Canonical Polyadic Decomposition

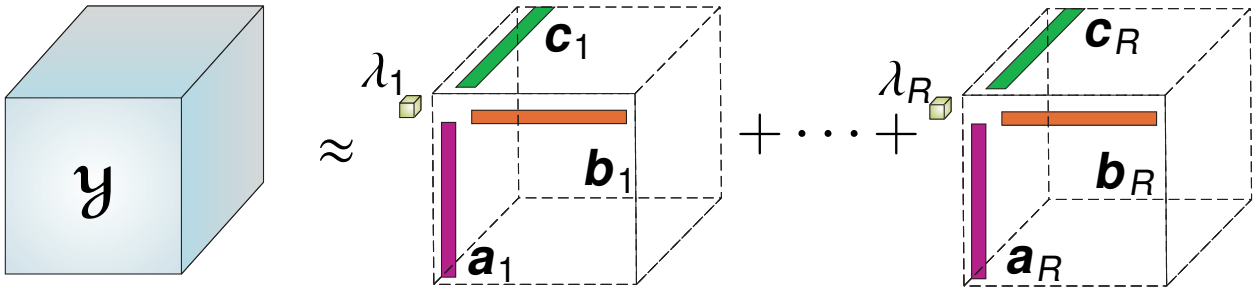
Center for Computational and Data-Intensive Science and Engineering (CDISE)
Skolkowo Institute of Science and Technology (SKOLTECH), Moscow, Russia

February 2, 2022

Part I

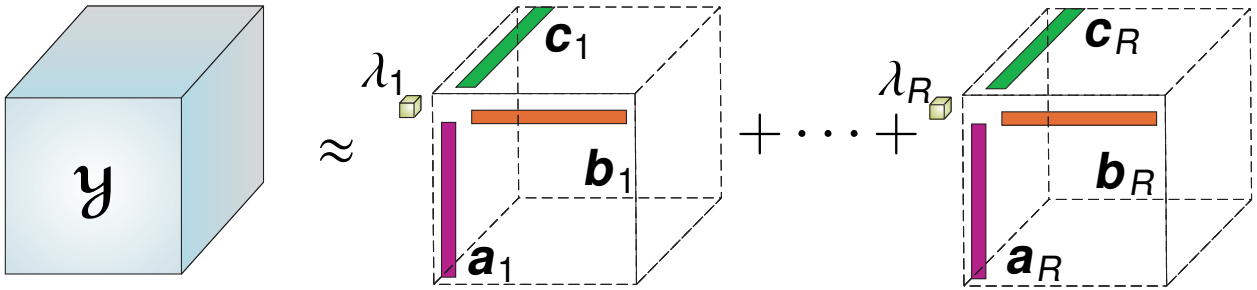
Research Summary: Challenging Problems in CANDECOMP/PARAFAC

CANDECOMP/PARAFAC (CPD)



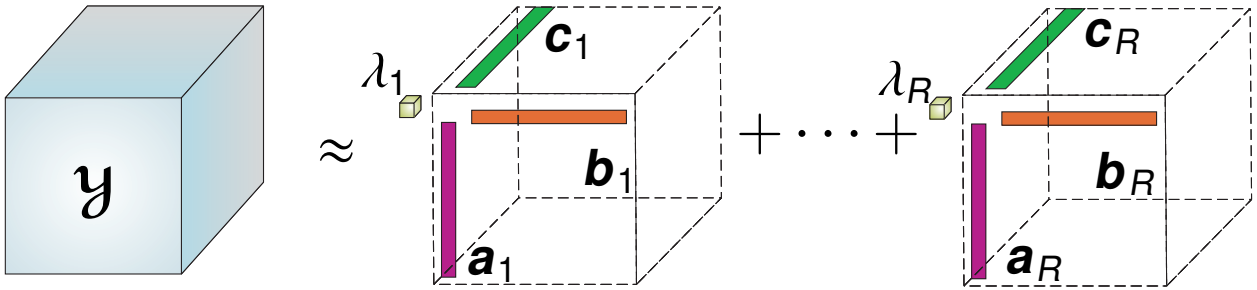
- Application of Tensors in a raw format may be intractable, as the required storage memory and a number of operations grow exponentially with the tensor order.
- To tackle this issue, represent higher-order tensors through multi-way operations over their latent components.
- Canonical Polyadic Decomposition (CPD): an extension of matrix factorization to extract rank-1 tensor patterns from multiway data.

CANDECOMP/PARAFAC (CPD)



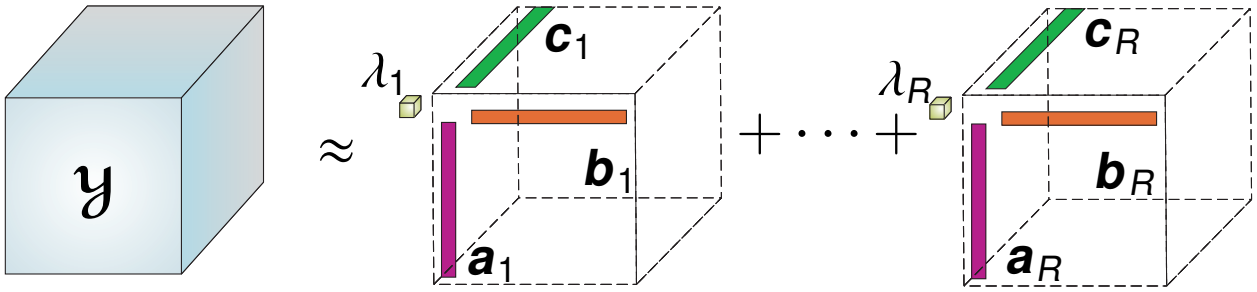
- 1927 first studied Hitchcock (1927)
- 1969 determined the complexity of matrix multiplication Strassen (1969).
- 1972 later known as parallel factor analysis (PARAFAC), a tool for chemometric analysis popularized by Harshman (1972),Carroll and Chang (1970) and KruskalKruskal (1977)
- Since the 1990's, the CPD has attracted attention in signal processing Sidiropoulos et al. (2000); Yeredor (2000); Comon and Rajih (2006).

CANDECOMP/PARAFAC (CPD)



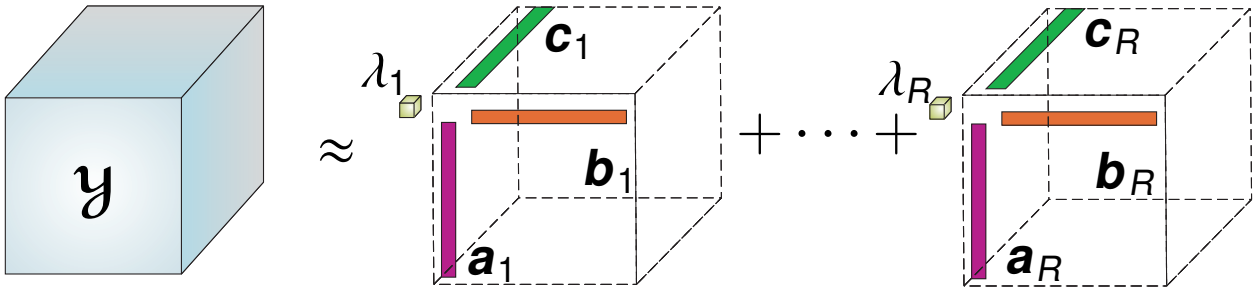
- 2000s New applications of CPD in feature extraction, data reconstruction, image completion and various tracking scenarios Nion and Sidiropoulos (2009); Phan et al. (2015).
- 2014-CPD of kernel tensors in convolutional can accelerate the inference of convolutional neural networks Jaderberg et al. (2014).
- In cyber security, the tensor can represent network traffic volume: time $\times$ senderIP $\times$ receiverIP $\times$ port.

CANDECOMP/PARAFAC (CPD)



- Over its long history, the model has been extended with various constraints
  with many efficient algorithms
- Despite the great successes, there are still many challenging problems in the CP tensor representation/decomposition.

CANDECOMP/PARAFAC (CPD)



- Computation for large volume and high order tensors, e.g., those of order $N = 10$ or 20
- Degeneracy

Challenging Problems in CPD

CPD can be achieved by minimizing the Frobenius norm of the error

$$\min \quad \|\mathcal{Y} - \hat{\mathcal{Y}}\|_F^2$$

where $\hat{\mathcal{Y}}$ is a low-rank approximation of $\mathcal{Y}$ in the CP format

$$\hat{y}_{i_1,i_2,\ldots,i_N} = \sum_{r=1}^{R} \lambda_r \, a_{i_1,r}^{(1)} \, a_{i_2,r}^{(2)} \, \cdots a_{i_N,r}^{(N)}$$

The tensor rank R can exceed the tensor dimension $R \geq I_n$.

Challenging Problems in CPD I

Prob.1: Nonlinear optimization

The optimisation is nonlinear with respect to all the parameters, but linear in parameters in each $\mathbf{A}^{(n)}$, or parameters in non-overlapping partitions of different factor matrices

$$D(\mathcal{Y}||\hat{\mathcal{Y}}) = \frac{1}{2}||\mathcal{Y} - \hat{\mathcal{Y}}||_F^2 = \frac{1}{2}||\mathbf{Y}_{(n)} - \mathbf{A}^{(n)}\mathbf{Z}_n^T||_F^2$$

where $\mathbf{Z}_n = \mathbf{A}^{(N)} \odot \cdots \odot \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \cdots \odot \mathbf{A}^{(1)}$, $\odot$ Khatri-Rao product of $\mathbf{A}^{(m)}$.

Prob.2: High computational cost $\mathcal{O}(NI^N R^3)$ due to the gradient

Computational costs of algorithms grow exponentially with the tensor order and tensor rank, due to computation of $\mathbf{Y}_{(n)}\,\mathbf{Z}_n$.
$\rightarrow$ Most algorithms are applicable to only tensors of low order, which never exceeds three.

Challenging Problems in CPD II

Prob.3: Degeneracy

Often occurs in some difficult scenarios, e.g.,

- when the rank exceeds the tensor dimension ($R > I_n$)
- when the loading components, $\boldsymbol{a}_r^{(n)}$ and $\boldsymbol{a}_s^{(n)}$, are highly collinear
- or when CPD does not provide an optimal solution

Prob.4: Second order optimisation is inadequate

Because inverse of large Hessian matrix of size $NIR \times NIR$.

Prob.5: CPD with additional constraints

nonnegativity, orthogonality, sparsity, smoothness constraints

Prob.6: A measure of performance

A scalar index to assess performance of a decomposition or of an algorithm

Challenging Problems in CPD III

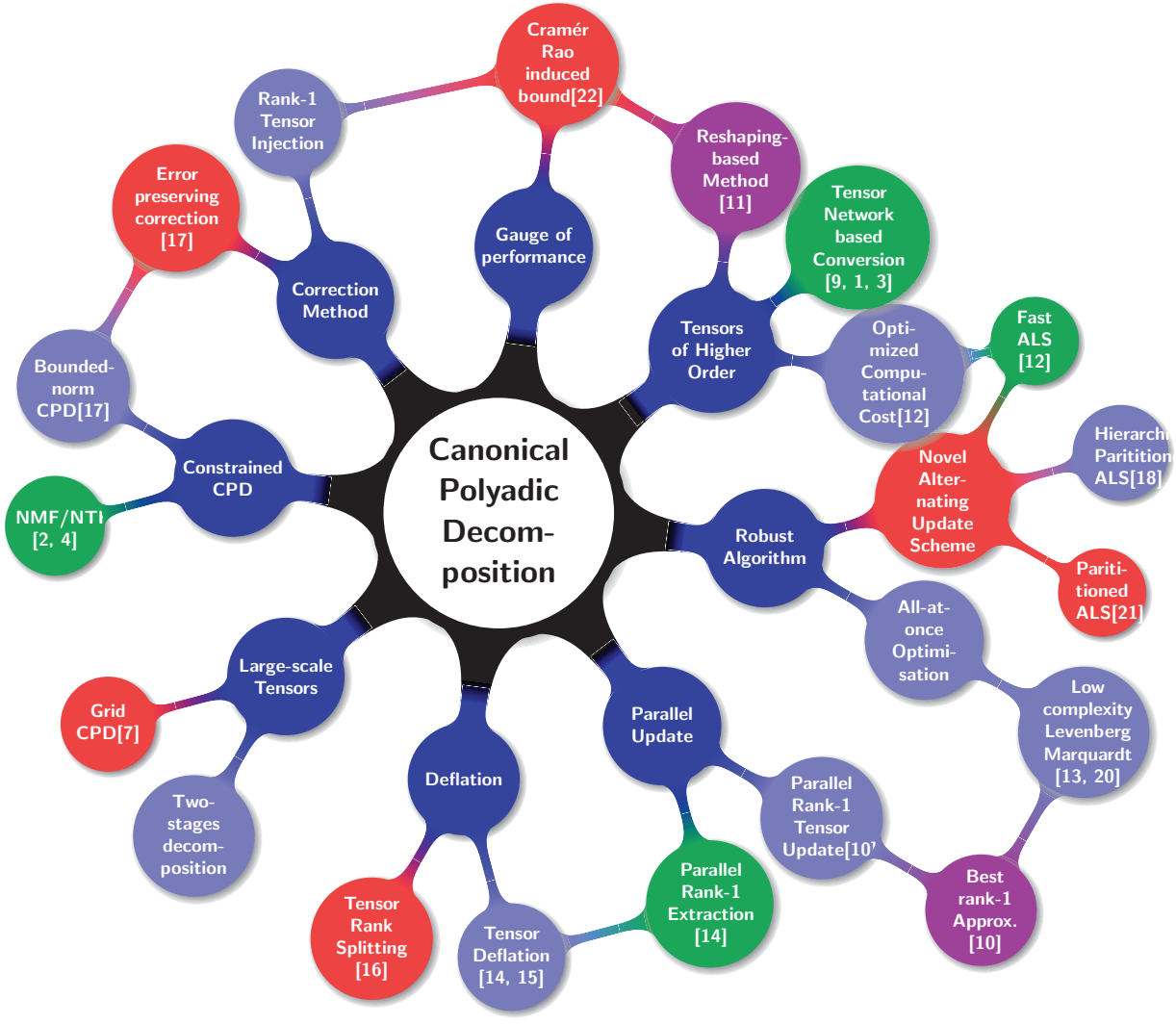
Prob.7: Decomposition of massive tensor data

Large-scale tensor cannot be handled in a computer.

Prob.8: Deflation (the most challenging problem)

Sequential extraction of rank-1 tensors of $\mathcal{Y}$?

- In matrix factorisations, e.g., SVD, rank-1 matrix components can be estimated via a deflation process.
- Unfortunately, this property does not hold for tensors because subtracting the best rank-1 tensor may increase the tensor rank (Stegeman and Comon 2010).



We deal with the main challenging problems for CPD and learn state-of-the-art algorithms for the problems.

Part II

Algorithms for CANDECOMP/-PARAFAC

Trilinear and Direct Trilinear Decompositions I

Generalized Rank Annihilation Method (GRAM) Booksh and Kowalski (1994); Booksh et al. (1994); Faber et al. (1994).

- Given $\mathcal{Y}$ a tensor of size $J \times J \times 2$ with only two frontal slices: $\mathbf{Y}_1$ and $\mathbf{Y}_2$.
  CPD of $\mathcal{Y}$ by $\mathbf{A} \in \mathbb{R}^{J \times J}$, $\mathbf{B} \in \mathbb{R}^{J \times J}$ and $\mathbf{C} = [\underline{\boldsymbol{c}}_1^T, \underline{\boldsymbol{c}}_2^T]^T \in \mathbb{R}^{2 \times J}$

$$\mathbf{Y}_1 = \mathbf{A}\,\mathbf{D}_1\,\mathbf{B}^T, \qquad \mathbf{Y}_2 = \mathbf{A}\,\mathbf{D}_2\,\mathbf{B}^T, \tag{1}$$

  where $\mathbf{D}_1 = \operatorname{diag}\left\{\underline{\boldsymbol{c}}_1\right\}$, $\mathbf{D}_2 = \operatorname{diag}\left\{\underline{\boldsymbol{c}}_2\right\}$.

Trilinear and Direct Trilinear Decompositions II

- Generalized Eigenvalue problem of two square matrices $\mathbf{Y}_1$ and $\mathbf{Y}_2$

$$\begin{aligned}\mathbf{Y}_1 (\mathbf{B}^T)^{-1} &= \mathbf{A}\,\mathbf{D}_1\,\mathbf{B}^T (\mathbf{B}^T)^{-1} = \mathbf{A}\,\mathbf{D}_2\,\mathbf{D}_2^{-1}\mathbf{D}_1 = \mathbf{A}\,\mathbf{D}_2\mathbf{B}^T(\mathbf{B}^T)^{-1}\mathbf{D} \\ &= \mathbf{Y}_2 (\mathbf{B}^T)^{-1}\mathbf{D}, \end{aligned} \tag{2}$$

  where $\mathbf{D} = \mathbf{D}_2^{-1}\mathbf{D}_1 = \text{diag}\{\underline{\boldsymbol{c}}_1 \oslash \underline{\boldsymbol{c}}_2\}$ is a diagonal matrix of generalized eigenvalues, and columns of $(\mathbf{B}^T)^{-1}$ are corresponding eigenvectors.

- Due to the scaling ambiguity, we can set the second row vector $\underline{\boldsymbol{c}}_2 = \mathbf{1}^T$, hence, the first row vector $\underline{\boldsymbol{c}}_1$ represents generalized eigenvalues:

$$\underline{\boldsymbol{c}}_1 = \text{diag}\{\mathbf{D}\}^T. \tag{3}$$

Trilinear and Direct Trilinear Decompositions III

Algorithm 1: GRAM

Input: $\underline{\mathbf{Y}}$: input data of size $J \times J \times 2$, J: number of basis components

Output: $\mathbf{A} \in \mathbb{R}^{J \times J}$, $\mathbf{B} \in \mathbb{R}^{J \times J}$ and $\mathbf{C} \in \mathbb{R}^{2 \times J}$ such that $\underline{\mathbf{Y}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$

1 **begin**

2 $\quad [\Phi, \mathbf{D}] = \mathtt{eig}(\mathbf{Y}_1, \mathbf{Y}_2)$;

3 $\quad \mathbf{B} = (\Phi^T)^{-1}$;

4 $\quad \mathbf{A} = \mathbf{Y}_2 \Phi$;

5 $\quad \mathbf{C} = [\mathrm{diag}\{\mathbf{D}\}, \mathbf{1}]^T$;

6 **end**

Trilinear and Direct Trilinear Decompositions IV

Direct Tri-Linear Decomposition (DTLD) (Sanchez and Kowalski)

Nonsquare matrices, $\mathbf{Y}_1$ and $\mathbf{Y}_2$, can be transformed to matrices $\mathbf{T}_q$ of size $J \times J$ by projecting on the two orthonormal matrices $\mathbf{U} \in \mathbb{R}^{I \times J}$ and $\mathbf{V} \in \mathbb{R}^{T \times J}$, usually via a singular value decomposition

$$\begin{aligned} [\mathbf{U}, \mathbf{S}, \mathbf{V}] &= \mathtt{svd}(\mathbf{Y}_1 + \mathbf{Y}_2, J) \\ \mathbf{T}_q &= \mathbf{U}^T \mathbf{Y}_q \mathbf{V}, \qquad (q = 1, 2). \end{aligned}$$

Applying the GRAM algorithm to the tensor $\mathcal{T}$, factors **A**, **B** and **C** can be found as

$$\begin{aligned} [\Phi, \mathbf{D}] &= \mathtt{eig}(\mathbf{T}_1, \mathbf{T}_2) && (4) \\ \mathbf{A} &= \mathbf{U}\mathbf{T}_2\Phi && (5) \\ \mathbf{B} &= \mathbf{V}(\Phi^T)^{-1} && (6) \\ \mathbf{C} &= [\mathrm{diag}\{\mathbf{D}\}, \mathbf{1}]^T. && (7) \end{aligned}$$

Trilinear and Direct Trilinear Decompositions V

The GRAM method is restricted to the tensor having only two slices.
For tensor with more than 2 slides, $\underline{\mathbf{Y}} \in \mathbb{R}^{I \times T \times Q}$ using the Tucker model with a core tensor $\underline{\mathbf{G}} \in \mathbb{R}^{J \times J \times 2}$

$$\underline{\mathbf{Y}} = [\![\underline{\mathbf{G}}; \mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t]\!], \tag{8}$$

where $\mathbf{A}_t \in \mathbb{R}^{I \times J}$, $\mathbf{B}_t \in \mathbb{R}^{T \times J}$, and $\mathbf{C}_t \in \mathbb{R}^{Q \times 2}$ are factors obtained by using the ALS Tucker algorithm.
The core tensor $\underline{\mathbf{G}}$ is then factorized using the GRAM algorithm

$$\underline{\mathbf{G}} = [\![\mathbf{A}_f, \mathbf{B}_f, \mathbf{C}_f]\!]. \tag{9}$$

The final model is given as $\underline{\mathbf{Y}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$ where

$$\mathbf{A} = \mathbf{A}_t \, \mathbf{A}_f \tag{10}$$
$$\mathbf{B} = \mathbf{B}_t \, \mathbf{B}_f \tag{11}$$
$$\mathbf{C} = \mathbf{C}_t \, \mathbf{C}_f. \tag{12}$$

Trilinear and Direct Trilinear Decompositions VI

Algorithm 2: DTLD

Input: $\underline{\mathbf{Y}}$: input data of size $I \times T \times Q$, J: number of basis components

Output: $\mathbf{A} \in \mathbb{R}^{I \times J}$, $\mathbf{B} \in \mathbb{R}^{T \times J}$ and $\mathbf{C} \in \mathbb{R}^{Q \times J}$ such that

$$\underline{\mathbf{Y}} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!]$$

begin

  $\underline{\mathbf{Y}} = [\![\underline{\mathbf{G}}; \mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t]\!]$ /* Tucker ALS */;

  $[\Phi, \mathbf{D}] = \mathtt{eig}(\mathbf{G}_1, \mathbf{G}_2)$;

  $\mathbf{A} = \mathbf{A}_t \, \mathbf{G}_2 \, \Phi$;

  $\mathbf{B} = \mathbf{B}_t \, (\Phi^T)^{-1}$;

  $\mathbf{C} = \mathbf{C}_t \, [\mathrm{diag}\{\mathbf{D}\}, \mathbf{1}]^T$ /* or $\mathbf{C} = \mathbf{Y}_{(3)} \, (\mathbf{B} \odot \mathbf{A})$ */;

end

Trilinear and Direct Trilinear Decompositions VII

DTLD

- Efficient initialization for inexact case or contrained CPD
- Work only for order-3 tensor
- rank-R must not exceed tensor dimensions

Generalized rank annihilation method for Higher Order CPD

Is there a closed form solution for CPD of tensors of higher order?

How???

The higher order tensor is first unfolded to be an order-3 tensor of size $I \times J \times K$ so that the two largest dimensions are greater than or equal to rank R, then factorized using DTLD.

Generalized rank annihilation method for Higher Order CPD

Is there a closed form solution for CPD of tensors of higher order?

How???

The higher order tensor is first unfolded to be an order-3 tensor of size $I \times J \times K$ so that the two largest dimensions are greater than or equal to rank R, then factorized using DTLD.

Generalized DTLD method for Higher Order CPD

Example

Given a rank-3 tensor, $\mathcal{Y}$, of size $4 \times 4 \times 5 \times 6$. Find its CPD $\mathcal{Y} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}]\!]$ using DTLD.

- Reshaping $\mathcal{Y}$ to an order-3 tensor, $\mathcal{Y}_{(3,4)}$, of size $4 \times 4 \times 30$ which is still of rank-3.
- Apply DTLD to $\mathcal{Y}_{(3,4)}$ and obtain a rank-3 CPD $\mathcal{Y}_{(3,4)} = [\![\mathbf{A}, \mathbf{B}, \mathbf{F}]\!]$
- Since $\mathbf{F} = \mathbf{D} \odot \mathbf{C}$ has Khatri-Rao product, $\boldsymbol{c}_r \boldsymbol{d}_r^T$ is the best rank-1 of the folding matrix, $\mathbf{F}_r$ of size 5×6 from $\boldsymbol{f}_r = \text{vec}(\mathbf{F}_r)$

$$\mathbf{F}_r \approx \boldsymbol{c}_r \boldsymbol{d}_r^T$$

using truncated SVD

Generalized DTLD method for Higher Order CPD

Example

Given a rank-3 tensor, $\mathcal{Y}$, of size $4 \times 4 \times 5 \times 6 \times 7$. Find its CPD $\mathcal{Y} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E}]\!]$ using DTLD.

