Numerical optimization-based tensor algorithms

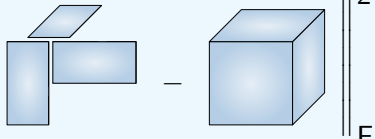
Nico Vervliet

Lieven De Lathauwer

EURASIP Summer School
August 29, 2018

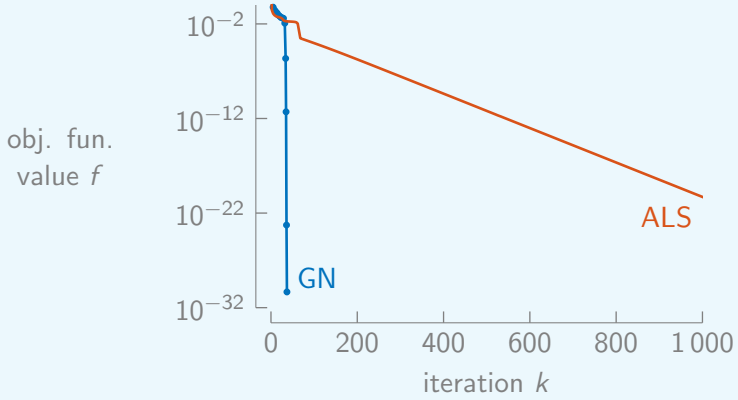
A second-order optimization framework for the computation of tensor decompositions

$$\min \left\| \quad - \quad \right\|_F^2$$



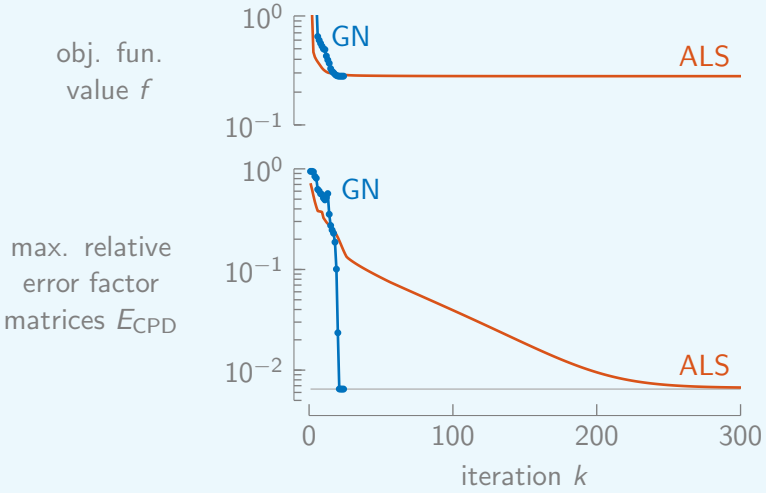
subject to parametric, box or soft constraints
coupling constraints
symmetry constraints

Alternating Least Squares versus Gauss–Newton: exact case



Rank-10 tensor of size $250 \times 250 \times 250$, correlated factor vectors.

Alternating Least Squares versus Gauss–Newton: noisy case

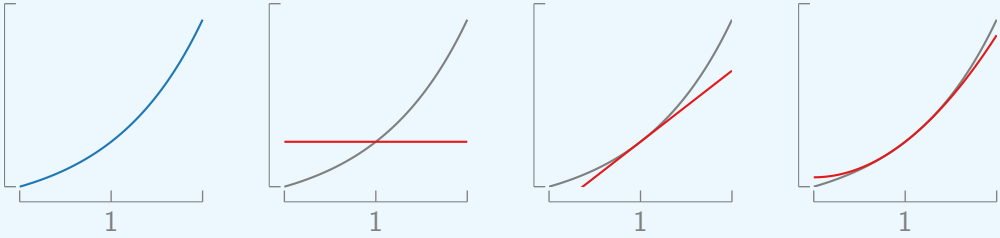


Rank-10 tensor of size $250 \times 250 \times 250$, correlated factor vectors, SNR is 20 dB.

Advantages of second-order information for tensor computations

- Good convergence properties
  Global convergence; up to quadratic convergence near local optima
- Robust to initialization
  Often few initializations required
- Less susceptible to swamps
- Easy to incorporate constraints
  E.g., parametric, box and soft constraints
- All multilinear structure can be exploited
- Same asymptotic complexity as ALS

Intermezzo: local approximation using a Taylor series expansion



$$
\begin{aligned}
f(x) &\approx f(1) \\
&\approx f(1) + \frac{\mathrm{d}f(1)}{\mathrm{d}x}(x-1) \\
&\approx f(1) + \frac{\mathrm{d}f(1)}{\mathrm{d}x}(x-1) + \frac{1}{2}\frac{\mathrm{d}^2 f(1)}{\mathrm{d}x^2}(x-1)^2
\end{aligned}
$$

Newton approach: local second-order approximation of objective function

Instead of using the nonlinear least squares objective function

$$\min_{\mathbf{z}} f(\mathbf{z}) \qquad \text{with} \qquad f(\mathbf{z}) = \frac{1}{2} ||m(\mathbf{z}) - \mathbf{t}||_{\mathrm{F}}^2 ,$$

The model is $m(\mathbf{z}) = \mathrm{vec}\left(\llbracket \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket\right)$ with variables $\mathbf{z} = \left[\mathrm{vec}\left(\mathbf{A}\right); \mathrm{vec}\left(\mathbf{B}\right); \mathrm{vec}\left(\mathbf{C}\right)\right]$ and $\mathbf{t} = \mathrm{vec}\left(\mathcal{T}\right)$.

