Krylov-Levenberg-Marquardt Algorithm for Structured Tucker Tensor Decompositions

Petr Tichavský
The Czech Academy of Sciences
Institute of Information Theory and Automation
Prague, Czech Republic

Anh-Huy Phan, and Andrzej Cichocki

ICASSP 2022, Singapore

Structured Tucker Decomposition

The Tucker decomposition of the tensor with multilinear rank (R_1, R_2, R_3) will be denoted as a quartet $[[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]]$ with a core tensor $\mathcal{K}$ of the size $R_1 \times R_2 \times R_3$ and factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ of the sizes $I_i \times R_i$, $i = 1, 2, 3$, respectively, such that

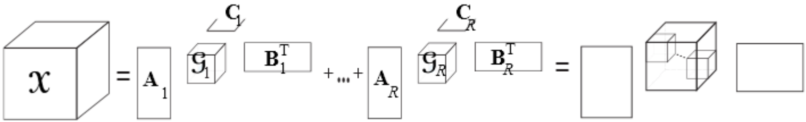
$$T_{ijk} \approx \sum_{p=1}^{R_1} \sum_{q=1}^{R_2} \sum_{r=1}^{R_3} K_{pqr} A_{ip} B_{jq} C_{kr} \tag{1}$$

Symbolically, we shall write

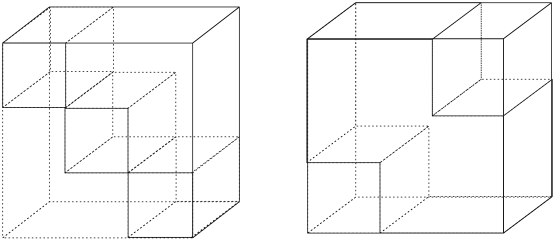
$$\mathcal{T} \approx [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]] \ . \tag{2}$$

Special cases: (1) CPD, (2) tensor chain (ring), (3) BTD

Examples: (1) Block term decomposition



(2) Block term decomposition with overlapping blocks



The task is to minimize the criterion

$$\varphi(\boldsymbol{\theta}) = \|\mathcal{T} - [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]]\|_F^2 \tag{3}$$

or (in the incomplete tensor case)

$$\varphi_W(\boldsymbol{\theta}) = \|\mathcal{W}^{1/2} \star (\mathcal{T} - [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]])\|_F^2 \tag{4}$$

with respect to components (vectors) of

$$\boldsymbol{\theta} = [\mathcal{K}(\mathcal{L}); \mathrm{vec}\,\mathbf{A}; \mathrm{vec}\,\mathbf{B}; \mathrm{vec}\,\mathbf{C}] \tag{5}$$

The weighting option allows to handle incomplete tensors and facilitate a tensor imputation.

Krylov-Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm consists in a sequence of iterations

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}' = \boldsymbol{\theta} - (\mathbf{H} + \mu \mathbf{I})^{-1}\mathbf{g}$$

where an error gradient, $\mathbf{g}$, and an approximate Hessian $\mathbf{H}$ are defined through a Jacobian matrix, $\mathbf{J}$, as

$$\mathbf{J} = \frac{\partial \mathrm{vec}(\mathcal{T})}{\partial \boldsymbol{\theta}} \quad (6)$$

$$\mathbf{g} = \mathbf{J}^T \mathbf{W} \mathrm{vec}(\mathcal{T} - \hat{\mathcal{T}}) \quad (7)$$

$$\mathbf{H} = \mathbf{J}^T \mathbf{W} \mathbf{J} \quad (8)$$

$\mathbf{W} = \mathrm{diag}(\mathrm{vec}(\mathcal{W}))$, applies in case of a weighted decomposition
μ is a damping parameter that is updated through the iterations.

Krylov-Levenberg-Marquardt Algorithm cont'd

- Nearly quadratic convergence
- Computationally prohibitive for large problems

Bottleneck:

- Computation of the Hessian matrix **H** (can be large in size) $O(N^2)$
- Inversion of **H**, or computation of $(\mathbf{H} + \mu\mathbf{I})^{1/2}$, complexity $O(N^3)$

In the KLM, the bottleneck is solved !