- Reshaping $\mathcal{Y}$ to an order-3 tensor, $\mathcal{Y}_{(2,3),(4,5)}$, of size $4 \times 20 \times 42$ which is still of rank-3.
- Apply DTLD to $\mathcal{Y}_{(3,4)}$ and obtain a rank-3 CPD $\mathcal{Y}_{(2,3),(4,5)} = [\![\mathbf{A}, \mathbf{U}, \mathbf{V}]\!]$
- It can be shown that $\mathbf{U} = \mathbf{B} \odot \mathbf{C}$, $\mathbf{V} = \mathbf{E} \odot \mathbf{D}$, hence $\boldsymbol{b}_r \boldsymbol{c}_r^T = \mathtt{reshape}(\boldsymbol{u}_r, [45])$ is the best rank-1 of the matrix of size 4×5 folded from $\boldsymbol{u}_r$
  $\boldsymbol{d}_r \boldsymbol{e}_r^T = \mathtt{reshape}(\boldsymbol{v}_r, [67])$ is the best rank-1 of the matrix of size 6×7 folded from $\boldsymbol{v}_r$

Alternating Least Squares Algorithms

Cost function

$$D(\mathcal{Y}\|\hat{\mathcal{Y}}) = \frac{1}{2}\|\mathcal{Y} - \hat{\mathcal{Y}}\|_F^2 = \frac{1}{2}\|\mathbf{Y}_{(n)} - \mathbf{A}^{(n)}\{\mathbf{A}\}^{\odot_{-n}^T}\|_F^2.$$

1. Initialize all $\mathbf{A}^{(n)}$ randomly or by using SVD Berry et al. (2007), or by selected fibers from the data $\mathcal{Y}$.
2. Estimate $\mathbf{A}^{(n)}$ from the approximation by solving

$$\min_{\mathbf{A}^{(n)}} D_F(\mathcal{Y}\|\hat{\mathcal{Y}}) = \frac{1}{2}\|\mathcal{Y} - \hat{\mathcal{Y}}\|_F^2, \text{with fixed } \mathbf{A}^{(m)}, m \neq n.$$

3. For nonnegative components, set all negative elements of $\mathbf{A}^{(n)}$ to zero or a small positive value ε.
4. Estimate other factors until convergence.

Alternating Least Squares Algorithms

Cost function

$$D(\mathcal{Y}||\hat{\mathcal{Y}}) = \frac{1}{2}||\mathcal{Y} - \hat{\mathcal{Y}}||_F^2 = \frac{1}{2}||\mathbf{Y}_{(n)} - \mathbf{A}^{(n)}\{\mathbf{A}\}^{\odot_{-n}^T}||_F^2.$$

1. Initialize all $\mathbf{A}^{(n)}$ randomly or by using SVD Berry et al. (2007), or by selected fibers from the data $\mathcal{Y}$.
2. Estimate $\mathbf{A}^{(n)}$ from the approximation by solving

$$\min_{\mathbf{A}^{(n)}} D_F(\mathcal{Y}||\hat{\mathcal{Y}}) = \frac{1}{2}||\mathcal{Y} - \hat{\mathcal{Y}}||_F^2, \text{ with fixed } \mathbf{A}^{(m)}, m \neq n.$$

3. For nonnegative components, set all negative elements of $\mathbf{A}^{(n)}$ to zero or a small positive value ε.
4. Estimate other factors until convergence.

Alternating Least Squares Algorithms

Cost function

$$D(\mathcal{Y}||\hat{\mathcal{Y}}) = \frac{1}{2}||\mathcal{Y} - \hat{\mathcal{Y}}||_F^2 = \frac{1}{2}||\mathbf{Y}_{(n)} - \mathbf{A}^{(n)}\{\mathbf{A}\}^{\odot_{-n}^T}||_F^2.$$

1. Initialize all $\mathbf{A}^{(n)}$ randomly or by using SVD Berry et al. (2007), or by selected fibers from the data $\mathcal{Y}$.
2. Estimate $\mathbf{A}^{(n)}$ from the approximation by solving

$$\min_{\mathbf{A}^{(n)}} D_F(\mathcal{Y}||\hat{\mathcal{Y}}) = \frac{1}{2}||\mathcal{Y} - \hat{\mathcal{Y}}||_F^2, \text{ with fixed } \mathbf{A}^{(m)}, m \neq n.$$

3. For nonnegative components, set all negative elements of $\mathbf{A}^{(n)}$ to zero or a small positive value ε.
4. Estimate other factors until convergence.

Alternating Least Squares Algorithms

Cost function

$$D(\mathcal{Y}||\hat{\mathcal{Y}}) = \frac{1}{2}||\mathcal{Y} - \hat{\mathcal{Y}}||_F^2 = \frac{1}{2}||\mathbf{Y}_{(n)} - \mathbf{A}^{(n)}\{\mathbf{A}\}^{\odot_{-n}^T}||_F^2.$$

1. Initialize all $\mathbf{A}^{(n)}$ randomly or by using SVD Berry et al. (2007), or by selected fibers from the data $\mathcal{Y}$.
2. Estimate $\mathbf{A}^{(n)}$ from the approximation by solving

$$\min_{\mathbf{A}^{(n)}} D_F(\mathcal{Y}||\hat{\mathcal{Y}}) = \frac{1}{2}||\mathcal{Y} - \hat{\mathcal{Y}}||_F^2, \text{ with fixed } \mathbf{A}^{(m)}, m \neq n.$$

3. For nonnegative components, set all negative elements of $\mathbf{A}^{(n)}$ to zero or a small positive value ε.
4. Estimate other factors until convergence.

Gradient of the cost function w.r.t $\mathbf{A}^{(n)}$

$$\mathbf{G}^{(n)} = \mathbf{Y}_{(n)} \left(\bigodot_{k \neq n} \mathbf{A}^{(k)} \right) - \mathbf{A}^{(n)} \left(\circledast_{k \neq n} \mathbf{A}^{(k)T} \mathbf{A}^{(k)} \right) \in \mathbb{R}^{I_n \times R},$$

ALS Algorithm for CP

$$\mathbf{A}^{(n)} \leftarrow \mathbf{Y}_{(n)} \left(\bigodot_{k \neq n} \mathbf{A}^{(k)} \right) \left(\left\{ \mathbf{A}^T \mathbf{A} \right\}^{\circledast -n} \right)^{\dagger}$$

ε is a small constant (typically, 10^{-16}). $\mathbf{A}^{\dagger}$ is the Moore-Penrose inverse of $\mathbf{A}$,

Pros: Simple, work well for general data.

Cons: High cost due to computing $(\mathbf{A}^T \mathbf{A}) \circledast \forall_{n} \{\mathbf{A}\}^{(n)}$

not guaranteed to converge to a global minimum or even a stationary point, but only to a solution where the cost function ceases to decrease (Kolda and Bader (2009); Acar et al. (2010)).

relatively slow for nearly collinear data (Langville et al. (2012)).

Gradient of the cost function w.r.t $\mathbf{A}^{(n)}$

$$\mathbf{G}^{(n)} = \mathbf{Y}_{(n)} \left(\bigodot_{k \neq n} \mathbf{A}^{(k)} \right) - \mathbf{A}^{(n)} \left(\circledast_{k \neq n} \mathbf{A}^{(k)T} \mathbf{A}^{(k)} \right) \in \mathbb{R}^{I_n \times R},$$

ALS Algorithm for CP

$$\mathbf{A}^{(n)} \leftarrow \mathbf{Y}_{(n)} \left(\bigodot_{k \neq n} \mathbf{A}^{(k)} \right) \left(\left\{ \mathbf{A}^T \mathbf{A} \right\}^{\circledast -n} \right)^{\dagger}$$

ε is a small constant (typically, 10^{-16}). $\mathbf{A}^{\dagger}$ is the Moore-Penrose inverse of $\mathbf{A}$,

Pros: simple, work well for general data.

Cons: high cost due to computing $\{\mathbf{A}\}^{\odot -n}$, $\mathbf{Y}_{(n)} \{\mathbf{A}\}^{\odot -n}$

not guaranteed to converge to a global minimum or even a stationary point, but only to a solution where the cost functions cease to decrease Kolda and Bader (2009); Berry et al. (2007).

relatively slow for nearly collinear data Langville et al. (2006).

Gradient of the cost function w.r.t $\mathbf{A}^{(n)}$

$$\mathbf{G}^{(n)} = \mathbf{Y}_{(n)} \left(\bigodot_{k \neq n} \mathbf{A}^{(k)} \right) - \mathbf{A}^{(n)} \left(\circledast_{k \neq n} \mathbf{A}^{(k)T} \mathbf{A}^{(k)} \right) \in \mathbb{R}^{I_n \times R},$$

ALS Algorithm for CP

$$\mathbf{A}^{(n)} \leftarrow \mathbf{Y}_{(n)} \left(\bigodot_{k \neq n} \mathbf{A}^{(k)} \right) \left(\left\{ \mathbf{A}^T \mathbf{A} \right\}^{\circledast -n} \right)^{\dagger}$$

ε is a small constant (typically, 10^{-16}). $\mathbf{A}^{\dagger}$ is the Moore-Penrose inverse of $\mathbf{A}$,

Pros: simple, work well for general data.

Cons: high cost due to computing $\{\mathbf{A}\}^{\odot -n}$, $\mathbf{Y}_{(n)} \{\mathbf{A}\}^{\odot -n}$

not guaranteed to converge to a global minimum or even a stationary point, but only to a solution where the cost functions cease to decrease Kolda and Bader (2009); Berry et al. (2007).

relatively slow for nearly collinear data Langville et al. (2006).

Gradient of the cost function w.r.t $\mathbf{A}^{(n)}$

$$\mathbf{G}^{(n)} = \mathbf{Y}_{(n)} \left(\bigodot_{k \neq n} \mathbf{A}^{(k)} \right) - \mathbf{A}^{(n)} \left(\circledast_{k \neq n} \mathbf{A}^{(k)T} \mathbf{A}^{(k)} \right) \in \mathbb{R}^{I_n \times R},$$

ALS Algorithm for CP

$$\mathbf{A}^{(n)} \leftarrow \mathbf{Y}_{(n)} \left(\bigodot_{k \neq n} \mathbf{A}^{(k)} \right) \left(\left\{ \mathbf{A}^T \mathbf{A} \right\}^{\circledast -n} \right)^{\dagger}$$

ε is a small constant (typically, 10^{-16}). $\mathbf{A}^{\dagger}$ is the Moore-Penrose inverse of $\mathbf{A}$,

Pros: simple, work well for general data.

Cons: high cost due to computing $\{\mathbf{A}\}^{\odot -n}$, $\mathbf{Y}_{(n)} \{\mathbf{A}\}^{\odot -n}$
not guaranteed to converge to a global minimum or even a stationary point, but only to a solution where the cost functions cease to decrease Kolda and Bader (2009); Berry et al. (2007).
relatively slow for nearly collinear data Langville et al. (2006).

Fast Computation of CP gradient and FastALS I

CP gradients - the most expensive steps in CP algorithms.

Computation of CP gradients $\mathbf{G}^{(n)} = \mathbf{Y}_{(n)} \left(\bigodot_{k \neq n} \mathbf{A}^{(k)} \right)$ costs

$\mathcal{O}(NRI_1 I_2 \dots I_N)$.

Remark: Order of dimensions

For the least cost to compute $\mathbf{G}^{(n)}$, the dimensions of $\mathcal{Y}$ should be in the ascending or descending order.

Without loss of generality, we assume $I_1 \leq I_2 \leq \cdots \leq I_N$.

Fast Computation of CP gradient and FastALS II

Each column of $\mathbf{G}^{(n)}$ is efficiently computed from "left-to-right"

$$\boldsymbol{g}_r^{(n)} = \mathbf{Y}_{(n)} \left(\bigotimes_{k \neq n} \boldsymbol{a}_r^{(k)} \right) = \left(\mathcal{Y} \,\bar{\times}_{k=1}^{n-1}\, \boldsymbol{a}_r^{(k)} \right) \bar{\times}_{l=2}^{N-n+1}\, \boldsymbol{a}_r^{(l+n-1)},$$

or from "right-to-left"

$$\mathbf{Y}_{(n)} \left(\bigotimes_{k \neq n} \boldsymbol{a}_r^{(k)} \right) = \left(\mathcal{Y} \,\bar{\times}_{l=n+1}^{N}\, \boldsymbol{a}_r^{(l)} \right) \bar{\times}_{k=1}^{n-1}\, \boldsymbol{a}_r^{(k)}.$$

Let $\mathcal{L}^{(r,n)} \triangleq \mathcal{Y} \,\bar{\times}_{k=1}^{n-1}\, \boldsymbol{a}_r^{(k)}$ and $\mathcal{R}^{(r,n)} \triangleq \mathcal{Y} \,\bar{\times}_{k=n+1}^{N}\, \boldsymbol{a}_r^{(k)}$. Then

Recursive computation

$$\begin{aligned} \mathcal{L}^{(r,n)} &= \mathcal{Y} \bar{\times}_{k=1}^{n-1}\, \boldsymbol{a}_r^{(k)} = \mathcal{L}^{(r,n-1)} \,\bar{\times}_1\, \boldsymbol{a}_r^{(n-1)}, \\ \mathcal{R}^{(r,n)} &= \mathcal{Y} \,\bar{\times}_{k=n+1}^{N}\, \boldsymbol{a}_r^{(k)} = \mathcal{R}^{(r,n+1)} \,\bar{\times}_{n+1}\, \boldsymbol{a}_r^{(n+1)}, \end{aligned}$$

Fast Computation of CP gradient and FastALS III

Note that $\mathcal{L}^{(r,n)}$ and $\mathcal{R}^{(r,n)}$ are computed without tensor permutation

$$\mathcal{L}^{(r,n)} = \text{reshape}\left(\mathbf{Y}^T_{(1:n-1)}\left(\bigotimes_{k=1}^{n-1} \boldsymbol{a}_r^{(k)}\right), [I_n, \ldots, I_N]\right),$$

$$\mathcal{R}^{(r,n)} = \text{reshape}\left(\mathbf{Y}_{(1:n)}\left(\bigotimes_{k=n+1}^{N} \boldsymbol{a}_r^{(k)}\right), [I_1, \ldots, I_n]\right).$$

Fast CP-gradients over all modes

Fast Computation of CP gradient and FastALS IV

- Compute $J_n = I_1 I_2 \cdots I_n$ and $K_n = I_{n+1} I_{n+2} \cdots I_N$, and find $n^\star$ such that
$$n^\star = \max\{n : J_n \leq K_{n-1}\}$$
- First compute gradient $\mathbf{G}^{(n^\star)}$ (or $\mathbf{G}^{(n^\star+1)}$)
- Sequentially compute other gradients $\mathbf{G}^{(n)}$ in the following order $n = n^\star - 1, n^\star - 2, \ldots, 1, n^\star + 1, n^\star + 2, \ldots, N$.

Computational cost: $R\,(2J_N + 2J_{n^\star} + 2K_{n^\star})$

Fast CP gradients over all modes

Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$, N factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$

Output: $\mathbf{G}^{(n)} = \mathbf{Y}_{(n)} \left(\bigodot_{k \neq n} \mathbf{A}^{(k)} \right) : I_n \times R,\ n = 1, 2, \ldots, N$

begin

  $n^\star = \max\{n : J_n \leq K_{n-1}\}$ where $J_n = I_1 I_2 \ldots I_n, K_n = I_{n+1}$

  for $n = n^\star, n^\star - 1, \ldots, 1, n^\star + 1, n^\star + 2, \ldots, N$ **do**

    if $(n == n^\star)$ **then**

      $\mathbf{R}^{(n)}_{(n)} = \texttt{reshape}(\mathcal{Y}, [J_n, K_n]) \left(\bigodot_{k=n+1}^{N} \mathbf{A}^{(k)} \right)$;

      $\mathbf{G}^{(n)} = \texttt{cp\_gradient}(\mathcal{R}^{(n)}, \{\mathbf{A}\}, \textit{'right'})$

    else if $(n == n^\star + 1)$ **then**

      $\mathbf{L}^{(n)}_{(1)} = \left(\bigodot_{k=1}^{n-1} \mathbf{A}^{(k)} \right)^T \texttt{reshape}(\mathcal{Y}, [J_{n-1}, K_{n-1}])$

      $\mathbf{G}^{(n)} = \texttt{cp\_gradient}(\mathcal{L}^{(n)}, \{\mathbf{A}\}, \textit{'left'})$

    else if $(n < n^\star)$ **then**

      for $r = 1, 2, \ldots, R$ **do** $\text{vec}\left(\mathcal{R}^{(r,n)}\right) \leftarrow \texttt{reshape}(\mathcal{R}^{(r,n+1)}, [J_n, I_{n+1}])\, \boldsymbol{a}_r^{(n+1)}$;

      $\mathbf{G}^{(n)} = \texttt{cp\_gradient}(\mathcal{R}^{(n)}, \{\mathbf{A}\}, \textit{'right'})$

    else

      for $r = 1, 2, \ldots, R$ **do** $\text{vec}\left(\mathcal{L}^{(r,n)}\right) \leftarrow \texttt{reshape}(\mathcal{L}^{(r,n-1)}, [I_{n-1}, K_{n-1}])^T\, \boldsymbol{a}_r^{(n-1)}$;

      $\mathbf{G}^{(n)} = \texttt{cp\_gradient}(\mathcal{L}^{(n)}, \{\mathbf{A}\}, \textit{'left'})$

    end

  end

end

Fast CP gradients over all modes

Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$, N factor matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$

Output: $\mathbf{G}^{(n)} = \mathbf{Y}_{(n)} \left(\bigodot_{k \neq n} \mathbf{A}^{(k)} \right) : I_n \times R,\ n = 1, 2, \ldots, N$

begin

  $n^\star = \max\{n : J_n \leq K_{n-1}\}$ where $J_n = I_1 I_2 \ldots I_n, K_n = I_{n+1}$

  for $n = n^\star, n^\star - 1, \ldots, 1, n^\star + 1, n^\star + 2, \ldots, N$ **do**

    if $(n == n^\star)$ **then**

      $\mathbf{R}^{(n)}_{(n)} = \texttt{reshape}(\mathcal{Y}, [J_n, K_n]) \left(\bigodot_{k=n+1}^{N} \mathbf{A}^{(k)} \right)$;

      $\mathbf{G}^{(n)} = \texttt{cp\_gradient}(\mathcal{R}^{(n)}, \{\mathbf{A}\}, \textit{'right'})$

    else if $(n == n^\star + 1)$ **then**

      $\mathbf{L}^{(n)}_{(1)} = \left(\bigodot_{k=1}^{n-1} \mathbf{A}^{(k)} \right)^T \texttt{reshape}(\mathcal{Y}, [J_{n-1}, K_{n-1}])$

      $\mathbf{G}^{(n)} = \texttt{cp\_gradient}(\mathcal{L}^{(n)}, \{\mathbf{A}\}, \textit{'left'})$

    else if $(n < n^\star)$ **then**

      for $r = 1, 2, \ldots, R$ **do** $\text{vec}\left(\mathcal{R}^{(r,n)}\right) \leftarrow \texttt{reshape}(\mathcal{R}^{(r,n+1)}, [J_n, I_{n+1}])\, \boldsymbol{a}_r^{(n+1)}$;

      $\mathbf{G}^{(n)} = \texttt{cp\_gradient}(\mathcal{R}^{(n)}, \{\mathbf{A}\}, \textit{'right'})$

    else

      for $r = 1, 2, \ldots, R$ **do** $\text{vec}\left(\mathcal{L}^{(r,n)}\right) \leftarrow \texttt{reshape}(\mathcal{L}^{(r,n-1)}, [I_{n-1}, K_{n-1}])^T\, \boldsymbol{a}_r^{(n-1)}$;

      $\mathbf{G}^{(n)} = \texttt{cp\_gradient}(\mathcal{L}^{(n)}, \{\mathbf{A}\}, \textit{'left'})$

    end

  end

end

---

function $\mathbf{G}^{(n)} = \texttt{cp\_gradient}(\mathcal{Z}^{(n)}, \{\mathbf{A}\}, \textit{side})$

for $r = 1, 2, \ldots, R$ **do**

  switch *side* **do**

    case *'right'*: **do** $\boldsymbol{g}_r^{(n)} = \texttt{reshape}(\mathcal{Z}^{(r,n)}, [J_{n-1}, I_n])^T \left(\bigotimes_{k=1}^{n-1} \boldsymbol{a}_r^{(k)} \right)$

    case *'left'*: **do** $\boldsymbol{g}_r^{(n)} = \texttt{reshape}(\mathcal{Z}^{(r,n)}, [I_n, K_n]) \left(\bigotimes_{k=n+1}^{N} \boldsymbol{a}_r^{(k)} \right)$

  end

end

FastALS vs ALS for order-4 Tensors

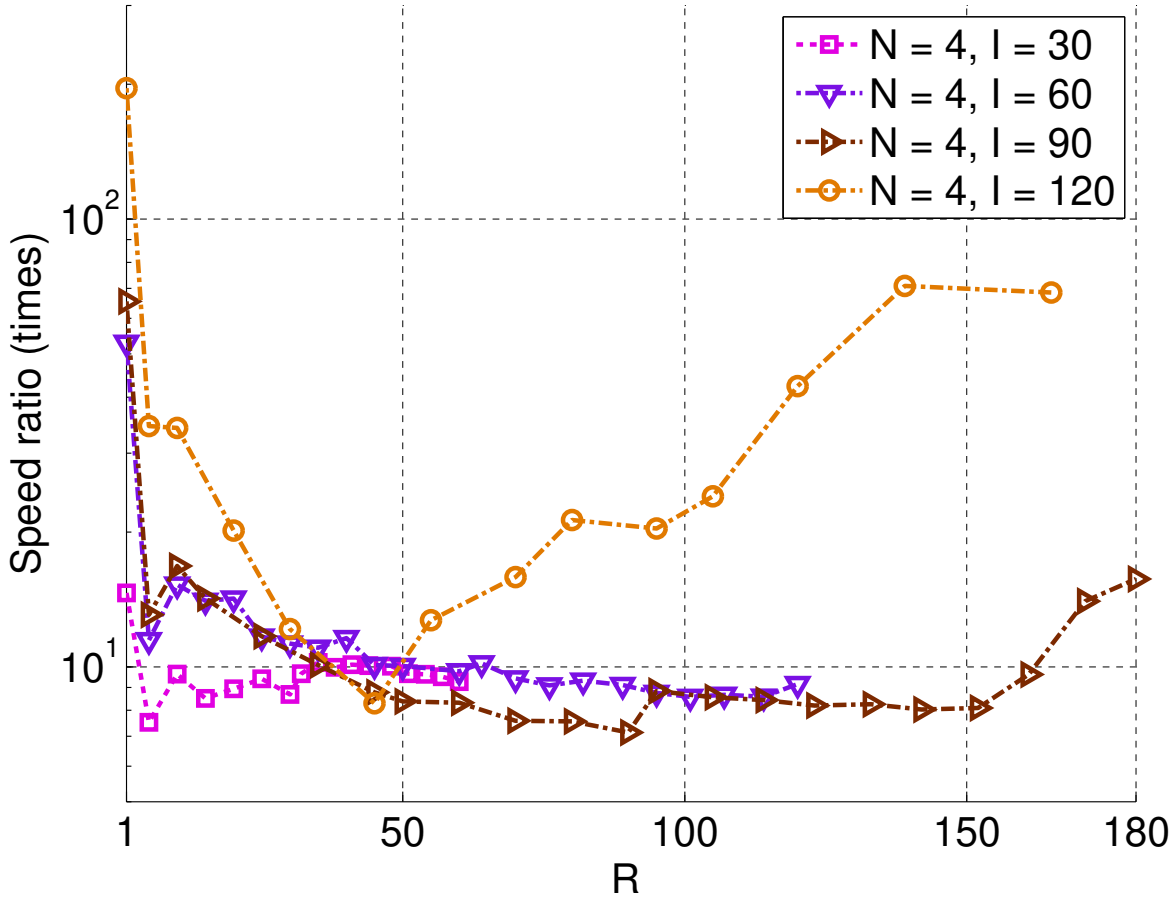
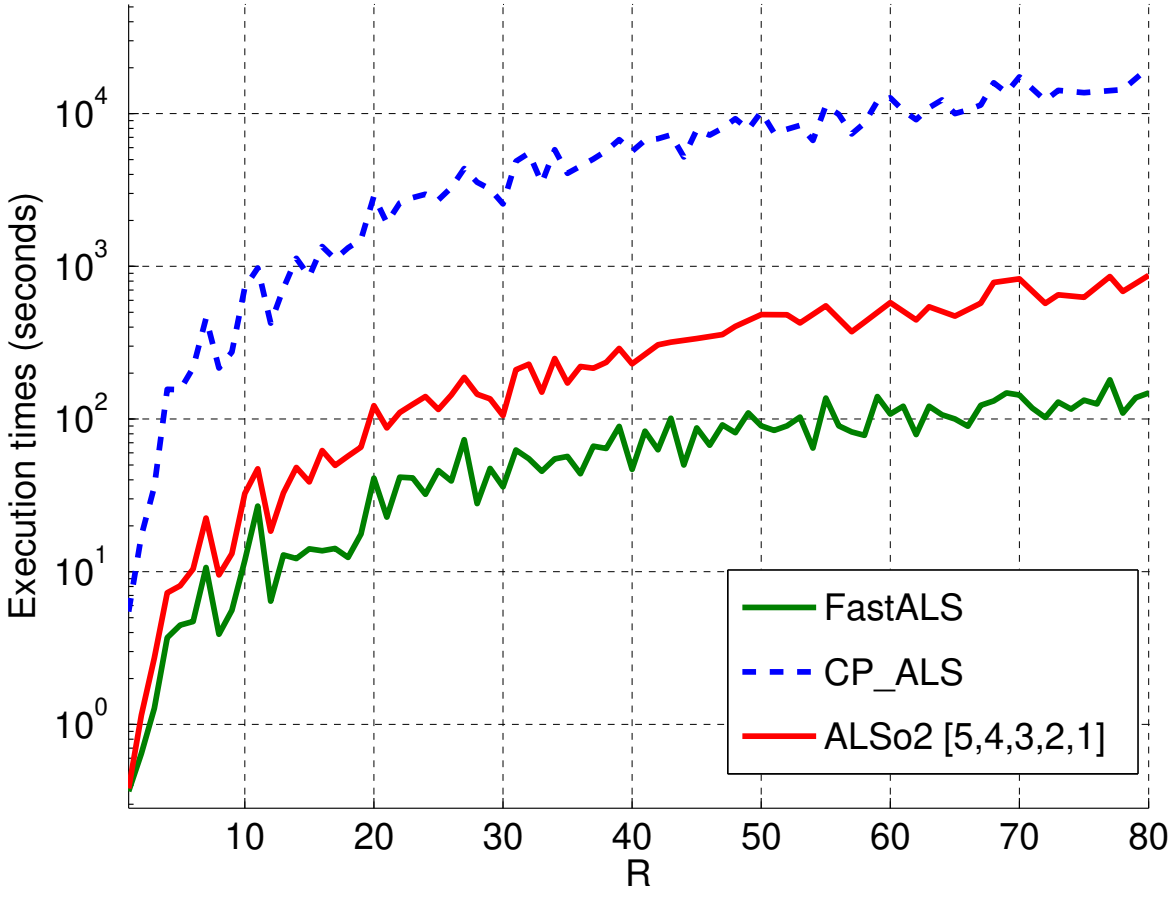


Figure: Speed-up ratios per iteration for the FastALS algorithm in comparison with the standard CP_ALS algorithm for factorizations of order-4 tensors with sizes $I_n = I$ for all n.

