Tensor Decompositions and their Applications in Machine Learning

Seminar: Efficient Inference and Large-Scale Machine Learning

Stephan Rabanser

Technische Universität München
Department of Informatics
Data Mining and Analytics
http://kdd.in.tum.de

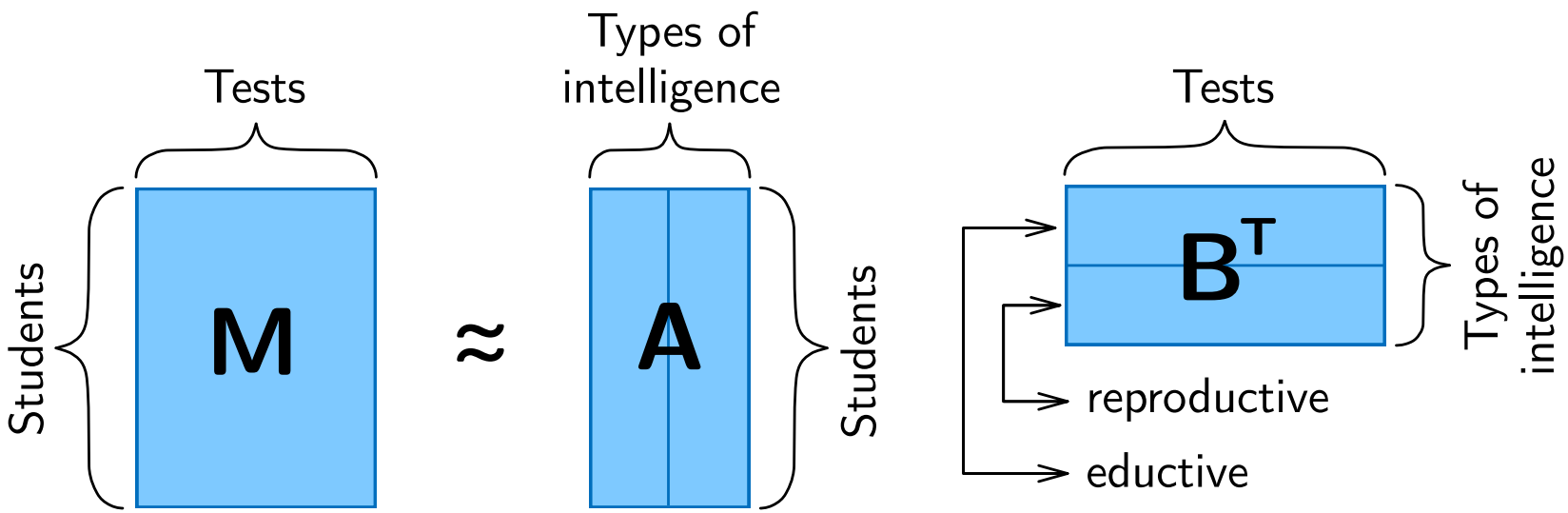
May 7, 2017

Table of contents

Matrix Decomposition: A Motivating Example

Spearman's Hypothesis

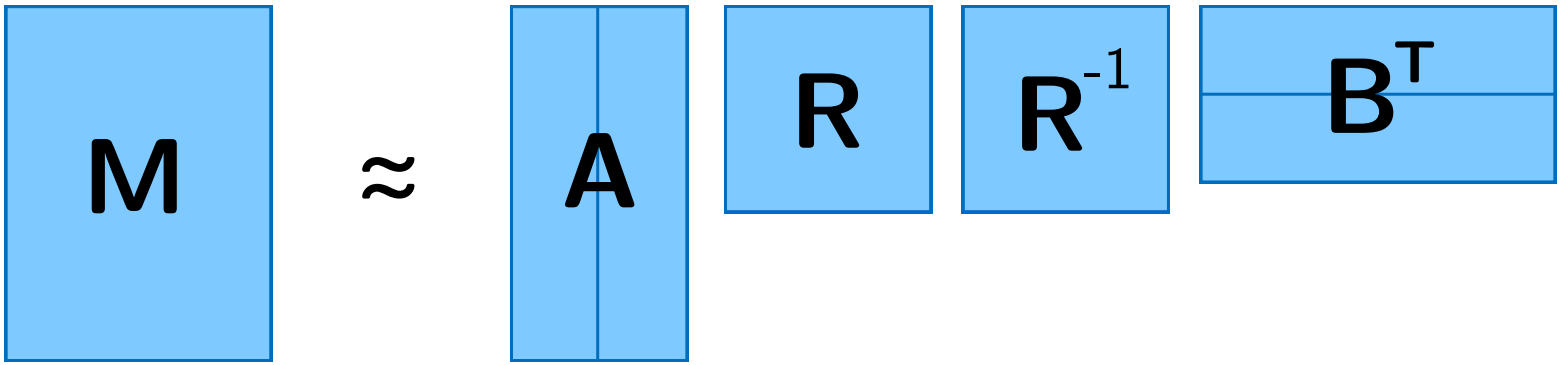
Charles Spearman, a British psychologist, supposed that human intelligence can be broken down into two factors: **eductive**[1] and **reproductive**[2] intelligence.
He invented what we refer to as **factor analysis**!



