Accelerating Probabilistic Tensor Canonical Polyadic Decomposition With Nonnegative Factors: An Inexact BCD Approach

Zhongtao Chen, Lei Cheng, and Yik-Chung Wu

Zhongtao Chen and Yik-Chung Wu are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong (e-mail:ztchen@eee.hku.hk; ycwu@eee.hku.hk).

Lei Cheng is with the ISEE College, Zhejiang University, Hangzhou, China (email: lei_cheng@zju.edu.cn).

Abstract

Recently, Bayesian modeling and variational inference (VI) were leveraged to enable the nonnegative factor matrix learning with automatic rank determination in tensor canonical polyadic decomposition (CPD), which has found various applications in big data analytics. However, since VI inherently performs block coordinate descent (BCD) steps over the functional space, it generally does not allow integration with modern large-scale optimization methods, making the scalability a critical issue. In this paper, it is revealed that the expectations of the variables updated by the VI algorithm is equivalent to the block minimization steps of a deterministic optimization problem. This equivalence further enables the adoption of inexact BCD method for devising a fast nonnegative factor matrix learning algorithm with automatic tensor rank determination. Numerical results using synthetic data and real-world applications show that the performance of the proposed algorithm is comparable with that of the VI-based algorithm, but with computation times reduced significantly.

Keywords: Tensor decomposition, nonnegative factors, automatic tensor rank determination

1. Introduction

The surging interest in tensor decompositions has been recently prompted by a proliferation of data-intensive applications [1, 2], which are awash in a sea of high-dimensional and multi-indexed datasets. In order to reveal the underlying informative patterns from these multi-indexed datasets, tensor decompositions have been proven to be very successful and found applications in various big data analytic tasks, ranging from biomedical data analysis [3, 4, 5] to data mining [6, 7, 8] and speech signal processing [9, 10, 11]. In many applications mentioned above, tensor canonical polyadic decomposition (CPD) with nonnegative factors [9] has played an important role, since its model parameters are with clear physical interpretations. In particular, each column in the decomposed nonnegative factor matrix represents one underlying latent component (e.g., a chemical species in biomedical samples [5] or a social group in social networks [6]); and the number of columns in each factor matrix (also known as tensor rank) specifies the number of hidden components.

Mathematically, in CPD with nonnegative factors, a set of nonnegative factor matrices $\{\boldsymbol{\Xi}^{(n)} \in \mathbb{R}^{J_n \times R}\}$ is sought from a N dimensional tensor $\mathcal{Y} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ via solving the following problem [9]:

$$\begin{aligned} \min_{\{\boldsymbol{\Xi}^{(n)}\}_{n=1}^{N}} \; & \| \mathcal{Y} - \underbrace{\sum_{r=1}^{R} \boldsymbol{\Xi}_{:,r}^{(1)} \circ \boldsymbol{\Xi}_{:,r}^{(2)} \circ \cdots \circ \boldsymbol{\Xi}_{:,r}^{(N)}}_{\triangleq [\![\boldsymbol{\Xi}^{(1)}, \boldsymbol{\Xi}^{(2)}, \cdots, \boldsymbol{\Xi}^{(N)}]\!]} \|_F^2 \\ \text{s.t.} \;\; & \boldsymbol{\Xi}^{(n)} \geq \mathbf{0}_{J_n \times R}, \quad n = 1, 2, ..., N, \end{aligned} \tag{1}$$

where symbol $\circ$ denotes vector outer product, shorthand notation $[\![\cdots]\!]$ is termed as the Kruskal operator, and $\| \cdot \|_F$ represents the Frobenius norm of the argument. In problem (1), it can be seen that the tensor data $\mathcal{Y}$ is decomposed into a summation of R rank-1 components $\{\boldsymbol{\Xi}_{:,r}^{(1)} \circ \boldsymbol{\Xi}_{:,r}^{(2)} \circ \cdots \circ \boldsymbol{\Xi}_{:,r}^{(N)}\}_{r=1}^{R}$, each determined by the r^{th} columns of all the factor matrices, i.e., $\{\boldsymbol{\Xi}_{:,r}^{(n)}\}_{n=1}^{N}$. Therefore, the set $\{\boldsymbol{\Xi}_{:,r}^{(n)}\}_{n=1}^{N}$ can be treated as the building block of the tensor CPD model, and the tensor rank R controls the model complexity (i.e., the number of building blocks) [12].

To solve problem (1), from the perspective of non-convex optimization, block coordinate descent (BCD) [13, 26, 27, 28] is the prevalent framework. In particular, problem (1) was found to be block multi-convex [13], in the

sense that the problem is convex with respect to each factor matrix after fixing other factor matrices. With each subproblem being solved with respect to a factor matrix, the utilization of block minimization gives the widely-used nonnegative alternating least-squares (NALS) method [9]. By exploring the block proximal update with Nesterov-based acceleration [14] or block prox-linear update with extrapolation [13], fast algorithms were developed. Furthermore, a hybrid scheme [15] combining alternating direction method of multipliers (ADMM) and alternating optimization (AO), and a framework [16] integrating randomized BCD and stochastic proximal gradient (SPG), were proposed to enable the nonnegative factor matrix learning with high computational speed and low memory requirement.

However, the above optimization algorithms require the knowledge of the tensor rank, which however is typically unknown in practice. For example, recent work [39] utilized the proximal update to tackle the newly introduced non-negative penalties and discussed the robustness against the inaccurate estimation of tensor ranks, yet did not estimate tensor rank explicitly. Although it is only a single number, its discrete nature makes its optimal determination a generally non-deterministic polynomial-time hard (NP-hard) problem [12].

To tackle this challenge, instead of acquiring the tensor rank estimate via trial-and-error, in which the algorithms under different tensor rank assumptions are run until the best data interpretation is achieved, a recent approach [17] recasts the tensor rank learning as a hyper-parameter inference step in the framework of Bayesian modeling and inference [18, 19, 31, 32]. More specifically, it adopts the sparsity-promoting nonnegative Gaussian-gamma prior distribution to encourage the sparseness for the columns of the non-negative factors. With variational inference (VI) algorithm [30], most of the columns of the nonnegative factor matrices will be driven to zero. Therefore, at convergence, the remaining number of non-zero columns in each factor matrix gives the estimate of tensor rank.

While the VI algorithm in [17] follows the same framework as that in other Bayesian CPD [20, 21, 22, 23, 24], due to the unique non-negative constraint on the factor matrices, there is an update step in the VI algorithm of [17] that does not admit closed-form expression, thus introducing a computational bottleneck to the algorithm. To speed up the VI-based algorithm in [17], it is natural to consider the advances from optimization community. However, VI inherently operates in functional space of probability distributions, rather than Euclidian space, in which most optimization theories and techniques are

developed [26, 27, 28]. Even though VI update steps could be interpreted as the natural gradient descent steps in the Riemannian space under certain conditions [33], it is still far from most large-scale optimization studies.

Fortunately, in probabilistic tensor CPD with nonnegative factors, this link indeed exists. More specifically, in this paper, it is shown that an equivalent optimization in the Euclidian space can be established that accurately mimics the expectation updates of the VI algorithm. Based on this newly found equivalence, a fast algorithm for the probabilistic tensor CPD with nonnegative factors is further developed under the framework of inexact BCD [13, 26, 27]. Different from existing optimization methods [13, 26, 27, 39], the proposed method is deduced from the analysis of the VI-based algorithm, and thus includes automatic tensor rank learning as an integrated feature. Numerical results on synthetic data and real-world datasets demonstrate the excellent performance of the proposed algorithm in terms of tensor rank learning, tensor recovery and computational time.

The remainder of this paper is organized as follows. Section 2 briefly reviews the probabilistic CPD model with nonnegative factors and the corresponding VI-based algorithm. In Section 3, an equivalent optimization problem in the Euclidian space is constructed, based on which a fast algorithm is derived in section 4. Numerical results with synthetic data and real-world datasets are reported in Section 5 and 6. Finally, conclusions are drawn in Section 7.

Notation: Boldface lowercase and uppercase letters will be used for vectors and matrices, respectively. Tensors are written as calligraphic letters. $\mathbb{E}[\,\cdot\,]$ denotes the expectation of its argument. Superscript T denotes transpose, and the operator $\mathrm{Tr}\,(\boldsymbol{A})$ denotes the trace of a matrix $\boldsymbol{A}$. $\mathcal{N}(\boldsymbol{x}|\boldsymbol{u},\boldsymbol{R})$ stands for the probability density function of a Gaussian vector $\boldsymbol{x}$ with mean $\boldsymbol{u}$ and covariance matrix $\boldsymbol{R}$. The $N \times N$ diagonal matrix with diagonal elements a_1 through a_N is represented as $\mathrm{diag}\{a_1, a_2, ..., a_N\}$, while $\boldsymbol{I}_M$ represents the $M \times M$ identity matrix. The $(i,j)^{th}$ element, the i^{th} row, and the j^{th} column of a matrix $\boldsymbol{A}$ are represented by $\boldsymbol{A}_{i,j}$, $\boldsymbol{A}_{i,:}$ and $\boldsymbol{A}_{:,j}$, respectively.

2. Brief Review of Probabilistic Tensor CPD With Nonnegative Factors [17]

In recent work [17], automatic tensor rank learning is achieved for tensor CPD with nonnegative factors. Its key idea is the adoption of the nonnegative Gaussian-gamma prior model for the probabilistic modeling of $\{\boldsymbol{\Xi}_{:,r}^{(n)}\}_{n=1}^{N}$, and

the utilization of VI to enable tractable algorithm derivation. In the section, we briefly review the key results of [17].

2.1. Probabilistic Modeling And Analysis

The probabilistic model proposed in [17] is specified at the top of the next page (labeled as Probabilistic NCPD Model). In its prior distributions, parameter γ_l denotes the precision (i.e., the inverse of the variance) of the l^{th} columns of $\{\mathbf{\Xi}^{(n)}_{:,l}\}_{n=1}^{N}$, and L is a pre-selected upper bound for the tensor rank. Function $\mathrm{U}\left(\mathbf{\Xi}^{(n)}_{:,l} \geq \mathbf{0}_{J_n \times 1}\right)$ is the unit-step function with value one when $\mathbf{\Xi}^{(n)}_{:,l} \geq \mathbf{0}_{J_n \times 1}$ and with value zero otherwise, which enforces the non-negativeness of the factor matrices. In [17], it was shown that the employed nonnegative Gaussian-gamma prior distribution not only satisfies conjugacy property[1] in exponential distribution family, but also is sparsity-promoting. Moreover, c_l^0 and d_l^0 were suggested to take very small values (e.g., 10^{-6}) to approach a non-informative prior for the precision parameter γ_l. In its likelihood model, parameter β represents the inverse of noise power, and was modeled to obey a non-informative gamma distribution $p(\beta|e^0, f^0) = \mathrm{gamma}(\beta|e^0, f^0)$ with $\{e^0, f^0\}$ being a very small number (e.g., 10^{-6}) [17]. The use of non-informative prior is common in Bayesian model and these parameters are not tuned to influence the performance [40]. To make the problem well-conditioned, and match the prior assumption of the probabilistic model in [17], we assume the factor matrices $\{\mathbf{\Xi}^{(n)}\}_{n=1}^{N}$ are of full column rank.