FastALS vs ALS for order-5 Tensors



ALSo2 Eigenvector (2012)

Figure: Execution times of CP algorithms in factorization of an order-5 EEG MI tensor of size 2 *dictionaries* $\times$ 23 *frequency bins* (8-30 Hz) $\times$ 50 *time frames* $\times$ 62 *channels* $\times$ 200 *trials*.

Line search I

Bro's Line search (LS)

- Predict the factors $\mathbf{A}^{(n)}$ from their previous estimates

$$\mathbf{A}_t^{(n)} = \mathbf{A}_{t-1}^{(n)} + \eta \left(\mathbf{A}_{t-1}^{(n)} - \mathbf{A}_{t-2}^{(n)} \right), \quad n = 1, 2, \ldots, N, \tag{13}$$

  where t denotes the iteration index, and η is stepsize.

- A heuristic stepsize η is iteratively searched to improve the fit

$$\eta \leftarrow \alpha \, \eta, \tag{14}$$

  where α is a suitably chosen scalar constant.

- The LS method can find a suitable step value but not an optimal step.

Line search II

Line Search for order-3 CPD

$$
\begin{aligned}
\hat{\mathcal{Y}} &= [\![\mathbf{A}_{t+1}, \mathbf{B}_{t+1}, \mathbf{C}_{t+1}]\!] \\
&= [\![\mathbf{A}_t + \eta\Delta\mathbf{A}_t, \mathbf{B}_t + \eta\Delta\mathbf{B}_t, \mathbf{C}_t + \eta\Delta\mathbf{C}_t]\!] \\
&= [\![\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t]\!] \\
&\quad +\eta([\![\Delta\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t]\!] + [\![\mathbf{A}_t, \Delta\mathbf{B}_t, \mathbf{C}_t]\!] + [\![\mathbf{A}_t, \mathbf{B}_t, \Delta\mathbf{C}_t]\!]) \\
&\quad +\eta^2([\![\Delta\mathbf{A}_t, \Delta\mathbf{B}_t, \mathbf{C}_t]\!] + [\![\Delta\mathbf{A}_t, \mathbf{B}_t, \Delta\mathbf{C}_t]\!] + [\![\mathbf{A}_t, \Delta\mathbf{B}_t, \Delta\mathbf{C}_t]\!]) \\
&\quad +\eta^3[\![\Delta\mathbf{A}_t, \Delta\mathbf{B}_t, \Delta\mathbf{C}_t]\!] \\
&= [\![\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t]\!] \\
&\quad +\eta[\![[\Delta\mathbf{A}_t, \mathbf{A}_t, \mathbf{A}_t], [\mathbf{B}_t, \Delta\mathbf{B}_t, \mathbf{B}_t], [\mathbf{C}_t, \mathbf{C}_t, \Delta\mathbf{C}_t]]\!] \\
&\quad +\eta^2[\![[\Delta\mathbf{A}_t, \Delta\mathbf{A}_t, \mathbf{A}_t], [\Delta\mathbf{B}_t, \mathbf{B}_t, \Delta\mathbf{B}_t], [\mathbf{C}_t, \Delta\mathbf{C}_t, \Delta\mathbf{C}_t]]\!] \\
&\quad +\eta^3[\![\Delta\mathbf{A}_t, \Delta\mathbf{B}_t, \Delta\mathbf{C}_t]\!]
\end{aligned}
$$

Line search III

Exact line search

Enhanced or Exact Line Search (ELS) Rajih et al. (2008) finds optimal step values at every even iteration by minimizing the modified cost function

$$D(\mathcal{Y}\|\hat{\mathcal{Y}}) = \frac{1}{2}\left\|\mathcal{Y} - [\![\mathbf{A}_t^{(1)} + \eta_1 \Delta\mathbf{A}_t^{(1)}, \mathbf{A}_t^{(2)} + \eta_2 \Delta\mathbf{A}_t^{(2)}, \mathbf{A}_t^{(3)} + \eta_2 \Delta\mathbf{A}_t^{(3)}]\!]\right\|_F^2,$$

where η_1, η_2, η_3 are stepsizes, and $\Delta\mathbf{A}_t^{(n)} = \mathbf{A}_{t-1}^{(n)} - \mathbf{A}_{t-2}^{(n)}$.
When $\eta_1 = \eta_2 = \eta_3 = \eta$, ELS find roots of a polynomial $\boldsymbol{p}(\mu)$ of degree 5. An optimal stepsize is the root of that polynomial $\boldsymbol{p}(\mu)$ which yields the smallest cost function value.

- ELS alleviates bottlenecks more efficiently than LS.
- But less efficient if the directions provided by the algorithm are bad Comon et al. (2009).
- High computational cost for higher order tensors, due to building up $(2N-1)$- order polynomials $\boldsymbol{p}(\eta)$.

Line search: Example

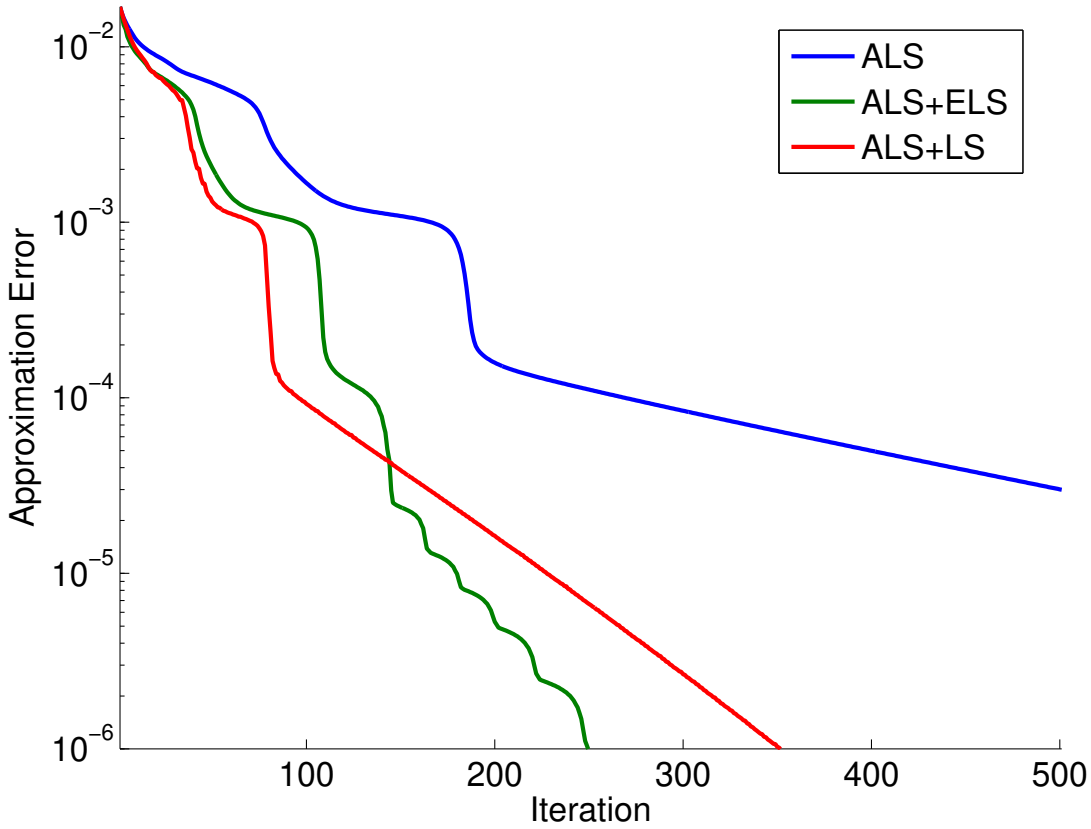


Figure: Approximation errors of the ALS with/without line searches for factorizations of tensors of size $30 \times 30 \times 30$ and rank $R = 10$ composed from highly correlated components which have $\boldsymbol{a}_r^{(n)T} \boldsymbol{a}_s^{(n)} \in [0.8, 0.999]$, $r \neq s$.

Rank-one Update Algorithm

Algorithm updates only one component during iterations.

Assume component $\boldsymbol{a}_r^{(n)}$ to be updated, all other components are fixed. $\hat{\mathcal{Y}}$ is split into two parts as follows

- A rank-one tensor $\mathcal{Y}^{(r)}$ is built up from components $\boldsymbol{a}_r^{(n)}$

$$\mathcal{Y}^{(r)} = \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)} \tag{15}$$

- A rank-$(R-1)$ CP tensor $\mathcal{Y}^{(-r)}$ is composed by N factors that is

$$\mathbf{A}^{(n)} = \left[\boldsymbol{a}_1^{(n)}, \ldots, \boldsymbol{a}_{r-1}^{(n)}, \boldsymbol{a}_{r+1}^{(n)}, \ldots, \boldsymbol{a}_R^{(n)}\right]$$

$$\mathcal{Y}^{(-r)} = \sum_{k \neq r} \boldsymbol{a}_k^{(1)} \circ \boldsymbol{a}_k^{(2)} \circ \cdots \circ \boldsymbol{a}_k^{(N)}$$

$$= \llbracket \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)} \rrbracket \tag{16}$$

Rank-one Update Algorithm

Algorithm updates only one component during iterations.

Assume component $\boldsymbol{a}_r^{(n)}$ to be updated, all other components are fixed. $\hat{\mathcal{Y}}$ is split into two parts as follows

Rank-one Update Algorithm

Algorithm updates only one component during iterations.
Assume component $\boldsymbol{a}_r^{(n)}$ to be updated, all other components are fixed. $\hat{\mathcal{Y}}$ is split into two parts as follows

- A rank-one tensor $\hat{\mathcal{Y}}^{(r)}$ is built up from components $\boldsymbol{a}_r^{(n)}$

$$\hat{\mathcal{Y}}^{(r)} = \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}. \tag{15}$$

- A rank-(R-1) CP tensor $\hat{\mathcal{Y}}^{(-r)}$ is composed by N factors $\mathbf{A}_{-r}^{(N)} = \left[\boldsymbol{a}_1^{(n)}, \ldots, \boldsymbol{a}_{r-1}^{(n)}, \boldsymbol{a}_{r+1}^{(n)}, \ldots, \boldsymbol{a}_R^{(n)}\right]$, that is

$$\begin{aligned} \hat{\mathcal{Y}}^{(-r)} &= \sum_{k \neq r} \boldsymbol{a}_k^{(1)} \circ \boldsymbol{a}_k^{(2)} \circ \cdots \circ \boldsymbol{a}_k^{(N)} \\ &= \mathcal{J} \times_1 \mathbf{A}_{-r}^{(1)} \times_2 \mathbf{A}_{-r}^{(2)} \cdots \times_N \mathbf{A}_{-r}^{(N)}. \end{aligned} \tag{16}$$

Rank-one Update Algorithm

Algorithm updates only one component during iterations. Assume component $\boldsymbol{a}_r^{(n)}$ to be updated, all other components are fixed. $\hat{\mathcal{Y}}$ is split into two parts as follows

- A rank-one tensor $\hat{\mathcal{Y}}^{(r)}$ is built up from components $\boldsymbol{a}_r^{(n)}$

$$\hat{\mathcal{Y}}^{(r)} = \boldsymbol{a}_r^{(1)} \circ \boldsymbol{a}_r^{(2)} \circ \cdots \circ \boldsymbol{a}_r^{(N)}. \tag{15}$$

- A rank-(R-1) CP tensor $\hat{\mathcal{Y}}^{(-r)}$ is composed by N factors $\mathbf{A}_{-r}^{(N)} = \left[\boldsymbol{a}_1^{(n)}, \ldots, \boldsymbol{a}_{r-1}^{(n)}, \boldsymbol{a}_{r+1}^{(n)}, \ldots, \boldsymbol{a}_R^{(n)}\right]$, that is

$$\begin{aligned} \hat{\mathcal{Y}}^{(-r)} &= \sum_{k \neq r} \boldsymbol{a}_k^{(1)} \circ \boldsymbol{a}_k^{(2)} \circ \cdots \circ \boldsymbol{a}_k^{(N)} \\ &= \mathcal{I} \times_1 \mathbf{A}_{-r}^{(1)} \times_2 \mathbf{A}_{-r}^{(2)} \cdots \times_N \mathbf{A}_{-r}^{(N)}. \end{aligned} \tag{16}$$

Rank-one update

- Approximation $\mathcal{Y} = \hat{\mathcal{Y}}^{(-r)} + \hat{\mathcal{Y}}^{(r)} + \mathcal{E}$.
- Define a residual tensor $\tilde{\mathcal{Y}}^{(r)} = \mathcal{Y} - \hat{\mathcal{Y}}^{(-r)} = \hat{\mathcal{Y}}^{(r)} + \mathcal{E}$.

Update rule for $\boldsymbol{a}_r^{(n)}$

$$\boldsymbol{a}_r^{(n)} \leftarrow \tilde{\boldsymbol{Y}}_{(n)}^{(r)} \{\boldsymbol{a}_r\}^{\odot_{-n}} \left(\left\{\boldsymbol{a}_r^T \boldsymbol{a}_r\right\}^{\circledast_{-n}}\right)^{\dagger} = \frac{\tilde{\mathcal{Y}}^{(r)} \bar{\times}_{-n} \{\boldsymbol{a}_r\}}{\gamma_r^{(n)}}$$

$$\gamma_r^{(n)} = \left\{\boldsymbol{a}_r^T \boldsymbol{a}_r\right\}^{\circledast_{-n}} = \begin{cases} \boldsymbol{a}_r^{(N)T} \boldsymbol{a}_r^{(N)}, & n \neq N \\ 1, & n = N. \end{cases}$$

Due to normalization $\boldsymbol{a}_r^{(n)} = \boldsymbol{a}_r^{(n)} / \|\boldsymbol{a}_r^{(n)}\|_2$ for $n = 1, 2, \ldots N-1$ after each iteration step, scaling factor $\gamma_r^{(n)}$ can be omitted.

Rank-one update

- Approximation $\mathcal{Y} = \hat{\mathcal{Y}}^{(-r)} + \hat{\mathcal{Y}}^{(r)} + \mathcal{E}$.
- Define a residual tensor $\tilde{\mathcal{Y}}^{(r)} = \mathcal{Y} - \hat{\mathcal{Y}}^{(-r)} = \hat{\mathcal{Y}}^{(r)} + \mathcal{E}$.

Update rule for $\boldsymbol{a}_r^{(n)}$

$$\boldsymbol{a}_r^{(n)} \leftarrow \tilde{\mathbf{Y}}_{(n)}^{(r)} \{\boldsymbol{a}_r\}^{\odot -n} \left(\left\{\boldsymbol{a}_r^T \boldsymbol{a}_r\right\}^{\circledast -n}\right)^{\dagger} = \frac{\tilde{\mathcal{Y}}^{(r)} \bar{\times}_{-n} \{\boldsymbol{a}_r\}}{\gamma_r^{(n)}}$$

$$\gamma_r^{(n)} = \left\{\boldsymbol{a}_r^T \boldsymbol{a}_r\right\}^{\circledast -n} = \begin{cases} \boldsymbol{a}_r^{(N)T} \boldsymbol{a}_r^{(N)}, & n \neq N \\ 1, & n = N. \end{cases}$$

Due to normalization $\boldsymbol{a}_r^{(n)} = \boldsymbol{a}_r^{(n)} / \|\boldsymbol{a}_r^{(n)}\|_2$ for $n = 1, 2, \ldots N-1$ after each iteration step, scaling factor $\gamma_r^{(n)}$ can be omitted.

Rank-one update

- Approximation $\mathcal{Y} = \hat{\mathcal{Y}}^{(-r)} + \hat{\mathcal{Y}}^{(r)} + \mathcal{E}$.
- Define a residual tensor $\tilde{\mathcal{Y}}^{(r)} = \mathcal{Y} - \hat{\mathcal{Y}}^{(-r)} = \hat{\mathcal{Y}}^{(r)} + \mathcal{E}$.

Update rule for $\boldsymbol{a}_r^{(n)}$

$$\boldsymbol{a}_r^{(n)} \leftarrow \tilde{\mathbf{Y}}_{(n)}^{(r)} \left\{\boldsymbol{a}_r\right\}^{\odot_{-n}} \left(\left\{\boldsymbol{a}_r^T \boldsymbol{a}_r\right\}^{\circledast_{-n}}\right)^{\dagger} = \frac{\tilde{\mathcal{Y}}^{(r)} \bar{\times}_{-n} \left\{\boldsymbol{a}_r\right\}}{\gamma_r^{(n)}}$$

$$\gamma_r^{(n)} = \left\{\boldsymbol{a}_r^T \boldsymbol{a}_r\right\}^{\circledast_{-n}} = \begin{cases} \boldsymbol{a}_r^{(N)T} \boldsymbol{a}_r^{(N)}, & n \neq N \\ 1, & n = N. \end{cases}$$

Due to normalization $\boldsymbol{a}_r^{(n)} = \boldsymbol{a}_r^{(n)} / \|\boldsymbol{a}_r^{(n)}\|_2$ for $n = 1, 2, \ldots N-1$ after each iteration step, scaling factor $\gamma_r^{(n)}$ can be omitted.

Efficient version

$$\begin{aligned} \boldsymbol{a}_r^{(n)} &\leftarrow \tilde{\mathcal{Y}}^{(r)} \bar{\times}_{-n} \{\boldsymbol{a}_r\} = \mathcal{Y} \bar{\times}_{-n} \{\boldsymbol{a}_r\} - \mathcal{Y}^{(-r)} \bar{\times}_{-n} \{\boldsymbol{a}_r\} \\ &= \mathcal{Y} \bar{\times}_{-n} \{\boldsymbol{a}_r\} - \mathcal{I} \times_n \mathbf{A}_{-r}^{(n)} \bar{\times}_{-n} \left\{\mathbf{A}_{-r}^T \boldsymbol{a}_r\right\} \\ &= \mathcal{Y} \bar{\times}_{-n} \{\boldsymbol{a}_r\} - \mathbf{A}_{-r}^{(n)} \left\{\mathbf{A}_{-r}^T \boldsymbol{a}_r\right\}^{\circledast -n}, \end{aligned}$$

where $\left\{\mathbf{A}_{-r}^T \boldsymbol{a}_r\right\} \triangleq \left\{\mathbf{A}_{-r}^{(1)^T} \boldsymbol{a}_r^{(1)}, \cdots, \mathbf{A}_{-r}^{(N)^T} \boldsymbol{a}_r^{(N)}\right\}$.

Note

- $\mathcal{Y} \bar{\times}_{-n} \{\boldsymbol{a}_r\}$ does not demand significant storage
- The ALS rule computes $\{\mathbf{A}^{(n)^T} \mathbf{A}^{(n)}\}^{\circledast -n}$ resulting in large-scale matrix $R^{I_n \times R}$

Efficient version

$$\begin{aligned} \boldsymbol{a}_r^{(n)} &\leftarrow \tilde{\mathcal{Y}}^{(r)} \bar{\times}_{-n} \{\boldsymbol{a}_r\} = \mathcal{Y} \bar{\times}_{-n} \{\boldsymbol{a}_r\} - \mathcal{Y}^{(-r)} \bar{\times}_{-n} \{\boldsymbol{a}_r\} \\ &= \mathcal{Y} \bar{\times}_{-n} \{\boldsymbol{a}_r\} - \mathcal{I} \times_n \mathbf{A}_{-r}^{(n)} \bar{\times}_{-n} \left\{\mathbf{A}_{-r}^T \boldsymbol{a}_r\right\} \\ &= \mathcal{Y} \bar{\times}_{-n} \{\boldsymbol{a}_r\} - \mathbf{A}_{-r}^{(n)} \left\{\mathbf{A}_{-r}^T \boldsymbol{a}_r\right\}^{\circledast_{-n}}, \end{aligned}$$

where $\left\{\mathbf{A}_{-r}^T \boldsymbol{a}_r\right\} \triangleq \left\{\mathbf{A}_{-r}^{(1)^T} \boldsymbol{a}_r^{(1)}, \cdots, \mathbf{A}_{-r}^{(N)^T} \boldsymbol{a}_r^{(N)}\right\}$.

Note

- $\mathcal{Y} \bar{\times}_{-n} \{\boldsymbol{a}_r\}$ does not demand significant storage.
- The ALS rule computes $\{\mathbf{A}\}^{\odot_{-n}}$ resulting a large-scale matrix $I^{N-1} \times R$.

Efficient version

$$\begin{aligned} \boldsymbol{a}_r^{(n)} &\leftarrow \tilde{\mathcal{Y}}^{(r)} \,\bar{\times}_{-n}\, \{\boldsymbol{a}_r\} = \mathcal{Y} \,\bar{\times}_{-n}\, \{\boldsymbol{a}_r\} - \mathcal{Y}^{(-r)} \,\bar{\times}_{-n}\, \{\boldsymbol{a}_r\} \\ &= \mathcal{Y} \,\bar{\times}_{-n}\, \{\boldsymbol{a}_r\} - \mathcal{I} \times_n \mathbf{A}_{-r}^{(n)} \bar{\times}_{-n} \left\{\mathbf{A}_{-r}^T \boldsymbol{a}_r\right\} \\ &= \mathcal{Y} \,\bar{\times}_{-n}\, \{\boldsymbol{a}_r\} - \mathbf{A}_{-r}^{(n)} \left\{\mathbf{A}_{-r}^T \boldsymbol{a}_r\right\}^{\circledast_{-n}}, \end{aligned}$$

where $\left\{\mathbf{A}_{-r}^T \boldsymbol{a}_r\right\} \triangleq \left\{\mathbf{A}_{-r}^{(1)^T} \boldsymbol{a}_r^{(1)}, \cdots, \mathbf{A}_{-r}^{(N)^T} \boldsymbol{a}_r^{(N)}\right\}$.

Note

- $\mathcal{Y} \,\bar{\times}_{-n}\, \{\boldsymbol{a}_r\}$ does not demand significant storage.
- The ALS rule computes $\{\mathbf{A}\}^{\odot_{-n}}$ resulting a large-scale matrix $I^{N-1} \times R$.

Algorithm 5: Rank-1 Update Algorithm for CP and NTF

begin
  Initialize all $\mathbf{A}^{(n)}$
  Normalize all $\boldsymbol{a}_{r_n}^{(n)}$ to unit length
  repeat
    for $n = 1$ *to* N **do** /* Update $\mathbf{A}^{(n)}$ */
      for $r = 1$ *to* R **do**
        $\boldsymbol{a}_r^{(n)} \leftarrow \mathcal{Y} \bar{\times}_{-n} \{\boldsymbol{a}_r\} - \mathbf{A}_{-r}^{(n)} \left\{\mathbf{A}_{-r}^T \boldsymbol{a}_r\right\}^{\circledast -n}$
        $\boldsymbol{a}_r^{(n)} \leftarrow \left[\boldsymbol{a}_r^{(n)}\right]_+$ /* Retrieve nonnegative entries */
        if $n \neq N$ **then** /* Normalize and fix scaling */
          $\boldsymbol{a}_r^{(N)} = \frac{\boldsymbol{a}_r^{(N)} \|\boldsymbol{a}_r^{(n)}\|_2}{\|\boldsymbol{a}_r^{(N)}\|_2^2}, \quad \boldsymbol{a}_r^{(n)} = \frac{\boldsymbol{a}_r^{(n)}}{\|\boldsymbol{a}_r^{(n)}\|_2}$
        end
      end
    end
  until *a stopping criterion is met*
end

Part III

Higher order CPD

High order CPD

- Most algorithms for order-3 CPD can be straightforwardly extended to higher order tensors.
- However, the algorithms become more complicated, computationally demanding, and often not applicable to relatively large scale tensors.
  - Complexity of CP gradients grows linearly with the tensor order N with a cost of $O\left(NR\prod_{n=1}^{N} I_n\right)$.
  - More time consuming due to more tensor unfoldings $\mathbf{Y}_{(n)}$ $(n = 2, 3, \ldots, N-1)$.
  - Line search extrapolation methodsHarshman (1970); Andersson and Bro (2000); Tomasi (2006); Rajih et al. (2008) become more complicated, and high computationally demanding.
- In practice, CPD algorithms may take a lot of iterations, and are very slow, when some factor matrices become nearly rank deficient Comon et al. (2009).