Can we recover $\boldsymbol{A}$ and $\boldsymbol{B}$ uniquely from $\boldsymbol{M}$? In general, no!

[1] eductive = the ability to make sense out of complexity
[2] reproductive = the ability to store and reproduce information

Rotation Problem



$$\boldsymbol{M} = \boldsymbol{A}\boldsymbol{B}^T = \boldsymbol{A}\boldsymbol{R}\boldsymbol{R}^{-1}\boldsymbol{B}^T$$
$$= (\boldsymbol{A}\boldsymbol{R})(\boldsymbol{R}^{-1}\boldsymbol{B}^T) = (\boldsymbol{A}\boldsymbol{R})(\boldsymbol{B}\boldsymbol{R}^{-T})^T = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}}^T$$

$\boldsymbol{A}$ and $\boldsymbol{B}$ are not determined uniquely unless we impose additional conditions on them (e.g. SVD relies on orthogonality constraints).

Can we achieve weaker uniqueness conditions when moving to higher-dimensional structures (= tensors)? Yes!

Tensor Basics

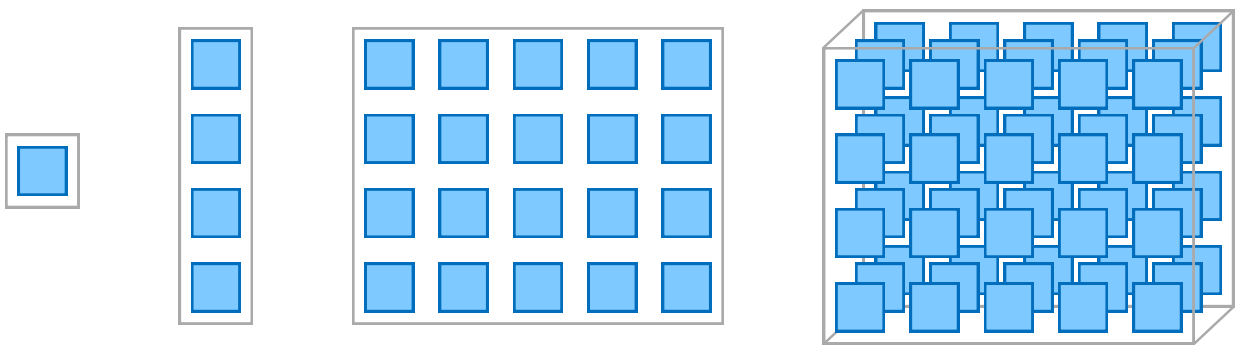
Tensors and Tensor Order

Tensors are generalizations of matrices to higher dimensions and can therefore be seen as multidimensional fields.

The *order*/*way*/*mode* of a tensor is the number of its dimensions.

- Zeroth order tensors (scalars): $x \in \mathbb{R}$
- First order tensors (vectors): $\boldsymbol{x} \in \mathbb{R}^{I_1}$
- Second order tensors (matrices): $\boldsymbol{X} \in \mathbb{R}^{I_1 \times I_2}$
- Third or higher order tensors: $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$

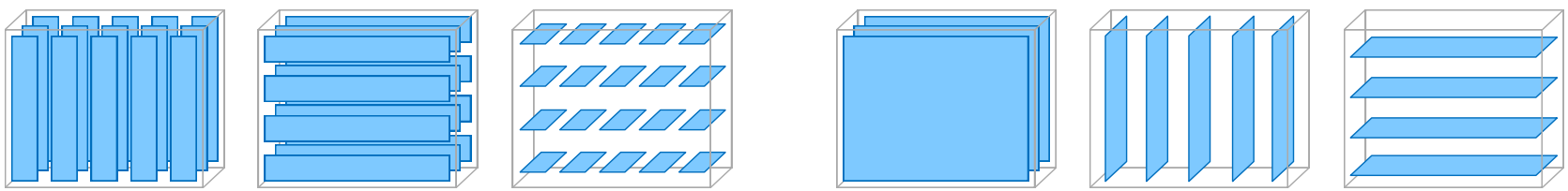
We can depict zeroth order tensors to third order tensors as follows:



Tensor Fibers and Slices

Fibers are created when fixing all but one index, *slices* (or *slabs*) are created when fixing all but two indices.
For a third order tensor

- fibers are given as $\boldsymbol{x}_{:jk} (= \boldsymbol{x}_{jk})$ (column), $\boldsymbol{x}_{i:k}$ (row), and $\boldsymbol{x}_{ij:}$ (tube)
- slices are given as $\boldsymbol{X}_{::k} (= \boldsymbol{X}_k)$ (frontal), $\boldsymbol{X}_{:j:}$ (lateral), and $\boldsymbol{X}_{i::}$ (horizontal)



Vector Outer and Inner Product

Outer Product (Tensor Product)

The outer vector of two n-sized vectors $\boldsymbol{a}$, $\boldsymbol{b}$ is defined as follows:

$$\boldsymbol{a} \circledcirc \boldsymbol{b} = \boldsymbol{a}\boldsymbol{b}^T$$

The outer product between two vectors produces a matrix.

