensions, in this case,* $R_1, R_2, R_3 \leqslant n$
  - *In relation to CP, Tucker is formed by taking all combinations of tensor products between columns of factor matrices, while CP takes only disjoint products*

Tucker Decomposition Basics

- The Tucker decomposition is used in many of the same contexts as CP
  - *If an exact decomposition with each $R_j < n$ is expected to exist*
  - *If an approximate decomposition with $R_j < n$ is expected to exist*
  - *If the factor matrices from an approximate decomposition with $R = O(1)$ are expected to contain information about the tensor data*
  - *Tucker is most often used for data compression and appears less often than CP in theoretical analysis*
- Basic properties and methods
  - *The Tucker decomposition is not unique (can pass transformations between core tensor and factor matrices, which also permit their orthogonalization)*
  - *Finding the best Tucker approximation is NP-hard (for $R = 1$, CP = Tucker)*
  - *If an exact decomposition exists, it can be computed by high-order SVD (HOSVD), which performs d SVDs on unfoldings*
  - *HOSVD obtains a good approximation with cost $O(n^{d+1})$ (reducible to $O(n^d R)$ via randomized SVD or QR with column pivoting)*
  - *Accuracy can be improved by iterative nonlinear optimization methods, such as high-order orthogonal iteration (HOOI)*

Tensor Train Decomposition

- The *tensor train decomposition* expresses an order d tensor as a chain of products of order 2 or order 3 tensors
  - *For an order 4 tensor, we can express the tensor train decomposition as*

$$t_{ijkl} = \sum_{p,q,r} u_{ip} v_{pjq} w_{qkr} z_{rl}$$

  - *More generally, the Tucker decomposition is defined as follows,*

$$t_{i_1 \dots i_d} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_{d-1}=1}^{R_{d-1}} u^{(1)}_{i_1 r_1} \left(\prod_{j=2}^{d-1} u^{(j)}_{r_{j-1} i_j r_j} \right) u^{(d)}_{r_{d-1} i_d}$$

  - *In physics literature, it is known as a matrix product state (MPS), as we can write it in the form,*

$$t_{i_1 \dots i_d} = \langle \boldsymbol{u}^{(1)}_{i_1}, \boldsymbol{U}^{(2)}_{i_2} \cdots \boldsymbol{U}^{(d-1)}_{i_{d-1}} \boldsymbol{u}^{(d)}_{i_d} \rangle$$

  - *For an equidimensional tensor, the ranks are bounded as $R_j \leqslant \min(n^j, n^{d-j})$*

Tensor Train Decomposition Basics

- Tensor train has applications in quantum simulation and in numerical PDEs
  - *Its useful whenever the tensor is low-rank or approximately low-rank, i.e., $R_j R_{j+1} < n^{d-1}$ for all $j < d - 1$*
  - *MPS (tensor train) and extensions are widely used to approximate quantum systems with $\Theta(d)$ particles/spins*
  - *Often the MPS is optimized relative to an implicit operator (often of a similar form, referred to as the matrix product operator (MPO))*
  - *Operators and solutions to some standard numerical PDEs admit tensor-train approximations that yield exponential compression*
- Basic properties and methods
  - *The tensor train decomposition is not unique (can pass transformations, permitting orthogonalization into canonical forms)*
  - *Approximation with tensor train is NP hard (for $R = 1$, CP = Tucker = TT)*
  - *If an exact decomposition exists, it can be computed by tensor train SVD (TTSVD), which performs $d - 1$ SVDs*
  - *TTSVD can be done with the cost $O(n^{d+1})$ or $O(n^d R)$ with faster low-rank SVD*
  - *Iterative (alternating) optimization is generally used when optimizing tensor train relative to an implicit operator or to refine TTSVD*

Summary of Tensor Decomposition Basics

We can compare the aforementioned decomposition for an order d tensor with all dimensions equal to n and all decomposition ranks equal to R

decomposition	CP	Tucker	tensor train
size	dnR	$dnR + R^d$	$2nR + (d-2)nR^2$
uniqueness	if $R \leqslant (3n-2)/2$	no	no
orthogonalizability	none	partial	partial
exact decomposition	NP hard	$O(n^{d+1})$	$O(n^{d+1})$
approximation	NP hard	NP hard	NP hard
typical method	ALS	HOSVD	TT-ALS (implicit)