2.2. Inference Algorithm Using VI

In Bayesian inference, the posterior distribution of each unknown parameter $\mathbf{\Theta}_k$ in the set $\mathbf{\Theta} = \{\{\mathbf{\Xi}^{(n)}\}_{n=1}^{N}, \beta, \{\gamma_l\}_{l=1}^{L}\}$ is aimed to be computed. Following the Bayes' rule, it can be derived by $p(\mathbf{\Theta}_k|\mathcal{Y}) = \int \frac{p(\mathbf{\Theta},\mathcal{Y})}{\int p(\mathbf{\Theta},\mathcal{Y}) d\mathbf{\Theta}} d\mathbf{\Theta}_{\{j\neq k\}}$, where $\mathbf{\Theta}_{\{j\neq k\}}$ denotes all the elements in the set $\mathbf{\Theta}$ except $\mathbf{\Theta}_k$. However, the multiple integrations involved do not allow an analytically tractable solution, due to the intricacy of the probabilistic model. Instead, in [17], VI were leveraged to derive a tractable algorithm. The key idea is to approximate the true posterior distribution by the variational probability density

[1] In Bayesian theory, a probability density function (pdf) $p(x)$ is said to be conjugate to a conditional pdf $p(y|x)$ if the resulting posterior $p(x|y)$ is in the same distribution family as $p(x)$.

Probabilistic NCPD Model: Probabilistic model for tensor CPD with nonnegative factors [17]

Likelihood function:

$$p\left(\mathcal{Y} \mid \boldsymbol{\Xi}^{(1)}, \boldsymbol{\Xi}^{(2)}, ..., \boldsymbol{\Xi}^{(N)}, \beta\right) \propto \exp\left(-\frac{\beta}{2} \parallel \mathcal{Y} - [\![\boldsymbol{\Xi}^{(1)}, \boldsymbol{\Xi}^{(2)}, ..., \boldsymbol{\Xi}^{(N)}]\!] \parallel_F^2\right)$$

Prior distributions:

$$\begin{aligned}
&p(\{\boldsymbol{\Xi}^{(n)}\}_{n=1}^N, \{\gamma_l\}_{l=1}^L) \\
&= \prod_{n=1}^{N}\prod_{l=1}^{L} \mathcal{N}(\boldsymbol{\Xi}_{:,l}^{(n)} | \mathbf{0}_{J_n \times 1}, \gamma_l^{-1} \boldsymbol{I}_{J_n}) \text{gamma}(\gamma_l | c_l^0, d_l^0) \text{U}\left(\boldsymbol{\Xi}_{:,l}^{(n)} \geq \mathbf{0}_{J_n \times 1}\right), \\
&p(\beta) = \text{gamma}(\beta | e^0, f^0).
\end{aligned}$$

function (pdf) $Q(\boldsymbol{\Theta})$ that minimizes the Kullback-Leibler (KL) divergence $\text{KL}\big(Q\left(\boldsymbol{\Theta}\right) \parallel p\left(\boldsymbol{\Theta} \mid \mathcal{Y}\right)\big) \triangleq -\mathbb{E}_{Q(\boldsymbol{\Theta})}\left\{\ln \frac{p(\boldsymbol{\Theta}|\mathcal{Y})}{Q(\boldsymbol{\Theta})}\right\}$, thus recasting the probabilistic inference problem into a functional optimization problem [30]. To achieve tractability, the variational pdf $Q(\boldsymbol{\Theta})$ is usually restricted to lie in the mean-field family $Q(\boldsymbol{\Theta}) = \prod_{k=1}^K Q(\boldsymbol{\Theta}_k)$, under which the recasted functional optimization problem can be generally stated as follows:

$$\min_{Q(\boldsymbol{\Theta})} -\mathbb{E}_{Q(\boldsymbol{\Theta})}\left[\ln p\left(\boldsymbol{\Theta}, \mathcal{Y}\right)\right] + \mathbb{E}_{Q(\boldsymbol{\Theta})}\left[Q\left(\boldsymbol{\Theta}\right)\right] \quad \text{s.t. } Q(\boldsymbol{\Theta}) = \prod_{k=1}^{K} Q(\boldsymbol{\Theta}_k), \tag{2}$$

where $p\left(\boldsymbol{\Theta}, \mathcal{Y}\right)$ is the joint pdf for Probabilistic NCPD Model. It has been shown in [30] that problem (2) enjoys the block multi-convex property in the functional space, in the sense that after fixing other variational pdfs $\{Q(\boldsymbol{\Theta}_j)\}_{j \neq k}$, the optimization problem for $Q(\boldsymbol{\Theta}_k)$ is convex and thus can be solved efficiently [30].

In [17], it was shown that $Q(\beta)$ and $Q(\gamma_l)$ are gamma distributions. That is, in the iteration $t+1$, $Q^{t+1}(\beta) = \text{gamma}(\beta | e^{t+1}, f^{t+1})$ and $Q^{t+1}(\gamma_l) = \text{gamma}(\gamma_l | c_l^{t+1}, d_l^{t+1})$, where $e^{t+1}, f^{t+1}, c_l^{t+1}$ and d_l^{t+1} are shown in Algorithm 1 (at the top of the next page). However, when deriving $Q^{t+1}(\boldsymbol{\Xi}^{(n)})$, a multivariate truncated Gaussian distribution would be obtained, of which the moments are very difficult to acquire [34]. Instead, [17] suggests further adopting

a Dirac delta functional simplification $Q^{t+1}\left(\boldsymbol{\Xi}^{(n)}\right) = \delta(\boldsymbol{\Xi}^{(n)} - \hat{\boldsymbol{\Xi}}^{(n),t+1})$, where $\hat{\boldsymbol{\Xi}}^{(n),t+1}$ is the point estimate of the parameter $\boldsymbol{\Xi}^{(n)}$. With this simplification, the optimal point estimate $\hat{\boldsymbol{\Xi}}^{(n),t+1}$, which is also the expectation of the variational pdf $Q^{t+1}\left(\boldsymbol{\Xi}^{(n)}\right)$, can be obtained via solving the following quadratic programming (QP) problem:

$$\min \xi^{t+1}(\boldsymbol{\Xi}^{(n)}) \;\; \text{s.t.} \;\; \boldsymbol{\Xi}^{(n)} \geq \mathbf{0}_{J_n \times L}, \tag{3}$$

where

$$\begin{aligned}\xi^{t+1}(\boldsymbol{\Xi}^{(n)}) =& \frac{e^t}{2f^t} \parallel \mathcal{Y} - [\![\hat{\boldsymbol{\Xi}}^{(1),t+1}, \cdots, \boldsymbol{\Xi}^{(n)}, \cdots, \hat{\boldsymbol{\Xi}}^{(N),t}]\!]\parallel_F^2 \\ &+ \frac{1}{2}\mathrm{Tr}\Big(\boldsymbol{\Xi}^{(n)}\mathrm{diag}\{\frac{c_1^t}{d_1^t}, ..., \frac{c_L^t}{d_L^t}\}\boldsymbol{\Xi}^{(n)T}\Big).\end{aligned} \tag{4}$$

Consequently, by iteratively updating variational pdfs $\{Q(\beta), \{Q(\gamma_l)\}_{l=1}^L\}$ and solving (3), the probabilistic tensor CPD algorithm was obtained and is summarized in Algorithm 1.

Notice that although the original unknown variables are $\boldsymbol{\Theta} = \{\{\boldsymbol{\Xi}^{(n)}\}_{n=1}^N, \beta, \{\gamma_l\}_{l=1}^N\}$, Algorithm 1 updates the hyper-parameters $\{e, f\}$ of β and hyper-parameters $\{c_l, d_l\}$ of γ_l rather than β and γ_l directly. This is because Bayesian algorithms estimate the distributions of the parameters, rather than the point estimates. After we obtain the distribution estimate, we can summarize the distribution using any statistical measure, such as mean or mode. For Algorithm 1, if the mean of the varational distribution is taken, the estimates of γ_l and β are $\mathbb{E}[\gamma_l] = c_l^t/d_l^t$ and $\mathbb{E}[\beta] = e^t/f^t$, respectively.

Since γ_l represents the precision of the l-th rank-1 component, the rank learning is achieved by pruning out components with c_l/d_l larger than a pre-defined threshold [17]. This is in fact equivalent to pruning the rank-1 components with small norms $\sum_{n=1}^N \frac{1}{2}[\hat{\boldsymbol{\Xi}}_{:,l}^{(n)}]^\top \hat{\boldsymbol{\Xi}}_{:,l}^{(n)}$ but with a different threshold. More specifically, according to the update of d_l in Algorithm 1, with the choice of small c_l^0 and d_l^0, e.g., 10^{-6}, components with small norms will lead to small d_l. Further with the fact that c_l is only updated once in the first iteration and will stay as constant, a small d_l would lead to a large c_l/d_l. Therefore, it is reasonable to prune out components with c_l/d_l larger than a certain threshold.

Algorithm 1: VI Based Probabilistic Tensor CPD with Nonnegative Factors [17]

Initializations: Choose $L > R$ and initial values $\{\hat{\boldsymbol{\Xi}}^{(n),0}\}_{n=1}^{N}$, $\{\{c_l^0, d_l^0\}_{l=1}^{L}, e^0, f^0\}$.

Iterations: For the iteration $t+1$ ($t \geq 0$),

For $n = 1$ **to** N

Update the parameter of $Q^{t+1}(\boldsymbol{\Xi}^{(n)})$

$$\hat{\boldsymbol{\Xi}}^{(n),t+1} = \arg\min_{\boldsymbol{\Xi}^{(n)} \geq \mathbf{0}_{J_n \times L}} \xi^{t+1}(\boldsymbol{\Xi}^{(n)}),$$

where $\xi^{t+1}(\boldsymbol{\Xi}^{(n)})$ is given in (4).

Here $\hat{\boldsymbol{\Xi}}^{(n),t+1}$ can be obtained by any off-the-shelf QP algorithm. The Nestorov-based acceleration (Algorithm 3 in Appendix B) is one particular choice.