Krylov Subspace Approximation

The expression $(\mathbf{H}+\mu\mathbf{I})^{-1}\mathbf{g}$ is replaced by the approximation

$$(\mathbf{H}+\mu\mathbf{I})^{-1}\mathbf{g} \approx \frac{1}{\mu}\mathbf{g} - \frac{1}{\mu}\mathbf{U}(\mu\mathbf{Q}^{-1}+\mathbf{U}^T\mathbf{U})^{-1}(\mathbf{U}^T\mathbf{g}) \qquad (9)$$

where columns of matrix $\mathbf{U}$ form an orthogonal basis of the linear hull of

$$[\mathbf{g}, \mathbf{H}\mathbf{g}, \mathbf{H}^2\mathbf{g}, \ldots, \mathbf{H}^{M-1}\mathbf{g}] \qquad (10)$$

and

$$\mathbf{Q} = \mathbf{U}^T\mathbf{H}\mathbf{U}\ . \qquad (11)$$

$\mathbf{U}$ and $\mathbf{Q}$ can be found by a Gram-Schmidt orthogonalization procedure, similar to *Lanczos* algorithm or *Arnoldi* iteration.

Fast computing of $\mathbf{y} = \mathbf{Hx}$: Let

$$\boldsymbol{\theta} = [\mathrm{vec}(\mathcal{K}); \mathrm{vec}(\mathbf{A}); \mathrm{vec}(\mathbf{B}); \mathrm{vec}(\mathbf{C})] \ . \tag{12}$$

The Jacobian matrix has now four parts,

$$\mathbf{J} = \frac{\partial \mathrm{vec}(\mathcal{T})}{\partial \boldsymbol{\theta}} = [\mathbf{J}_K, \mathbf{J}_A, \mathbf{J}_B, \mathbf{J}_C] \ . \tag{13}$$

We need to deal with products of the type $\mathbf{Jx}$, and therefore, we write the arbitrary vector $\mathbf{x}$ as a concatenation of four parts,

$$\mathbf{x} = [\mathrm{vec}\,\boldsymbol{\mathcal{X}}_K; \mathrm{vec}\,\mathbf{X}_A; \mathrm{vec}\,\mathbf{X}_B; \mathrm{vec}\,\mathbf{X}_C] \tag{14}$$

where $\boldsymbol{\mathcal{X}}_K$ is a tensor of the shape of $\mathcal{K}$, and $\mathbf{X}_A, \mathbf{X}_B$, and $\mathbf{X}_C$ are matrices of the sizes of $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$, respectively.
Similarly, $\mathbf{y} = \mathbf{Hx}$ would have four parts as well,

$$\mathbf{y} = [\mathrm{vec}\,\boldsymbol{\mathcal{Y}}_K; \mathrm{vec}\,\mathbf{Y}_A; \mathrm{vec}\,\mathbf{Y}_B; \mathrm{vec}\,\mathbf{Y}_C] \ . \tag{15}$$

For the unweighted case,

$$\mathcal{Y}_K = [[\mathcal{Z}, \mathbf{A}^T, \mathbf{B}^T, \mathbf{C}^T]] \qquad \mathcal{Y}_A = \mathbf{Z}_{(1)}[[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{B}, \mathbf{C}]]_{(1)}^T$$

$$\mathcal{Y}_B = \mathbf{Z}_{(2)}[[\mathcal{K}, \mathbf{A}, \mathbf{I}_{R_2}, \mathbf{C}]]_{(2)}^T \qquad \mathcal{Y}_C = \mathbf{Z}_{(3)}[[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{I}_{R_3}]]_{(3)}^T$$

where

$$\begin{aligned} \mathcal{Z} &= [[\mathcal{X}_K, \mathbf{A}, \mathbf{B}, \mathbf{C}]] + [[\mathcal{K}, \mathbf{X}_A, \mathbf{B}, \mathbf{C}]] \\ &+ [[\mathcal{K}, \mathbf{A}, \mathbf{X}_B, \mathbf{C}]] + [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{X}_C]] \, . \end{aligned} \tag{16}$$