NOTE

An approach for high dimensional large scale tensor is prerequisite.

The First Example

- Given an order-4 tensor $\mathcal{Y} = \mathcal{I} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{A}^{(3)} \times_4 \mathbf{A}^{(4)}$.
- An unfolding tensor $\mathcal{Y}_{[\![1,2,(3,4)]\!]}$ is given by

$$\mathcal{Y}_{[\![1,2,(3,4)]\!]} = \mathcal{I} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \mathbf{B}$$

  where $\mathbf{B} = \mathbf{A}^{(4)} \odot \mathbf{A}^{(3)}$.
- Since CPD is unique under "mild" conditions, one can perform higher order CPD through a lower order CPD.
  For example, $\boldsymbol{a}_r^{(3)}$ and $\boldsymbol{a}_r^{(4)}$ are the best rank-one approximation of $\texttt{reshape}(\boldsymbol{b}_r, [I_3, I_4])$ using SVD.

Definition (**Reshaping)**

The reshape operator for a tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ to a size specified by a vector $\boldsymbol{L} = [L_1, L_2, \ldots, L_M]$ with $\prod_{m=1}^{M} L_m = \prod_{n=1}^{N} I_n$ returns an order-M tensor $\mathcal{X}$, such that $\text{vec}(\mathcal{Y}) = \text{vec}(\mathcal{X})$, and is expressed as $\mathcal{X} = \texttt{reshape}(\mathcal{Y}, \boldsymbol{L}) \in \mathbb{R}^{L_1 \times L_2 \times \cdots \times L_M}$.

Definition (**Tensor transposition** Ragnarsson and Loan (2012))

If $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and $\boldsymbol{p}$ is a permutation of $[1, 2, \ldots, N]$, then $\boldsymbol{p}$-transpose of $\mathcal{A}$ is defined by

$$\mathcal{A}^{<\boldsymbol{p}>}(i_{p_1}, \ldots, i_{p_N}) = \mathcal{A}(i_1, \ldots, i_N), \quad \mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{I} = [I_1, I_2, \ldots, I_N].$$

Definition (**Generalized tensor unfolding)**

Reshaping a $\boldsymbol{p}$-transpose $\mathcal{Y}^{<\boldsymbol{p}>}$ to an order-M tensor of size $\boldsymbol{L} = [L_1, L_2, \ldots, L_M]$ with $L_m = \prod_{k \in \boldsymbol{I}_m} I_k$, where

$[\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_M] \equiv [p_1, p_2, \ldots, p_N]$, $\boldsymbol{I}_m = [\boldsymbol{I}_m(1), \ldots, \boldsymbol{I}_m(K_m)]$

$$\mathcal{Y}_{[\![\boldsymbol{I}]\!]} \triangleq \texttt{reshape}(\mathcal{Y}^{<\boldsymbol{p}>}, \boldsymbol{L}), \qquad \boldsymbol{I} = [\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_M].$$

Higher order CPD through Tensor UnFolding

Simple unFolding CPD (FCP)

Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$, rank R,
Unfolding $\boldsymbol{I} = [\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_M]$ where $\boldsymbol{I}_m = [I_m(1), \ldots, I_m(K_m)]$, $M \geq 3$
Output: $\boldsymbol{\lambda} \in \mathbb{R}^N$, N matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$

begin

/* **Stage 1:** Optional compression of unfolding tensor $\mathcal{Y}_{[\![\boldsymbol{I}]\!]}$ */

$[\![\mathcal{G}, \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(M)}]\!] = \texttt{Tucker\_decomposition}(\mathcal{Y}_{[\![\boldsymbol{I}]\!]}, \min(\boldsymbol{I}, R))$

/* Unfolding $\mathcal{Y}$ to be a lower order tensor $\mathcal{Y}_{[\![\boldsymbol{I}]\!]}$ and */
/* Perform multilinear low-rank approximation to $\mathcal{Y}_{[\![\boldsymbol{I}]\!]}$ using High order SVD or Higher Order Orthogonal Iteration. */

end

Higher order CPD through Tensor UnFolding

Simple unFolding CPD (FCP)

Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$, rank R,
Unfolding $\boldsymbol{I} = [\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_M]$ where $\boldsymbol{I}_m = [I_m(1), \ldots, I_m(K_m)]$, $M \geq 3$

Output: $\boldsymbol{\lambda} \in \mathbb{R}^N$, N matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$

begin

/* **Stage 1:** Optional compression of unfolding tensor $\mathcal{Y}_{[\![\boldsymbol{I}]\!]}$ */

$[\![\mathcal{G}, \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(M)}]\!] = \mathtt{TD}(\mathcal{Y}_{[\![\boldsymbol{I}]\!]}, \min(\boldsymbol{I}, R))$

/* **Stage 2:** CPD of $\mathcal{G}$ and back projection------ */

$[\![\boldsymbol{\lambda}; \mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(M)}]\!] = \mathtt{CPD}(\mathcal{G}, R)$

for $m = 1, 2, \ldots, M$ **do** $\mathbf{B}^{(m)} \leftarrow \mathbf{U}^{(m)}\, \mathbf{B}^{(m)}$

end

Higher order CPD through Tensor UnFolding

Simple unFolding CPD (FCP)

Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$, rank R,
Unfolding $\boldsymbol{I} = [\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_M]$ where $\boldsymbol{I}_m = [I_m(1), \ldots, I_m(K_m)]$, $M \geq 3$

Output: $\boldsymbol{\lambda} \in \mathbb{R}^N$, N matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$

begin

  /* **Stage 1:** Optional compression of unfolding tensor $\mathcal{Y}_{[\![\boldsymbol{I}]\!]}$ */

  $[\![\mathcal{G}, \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(M)}]\!] = \mathtt{TD}(\mathcal{Y}_{[\![\boldsymbol{I}]\!]}, \min(\boldsymbol{I}, R))$

  /* **Stage 2:** CPD of $\mathcal{G}$ and back projection------ */

  $[\![\boldsymbol{\lambda}; \mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(M)}]\!] = \mathtt{CPD}(\mathcal{G}, R)$

  for $m = 1, 2, \ldots, M$ **do** $\mathbf{B}^{(m)} \leftarrow \mathbf{U}^{(m)}\, \mathbf{B}^{(m)}$

  /* **Stage 3:** Estimate factors for high order tensors- */

  for $m = 1, 2, \ldots, M$ **do**

    for $r = 1, 2, \ldots, R$ **do**

      $\mathcal{B}_r = \mathtt{reshape}(\boldsymbol{b}_r^{(m)}, [I_{I_m(1)}, \ldots, I_{I_m(K_m)}])$

      $[\![g; \boldsymbol{a}_r^{(I_m(1))}, \ldots, \boldsymbol{a}_r^{(I_m(K_m))}]\!] = \mathtt{TD}(\mathcal{B}_r, 1)$

      $\lambda_r \leftarrow \lambda_r\, g$

    end

Higher order CPD through Tensor UnFolding

Simple unFolding CPD (FCP)

Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$, rank R,
Unfolding $\boldsymbol{I} = [\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_M]$ where $\boldsymbol{I}_m = [I_m(1), \ldots, I_m(K_m)]$, $M \geq 3$

Output: $\boldsymbol{\lambda} \in \mathbb{R}^N$, N matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$

begin

/* **Stage 1:** Optional compression of unfolding tensor $\mathcal{Y}_{[\![\boldsymbol{I}]\!]}$ */

$[\![\mathcal{G}, \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(M)}]\!] = \mathtt{TD}(\mathcal{Y}_{[\![\boldsymbol{I}]\!]}, \min(\boldsymbol{I}, R))$

/* **Stage 2:** CPD of $\mathcal{G}$ and back projection------ */

$[\![\boldsymbol{\lambda}; \mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(M)}]\!] = \mathtt{CPD}(\mathcal{G}, R)$

for $m = 1, 2, \ldots, M$ **do** $\mathbf{B}^{(m)} \leftarrow \mathbf{U}^{(m)}\, \mathbf{B}^{(m)}$

/* **Stage 3:** Estimate factors for high order tensors- */

for $m = 1, 2, \ldots, M$ **do**

for $r = 1, 2, \ldots, R$ **do**

$\mathcal{B}_r = \mathtt{reshape}(\boldsymbol{b}_r^{(m)}, [I_{m(1)}, \ldots$

$[\![g; \boldsymbol{a}_r^{(I_m(1))}, \ldots, \boldsymbol{a}_r^{(I_m(K_m))}]\!] = \mathtt{TD}(\mathcal{B}_r, 1)$

$\lambda_r \leftarrow \lambda_r\, g$

end

Loop over all columns of matrices $\mathbf{B}^{(m)}$

reshape them to tensors/matri-
ces

Higher order CPD through Tensor UnFolding

Simple unFolding CPD (FCP)

Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$, rank R,
Unfolding $\boldsymbol{I} = [\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_M]$ where $\boldsymbol{I}_m = [I_m(1), \ldots, I_m(K_m)]$, $M \geq 3$

Output: $\boldsymbol{\lambda} \in \mathbb{R}^N$, N matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$

begin

/* **Stage 1:** Optional compression of unfolding tensor $\mathcal{Y}_{[\![\boldsymbol{I}]\!]}$ */

$[\![\mathcal{G}, \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(M)}]\!] = \mathtt{TD}(\mathcal{Y}_{[\![\boldsymbol{I}]\!]}, \min(\boldsymbol{I}, R))$

/* **Stage 2:** CPD of $\mathcal{G}$ and back projection------ */

$[\![\boldsymbol{\lambda}; \mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(M)}]\!] = \mathtt{CPD}(\mathcal{G}, R)$

for $m = 1, 2, \ldots, M$ **do** $\mathbf{B}^{(m)} \leftarrow \mathbf{U}^{(m)}\, \mathbf{B}^{(m)}$

/* **Stage 3:** Estimate factors for high order tensors- */

for $m = 1, 2, \ldots, M$ **do**

for $r = 1, 2, \ldots, R$ **do**

$\mathcal{B}_r = \mathtt{reshape}(\boldsymbol{b}_r^{(m)}, [I_{l_m(1)}, \ldots, I_{l_m(K_m)}])$

$[\![g; \boldsymbol{a}_r^{(l_m(1))}, \ldots, \boldsymbol{a}_r^{(l_m(K_m))}]\!] = \mathtt{TD}(\mathcal{B}_r, 1)$

$\lambda_r \leftarrow \lambda_r\, g$

end

Rank-one tensor approximation

Higher order CPD through Tensor UnFolding

Simple unFolding CPD (FCP)

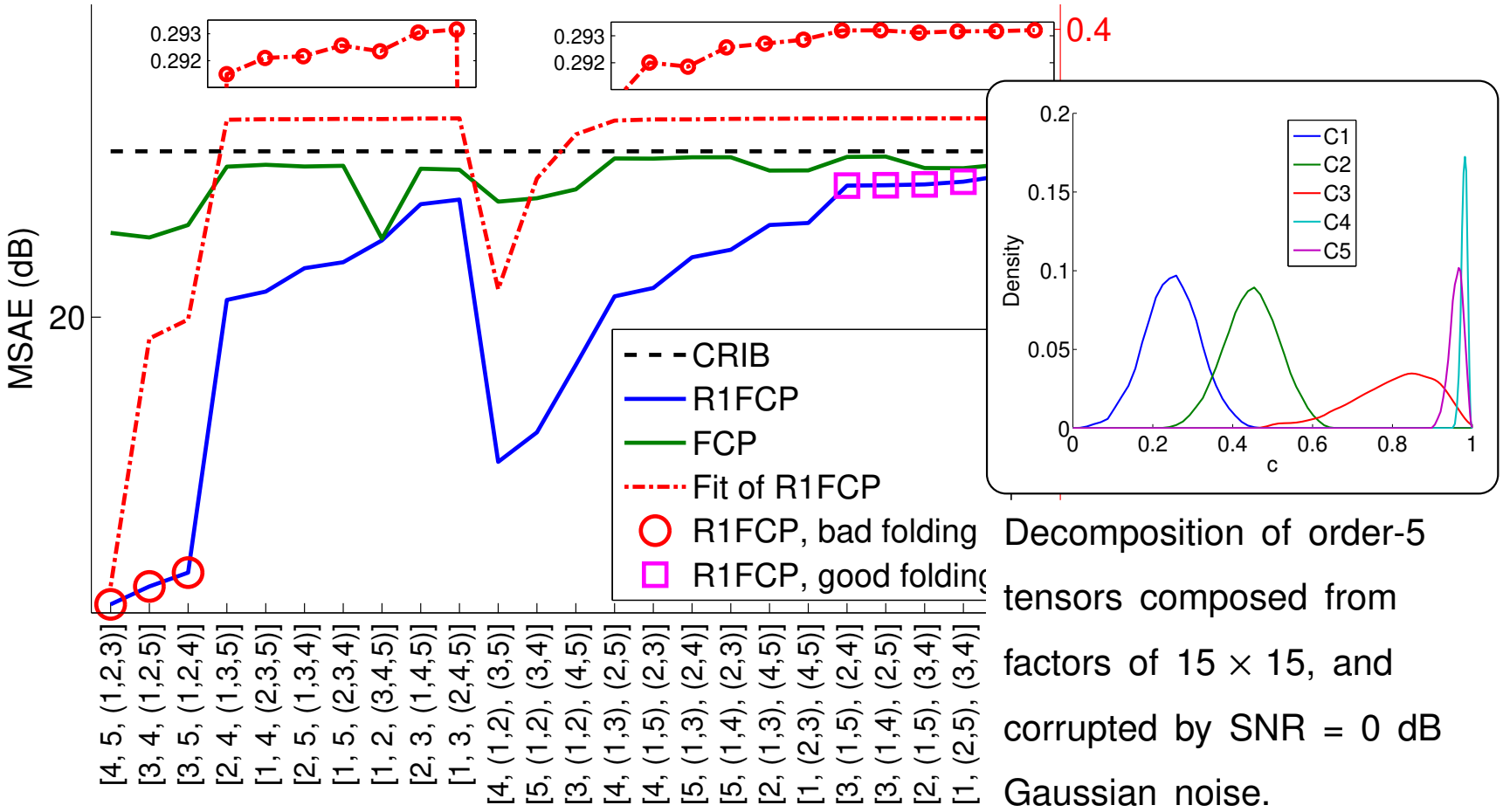
Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$, rank R,
Unfolding $\boldsymbol{I} = [\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_M]$ where $\boldsymbol{I}_m = [I_m(1), \ldots, I_m(K_m)]$, $M \geq 3$
Output: $\boldsymbol{\lambda} \in \mathbb{R}^N$, N matrices $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$

Simple or Rank-one FCP

- Sensitive to unfolding rules.
- Worst performance if unfolding modes having low-collinearity.

/* Stage 3: Estimate factors for high order tensors- */
for $m = 1, 2, \ldots, M$ **do**
  for $r = 1, 2, \ldots, R$ **do**
    $\mathcal{B}_r = \text{reshape}(\boldsymbol{b}_r^{(m)}, [I_{I_m(1)}, \ldots, I_{I_m(K_m)}])$
    $[\![g; \boldsymbol{a}_r^{(I_m(1))}, \ldots, \boldsymbol{a}_r^{(I_m(K_m))}]\!] = \text{TD}(\mathcal{B}_r, 1)$
    $\lambda_r \leftarrow \lambda_r g$
  end

Loss of Accuracy of the simple FCP



Decomposition of order-5 tensors composed from factors of 15×15, and corrupted by SNR = 0 dB Gaussian noise.

Figure: R1FCP caused high loss of accuracy when combining modes 1 and 2 with low collinearity degrees.

The Second Example I

- Consider a rank-R CPD and a simple unfolding $\boldsymbol{I} = [1, 2, (3, 4)]$

$$\begin{aligned} \mathcal{Y} &= [\![\boldsymbol{\lambda}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \mathbf{A}^{(3)}, \mathbf{A}^{(4)}]\!] + \mathcal{E}_1, \quad \boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_R] \\ \mathcal{Y}_{[\![\boldsymbol{I}]\!]} &= [\![\boldsymbol{\mu}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \mathbf{B}^{(3)}]\!] + \mathcal{E}_2, \quad \boldsymbol{\mu} = [\mu_1, \ldots, \mu_R] \end{aligned}$$

- If the noise variance is low, $\boldsymbol{\lambda} \approx \boldsymbol{\mu}, \quad \mathbf{A}^{(k)} \approx \mathbf{B}^{(k)}$, for $k = 1, 2$.
- Put $\mathbf{F}_r = \texttt{reshape}(\boldsymbol{b}_r^{(3)}, [I_3, I_4])$, and let $\mathbf{F}_r = \mathbf{U}_r \Sigma_r \mathbf{V}_r^T$ be Singular Value Decomposition of $\mathbf{F}_r$

$$\begin{aligned} \mathbf{U}_r &= [\boldsymbol{u}_{r1}, \ldots, \boldsymbol{u}_{rT}], \qquad \mathbf{V}_r = [\boldsymbol{v}_{r1}, \ldots, \boldsymbol{v}_{rT}] \\ \boldsymbol{\sigma}_r &= [\sigma_{r1}, \sigma_{r2}, \ldots, \sigma_{rT}], \quad 1 \geq \sigma_{r1} \geq \sigma_{r2} \geq \cdots \geq \sigma_{rT} \geq 0 \,. \end{aligned}$$

- Theoretically, $\mathbf{F}_r$ has rank one, and

$$\mathcal{Y} \approx \mathcal{Y}_R = [\![\boldsymbol{\mu}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \widetilde{\mathbf{B}}^{(N-1)}, \widetilde{\mathbf{B}}^{(N)}]\!]$$

where $\widetilde{\mathbf{B}}^{(N-1)} = [\boldsymbol{u}_{11}, \ldots, \boldsymbol{u}_{R1}]$ and $\widetilde{\mathbf{B}}^{(N)} = [\boldsymbol{v}_{11}, \ldots, \boldsymbol{v}_{R1}]$.

The Second Example II

- In general, a numerical rank of $\mathbf{F}_r$ is $J_r \geq 1$, $r = 1, \ldots, R$ defined as $\sum_{j=1}^{J_r} \sigma_{rj}^2 \geq \tau$ where $0 < \tau < 1$, e.g., $\tau = 0.98$.
- We can write a rank-J tensor approximation of $\mathcal{Y}$, $J = \sum_{r=1}^{R} J_r$

$$\begin{aligned} \mathcal{Y}_J &= \sum_{r=1}^{R} \mu_r \, \boldsymbol{b}_r^{(1)} \circ \boldsymbol{b}_r^{(2)} \circ \left(\sum_{j=1}^{J_r} \sigma_{rj} \, (\boldsymbol{u}_{rj} \circ \boldsymbol{v}_{rj}) \right) \\ &= [\![\widetilde{\lambda}; \widetilde{\mathbf{A}}^{(1)}, \widetilde{\mathbf{A}}^{(2)}, \widetilde{\mathbf{A}}^{(3)}, \widetilde{\mathbf{A}}^{(4)}]\!] \end{aligned}$$

where

$$\begin{aligned} \widetilde{\mathbf{A}}^{(k)} &= \mathbf{B}^{(k)} \mathbf{M} \quad \text{for} \quad k = 1, 2 \\ \widetilde{\mathbf{A}}^{(3)} &= [\mathbf{U}_1, \ldots, \mathbf{U}_R] \\ \widetilde{\mathbf{A}}^{(4)} &= [\mathbf{V}_1, \ldots, \mathbf{V}_R] \\ \mathbf{M} &= \text{bdiag}(\mathbf{1}_{1 \times J_1}, \ldots, \mathbf{1}_{1 \times J_R}) \\ \widetilde{\lambda} &= [\mu_1 \sigma_{11}, \ldots, \mu_1 \sigma_{1J_1}, \ldots, \mu_R \sigma_{R1}, \ldots, \mu_R \sigma_{RJ_R}]. \end{aligned}$$

The Second Example III

- The desired rank–R approximation of $\mathcal{Y}$ can be obtained by applying a structured CPD algorithm to $\mathcal{Y}_J$ using $\mathcal{Y}_R$ as an initial value.
- The method can be easily extended for higher order tensor, and multimode unfoldings.

FCP

Input: Data tensor $\mathcal{Y}$: $(I_1 \times I_2 \times \cdots \times I_N)$, rank R, threshold $\tau(\geq 0.98)$
Unfolding rule $\boldsymbol{I} = [\boldsymbol{I}_1, \boldsymbol{I}_2, \ldots, \boldsymbol{I}_M]$, $\boldsymbol{I}_m = [I_m(1), I_m(2), \ldots, I_m(K_m)]$
Output: Rank-R Kruskal tensor $\mathcal{Y}_R = [\![\lambda; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!]$

begin

/* **Stage 1:** Tensor unfolding and compression-------- */

$[\![\mathcal{G}, \mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(M)}]\!] = \mathrm{TD}(\mathcal{Y}_{[\![\boldsymbol{I}]\!]}, \min(\boldsymbol{I}, R))$

/* **Stage 2:** CPD of $\mathcal{G}$ and back projection --------- */

$[\![\lambda; \mathbf{B}^{(1)}, \ldots, \mathbf{B}^{(M)}]\!] = \mathrm{CPD}(\mathcal{G}, R)$

$\mathcal{Y}_R = [\![\lambda; \mathbf{U}^{(1)}\mathbf{B}^{(1)}, \ldots, \mathbf{U}^{(M)}\mathbf{B}^{(M)}]\!]$ /* rank-R K-tensor */

/* **Stage 3:** Sequential estimation ------------ */

foreach *group* $\boldsymbol{I}_m$ *with* $K_m \geq 2$ **do**

$n = K_1 + \cdots + K_{m-1}$

for $k = 1, 2, \ldots, K_m$ **do**

/* **Stage 3a:** Construction of rank-J K-tensor---- */

$[\mathcal{Y}_J, \mathcal{Y}_R] = \mathrm{twomodeFCP}(\mathcal{Y}_R, n + k, I_{I_m(k)})$

/* **Stage 3b:** Rank-J to rank-R K-tensor------ */

if $J > R$ **then** $\mathcal{Y}_R = \mathrm{structuredCPD}(\mathcal{Y}_J, R, \mathcal{Y}_R)$

end

end

/* **Stage 4:** Refinement if needed ------------ */

$\mathcal{Y}_R = \mathrm{CPD}(\mathcal{Y}, R, \mathcal{Y}_R)$

end

function $[\mathcal{Y}_J, \mathcal{Y}_R] = \texttt{twomodeFCP}(\mathcal{Y}_R, n, T)$

Input: Order-N rank-R Kruskal tensor $\mathcal{Y}_R = [\![\lambda; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}]\!]$, with $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R}$ to be reconstructed

Output: Two order-$(N+1)$ rank-J and rank-R K-tensors $\mathcal{Y}_J$, $\mathcal{Y}_R$

for $r = 1, 2, \ldots, R$ **do**

$\quad \mathbf{F}_r = \texttt{reshape}(\boldsymbol{a}_r^{(n)}, [T, I_n/T])$

$\quad \mathbf{F}_r \approx \mathbf{U}_r \, \mathrm{diag}(\boldsymbol{\sigma}_r) \, \mathbf{V}_r^T$ /* rank-J_r truncated SVD, $\|\boldsymbol{\sigma}_r\|_2^2 \geq \tau$ */

end

/* Construct a rank-J K-tensor, $J = J_1 + J_2 + \cdots + J_R$ ------ */

$\mathbf{M} = \mathrm{bdiag}(\mathbf{1}_{1\times J_1}, \ldots, \mathbf{1}_{1\times J_R})$

$\widetilde{\lambda} = [\lambda_1 \boldsymbol{\sigma}_1, \lambda_2 \boldsymbol{\sigma}_2, \ldots, \lambda_R \boldsymbol{\sigma}_R]$, $\mathbf{G} = [\mathbf{U}_1, \ldots, \mathbf{U}_R], \mathbf{H} = [\mathbf{V}_1, \ldots, \mathbf{V}_R]$

$\mathcal{Y}_J = [\![\widetilde{\lambda}; \mathbf{A}^{(1)}\mathbf{M}, \ldots, \mathbf{A}^{(n-1)}\mathbf{M}, \mathbf{G}, \mathbf{H}, \mathbf{A}^{(n+1)}\mathbf{M}, \ldots, \mathbf{A}^{(N)}\mathbf{M}]\!]$

/* Construct a rank-R Kruskal tensor ---------- */

$\lambda \leftarrow [\lambda_1 \sigma_{11}, \lambda_2 \sigma_{21}, \ldots, \lambda_R \sigma_{R1}], \mathbf{G} = [\boldsymbol{u}_{11}, \ldots, \boldsymbol{u}_{R1}], \mathbf{H} = [\boldsymbol{v}_{11}, \ldots, \boldsymbol{v}_{R1}]$

$\mathcal{Y}_R = [\![\lambda; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(n-1)}, \mathbf{G}, \mathbf{H}, \mathbf{A}^{(n+1)}, \ldots, \mathbf{A}^{(N)}]\!]$

Decomposition of the ORL face database

- An order-5 Gabor feature tensor of size $16 \times 16 \times 8 \times 4 \times 400$ was constructed from the ORL faces for 8 different orientations at 4 scales.
- $\mathcal{Y}$ was unfolded to be an order-3 tensor using $\boldsymbol{I} = [1, 2, (3, 4, 5)]$.
- The factor $\mathbf{A}^{(5)} \in \mathbb{R}^{400 \times R}$ comprised compressed features which were used to cluster faces using the K-means algorithm.