End

Update the parameter of $Q^{t+1}(\gamma_l)$

$$c_l^{t+1} = \sum_{n=1}^{N} \frac{J_n}{2} + c_l^0$$

$$d_l^{t+1} = \sum_{n=1}^{N} \frac{1}{2} \left[\hat{\boldsymbol{\Xi}}_{:,l}^{(n),t+1}\right]^T \hat{\boldsymbol{\Xi}}_{:,l}^{(n),t+1} + d_l^0$$

Update the parameter of $Q^{t+1}(\beta)$

$$e^{t+1} = \frac{\prod_{n=1}^{N} J_n}{2} + e^0$$

$$f^{t+1} = \frac{1}{2} \parallel \mathcal{Y} - [\![\hat{\boldsymbol{\Xi}}^{(1),t+1}, \hat{\boldsymbol{\Xi}}^{(2),t+1}, \cdots, \hat{\boldsymbol{\Xi}}^{(N),t+1}]\!] \parallel_F^2 + f^0$$

Until Convergence

3. Equivalent Optimization Problem

From the review of the algorithm above, it can be seen that the computational bottleneck is due to solving problem (3), while the updates of other variational pdfs take closed-forms, and thus are obtained with inexpensive

computations. To alleviate the computational burden, first-order method with Nesterov-based acceleration was implemented in [17] and summarized in Algorithm 3 in Appendix B, by which saving of running time was observed compared to the projected gradient descent method. However, even with advanced acceleration scheme, the QP problem (3) still needs many iterations to compute a solution, thus is time-consuming.

Since Algorithm 1 resembles an algorithm from conventional optimization, one might contemplate borrowing the idea from inexact BCD methods [13, 26, 27]. That is, in each VI iteration, the QP problem (3) is not completely solved, but only a better solution is provided, thus avoiding troublesome inner-loop computations. However, inexact BCD was developed for deterministic optimization, and it is not known if the solution obtained from such modification would be meaningful from VI perspective. In particular, the convergence of mean-field VI, under which Algorithm 1 is developed, is established by assuming each subproblem is optimally solved [37]. To the best knowledge of authors, there is still no rigorous convergence analysis for the VI algorithm with inexact update steps.

However, a closer inspection on the update equations in Algorithm 1 reveals that although the variational pdfs for $\{Q(\beta), \{Q(\gamma_l)\}_{l=1}^L\}$ were optimized via updating their parameters $\{e, f, \{c_l, d_l\}_{l=1}^L\}$, only the expectations of β and γ_l, i.e., $\{e/f, \{c_l/d_l\}_{l=1}^L\}$ were passed to the factor matrix update in (3). Therefore, if $\{e, f, \{c_l, d_l\}_{l=1}^L\}$ could be acquired from certain optimization subproblems, Algorithm 1 could be interpreted as the updating steps of a BCD algorithm solving a deterministic optimization problem.

Although challenging as it may seem, it turns out that Algorithm 1 is related to solving the following deterministic optimization problem:

$$\min_{\{\boldsymbol{\Xi}^{(n)} \geq \mathbf{0}\}_{n=1}^N, \{\gamma_l\}_{l=1}^L, \beta} g(\boldsymbol{\Theta}), \tag{5}$$

where

$$\begin{aligned} & g(\boldsymbol{\Theta}) \\ & = -\frac{\prod_{n=1}^N J_n}{2} \ln \beta + \frac{\beta}{2} \parallel \mathcal{Y} - [\![\boldsymbol{\Xi}^{(1)}, \boldsymbol{\Xi}^{(2)}, \cdots, \boldsymbol{\Xi}^{(N)}]\!] \parallel_F^2 - \sum_{n=1}^N \frac{J_n}{2} \sum_{l=1}^L \ln \gamma_l \\ & \quad + \sum_{n=1}^N \frac{1}{2} \mathrm{Tr}\Big(\boldsymbol{\Xi}^{(n)} \boldsymbol{\Gamma} \boldsymbol{\Xi}^{(n)T}\Big) - \sum_{l=1}^L \left[c_l^0 \ln \gamma_l - d_l^0 \gamma_l\right] - e^0 \ln \beta + f^0 \beta, \end{aligned} \tag{6}$$

with $\mathbf{\Gamma} = \text{diag}\{\gamma_1, ..., \gamma_L\}$. To see the connections between solving (5) and Algorithm 1, BCD method could be employed. That is, in each iteration, after fixing other unknown parameters $\{\mathbf{\Theta}_j\}_{j \neq k}$ at their last updated values, $\mathbf{\Theta}_k$ is updated as follows:

$$\mathbf{\Theta}_k^{t+1} = \arg\min_{\mathbf{\Theta}_k} g(\mathbf{\Theta}_1^{t+1}, ..., \mathbf{\Theta}_{k-1}^{t+1}, \mathbf{\Theta}_k, \mathbf{\Theta}_{k+1}^{t}, ..., \mathbf{\Theta}_K^{t}). \tag{7}$$

Next, we present the key proposition of this paper, which links the problem (5) to Algorithm 1.

Proposition 1. *Assume each initial value of the unknown parameter $\mathbf{\Theta}_k^0$ in (7) equal to the expectation of $\mathbf{\Theta}_k$ with respect to the initial variational pdf $Q^0(\mathbf{\Theta}_k)$ in Algorithm 1. With the same update schedule for various parameter blocks, in each iteration, the result of the block minimization update* (7) *for parameter $\mathbf{\Theta}_k$ equals to the expectation of $\mathbf{\Theta}_k$ with respect to the variational pdf $Q(\mathbf{\Theta}_k)$ from Algorithm 1, i.e., $\mathbf{\Theta}_k^t = \mathbb{E}_{Q^t(\mathbf{\Theta}_k)}[\mathbf{\Theta}_k]$.*
Proof: See Appendix A.

As pointed out in **Proposition 1**, Algorithm 1 is indeed related to parameter block minimization of the optimization problem (5), and is further illustrated in Figure 1. This new interpretation of Algorithm 1 opens up the possibility of using inexact BCD to accelerate Algorithm 1.

Notice that the problem formulation (5) is established with reference to the analysis of the Bayesian approach [17], and thus solving (5) yields estimates of both tensor rank and noise power. This is in contrast to conventional optimization formulations [13] which requires tensor rank or regularization parameter tuning.

4. Acceleration via Inexact BCD Approach

Inexact BCD refers to a class of methods that do not seek the optimal solution in each BCD block minimization (7) while still guarantee the convergence to a stationary point. Among all the recent advances of inexact BCD [13, 26, 27], we demonstrate in this paper the framework from [13] while other frameworks are also applicable. In particular, in [13], the convergence of inexact BCD is established if each of the update step is either a block minimization with strongly convex objective function, a proximal update with regularization parameter being positive and upper bounded, or a prox-linear

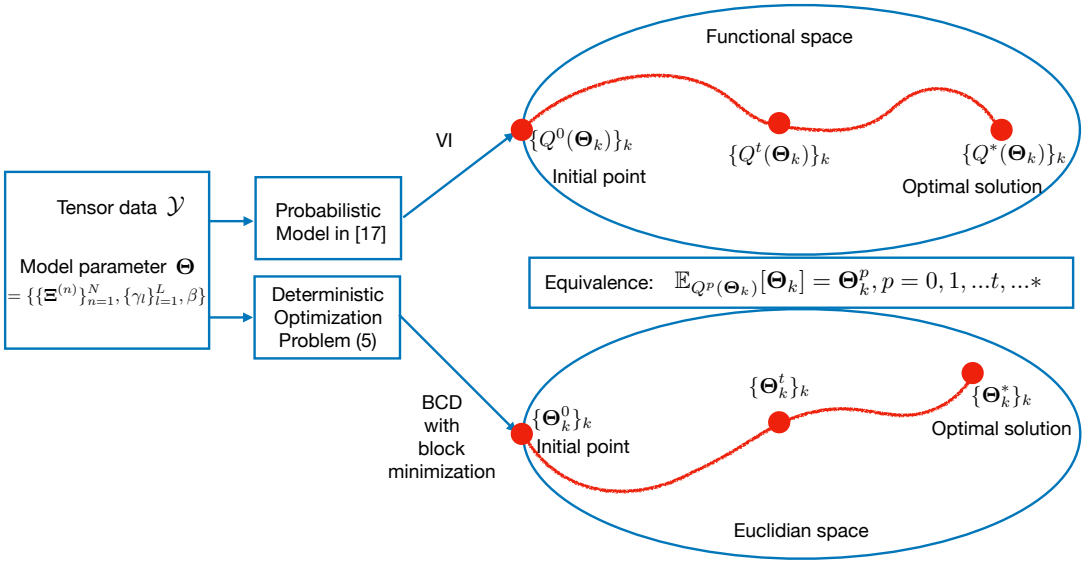

Figure 1: Illustration of equivalent optimization.

update with extrapolation and monotonicity check. Although [13] uses the CPD problem (1) to exemplify its framework, tensor rank determination and noise power learning cannot be easily incorporated. In the following, by using the equivalent optimization problem (5), we demonstrate how the update steps for parameters $\{\{\gamma_l\}_{l=1}^{L}, \beta\}$ could be seamlessly integrated with the nonnegative factor learning in the inexact BCD framework.

More specifically, for updating noise precision parameter β in the iteration $t+1 (t \geq 0)$, after fixing other parameters in problem (5), the subproblem is expressed as:

$$\min_{\beta} h^{t+1}(\beta), \tag{8}$$

where

$$\begin{aligned} h^{t+1}(\beta) = &- \left(\frac{\prod_{n=1}^{N} J_n}{2} + e^0 \right) \ln \beta \\ &+ \beta \left(\frac{1}{2} \parallel \mathcal{Y} - [\![\boldsymbol{\Xi}^{(1),t+1}, \boldsymbol{\Xi}^{(2),t+1}, \cdots, \boldsymbol{\Xi}^{(N),t+1}]\!] \parallel_F^2 + f^0 \right). \end{aligned} \tag{9}$$

However, the objective function $h^{t+1}(\beta)$ is not strongly convex since the second-order derivative $\nabla^2_\beta h^{t+1}(\beta) = \left(\frac{\prod_{n=1}^N J_n}{2} + e^0\right)\frac{1}{\beta^2}$ can be arbitrarily close to zero as $\beta \to \infty$. Consequently, the block minimization scheme cannot be used. To guarantee convergence, as suggested in [13, 25, 26, 27, 28], a proximal term $\frac{\mu_\beta}{2}(\beta - \beta^t)^2$ is added to the objective function in (8), giving the following optimization problem:

$$\min_{\beta} h^{t+1}(\beta) + \frac{\mu_\beta}{2}(\beta - \beta^t)^2, \tag{10}$$

with parameter $0 < \mu_\beta < \infty$. After setting the derivative of the objective function of (10) to zero, it can be shown that the optimal solution takes a closed-form:

$$\beta^{t+1} = \frac{-\left(f^{t+1} - \mu_\beta \beta^t\right) + \sqrt{\left(f^{t+1} - \mu_\beta \beta^t\right)^2 + 4\mu_\beta e^{t+1}}}{2\mu_\beta}, \tag{11}$$

in which

$$e^{t+1} = \frac{\prod_{n=1}^N J_n}{2} + e^0, \tag{12}$$

$$f^{t+1} = \frac{1}{2} \parallel \mathcal{Y} - [\![\mathbf{\Xi}^{(1),t+1}, \mathbf{\Xi}^{(2),t+1}, \cdots, \mathbf{\Xi}^{(N),t+1}]\!] \parallel_F^2 + f^0. \tag{13}$$

