Extending the SVD to higher dimensions with Tensor Decompositions

David Grenier

University of Rhode Island

2024

Introduction

Introduction

- **Overview**: In this talk we shall introduce the concept of tensors and examine two decomposition methods related to SVD.
- **Main source**: Kolda, Tamara and Bader, Brett. "Tensor Decompositions and Applications." *SIAM Review*, vol 51, no. 3, 2009, pp 455-500.
- **Audience**: It is assumed you have taken MTH 215 and one higher-level linear algebra class (MTH 418/518, MTH 513, or MTH 472/AMD 590)

What is a tensor?

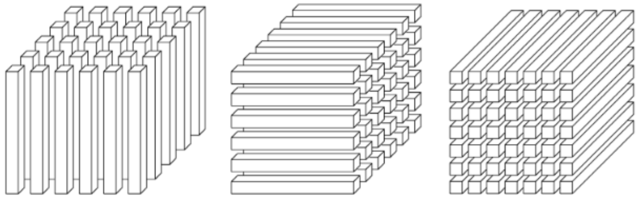
$$a = a \qquad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Scalar **Vector** **Matrix**

$$\mathcal{A} = \begin{bmatrix} a_{111} & a_{121} & a_{131} \\ a_{211} & a_{221} & a_{231} \\ a_{311} & a_{321} & a_{331} \end{bmatrix} \begin{bmatrix} a_{112} & a_{122} & a_{132} \\ a_{212} & a_{222} & a_{232} \\ a_{312} & a_{322} & a_{332} \end{bmatrix} \begin{bmatrix} a_{113} & a_{123} & a_{133} \\ a_{213} & a_{223} & a_{233} \\ a_{313} & a_{323} & a_{333} \end{bmatrix}$$

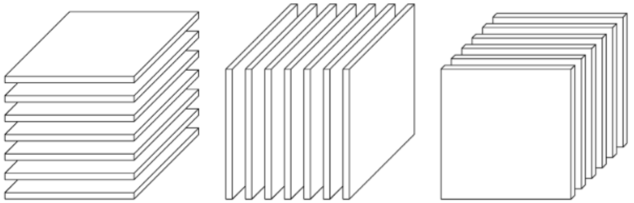
Tensor

What is a tensor?



(a) Mode-1 (column) fibers: $\mathbf{x}_{:jk}$ (b) Mode-2 (row) fibers: $\mathbf{x}_{i:k}$ (c) Mode-3 (tube) fibers: $\mathbf{x}_{ij:}$

Fig. 2.1 *Fibers of a 3rd-order tensor.*



(a) Horizontal slices: $\mathbf{X}_{i::}$ (b) Lateral slices: $\mathbf{X}_{:j:}$ (c) Frontal slices: $\mathbf{X}_{::k}$ (or $\mathbf{X}_k$)

What is decomposition?

Decomposition is the factorization of a matrix or tensor into the product of component matrices, tensors, or vectors. This is often done to take advantage of the structural properties of one or more of the components.

Examples of matrix decompositions:

- *Eigenvalue Decomposition*: $\mathbf{A} = \mathbf{X}\boldsymbol{\Lambda}\mathbf{X}^{-1}$
- *Cholesky Decomposition*: $\mathbf{A} = \mathbf{R}^T\mathbf{R}$
- *QR Decomposition*: $\mathbf{A} = \mathbf{QR}$
- *Schur Decomposition*: $\mathbf{A} = \mathbf{UTU}^H$
- *Singular Value Decomposition*: $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$

Singular Value Decomposition (SVD)

Quick refresher

SVD is a fundamental technique in linear algebra for analyzing and decomposing matrices. It expresses any matrix **A** as the product of three matrices:

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

where:

- **U** and **V**are orthogonal matrices containing the left and right singular vectors **u** and **v**.
- $\boldsymbol{\Sigma}$ is a diagonal matrix with singular values $\sigma_1, \sigma_2, \ldots, \sigma_n$.
- r is the rank of the matrix **A**.

Applications of SVD

- **Data Compression:** Enables efficient storage and processing by identifying and discarding less significant components of large datasets, notably in image processing.
- **Noise Reduction:** Improves data quality by separating significant data components (signal) from the less significant ones (noise), essential in signal processing and data mining.
- **Feature Extraction:** Facilitates machine learning by simplifying datasets to their most meaningful bases, improving predictive model efficiency and accuracy.
- **Principal Component Analysis (PCA):** A direct application of SVD, PCA simplifies data analysis by reducing dimensions and highlighting essential features, improving visualization and model accuracy.

