# CS 598 EVS: Tensor Computations

## Course Overview

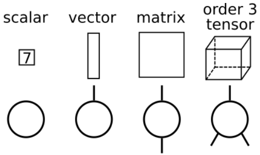
Edgar Solomonik

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 $i \in \{1, \dots, s_1\}, j \in \{1, \dots, s_2\}, k \in \{1, \dots, s_3\}$

# 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 $\boldsymbol{v} = \text{vec}(\boldsymbol{A})$, which stacks the columns of $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ to produce $\boldsymbol{v} \in \mathbb{R}^{mn}$
- For a tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{s_1 \times s_2 \times s_3}$, $\boldsymbol{v} = \text{vec}(\boldsymbol{\mathcal{T}})$ 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,

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

# Tensor Contractions

A *tensor contraction* multiplies two tensors to produce a third

- Examples: inner product, outer product, tensor product, Hadamard (elementwise) product, matrix multiplication
- One higher order example is tensor-times-matrix (TTM), e.g.,

$$t_{ijkl} = \sum_{q} u_{ijql} v_{qk}$$

- A common contraction between two high order tensors is

$$t_{abij} = \sum_{p,q} u_{apiq} v_{pbqj}$$

- Tensor contractions can be reduced to products of matrices and/or vectors by transposing modes and matricizing both operands, then folding and transposing the product

# Tensor Contraction Expressions

In most applications, we wish to evaluate mathematical expressions involving contraction of more than two tensors

- Contractions of more than two tensors are also sometimes referred to as *einsums* (short for Einstein summation, in reference to Einstein's convention for omitting summation indices)
- One important example is the 'MTTKRP' kernel

$$u_{ir} = \sum_{j,k} t_{ijk} v_{jr} w_{kr}$$

- Efficient evaluation of such kernels requires specialized algorithms
- Contraction algorithms must also be adapted to leverage tensor properties such as symmetry with respect to permutation of modes, block-wise group symmetries, and data sparsity

# Tensor Decompositions

Tensor decompositions express a tensor as a contraction of *factors*

- Canonical polyadic (CP) decomposition, factors are three matrices:

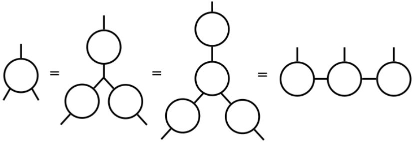
$$t_{ijk} = \sum_{r=1}^{R} u_{ir} v_{jr} w_{kr}$$

- Tucker decomposition, factors are three orthogonal matrices and a core tensor:

$$t_{ijk} = \sum_{p,q,r} u_{ip} v_{jq} w_{kr} z_{pqr}$$

- Tensor train decomposition, factors are matrices or order 3 tensors:

$$t_{i_1 i_2 i_3 i_4} = \sum_{j_1, j_2, j_3} u_{i_1 j_1} v_{j_1 i_2 j_2} w_{j_2 i_3 j_3} z_{j_3 i_4}$$



# Applications of Tensor Decompositions

- Tensor decompositions provide a mechanism for approximating tensor datasets with a smaller number of degrees of freedom
  - polynomial improvements are obtained in electronic structure calculations
  - exponential improvements are obtained for representing some quantum states
- With imposition of constraints (e.g., nonnegativity or orthogonality), they can be used for data mining tasks such as high-order clustering
  - in the presence of missing data, tensor decompositions may be used to perform tensor completion
- When the tensor represents an operator or mapping, tensor decompositions can be used to find reduced structure
  - fast algorithms, such as FFT and Strassen's matrix multiplication algorithm, may be viewed as tensor decompositions

# Tensor Decomposition Theory

- Many basic decomposition/approximation problems are formally NP-hard
- A considerable amount of theory focuses on CP decomposition and CP rank, some will be surveyed in this course
- A few alternate notions of tensor eigenvalues and singular values exist, and may be loosely tied to decompositions
- Stability and conditioning results exist for the tensor as an operator and CP decomposition as a problem

# Tensor Decomposition Algorithms

- Approximation with tensor decomposition is generally formulated as a nonlinear least squares (NLS) problem
- Optimization methods usually involve successive quadratic approximation (Newton-based methods) as opposed to gradient-based methods
- Alternating least squares (ALS) decouples nonlinear problem into subproblems on subsets of variables that are quadratic and solves each in an alternating manner
- Other optimization methods, such as interior point and ADMM, are often employed in the presence of constraints
- In all cases, methods are specialized to work efficiently for tensor decompositions and may be adapted for sparsity

# Tensor Networks

- Tensor network methods take as input a tensor that is already decomposed
- Goal is generally to learn something about an operator described by a tensor network
- Often want to compute extremal eigenpairs of matrix $\boldsymbol{M}$ a tensor folding of which $\boldsymbol{\mathcal{T}}$ is described by the tensor network, e.g.,

$$\boldsymbol{M} = \boldsymbol{A} \otimes \boldsymbol{B} + \boldsymbol{C} \otimes \boldsymbol{D}$$

- Unknowns, e.g., eigenvectors in eigenproblem above, often also represented implicitly by a tensor decomposition
- These methods are prevalent for studying quantum systems, which involve a Hamiltonian acting on a space that is exponential in size relative to the system
- In this context, tensor networks are also effective for time-evolution

# Tensor Network Theory and Algorithms

- Different classes of functions have low rank with respect to different tensor networks
- 1D and 2D tensor networks are most widely used for quantum systems
- Successive (alternating) quadratic optimization also widely used for tensor networks
- *Canonical forms* propagate orthogonality conditions to ensure stability
- Naive contraction of 2D tensor networks has exponential cost, various approximate algorithms exist
- Other tensor networks trade-off connectivity and contractibility