Similarly, for updating parameter γ_l, the subproblem is

$$\min_{\gamma_l} h^{t+1}(\gamma_l), \tag{14}$$

where

$$h^{t+1}(\gamma_l) = -(\sum_{n=1}^N \frac{J_n}{2} + c_l^0)\ln \gamma_l + \gamma_l (d_l^0 + \sum_{n=1}^N \frac{1}{2}\left[\mathbf{\Xi}_{:,l}^{(n),t+1}\right]^T \mathbf{\Xi}_{:,l}^{(n),t+1}) \tag{15}$$

is not strongly convex. Therefore, a proximal term $\frac{\mu_{\gamma_l}}{2}(\gamma_l - \gamma_l^t)^2$ with $0 < \mu_{\gamma_l} < \infty$ is added to the objective function in (14) as follows [13, 25, 26, 27, 28]:

$$\min_{\gamma_l} h^{t+1}(\gamma_l) + \frac{\mu_{\gamma_l}}{2}(\gamma_l - \gamma_l^t)^2, \tag{16}$$

of which the optimal solution can be shown to be

$$\gamma_l^{t+1} = \frac{-\left(d_l^{t+1} - \mu_{\gamma_l}\gamma_l^t\right) + \sqrt{\left(d_l^{t+1} - \mu_{\gamma_l}\gamma_l^t\right)^2 + 4\mu_{\gamma_l} c_l^{t+1}}}{2\mu_{\gamma_l}}, \tag{17}$$

where parameters c_l^{t+1} and d_l^{t+1} are

$$c_l^{t+1} = \sum_{n=1}^{N} \frac{J_n}{2} + c_l^0, \tag{18}$$

$$d_l^{t+1} = \sum_{n=1}^{N} \frac{1}{2} \left[\mathbf{\Xi}_{:,l}^{(n),t+1} \right]^T \mathbf{\Xi}_{:,l}^{(n),t+1} + d_l^0. \tag{19}$$

For updating each nonnegative factor in the iteration $t+1 (t \geq 0)$, after fixing other parameters, the subproblem can be formulated as:

$$\min_{\mathbf{\Xi}^{(n)} \geq \mathbf{0}} h^{t+1}(\mathbf{\Xi}^{(n)}), \tag{20}$$

where

$$h^{t+1}(\mathbf{\Xi}^{(n)}) = \frac{\beta^t}{2} \parallel \mathcal{Y} - [\![\mathbf{\Xi}^{(1),t+1}, \cdots, \mathbf{\Xi}^{(n)}, \cdots, \mathbf{\Xi}^{(N),t}]\!] \parallel_F^2 + \frac{1}{2} \mathrm{Tr} \Big(\mathbf{\Xi}^{(n)} \mathbf{\Gamma}^t \mathbf{\Xi}^{(n)T} \Big). \tag{21}$$

In (21), $\mathbf{\Gamma}^t = \mathrm{diag}\{\gamma_1^t, ..., \gamma_L^t\}$. After expanding the Frobenius norm and some algebraic operations, problem (20) can be equivalently formulated as:

$$\min_{\mathbf{\Xi}^{(n)} \geq \mathbf{0}} c^{t+1}(\mathbf{\Xi}^{(n)}), \tag{22}$$

where

$$c^{t+1}(\mathbf{\Xi}^{(n)}) = \frac{1}{2} \mathrm{Tr} \left(\mathbf{\Xi}^{(n)} \left[\beta^t \left(\mathbf{B}^{(n),t} \right)^T \mathbf{B}^{(n),t} + \mathbf{\Gamma}^t \right] \mathbf{\Xi}^{(n)T} - 2\beta^t \mathbf{\Xi}^{(n)} \left(\mathbf{B}^{(n),t} \right)^T \mathcal{Y}^{(n)} \right). \tag{23}$$

In (23), matrix $\mathbf{B}^{(n),t} = \mathbf{\Xi}^{(N),t} \diamond \cdots \diamond \mathbf{\Xi}^{(n+1),t} \diamond \mathbf{\Xi}^{(n-1),t+1} \diamond \cdots \diamond \mathbf{\Xi}^{(1),t+1}$, where $\diamond$ denotes the Khatri-Rao product. $\mathcal{Y}^{(k)}$ is a matrix obtained by unfolding the tensor $\mathcal{Y}$ along its k^{th} dimension.

The most straightforward method to solve (22) is the projected gradient method, which needs iterative execution of $\mathbf{\Xi}^{(n),t+1} = \left[\mathbf{\Xi}^{(n),t} - \mu_t \nabla c^{t+1}(\mathbf{\Xi}^{(n)}) \right]_+$ where μ_t is the step size,

$$\nabla c^{t+1}(\mathbf{\Xi}^{(n)}) = \mathbf{\Xi}^{(n)} \left[\beta^t \left(\mathbf{B}^{(n),t} \right)^T \mathbf{B}^{(n),t} + \mathbf{\Gamma}^t \right] - \beta^t \mathcal{Y}^{(n)T} \mathbf{B}^{(n),t}, \tag{24}$$

and $[\cdot]_+$ denotes projecting each element of the argument to $[0, \infty]$ (i.e., $[x]_+ = x$ if $x \geq 0$ and $[x]_+ = 0$ otherwise). However, the projected gradient method requires multiple iterations to converge and thus is computationally demanding.

Fortunately, it is not necessary to obtain the optimal solution of (22) in the inexact BCD framework. Instead, we can construct a prox-linear update step with careful extrapolations and monotonicity-guaranteed modifications. In particular, in the iteration $t+1$, the extrapolation parameter w_n^t is computed by [13, 26, 27, 29]:

$$w_n^t = \min\left(\hat{w}_n^t, p_w\sqrt{\frac{L_n^{t-1}}{L_n^t}}\right), \tag{25}$$

where $p_w < 1$ is preselected, $\hat{w}_n^t = \frac{s_{t-1}}{s_{t+1}}$ with $s_0 = 1$, $s_{t+1} = \frac{1}{2}(1 + \sqrt{1 + 4s_t^2})$, and L_n^t is assigned to be the spectral norm of the Hessian matrix of $c^{t+1}(\boldsymbol{\Xi}^{(n)})$, i.e.,

$$L_n^t = ||\beta^t \left(\mathbf{B}^{(n),t}\right)^T \mathbf{B}^{(n),t} + \boldsymbol{\Gamma}^t||_2. \tag{26}$$

Using (25), the extrapolated factor matrix $\hat{\boldsymbol{M}}^{(n),t}$ is with the expression:

$$\hat{\boldsymbol{M}}^{(n),t} = \boldsymbol{\Xi}^{(n),t} + w_n^t(\boldsymbol{\Xi}^{(n),t} - \boldsymbol{\Xi}^{(n),t-1}). \tag{27}$$

Based on (27), the prox-linear update can be expressed as [13]:

$$\boldsymbol{\Xi}^{(n),t+1} = \arg\min_{\boldsymbol{\Xi}^{(n)} \geq \mathbf{0}} \left\langle \nabla c^{t+1}(\hat{\boldsymbol{M}}^{(n),t}), \boldsymbol{\Xi}^{(n)} - \hat{\boldsymbol{M}}^{(n),t} \right\rangle + \frac{L_n^t}{2}||\boldsymbol{\Xi}^{(n)} - \hat{\boldsymbol{M}}^{(n),t}||_F^2, \tag{28}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product of the arguments. It can be shown that the solution of (28) takes the following closed-form:

$$\boldsymbol{\Xi}^{(n),t+1} = \left[\hat{\boldsymbol{M}}^{(n),t} - \frac{1}{L_n^t}\nabla c^{t+1}(\hat{\boldsymbol{M}}^{(n),t})\right]_+, \tag{29}$$

from which it can be seen that the prox-linear update only needs one-step computation and thus can be run very fast.

Besides the extrapolation, monotonicity-guaranteed modification is needed to ensure the convergence [13]. More specifically, after updating all the parameters in $\boldsymbol{\Theta}$, whether the objective function of problem (5) is decreased

(i.e., $g(\boldsymbol{\Theta}^{t+1}) \leq g(\boldsymbol{\Theta}^t)$) is tested. If not, prox-linear update (29) for each factor matrix $\boldsymbol{\Xi}^{(n),t+1}$ will be re-executed without the extrapolation, i.e.,

$$\boldsymbol{\Xi}^{(n),t+1} = \left[\boldsymbol{\Xi}^{(n),t} - \frac{1}{L_n^t}\nabla c^{t+1}(\boldsymbol{\Xi}^{(n),t})\right]_+. \tag{30}$$

Using (11), (17), (29), (30), the inexact BCD based algorithm for probabilistic tensor CPD with nonnegative factors is summarized in Algorithm 2. In the following, discussions on its convergence property, automatic tensor rank learning, and computational complexity are presented.

4.1. Convergence Property

Algorithm 2 is developed under the inexact BCD framework established in [13]. Due to the added proximal terms, the subproblems (10) and (16) are both strongly convex with a positive finite modulus [25, 28]. On the other hand, when updating factor matrices, the extrapolation scheme and monotonicity-guaranteed modification follow those suggested in [13]. Therefore, according to Theorem 2.8 of [13], assume that the step size L_n^t is positive and upper bounded (which always holds under finite $\{\gamma_l\}_{l=1}^L$ according to (26)), the limit point generated by the proposed algorithm is guaranteed to be a stationary point of the objective function (5).

4.2. Automatic Tensor Rank Learning

Although the prox-linear update step in Algorithm 2 resembles that of [13], these similarities are only on a superficial level, and there is a significant difference due to the tensor rank learning. More specifically, for the proposed algorithm, when computing the gradient function (24), there is a shrinkage term $\boldsymbol{\Gamma}^t$ that collects the precision parameters $\{\gamma_l^t\}_{l=1}^L$. If one of the precision parameter γ_l^t goes to a very large number, e.g., 10^6, the update step using the gradient would push the l^{th} column of the factor matrix to be all zero, thus achieving automatic column pruning (and tensor rank learning). With the precision parameters $\{\gamma_l^t\}_{l=1}^L$ being iteratively updated together with other parameters in the optimization problem (5), which is deduced from the analysis of the VI-based algorithm proposed in [17], the proposed algorithm integrates the merits from [13] and [17] in a non-trivial way.