Newton approach: local second-order approximation of objective function

Instead of using the nonlinear least squares objective function

$$\min_{\mathbf{z}} f(\mathbf{z}) \qquad \text{with} \qquad f(\mathbf{z}) = \frac{1}{2} ||m(\mathbf{z}) - \mathbf{t}||_{\mathrm{F}}^2 ,$$

we solve

$$\min_{\mathbf{p}_k} \tilde{f}(\mathbf{p}_k) \qquad \text{with} \qquad \tilde{f}(\mathbf{p}_k) \approx f(\mathbf{z}_k) + \mathbf{p}_k^{\mathsf{T}} \cdot \underbrace{\nabla_{\mathbf{z}} f(\mathbf{z}_k)}_{\mathbf{g}_k} + \frac{1}{2} \mathbf{p}_k^{\mathsf{T}} \cdot \underbrace{\nabla_{\mathbf{z}}^2 f(\mathbf{z}_k)}_{\mathbf{H}_k} \cdot \mathbf{p}_k$$

The model is $m(\mathbf{z}) = \mathrm{vec}\left(\llbracket \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket\right)$ with variables $\mathbf{z} = \left[\mathrm{vec}\left(\mathbf{A}\right); \mathrm{vec}\left(\mathbf{B}\right); \mathrm{vec}\left(\mathbf{C}\right)\right]$ and $\mathbf{t} = \mathrm{vec}\left(\mathcal{T}\right)$.

Newton approach: local second-order approximation of objective function

Instead of using the nonlinear least squares objective function

$$\min_{\mathbf{z}} f(\mathbf{z}) \qquad \text{with} \qquad f(\mathbf{z}) = \frac{1}{2}\,||m(\mathbf{z}) - \mathbf{t}||_{\mathrm{F}}^2\,,$$

we solve

$$\min_{\mathbf{p}_k} \tilde{f}(\mathbf{p}_k) \qquad \text{with} \qquad \tilde{f}(\mathbf{p}_k) \approx f(\mathbf{z}_k) + \mathbf{p}_k^{\mathsf{T}} \cdot \underbrace{\nabla_{\mathbf{z}} f(\mathbf{z}_k)}_{\mathbf{g}_k} + \frac{1}{2}\mathbf{p}_k^{\mathsf{T}} \cdot \underbrace{\nabla_{\mathbf{z}}^2 f(\mathbf{z}_k)}_{\mathbf{H}_k} \cdot \mathbf{p}_k$$

and $\mathbf{p}_k$ can be found from

$$\mathbf{H}_k \mathbf{p}_k = -\mathbf{g}_k$$

The variables are then updated as

$$\mathbf{z}_{k+1} = \mathbf{z}_k + \alpha_k \mathbf{p}_k$$

The model is $m(\mathbf{z}) = \mathrm{vec}\,(\llbracket \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket)$ with variables $\mathbf{z} = \left[\mathrm{vec}\,(\mathbf{A})\,; \mathrm{vec}\,(\mathbf{B})\,; \mathrm{vec}\,(\mathbf{C})\right]$ and $\mathbf{t} = \mathrm{vec}\,(\mathcal{T})$.

Quasi-Newton and Gauss–Newton

Unfortunately, computing the Hessian $\mathbf{H}_k$ is

- expensive to compute
- not always positive semidefinite

Therefore, $\mathbf{H}_k$ is approximated:

- for gradient or steepest descent, $\mathbf{H}_k = \mathbf{I}$
- for NCG, $\mathbf{H}_k = \mathbf{I} - \gamma \mathbf{p}_{k-1} \boldsymbol{\delta}^{\mathsf{T}}$
- for BFGS, $\mathbf{H}_k = \mathbf{H}_{k-1} + \mathbf{U}_{k-1} + \mathbf{V}_{k-1}$
- for Gauss–Newton (GN), $\mathbf{H}_k = \mathbf{J}_k^{\mathsf{T}} \mathbf{J}_k$
- for Levenberg–Marquardt (LM), $\mathbf{H}_k = \mathbf{J}_k^{\mathsf{T}} \mathbf{J}_k + \lambda \mathbf{I}$

Quasi-Newton and Gauss–Newton

Unfortunately, computing the Hessian $\mathbf{H}_k$ is

- expensive to compute
- not always positive semidefinite

Therefore, $\mathbf{H}_k$ is approximated:

- for gradient or steepest descent, $\mathbf{H}_k = \mathbf{I}$
- for NCG, $\mathbf{H}_k = \mathbf{I} - \gamma \mathbf{p}_{k-1} \boldsymbol{\delta}^\mathsf{T}$
- for BFGS, $\mathbf{H}_k = \mathbf{H}_{k-1} + \mathbf{U}_{k-1} + \mathbf{V}_{k-1}$
- for Gauss–Newton (GN), $\mathbf{H}_k = \mathbf{J}_k^\mathsf{T} \mathbf{J}_k$
- for Levenberg–Marquardt (LM), $\mathbf{H}_k = \mathbf{J}_k^\mathsf{T} \mathbf{J}_k + \lambda \mathbf{I}$

