Supplementary Material

A flexible optimization framework for regularized matrix-tensor factorizations with linear couplings

Carla Schenker, Jeremy E. Cohen, Evrim Acar

June 30, 2020

1 Update equations for consensus variable $\boldsymbol{\Delta}_1$

Here, we give the explicit closed-form updates for the consensus variable $\boldsymbol{\Delta}_1$ for the five different cases of linear couplings described in Section IV-A.

Table 5: Update equations for $\boldsymbol{\Delta}_1$ for different coupling structures

coupling type	update for $\boldsymbol{\Delta}_1$
1	$\boldsymbol{\Delta}_1^{(k+1)} = \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{C}_{i,1}^{(k+1)} + {\boldsymbol{\mu}_{i,1}}_{(\Delta)}^{(k)} \right)$
2a	$\boldsymbol{\Delta}_1^{(k+1)} = \frac{1}{N} \sum_{i=1}^{N} \left(\tilde{\mathbf{H}}_{i,1} \mathbf{C}_{i,1}^{(k+1)} + {\boldsymbol{\mu}_{i,1}}_{(\Delta)}^{(k)} \right)$
2b	$\boldsymbol{\Delta}_1^{(k+1)} = \left[\sum_{i=1}^{N} \left(\tilde{\mathbf{H}}_{i,1}^{\Delta^T} \tilde{\mathbf{H}}_{i,1}^{\Delta} \right) \right]^{-1} \left[\sum_{i=1}^{N} \tilde{\mathbf{H}}_{i,1}^{\Delta^T} \left(\mathbf{C}_{i,1}^{(k+1)} + {\boldsymbol{\mu}_{i,1}}_{(\Delta)}^{(k)} \right) \right]$
3a	$\boldsymbol{\Delta}_1^{(k+1)} = \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{C}_{i,1}^{(k+1)} \hat{\mathbf{H}}_{i,1} + {\boldsymbol{\mu}_{i,1}}_{(\Delta)}^{(k)} \right)$
3b	$\boldsymbol{\Delta}_1^{(k+1)} = \left[\sum_{i=1}^{N} \left(\mathbf{C}_{i,1}^{(k+1)} + {\boldsymbol{\mu}_{i,1}}_{(\Delta)}^{(k)} \right) \hat{\mathbf{H}}_{i,1}^{\Delta^T} \right] \left[\sum_{i=1}^{N} \left(\hat{\mathbf{H}}_{i,1}^{\Delta} \hat{\mathbf{H}}_{i,1}^{\Delta^T} \right) \right]^{-1}$

2 Restrictions on transformation matrices

In order for the coupled model to make sense, the transformation matrices $\mathbf{H}_{i,1}$ and $\mathbf{H}_{i,1}^{\Delta}$ should fulfil the following two properties, which will also ensure that the algorithm works as intended:

1. Given all coupled factor matrices $\{\mathbf{C}_{i,1}\}_{i \leq N}$ and transformation matrices, the consensus variable $\boldsymbol{\delta}_1$ should be uniquely defined via

$$\boldsymbol{\delta}_1 = \underset{\mathbf{z}}{\operatorname{argmin}} \sum_{i=1}^{N} \left\| \mathbf{H}_{i,1} \operatorname{vec}(\mathbf{C}_{i,1}) - \mathbf{H}_{i,1}^{\Delta} \mathbf{z} + {\boldsymbol{\mu}_{i,1}}_{(\delta)} \right\|_2^2 .$$

2. For any given $\boldsymbol{\Delta}_1$, there should exist at least one solution $\mathbf{C}_{i,1}$ to the equation

$$\mathbf{H}_{i,1} \operatorname{vec}(\mathbf{C}_{i,1}) = \mathbf{H}_{i,1}^{\Delta} \operatorname{vec}(\boldsymbol{\Delta}_1)$$

for each coupled tensor i.

This leads to some restrictions on the transformation matrices for the cases described in Section IV-A:

Case 2a: Here, the first property is always fulfilled. To ensure the second property, we require all transformation matrices $\tilde{\mathbf{H}}_{i,1}$ to be right-invertible, since we want the corresponding linear map to be surjective. It follows that $n_\Delta \leq \min_i n_{1_i}$, since all $\tilde{\mathbf{H}}_{i,1}$ are supposed to have linearly independent rows. Furthermore, we note that the solution $\mathbf{X}$ of the Sylvester equation $\mathbf{AX} + \mathbf{XB} = \mathbf{C}$ is unique if and only if $\mathbf{A}$ and $-\mathbf{B}$ have distinct spectra, which will almost always be the case for the matrices in this algorithm (equation (11)).