Extending SVD to Tensors (HOSVD)

Motivational examples

Envisioning a Higher-Order SVD (HOSVD) opens up new avenues for analyzing multidimensional data, with applications including:

- **Longitudinal Data Analysis:** A 3-D tensor with axes for school districts, standardized test scores, and years could leverage HOSVD for trend analysis and noise reduction, enhancing educational research and policy-making.
- **Video Compression:** In a 4-D tensor representing video (pixels in two dimensions, color channels, and time), HOSVD could significantly optimize storage and streaming by compressing data without compromising quality.
- **Multifaceted Social Network Analysis:** Considering a tensor with dimensions for users, interactions (e.g., likes, comments), and time, HOSVD can unearth evolving patterns and clusters in social dynamics, offering insights into user behavior and content popularity over time.

Multiple methods

There is no single, obvious technique to extend the SVD to tensors. Instead, we have different techniques that capture different aspects of the matrix SVD.

We will talk about two methods today: CP Decomposition and Tucker1.

Method 1: CPD

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \qquad \longrightarrow \qquad \mathcal{X} = \sum_{i=1}^{r} \lambda_i \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \circ \mathbf{a}^{(N)}$$

To extend the SVD property that a matrix is the sum of r rank-one outer products of singular vectors **u** and **v**, we use the tensor decomposition **Canonical Polyadic Decomposition (CPD)**.

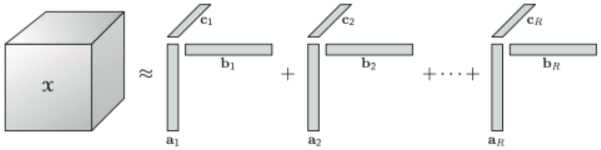


Fig. 3.1 *CP decomposition of a three-way array.*

Uses of CPD

Advantages

- **Uniqueness:** Guarantees unique decomposition, enhancing interpretability.
- **Simplicity:** Extends matrix decomposition concepts to tensors straightforwardly.

Applications

- **Chemometrics:** For complex chemical data analysis.
- **Signal Processing:** Improves clarity in telecommunications.
- **Data Fusion:** Merges data from multiple sources effectively.

Method 2: Tucker1

$$\mathbf{A} = \mathbf{U\Sigma V}^T \qquad \longrightarrow \qquad \mathcal{X} = \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_N \mathbf{A}^{(N)}$$

To extend the SVD property that a matrix is a core matrix multiplied by a vector in each direction, we use the **Tucker1** decomposition.

TENSOR DECOMPOSITIONS AND APPLICATIONS 475

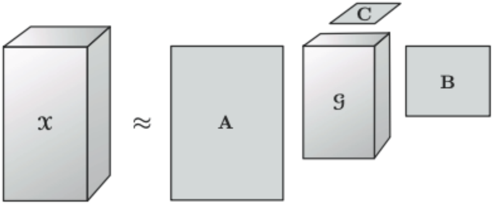


Fig. 4.1 *Tucker decomposition of a three-way array.*

Uses of Tucker1

Advantages

- **Flexibility:** Allows adaptable data compression and insights.
- **Hierarchy:** Reveals data structure and mode interactions.

Applications

- **Image Processing:** For advanced compression and feature preservation.
- **Machine Learning:** Enhances algorithms by identifying patterns.
- **Neuroscience:** Uncovers brain activity patterns across conditions.

Tools used in Tensor Decomposition

New tools

The decomposition algorithms Alternating Least Squares (for CPD) and Tucker1 involve transforming tensors into matrices, using specialized matrix products to get matrices into an advantageous form, transforming matrices back into tensors, and a specialized method for a tensor-matrix product. We shall review these before diving deeper into our two decompositions.