Inner Product

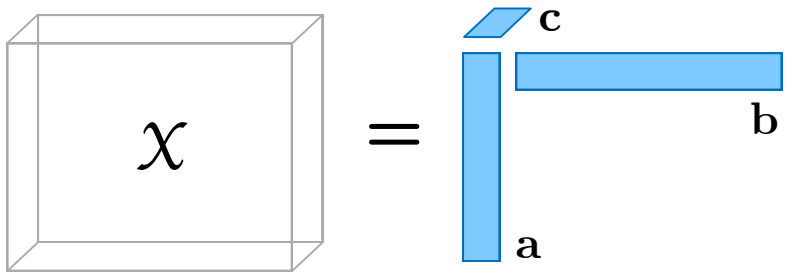
The inner product of two n-sized vectors $\boldsymbol{a}$, $\boldsymbol{b}$ is defined as follows:

$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \boldsymbol{a}^T \boldsymbol{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

The inner product between two vectors produces a scalar.

Rank-One Tensors

A N-way tensor is of *rank-one* if it can be strictly decomposed into the (vector) outer product of N vectors.



$$\mathcal{X} = \boldsymbol{a}^{(1)} \circledcirc \boldsymbol{a}^{(2)} \circledcirc \cdots \circledcirc \boldsymbol{a}^{(N)} \text{ with } x_{i_1 i_2 \cdots i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2} \cdots a^{(N)}_{i_N}$$

Examples

- Rank-one matrix: $\boldsymbol{X} = \boldsymbol{a} \circledcirc \boldsymbol{b}$
- Rank-one 3-way tensor: $\mathcal{X} = \boldsymbol{a} \circledcirc \boldsymbol{b} \circledcirc \boldsymbol{c}$

Tensor Rank

The *rank* of a tensor $\text{rank}(\mathcal{X}) = R$ is defined as the minimum number of rank-one tensors which are needed to produce $\mathcal{X}$.

$$\mathcal{X} = \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r^{(1)} \circledcirc \boldsymbol{a}_r^{(2)} \circledcirc \cdots \circledcirc \boldsymbol{a}_r^{(n)} = [\![\boldsymbol{\lambda}; \boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \cdots, \boldsymbol{A}^{(n)}]\!]$$

A specific $\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_R \end{bmatrix}$ is called *factor matrix* and holds the combination of the vectors from the rank-one components as columns. The additional λ_r term is used to absorb the respective weights during normalization of the factor matrices' columns.

Examples

- Rank-R matrix: $\boldsymbol{X} = \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r \circledcirc \boldsymbol{b}_r = [\![\boldsymbol{\lambda}; \boldsymbol{A}, \boldsymbol{B}]\!]$
- Rank-R 3-way tensor: $\mathcal{X} = \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r \circledcirc \boldsymbol{b}_r \circledcirc \boldsymbol{c}_r$

Tensor Rigidness

Slices of a low-rank tensor are themselves low-rank matrices.
A low-rank tensor is not just a collection of low-rank matrices, but there exist relationships between these slices. In fact, the slices are different scalings of the same set of rank-1 matrices.

$$\mathcal{X}_k = \sum_{r=1}^{R} \left(\boldsymbol{a}_r \circledcirc \boldsymbol{b}_r \right) c_{kr}$$

For true underlying factors, subtracting off scalings of the same rank-1 matrix has to decrease the rank of each slice.

$$\mathcal{X}_k - c_k(\boldsymbol{a} \circledcirc \boldsymbol{b})$$

This makes tensors way more rigid than matrices!

Multilinear Tensor Transformation

A tensor $\mathcal{X}$ can be transformed on multiple dimensions by hitting each of the dimensions with a (potentially different) transformation tensor from the left side.

$$\tilde{\mathcal{X}} = \mathcal{X}(\mathcal{B}, \mathcal{C}, \mathcal{D}) = \sum_{r=1}^{R} \lambda_r \langle \mathcal{B} a_r \rangle \langle \mathcal{C} b_r \rangle \langle \mathcal{D} c_r \rangle$$

Hitting a tensor $\mathcal{X}$ with another tensor $\hat{\mathcal{X}}$ on a specific dimension corresponds to an inner product between $\hat{\mathcal{X}}$ and the vector on the corresponding dimension.

Tensor Applications in Machine Learning

Tensor Applications in Machine Learning

- Unsupervised learning settings
  - Topic and mixture modelling
  - Dimensionality reduction
  - Source separation and clustering
- More natural representation for some data
  - Timed data as tensor slices
- Pattern and anomaly detection in graphs
  - Intrusion detection
  - Social network analysis

Case Study: Learning Mixture of Gaussians

Typical problems in unsupervised learning settings

- **Conditions for Identifiability**: We have to determine whether a proposed model contains all relevant information for parameter derivation.
- **Efficient Learning of Latent Variable Models**: We have to determine whether we can estimate the parameters for the proposed model in an efficient (and hopefully optimal) way.