Case 2b: To ensure that $\boldsymbol{\Delta}_1$ is uniquely defined by the union of all coupled factor matrices $\{\mathbf{C}_{i,1}\}$, we require the matrix

$$\left[\tilde{\mathbf{H}}_{1,1}^{\Delta^T} | \tilde{\mathbf{H}}_{2,1}^{\Delta^T} | ... | \tilde{\mathbf{H}}_{N,1}^{\Delta^T}\right]$$

of size $n_{\Delta_1} \times \sum_i n_{1_i}$ to have rank n_{Δ_1}. This is sufficient for the invertibility of the matrix $\sum_i^N \left(\tilde{\mathbf{H}}_{i,1}^{\Delta^T} \tilde{\mathbf{H}}_{i,1}^{\Delta}\right)$, which is needed for the update of $\boldsymbol{\Delta}_1$ given in table 5.

Case 3a: Analogously to Case 2a, here we require all transformation matrices $\hat{\mathbf{H}}_{i,1}$ to be left-invertible. This implies the size restriction $R_{\Delta_1} \leq \min_i R_i$.

Case 3b: Here, similar to Case 2b, in order to ensure the uniqueness of $\boldsymbol{\Delta}_1$, we require the matrix

$$\left[\hat{\mathbf{H}}_{1,1}^{\Delta} | \hat{\mathbf{H}}_{2,1}^{\Delta} | ... | \hat{\mathbf{H}}_{N,1}^{\Delta}\right]$$

of size $R_{\Delta_1} \times \sum_i^N R_i$ to have rank R_{Δ_1}.

3 Loss function gradient

The derivative of $\mathcal{L}(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{X}})$, where $\boldsymbol{\mathcal{X}} = [\![\mathbf{C}_{i,1}, \mathbf{C}_{i,2}, ..., \mathbf{C}_{i,D_i}]\!]$, with respect to the factor matrix $\mathbf{C}_{i,d}$ is given by

$$\frac{\partial \mathcal{L}(\boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{X}})}{\partial \mathbf{C}_{i,d}} = \boldsymbol{\mathcal{Y}}_{[d]} \mathbf{M}_{i,d},$$

where $\boldsymbol{\mathcal{Y}}_{[d]}$ is the mode d unfolding of the tensor $\boldsymbol{\mathcal{Y}}$ defined by

$$y_j = \frac{\partial \ell(t_j, x_j)}{\partial x_j},$$

where ℓ is the element-wise loss function , see [1] for a detailed derivation of this. Thus, one can take advantage of efficient implementations of the matricized tensor times Khatri-Rao product for the computation of the gradient. To obtain the complete gradient of the objective function in line 3 of Algorithm 2, the gradient of the terms related to coupling and constraints have to be added, which is straightforward, but changes slightly with the different linear coupling options and is omitted here.

4 Efficient implementations

In all types of linear couplings described in Section IV-A, with the exception of Case 2a, the update of the factor matrix $\mathbf{C}_{i,1}$ reduces to the solution of a linear system where the matrix inverse is only of size $R_i \times R_i$. Note that this matrix inverse is never explicitly computed. Instead, a Cholesky decomposition which costs $O(R_i^3)$ is precomputed outside the ADMM loop. Thus, solving the linear systems at each ADMM iteration reduces to one forward- and one backward-substitution with complexity $O(R_i^2 n_{1_i})$. Also for the update of $\boldsymbol{\Delta}_1$, a Cholesky decomposition of the matrices $\sum_i^N \left(\tilde{\mathbf{H}}_{i,1}^{\Delta^T} \tilde{\mathbf{H}}_{i,1}^{\Delta}\right)$ and $\left[\sum_{i=1}^N \left(\hat{\mathbf{H}}_{i,1}^{\Delta} \hat{\mathbf{H}}_{i,1}^{\Delta^T}\right)\right]$ in Cases 2b and 3b respectively, can

be precomputed at the beginning of the AO-ADMM algorithm. In Case 2b, this can be expensive, with the Cholesky decomposition having a complexity of $O(n_{\Delta_1}^3)$ and each forward and backward-substitution $O(n_{\Delta_1}^2 R_i)$, in contrast to $O(R_\Delta^3)$ and $O(R_\Delta^2 n_1)$ for Case 3b. In Cases 1, 2a and 3a, the update of $\mathbf{\Delta}_1$ is computed as an average with complexity $O(Nn_1R)$, $O(Nn_{\Delta_1}n_{1_i}R)$ and $O(Nn_1R_iR_\Delta)$, respectively.