We focus on GN in this talk

Direct and iterative methods for solving $\mathbf{Hp} = -\mathbf{g}$

- Using direct algorithms
  - Pseudoinversion

$$\mathbf{p} = -\mathbf{H}^{\dagger}\mathbf{g}$$

  - LDL factorization

$$\mathbf{p} = -\mathbf{L}^{-\intercal}\mathbf{D}^{\dagger}\mathbf{L}^{-1}\mathbf{g}$$

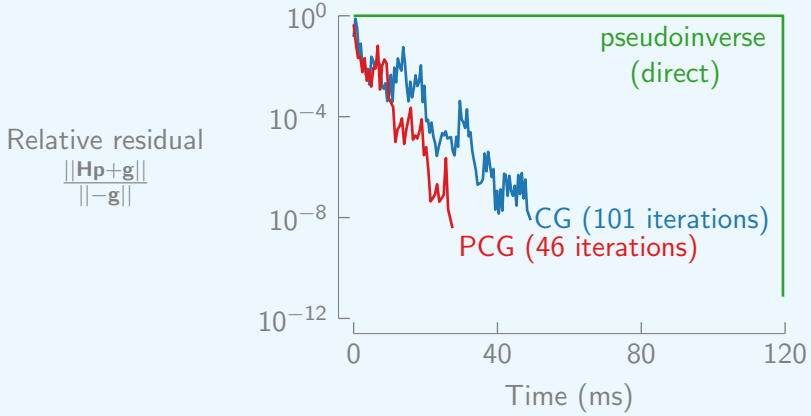
LDL factorization: $\mathbf{H} = \mathbf{L}\mathbf{D}\mathbf{L}^{\intercal}$ with $\mathbf{L}$ lower triangular and $\mathbf{D}$ diagonal.

Direct and iterative methods for solving $\mathbf{Hp} = -\mathbf{g}$

- Using direct algorithms
  - Pseudoinversion
    $$\mathbf{p} = -\mathbf{H}^{\dagger}\mathbf{g}$$
  - LDL factorization
    $$\mathbf{p} = -\mathbf{L}^{-\intercal}\mathbf{D}^{\dagger}\mathbf{L}^{-1}\mathbf{g}$$
- Using conjugate gradients (inexact)
  - Iterative solution using only products of form
    $$\mathbf{y}_l = \mathbf{H}\mathbf{x}_{l-1}$$
  - Preconditioning techniques
    $$\mathbf{M}^{-1}\mathbf{H}\mathbf{p} = -\mathbf{M}^{-1}\mathbf{g}$$

LDL factorization: $\mathbf{H} = \mathbf{L}\mathbf{D}\mathbf{L}^{\intercal}$ with $\mathbf{L}$ lower triangular and $\mathbf{D}$ diagonal.

Direct and iterative methods for solving $\mathbf{Hp} = -\mathbf{g}$



Rank-5 tensor of size $40 \times 40 \times 40$, correlated factor vectors, SNR is 20 dB.

Inexact Gauss–Newton for CPD

Goal: find best variables $\mathbf{z} = \left[\text{vec}\left(\mathbf{A}\right); \text{vec}\left(\mathbf{B}\right); \text{vec}\left(\mathbf{C}\right)\right]$ via

$$\min_{\mathbf{z}} \frac{1}{2} ||[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] - \mathcal{T}||_{\text{F}}^2$$

Starting from $\mathbf{z}_0$, each iteration we

- solve

  $$\mathbf{H}_k \mathbf{p}_k = -\mathbf{g}_k$$

  for $\mathbf{p}_k$ with $\mathbf{H}_k = \mathbf{J}_k^{\text{T}} \mathbf{J}_k$
- and update

  $$\mathbf{z}_k = \mathbf{z}_{k-1} + \alpha_k \mathbf{p}_k$$

Inexact Gauss–Newton for CPD

Solve system $\mathbf{Hp} = -\mathbf{g}$ with CG using only Hessian-vector products

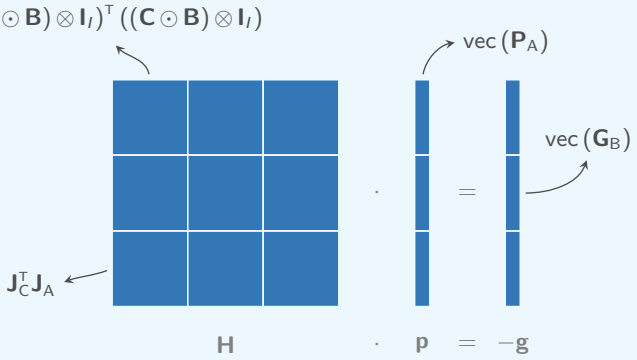
$$\mathbf{y}_l = \mathbf{J}^{\mathsf{T}}\mathbf{J}\mathbf{x}_{l-1}$$

starting from $\mathbf{x}_0$.