R	Algorithm	Fit (%)	Time (seconds)	Ratio $\frac{\text{ALS}}{\text{FCP}}$	ACC (%)	NMI (%)
30	FCP	60.59	24	**39**	85.00	92.91
	ALS	60.56	927		85.25	93.22
40	FCP	62.46	39	**41**	84.25	92.57
	ALS	62.63	1599		85.50	93.68
60	FCP	65.47	105	**162**	83.00	91.62
	ALS	65.64	16962		81.38	91.44

Part IV

Damped Gauss-Newton Algorithms

Damped Gauss-Newton Algorithm ?

$$
\begin{aligned}
D &= \|\mathcal{Y} - \mathcal{J} \times_1 \mathbf{A}^{(1)} \cdots \times_N \mathbf{A}^{(N)}\|_F^2 + P_l(\{\mathbf{A}^{(n)}\}) + P_s(\{\mathbf{A}^{(n)}\}), \\
P_l &= -\sum_{n=1}^{N} \alpha_n \sum_{i_n=1}^{I_n} \sum_{r=1}^{R} \log(a_{i_n r}^{(n)}), \quad P_s = \sum_{n=1}^{N} \beta_n \left\| \mathbf{A}^{(n)} \right\|_1,
\end{aligned}
$$

$\alpha_n \geq 0, \beta_n \geq 0 \; \forall n.$

Damped Gauss-Newton update rule

$$
\mathbf{a} \leftarrow \mathbf{a} - (\mathbf{H} + \mu \mathbf{I})^{-1} \mathbf{g},
$$

$\mathbf{a}$ $= \left[\mathrm{vec}\left(\mathbf{A}^{(1)}\right)^T \mathrm{vec}\left(\mathbf{A}^{(2)}\right)^T \cdots \mathrm{vec}\left(\mathbf{A}^{(N)}\right)^T \right]^T$

$\mathbf{g}$ gradient $R \sum I_n \times 1$

$\mathbf{H}$ approximate Hessian $R \sum I_n \times R \sum I_n$

$\mu > 0$: damping parameter,

Damped Gauss-Newton Algorithm ?

$$
\begin{aligned}
D &= \|\mathcal{Y} - \mathcal{J} \times_1 \mathbf{A}^{(1)} \cdots \times_N \mathbf{A}^{(N)}\|_F^2 + P_l(\{\mathbf{A}^{(n)}\}) + P_s(\{\mathbf{A}^{(n)}\}), \\
P_l &= -\sum_{n=1}^{N} \alpha_n \sum_{i_n=1}^{I_n} \sum_{r=1}^{R} \log(a_{i_n r}^{(n)}), \quad P_s = \sum_{n=1}^{N} \beta_n \left\|\mathbf{A}^{(n)}\right\|_1,
\end{aligned}
$$

$\alpha_n \geq 0, \beta_n \geq 0 \; \forall n.$

Damped Gauss-Newton update rule

$$
\mathfrak{a} \leftarrow \mathfrak{a} - \left(\mathbf{H} + \mu \mathbf{I}\right)^{-1} \boldsymbol{g},
$$

$$
\mathfrak{a} = \left[\text{vec}\left(\mathbf{A}^{(1)}\right)^T \text{vec}\left(\mathbf{A}^{(2)}\right)^T \cdots \text{vec}\left(\mathbf{A}^{(N)}\right)^T\right]^T
$$

$\boldsymbol{g}$ gradient $R \sum I_n \times 1$

$\mathbf{H}$ approximate Hessian $R \sum I_n \times R \sum I_n$

$\mu > 0$: damping parameter,

Damped Gauss-Newton Algorithm ?

$$
\begin{aligned}
D &= \|\mathcal{Y} - \mathcal{J} \times_1 \mathbf{A}^{(1)} \cdots \times_N \mathbf{A}^{(N)}\|_F^2 + P_l(\{\mathbf{A}^{(n)}\}) + P_s(\{\mathbf{A}^{(n)}\}), \\
P_l &= -\sum_{n=1}^{N} \alpha_n \sum_{i_n=1}^{I_n} \sum_{r=1}^{R} \log(a_{i_n r}^{(n)}), \quad P_s = \sum_{n=1}^{N} \beta_n \left\|\mathbf{A}^{(n)}\right\|_1,
\end{aligned}
$$

$\alpha_n \geq 0, \beta_n \geq 0 \; \forall n.$

Damped Gauss-Newton update rule

$$
\mathfrak{a} \leftarrow \mathfrak{a} - \left(\mathbf{H} + \mu \mathbf{I}\right)^{-1} \boldsymbol{g},
$$

$$
\begin{aligned}
\mathfrak{a} &= \left[\mathrm{vec}\left(\mathbf{A}^{(1)}\right)^T \mathrm{vec}\left(\mathbf{A}^{(2)}\right)^T \cdots \mathrm{vec}\left(\mathbf{A}^{(N)}\right)^T\right]^T \\
\boldsymbol{g} &\quad \text{gradient } R \sum I_n \times 1 \\
\mathbf{H} &\quad \text{approximate Hessian } R \sum I_n \times R \sum I_n \\
\mu > 0 : &\quad \text{damping parameter,}
\end{aligned}
$$

Damped Gauss-Newton Algorithm ?

$$
\begin{aligned}
D &= \|\mathcal{Y} - \mathcal{J} \times_1 \mathbf{A}^{(1)} \cdots \times_N \mathbf{A}^{(N)}\|_F^2 + P_l(\{\mathbf{A}^{(n)}\}) + P_s(\{\mathbf{A}^{(n)}\}), \\
P_l &= -\sum_{n=1}^{N} \alpha_n \sum_{i_n=1}^{I_n} \sum_{r=1}^{R} \log(a_{i_n r}^{(n)}), \quad P_s = \sum_{n=1}^{N} \beta_n \left\|\mathbf{A}^{(n)}\right\|_1,
\end{aligned}
$$

$\alpha_n \geq 0, \beta_n \geq 0 \; \forall n.$

Damped Gauss-Newton update rule

$$
\mathfrak{a} \leftarrow \mathfrak{a} - (\mathbf{H} + \mu \mathbf{I})^{-1} \; \boldsymbol{g},
$$

$$
\mathfrak{a} = \left[\mathrm{vec}\left(\mathbf{A}^{(1)}\right)^T \mathrm{vec}\left(\mathbf{A}^{(2)}\right)^T \cdots \mathrm{vec}\left(\mathbf{A}^{(N)}\right)^T\right]^T
$$

$\boldsymbol{g}$ gradient $R \sum I_n \times 1$

$\mathbf{H}$ approximate Hessian $R \sum I_n \times R \sum I_n$

$\mu > 0$: damping parameter,

Damped Gauss-Newton Algorithm ?

$$
\begin{aligned}
D &= \|\mathcal{Y} - \mathcal{J} \times_1 \mathbf{A}^{(1)} \cdots \times_N \mathbf{A}^{(N)}\|_F^2 + P_l(\{\mathbf{A}^{(n)}\}) + P_s(\{\mathbf{A}^{(n)}\}), \\
P_l &= -\sum_{n=1}^{N} \alpha_n \sum_{i_n=1}^{I_n} \sum_{r=1}^{R} \log(a_{i_n r}^{(n)}), \quad P_s = \sum_{n=1}^{N} \beta_n \left\|\mathbf{A}^{(n)}\right\|_1,
\end{aligned}
$$

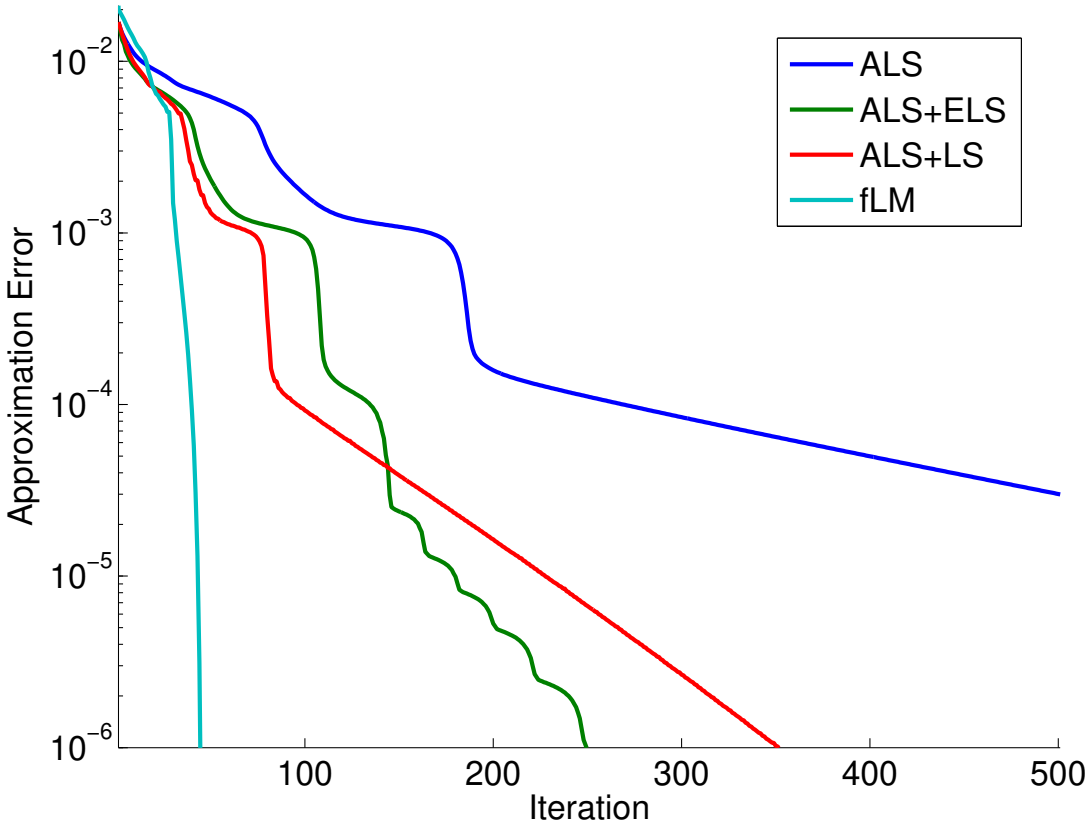
$\alpha_n \geq 0, \beta_n \geq 0 \; \forall n.$

Damped Gauss-Newton update rule

$$
\mathbf{a} \leftarrow \mathbf{a} - \left(\mathbf{H} + \mu \mathbf{I}\right)^{-1} \boldsymbol{g},
$$

$\mathbf{a}$	$= \left[\text{vec}\left(\mathbf{A}^{(1)}\right)^T \text{vec}\left(\mathbf{A}^{(2)}\right)^T \cdots \text{vec}\left(\mathbf{A}^{(N)}\right)^T\right]^T$
$\boldsymbol{g}$	gradient $R \sum I_n \times 1$
$\mathbf{H}$	approximate Hessian $R \sum I_n \times R \sum I_n$
$\mu > 0$:	damping parameter,

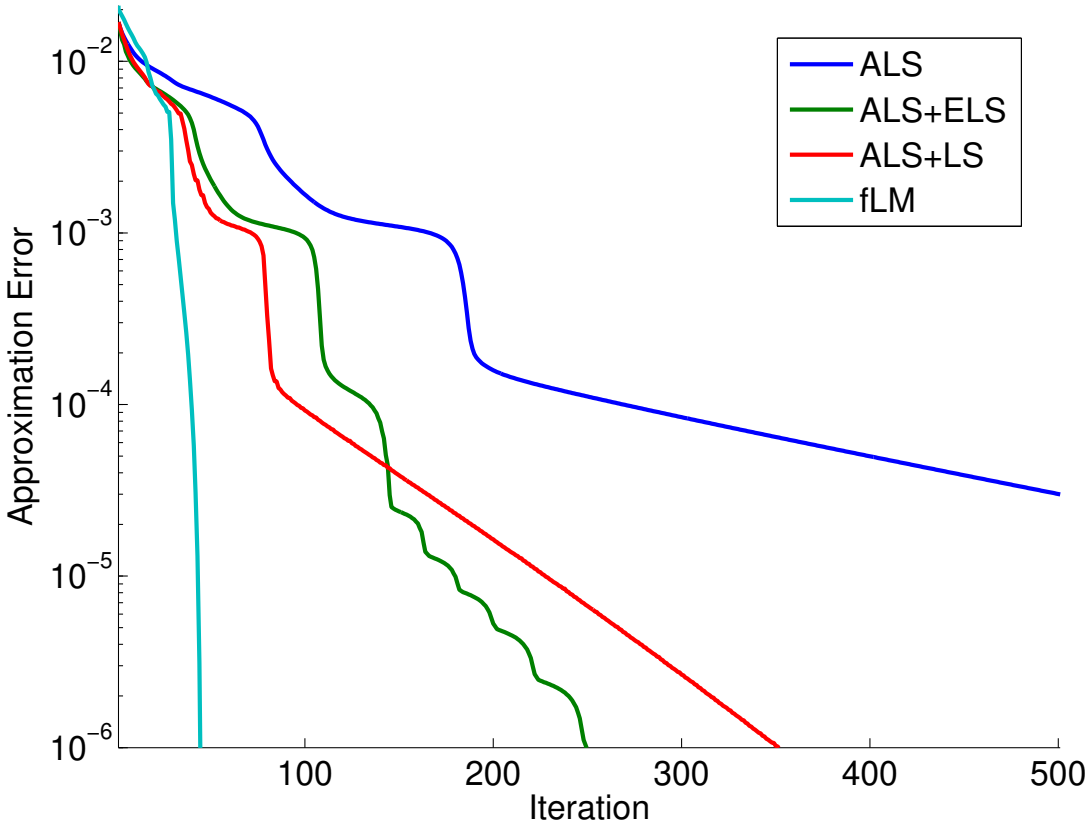
dampled Gauss-Newton vs ALS



Tensor of size $30 \times 30 \times 30$ and rank-$R = 10$ composed by highly correlated components which have $\boldsymbol{a}_r^{(n)T} \boldsymbol{a}_s^{(n)} \in [0.8, 0.999]$, $r \neq s$.

- LM works well even for difficult cases such as tensors with highly collinear components, or different magnitudes of factors.
- But high computational cost due to computing Jacobian, gradient, Hessian and inverse of Hessian.

dampled Gauss-Newton vs ALS



Tensor of size $30 \times 30 \times 30$ and rank-$R = 10$ composed by highly correlated components which have $\boldsymbol{a}_r^{(n)T} \boldsymbol{a}_s^{(n)} \in [0.8, 0.999]$, $r \neq s$.

- LM works well even for difficult cases such as tensors with highly collinear components, or different magnitudes of factors.
- But high computational cost due to computing Jacobian, gradient, Hessian and inverse of Hessian.

Jacobian, Hessian and gradient matrices

$$\mathbf{J} = \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{\mathfrak{a}}} = \left[\frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(1)}} \quad \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(2)}} \quad \cdots \quad \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(N)}} \right] \in \mathbb{R}^{(\prod I_n) \times R \sum I_n}.$$

$$\boldsymbol{g} = \mathbf{J}^T(\hat{\boldsymbol{y}} - \boldsymbol{y}) + \frac{\partial P_l}{\partial \boldsymbol{\mathfrak{a}}} + \frac{\partial P_s}{\partial \boldsymbol{\mathfrak{a}}}$$

$$= \mathbf{J}^T(\hat{\boldsymbol{y}} - \boldsymbol{y}) - \left[\left(\alpha_n \operatorname{vec}\left(\mathbf{A}^{(n)}\right)^{\bullet[-1]} - \beta_n \right)^T \right]_{n=1}^{N} \in \mathbb{R}^{R \sum I_n}.$$

$$\mathbf{H} = \mathbf{J}^T \mathbf{J} + \frac{\partial^2 P_l}{\partial \boldsymbol{\mathfrak{a}}^2} + \frac{\partial^2 P_s}{\partial \boldsymbol{\mathfrak{a}}^2}$$

$$= \mathbf{J}^T \mathbf{J} + \bigoplus_{n=1}^{N} \operatorname{diag}\left\{ \alpha_n \operatorname{vec}\left(\mathbf{A}^{(n)}\right)^{\bullet[-2]} \right\} \in \mathbb{R}^{R \sum I_n \times R \sum I_n}.$$

Update rule is computationally demanding due to construction of Jacobian, and inverse of large Hessians.

Jacobian, Hessian and gradient matrices

$$\mathbf{J} = \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{\mathfrak{a}}} = \left[\frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(1)}} \quad \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(2)}} \quad \cdots \quad \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(N)}} \right] \in \mathbb{R}^{(\prod I_n) \times R \sum I_n}.$$

$$\boldsymbol{g} = \mathbf{J}^T(\hat{\boldsymbol{y}} - \boldsymbol{y}) + \frac{\partial P_l}{\partial \boldsymbol{\mathfrak{a}}} + \frac{\partial P_s}{\partial \boldsymbol{\mathfrak{a}}}$$

$$= \mathbf{J}^T(\hat{\boldsymbol{y}} - \boldsymbol{y}) - \left[\left(\alpha_n \operatorname{vec}\left(\mathbf{A}^{(n)}\right)^{\bullet[-1]} - \beta_n \right)^T \right]_{n=1}^{N \quad T} \in \mathbb{R}^{R \sum I_n}.$$

$$\mathbf{H} = \mathbf{J}^T \mathbf{J} + \frac{\partial^2 P_l}{\partial \boldsymbol{\mathfrak{a}}^2} + \frac{\partial^2 P_s}{\partial \boldsymbol{\mathfrak{a}}^2}$$

$$= \mathbf{J}^T \mathbf{J} + \bigoplus_{n=1}^{N} \operatorname{diag}\left\{ \alpha_n \operatorname{vec}\left(\mathbf{A}^{(n)}\right)^{\bullet[-2]} \right\} \in \mathbb{R}^{R \sum I_n \times R \sum I_n}.$$

Update rule is computationally demanding due to construction of Jacobian, and inverse of large Hessians.

Jacobian, Hessian and gradient matrices

$$
\begin{aligned}
\mathbf{J} &= \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{\mathfrak{a}}} = \left[\begin{array}{cccc} \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(1)}} & \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(2)}} & \cdots & \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(N)}} \end{array} \right] \in \mathbb{R}^{(\prod I_n) \times R \sum I_n}. \\
\boldsymbol{g} &= \mathbf{J}^T(\hat{\boldsymbol{y}} - \boldsymbol{y}) + \frac{\partial P_l}{\partial \boldsymbol{\mathfrak{a}}} + \frac{\partial P_s}{\partial \boldsymbol{\mathfrak{a}}} \\
&= \mathbf{J}^T(\hat{\boldsymbol{y}} - \boldsymbol{y}) - \left[\left(\alpha_n \operatorname{vec}\left(\mathbf{A}^{(n)}\right)^{\bullet[-1]} - \beta_n \right)^T \right]_{n=1}^{N \quad T} \in \mathbb{R}^{R \sum I_n}. \\
\mathbf{H} &= \mathbf{J}^T \mathbf{J} + \frac{\partial^2 P_l}{\partial \boldsymbol{\mathfrak{a}}^2} + \frac{\partial^2 P_s}{\partial \boldsymbol{\mathfrak{a}}^2} \\
&= \mathbf{J}^T \mathbf{J} + \bigoplus_{n=1}^{N} \operatorname{diag}\left\{ \alpha_n \operatorname{vec}\left(\mathbf{A}^{(n)}\right)^{\bullet[-2]} \right\} \in \mathbb{R}^{R \sum I_n \times R \sum I_n}.
\end{aligned}
$$

Update rule is computationally demanding due to construction of Jacobian, and inverse of large Hessians.

Jacobian, Hessian and gradient matrices

$$
\begin{aligned}
\mathbf{J} &= \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{\mathfrak{a}}} = \left[\begin{array}{cccc} \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(1)}} & \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(2)}} & \cdots & \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(N)}} \end{array} \right] \in \mathbb{R}^{(\prod I_n) \times R \sum I_n}. \\
\boldsymbol{g} &= \mathbf{J}^T(\hat{\boldsymbol{y}} - \boldsymbol{y}) + \frac{\partial P_l}{\partial \boldsymbol{\mathfrak{a}}} + \frac{\partial P_s}{\partial \boldsymbol{\mathfrak{a}}} \\
&= \mathbf{J}^T(\hat{\boldsymbol{y}} - \boldsymbol{y}) - \left[\left(\alpha_n \operatorname{vec}\left(\mathbf{A}^{(n)}\right)^{\bullet[-1]} - \beta_n \right)^T \right]_{n=1}^{N \quad T} \in \mathbb{R}^{R \sum I_n}. \\
\mathbf{H} &= \mathbf{J}^T \mathbf{J} + \frac{\partial^2 P_l}{\partial \boldsymbol{\mathfrak{a}}^2} + \frac{\partial^2 P_s}{\partial \boldsymbol{\mathfrak{a}}^2} \\
&= \mathbf{J}^T \mathbf{J} + \bigoplus_{n=1}^{N} \operatorname{diag}\left\{ \alpha_n \operatorname{vec}\left(\mathbf{A}^{(n)}\right)^{\bullet[-2]} \right\} \in \mathbb{R}^{R \sum I_n \times R \sum I_n}.
\end{aligned}
$$

Update rule is computationally demanding due to construction of Jacobian, and inverse of large Hessians.