Furthermore, the matricized tensor times Khatri-Rao product $\boldsymbol{\mathcal{T}}_{i[1]}\mathbf{M}_{i,1}$, can be computed efficiently [2], for which we use the `mttkrp` function from the Tensor Toolbox [3]. It can be precomputed outside the ADMM loop. Also the product of Khatri-Rao products $\mathbf{M}_{i,1}^T\mathbf{M}_{i,1}$ can be computed efficiently using the relation [2]

$$\mathbf{M}_{i,1}^T\mathbf{M}_{i,1} = \mathbf{C}_{i,2}^T\mathbf{C}_{i,2} * ... * \mathbf{C}_{i,D_i}^T\mathbf{C}_{i,D_i}$$

with precomputed products $\{\mathbf{C}_{i,d}^T\mathbf{C}_{i,d}\}_{i\leq N, d\leq D_i}$. These products can be stored throughout the whole AO-ADMM algorithm and need only to be updated for each mode after the corresponding outer AO iteration.

The evaluation of the residual $f_{\text{tensors}}^{(k)}$ for the stopping condition can be computationally expensive. It is, therefore, desirable to reuse as many previous computations as possible. For tensors $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{\mathcal{M}}$, the difference in squared Frobenius norm can be calculated as

$$\| \boldsymbol{\mathcal{T}} - \boldsymbol{\mathcal{M}} \|_F^2 = \| \boldsymbol{\mathcal{T}} \|_F^2 + \| \boldsymbol{\mathcal{M}} \|_F^2 - 2 \langle \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{M}} \rangle$$

Here, $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{T}}_i$ is the given tensor, which does not change, and it is enough to compute its norm only once. The same holds for other loss functions, *e.g.*, for KL-divergence, it is sufficient to compute the constant term $\sum_{j\in\mathcal{J}} [t_j \log t_j - t_j]$ only once. Furthermore, in the case of Frobenius norm, the following holds:

$$\| \boldsymbol{\mathcal{M}} \|_F^2 = \| [\![\mathbf{C}_{i,1}, \mathbf{C}_{i,2}, ..., \mathbf{C}_{i,D_i}]\!] \|_F^2 = \mathbf{e}^T \left[\mathbf{C}_{i,1}^T\mathbf{C}_{i,1} * \mathbf{C}_{i,2}^T\mathbf{C}_{i,2} * ... * \mathbf{C}_{i,D_i}^T\mathbf{C}_{i,D_i}\right] \mathbf{e},$$

where $\mathbf{e}$ is a column vector of 1s with matching length. This is useful since the products $\mathbf{C}_{i,d}^T\mathbf{C}_{i,d}$ can be reused. The tensor inner product $\langle \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{M}} \rangle$ can also be computed efficiently via

$$\langle \boldsymbol{\mathcal{T}}, \boldsymbol{\mathcal{M}} \rangle = \langle \boldsymbol{\mathcal{T}}_i, [|\mathbf{C}_{i,1}, \mathbf{C}_{i,2}, ..., \mathbf{C}_{i,D_i}|] \rangle = \mathbf{e}^T \left[\boldsymbol{\mathcal{T}}_{i[d]}\mathbf{M}_{i,d} * \mathbf{C}_{i,d}\right] \mathbf{e},$$

where again the matricized tensor times Khatri-Rao product $\boldsymbol{\mathcal{T}}_{i[d]}\mathbf{M}_{i,d}$ from the last updated mode d of tensor i can be reused.

5 Additional plots and experiments

Experiment 1a. This experiment is described in Section VI-B 1a. Figure 13 shows the distribution of computing times after reaching a relative tolerance of 10^{-6} difference in function value. After reaching this tolerance, FMS has typically converged for this problem.