Partition all computed variables according to factor matrices:

$$\mathbf{g} = \left[\mathbf{g}_A; \mathbf{g}_B; \mathbf{g}_C\right]$$
$$\mathbf{p} = \left[\mathbf{p}_A; \mathbf{p}_B; \mathbf{p}_C\right]$$
$$\mathbf{J} = \left[\mathbf{J}_A, \mathbf{J}_B, \mathbf{J}_C\right]$$
$$\mathbf{x} = \left[\mathbf{x}_A; \mathbf{x}_B; \mathbf{x}_C\right]$$
$$\mathbf{y} = \left[\mathbf{y}_A; \mathbf{y}_B; \mathbf{y}_C\right]$$

Inexact Gauss–Newton for CPD



Inexact Gauss–Newton for CPD

Gradient is computed using matricized tensor times Khatri–Rao product (MTKRPROD) on residual $\mathcal{R} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] - \mathcal{T}$

$$\mathbf{g}_\mathsf{A} = \operatorname{vec}\left(\mathbf{R}_{(1)}(\mathbf{C} \odot \mathbf{B})\right)$$
$$\mathbf{g}_\mathsf{B} = \operatorname{vec}\left(\mathbf{R}_{(2)}(\mathbf{C} \odot \mathbf{A})\right)$$
$$\mathbf{g}_\mathsf{C} = \operatorname{vec}\left(\mathbf{R}_{(3)}(\mathbf{B} \odot \mathbf{A})\right)$$

Gramian-vector product $\mathbf{y} = \mathbf{J}^\mathsf{T}\mathbf{J}\mathbf{x}$ is computed as

$$\mathbf{y}_\mathsf{A} = \mathbf{J}_\mathsf{A}^\mathsf{T}\mathbf{J}_\mathsf{A}\mathbf{x}_\mathsf{A} + \mathbf{J}_\mathsf{A}^\mathsf{T}\mathbf{J}_\mathsf{B}\mathbf{x}_\mathsf{B} + \mathbf{J}_\mathsf{A}^\mathsf{T}\mathbf{J}_\mathsf{C}\mathbf{x}_\mathsf{C}$$

and so on for $\mathbf{y}_\mathsf{B}$ and $\mathbf{y}_\mathsf{C}$, and

$$\mathbf{J}_\mathsf{A}^\mathsf{T}\mathbf{J}_\mathsf{A}\mathbf{x}_\mathsf{A} = \operatorname{vec}(\mathbf{X}_\mathsf{A}\mathbf{W}) \quad \text{with} \quad \mathbf{W} = (\mathbf{C}^\mathsf{T}\mathbf{C}) * (\mathbf{B}^\mathsf{T}\mathbf{B})$$
$$\mathbf{J}_\mathsf{A}^\mathsf{T}\mathbf{J}_\mathsf{B}\mathbf{x}_\mathsf{B} = \operatorname{vec}(\mathbf{A}\mathbf{W}) \quad \text{with} \quad \mathbf{W} = (\mathbf{C}^\mathsf{T}\mathbf{C}) * (\mathbf{X}_\mathsf{B}^\mathsf{T}\mathbf{B})$$
$$\mathbf{J}_\mathsf{A}^\mathsf{T}\mathbf{J}_\mathsf{C}\mathbf{x}_\mathsf{C} = \operatorname{vec}(\mathbf{A}\mathbf{W}) \quad \text{with} \quad \mathbf{W} = (\mathbf{X}_\mathsf{C}^\mathsf{T}\mathbf{C}) * (\mathbf{B}^\mathsf{T}\mathbf{B})$$

Parametric constraints

Each factor matrix, e.g., $\mathbf{A}$, is a function of some underlying variable $\mathbf{z}$.

- Nonnegativity by squaring

$$\mathbf{A} = \mathbf{Z} * \mathbf{Z}$$

Parametric constraints

Each factor matrix, e.g., $\mathbf{A}$, is a function of some underlying variable $\mathbf{z}$.

- Nonnegativity by squaring

$$\mathbf{A} = \mathbf{Z} * \mathbf{Z}$$

- Normalized matrix

$$a_{ir} = z_{ir} / \sqrt{\sum_k z_{kr}^2}$$

Parametric constraints

Each factor matrix, e.g., $\mathbf{A}$, is a function of some underlying variable $\mathbf{z}$.

- Nonnegativity by squaring

$$\mathbf{A} = \mathbf{Z} * \mathbf{Z}$$

- Normalized matrix

$$a_{ir} = z_{ir} / \sqrt{\sum_k z_{kr}^2}$$

- Polynomial constraint

$$a_{ir} = z_{0r} + z_{1r} t_i + z_{2r} t_i^2 + \ldots + z_{dr} t_i^d$$

or in matrix form $\mathbf{A} = \mathbf{MZ}$ with

$$a_{ir} = \underbrace{\begin{bmatrix} 1 & t_i & t_i^2 & \ldots & t_i^d \end{bmatrix}}_{\mathbf{M}(i,:)} \underbrace{\begin{bmatrix} z_{0r} & z_{1r} & z_{2r} & \ldots & z_{dr} \end{bmatrix}^{\mathsf{T}}}_{\mathbf{z}_r^{\mathsf{T}}}$$

Parametric constraints

Each factor matrix, e.g., $\mathbf{A}$, is a function of some underlying variable $\mathbf{z}$.