The error gradient is $\mathbf{g} = [\mathbf{g}_K; \mathbf{g}_A; \mathbf{g}_B; \mathbf{g}_C]$

$$\mathbf{g}_K = \text{vec}\,[[\mathcal{E}, \mathbf{A}^T, \mathbf{B}^T, \mathbf{C}^T]] \qquad \mathbf{g}_A = \text{vec}\,\{\mathbf{E}_{(1)}[[\mathcal{K}, \mathbf{I}_{R_1}, \mathbf{B}, \mathbf{C}]]_{(1)}^T\}$$

$$\mathbf{g}_B = \text{vec}\,\{\mathbf{E}_{(2)}[[\mathcal{K}, \mathbf{A}, \mathbf{I}_{R_2}, \mathbf{C}]]_{(2)}^T\} \quad \mathbf{g}_C = \text{vec}\,\{\mathbf{E}_{(3)}[[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{I}_{R_3}]]_{(3)}^T\} \, .$$

$$\mathcal{E} = \mathcal{T} - [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]] \, .$$

Extensions

- Weighted decompositions
- Linear transformation of the estimated parameters (e.g. symmetric or partially symmetric decompositions such as, e.g. $\mathbf{B} = \mathbf{C}$
- nonnegativity constraints
- constrains on *sensitivity*

$$s(\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}) = \lim_{\sigma^2 \to 0} \frac{1}{\sigma^2} \mathrm{E}\{\|[[\mathcal{K} + \delta\mathcal{K}, \mathbf{A} + \delta\mathbf{A}, \mathbf{B} + \delta\mathbf{B}, \mathbf{C} + \delta\mathbf{C}]] - [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]]\|_F^2\} \ , \qquad (17)$$

where $\delta\mathcal{K}, \delta\mathbf{A}, \delta\mathbf{B}$ and $\delta\mathbf{C}$ are random Gaussian-distributed perturbations of the core tensor and the factor matrices with i.i.d elements $\mathcal{N}(0, \sigma^2)$.

Example (BTD)

- Block-diagonal core tensor $\mathcal{K}$ of the size $15 \times 15 \times 15$ with three blocks of the size $5 \times 5 \times 5$ on its main diagonal, at random, having i.i.d. $\mathcal{N}(0, 1)$ distribution.
- The factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ have the size 12×15 and the tensor $\mathcal{T} = [[\mathcal{K}, \mathbf{A}, \mathbf{B}, \mathbf{C}]]$ has the size $12 \times 12 \times 12$. It means that $\mathcal{T}$ having $12^3 = 1728$ elements is smaller in size than the core tensor. The number of the model parameters is $3 \times 5^3 + 3 \times 12 \times 15 = 915$.
- No additive noise.

We compare the performance of three decomposition algorithms: (1) KLM with $M = 30$, (2) KLM with bounded sensitivity and $M = 30$, and (3) the NLS algorithm of Tensorlab.

	KLM	KLM-BND	NLS
ERROR	6.2	3.9	6.5
RATIO OF SUCCESS. RUNS	12%	39%	12%
TIME [s]	22.2	43.9	238.6
SENSITIVITY	$3.56 \cdot 10^7$	$0.25 \cdot 10^7$	$3.49 \cdot 10^7$

Table 1. Median fitting error per tensor element, ratio of successful runs, time of execution, and median sensitivity of output.

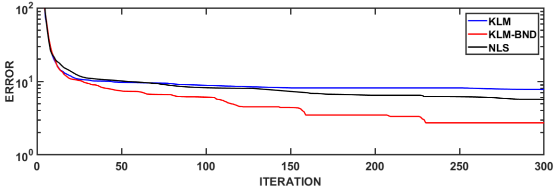


Fig. 1. Medians of the learning curves.

Conclusions

- We presented novel algorithms for structured or constrained Tucker tensor decomposition
- In the paper, we presented an application in block term decomposition, tensor chain modeling, classification of handwritten digits, and the compression of convolutional layers in neural networks.
- The KLM algorithm allows to seek decompositions with limited sensitivity.
- Matlab codes are available on the Internet at https://github.com/Tichavsky/tensor-decomposition