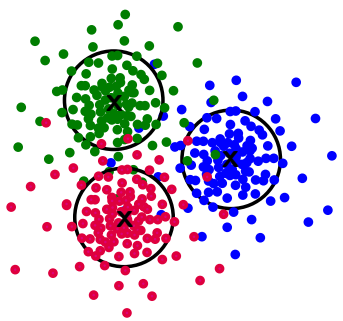
The following approach

- allows us to extract all important information for model construction
- enables us to learn gaussian mixture models efficiently

Learning Mixture of Gaussians (cont.)

Gaussian Mixture Model notation

- k: number of mixture components (latent factors).
- $\boldsymbol{h} \in [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k]$: categorical hidden variable represented in basis vector form (= hot-one encoding). $\mathbb{E}[\boldsymbol{h}] = \boldsymbol{w}$.
- $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_k]$: means of the mixture components.
- $\boldsymbol{\mu}_C = \boldsymbol{0}$ (zero mean) and $\boldsymbol{\Sigma}_C = \sigma^2 \boldsymbol{I}$ (spherical covariance).
- Sample generation: $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{h} + \boldsymbol{z}$ with $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{\mu}_C, \boldsymbol{\Sigma}_C)$.



Learning Mixture of Gaussians (cont.)

Method of Moments

- Alternative to maximum likelihood estimation.
- Extract model information by building up higher order moments of the underlying probability distribution.

In the GMM case, we can compute the first three moments as follows:

- $\boldsymbol{\mu} = \mathbb{E}[\boldsymbol{x}] = \boldsymbol{A}\boldsymbol{w} = \sum_{i \in [k]} w_i \boldsymbol{a}_i$
- $\boldsymbol{\Sigma} = \mathbb{E}[\boldsymbol{x} \otimes \boldsymbol{x}] = \ldots = \sum_{i \in [k]} w_i \boldsymbol{a}_i \otimes \boldsymbol{a}_i + \sigma^2 \boldsymbol{I}$
- $\boldsymbol{\mathcal{T}} = \mathbb{E}[\boldsymbol{x} \otimes \boldsymbol{x} \otimes \boldsymbol{x}] = \ldots = \sum_{i \in [k]} w_i \boldsymbol{a}_i \otimes \boldsymbol{a}_i \otimes \boldsymbol{a}_i + \sigma^2 \sum_{i \in [k]} (\boldsymbol{\mu} \otimes \boldsymbol{e}_i \otimes \boldsymbol{e}_i + \boldsymbol{e}_i \otimes \boldsymbol{\mu} \otimes \boldsymbol{e}_i + \boldsymbol{e}_i \otimes \boldsymbol{e}_i \otimes \boldsymbol{\mu})$

Learning Mixture of Gaussians (cont.)

We observe a similar structure in these moments:

- $\boldsymbol{\mu} = \sum_{i \in [k]} w_i \boldsymbol{a}_i$
- $\boldsymbol{\Sigma} = \sum_{i \in [k]} w_i \boldsymbol{a}_i \otimes \boldsymbol{a}_i + \sigma^2 \boldsymbol{I}$
- $\mathcal{T} = \sum_{i \in [k]} w_i \boldsymbol{a}_i \otimes \boldsymbol{a}_i \otimes \boldsymbol{a}_i + \sigma^2 \sum_{i \in [d]} (\boldsymbol{\mu} \otimes \boldsymbol{e}_i \otimes \boldsymbol{e}_i + \boldsymbol{e}_i \otimes \boldsymbol{\mu} \otimes \boldsymbol{e}_i + \boldsymbol{e}_i \otimes \boldsymbol{e}_i \otimes \boldsymbol{\mu})$

σ^2 is estimated by the smallest eigenvalue of the second moment $\boldsymbol{\Sigma}$. We can therefore reduce the moment computation to:

- $\boldsymbol{m}_1 = \sum_{i \in [k]} w_i \boldsymbol{a}_i$
- $\boldsymbol{M}_2 = \sum_{i \in [k]} w_i \boldsymbol{a}_i \otimes \boldsymbol{a}_i$
- $\mathcal{M}_3 = \sum_{i \in [k]} w_i \boldsymbol{a}_i \otimes \boldsymbol{a}_i \otimes \boldsymbol{a}_i$

Many latent variable models can be reduced to this form!

Learning Mixture of Gaussians (cont.)

If we now assume $\boldsymbol{A}$ to be (columnwise) orthogonal, then the $\boldsymbol{a}_i$'s are the eigenvectors of the third-order moment tensor $\mathcal{M}_3$. If we knew for example $\boldsymbol{a}_1$, then hitting $\mathcal{M}_3$ with $\boldsymbol{a}_1$ on two dimensions yields:

$$\mathcal{M}_3(\boldsymbol{I}, \boldsymbol{a}_1, \boldsymbol{a}_1) = \sum_{i \in [k]} w_i \langle \boldsymbol{a}_i, \boldsymbol{a}_1 \rangle^2 \boldsymbol{a}_i = w_1 \boldsymbol{a}_1$$

This directly corresponds to the concept of matrix eigenvectors! However, we are still facing two problems at this point:

- Can we transform any given tensor to an orthogonal tensor?
- How do we find the eigenvectors of an orthogonal tensor?