- Mode-*n* unfoldings
- *n*-mode tensor-matrix product
- Hadamard product
- Kronecker product
- Khatri-Rao products

Mode-n unfoldings

$$\mathcal{X} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

$$\mathcal{X}^{(1)} = \mathbf{X}_{(1)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 \end{bmatrix}$$

$$\mathcal{X}^{(2)} = \mathbf{X}_{(2)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 & 4 \end{bmatrix}$$

$$\mathcal{X}^{(3)} = \mathbf{X}_{(3)} = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 & 4 & 4 & 4 \\ 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 & 4 & 4 & 4 \end{bmatrix}$$

The n-mode tensor-matrix product

$$\mathcal{G} = \mathcal{X} \times_1 \mathbf{A} \qquad \Longleftrightarrow \qquad \mathbf{G}_{(1)} = \mathbf{A}\mathbf{X}_{(1)}$$

For example. Let:

$$\mathcal{X} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$\begin{aligned} \mathbf{G}_{(1)} &= \mathbf{A}\mathbf{X}_{(1)} \\ &= \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{bmatrix} \\ &= \begin{bmatrix} a+3b & 2a+4b & 5a+7b & 6a+8b \\ c+3d & 2c+4d & 5c+7d & 6c+8d \end{bmatrix} \end{aligned}$$

$$\mathcal{G} = \begin{bmatrix} a+3b & 2a+4b \\ c+3d & 2c+4d \end{bmatrix}\begin{bmatrix} 5a+7b & 6a+8b \\ 5c+7d & 6c+8d \end{bmatrix}$$

Hadamard product

Let $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{m \times n}$. Then the **Hadamard Product** $(\mathbf{A} * \mathbf{B})$ is the elementwise matrix product.

$$\mathbf{A} * \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{12}b_{12} & \cdots & a_{1n}b_{1n} \\ a_{21}b_{21} & a_{22}b_{22} & \cdots & a_{2n}b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{m1} & a_{m2}b_{m2} & \cdots & a_{mn}b_{mn} \end{bmatrix}$$

Kronecker and Khatri-Rao products

Let $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{B} \in \mathbb{R}^{p\times q}$. Then the **Kronecker Product** $(\mathbf{A} \otimes \mathbf{B})$ is a $mn \times pq$ block matrix made up of mn copies of $\mathbf{B}$ scaled by a_{ij}.

$$\begin{aligned}\mathbf{A} \otimes \mathbf{B} &= \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2n}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & a_{m2}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_1 \otimes \mathbf{b}_2 & \mathbf{a}_1 \otimes \mathbf{b}_3 & \cdots & \mathbf{a}_n \otimes \mathbf{b}_{q-1} & \mathbf{a}_n \otimes \mathbf{b}_q \end{bmatrix}\end{aligned}$$

Let $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{B} \in \mathbb{R}^{p\times n}$. Then the **Katri-Rao Product** $(\mathbf{A} \odot \mathbf{B})$ is the "matching columnwise" Kronecker product.

$$\mathbf{A} \odot \mathbf{B} = \begin{bmatrix} \mathbf{a}_1 \otimes \mathbf{b}_1 & \mathbf{a}_2 \otimes \mathbf{b}_2 & \cdots & \mathbf{a}_n \otimes \mathbf{b}_n \end{bmatrix}$$

CP Decomposition

CPD Overview (redux)

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \qquad \longrightarrow \qquad \mathcal{X} = \sum_{i=1}^{r} \lambda_i \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \circ \mathbf{a}^{(N)}$$

To extend the SVD property that a matrix is the sum of r rank-one outer products of singular vectors **u** and **v**, we use the tensor decomposition **Canonical Polyadic Decomposition (CPD)**.

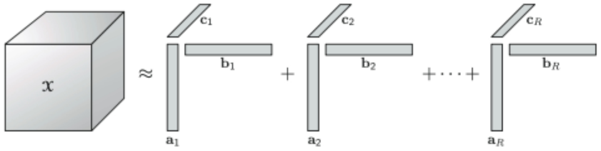


Fig. 3.1 *CP decomposition of a three-way array.*

A note on nomenclature

The decomposition of a tensor into a sum of rank-one tensors, introduced by Hitchcock in 1927 as the "polyadic form," was later rediscovered in the 1970s. Harshman named it PARAFAC (parallel factors), while Carroll and Chang called it CANDECOMP (canonical decomposition).

Today, this decomposition is widely known as CANDECOMP/PARAFAC or CP Decomposition (CPD). "Canonical Polyadic Decomposition" is a backronym for CPD, linking it to Hitchcock's original work.