To further speed up the computation, pruning can be executed on the fly, rather than after Algorithm 2 converges. In particular, when executing

Algorithm 2: Inexact BCD Based Probabilistic Tensor CPD with Nonnegative Factors

Initializations: Choose $L > R, p_w < 1$, $\{\mu_{\gamma_l}\}_{l=1}^{L}$, μ_β and initial values $\{\boldsymbol{\Xi}^{(n),0}, \boldsymbol{\Xi}^{(n),-1}\}_{n=1}^{N}$, $\{c_l^0, d_l^0\}_{l=1}^{L}$, e^0, f^0.

Iterations: For the iteration $t+1$ ($t \geq 0$),

For $n = 1$ **to** N

Update factor matrix $\boldsymbol{\Xi}^{(n),t+1}$:

Compute extrapolation parameter w_n^t using (25).

Compute extrapolated factor matrix:

$$\hat{\boldsymbol{M}}^{(n),t} = \boldsymbol{\Xi}^{(n),t} + w_n^t(\boldsymbol{\Xi}^{(n),t} - \boldsymbol{\Xi}^{(n),t-1}).$$

Update factor matrix:

$$\boldsymbol{\Xi}^{(n),t+1} = \left[\hat{\boldsymbol{M}}^{(n),t} - \frac{1}{L_n^t}\nabla c^{t+1}(\hat{\boldsymbol{M}}^{(n),t})\right]_+,$$

where $\nabla c^{t+1}(\hat{\boldsymbol{M}}^{(n),t})$ is computed using (24), and L_n^t is computed using (26).

End

Update parameter γ_l^{t+1}:

$$\gamma_l^{t+1} = \frac{-\left(d_l^{t+1} - \mu_{\gamma_l}\gamma_l^t\right) + \sqrt{\left(d_l^{t+1} - \mu_{\gamma_l}\gamma_l^t\right)^2 + 4\mu_{\gamma_l}c_l^{t+1}}}{2\mu_{\gamma_l}},$$

where c_l^{t+1} and d_l^{t+1} are computed via (18) and (19), respectively.

Update parameter β^{t+1}:

$$\beta^{t+1} = \frac{-\left(f^{t+1} - \mu_\beta\beta^t\right) + \sqrt{\left(f^{t+1} - \mu_\beta\beta^t\right)^2 + 4\mu_\beta e^{t+1}}}{2\mu_\beta},$$

where e^{t+1} and f^{t+1} are computed via (12) and (13), respectively.

Monotonicity check:

Let $\boldsymbol{\Theta}^{t+1} = \{\{\boldsymbol{\Xi}^{(n),t+1}\}_{n=1}^{N}, \{\gamma_l^{t+1}\}_{l=1}^{L}, \beta^{t+1}\}$.

If $g(\boldsymbol{\Theta}^{t+1}) > g(\boldsymbol{\Theta}^t)$:

$$\boldsymbol{\Xi}^{(n),t+1} = \left[\boldsymbol{\Xi}^{(n),t} - \frac{1}{L_n^t}\nabla c^{t+1}(\boldsymbol{\Xi}^{(n),t})\right]_+.$$

Until Convergence

Algorithm 2 in the iteration $t+1 (t \geq 0)$, if parameter γ_l^{t+1} is with a very large value, e.g., 10^6, it indicates that the l^{th} building block $\{\boldsymbol{\Xi}_{:,l}^{(n),t+1}\}$ plays no role in data interpretation, and thus can be safely pruned out. Notice that after pruning $\{\boldsymbol{\Xi}_{:,l}^{(n),t+1}\}$, it is equivalent to restart Algorithm 2 for optimization problem (5) with a reduced model and the current parameter estimate $\boldsymbol{\Theta}^{t+1}$ acting as the initializations. Thus, the pruning steps will not affect the convergence property of the algorithm. This pruning principle was widely used in sparse Bayesian learning based applications [18, 19, 20, 21, 22, 23, 24].

4.3. Computational Complexity

For each iteration, it is easy to see that the computational complexity is dominated by computing the gradient function in (24), costing $O(\prod_{n=1}^{N} J_n L)$. Consequently, the proposed algorithm is with the computational complexity $O(q(\prod_{n=1}^{N} J_n L))$, where q is the iteration number at convergence. In contrast to the VI-based algorithm in [17], in which each factor matrix was updated via multiple iterations to solve the block minimization problem, the computational complexity is $O(\sum_{r=1}^{q} m_r \prod_{n=1}^{N} J_n L)$, where m_r is the number of iterations to solve the block minimization problem. Since the proposed algorithm does not involve any inner iterations, and thus is expected to save running time significantly, which will be corroborated in the numerical studies in the next section. Note that the NALS algorithm, whose computational complexity is $O(q(\prod_{n=1}^{N} J_n L))$, has a complexity of the same order of the proposed algorithm.

5. Numerical Results on Synthetic Data

In this section, numerical results are presented to assess the performance of the proposed algorithm using synthetic data. For all algorithms, the preselected rank upper bound $L = \min\{J_1, J_2, ..., J_N\}$ is set to be the minimum dimension, unless otherwise stated. The initial factor matrix $\boldsymbol{\Xi}^{(n),0}$ is set as the singular value decomposition (SVD) approximation $\boldsymbol{U}_{:,1:L} (\boldsymbol{S}_{1:L,1:L})^{\frac{1}{2}}$, where $[\boldsymbol{U}, \boldsymbol{S}, \boldsymbol{V}] = \text{SVD}[\mathcal{Y}^{(n)}]$. Parameters $\{\{c_l^0, d_l^0\}_{l=1}^{L}, e^0, f^0\}$ are all set to be 10^{-6} to indicate the non-informativeness of the prior distributions [17]. The algorithms are deemed to be converged when the change of residual is below a threshold, i.e., $\frac{1}{\prod J_n} \parallel [\![\boldsymbol{\Xi}^{(1),t+1}, \cdots, \boldsymbol{\Xi}^{(N),t+1}]\!] - [\![\boldsymbol{\Xi}^{(1),t}, \cdots, \boldsymbol{\Xi}^{(N),t}]\!] \parallel_F < \epsilon$. For the proposed algorithm, parameter p_w is set to be 0.99, and $\{\{\mu_{\gamma_l}\}_{l=1}^{L}, \mu_\beta\}$ are set to be 10^{-3}. All experiments were conducted in Matlab R2021a with an Intel Core i5 CPU at 2.5 GHz.

5.1. Simulation Setup

The noise-free data tensor $\mathcal{X} = [\![\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \boldsymbol{A}^{(3)}]\!] \in \mathbb{R}^{20\times 50\times 100}$ with rank $R = 10$ is considered in this subsection. Each element in the factor matrices $\{\boldsymbol{A}^{(n)}\}_{n=1}^{3}$ is independently drawn from a uniform distribution $[0, 1]$ and thus is nonnegative. The observation data model $\mathcal{Y} = \mathcal{X} + \mathcal{W}$, where each element of the noise tensor $\mathcal{W}$ is independently drawn from a zero-mean Gaussian distribution with variance σ_w^2. The SNR is defined as $10\log_{10}\left(\| \mathcal{X} \|_F^2 / \mathbb{E}_{p(\mathcal{W})}[\| \mathcal{W} \|_F^2]\right) = 10\log_{10}\left(\| \mathcal{X} \|_F^2 / (10^5\sigma_w^2)\right)$. All simulation results in this subsection are obtained by averaging 100 Monte-Carlo runs.

5.2. Comparisons with PNCPD and PNCPD-A

Firstly, the tensor rank learning capability of the VI-based algorithm (labeled as PNCPD), its accelerated version using Nesterov scheme (labeled as PNCPD-A), and the proposed algorithm (labeled as PNCPD-I) under different SNRs is evaluated in Figure 2, in which the vertical bars show the mean and the error bars indicating the standard deviation of tensor rank estimates. The black horizontal dashed line shows the true tensor rank. From Figure 2, it can be seen that the three algorithms have similar rank estimation abilities. In particular, they can learn the true tensor rank with 100% accuracy when SNR is larger than or equal to 10 dB. However, when SNR is smaller than or equal to 5 dB, it can be observed that the three algorithms fail to recover the true tensor rank, since the noise with large power masks the low-rank structure of the data. Furthermore, an interesting observation is that when SNR is small, the proposed algorithm shows different rank learning behavior from those of the benchmarking algorithms. It suggests that in the region of low SNR (or equivalently the model fitting error is large), the proposed algorithm would converge to different stationary points from those of the benchmarking algorithms.

In addition to the tensor rank learning performance, the tensor recovery performance of the proposed algorithm is illustrated in Figure 3, where the root mean-square-error (RMSE) $\left(\frac{1}{\prod_n J_n}\left\|[\![\boldsymbol{\Xi}^{(1)}, \boldsymbol{\Xi}^{(2)}, \boldsymbol{\Xi}^{(3)}]\!] - \mathcal{X}\right\|_F^2\right)^{\frac{1}{2}}$ is adopted as the assessment measure. In Figure 3, the NALS method with the exact tensor rank $R = 10$ is employed as the benchmarking algorithm. From Figure 3, it can be seen that PNCPD-I, PNCPD, and PNCPD-A yield nearly the same RMSE. This result aligns with the observation from Figure 2, where all three algorithms have similar rank estimation ability over SNR = 0 dB

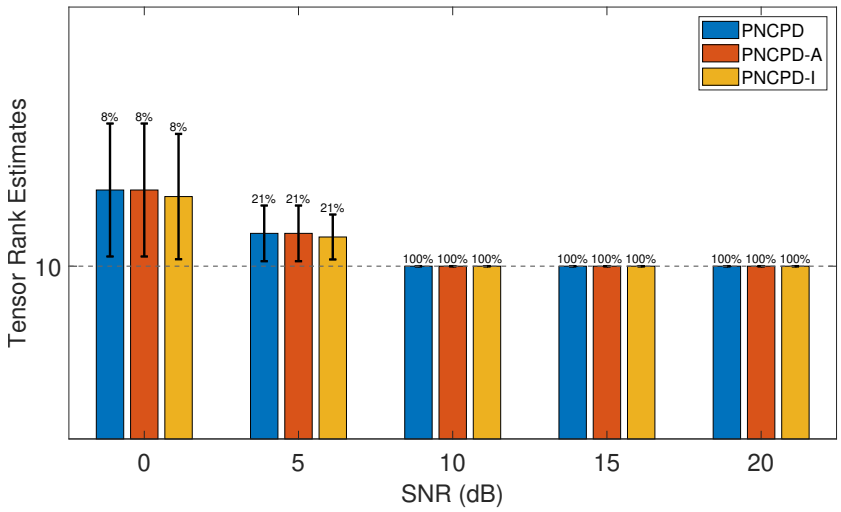

Figure 2: Tensor rank learning performance under different SNRs.