Learning Mixture of Gaussians (cont.)

Orthogonalization Through Whitening

Using the second order moment, we can obtain a whitening transformation, that orthogonalizes $\boldsymbol{A}$. The transformation has to be invertible, which is why we want $\boldsymbol{A}$ to have full column rank.

$$\boldsymbol{M}_2 = \boldsymbol{U}\text{Diag}(\tilde{\lambda})\boldsymbol{U}^T \;\Rightarrow\; \boldsymbol{W} = \boldsymbol{U}\text{Diag}(\tilde{\lambda}^{-1/2})$$

The orthogonalized moment $\mathcal{V}$ can then be computed as follows:

$$\begin{aligned}\mathcal{V} = \mathfrak{M}_3(\boldsymbol{W}, \boldsymbol{W}, \boldsymbol{W}) &= \sum_{i\in[k]} \frac{1}{\sqrt{w_i}}(\boldsymbol{W}^T\boldsymbol{a}_i\sqrt{w_i})^{\otimes 3} \\ &= \sum_{i\in[k]} \lambda_i \boldsymbol{v}_i \otimes \boldsymbol{v}_i \otimes \boldsymbol{v}_i\end{aligned}$$

Learning Mixture of Gaussians (cont.)

Tensor Decomposition Through Power Method

Now we can find the eigenvectors of the orthogonalized tensor $\mathcal{V}$ by applying the tensor power method, a natural extension of the matrix power method. After randomly initializing a vector $\boldsymbol{v}$ we can feed the vector into a tensor power step and repeat this step until convergence:

$$\boldsymbol{v} \mapsto \frac{\mathcal{V}(\boldsymbol{I}, \boldsymbol{v}, \boldsymbol{v})}{||\mathcal{V}(\boldsymbol{I}, \boldsymbol{v}, \boldsymbol{v})||}$$

Afterwards we deflate the tensor, i.e. remove eigenvalue λ with eigenvector $\boldsymbol{v}$ that we just extracted from the tensor as follows:

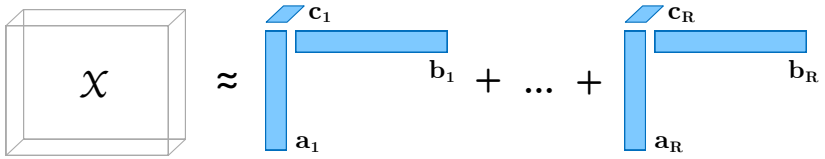
$$\mathcal{V}_{\text{next}} = \mathcal{V} - \lambda \boldsymbol{v} \circledcirc \boldsymbol{v} \circledcirc \boldsymbol{v}$$

This process can be repeated until we have extracted the k dominant eigenvalues/-vectors.

Tensor Decomposition Algorithms

Canonical Polyadic Decomposition (CPD)

The Canonical Polyadic Decomposition (CPD) aims at approximating a tensor $\mathcal{X}$ as good as possible with a low-rank tensor $\hat{\mathcal{X}}$.



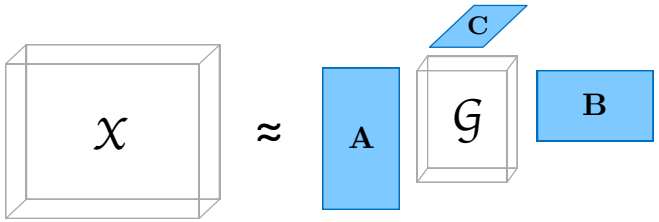
$$\min_{\hat{\mathcal{X}}} ||\mathcal{X} - \hat{\mathcal{X}}|| \quad \text{with} \quad \hat{\mathcal{X}} = \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r \circledcirc \boldsymbol{b}_r \circledcirc \boldsymbol{c}_r = [\![\boldsymbol{\lambda}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$$

Compute $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ through:

- **Jennrich's algorithm**: eigendecomposition of tensor slices
- **Alternating least squares (ALS) algorithm**: fix all factor matrices except for one in order to optimize for the non-fixed matrix

Tucker Decomposition

The Tucker decomposition can be thought of as a higher-order PCA. $\mathcal{G} \in \mathbb{R}^{P \times Q \times R}$ is called core tensor.



$$\min_{\hat{\mathcal{X}}} ||\mathcal{X} - \hat{\mathcal{X}}|| \quad \text{with} \quad \hat{\mathcal{X}} = \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{pqr} \boldsymbol{a}_r \circledcirc \boldsymbol{b}_r \circledcirc \boldsymbol{c}_r$$
$$= [\![\mathcal{G}; \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]\!]$$

Compute $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ through:

- **Higher order singular value decomposition (HOSVD)**
- **Higher order orthogonal iteration (HOOI)**

Thanks! :)