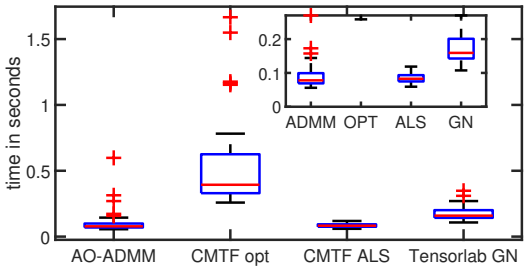


Figure 13: Experiment 1a with coupling Case 1 and congruence 0.5 in all factor matrices: Boxplots showing distribution of computing times of different algorithms after reaching a relative tolerance of 10^{-6}.

Experiment 1b. This experiment is described in Section VI-B 1b. Figure 14 shows the distribution of computing times after reaching a relative tolerance of 10^{-8} difference in function value. Since this problem is more difficult than the previous one, it typically takes longer to find the correct factors, therefore the tolerance is chosen as 10^{-8}.

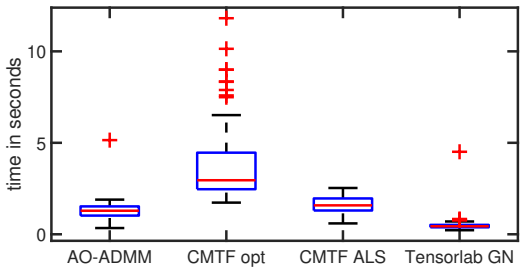
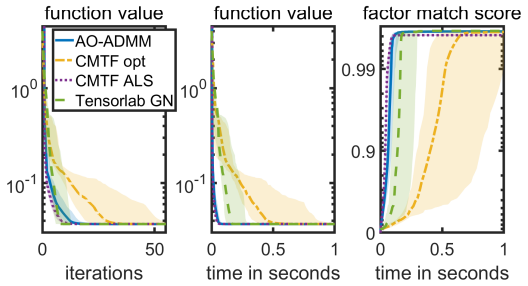
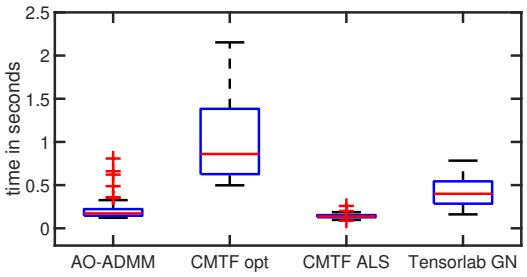


Figure 14: Experiment 1b with coupling Case 1 and congruence 0.9 in all factor matrices: Boxplots showing distribution of computing times of different algorithms after reaching a relative tolerance of 10^{-8}.

Experiment 1c. Unbalanced Data Size. In this additional experiment, we study the impact of unbalanced dimensions in the data sets. The setting corresponds to Experiment 1a as described in Section VI-B 1a. The tensor has now a size of $40 \times 500 \times 60$ and the matrix is of size 40×600. Factors have a congruence of 0.5. As shown in Figure 15, AO-ADMM performs as well as CMTF-ALS and all methods can find the true factors. For this case, we observe that Tensorlab GN is more sensitive to initializations compared to other methods, see Table 6.



(a) Median and quartiles of function values and FMS for different algorithms.



(b) Boxplots showing distribution of computing times of different algorithms after reaching a relative tolerance of 10^{-6}.

Figure 15: Experiment 1c with coupling Case 1 and congruence 0.5 and unbalanced sizes.

Experiment 2. Here, we study the effect of different number of maximum inner iterations m ($m = 3, 5, 10, 20$) in the AO-ADMM algorithm 2 in the setting of experiment 2 (Section VI-B 2) with non-negativity constraints. As shown in Figure 16, there is almost no difference in convergence speed and $m = 5$ seems to be a good choice.

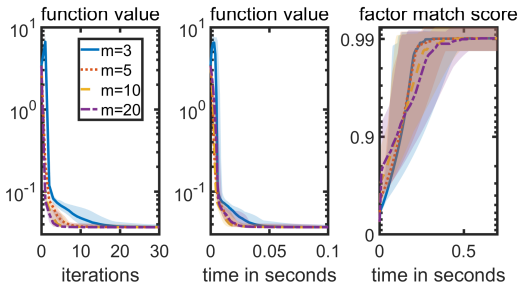


Figure 16: Experiment 2a: Median and minimum/maximum function values and FMS for AO-ADMM with different number of maximum inner iterations m.