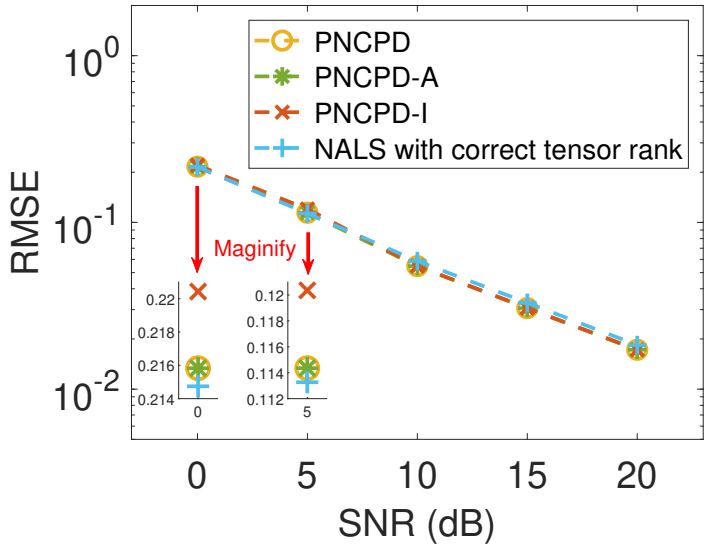

Figure 3: Tensor recovery performance under different SNRs.

to 20 dB. They perform slightly better than NALS with correct rank since PNCPD, PNCPD-A and PNCPD-I are Bayesian algorithms with noise component explicitly modeled. However, when SNR is smaller than or equal to 5 dB, the RMSEs of the three algorithms perform slightly worse than the

Table I: Average running time (in seconds) of the algorithms.

SNR (dB)	0	5	10	15	20
PNCPD	96.9882	80.9924	39.8039	42.1420	22.5843
PNCPD-A	89.0141	61.2056	30.6726	23.6034	12.6460
PNCPD-I	**3.0514**	**7.6629**	**5.7024**	**4.9418**	**4.0871**
NALS (trail-and-error, rank 1-20)	8.5160	10.0740	12.9480	17.5740	22.1660
NALS (with correct rank 10)	0.1023	0.1531	0.1860	0.1873	0.2216

benchmarking NALS algorithm, due to the inaccurate tensor rank learning. The degradations from NALS are small since PNCPD, PNCPD-A, PNCPD-I overestimate the tensor rank and the extra components recovered by them are with small norms. The RMSEs of the PNCPD-I are also different from PNCPD and PNCPD-A, since they converge to different stationary points.

Aside from the tensor recovery performance evaluations, we further report the RMSEs of factors to evaluate the tensor factor recovery performance. Due to the permutation and scaling ambiguity, the RMSE on the factors is found by solving the following optimization problem,

$$\min_{\boldsymbol{P},\{\boldsymbol{\Delta}^{(n)}\}_{n=1}^{3}} \left(\sum_{n=1}^{3} \frac{1}{J_n \max(R,\hat{R})} \left\| \hat{\boldsymbol{A}}^{(n)} - \hat{\boldsymbol{\Xi}}^{(n)} \boldsymbol{P} \boldsymbol{\Delta}^{(n)} \right\|_F^2 \right)^{\frac{1}{2}}, \tag{31}$$

where the permutation matrix $\boldsymbol{P}$ and diagonal scaling matrices $\{\boldsymbol{\Delta}^{(n)}\}_{n=1}^{3}$ are found by greedy least-squares column matching algorithm [35]. To cope with the situation where the estimated rank $\hat{R}$ does not equal to the true rank R, we construct $\hat{\boldsymbol{A}}^{(n)}$ and $\hat{\boldsymbol{\Xi}}$ to ensure compatible matrix calculations. In particular, if $\hat{R} < R$, we construct $\hat{\boldsymbol{\Xi}}^{(n)}$ by adding zero columns in the estimated factor matrix $\boldsymbol{\Xi}^{(n)}$, i.e., $\hat{\boldsymbol{\Xi}}^{(n)} = \left[\hat{\boldsymbol{\Xi}}^{(n)}, \boldsymbol{0}_{J_n \times (R-\hat{R})}\right]$, while the true factor matrix remains the same, i.e., $\hat{\boldsymbol{A}}^{(n)} = \boldsymbol{A}^{(n)}$. On the other hand, if $\hat{R} > R$, we set $\hat{\boldsymbol{A}}^{(n)} = \left[\hat{\boldsymbol{A}}^{(n)}, \boldsymbol{0}_{J_n \times (\hat{R}-R)}\right]$, $\hat{\boldsymbol{\Xi}}^{(n)} = \boldsymbol{\Xi}^{(n)}$. From Figure 4, PNCPD-I, PNCPD, and PNCPD-A gives similar RMSEs of factors and the RMSEs are near to that of the benchmarking NALS algorithm. However, since they give inaccurate tensor rank estimates when SNR is smaller than or equal to 5 dB, the RMSEs of factors are slightly worse than the benchmarking NALS algorithm when SNR is low.

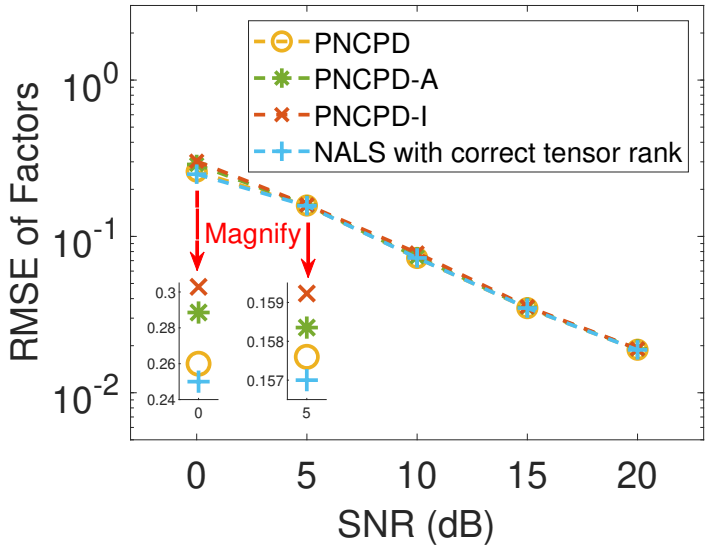

Figure 4: Factor recovery performance under different SNRs.

Although the proposed algorithm and the benchmarking algorithms are with similar performance in tensor recovery and nonnegative factor learning, the running time of these algorithms are largely different, as reported in Table I. From Table I, it is clear that the proposed algorithm has a speed-up ranging from 3× to more than 25× compared to the other two algorithms, because it does not need to solve the optimal factor matrix exactly in each iteration (in contrast to the PNCPD and the PNCPD-A methods). Furthermore, we report the sum of times running NALS with different ranks (1 to 20 in this example), since NALS needs the true tensor rank, which can only be found using trial-and-error. From Table I, it can be seen that the proposed algorithm is advantageous in running time compared with NALS algorithm using trail-and-error. By further comparing the running time of NALS using trial-and error and that of NALS with the correct rank, it can be seen that the increase of running time without rank specification is significant.

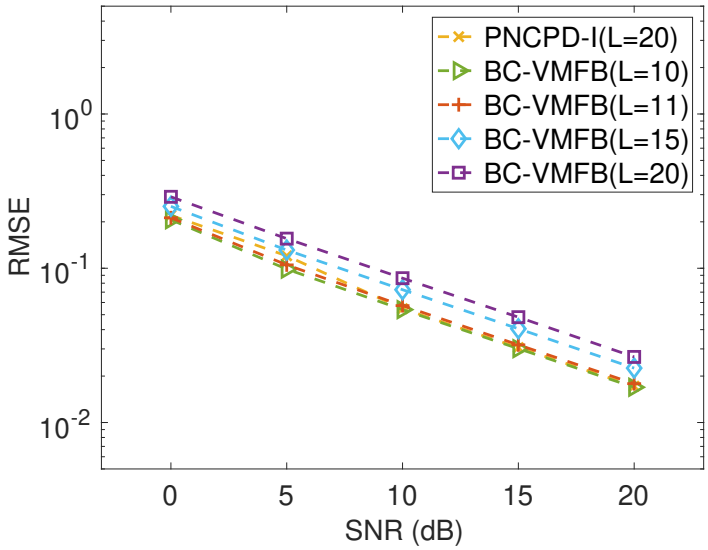

Figure 5: Estimation accuracy performance of PNCPD-I and BC-VMFB under different SNRs.

5.3. Comparisons with BC-VMFB [39]

Further comparisons are conducted with block coordinate variable metric forward-backward algorithm[2] [39] (labeled as BC-VMFB). BC-VMFB is a deterministic algorithm that promotes sparsity in factor matrices and circumvents hyper-parameter cross-validation by utilizing the discrepancy criterion.

We firstly present the estimation accuracy results in Figure 5, where L denotes the pre-selected rank upper bound. BC-VMFB achieves nearly the same estimation accuracy as PNCPD-I when L is 10 or 11, which is exactly the true tensor rank or near the true tensor rank. However, BC-VMFB does not yield comparable estimation accuracy when L is large ($L = 15$ or 20 in this case). On the other hand, PNCPD-I yields excellent estimation accuracy performance even when L is large ($L = 20$).

The inaccurate estimation of BC-VMFB in large L cases is due to its incapability to estimate the true tensor rank. To see this, we report in Fig-

[2]We greatly appreciate Prof. Caroline Chaux and Prof. Xuan Vu for sharing the code with us. The parameter settings in all experiments are the same as that of the provided code.

Table II: Average running time (in seconds) of PNCPD-I and BC-VMFB.

SNR (dB)	0	5	10	15	20
PNCPD-I ($L = 20$)	**3.0514**	**7.6629**	**5.7024**	**4.9418**	**4.0871**
BC-VMFB ($L = 20$)	8.9795	9.0328	9.0871	9.0651	7.8191
BC-VMFB ($L = 10$)	6.1816	6.2197	6.2308	6.3023	6.3844
BC-VMFB ($L = 11$)	5.8309	6.0292	6.4036	6.5674	6.4362
BC-VMFB ($L = 15$)	7.7495	7.8174	7.8259	7.7745	7.7719

ure 6 the sorted length scales of rank-1 components in one Monte Carlo trial under SNR is 20 dB, where the length scale of the l-th component is calculated as $a_l = \sum_{n=1}^{3} \left\| \mathbf{\Xi}^{(n)}_{:,l} \right\|_2^2$. BC-VMFB can identify the columns sparsity pattern when $L = 11$, but fails to uncover the pattern when L is 15 or 20. On the other hand, we present the learnt length scales of PNCPD-I for comparison. It can be seen that PNCPD-I successfully learns the tensor rank in all cases given different L. Therefore, the sparsity in BC-VMFB [39] is not strong enough to reveal the true tensor rank when the rank upper bound is too large, while PNCPD-I still yields excellent tensor rank estimation performance.