Backup

Tensor Reorderings

We can *vectorize* a given matrix $\boldsymbol{X} \in \mathbb{R}^{I \times J}$ by vertically stacking the columns of $\boldsymbol{X}$ into a tall vector.

$$\text{vec}(\boldsymbol{X}) = \begin{bmatrix} \boldsymbol{x}_{:1} \\ \boldsymbol{x}_{:2} \\ \vdots \\ \boldsymbol{x}_{:J} \end{bmatrix}$$

The mode-n matricization of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ (denoted $\boldsymbol{X}_{(n)}$) turns the mode-n fibers of $\mathcal{X}$ into the columns of $\boldsymbol{X}_{(n)}$. Let $x \in \mathcal{X}$ be an element of a tensor and $m \in \boldsymbol{M}$ be an element of the unfolded tensor. Then the mode-n matricization is defined as:

$$x_{i_1, i_2, \ldots, i_N} \mapsto m_{i_n, j} \text{ with } j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^{N} \left((i_k - 1) \prod_{\substack{m=1 \\ m \neq n}}^{k-1} I_m \right)$$

Tensor Reorderings (cont.)

Example

Let $\mathcal{X}$ be a tensor with the following frontal slices:

$$\boldsymbol{X}_1 = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix} \qquad \boldsymbol{X}_2 = \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix}$$

Then the three mode-n matricizations are:

$$\boldsymbol{X}_{(1)} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix} \qquad \boldsymbol{X}_{(2)} = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{bmatrix}$$

$$\boldsymbol{X}_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 & \cdots & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & \cdots & 21 & 22 & 23 & 12 \end{bmatrix}$$

Important Matrix/Tensor Products

Kronecker Product

The Kronecker product between two arbitrarily-sized matrices $\boldsymbol{A} \in \mathbb{R}^{I \times J}$ and $\boldsymbol{B} \in \mathbb{R}^{K \times L}$ is a generalization of the outer product from vectors to matrices.

$$\boldsymbol{A} \otimes \boldsymbol{B} := \begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots & a_{1J}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots & a_{2J}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\boldsymbol{B} & a_{I2}\boldsymbol{B} & \cdots & a_{IJ}\boldsymbol{B} \end{bmatrix}$$
$$= \begin{bmatrix} \boldsymbol{a}_1 \otimes \boldsymbol{b}_1 & \boldsymbol{a}_1 \otimes \boldsymbol{b}_2 & \cdots & \boldsymbol{a}_J \otimes \boldsymbol{b}_{L-1} & \boldsymbol{a}_J \otimes \boldsymbol{b}_L \end{bmatrix}$$

The resulting matrix will be of size $(IK) \times (JL)$.

Important Matrix/Tensor Products (cont.)

Khatri-Rao Product

The Khatri-Rao product between two matrices $\boldsymbol{A} \in \mathbb{R}^{I \times K}$ and $\boldsymbol{B} \in \mathbb{R}^{J \times K}$ corresponds to the column-wise Kronecker product.

$$\boldsymbol{A} \odot \boldsymbol{B} := \begin{bmatrix} \boldsymbol{a}_1 \otimes \boldsymbol{b}_1 & \boldsymbol{a}_2 \otimes \boldsymbol{b}_2 & \cdots & \boldsymbol{a}_K \otimes \boldsymbol{b}_K \end{bmatrix}$$

The resulting matrix will be of size $(IJ) \times K$.

Important Matrix/Tensor Products (cont.)

Hadamard Product

The Hadamard product between two same-sized matrices $\boldsymbol{A} \in \mathbb{R}^{I \times J}$ and $\boldsymbol{B} \in \mathbb{R}^{I \times J}$ corresponds to the element-wise matrix product.

$$\boldsymbol{A} * \boldsymbol{B} := \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1J}b_{1J} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2J}b_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}b_{I1} & a_{I2}b_{I2} & \cdots & a_{IJ}b_{IJ} \end{bmatrix}$$

The resulting matrix will also be of size $I \times J$.

Important Matrix/Tensor Products (cont.)

n-mode Product

The *n*-mode product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ and a matrix $\boldsymbol{M} \in \mathbb{R}^{J \times I_n}$ is denoted by $\mathcal{Y} = \mathcal{X} \times_n \boldsymbol{M}$. Intuitively, each mode-n fiber is multiplied by the matrix $\boldsymbol{M}$.

$$(\mathcal{X} \times_n \boldsymbol{M})_{i_1 \cdots i_{n-1} j i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \cdots i_N} m_{j i_n}$$

The *n*-mode product can also be expressed through matricized tensors as $\boldsymbol{Y}_{(n)} = \boldsymbol{M}\boldsymbol{X}_{(n)}$.
For vectors, we analogously define:

$$(\mathcal{X} \times_n \boldsymbol{v})_{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \cdots i_N} v_{i_n}$$

Jennrich's Algorithm

If $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ are full rank, then $\mathcal{X} = \sum_{r=1}^{R} \lambda_r \boldsymbol{a}_r \circledcirc \boldsymbol{b}_r \circledcirc \boldsymbol{c}_r$ is unique up to rank permutation and feature scaling.