Experiment 2b. Unbalanced Data Size. We study here the impact of unbalanced dimensions in data sets together with non-negativity constraints, in the setting of experiment 2 (Section VI-B 2). The tensor has a size of $40 \times 500 \times 60$ and the matrix is of size 40×600. Figure 17 shows that, the relative performance of the methods is identical.

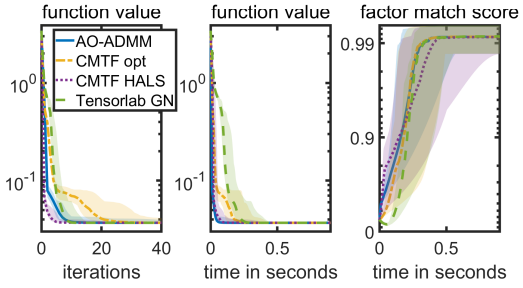


Figure 17: Experiment 2b with coupling Case 1, nonnegativity constraints and unbalanced sizes: Median and minimum/maximum function values and FMS for different algorithms.

Experiment 3. This experiment is described in Section VI-B 3. Figure 18 shows the distribution of computing times after reaching a relative tolerance of 10^{-8} difference in function value.

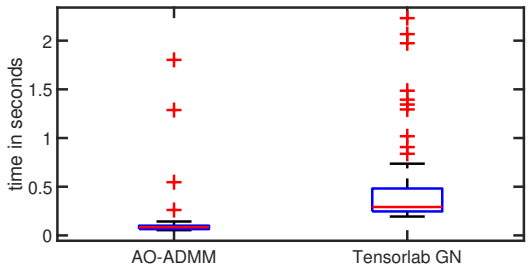


Figure 18: Experiment 3: Boxplots showing distribution of computing times of AO-ADMM and Tensorlab GN after reaching a rel. tol. of 10^{-8}.

Experiment 4. This experiment is described in Section VI-B 4. Figure 19 shows the distribution of computing times after reaching a relative tolerance of 10^{-8} difference in function value.

Table 6: Failed runs experiments 1c, 2b

Exp.	out of	AO-ADMM	CMTF opt	CMTF ALS	TL GN
1c	all/ best	43/0	60/0	60/0	173/0
2b	all/ best	0/0	0/0	0/0	0/0

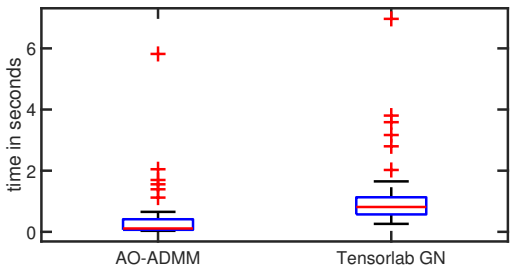


Figure 19: Experiment 4: Boxplots showing distribution of computing times of AO-ADMM and Tensorlab GN after reaching a rel. tol. of 10^{-8}.

Experiment 5. This experiment is described in Section VI-B 5. Figure 20 shows the distribution of computing times after reaching a relative tolerance of 10^{-6} difference in function value.

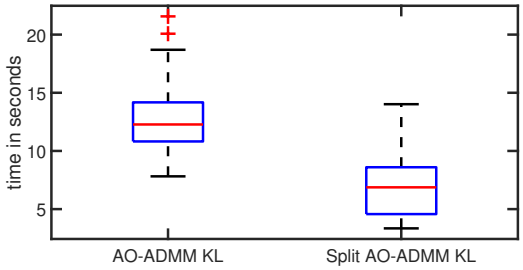


Figure 20: Experiment 5: Boxplots showing distribution of computing times of different algorithms after reaching a rel. tol. of 10^{-6}.