These terms all refer to the same decomposition technique:

- Polyadic Form
- PARAFAC
- CANDECOMP
- CANDECOMP/PARAFAC (CP)
- CP Decomposition (CPD)
- Canonical Polyadic Decomposition (CPD)

Notes on tensor rank

1. A tensor that is the outer product of n vectors is **rank one**. I.E. $\boldsymbol{\mathcal{X}} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \cdots \circ \mathbf{a}^{(n)}$
2. The rank of a tensor is $\boldsymbol{\mathcal{X}}$ is defined as "the smallest number of rank-one tensors that generate $\boldsymbol{\mathcal{X}}$ as their sum." (Kolda and Barrett, 2009)
3. One can find the rank of a matrix by the number of pivots, the dimension of the range, etc. There is no method for finding the rank of a tensor. We can bound the ranks of certain **kinds** of tensors, but finding a specific tensor's rank is NP-hard.
4. Tensor rank-decompositions are unique up to permutation and scaling.
5. The singular vectors that make up a rank k approximation of a matrix are a subset of those that make up a rank $k+1$ approximation. This is not true of approximations of a tensor $\boldsymbol{\mathcal{X}}$ using CPD.
6. Sometimes lower-rank approximations have smaller error than higher rank approximations.

Decomposition method

Now that we have discussed the concepts of tensor rank and CP Decomposition, we can finally explore how the decomposition is achieved.

To find a rank R approximation, we use the iterative Alternating Least Squares algorithm to find an approximation $\hat{\boldsymbol{\mathcal{X}}}$ of N ordered tensor $\boldsymbol{\mathcal{X}}$ such that:

$$\hat{\boldsymbol{\mathcal{X}}} = \sum_{i=1}^{R} \mathbf{a_i}^{(1)} \circ \mathbf{a_i}^{(2)} \circ \cdots \circ \mathbf{a_i}^{(N)}$$

Note that the vectors $\mathbf{a_i}^{(n)}$ are the columns of matrices $\mathbf{A}^{(n)}$ in the algorithm.

As the actual rank of $\boldsymbol{\mathcal{X}}$ is unknown, it may be necessary to attempt approximations of several ranks R until the error $||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}||$ is below some threshold.

Alternating Least Squares algorithm

```
Algorithm CP-ALS($\mathcal{X}, R$)
  initialize $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ for $n = 1, \ldots, N$
  repeat
      for $n = 1, \ldots, N$ do
          $\mathbf{V} \leftarrow \mathbf{A}^{(1)T}\mathbf{A}^{(1)} * \cdots * \mathbf{A}^{(n-1)T}\mathbf{A}^{(n-1)} * \mathbf{A}^{(n+1)T}\mathbf{A}^{(n+1)} * \cdots * \mathbf{A}^{(N)T}\mathbf{A}^{(N)}$
          $\mathbf{A}^{(n)} \leftarrow \boldsymbol{\mathcal{X}}^{(n)}\left(\mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)}\right)\mathbf{V}^{\dagger}$
          normalize columns of $\mathbf{A}^{(n)}$ (storing norms as $\boldsymbol{\lambda}$)
      end for
  until fit ceases to improve or maximum iterations exhausted
  return $\boldsymbol{\lambda}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(N)}$
```

$*$	Hadamard Product
$\odot$	Khatri-Rao Product
$\dagger$	Moore-Penrose inverse
$\mathbf{A}^{(n)}$	matrix whose columns are the nth vector in our outer product.
$\mathcal{X}^{(n)}$	is the *n*-mode unfolding of $\mathcal{X}$.

CP-ALS example: Rank-2 approximation of 3-tensor

$$\mathcal{X} = (\mathbf{a}_1 \circ \mathbf{b}_1 \circ \mathbf{c}_1) + (\mathbf{a}_2 \circ \mathbf{b}_2 \circ \mathbf{c}_2); \qquad \mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{b}_1 & \mathbf{b}_2 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} \mathbf{c}_1 & \mathbf{c}_2 \end{bmatrix}$$