We can use Jennrich's algorithm to recover the factor matrices.

1. Choose random vectors $\boldsymbol{x}$ and $\boldsymbol{y}$.
2. $\mathcal{X}(\boldsymbol{I}, \boldsymbol{I}, \boldsymbol{x}) = \sum_{r=1}^{R} \langle \boldsymbol{c}_r, \boldsymbol{x} \rangle \boldsymbol{a}_r \circledcirc \boldsymbol{b}_r = \boldsymbol{A}\text{Diag}(\langle \boldsymbol{c}_r, \boldsymbol{x} \rangle)\boldsymbol{B}^T$.
3. $\mathcal{X}(\boldsymbol{I}, \boldsymbol{I}, \boldsymbol{y}) = \sum_{r=1}^{R} \langle \boldsymbol{c}_r, \boldsymbol{y} \rangle \boldsymbol{a}_r \circledcirc \boldsymbol{b}_r = \boldsymbol{A}\text{Diag}(\langle \boldsymbol{c}_r, \boldsymbol{y} \rangle)\boldsymbol{B}^T$.
4. $\mathcal{X}(\boldsymbol{I}, \boldsymbol{I}, \boldsymbol{x})\ \mathcal{X}(\boldsymbol{I}, \boldsymbol{I}, \boldsymbol{y})^\dagger = \boldsymbol{A}\text{Diag}(\langle \boldsymbol{c}_r, \boldsymbol{x} \rangle)\text{Diag}(\langle \boldsymbol{c}_r, \boldsymbol{y} \rangle)^\dagger \boldsymbol{A}^\dagger$
   $\Rightarrow$ eigendecomposition to find $\boldsymbol{A}$
5. $\mathcal{X}(\boldsymbol{I}, \boldsymbol{I}, \boldsymbol{x})^\dagger\ \mathcal{X}(\boldsymbol{I}, \boldsymbol{I}, \boldsymbol{y}) = (\boldsymbol{B}^T)^\dagger \text{Diag}(\langle \boldsymbol{c}_r, \boldsymbol{x} \rangle)^\dagger \text{Diag}(\langle \boldsymbol{c}_r, \boldsymbol{y} \rangle)\boldsymbol{B}^T$
   $\Rightarrow$ eigendecomposition to find $\boldsymbol{B}$
6. Match up the factors and solve a linear system to find $\boldsymbol{C}$.

However, we are still not using the full tensor structure.

Alternating Least Squares (ALS)

Fix all factor matrices except for one in order to optimize for the non-fixed matrix and then repeat this step for every matrix repeatedly until some stopping criterion is satisfied.

$$\boldsymbol{A} \leftarrow \arg\min_{\boldsymbol{A}} ||\boldsymbol{X}_{(1)} - (\boldsymbol{C} \odot \boldsymbol{B})\boldsymbol{A}^T||$$

$$\boldsymbol{B} \leftarrow \arg\min_{\boldsymbol{B}} ||\boldsymbol{X}_{(2)} - (\boldsymbol{C} \odot \boldsymbol{A})\boldsymbol{B}^T||$$

$$\boldsymbol{C} \leftarrow \arg\min_{\boldsymbol{C}} ||\boldsymbol{X}_{(3)} - (\boldsymbol{B} \odot \boldsymbol{A})\boldsymbol{C}^T||$$

The factor matrices are unique if the following holds:

- $\min_{1,\ldots,N} \left(\prod_{\substack{m=1 \\ m \neq n}}^{N} \text{rank}(\boldsymbol{A}^{(m)}) \right) \geq R$
- For $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$: $R \leq K$ and $R(R-1) \leq \frac{I(I-1)J(J-1)}{2}$

Canonical Polyadic Decomposition (CPD) (cont.)

Optimal ALS Solution

$$\hat{\boldsymbol{A}} = \boldsymbol{X}_{(1)}[(\boldsymbol{C} \odot \boldsymbol{B})^T]^\dagger = \boldsymbol{X}_{(1)}(\boldsymbol{C} \odot \boldsymbol{B})(C^T C * B^T B)^\dagger$$
$$\hat{\boldsymbol{B}} = \boldsymbol{X}_{(2)}[(\boldsymbol{C} \odot \boldsymbol{A})^T]^\dagger = \boldsymbol{X}_{(2)}(\boldsymbol{C} \odot \boldsymbol{A})(C^T C * A^T A)^\dagger$$
$$\hat{\boldsymbol{C}} = \boldsymbol{X}_{(3)}[(\boldsymbol{B} \odot \boldsymbol{A})^T]^\dagger = \boldsymbol{X}_{(3)}(\boldsymbol{B} \odot \boldsymbol{A})(B^T B * A^T A)^\dagger$$

However, this setting can be numerically unstable due to potential ill-conditioning.

Canonical Polyadic Decomposition (CPD) (cont.)