Finally, we present the running time results in Table II. It is seen that PNCPD-I achieves the best running time given $L = 20$.

6. Real-world Dataset Experiments

In this section, numerical results using two real-world datasets are presented: 1) the amino acids fluorescence excitation-emission measured (EEM) dataset[3] [36]; and 2) the ENRON Email corpus dataset[4] [7].

The first dataset $\mathcal{Y}$ is with size $5 \times 201 \times 61$, and consists of five laboratory-made samples. Each sample contains different amounts of three types of amino acids (tyrosine, tryptophan and phenylalanine) dissolved in phosphate buffered water. The samples were measured by fluorescence and were corrupted by Gaussian noise with power 0.01, resulting in SNR = 10.1 dB. The goal of EEM data analytic is to mitigate the noise corruption and recover

[3] http://www.models.life.ku.dk

[4] The original source of the data is from [7], and we greatly appreciate Prof. Vagelis Papalexakis for sharing the data with us.

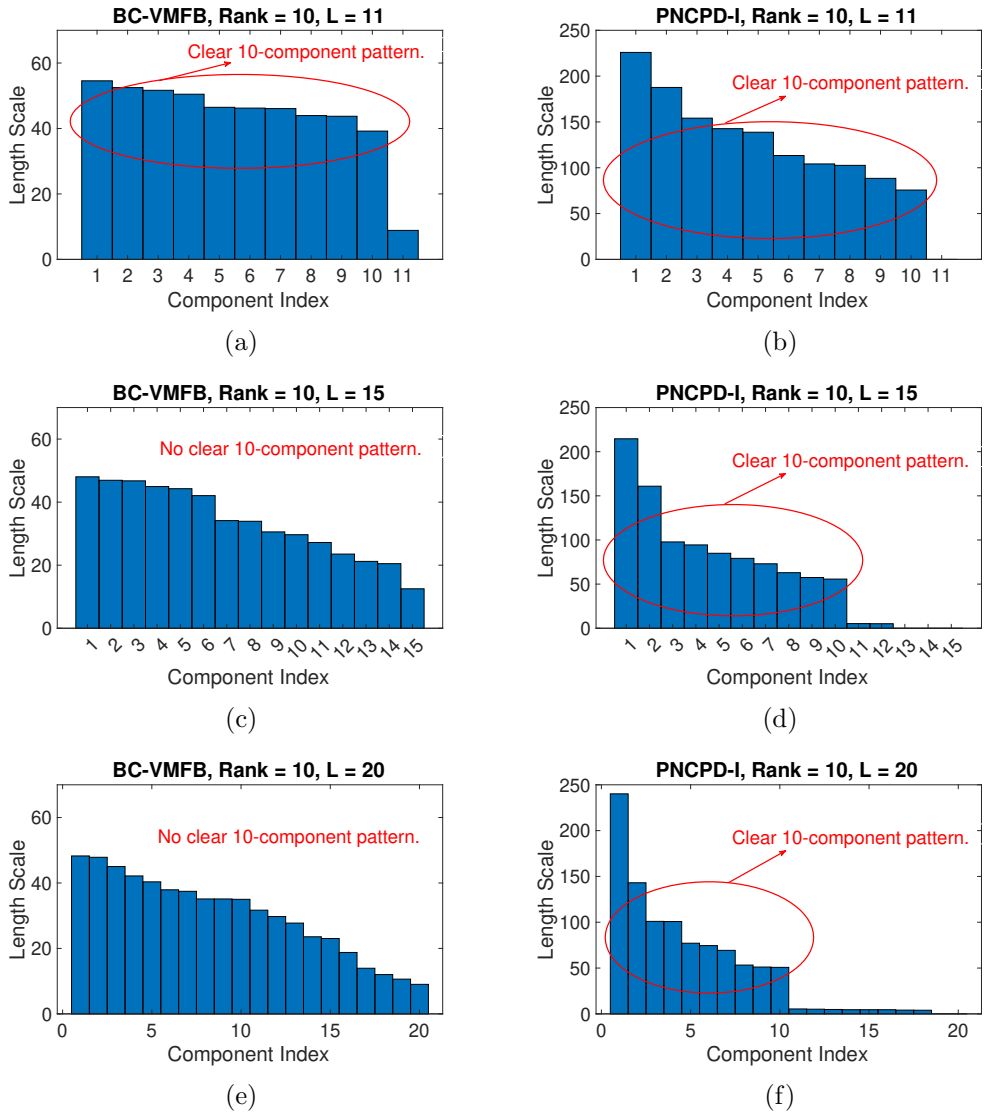

Figure 6: Tensor rank learning performance of PNCPD-I and BC-VMFB under SNR = 20 dB in one Monte Carlo trial.

the spectrum of each latent component, which has been shown to suit the model of tensor CPD with nonnegative factors. Since there are three types of amino acids (i.e., latent components), the true tensor rank is 3, which is however unknown in practice.

Assuming no knowledge of tensor rank, the proposed algorithm, the

Table II: Average running time (in seconds) of the algorithms with automatic rank determination (real-world datasets)

Dataset	EEM Dataset	ENRON Email Dataset
PNCPD	4.79	224.87
PNCPD-A	2.89	87.81
PNCPD-I	**0.70**	**11.65**

PNCPD method and the PNCPD-A method were run to decompose the EEM tensor data with initial rank $L = 5$. To assess the performance of noise mitigation, the fit value [9], which is defined as $(1 - \frac{||\hat{\mathcal{X}}-\mathcal{X}||_F}{||\mathcal{X}||_F}) \times 100\%$, is adopted, where $\hat{\mathcal{X}}$ represents the reconstructed EEM tensor data from the algorithm and $\mathcal{X}$ denotes the clean data [36]. It is found that the three algorithms all correctly estimate the true tensor rank $R = 3$ and achieve the same fit value 95.73%, while the proposed algorithm is the fastest among the three algorithms (as shown in Table II). In Figure 7, for the first slice of the tensor data (i.e., $\mathcal{X}(1, :, :) \in \mathbb{R}^{201\times61}$), the clean data, the noise-corrupted data and the reconstructed data from the proposed algorithm are presented, from which it can be seen that the proposed method mitigates the noise effectively. To see how the proposed algorithm recover the spectrum of the latent components, in Figure 8, the excitation spectra, which are obtained from the decomposed nonnegative factor matrices using the three algorithms, are depicted. It is clear that the proposed algorithm estimates nearly the same excitation spectra as those from the benchmarking algorithms. This suggests that in this application with a relatively high SNR, the proposed algorithm converges to nearly the same stationary points as those of the benchmarking algorithm.

Then, we analyze the ENRON Email corpus dataset $\mathcal{Y} \in \mathbb{R}^{184\times184\times44}$. This dataset collects the email communication records between 184 people within 44 months, in which each entry denotes the number of emails exchanged between two particular people within a particular month. By fitting this dataset into the tensor CPD models with nonnegative factors, the information of sending roles and receiving roles are encoded in factor matrices $\mathbf{\Xi}^{(1)}$ and $\mathbf{\Xi}^{(2)}$, while $\mathbf{\Xi}^{(3)}$ captures the temporal interactions between sender groups and receiver groups [17]. The nonnegative constraint naturally enhances the interpretability of these factors. Notice that $\mathbf{\Xi}^{(1)}$ and $\mathbf{\Xi}^{(2)}$ cluster the people into different groups. The number of groups, which is practically

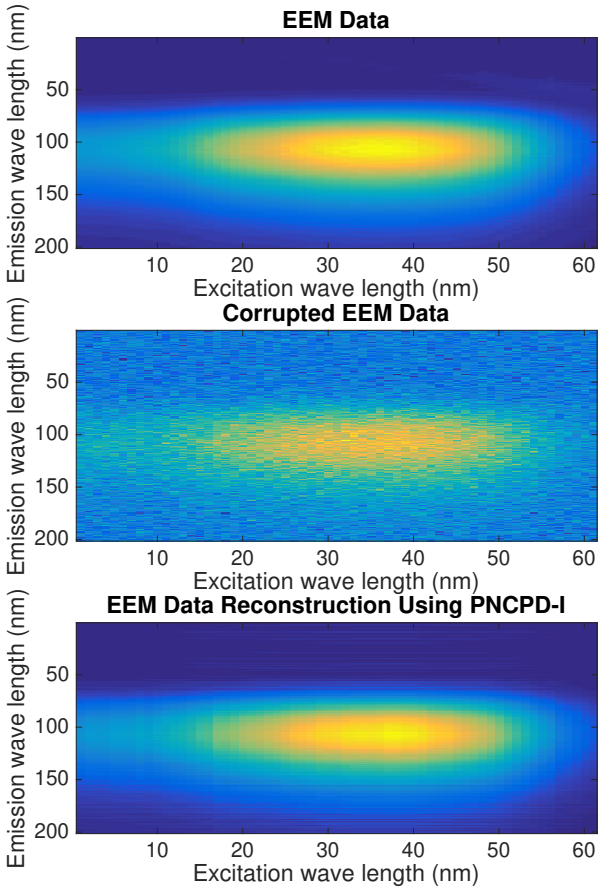

Figure 7: EEM data reconstruction using the proposed algorithm.

unknown, is specified by the tensor rank, or the number of columns in $\mathbf{\Xi}^{(1)}$ and $\mathbf{\Xi}^{(2)}$. With the initial tensor rank $L = 44$, the proposed algorithm, the PNCPD algorithm and the PNCPD-A algorithm were run to decompose the dataset. It has been found that all three algorithms estimate the tensor rank to be 4, indicating that there are 4 groups of people. This is consistent with the results from [6, 7], which are obtained via trial-and-error experiments. From Table II, it is clear that the proposed method runs much faster than the two benchmarking algorithms. Denote the model fitting error as $||\mathcal{Y} - \hat{\mathcal{X}}||_F$ where $\hat{\mathcal{X}}$ is the reconstructed tensor from the decomposed nonneg-

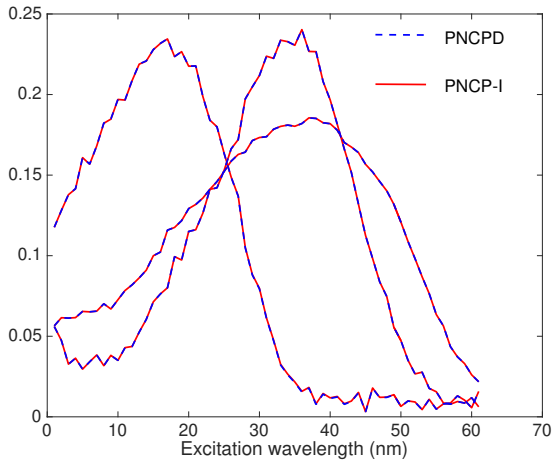

Figure 8: The estimates of excitation spectra.