- Nonnegativity by squaring

$$\mathbf{A} = \mathbf{Z} * \mathbf{Z}$$

- Normalized matrix

$$a_{ir} = z_{ir} / \sqrt{\sum_k z_{kr}^2}$$

- Polynomial constraint

$$a_{ir} = z_{0r} + z_{1r} t_i + z_{2r} t_i^2 + \ldots + z_{dr} t_i^d$$

or in matrix form $\mathbf{A} = \mathbf{MZ}$ with

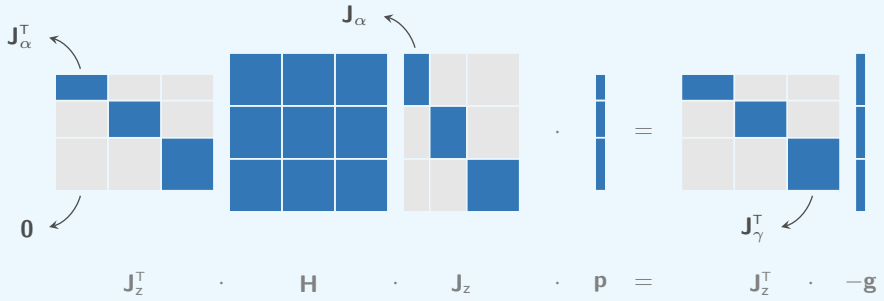
$$a_{ir} = \underbrace{\begin{bmatrix} 1 & t_i & t_i^2 & \ldots & t_i^d \end{bmatrix}}_{\mathbf{M}(i,:)} \underbrace{\begin{bmatrix} z_{0r} & z_{1r} & z_{2r} & \ldots & z_{dr} \end{bmatrix}^{\mathsf{T}}}_{\mathbf{z}_r^{\mathsf{T}}}$$

- Hankel structure

$$a_{ir} = z_{i+r-1}$$

Parametric constraints

Optimize w.r.t. underlying variables $\mathbf{z} = [\boldsymbol{\alpha}; \boldsymbol{\beta}; \boldsymbol{\gamma}]$ instead of factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$



Example: nonnegativity constraint using squared variables

Suppose $\mathbf{A} = \mathbf{D} * \mathbf{D}$, i.e., $\boldsymbol{\alpha} = \mathrm{vec}\,(\mathbf{D})$, then

- the gradient w.r.t. $\boldsymbol{\alpha}$ is computed as

$$\mathbf{J}_{\alpha}^{\mathrm{T}} \mathrm{vec}\,(\mathbf{G}_{\mathrm{A}}) = \mathrm{vec}\,(2\mathbf{D} * \mathbf{G}_{\mathrm{A}})$$

Example: nonnegativity constraint using squared variables

Suppose $\mathbf{A} = \mathbf{D} * \mathbf{D}$, i.e., $\boldsymbol{\alpha} = \mathrm{vec}(\mathbf{D})$, then

- the gradient w.r.t. $\boldsymbol{\alpha}$ is computed as

$$\mathbf{J}_{\alpha}^{\mathrm{T}} \mathrm{vec}(\mathbf{G}_{\mathrm{A}}) = \mathrm{vec}(2\mathbf{D} * \mathbf{G}_{\mathrm{A}})$$

- the Gramian-vector product w.r.t. $\boldsymbol{\alpha}$ is computed as

$$\begin{aligned} \tilde{\mathbf{X}}_{\mathrm{A}} &= 2\mathbf{D} * \mathbf{X}_{\alpha} \\ \mathrm{vec}\left(\tilde{\mathbf{Y}}_{\mathrm{A}}\right) &= \mathbf{H}_{\mathrm{AA}} \mathrm{vec}\left(\tilde{\mathbf{X}}_{\mathrm{A}}\right) + \mathbf{H}_{\mathrm{AB}} \mathrm{vec}(\mathbf{X}_{\mathrm{B}}) + \mathbf{H}_{\mathrm{AC}} \mathrm{vec}(\mathbf{X}_{\mathrm{C}}) \\ \mathbf{Y}_{\alpha} &= 2\mathbf{D} * \tilde{\mathbf{Y}}_{\mathrm{A}} \end{aligned}$$

Example: polynomial constraints

Suppose $\mathbf{A} = \mathbf{MQ}$, i.e., $\boldsymbol{\alpha} = \mathrm{vec}\,(\mathbf{Q})$, then

- the gradient w.r.t. $\boldsymbol{\alpha}$ is computed as

$$\mathbf{J}_{\alpha}^{\mathsf{T}} \mathrm{vec}\,(\mathbf{G}_{\mathrm{A}}) = \mathrm{vec}\,(\mathbf{M}^{\mathsf{T}} \mathbf{G}_{\mathrm{A}})$$

Example: polynomial constraints

Suppose $\mathbf{A} = \mathbf{M}\mathbf{Q}$, i.e., $\boldsymbol{\alpha} = \mathrm{vec}\,(\mathbf{Q})$, then

- the gradient w.r.t. $\boldsymbol{\alpha}$ is computed as

$$\mathbf{J}_{\alpha}^{\mathsf{T}}\mathrm{vec}\,(\mathbf{G}_{\mathsf{A}}) = \mathrm{vec}\,(\mathbf{M}^{\mathsf{T}}\mathbf{G}_{\mathsf{A}})$$