Jacobian matrix

$$\mathbf{J} = \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathfrak{a}} = \left[\frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(1)}} \quad \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(2)}} \quad \cdots \quad \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(N)}} \right].$$

Jacobian matrix

$$\mathbf{J} = \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{\mathfrak{a}}} = \left[\begin{array}{cccc} \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(1)}} & \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(2)}} & \cdots & \frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(N)}} \end{array} \right].$$

$$\begin{aligned} \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_r^{(n)}} &= \frac{\partial \operatorname{vec}\left(\hat{\mathcal{Y}}\right)}{\partial \boldsymbol{a}_r^{(n)}} = \mathbf{P}_n \frac{\partial \operatorname{vec}\left(\hat{\mathbf{Y}}_{(n)}\right)}{\partial \boldsymbol{a}_r^{(n)}} = \mathbf{P}_n \frac{\partial \sum_{r=1}^{R} \boldsymbol{a}_r^{(n)} \left(\boldsymbol{a}_r^{\otimes -n}\right)^T}{\partial \boldsymbol{a}_r^{(n)}} \\ &= \mathbf{P}_n \left(\boldsymbol{a}_r^{\otimes -n} \otimes \mathbf{I}_{I_n}\right). \end{aligned}$$

$\mathbf{P}_n$ is a permutation matrix: $\operatorname{vec}(\mathcal{Y}) = \mathbf{P}_n \operatorname{vec}\left(\mathbf{Y}_{(n)}\right)$

Fast computation of Gradient

Jacobian

$$\mathbf{J} = \left[\mathbf{P}_1 \left(\mathbf{A}^{\odot -1} \otimes \mathbf{I}_{I_1}\right) \quad \cdots \quad \mathbf{P}_n \left(\mathbf{A}^{\odot -n} \otimes \mathbf{I}_{I_n}\right) \quad \cdots \quad \mathbf{P}_N \left(\mathbf{A}^{\odot -N} \otimes \mathbf{I}_{I_N}\right)\right]$$

Fast computation of Gradient

Jacobian

$$\mathbf{J} = \left[\mathbf{P}_1 \left(\mathbf{A}^{\odot -1} \otimes \mathbf{I}_{I_1}\right) \quad \cdots \quad \mathbf{P}_n \left(\mathbf{A}^{\odot -n} \otimes \mathbf{I}_{I_n}\right) \quad \cdots \quad \mathbf{P}_N \left(\mathbf{A}^{\odot -N} \otimes \mathbf{I}_{I_N}\right)\right]$$

Gradient matrix

$$\boldsymbol{g} = \mathbf{J}^T(\hat{\boldsymbol{y}} - \boldsymbol{y}) + \frac{\partial P_l}{\partial \mathfrak{a}} + \frac{\partial P_s}{\partial \mathfrak{a}}$$

Fast computation of Gradient

Jacobian

$$\mathbf{J} = \left[\mathbf{P}_1 \left(\mathbf{A}^{\odot -1} \otimes \mathbf{I}_{I_1}\right) \quad \cdots \quad \mathbf{P}_n \left(\mathbf{A}^{\odot -n} \otimes \mathbf{I}_{I_n}\right) \quad \cdots \quad \mathbf{P}_N \left(\mathbf{A}^{\odot -N} \otimes \mathbf{I}_{I_N}\right)\right]$$

Gradient matrix

$$\boldsymbol{g} = \mathbf{J}^T(\hat{\boldsymbol{y}} - \boldsymbol{y}) + \frac{\partial P_l}{\partial \mathfrak{a}} + \frac{\partial P_s}{\partial \mathfrak{a}}$$

$$\begin{aligned}
\left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \mathbf{A}^{(n)}}\right)^T (\boldsymbol{y} - \hat{\boldsymbol{y}}) &= \left(\mathbf{A}^{\odot -n\,T} \otimes \mathbf{I}_n\right) \mathbf{P}_n^T \operatorname{vec}\left(\mathcal{Y} - \hat{\mathcal{Y}}\right) \\
&= \left(\mathbf{A}^{\odot -n\,T} \otimes \mathbf{I}_n\right) \operatorname{vec}\left(\mathbf{Y}_{(n)} - \hat{\mathbf{Y}}_{(n)}\right) \\
&= \operatorname{vec}\left(\mathbf{Y}_{(n)}\mathbf{A}^{\odot -n} - \mathbf{A}^{(n)}\Gamma^{(n,n)}\right).
\end{aligned}$$

$$\Gamma^{(n,n)} = (\mathbf{A}^{(N)T}\mathbf{A}^{(N)}) \circledast \cdots \circledast (\mathbf{A}^{(n+1)T}\mathbf{A}^{(n+1)}) \circledast (\mathbf{A}^{(n-1)T}\mathbf{A}^{(n-1)}) \circledast \cdots \circledast (\mathbf{A}^{(1)T}\mathbf{A}^{(1)})$$

Fast computation of Gradient

Gradient

$$\boldsymbol{g} = \begin{bmatrix} \operatorname{vec}\left(-\mathbf{Y}_{(1)}\mathbf{A}^{\odot -1} + \mathbf{A}^{(1)}\Gamma^{(1,1)} - \alpha_1 \left(\mathbf{A}^{(1)}\right)^{\bullet[-1]} + \beta_1\right) \\ \vdots \\ \operatorname{vec}\left(-\mathbf{Y}_{(n)}\mathbf{A}^{\odot -n} + \mathbf{A}^{(n)}\Gamma^{(n,n)} - \alpha_n \left(\mathbf{A}^{(n)}\right)^{\bullet[-1]} + \beta_n\right) \\ \vdots \\ \operatorname{vec}\left(-\mathbf{Y}_{(N)}\mathbf{A}^{\odot -N} + \mathbf{A}^{(N)}\Gamma^{(N,N)} - \alpha_N \left(\mathbf{A}^{(N)}\right)^{\bullet[-1]} + \beta_N\right) \end{bmatrix}.$$

$$\Gamma^{(n,n)} = \circledast_{k \neq n} \mathbf{A}^{(k)T}\mathbf{A}^{(k)} \in \mathbb{R}^{R \times R}$$

Fast computation of Hessian

- Inverse of $\mathbf{H}$ is too computationally expensive and of order $\mathcal{O}(R^3(I_1 + I_2 + \cdots + I_N)^3)$.

$$\mathbf{H} + \mu\mathbf{I} = \mathbf{J}^T\mathbf{J} + \bigoplus_{n=1}^{N} \operatorname{diag}\left\{\alpha_n \operatorname{vec}\left(\mathbf{A}^{(n)}\right)^{\bullet[-2]} + \mu\right\} \in \mathbb{R}^{R\sum I_n \times R\sum I_n}$$

- But approximate Hessian is rank-defficient. Expression of $\mathbf{H}$ by low-rank adjustment can reduce computational cost.

$$\mathbf{H} + \mu\mathbf{I} = \begin{bmatrix} \mathbf{H}^{(1,1)} & \cdots & \mathbf{H}^{(1,m)} & \cdots & \mathbf{H}^{(1,N)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{H}^{(n,1)} & \cdots & \mathbf{H}^{(n,m)} & \cdots & \mathbf{H}^{(n,N)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{H}^{(N,1)} & \cdots & \mathbf{H}^{(N,m)} & \cdots & \mathbf{H}^{(N,N)} \end{bmatrix}.$$

Fast computation of Hessian

- Inverse of $\mathbf{H}$ is too computationally expensive and of order $\mathcal{O}(R^3(I_1 + I_2 + \cdots + I_N)^3)$.

$$\mathbf{H} + \mu\mathbf{I} = \mathbf{J}^T\mathbf{J} + \bigoplus_{n=1}^{N} \operatorname{diag}\left\{\alpha_n \operatorname{vec}\left(\mathbf{A}^{(n)}\right)^{\bullet[-2]} + \mu\right\} \in \mathbb{R}^{R\sum I_n \times R\sum I_n}$$

- But approximate Hessian is rank-defficient. Expression of $\mathbf{H}$ by low-rank adjustment can reduce computational cost.

$$\mathbf{H} + \mu\mathbf{I} = \begin{bmatrix} \mathbf{H}^{(1,1)} & \cdots & \mathbf{H}^{(1,m)} & \cdots & \mathbf{H}^{(1,N)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{H}^{(n,1)} & \cdots & \mathbf{H}^{(n,m)} & \cdots & \mathbf{H}^{(n,N)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{H}^{(N,1)} & \cdots & \mathbf{H}^{(N,m)} & \cdots & \mathbf{H}^{(N,N)} \end{bmatrix}.$$

Expression of block matrix $\mathbf{H}^{(n,m)}$ I

$$\begin{aligned}\mathbf{H} &= \mathbf{J}^T\mathbf{J} \\ &= \begin{bmatrix} \mathbf{H}_{1,1}^{(1,1)} & \cdots & \mathbf{H}_{1,R}^{(1,1)} & \cdots & \mathbf{H}_{r,s}^{(1,m)} & \cdots & \mathbf{H}_{R,R}^{(1,N)} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{H}_{1,1}^{(n,1)} & \cdots & \mathbf{H}_{1,R}^{(n,1)} & \cdots & \mathbf{H}_{r,s}^{(n,m)} & \cdots & \mathbf{H}_{R,R}^{(n,N)} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{H}_{1,1}^{(N,1)} & \cdots & \mathbf{H}_{1,R}^{(N,1)} & \cdots & \mathbf{H}_{r,s}^{(N,m)} & \cdots & \mathbf{H}_{R,R}^{(N,N)} \end{bmatrix}, \quad (17)\end{aligned}$$

where block matrices $\mathbf{H}_{r,s}^{(n,m)} \in \mathbb{R}^{I_n \times I_m}$ are computed as

$$\mathbf{H}_{r,s}^{(n,m)} = \left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_r^{(n)}}\right)^T \left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_s^{(m)}}\right), \quad (18)$$

$n, m = 1, 2, \ldots, N$, $r, s = 1, 2, \ldots, R$.

Expression of block matrix $\mathbf{H}^{(n,m)}$ II

Theorem (Permutation in a Kronecker product)

Multiplication of a permutation matrix $\mathbf{P}_n$ with a Kronecker product of N vectors $\boldsymbol{a}^{(n)} \in \mathbb{R}^{I_n}, n = 1, 2, \ldots, N$ will move the n-th item $\boldsymbol{a}^{(n)}$ to the last term, that is

$$\begin{aligned}&\left(\boldsymbol{a}^{(N)} \otimes \cdots \otimes \boldsymbol{a}^{(2)} \otimes \boldsymbol{a}^{(1)}\right)\\ &= \mathbf{P}_n \left(\boldsymbol{a}^{(N)} \otimes \cdots \otimes \boldsymbol{a}^{(n+1)} \otimes \boldsymbol{a}^{(n-1)} \otimes \cdots \otimes \boldsymbol{a}^{(1)} \otimes \boldsymbol{a}^{(n)}\right). \quad (19)\end{aligned}$$

Theorem (Rank-one block matrices $\mathbf{H}_{r,s}^{(n,m)}$)

Block matrices $\mathbf{H}_{r,s}^{(n,m)}$, $n \neq m$ are rank-one matrices

$$\mathbf{H}_{r,s}^{(n,m)} = \left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_r^{(n)}}\right)^T \left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_s^{(m)}}\right) = \gamma_{r\,s}^{(n,m)}\, \boldsymbol{a}_r^{(n)}\, \boldsymbol{a}_s^{(m)\,T} \quad \in \mathbb{R}^{I_n \times I_m}. \quad (20)$$

Expression of block matrix $\mathbf{H}^{(n,m)}$ III

$$\begin{aligned}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_r^{(n)}} &= \mathbf{P}_n \left[\begin{array}{cccc} \boldsymbol{a}_r^{\otimes -n} \otimes \boldsymbol{e}_1^{(n)} & \boldsymbol{a}_r^{\otimes -n} \otimes \boldsymbol{e}_2^{(n)} & \cdots & \boldsymbol{a}_r^{\otimes -n} \otimes \boldsymbol{e}_{I_n}^{(n)} \end{array} \right] \\ &= \left[\mathbf{P}_n \left(\boldsymbol{a}_r^{\otimes -n} \otimes \boldsymbol{e}_{i_n}^{(n)} \right) \right]_{i_n=1}^{I_n} \quad \in \mathbb{R}^{(I_1 I_2 \ldots I_N) \times I_n}, \end{aligned} \tag{21}$$

where $\boldsymbol{e}_{i_n}^{(n)}$ is the i_n-th column of the identity matrix $\mathbf{I}_n \in \mathbb{R}^{I_n \times I_n}$.

$$\begin{aligned}\left[\mathbf{H}_{r,s}^{(n,m)} \right]_{i_n, i_m} &= \left[\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_r^{(n)}} \right]_{i_n}^T \left[\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_s^{(m)}} \right]_{i_m} \\ &= \left(\boldsymbol{a}_r^{\otimes -n\, T} \otimes \boldsymbol{e}_{i_n}^{(n)\, T} \right) \mathbf{P}_n^T \mathbf{P}_m \left(\boldsymbol{a}_s^{\otimes -m} \otimes \boldsymbol{e}_{i_m}^{(m)} \right) \end{aligned}$$

Expression of block matrix $\mathbf{H}^{(n,m)}$ IV

$$\begin{aligned}
\left[\mathbf{H}_{r,s}^{(n,m)}\right]_{i_n,i_m} &= \left(\boldsymbol{a}_r^{(N)\,T} \otimes \cdots \otimes \boldsymbol{a}_r^{(n+1)\,T} \otimes \boldsymbol{e}_{i_n}^{(n)\,T} \otimes \boldsymbol{a}_r^{(n-1)\,T} \otimes \cdots \otimes \boldsymbol{a}_r^{(1)\,T}\right) \\
&\quad \left(\boldsymbol{a}_s^{(N)} \otimes \cdots \otimes \boldsymbol{a}_s^{(m+1)} \otimes \boldsymbol{e}_{i_m}^{(m)} \otimes \boldsymbol{a}_s^{(m-1)} \otimes \cdots \otimes \boldsymbol{a}_s^{(1)}\right) \\
&= \left(\prod_{k\neq n,m} \left(\boldsymbol{a}_r^{(k)\,T}\, \boldsymbol{a}_s^{(k)}\right)\right)\left(\left(\boldsymbol{a}_r^{(m)\,T}\, \boldsymbol{e}_{i_m}^{(m)}\right) \otimes \left(\boldsymbol{e}_{i_n}^{(n)\,T}\, \boldsymbol{a}_s^{(n)}\right)\right) \\
&= \gamma_{r\,s}^{(n,m)}\, a_{i_m\,r}^{(m)}\, a_{i_n\,s}^{(n)}\,, \qquad (22)
\end{aligned}$$

where $\gamma_{rs}^{(n,m)}$ is the (r,s) element of $\Gamma^{(n,m)} = \left(\mathbf{A}^T\,\mathbf{A}\right)^{\circledast -(n,m)}$. This leads to a compact

$$\mathbf{H}_{r,s}^{(n,m)} = \left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_r^{(n)}}\right)^T \left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_s^{(m)}}\right) = \gamma_{r\,s}^{(n,m)}\, \boldsymbol{a}_r^{(n)}\, \boldsymbol{a}_s^{(m)\,T} \quad \in \mathbb{R}^{I_n \times I_m}\,. \qquad (23)$$

Expression of block matrix $\mathbf{H}^{(n,m)}$ V

Theorem (Diagonal block matrices $\mathbf{H}_{r,s}^{(n,n)}$)

Block matrices $\mathbf{H}_{r,s}^{(n,n)}$, $n \neq m$ *are diagonal matrices.*

$$\begin{aligned}
\mathbf{H}_{r,s}^{(n,n)} &= \left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_r^{(n)}}\right)^T \left(\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_s^{(n)}}\right) = \left(\boldsymbol{a}_r^{\otimes_{-n}\,T} \otimes \mathbf{I}_n\right) \mathbf{P}_n^T \mathbf{P}_n \left(\boldsymbol{a}_s^{\otimes_{-n}\,T} \otimes \mathbf{I}_n\right) \\
&= \left(\prod_{k \neq n} \boldsymbol{a}_r^{(k)\,T}\, \boldsymbol{a}_s^{(k)}\right) \mathbf{I}_n = \gamma_{rs}^{(n,n)}\, \mathbf{I}_n\,, \qquad (24)
\end{aligned}$$

where $\gamma_{rs}^{(n,n)}$ are the (r,s) element of $\Gamma^{(n,n)} = \left(\mathbf{A}^T\,\mathbf{A}\right)^{\circledast_{-(n)}}$.

Expression of block matrix $\mathbf{H}^{(n,m)}$ VI

Theorem (Diagonal block $\mathbf{H}^{(n,n)}$)

A block matrix $\mathbf{H}^{(n,n)}$ *for* $n = 1, 2, \ldots, N$ *on the diagonal of the Hessian* $\mathbf{H}$

$$\mathbf{H}^{(n,n)} = \Gamma^{(n,n)} \otimes \mathbf{I}_n \,.$$

Proof.

Expression of block matrix $\mathbf{H}^{(n,m)}$ VII

$$\begin{aligned}
\mathbf{H}^{(n,n)} &= \begin{bmatrix}
\gamma_{1,1}^{(n,n)}\mathbf{I}_n & \cdots & \gamma_{r,1}^{(n,n)}\mathbf{I}_n & \cdots & \gamma_{R,1}^{(n,n)}\mathbf{I}_n \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\gamma_{1,s}^{(n,n)}\mathbf{I}_n & \cdots & \gamma_{r,s}^{(n,n)}\mathbf{I}_n & \cdots & \gamma_{r,s}^{(n,n)}\mathbf{I}_n \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\gamma_{1,R}^{(n,n)}\mathbf{I}_n & \cdots & \gamma_{r,R}^{(n,n)}\mathbf{I}_n & \cdots & \gamma_{R,R}^{(n,n)}\mathbf{I}_n
\end{bmatrix} \\
&= \Gamma^{(n,n)} \otimes \mathbf{I}_n \,. \qquad (25)
\end{aligned}$$

□

Expression of block matrix $\mathbf{H}^{(n,m)}$ VIII

Theorem (Block matrix $\mathbf{H}^{(n,m)}$)

$\mathbf{H}^{(n,m)}$ *for* $\forall n \neq m$ *can be decomposed as*

$$\mathbf{H}^{(n,m)} = \left(\mathbf{I}_R \otimes \mathbf{A}^{(n)}\right) \mathbf{P}_{R,R} \operatorname{diag}\left(\operatorname{vec}\left(\Gamma^{(n,m)}\right)\right) \left(\mathbf{I}_R \otimes \mathbf{A}^{(m)\,T}\right).$$

Expression of block matrix $\mathbf{H}^{(n,m)}$ IX

Definition

Vectorization of an $I \times J$ matrix $\mathbf{X}$ can be established via its transpose matrix $\mathbf{X}^T$ by an $IJ \times IJ$ permutation matrix $\mathbf{P}_{I,J}$

$$\text{vec}(\mathbf{X}) = \mathbf{P}_{I,J} \, \text{vec}\left(\mathbf{X}^T\right) . \tag{26}$$

Theorem (Interchange in Kronnecker product)

Consider the permutation matrices $\mathbf{P}_{I,P}$ and $\mathbf{P}_{J,Q}$. Then

$$\mathbf{B} \otimes \mathbf{A} = \mathbf{P}_{I,P}^T \, (\mathbf{A} \otimes \mathbf{B}) \, \mathbf{P}_{J,Q} , \tag{27}$$

for all matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{P \times Q}$.

Proof.

This theorem is replicated from the problem 4.3.10 in Horn and Johnson (1991). See proof in Horn and Johnson (1991). □

Expression of block matrix $\mathbf{H}^{(n,m)}$ X

Theorem (Concatenation of Kronnecker products)

Concatenation of the Kronecker products forms a Kronecker product of the concatenation matrix

$$[\mathbf{A} \otimes \mathbf{C} \quad \mathbf{D} \otimes \mathbf{C}] = [\mathbf{A} \quad \mathbf{D}] \otimes \mathbf{C}\,, \tag{28}$$

and

$$[\mathbf{A} \otimes \mathbf{B} \quad \mathbf{A} \otimes \mathbf{C}] = (\mathbf{A} \otimes [\mathbf{B} \quad \mathbf{C}])\ \mathbf{P}\,, \tag{29}$$

where $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ *and* $\mathbf{D}$ *are matrices of size* $I \times J$, $P \times Q$, $P \times R$ *and* $I \times K$, *respectively, and* $\mathbf{P}$ *denotes a permutation matrix.*

Expression of block matrix $\mathbf{H}^{(n,m)}$ XI

$$
\begin{aligned}
[\mathbf{A}\otimes\mathbf{C} \quad \mathbf{D}\otimes\mathbf{C}] &= \left[\begin{array}{cccccc} \boldsymbol{a}_1\otimes\mathbf{C} & \cdots & \boldsymbol{a}_M\otimes\mathbf{C} & \boldsymbol{d}_1\otimes\mathbf{C} & \cdots & \boldsymbol{d}_K\otimes\mathbf{C} \end{array}\right] \\
&= \left[\begin{array}{cccccc} \boldsymbol{a}_1 & \cdots & \boldsymbol{a}_M & \boldsymbol{d}_1 & \cdots & \boldsymbol{d}_K \end{array}\right]\otimes\mathbf{C} \\
&= \left[\begin{array}{cc} \mathbf{A} & \mathbf{D} \end{array}\right]\otimes\mathbf{C}. \qquad (30)
\end{aligned}
$$

By employing Theorem 12, the second concatenation (29) is proved as follows

$$
\begin{aligned}
[\mathbf{A}\otimes\mathbf{B} \quad \mathbf{A}\otimes\mathbf{C}] &= \left[\mathbf{P}_{P,I}^{T}\left(\mathbf{B}\otimes\mathbf{A}\right)\mathbf{P}_{Q,J} \quad \mathbf{P}_{P,I}^{T}\left(\mathbf{C}\otimes\mathbf{A}\right)\mathbf{P}_{R,J}\right] \\
&= \mathbf{P}_{P,I}^{T}\left([\mathbf{B} \quad \mathbf{C}]\otimes\mathbf{A}\right)\left[\begin{array}{cc} \mathbf{P}_{Q,J} & \\ & \mathbf{P}_{R,J} \end{array}\right] \\
&= \mathbf{P}_{P,I}^{T}\mathbf{P}_{P,I}\left(\mathbf{A}\otimes[\mathbf{B} \quad \mathbf{C}]\right)\mathbf{P}_{Q+R,J}^{T}\left[\begin{array}{cc} \mathbf{P}_{Q,J} & \\ & \mathbf{P}_{R,J} \end{array}\right] \\
&= \left(\mathbf{A}\otimes[\mathbf{B} \quad \mathbf{C}]\right)\mathbf{P}. \qquad (31)
\end{aligned}
$$

Expression of block matrix $\mathbf{H}^{(n,m)}$ XII

$$\left[\begin{array}{ccccc} \mathbf{A}\otimes\boldsymbol{b}_1 & \cdots & \mathbf{A}\otimes\boldsymbol{b}_q & \cdots & \mathbf{A}\otimes\boldsymbol{b}_Q \end{array}\right] = (\mathbf{A}\otimes\mathbf{B})\,\mathbf{P}_{J,Q}\,.$$

$$\begin{aligned}
\mathbf{H}^{(n,m)} &= \left[\begin{array}{ccccccc}
\gamma_{1,1}^{(n,m)}\,\boldsymbol{a}_1^{(n)}\,\boldsymbol{a}_1^{(m)\,T} & \cdots & \gamma_{r,1}^{(n,m)}\,\boldsymbol{a}_r^{(n)}\,\boldsymbol{a}_1^{(m)\,T} & \cdots & \gamma_{R,s}^{(n,m)}\,\boldsymbol{a}_R^{(n)}\,\boldsymbol{a}_1^{(m)\,T} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\gamma_{1,s}^{(n,m)}\,\boldsymbol{a}_1^{(n)}\,\boldsymbol{a}_s^{(m)\,T} & \cdots & \gamma_{r,s}^{(n,m)}\,\boldsymbol{a}_r^{(n)}\,\boldsymbol{a}_s^{(m)\,T} & \cdots & \gamma_{R,s}^{(n,m)}\,\boldsymbol{a}_R^{(n)}\,\boldsymbol{a}_s^{(m)\,T} \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
\gamma_{1,R}^{(n,m)}\,\boldsymbol{a}_1^{(n)}\,\boldsymbol{a}_R^{(m)\,T} & \cdots & \gamma_{r,R}^{(n,m)}\,\boldsymbol{a}_r^{(n)}\,\boldsymbol{a}_R^{(m)\,T} & \cdots & \gamma_{R,R}^{(n,m)}\,\boldsymbol{a}_R^{(n)}\,\boldsymbol{a}_R^{(m)\,T}
\end{array}\right] \\
&= \left[\begin{array}{ccccc} \left(\mathbf{I}_R\otimes\boldsymbol{a}_1^{(n)}\right)\mathbf{D}_1^{(n,m)}\,\mathbf{A}^{(n)\,T} & \cdots & \left(\mathbf{I}_R\otimes\boldsymbol{a}_r^{(n)}\right)\mathbf{D}_r^{(n,m)}\,\mathbf{A}^{(n)\,T} & \cdots & \left(\mathbf{I}_R\otimes\boldsymbol{a}_R^{(n)}\right)\mathbf{D}_R^{(n,m)}\,\mathbf{A}^{(n} \end{array}\right] \\
&= \left[\begin{array}{ccccc} \left(\mathbf{I}_R\otimes\boldsymbol{a}_1^{(n)}\right) & \cdots & \left(\mathbf{I}_R\otimes\boldsymbol{a}_r^{(n)}\right) & \cdots & \left(\mathbf{I}_R\otimes\boldsymbol{a}_R^{(n)}\right) \end{array}\right]\mathbf{D}^{(n,m)}\left(\mathbf{I}_R\otimes\mathbf{A}^{(m)\,T}\right) \\
&= \left(\mathbf{I}_R\otimes\mathbf{A}^{(n)}\right)\mathbf{P}_{R,R}\,\mathbf{D}^{(n,m)}\left(\mathbf{I}_R\otimes\mathbf{A}^{(m)\,T}\right),
\end{aligned}$$

where $\mathbf{D}_r^{(n,m)} = \operatorname{diag}\left(\boldsymbol{\gamma}_r^{(n,m)}\right)$ is the diagonal matrix whose diagonal entries are the r-th row of the matrix $\Gamma^{(n,m)}$, $\mathbf{D}^{(n,m)} = \operatorname{diag}\left(\operatorname{vec}\left(\Gamma^{(n,m)}\right)\right)$ is the diagonal matrix whose the main diagonal is $\operatorname{vec}\left(\Gamma^{(n,m)}\right)$, $\mathbf{P}_{R,R}$ is the permutation matrix of an $R\times R$ matrix.

Expression of block matrix $\mathbf{H}^{(n,m)}$ XIII

Theorem (Expression of block matrix $\mathbf{H}^{(n,m)}$)

A block matrix $\mathbf{H}^{(n,m)}$ *can be expressed by*

$$\mathbf{H}^{(n,m)} = \begin{cases} \Gamma^{(n,n)} \otimes \mathbf{I}_{I_n} + \operatorname{diag}\left\{\alpha_n \operatorname{vec}\left(\mathbf{A}^{(n)}\right)^{\bullet[-2]} + \mu\right\}, & n = m \\ (\mathbf{I}_R \otimes \mathbf{A}^{(n)})\, \Pi_{R,R} \operatorname{diag}(\operatorname{vec}(\Gamma^{(n,m)}))(\mathbf{I}_R \otimes \mathbf{A}^{(m)T}), & n \neq m \end{cases}$$

where $\Gamma^{(n,m)} = \circledast_{k \neq m,n} \mathbf{A}^{(k)T}\, \mathbf{A}^{(k)}$,

$\Pi_{R,R}$ *is the commutation matrix:* $\operatorname{vec}(\mathbf{X}_{R\times R}) = \Pi_{R,R} \operatorname{vec}\left(\mathbf{X}^T\right)$.

Theorem (Low rank Adjustment for the Hessian **H**)

The Hessian **H** *can be decomposed into low rank matrices under the form as*

$$\mathbf{H} = \mathbf{G} + \mathbf{Z}\mathbf{K}\mathbf{Z}^T. \tag{33}$$

where **G** *is an invertible matrix, matrix* **Z** *consists of* $N R^2$ *columns, and kernel matrix* **K** *has size of* $N R^2 \times N R^2$.

