CS 598 EVS: Tensor Computations

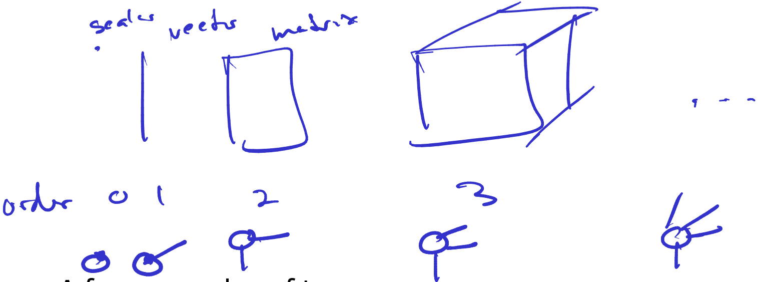
Basics of Tensor Computations

Edgar Solomonik

University of Illinois at Urbana-Champaign

Tensors

A *tensor* is a collection of elements



A few examples of tensors are

Reshaping Tensors

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

$$\mathrm{vec}(A) = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$$

$$A = \begin{bmatrix} a_1 & \cdots & a_n \end{bmatrix}$$

folding → increases order

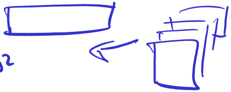
unfolding

decrease the order

matrix → vector

3rd order tensor → matrix

matricization

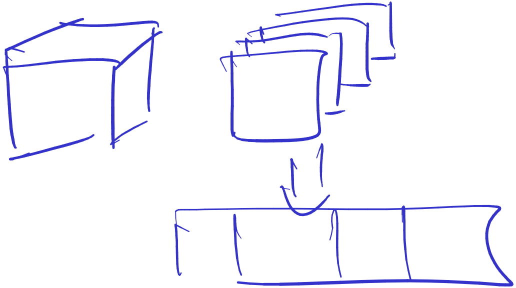


$$m_{ij} = t_{i j_1 j_2}$$

$$j = (j_1 - 1)\cdot n + j_2$$

$$T \in R^{m\times n\times k} \quad T_{(1)} \in R^{m\times(nk)}$$

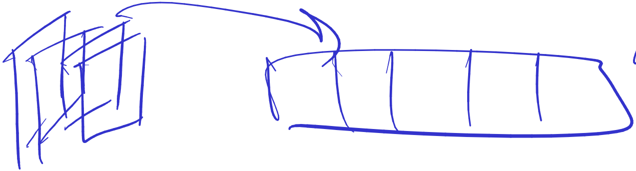
T $\quad T_{(1)}$



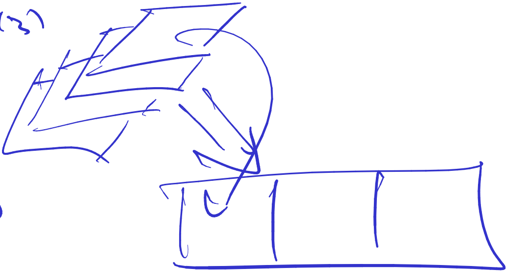
$T_{(2)} \in R^{n \times (mk)}$

$T_{(3)} \in R^{k \times (mn)}$

$T_{(2)}$



$T_{(3)}$



Matrices and Tensors as Operators and Multilinear Forms

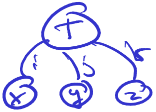
- What is a matrix?

$\mathbb{R}^m \to \mathbb{R}^n$

2D array $\quad A \quad$ $f_A(x) = Ax$

$\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$

$f_A(x, y) = x^T A y$



- What is a tensor?

N-D array $\quad$ $f_T(x, y, z) = \sum_{ijk} t_{ijk} x_i y_j z_k$

$f_T(y, z) = \sum_{jk} t_{ijk} y_j z_k = x_i$

Tensor Transposition

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

$$u_{ijk} = t_{kji}$$

$3! = 6$ permutations

$$w_{il} = \sum_{jk} t_{ijk} u_{jlk}$$

contraction

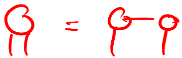
transposition ↓ $v_{ljk} = u_{jlk}$

$$w_{il} = \sum_{jk} t_{ijk} v_{ljk}$$

unfolding ↓

$$W = T_{(1)} \cdot V_{(1)}^T$$

matrix mult.



Tensor Symmetry

$A = A^T$

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

$$t_{i_1 \dots i_j \dots i_k \dots i_N} = t_{i_1 \dots i_k \dots i_j \dots i_N}$$

$$t_{ijk} = t_{jik} = t_{kij} = t_{kji} = t_{kji}$$

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

$A = -A^T$

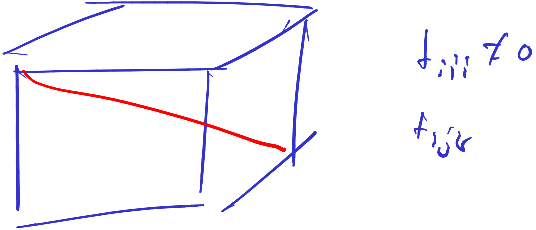
$$t_{i_1 \dots i_j \dots i_k \dots i_N} = -t_{i_1 \dots i_k \dots i_j \dots i_N}$$