- the Gramian-vector product w.r.t. $\boldsymbol{\alpha}$ is computed as

$$\begin{aligned}
\tilde{\mathbf{X}}_{\mathsf{A}} &= \mathbf{M}\mathbf{X}_{\alpha} \\
\mathrm{vec}\left(\tilde{\mathbf{Y}}_{\mathsf{A}}\right) &= \mathbf{H}_{\mathsf{AA}}\mathrm{vec}\left(\tilde{\mathbf{X}}_{\mathsf{A}}\right) + \mathbf{H}_{\mathsf{AB}}\mathrm{vec}\,(\mathbf{X}_{\mathsf{B}}) + \mathbf{H}_{\mathsf{AC}}\mathrm{vec}\,(\mathbf{X}_{\mathsf{C}}) \\
\mathbf{Y}_{\alpha} &= \mathbf{M}^{\mathsf{T}}\tilde{\mathbf{Y}}_{\mathsf{A}}
\end{aligned}$$

Active set methods for bound constraints

$$\min_{\mathbf{z}} \frac{1}{2} ||[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] - \mathcal{T}||_{\mathsf{F}}^2 \quad \text{subject to} \quad \mathbf{l} \leq \mathbf{z} \leq \mathbf{u}$$

Define the active set $\mathcal{A}$ and the inactive set $\mathcal{I}$ and partition accordingly

$$\mathcal{A} = \{i \mid l_i = z_i \text{ or } z_i = u_i\},$$
$$\mathcal{I} = \{i \mid l_i < z_i \text{ and } z_i < u_i\},$$

Active set methods for bound constraints

$$\min_{\mathbf{z}} \frac{1}{2} ||[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] - \mathcal{T}||_{\mathsf{F}}^2 \quad \text{subject to} \quad \mathbf{l} \leq \mathbf{z} \leq \mathbf{u}$$

Define the active set $\mathcal{A}$ and the inactive set $\mathcal{I}$ and partition accordingly

$$\mathcal{A} = \{i \mid l_i = z_i \text{ or } z_i = u_i\},$$
$$\mathcal{I} = \{i \mid l_i < z_i \text{ and } z_i < u_i\},$$

The step $\mathbf{p}$ is then computed from

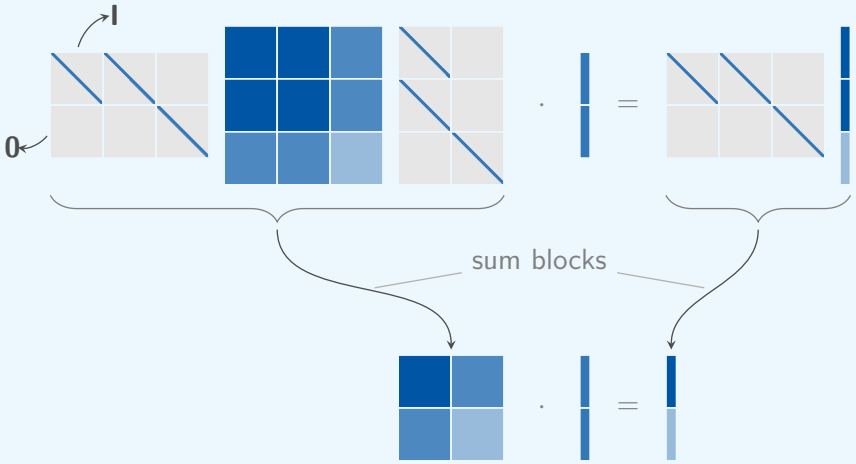
$$\mathbf{H} \begin{bmatrix} \tilde{\mathbf{p}}_{\mathcal{I}} \\ \mathbf{0} \end{bmatrix} = - \begin{bmatrix} \mathbf{g}_{\mathcal{I}} \\ \mathbf{0} \end{bmatrix}$$
$$\tilde{\mathbf{p}}_{\mathcal{A}} = -\mathbf{g}_{\mathcal{A}}.$$

and

$$p_i = \begin{cases} l_i - z_i & \text{if } z_i + \tilde{p}_i \leq l_i \\ u_i - z_i & \text{if } z_i + \tilde{p}_i \geq u_i \\ \tilde{p}_i & \text{otherwise} \end{cases}$$

Symmetry constraints

$$\min_{\mathbf{z}} \frac{1}{2} ||[\![\mathbf{A}, \mathbf{A}, \mathbf{C}]\!] - \mathcal{T}||_{\mathrm{F}}^2$$