6 Extension to flexible couplings

So far, we have only assumed *hard* linear coupling structures, $\mathbf{H}_{i,1}\operatorname{vec}(\mathbf{C}_{i,1}) = \mathbf{H}_{i,1}^{\Delta}\operatorname{vec}(\mathbf{\Delta}_1)$, in the sense of exact equality. This is a very strict assumption and is oftentimes relaxed to allow for small variations between coupled factors from different datasets [4–6], referred to as *soft* or *flexible* coupling. Here, we derive a variant of our algorithmic framework which covers the case when the coupling relationship between the factors $\mathbf{C}_{i,1}$ is described by a joint density probability function, as discussed in [4]. For example, define the joint density of $\mathbf{C}_{i,1}$ as

$$p(\mathbf{C}_{1,1},\ldots,\mathbf{C}_{n,1}) = p(\mathbf{C}_{1,1},\ldots,\mathbf{C}_{N,1}|\mathbf{\Delta}_1)p(\mathbf{\Delta}_1) = p(\mathbf{\Delta}_1)\prod_{i=1}^{N} p(\mathbf{C}_{i,1}|\mathbf{\Delta}_1)$$

where $p(\mathbf{\Delta}_1)$ is a prior on the consensus variable $\mathbf{\Delta}_1$. Conditional independence of $\mathbf{C}_{i,1}$ is assumed. Then, using a MAP estimator, the optimization problem for mode 1 with flexible

couplings takes the following form:

$$\underset{\{\mathbf{C}_{i,1}\}_{i\leq N},\boldsymbol{\Delta}_1}{\operatorname{argmin}} \sum_{i=1}^{N} \left[w_i \mathcal{L}_i\left(\boldsymbol{\mathcal{T}}_i, [\![\mathbf{C}_{i,1}, \mathbf{C}_{i,2}, ..., \mathbf{C}_{i,D_i}]\!]\right) + g_{i,1}\left(\mathbf{C}_{i,1}\right) - \log(p(\mathbf{C}_{i,1}|\boldsymbol{\Delta}_1))\right] - \log(p(\boldsymbol{\Delta}_1)) \tag{17}$$

Note that problem (17) reduces to the following problem when $p(\boldsymbol{\Delta}_1)$ is flat, and $p(\mathbf{C}_{i,1}|\boldsymbol{\Delta}_1)$ is Gaussian with standard deviation ν_i:

$$\underset{\{\mathbf{C}_{i,1}\}_{i\leq N},\boldsymbol{\Delta}_1}{\operatorname{argmin}} \sum_{i=1}^{N} \left[w_i \mathcal{L}_i\left(\boldsymbol{\mathcal{T}}_i, [\![\mathbf{C}_{i,1}, \mathbf{C}_{i,2}, ..., \mathbf{C}_{i,D_i}]\!]\right) + g_{i,1}\left(\mathbf{C}_{i,1}\right) + \frac{1}{\nu_i^2} \parallel \mathbf{C}_{i,1} - \boldsymbol{\Delta}_1 \parallel_F^2\right]$$

We propose to use ADMM to solve problem (17) similar to before, but with a different splitting of variables. Variables $\mathbf{C}_{i,1}$ are split two times, once for handling the set constraint, and once for accounting for the coupling function. Additionally, hidden variable $\boldsymbol{\Delta}_1$ is split to account for its prior. This yields the following optimization problem:

$$\begin{aligned}
\underset{\{\mathbf{C}_{i,1},\mathbf{Z}_{i,1},\mathbf{B}_{i,1}\}_{i\leq N},\boldsymbol{\Delta}_1,\mathbf{Z}_{\boldsymbol{\Delta}_1}}{\operatorname{argmin}} \quad & \sum_{i=1}^{N} \left[w_i \mathcal{L}_i\left(\boldsymbol{\mathcal{T}}_i, [\![\mathbf{C}_{i,1}, \mathbf{C}_{i,2}, ..., \mathbf{C}_{i,D_i}]\!]\right) + g_{i,1}\left(\mathbf{Z}_{i,1}\right) + f_{i,1}\left(\mathbf{B}_{i,1}, \boldsymbol{\Delta}_1\right)\right] + h\left(\mathbf{Z}_{\boldsymbol{\Delta}_1}\right) \\
\text{s.t.} \quad & \mathbf{C}_{i,1} = \mathbf{B}_{i,1} \\
& \mathbf{C}_{i,1} = \mathbf{Z}_{i,1} \\
& \boldsymbol{\Delta}_1 = \mathbf{Z}_{\boldsymbol{\Delta}_1}
\end{aligned} \tag{18}$$

where $f_{i,1}(\cdot,\cdot) = -\log(p(\cdot|\cdot))$ and $h(\cdot) = -\log(p(\cdot))$ denote the negative log-likelihood of the coupling probability density function and the prior distribution on $\boldsymbol{\Delta}_1$, respectively. Applying ADMM to problem (18) leads to Algorithm 4 presented below. Also here, special forms of linear