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

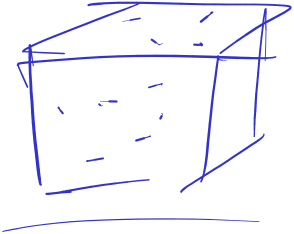
common in physics/chemistry

Tensor Sparsity

We say a tensor $\mathcal{T}$ is *diagonal* if for some v, If most of the tensor entries are



zeros, the tensor is *sparse*



Tensor Products and Kronecker Products

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

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

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

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		
outer product		
pointwise product		
Hadamard product		
matrix multiplication		
batched mat.-mul.		
tensor times matrix		

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

Unfolding the tensors reduces the tensor contraction to matrix multiplication

Properties of Einsums

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

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

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

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

Identities with Kronecker and Khatri-Rao Products

- Matrix multiplication is distributive over the Kronecker product

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

Multilinear Tensor Operations

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

Batched Multilinear Operations

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

Tensor Norm and Conditioning of Multilinear Functions

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

Conditioning of Multilinear Functions

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

Well-conditioned Tensors

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

Ill-conditioned Tensors

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

CP Decomposition

- The *canonical polyadic or CANDECOMP/PARAFAC (CP) decomposition* expresses an order d tensor in terms of d factor matrices

CP Decomposition Basics

- The CP decomposition is useful in a variety of contexts

- Basic properties and methods

Tucker Decomposition

- The *Tucker decomposition* expresses an order d tensor via a smaller order d core tensor and d factor matrices

Tucker Decomposition Basics

- The Tucker decomposition is used in many of the same contexts as CP
- Basic properties and methods

Tensor Train Decomposition

- The *tensor train decomposition* expresses an order d tensor as a chain of products of order 2 or order 3 tensors

Tensor Train Decomposition Basics

- Tensor train has applications in quantum simulation and in numerical PDEs

- Basic properties and methods

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			
uniqueness			
orthogonalizability			
exact decomposition			
approximation			
typical method			

Bilinear Algorithms

A bilinear algorithm (V. Pan, 1984) $\Lambda = (\boldsymbol{F}^{(A)}, \boldsymbol{F}^{(B)}, \boldsymbol{F}^{(C)})$ computes

$$\boldsymbol{c} = \boldsymbol{F}^{(C)}[(\boldsymbol{F}^{(A)T}\boldsymbol{a}) \odot (\boldsymbol{F}^{(B)T}\boldsymbol{b})],$$

where $\boldsymbol{a}$ and $\boldsymbol{b}$ are inputs and $\odot$ is the Hadamard (pointwise) product.

Bilinear Algorithms as Tensor Factorizations

- A bilinear algorithm corresponds to a CP tensor decomposition

- For multiplication of $n \times n$ matrices, we can define a *matrix multiplication tensor* and consider algorithms with various bilinear rank

Strassen's Algorithm

Strassen's algorithm $\begin{bmatrix} \boldsymbol{C}_{11} & \boldsymbol{C}_{12} \\ \boldsymbol{C}_{21} & \boldsymbol{C}_{22} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{B}_{11} & \boldsymbol{B}_{12} \\ \boldsymbol{B}_{21} & \boldsymbol{B}_{22} \end{bmatrix}$

$$
\begin{aligned}
\boldsymbol{M}_1 &= (\boldsymbol{A}_{11} + \boldsymbol{A}_{22}) \cdot (\boldsymbol{B}_{11} + \boldsymbol{B}_{22}) \\
\boldsymbol{M}_2 &= (\boldsymbol{A}_{21} + \boldsymbol{A}_{22}) \cdot \boldsymbol{B}_{11} \\
\boldsymbol{M}_3 &= \boldsymbol{A}_{11} \cdot (\boldsymbol{B}_{12} - \boldsymbol{B}_{22}) \\
\boldsymbol{M}_4 &= \boldsymbol{A}_{22} \cdot (\boldsymbol{B}_{21} - \boldsymbol{B}_{11}) \\
\boldsymbol{M}_5 &= (\boldsymbol{A}_{11} + \boldsymbol{A}_{12}) \cdot \boldsymbol{B}_{22} \\
\boldsymbol{M}_6 &= (\boldsymbol{A}_{21} - \boldsymbol{A}_{11}) \cdot (\boldsymbol{B}_{11} + \boldsymbol{B}_{12}) \\
\boldsymbol{M}_7 &= (\boldsymbol{A}_{12} - \boldsymbol{A}_{22}) \cdot (\boldsymbol{B}_{21} + \boldsymbol{B}_{22})
\end{aligned}
$$

$$
\begin{aligned}
\boldsymbol{C}_{11} &= \boldsymbol{M}_1 + \boldsymbol{M}_4 - \boldsymbol{M}_5 + \boldsymbol{M}_7 \\
\boldsymbol{C}_{21} &= \boldsymbol{M}_2 + \boldsymbol{M}_4 \\
\boldsymbol{C}_{12} &= \boldsymbol{M}_3 + \boldsymbol{M}_5 \\
\boldsymbol{C}_{22} &= \boldsymbol{M}_1 - \boldsymbol{M}_2 + \boldsymbol{M}_3 + \boldsymbol{M}_6
\end{aligned}
$$

By performing the nested calls recursively, Strassen's algorithm achieves cost,