Theorem (Low rank Adjustment for the approximate Hessian **H**)

The approximate Hessian **H** *can be decomposed as*

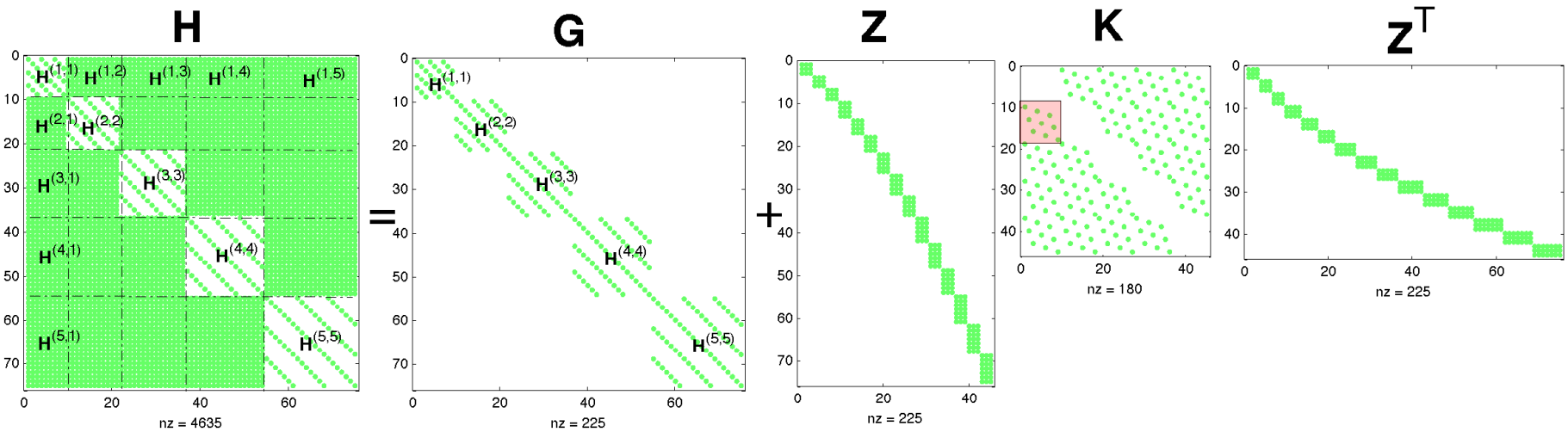
$$\mathbf{H} = \mathbf{G} + \mathbf{Z}\mathbf{K}\mathbf{Z}^T.$$

$$\mathbf{G} = \mathbf{P}\left(\bigoplus_{n=1}^{N}\bigoplus_{i_n=1}^{I_n}\left(\Gamma^{(n,n)} + \operatorname{diag}\{\alpha_n \boldsymbol{a}_{i_n:}^{(n)\bullet[-2]} + \mu\}\right)\right)\mathbf{P}^T,$$

$$\mathbf{Z} = \bigoplus_{n=1}^{N}\left(\mathbf{I}_R \otimes \mathbf{A}^{(n)}\right) \in \mathbb{R}^{\prod I_n \times NR^2},$$

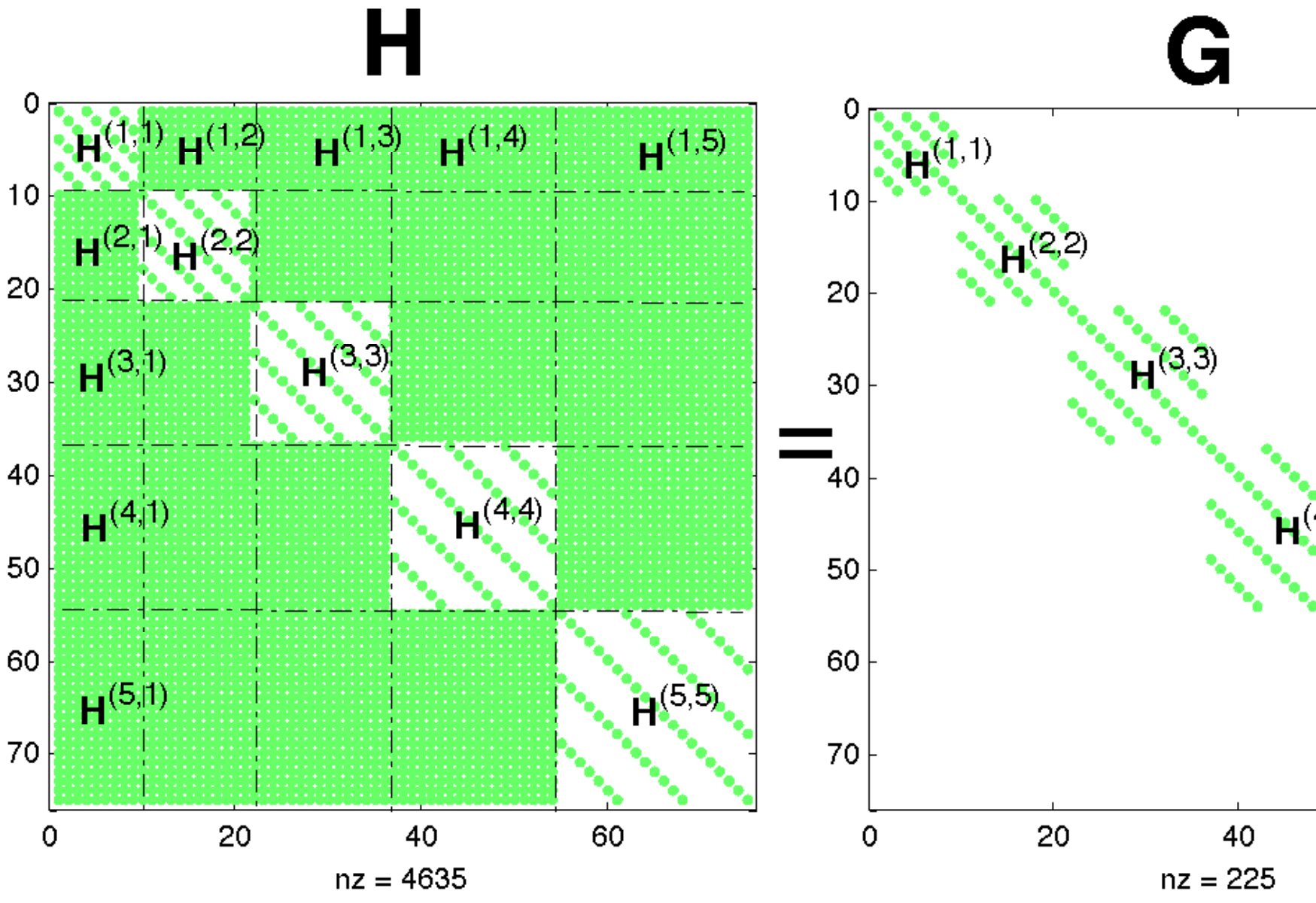
$$\mathbf{K} = \left[\mathbf{K}^{(n,m)}\right]_{n,m} \in \mathbb{R}^{NR^2 \times NR^2},\ \mathbf{K}^{(n,m)} = \begin{cases} \Pi_{R,R}\operatorname{diag}\left(\operatorname{vec}\left(\Gamma^{(n,m)}\right)\right), & m \neq n, \\ \mathbf{0}, & m = n. \end{cases}$$

Low Rank Adjustment for **H**

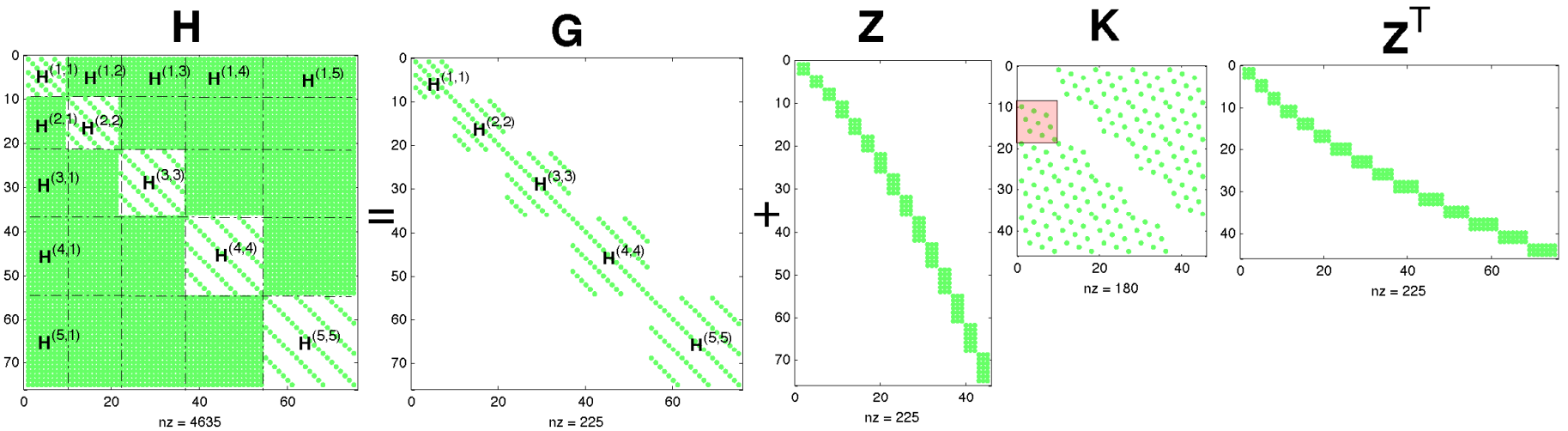


Approximate Hessian for a 5-D tensor. Green dots indicate nonzero elements.

Low Rank Adjustment for **H**



Low Rank Adjustment for **H**

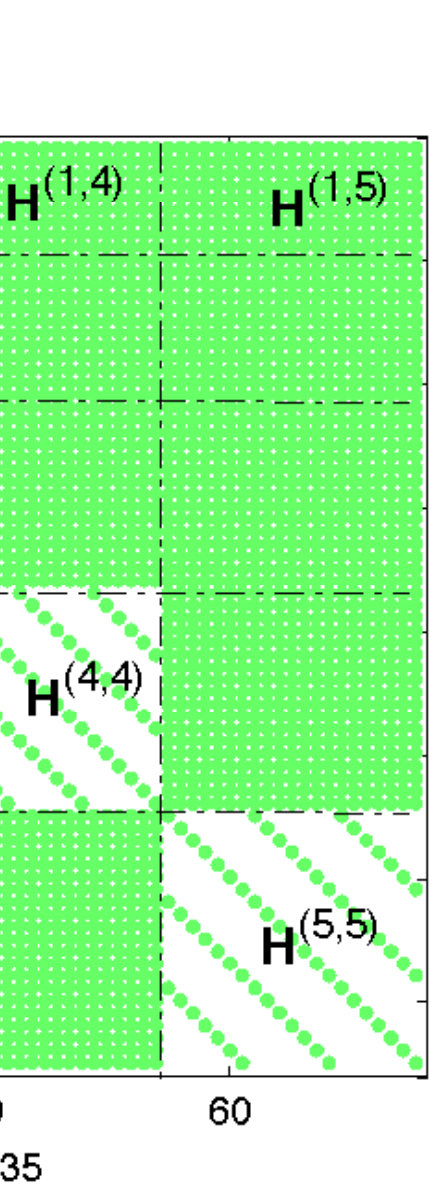


Approximate Hessian for a 5-D tensor. Green dots indicate nonzero elements.

- According to the binomial inverse theorem Horn and Johnson (2012)

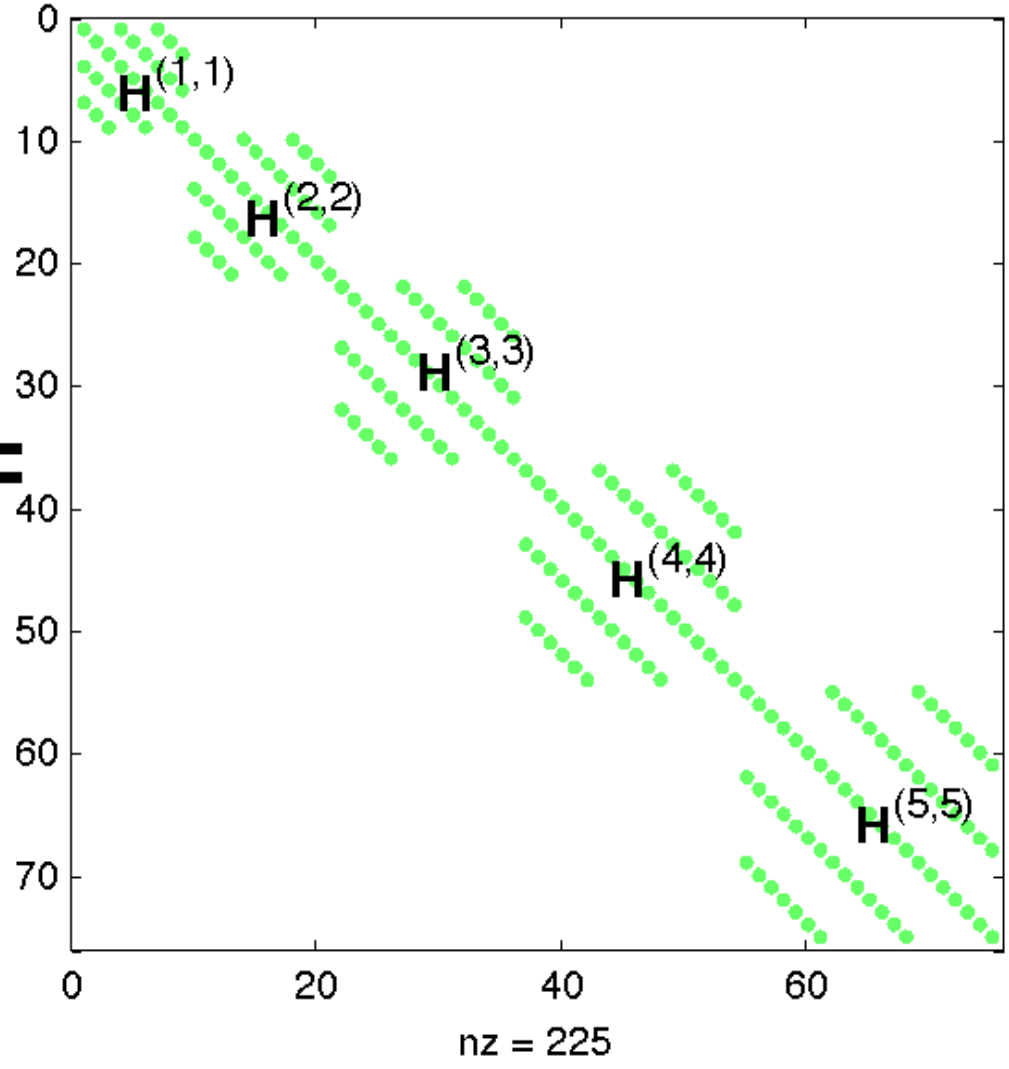
$$\mathbf{H}^{-1} = \mathbf{G}^{-1} - \mathbf{G}^{-1}\mathbf{Z}\left(\mathbf{K}^{-1} + \mathbf{Z}^{T}\mathbf{G}^{-1}\mathbf{Z}\right)^{-1}\mathbf{Z}^{T}\mathbf{G}^{-1}.$$

Low Rank Adjustment for **H**



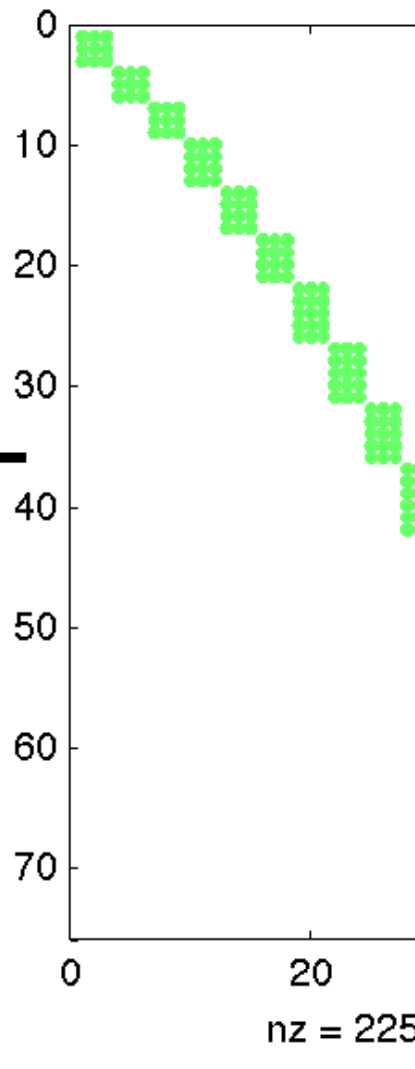
=

G

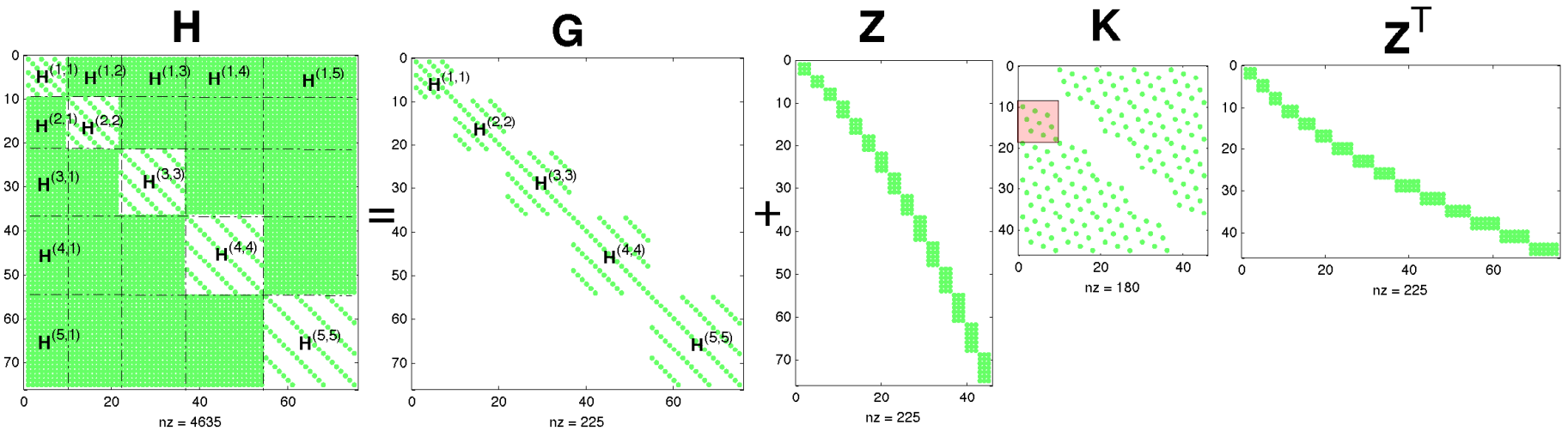


+

Z



Low Rank Adjustment for **H**



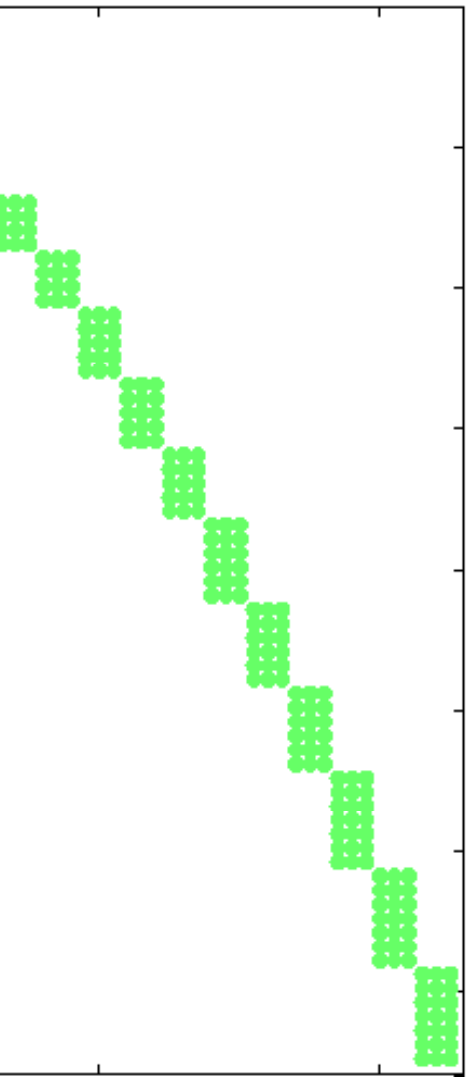
Approximate Hessian for a 5-D tensor. Green dots indicate nonzero elements.

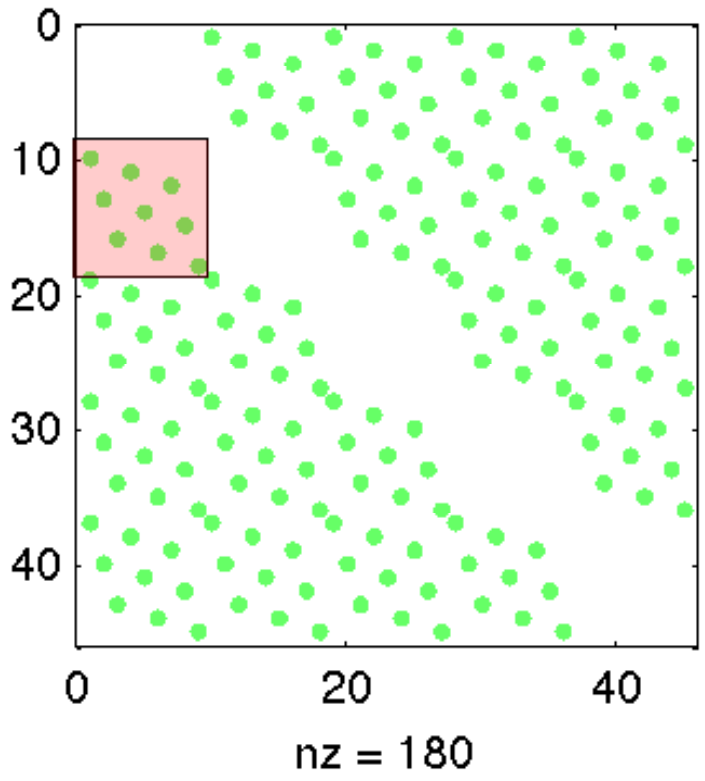
- $\mathbf{G}^{-1} = \mathbf{P}\left(\bigoplus_{n=1}^{N}\bigoplus_{i_n=1}^{I_n}\Theta_{i_n}^{(n)}\right)\mathbf{P}^T,$

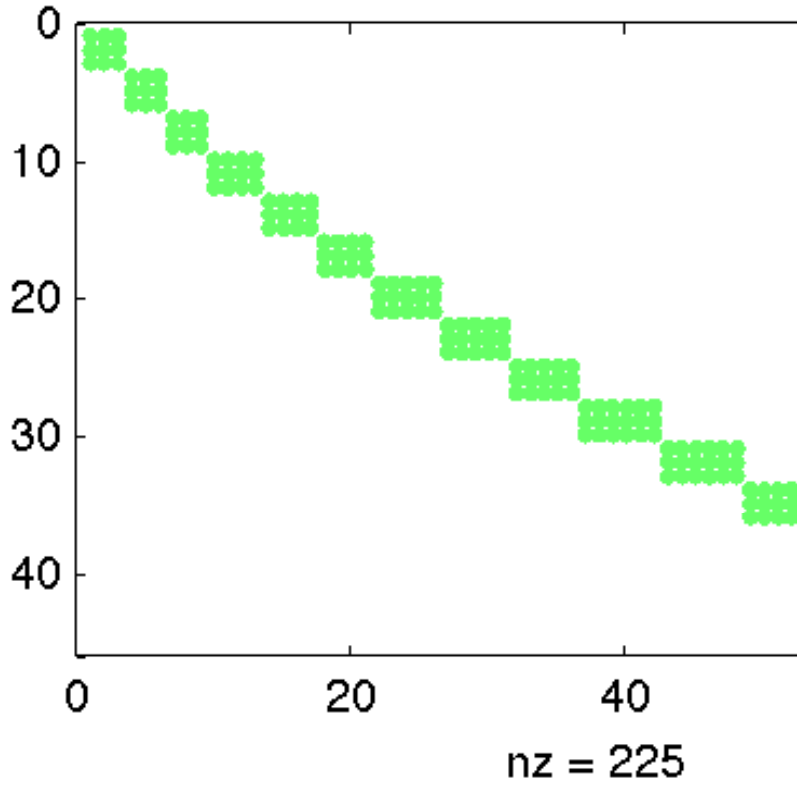
  where $\Theta_{i_n}^{(n)} = \left(\Gamma^{(n,n)} + \operatorname{diag}\left\{\alpha_n \boldsymbol{a}_{i_n:}^{(n)\bullet[-2]} + \mu\right\}\right)^{-1} \in \mathbb{R}^{R\times R}.$

Low Rank Adjustment for **H**

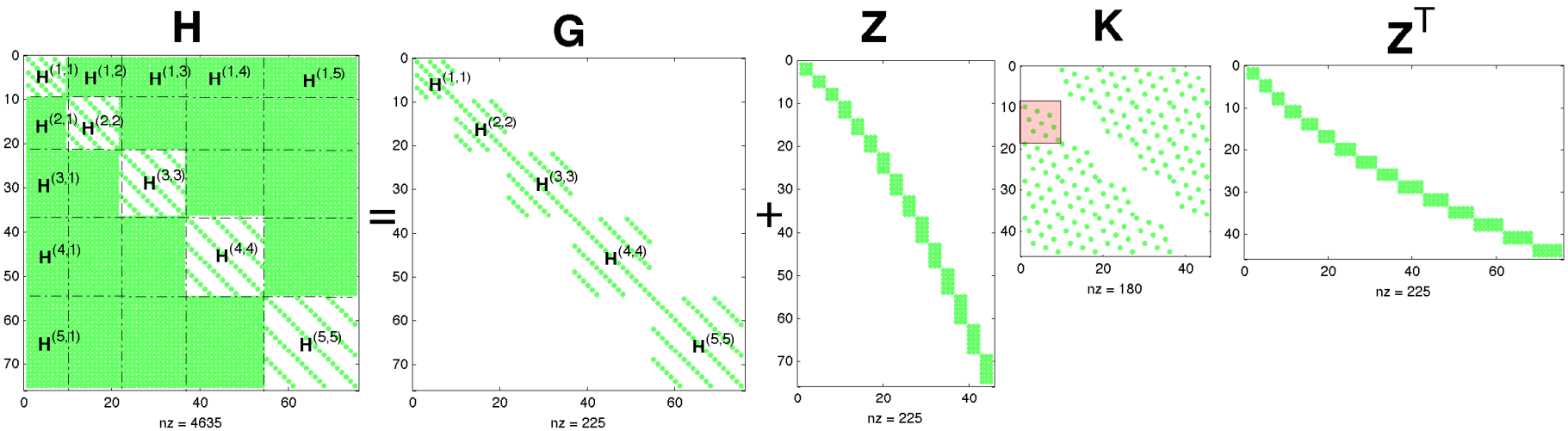
Z **K** $\mathbf{Z}^T$







Low Rank Adjustment for **H**



Approximate Hessian for a 5-D tensor. Green dots indicate nonzero elements.