Coupling constraints

Two tensors $\mathcal{T}_1$ and $\mathcal{T}_2$ can be factorized jointly using

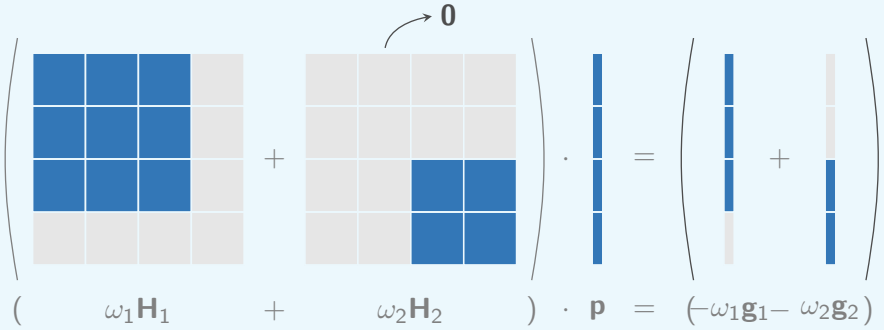
$$\min_{\mathbf{z}} \frac{\omega_1}{2} ||[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] - \mathcal{T}_1||_{\mathrm{F}}^2 + \frac{\omega_2}{2} ||[\![\mathbf{D}, \mathbf{E}, \mathbf{F}]\!] - \mathcal{T}_2||_{\mathrm{F}}^2$$

The factorizations can be coupled by

- coupling factor matrices, e.g., $\mathbf{A} = \mathbf{D}$
- partial coupling, e.g., $\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \end{bmatrix}$ and $\mathbf{D} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{d}_3 \end{bmatrix}$
- coupling through variables $\mathbf{A} = h_1(\boldsymbol{\alpha})$ and $\mathbf{D} = h_2(\boldsymbol{\alpha})$

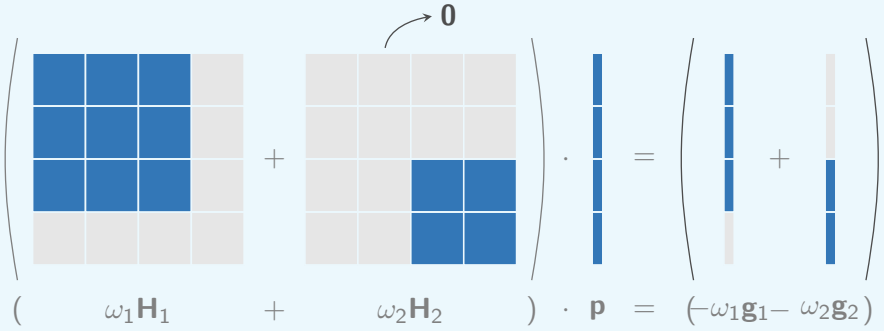
Coupling constraints: coupled tensor matrix factorization example

$$\min_{\mathbf{z}} \frac{\omega_1}{2} ||[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] - \mathcal{T}||_{\mathrm{F}}^2 + \frac{\omega_2}{2} ||\mathbf{C}\mathbf{D}^{\mathsf{T}} - \mathbf{M}||_{\mathrm{F}}^2$$



Coupling constraints: coupled tensor matrix factorization example

$$\min_{\mathbf{z}} \frac{\omega_1}{2} ||[\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] - \mathcal{T}||_{\mathrm{F}}^2 + \frac{\omega_2}{2} ||\mathbf{C}\mathbf{D}^{\mathsf{T}} - \mathbf{M}||_{\mathrm{F}}^2$$



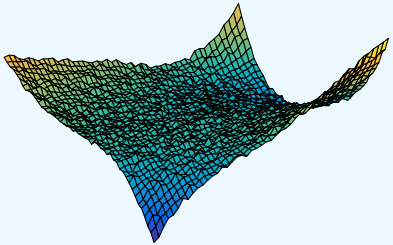
Soft coupling constraints can be handled similarly.

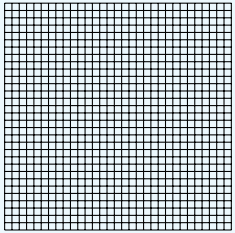
Multigrid sampling and coupled decompositions

Approximate a function $h(x, y)$ by

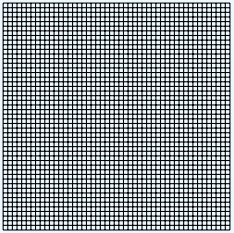
$$\tilde{h}(x, y) = \sum_{r=1}^{3} a_r(x) b_r(y)$$

using two noisy measurements $\mathbf{H}_1$ and $\mathbf{H}_2$ on two different grids





$\mathbf{H}_1$



$\mathbf{H}_2$

Multigrid sampling and coupled decompositions

Perform coupled decomposition of $\mathbf{H}_1$ and $\mathbf{H}_2$ with additional polynomial constraints:

$$\mathbf{H}_1 \approx \mathbf{A}\mathbf{B}^\mathsf{T} \qquad \mathbf{H}_2 \approx \mathbf{C}\mathbf{D}^\mathsf{T}$$

with

$$\mathbf{A} = \mathbf{M}_1\mathbf{Q}_1 \qquad \mathbf{B} = \mathbf{M}_1\mathbf{Q}_2$$
$$\mathbf{C} = \mathbf{M}_2\mathbf{Q}_1 \qquad \mathbf{D} = \mathbf{M}_2\mathbf{Q}_2$$

by solving the optimization problem

$$\min_{\mathbf{Q}_1,\mathbf{Q}_2} \frac{\omega_1}{2\cdot\Omega}\left|\left|\mathbf{H}_1 - \mathbf{A}\mathbf{B}^\mathsf{T}\right|\right|_\mathsf{F}^2 + \frac{\omega_2}{2\cdot\Omega}\left|\left|\mathbf{H}_2 - \mathbf{C}\mathbf{D}^\mathsf{T}\right|\right|_\mathsf{F}^2$$
$$\text{subject to } \mathbf{A} = \mathbf{M}_1\mathbf{Q}_1, \quad \mathbf{B} = \mathbf{M}_1\mathbf{Q}_2, \quad \mathbf{C} = \mathbf{M}_2\mathbf{Q}_1, \quad \mathbf{D} = \mathbf{M}_2\mathbf{Q}_2$$

$\mathbf{M}_1$ and $\mathbf{M}_2$ are (known) evaluated basis functions. Ω is a normalization factor.