Algorithm 4 ADMM for subproblem w.r.t. mode 1 of regularized CPD with flexible couplings

while convergence criterion is not met **do**
  for $i = 1, ..., N$ **do**

$$\mathbf{C}_{i,1}^{(k+1)} = \underset{\mathbf{X}}{\operatorname{argmin}}\ w_i \mathcal{L}_i\left(\boldsymbol{\mathcal{T}}_i, [\![\mathbf{X}, \mathbf{C}_{i,2}, ..., \mathbf{C}_{i,D_i}]\!]\right) + \frac{\rho}{2}\left(\left\| \mathbf{X} - \mathbf{Z}_{i,1}^{(k)} + \boldsymbol{\mu}_{i,1}{}_{(z)}^{(k)} \right\|_F^2 + \left\| \mathbf{X} - \mathbf{B}_{i,1}^{(k)} + \boldsymbol{\mu}_{i,1}{}_{(B)}^{(k)} \right\|_F^2\right)$$

  end for

$$\boldsymbol{\Delta}_1^{(k+1)} = \operatorname{prox}_{\frac{1}{\rho}\sum_{i=1}^{N} f_{i,1}(\mathbf{B}_{i,1}^{(k)},\cdot)}(\mathbf{Z}_{\Delta_1}^{(k)} - \mu_{\Delta_1}^{(k)})$$

  for $i = 1, ..., N$ **do**

$$\mathbf{Z}_{i,1}^{(k+1)} = \operatorname{prox}_{\frac{1}{\rho} g_{i,1}}\left(\mathbf{C}_{i,1}^{(k+1)} + \boldsymbol{\mu}_{i,1}{}_{(z)}^{(k)}\right)$$

$$\mathbf{B}_{i,1}^{(k+1)} = \operatorname{prox}_{\frac{1}{\rho} f_{i,1}(\cdot,\boldsymbol{\Delta}_1^{(k+1)})}\left(\mathbf{C}_{i,1}^{(k+1)} + \boldsymbol{\mu}_{i,1}{}_{(B)}^{(k)}\right)$$

$$\boldsymbol{\mu}_{i,1}{}_{(z)}^{(k+1)} = \boldsymbol{\mu}_{i,1}{}_{(z)}^{(k)} + \mathbf{C}_{i,1}^{(k+1)} - \mathbf{Z}_{i,1}^{(k+1)}$$

$$\boldsymbol{\mu}_{i,1}{}_{(B)}^{(k+1)} = \boldsymbol{\mu}_{i,1}{}_{(B)}^{(k)} + \mathbf{C}_{i,1}^{(k+1)} - \mathbf{B}_{i,1}^{(k+1)}$$

$$\boldsymbol{\mu}_{i,1}{}_{(\Delta)}^{(k+1)} = \boldsymbol{\mu}_{i,1}{}_{(\Delta)}^{(k)} + \boldsymbol{\Delta}_1^{(k+1)} - \mathbf{Z}_{\Delta_1}^{(k+1)}$$

  end for
  $k = k + 1$
end while

coupling transformations can be incorporated, for which we refer to [4].

References

[1] D. Hong, T. G. Kolda, and J. A. Duersch, "Generalized canonical polyadic tensor decomposition," *SIAM Review*, vol. 62, no. 1, pp. 133–163, 2020.

[2] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, Sep. 2009.

[3] B. W. Bader, T. G. Kolda *et al.*, "Matlab tensor toolbox version 3.1," Jun. 2019. [Online]. Available: https://www.tensortoolbox.org

[4] R. Cabral Farias, J. E. Cohen, and P. Comon, "Exploring multimodal data fusion through joint decompositions with flexible couplings," *IEEE Trans. Sig. Proc.*, vol. 64, no. 18, pp. 4830–4844, Sep. 2016.

[5] B. Rivet, M. Duda, A. Guérin-Dugué, C. Jutten, and P. Comon, "Multimodal approach to estimate the ocular movements during EEG recordings: a coupled tensor factorization method," in *EMBC'15*, 2015, pp. 6983–6986.

[6] C. Chatzichristos, M. Davies, J. Escudero, E. Kofidis, and S. Theodoridis, "Fusion of EEG and fMRI via soft coupled tensor decompositions," *EUSIPCO'18*, pp. 56–60, 2018.