Algorithm CP-ALS $(\mathcal{X}, 1)$

```
initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$
repeat
    $\mathbf{V}_A \leftarrow \mathbf{B}^T\mathbf{B} * \mathbf{C}^T\mathbf{C}$
    $\mathbf{A} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})\mathbf{V_A}^{\dagger}$
    $\boldsymbol{\lambda}, \mathbf{A} \leftarrow \begin{bmatrix} ||\mathbf{a}_1|| & ||\mathbf{a}_2|| \end{bmatrix}, \begin{bmatrix} \frac{\mathbf{a}_1}{||a_1||} & \frac{\mathbf{a}_2}{||\mathbf{a}_2||} \end{bmatrix}$

    $\mathbf{V_B} \leftarrow \mathbf{A}^T\mathbf{A} * \mathbf{C}^T\mathbf{C}$
    $\mathbf{B} \leftarrow \mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A})\mathbf{V_B}^{\dagger}$
    $\boldsymbol{\lambda}, \mathbf{B} \leftarrow \begin{bmatrix} ||\mathbf{b}_1|| & ||\mathbf{b}_2|| \end{bmatrix}, \begin{bmatrix} \frac{\mathbf{b}_1}{||b_1||} & \frac{\mathbf{b}_2}{||\mathbf{b}_2||} \end{bmatrix}$

    $\mathbf{V_C} \leftarrow \mathbf{A}^T\mathbf{A} * \mathbf{B}^T\mathbf{B}$
    $\mathbf{C} \leftarrow \mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})\mathbf{V_C}^{\dagger}$
    $\boldsymbol{\lambda}, \mathbf{C} \leftarrow \begin{bmatrix} ||\mathbf{c}_1|| & ||\mathbf{c}_2|| \end{bmatrix}, \begin{bmatrix} \frac{\mathbf{c}_1}{||c_1||} & \frac{\mathbf{c}_2}{||\mathbf{c}_2||} \end{bmatrix}$
until fit ceases to improve or maximum iterations exhausted
return $\boldsymbol{\lambda}, \mathbf{A}, \mathbf{B}, \mathbf{C}$
```

A note on $\boldsymbol{\lambda}$

During each iteration of ALS we generate three $\boldsymbol{\lambda}$ values $\boldsymbol{\lambda}_A, \boldsymbol{\lambda}_B, \boldsymbol{\lambda}_C$ that scale our normalized $\mathbf{A}, \mathbf{B}, \mathbf{C}$ matrices.

In our first iteration the values of $\boldsymbol{\lambda}$ were quite different.

$$\boldsymbol{\lambda}_A = \begin{bmatrix} 1.811 \\ 3.8909 \end{bmatrix} \qquad \boldsymbol{\lambda}_B = \begin{bmatrix} 6.3606 \\ 16.0425 \end{bmatrix} \qquad \boldsymbol{\lambda}_C = \begin{bmatrix} 31.9686 \\ 27.9719 \end{bmatrix}$$

However, the values of $\boldsymbol{\lambda}$ move towards each other quickly. On only the second iteration we have:

$$\boldsymbol{\lambda}_A = \begin{bmatrix} 32.5400 \\ 28.4934 \end{bmatrix} \qquad \boldsymbol{\lambda}_B = \begin{bmatrix} 32.3805 \\ 28.6967 \end{bmatrix} \qquad \boldsymbol{\lambda}_C = \begin{bmatrix} 32.3996 \\ 28.6951 \end{bmatrix}$$

They eventually converge so that

$$\boldsymbol{\lambda} \approx \boldsymbol{\lambda}_A \approx \boldsymbol{\lambda}_B \approx \boldsymbol{\lambda}_C$$

Results

Note that we only use **one** of these $\boldsymbol{\lambda}$ values in our approximation.

$\hat{\boldsymbol{\mathcal{X}}} = \sum_{i=1}^{2} \lambda_i \cdot (\mathbf{a}_i \circ \mathbf{b}_i \circ \mathbf{c}_i)$

- 17,192 iterations
- error: 3.7399×10^{-6}

$\hat{\boldsymbol{\mathcal{X}}} = \sum_{i=1}^{2} (\lambda_{A_i} \mathbf{a}_i) \circ (\lambda_{B_i} \mathbf{b}_i) \circ (\lambda_{C_i} \mathbf{c}_i)$

- 20,563 iterations
- error: 463.78

Where error is $||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}||$.

Tucker Decomposition

Tucker1 (redux)

$$\mathbf{A} = \mathbf{U\Sigma V}^T \quad \longrightarrow \quad \mathcal{X} = \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_N \mathbf{A}^{(N)}$$

To extend the SVD property that a matrix is a core matrix multiplied by a vector in each direction, we use the **Tucker1** decomposition.