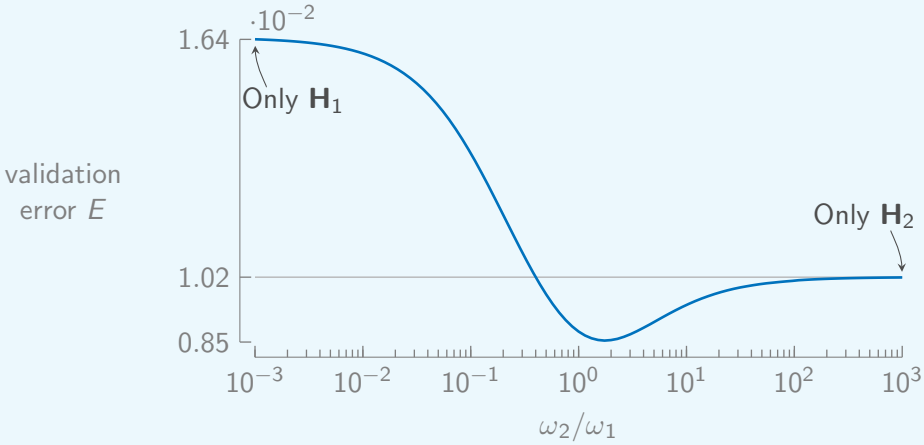
Multigrid sampling and coupled decompositions

Implementation using Tensorlab 4.0 syntax

```
model.variables.Q1 = rand(5,3);
model.variables.Q2 = rand(5,3);
model.factors.A    = {'Q1', struct_matvec(M1)};
model.factors.B    = {'Q2', struct_matvec(M1)};
model.factors.C    = {'Q1', struct_matvec(M2)};
model.factors.D    = {'Q2', struct_matvec(M2)};
model.factorizations.H1.data   = H1;
model.factorizations.H1.cpd    = {'A', 'B'};
model.factorizations.H1.weight = omega1;
model.factorizations.H2.data   = H2;
model.factorizations.H2.cpd    = {'C', 'D'};
model.factorizations.H2.weight = omega2;

sol = sdf_nls(model, 'CGMaxIter', 50);
```

Multigrid sampling and coupled decompositions



Conclusion

Second-order algorithms such as (inexact) Gauss–Newton are well suited for coupled and constrained tensor decomposition thanks to

- favorable convergence properties, and
- the possibility to exploit multilinear structure.

Conclusion

Second-order algorithms such as (inexact) Gauss–Newton are well suited for coupled and constrained tensor decomposition thanks to

- favorable convergence properties, and
- the possibility to exploit multilinear structure.

For more information, software and tutorials, see

- book chapter on "Numerical optimization-based algorithms for data fusion"
- Tensorlab (www.tensorlab.net)
- user guide (www.tensorlab.net/doc)
- demos (www.tensorlab.net/demos)

Numerical optimization-based tensor algorithms

Nico Vervliet

Lieven De Lathauwer

EURASIP Summer School
August 29, 2018

Bibliography

Overview of techniques

- N. Vervliet and L. De Lathauwer, "Numerical optimization based algorithms for data fusion", in *Data Fusion Methodology and Applications*, M. Cocchi, Ed., Accepted for publication., Elsevier, 2018

General tensor techniques

- A. Cichocki, D. Mandic, *et al.*, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis", *IEEE Signal Process. Mag.*, vol. 32, no. 2, pp. 145–163, Mar. 2015
- N. D. Sidiropoulos, L. De Lathauwer, *et al.*, "Tensor decomposition for signal processing and machine learning", *IEEE Trans. Signal Process.*, vol. 65, no. 13, pp. 3551–3582, Jul. 2017

Software

- N. Vervliet, O. Debals, *et al.*, *Tensorlab 3.0*, Available online at `https://www.tensorlab.net`, Mar. 2016

Bibliography

General optimization

- J. Nocedal and S. J. Wright, *Numerical Optimization*, Second edition. New York: Springer, 2006
- L. Sorber, M. Van Barel, and L. De Lathauwer, "Unconstrained optimization of real functions in complex variables", *SIAM J. Optim.*, vol. 22, no. 3, pp. 879–898, Jan. 2012

Underlying techniques

- L. Sorber, M. Van Barel, and L. De Lathauwer, "Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-$(L_r, L_r, 1)$ terms, and a new generalization", *SIAM J. Optim.*, vol. 23, no. 2, pp. 695–720, Apr. 2013
- L. Sorber, M. Van Barel, and L. De Lathauwer, "Structured data fusion", *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 4, pp. 586–600, Jun. 2015