CPD Through ALS

procedure CP-ALS($\mathcal{X}$, R)
  initialize $\boldsymbol{A}^{(n)} \in R^{I_n \times R}$ for $n = 1, \ldots, N$
  repeat
    for $n = 1, \ldots, N$ **do**
      $\boldsymbol{V} \leftarrow \boldsymbol{A}^{(1)T}\boldsymbol{A}^{(1)} * \ldots * \boldsymbol{A}^{(n-1)T}\boldsymbol{A}^{(n-1)} * \boldsymbol{A}^{(n+1)T}\boldsymbol{A}^{(n+1)} *$
      $\hookrightarrow \ \ldots * \boldsymbol{A}^{(N)T}\boldsymbol{A}^{(N)}$
      $\boldsymbol{A}^{(n)} \leftarrow \boldsymbol{X}_{(n)}(\boldsymbol{A}^{(N)} \odot \ldots \odot \boldsymbol{A}^{(n+1)} \odot \boldsymbol{A}^{(n-1)} \odot \ldots \odot \boldsymbol{A}^{(1)})\boldsymbol{V}^{\dagger}$
      normalize columns of $\boldsymbol{A}^{(n)}$ *(optional)*
      store norms as $\boldsymbol{\lambda}$ *(optional)*
    end for
  until stopping criterion satisfied
  return $\boldsymbol{\lambda}, \boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}$
end procedure

Canonical Polyadic Decomposition (CPD) (cont.)

Tensor Rank Peculiarities

- There is no trivial algorithm to determine the rank of a tensor as the problem is NP-hard.
- The rank of a tensor with real entries can be different over $\mathbb{R}$ and $\mathbb{C}$.
- The maximum rank of a tensor represents the largest attainable rank. The typical rank is any rank that occurs with probability ≥ 0. These two ranks are usually not the same for tensors (while they are for matrices).
- The rank of a tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ is constant under permutation and bounded from above by the following equation:

$$\text{rank}(\mathcal{X}) \leq \min\{IJ, IK, JK\}$$

Tucker Decomposition (cont.)

The Tucker decomposition is not unique! This follows from the fact that all columns of the factor matrices can interact with arbitrary other columns through the core tensor.

CPD can be thought of a special case of the Tucker decomposition with a superdiagonal core.

Tucker Decomposition (cont.)

n-rank

The n-rank of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, denoted $R_n = \text{rank}_n(\mathcal{X})$, corresponds to the column rank of $\boldsymbol{X}_{(n)}$.

Higher Order Singular Value Decomposition (HOSVD)

procedure $\text{HOSVD}(\mathcal{X}, R_1, \ldots, R_N)$
  for n $= 1,\ldots,$N **do**
    $\boldsymbol{A}^{(n)} \leftarrow R_n$ leading left singular vectors of $\boldsymbol{X}_{(n)}$
  end for
  $\mathcal{G} \leftarrow \mathcal{X} \times_1 \boldsymbol{A}^{(1)T} \times_2 \cdots \times_N \boldsymbol{A}^{(N)T}$
  return $\mathcal{G}, \boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}$
end procedure

This algorithm does not give an optimal fit, but it is a good starting point for a successive application of the ALS algorithm!

Tucker Decomposition (cont.)

Higher Order Orthogonal Iteration (HOOI)

procedure $\mathrm{HOOI}(\mathcal{X}, R_1, \ldots, R_N)$
  initialize $\boldsymbol{A}^{(n)} \in R^{I_n \times R}$ for $n = 1, \ldots, N$ using HOSVD
  repeat
    for n = 1,. . . ,N **do**
      $\mathcal{Y} \leftarrow \mathcal{X} \times_1 \boldsymbol{A}^{(1)T} \times_2 \cdots \times_{n-1} \boldsymbol{A}^{(n-1)T} \times_{n+1} \boldsymbol{A}^{(n+1)T} \times_{n+2} \hookrightarrow \cdots \times_N \boldsymbol{A}^{(N)T}$
      $\boldsymbol{A}^{(n)} \leftarrow R_n$ leading left singular vectors of $\boldsymbol{Y}_{(n)}$
    end for
  until stopping criterion satisfied
  $\mathcal{G} \leftarrow \mathcal{X} \times_1 \boldsymbol{A}^{(1)T} \times_2 \cdots \times_N \boldsymbol{A}^{(N)T}$
  return $\mathcal{G}, \boldsymbol{A}^{(1)}, \ldots, \boldsymbol{A}^{(N)}$
end procedure

CPD vs. Tucker

CPD	Tucker
linear tensor decomposition	multilinear tensor decomposition
column-matching interaction	various interactions
true underlying factors	compression
unique	not unique

Other Decompositions

- **Individual differences in scaling** (**INDSCAL**): special case of CPD for 3-way tensors with two symmetric modes.
- **Parallel factors for cross products** (**PARAFAC2**): CPD which can be applied to a collection of matrices with the same amount of columns, but a different number of rows.
- **CANDECOMP with linear constraints** (**CANDELINC**): add constraints to CPD's factor matrices through domain/user knowledge.
- **Decomposition into directional components** (**DEDICOM**)
- **PARAFAC and Tucker 2** (**PARATUCK2**)