- $\mathfrak{K} = \mathbf{K}^{-1}$ is a partitioned matrix

$$\mathfrak{K}^{(n,m)} = \begin{cases} -\dfrac{N-2}{N-1}\,\Pi_{R,R}\ \mathrm{diag}\left(\mathrm{vec}\left(\mathbf{A}^{(n)T}\mathbf{A}^{(n)}\right) \oslash \mathrm{vec}\left(\Gamma^{(n,n)}\right)\right), & n = m, \\ \dfrac{1}{N-1}\,\Pi_{R,R}\ \mathrm{diag}\left(\mathbf{1} \oslash \mathrm{vec}\left(\Gamma^{(n,m)}\right)\right), & n \neq m. \end{cases}$$

Damped Gauss-Newton update rule

$$\mathbf{a} \leftarrow \mathbf{a} - \left(\mathbf{H} + \mu \mathbf{I}\right)^{-1} \boldsymbol{g}, \tag{34}$$

Fast dGN for NTF

$$\mathbf{a} \leftarrow \mathbf{a} - \mathbf{G}^{-1} \boldsymbol{g} + \mathbf{L}\, \Phi^{-1} \left(\mathbf{L}^T \boldsymbol{g}\right), \tag{35}$$

Damped Gauss-Newton update rule

$$\mathbf{a} \leftarrow \mathbf{a} - \left(\mathbf{H} + \mu \mathbf{I}\right)^{-1} \boldsymbol{g}, \tag{34}$$

Fast dGN for NTF

$$\mathbf{a} \leftarrow \mathbf{a} - \mathbf{G}^{-1}\,\boldsymbol{g} + \mathbf{L}\,\Phi^{-1}\left(\mathbf{L}^{T}\,\boldsymbol{g}\right), \tag{35}$$

$$\begin{aligned}
\Phi &= \mathcal{K} + \left(\mathbf{I}_N \otimes \Pi_{R,R}\right) \bigoplus_{n=1}^{N} \mathbf{W}^{(n)} \left(\mathbf{I}_N \otimes \Pi_{R,R}\right) \in \mathbb{R}^{NR^2 \times NR^2} \\
\mathbf{W}^{(n)} &= \left[\mathbf{W}^{(n)}_{r,s}\right], \quad \mathbf{W}^{(n)}_{r,s} = \sum_{i_n=1}^{I_n} \Theta^{(n)}_{i_n}\, a^{(n)}_{i_n,r}\, a^{(n)}_{i_n,s} \quad \in \mathbb{R}^{R \times R}
\end{aligned}$$

Damped Gauss-Newton update rule

$$\mathfrak{a} \leftarrow \mathfrak{a} - \left(\mathbf{H} + \mu \mathbf{I}\right)^{-1} \boldsymbol{g}, \tag{34}$$

Fast dGN for NTF

$$\mathfrak{a} \leftarrow \mathfrak{a} - \mathbf{G}^{-1}\,\boldsymbol{g} + \mathbf{L}\,\Phi^{-1}\left(\mathbf{L}^{T}\,\boldsymbol{g}\right), \tag{35}$$

$$\begin{aligned}
\Phi &= \mathfrak{X} + \left(\mathbf{I}_N \otimes \Pi_{R,R}\right) \bigoplus_{n=1}^{N} \mathbf{W}^{(n)} \left(\mathbf{I}_N \otimes \Pi_{R,R}\right) \in \mathbb{R}^{NR^2 \times NR^2} \\
\mathbf{L} &= \mathbf{G}^{-1}\,\mathbf{Z}^{T} = \mathbf{P} \left(\bigoplus_{n=1}^{N} \Upsilon^{(n)} \right) \left(\mathbf{I}_N \otimes \Pi_{R,R}\right) \quad \in \mathbb{R}^{R \sum I_n \times NR^2} \\
\Upsilon^{(n)} &= \left[\boldsymbol{a}_{1:}^{(n)T} \otimes \Theta_{1}^{(n)} \;\cdots\; \boldsymbol{a}_{I_N:}^{(n)T} \otimes \Theta_{I_N}^{(n)} \right]^{T} \quad \in \mathbb{R}^{RI_n \times R^2}
\end{aligned}$$

Damped Gauss-Newton update rule

$$\mathfrak{a} \leftarrow \mathfrak{a} - \left(\mathbf{H} + \mu \mathbf{I}\right)^{-1} \boldsymbol{g}, \tag{34}$$

Fast dGN for NTF

$$\mathfrak{a} \leftarrow \mathfrak{a} - \mathbf{G}^{-1}\, \boldsymbol{g} + \mathbf{L}\, \Phi^{-1} \left(\mathbf{L}^T\, \boldsymbol{g}\right), \tag{35}$$

- For $R \ll I_n, \forall n$, Φ is much smaller than $\mathbf{H} \in \mathbb{R}^{RT \times RT}$.
- Φ^{-1} requires computational complexity of order $O(N^3\, R^6)$, while $\mathbf{H}^{-1}$ is of order $O(N^3\, R^3\, I^3)$ ($I_1 = \cdots = I_N = I$).
- We don't need to construct the approximate Hessian $\mathbf{H}$ and the Jacobian $\mathbf{J}$.

Fast LM with cost $O(NR^6)$

Fast inverse of the approximate Hessian $\mathbf{H}$ ($O(N^3R^6)$)

$$\mathbf{H}^{-1} = \mathbf{G}^{-1} - \mathbf{G}^{-1}\mathbf{Z}\left(\mathbf{K}^{-1} + \mathbf{Z}^T\mathbf{G}^{-1}\mathbf{Z}\right)^{-1}\mathbf{Z}^T\mathbf{G}^{-1}.$$

Fast LM with cost $O(NR^6)$

Fast inverse of the approximate Hessian $\mathbf{H}$ ($O(N^3R^6)$)

$$\mathbf{H}^{-1} = \mathbf{G}^{-1} - \mathbf{G}^{-1}\mathbf{Z}\left(\mathbf{K}^{-1} + \mathbf{Z}^T\mathbf{G}^{-1}\mathbf{Z}\right)^{-1}\mathbf{Z}^T\mathbf{G}^{-1}.$$

Low-rank adjustment form of $\mathbf{K}$

$$\begin{aligned}
\mathbf{K} &= \mathbf{K}_0 + \mathbf{D}\mathbf{F}\mathbf{D}^T \quad \in \mathbb{R}^{NR^2\times NR^2} \\
\mathbf{K}_0 &= \operatorname{bdiag}(\mathbf{K}_n)_{n=1}^N, \quad \mathbf{K}_n = -\mathbf{P}_R \operatorname{dvec}(\Gamma_{nn} \oslash \mathbf{C}_n) \\
\mathbf{D} &= \begin{bmatrix} \mathbf{D}_1 \\ \vdots \\ \mathbf{D}_N \end{bmatrix}, \qquad \mathbf{D}_n = \operatorname{dvec}(1 \oslash \mathbf{C}_n) \\
\mathbf{F} &= \mathbf{P}_R \operatorname{dvec}(\Gamma) \quad \in \mathbb{R}^{R^2\times R^2}, \qquad \Gamma = \circledast_{n=1}^N \mathbf{C}_n\,.
\end{aligned}$$

Fast LM with cost $O(NR^6)$

Fast inverse of the approximate Hessian $\mathbf{H}$ ($O(N^3R^6)$)

$$\mathbf{H}^{-1} = \mathbf{G}^{-1} - \mathbf{G}^{-1}\mathbf{Z}\left(\mathbf{K}^{-1} + \mathbf{Z}^T\mathbf{G}^{-1}\mathbf{Z}\right)^{-1}\mathbf{Z}^T\mathbf{G}^{-1}.$$

Fast inverse of $\mathbf{K}$ ($O(NR^6)$)

$$\mathbf{K}^{-1} = \mathbf{K}_0^{-1} - \mathbf{K}_0^{-1}\mathbf{D}\left(\mathbf{F}^{-1} + \mathbf{D}^T\mathbf{K}_0^{-1}\mathbf{D}\right)^{-1}\mathbf{D}^T\mathbf{K}_0^{-1}$$

Fast LM with cost $O(NR^6)$

Fast inverse of the approximate Hessian $\mathbf{H}$ ($O(N^3R^6)$)

$$\mathbf{H}^{-1} = \mathbf{G}^{-1} - \mathbf{G}^{-1}\mathbf{Z}\left(\mathbf{K}^{-1} + \mathbf{Z}^T\mathbf{G}^{-1}\mathbf{Z}\right)^{-1}\mathbf{Z}^T\mathbf{G}^{-1}.$$

Fast inverse of $\mathbf{H}$ ($O(NR^6)$)

$$\mathbf{H}^{-1} = \mathbf{G}^{-1} - \mathbf{G}^{-1}\mathbf{Z}\left(\mathbf{J}_\mu^{-1} + \mathbf{L}_\mu\mathbf{S}_\mu^{-1}\mathbf{L}_\mu^T\right)\mathbf{Z}^T\mathbf{G}^{-1}$$

where

$$\begin{aligned}
\mathbf{S}_\mu &= \mathbf{Q} - \sum_{n=1}^{N}\mathbf{D}_n\mathbf{K}_n^{-1}\mathbf{J}_n^{-1}\mathbf{K}_n^{-1}\mathbf{D}_n \\
\mathbf{L}_\mu &= \mathbf{J}_\mu^{-1}\mathbf{K}_0^{-1}\mathbf{D}\,. \\
\mathbf{Q} &= -(N-1)\,\mathbf{P}_R\,\mathrm{dvec}(1 \oslash \Gamma) \\
\mathbf{J}_\mu &= \mathrm{bdiag}(\mathbf{J}_n)_{n=1}^{N}, \\
\mathbf{J}_n &= -\mathbf{P}_R\,\mathrm{dvec}(\mathbf{C}_n \oslash \Gamma_{nn}) + (\Gamma_{nn} + \mu\mathbf{I}_R)^{-1} \otimes \mathbf{C}_n\,.
\end{aligned}$$

Selection of barrier and sparse parameters

- An heuristic approach: initialize $\alpha = \alpha_1 = \cdots = \alpha_N$ by a large enough value, then slowly descend down to near-zero after each 10 iterations **?**.
- An alternative approach is based on the Karush-Kuhn-Tucker condition Rojas and Steihaug (2002); Ding et al. (2006)

$$\min \|\mathbf{B}^{(n)}\mathbf{x} - \mathbf{y}^{(n)}\|_2^2 \text{ such that } \begin{cases} \mathbf{B}^{(n)}_{\mathcal{P}}\mathbf{x} = \mathbf{y}^{(n)}_{\mathcal{P}}, \\ \mathbf{x}_{\mathcal{N}} = [x_i, i \in \mathcal{N}] \geq 0, \end{cases} \quad \forall n,$$

  where $\mathbf{B}^{(n)} = [\mathbf{I}, \text{vec}(-\mathbf{A}^{(n)})]$, and $\mathbf{y}^{(n)} = [\mathbf{x}^{(n)}_+, 0]^T$.
- For NTF without P_s,

$$\alpha_n = \max\left(0, \min\left(\text{vec}\left(-\mathbf{A}^{(n)} \circledast \mathbf{F}^{(n)}\right)\right)\right), \quad \forall n.$$

Selection of barrier and sparse parameters

- An heuristic approach: initialize $\alpha = \alpha_1 = \cdots = \alpha_N$ by a large enough value, then slowly descend down to near-zero after each 10 iterations **?**.
- An alternative approach is based on the Karush-Kuhn-Tucker condition Rojas and Steihaug (2002); Ding et al. (2006)

$$\mathfrak{a} \geq 0\,, \quad \mathfrak{a} \circledast \boldsymbol{g} = 0\,, \quad \boldsymbol{g} \geq 0\,.$$

$$\mathbf{A}^{(n)} \circledast \left(\mathbf{F}^{(n)} + \alpha_n \left(\mathbf{A}^{(n)}\right)^{\bullet[-1]} - \beta_n\right) = \mathbf{0}\,, \quad \forall n, \qquad (36)$$

$$\mathbf{F}^{(n)} + \alpha_n \left(\mathbf{A}^{(n)}\right)^{\cdot[-1]} - \beta_n \leq \mathbf{0}\,, \quad \forall n. \qquad (37)$$

$$\min_{\boldsymbol{x}} \|\mathbf{B}^{(n)}\boldsymbol{x} - \boldsymbol{u}^{(n)}\|_2^2 \text{ such that } \begin{cases} \mathbf{B}^{(n)}\boldsymbol{x} \leq \boldsymbol{u}^{(n)}, \\ \boldsymbol{x} = [\alpha_n\, \beta_n]^T \geq \mathbf{0}, \end{cases} \quad \forall n,$$

where $\mathbf{B}^{(n)} = \left[\mathbf{1}\ \operatorname{vec}\left(-\mathbf{A}^{(n)}\right)\right]$, and $\boldsymbol{u}^{(n)} = \operatorname{vec}\left(-\mathbf{A}^{(n)} \circledast \mathbf{F}^{(n)}\right)$.

- For NTF without P_s,

$$\alpha_n = \max\left(0, \min\left(\operatorname{vec}\left(-\mathbf{A}^{(n)} \circledast \mathbf{F}^{(n)}\right)\right)\right), \quad \forall n.$$

Selection of barrier and sparse parameters

- An heuristic approach: initialize $\alpha = \alpha_1 = \cdots = \alpha_N$ by a large enough value, then slowly descend down to near-zero after each 10 iterations **?**.
- An alternative approach is based on the Karush-Kuhn-Tucker condition Rojas and Steihaug (2002); Ding et al. (2006)

$$\min_{\boldsymbol{x}} \|\mathbf{B}^{(n)}\boldsymbol{x} - \boldsymbol{u}^{(n)}\|_2^2 \text{ such that } \begin{cases} \mathbf{B}^{(n)}\,\boldsymbol{x} \leq \boldsymbol{u}^{(n)}, \\ \boldsymbol{x} = [\alpha_n\,\beta_n]^T \geq \mathbf{0}, \end{cases} \quad \forall n,$$

  where $\mathbf{B}^{(n)} = \left[\mathbf{1}\ \text{vec}\left(-\mathbf{A}^{(n)}\right)\right]$, and $\boldsymbol{u}^{(n)} = \text{vec}\left(-\mathbf{A}^{(n)} \circledast \mathbf{F}^{(n)}\right)$.
- For NTF without P_s,

$$\alpha_n = \max\left(0, \min\left(\text{vec}\left(-\mathbf{A}^{(n)} \circledast \mathbf{F}^{(n)}\right)\right)\right), \quad \forall n.$$

Selection of barrier and sparse parameters

- An heuristic approach: initialize $\alpha = \alpha_1 = \cdots = \alpha_N$ by a large enough value, then slowly descend down to near-zero after each 10 iterations **?**.
- An alternative approach is based on the Karush-Kuhn-Tucker condition Rojas and Steihaug (2002); Ding et al. (2006)

$$\min_{\boldsymbol{x}} ||\mathbf{B}^{(n)}\boldsymbol{x} - \boldsymbol{u}^{(n)}||_2^2 \text{ such that } \begin{cases} \mathbf{B}^{(n)}\,\boldsymbol{x} \leq \boldsymbol{u}^{(n)}, \\ \boldsymbol{x} = [\alpha_n\, \beta_n]^T \geq \mathbf{0}, \end{cases} \quad \forall n,$$

  where $\mathbf{B}^{(n)} = \left[\mathbf{1}\ \text{vec}\left(-\mathbf{A}^{(n)}\right)\right]$, and $\boldsymbol{u}^{(n)} = \text{vec}\left(-\mathbf{A}^{(n)} \circledast \mathbf{F}^{(n)}\right)$.
- For NTF without P_s,
  $\alpha_n = \max\left(0, \min\left(\text{vec}\left(-\mathbf{A}^{(n)} \circledast \mathbf{F}^{(n)}\right)\right)\right), \quad \forall n.$

fast LM vs dampled Gauss-Newton (dGN)

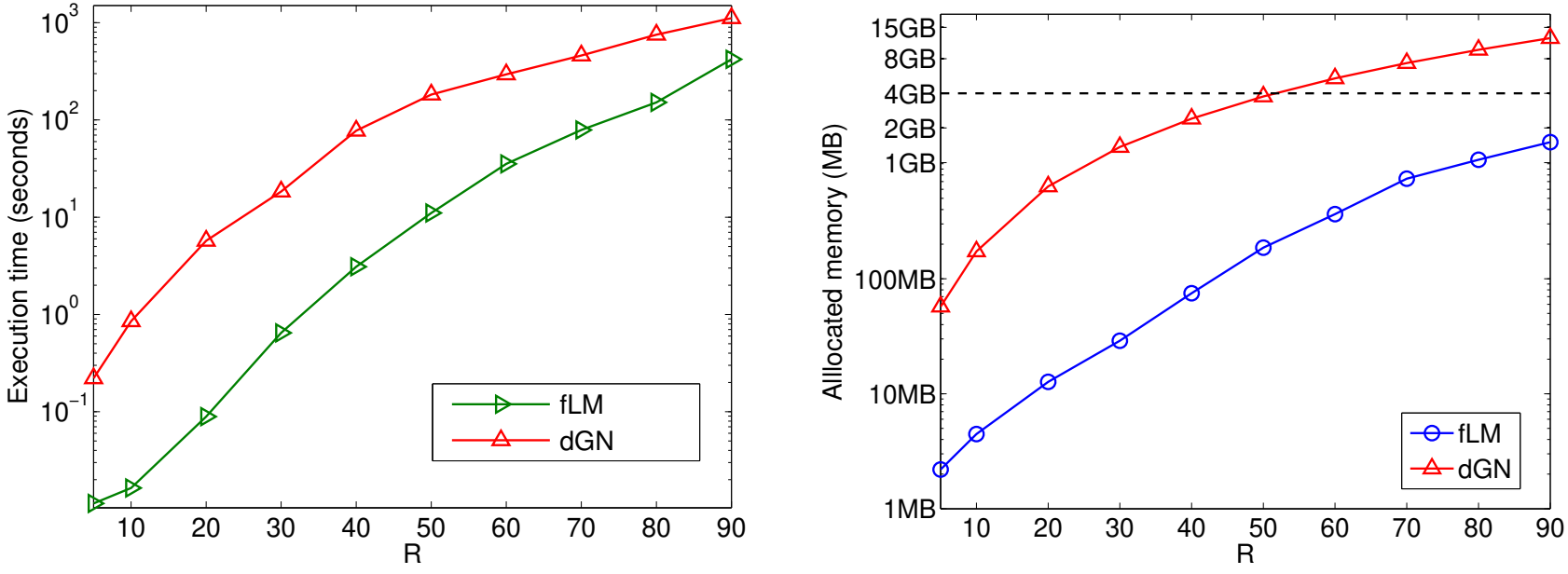


Figure: Memory requirements and execution time per iteration of fLM algorithms in approximation of 100×100×100 dimensional tensors by rank-R tensors where $R = 5, 10, 20, \ldots, 90$.

dGN: damped Gauss-Newton algorithm in the package PARAFAC3W.Tomasi (2003)

References I

Andersson, C. and Bro, R. (2000). The N-way toolbox for MATLAB. *Chemometrics and Intelligent Laboratory Systems*, 52(1):1–4.

Berry, M., Browne, M., Langville, A., Pauca, P., and Plemmons, R. (2007). Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics and Data Analysis*, 52(1):155–173.

Booksh, K. and Kowalski, B. (1994). Error analysis of the generalized rank annihilation method. *Journal of Chemometrics*, 8.

Booksh, K., Lin, Z., Wang, Z., and Kowalski, B. (1994). Extension of trilinear decomposition method with an application to the flow probe sensor. *Analytical Chemistry*, 66.

Carroll, J. and Chang, J. (1970). Analysis of individual differences in multidimensional scaling via an *n*-way generalization of Eckart–Young decomposition. *Psychometrika*, 35(3):283–319.

References II

Comon, P., Luciani, X., and de Almeida, A. L. F. (2009). Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics*, 23.

Comon, P. and Rajih, M. (2006). Blind identification of under-determined mixtures based on the characteristic function. *Signal Processing*, 86(9):2271 – 2281. Special Section: Signal Processing in {UWB} Communications.

Ding, C., Li, T., Peng, W., and Park, H. (2006). Orthogonal nonnegative matrix tri-factorizations for clustering. In *KDD06*, pages 126–135, New York, NY, USA. ACM Press.

Eigenvector (2012). PLS Toolbox.

Faber, N., Buydens, L., and Kateman, G. (1994). Generalized rank annihilation. iii: Practical implementation. *Journal of Chemometrics*, 8.

References III

Harshman, R. (1970). Foundations of the PARAFAC procedure: Models and conditions for an explanatory multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84.

Harshman, R. A. (1972). Determination and proof of minimum uniqueness conditions for PARAFAC1. *UCLA Working Papers in Phonetics*, 22.

Hitchcock, F. (1927). Multiple invariants and generalized rank of a *p*-way matrix or tensor. *Journal of Mathematics and Physics*, 7:39–79.

Horn, R. A. and Johnson, C. R. (1991). *Topics in matrix analysis*. Cambridge University Press, Cambridge.

Horn, R. A. and Johnson, C. R. (2012). *Matrix Analysis*. Cambridge University Press.

References IV

Jaderberg, M., Vedaldi, A., and Zisserman, A. (2014). Speeding up convolutional neural networks with low rank expansions. *CoRR*, abs/1405.3866.

Kolda, T. and Bader, B. (2009). Tensor decompositions and applications. *SIAM Review*, 51(3):455–500.

Kruskal, J. (1977). Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra Appl.*, 18:95–138.

Langville, A. N., Meyer, C. D., and Albright, R. (2006). Initializations for the nonnegative matrix factorization. In *Proc. of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Philadelphia, USA.

Nion, D. and Sidiropoulos, N. (2009). Adaptive Algorithms to Track the PARAFAC Decomposition of a Third-Order Tensor. *IEEE Transactions on Signal Processing*, 57(6):2299–2310.

References V

Phan, A.-H., Tichavský, P., and Cichocki, A. (2015). Tensor deflation for CANDECOMP/PARAFAC. Part 1: Alternating Subspace Update Algorithm. *IEEE Transaction on Signal Processing*, 63(12):5924–5938.

Ragnarsson, S. and Loan, C. F. V. (2012). Block tensor unfoldings. *SIAM J. Matrix Analysis Applications*, 33(1):149–169.

Rajih, M., Comon, P., and Harshman, R. A. (2008). Enhanced line search: A novel method to accelerate PARAFAC. *SIAM Journal of Matrix Analysis and Applications*, 30(3):1128–1147.

Rojas, M. and Steihaug, T. (2002). An interior-point trust-region-based method for large-scale non-negative regularization. *Inverse Problems*, 18:1291–1307.

Sidiropoulos, N., Bro, R., and Giannakis, G. (2000). Parallel factor analysis in sensor array processing. *IEEE Transactions on Signal Processing*, 48(8):2377–2388.

References VI

Strassen, V. (1969). Gaussian elimination is not optimal. *Numer. Math.*, 13(4):354–356.

Tomasi, G. (2003). INDAFAC and PARAFAC3W. `http://www.models.kvl/dk/source/indafac/index.asp`.

Tomasi, G. (2006). *Practical and Computational Aspects in Chemometric Data Analysis*. PhD thesis, Royal Veterinary and Agricultural University, Frederiksberg, Denmark.

Yeredor, A. (2000). Blind source separation via the second characteristic function. *Signal Processing*, 80(5):897–902.