TENSOR DECOMPOSITIONS AND APPLICATIONS 475

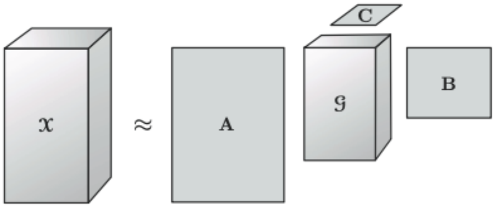


Fig. 4.1 *Tucker decomposition of a three-way array.*

Tucker1 (HOSVD) algorithm

```
Algorithm HOSVD($\mathcal{X}, R_1, R_2, \ldots, R_N$)
  for $n = 1, \ldots, N$ do
      $\mathbf{A}^{(n)} \leftarrow R_n$ leading singular vectors of $\mathbf{X}_{(n)}$
  end for
  $\mathcal{G} \leftarrow \boldsymbol{\mathcal{X}} \times_1 \mathbf{A}^{(1)T} \times_2 \mathbf{A}^{(2)T} \times_3 \cdots \times_N \mathbf{A}^{(N)T}$
  return $\mathcal{G}, \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \cdots, \mathbf{A}^{(N)}$
```

$\times_n$ the n-mode (matrix) product.

$\mathbf{A}^{(n)}$ matrix whose columns are the nth vector in our outer product.

$\mathcal{X}^{(n)}$ is the n-mode unfolding of $\mathcal{X}$.

Tucker1 example: Rank-2 approximation of 3-tensor

Algorithm HOSVD($\mathcal{X}, 2, 2, 2$)

```
    $\mathbf{A} \leftarrow$ 2 leading singular vectors of $\mathbf{X}_{(1)}$
    $\mathbf{B} \leftarrow$ 2 leading singular vectors of $\mathbf{X}_{(2)}$
    $\mathbf{C} \leftarrow$ 2 leading singular vectors of $\mathbf{X}_{(3)}$
    $\boldsymbol{\mathcal{G}} \leftarrow \boldsymbol{\mathcal{X}} \times_1 \mathbf{A} \times_2 \mathbf{B} \times_3 \mathbf{C}$
    **return** $\mathcal{G}, \mathbf{A}, \mathbf{B}, \mathbf{C}$
```

Tucker1 Result

$$\mathcal{X} = \begin{bmatrix} 3 & 1 \\ 4 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\mathcal{G} = \begin{bmatrix} -5.9053 & -0.0113 \\ 0.0513 & -0.4959 \end{bmatrix} \begin{bmatrix} -0.0215 & 1.3519 \\ -0.0561 & 0.2192 \end{bmatrix}$$

$$\mathbf{A} = \begin{bmatrix} -0.5669 & 0.8238 \\ -0.8238 & -0.5669 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} -0.8715 & 0.4904 \\ -0.4904 & -0.8715 \end{bmatrix}$$

$$\mathbf{C} = \begin{bmatrix} -0.9198 & -0.3925 \\ -0.3925 & 0.9198 \end{bmatrix}$$

$$\text{error} = \left\| \mathcal{X} - \hat{\mathcal{X}} \right\| = 2.4324 \times 10^{-15}$$

Conclusion

In review

- Introduction to tensors.
- Understanding tensor rank vs. matrix rank.
- Properties of the SVD preserved by CPD and Tucker1.
- CP Decomposition with the ALS algorithm
- HOSVD with the Tucker1 algorithm.

Further investigation

- Deeper understanding of properties of Hadamard, Kronecker, and Khatri-Rao products that make them useful in ALS and Tucker1.
- Other decomposition methods mentioned in Kolda and Bader (2009): INDSCAL, PARAFAC2, CANDELINC, DEDICOM, etc.
- Newer methods developed since 2009, i.e. Tensor Train.
- Integration of these methodologies into MATLAB and Python.

Supplemental materials

The following materials are available as a supplement. They did not fit well in the presentation.

- MATLAB code for our CPD toy example.
- MATLAB code for our Tucker1 toy example.
- Hand calculations for first iteration of CPD toy example.

https://github.com/DavidGrenierRI/Tensor_Decomposition/
david_grenier@uri.edu

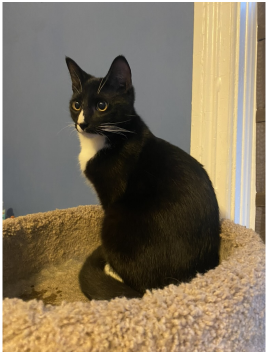
Questions?

Special thanks

Special thanks to Drs. Perovic and Baglama for introducing me to the world of linear algebra. Also thanks to my three research assistants, without whom this project would not be possible.

Andrey Barkov

René Descats


Blaise Pawscal