Table III: Clustering results from the algorithms.

Algorithm	Clustering Groups
PNCPD & PNCPD-A	Legal; Government Affair Executive; Trading/Top Executive; Pipeline
PNCPD-I	Legal; Government Affair Executive; Trading/Top Executive; Governing Board

ative factors. The proposed algorithm is with model fitting error 345.2, while the benchmarking algorithms are with model fitting error 345.7, indicating that they converge to different stationary points. This is also reflected in the clustering results obtained from the learnt nonnegative factor matrices, as shown in Table III. More specifically, the proposed algorithm gives very similar clustering results for the first three clustering groups. But it does not render the pipeline group, and instead finds another governing board group (including the president and vice-presidents). This discrepancy suggests another reasonable interpretation for the dataset using unsupervised learning.

7. Conclusions

In this paper, probabilistic tensor CPD with nonnegative factors was investigated under the lens of equivalent deterministic optimization. In partic-

ular, it has been shown that the expectations of the variables updated by the VI-based algorithm are the same as the block minimization steps of a deterministic optimization problem. Based on this equivalence, a fast algorithm based on inexact BCD framework was further developed in this paper. Numerical results using synthetic data and real-world datasets were presented to show that there is no loss in performance of the proposed algorithm in terms of tensor recovery and tensor rank learning, while the computational time is significantly reduced.

8. Acknowledgement

This work was supported in part by the NSFC under Grant 62001309, and in part by Science and Technology on Sonar Laboratory under Grant 6142109KF212204, and in part by the General Research Fund from the Hong Kong Research Grant Council through Project 17207018.

Appendix A. Proof of Proposition 1

We prove **Proposition 1** by induction on t. Assume $\mathbf{\Theta}_k^{t-1} = \mathbb{E}_{Q^{t-1}(\mathbf{\Theta}_k)}[\mathbf{\Theta}_k]$ holds, that is, $\mathbf{\Xi}^{(n),t-1} = \mathbb{E}_{Q^{t-1}(\mathbf{\Xi}^{(n)})}[\mathbf{\Xi}^{(n)}] = \hat{\mathbf{\Xi}}^{(n),t-1}, \forall n$, $\gamma_l^{t-1} = \mathbb{E}_{Q^{t-1}(\gamma_l)}[\gamma_l] = c_l^{t-1}/d_l^{t-1}, \forall l$, $\beta^{t-1} = \mathbb{E}_{Q^{t-1}(\beta)}[\beta] = e^{t-1}/f^{t-1}$. We prove $\mathbf{\Theta}_k^t = \mathbb{E}_{Q^t(\mathbf{\Theta}_k)}[\mathbf{\Theta}_k]$ in the following.

We firstly prove $\mathbf{\Xi}^{(n),t} = \mathbb{E}_{Q^t(\mathbf{\Xi}^{(n)})}[\mathbf{\Xi}^{(n)}] = \hat{\mathbf{\Xi}}^{(n),t}, \forall n$, by induction on n. It can be shown that

$$\mathbf{\Xi}^{(1),t} = \arg\min_{\mathbf{\Xi}^{(1),t}\geq \mathbf{0}} \frac{\beta^{t-1}}{2} \parallel \mathcal{Y} - [\![\mathbf{\Xi}^{(1),t}, \mathbf{\Xi}^{(2),t-1}, \cdots, \mathbf{\Xi}^{(N),t-1}]\!]\parallel_F^2 + \frac{1}{2}\mathrm{Tr}\Big(\mathbf{\Xi}^{(1),t}\mathrm{diag}\{\gamma_1^{t-1}, ..., \gamma_L^{t-1}\}\left[\mathbf{\Xi}^{(1),t}\right]^T\Big). \tag{A.1}$$

Comparing (A.1) to the expression of $\hat{\mathbf{\Xi}}^{(1),t}$ in Algorithm 1, and utilizing the facts $\gamma_l^{t-1} = c_l^{t-1}/d_l^{t-1}, \forall l$ and $\beta^{t-1} = e^{t-1}/f^{t-1}$, it is seen that $\mathbf{\Xi}^{(1),t} = \mathbb{E}_{Q^t(\mathbf{\Xi}^{(1)})}[\mathbf{\Xi}^{(1)}]$. Now we assume $\mathbf{\Xi}^{(n-1),t} = \mathbb{E}_{Q^t(\mathbf{\Xi}^{(n-1)})}[\mathbf{\Xi}^{(n-1)}]$ for some n. Notice that

$$\mathbf{\Xi}^{(n),t} = \arg\min_{\mathbf{\Xi}^{(n),t}\geq \mathbf{0}} \frac{\beta^{t-1}}{2} \parallel \mathcal{Y} - [\![\mathbf{\Xi}^{(1),t}, \cdots, \mathbf{\Xi}^{(n),t}, \cdots, \mathbf{\Xi}^{(N),t-1}]\!]\parallel_F^2 + \frac{1}{2}\mathrm{Tr}\Big(\mathbf{\Xi}^{(n),t}\mathrm{diag}\{\gamma_1^{t-1}, ..., \gamma_L^{t-1}\}\left[\mathbf{\Xi}^{(n),t}\right]^T\Big). \tag{A.2}$$

By (A.2), $\mathbf{\Xi}^{(n),t} = \mathbb{E}_{Q^t(\mathbf{\Xi}^{(n)})}[\mathbf{\Xi}^{(n)}]$. Therefore, by induction, $\mathbf{\Xi}^{(n),t} = \mathbb{E}_{Q^t(\mathbf{\Xi}^{(n)})}[\mathbf{\Xi}^{(n)}], \forall n$.

Next, we prove $\gamma_l^t = \mathbb{E}_{Q^t(\gamma_l)}[\gamma_l] = c_l^t/d_l^t, \forall l$. From (5), the subproblem for optimizing γ_l^t is

$$\min_{\gamma_l^t} -\left(\sum_{n=1}^{N}\frac{J_n}{2}+c_l^0\right)\ln\gamma_l^t+\gamma_l^t\left(\sum_{n=1}^{N}\frac{1}{2}\left[\mathbf{\Xi}_{:,l}^{(n),t}\right]^T\mathbf{\Xi}_{:,l}^{(n),t}+d_l^0\right), \forall l. \tag{A.3}$$

Since problem (A.3) is convex, by setting its derivative to zero, the optimal solution

$$\gamma_l^t = \frac{\sum_{n=1}^{N}\frac{J_n}{2}+c_l^0}{\sum_{n=1}^{N}\frac{1}{2}\left[\mathbf{\Xi}_{:,l}^{(n),t}\right]^T\mathbf{\Xi}_{:,l}^{(n),t}+d_l^0}. \tag{A.4}$$

As shown previously $\mathbf{\Xi}^{(n),t} = \hat{\mathbf{\Xi}}^{(n),t}$, it can be concluded that $\gamma_l^t = c_l^t/d_l^t$ with the expressions of c_l^t and d_l^t given in Algorithm 1.

Similarly, the subproblem that optimizes β^t is

$$\min_{\beta^t} -\left(\frac{\prod_{n=1}^{N}J_n}{2}+e^0\right)\ln\beta^t+\beta^t\left(\frac{1}{2}\|\mathcal{Y}-[\![\mathbf{\Xi}^{(1),t},\mathbf{\Xi}^{(2),t},\cdots,\mathbf{\Xi}^{(N),t}]\!]\|_F^2+f^0\right), \tag{A.5}$$

and the optimal

$$\beta^t = \frac{\frac{\prod_{n=1}^{N}J_n}{2}+e^0}{\frac{1}{2}\|\mathcal{Y}-[\![\mathbf{\Xi}^{(1),t},\mathbf{\Xi}^{(2),t},\cdots,\mathbf{\Xi}^{(N),t}]\!]\|_F^2+f^0}. \tag{A.6}$$

It is easy to see that $\beta^t = e^t/f^t = \mathbb{E}_{Q^t(\beta)}[\beta]$.

Therefore, it can be concluded that given the same initial value (i.e., $\mathbb{E}_{Q^0(\mathbf{\Theta}_k)}[\mathbf{\Theta}_k] = \mathbf{\Theta}_k^0$) and the same block update schedule, we have $\mathbb{E}_{Q^t(\mathbf{\Theta}_k)}[\mathbf{\Theta}_k] = \mathbf{\Theta}_k^t, \forall t$, since their update equations are exactly the same. This completes the proof of **Proposition 1**.

Appendix B. Nestorov-based Acceleration

The Nestorov-based acceleration algorithm for solving (3) is summarized in Algorithm 3.

REFERENCES

Algorithm 3: Nestorov-based acceleration for solving (3) [17]

At iteration $t+1$ in Algorithm 1, for a fixed n, we need to solve $\hat{\boldsymbol{\Xi}}^{(n),t+1} = \arg\min_{\boldsymbol{\Xi}^{(n)} \geq \mathbf{0}} \xi^{t+1}(\boldsymbol{\Xi})$, where $\xi^{t+1}(\boldsymbol{\Xi})$ is given in (4). Set

$$\boldsymbol{H} = \frac{e^t}{f^t}\Big(\overset{N}{\underset{k=1,k\neq n}{\diamond}} \hat{\boldsymbol{\Xi}}^{(k),s}\Big)^T\Big(\overset{N}{\underset{k=1,k\neq n}{\diamond}} \hat{\boldsymbol{\Xi}}^{(k),s}\Big) + \mathrm{diag}\{\frac{c_1^t}{d_1^t}, ..., \frac{c_L^t}{d_L^t}\},$$
$$\boldsymbol{F} = \frac{e^t}{f^t}\Big(\overset{N}{\underset{k=1,k\neq n}{\diamond}} \hat{\boldsymbol{\Xi}}^{(k),s}\Big)\mathcal{Y}^{(n)T},$$

where s denotes the most recent update index, i.e., $s = t+1$ when $k < n$ and $s = t$ otherwise. Then, set $L = \max(\mathrm{eig}(\boldsymbol{H}))$, $\mu = \min(\mathrm{eig}(\boldsymbol{H}))$, $\boldsymbol{A}_0 = \boldsymbol{B}_0 = \hat{\boldsymbol{\Xi}}^{(n),t}$.

Iterations: For the iteration $i+1$ $(i > 0)$,

$$\boldsymbol{A}_{i+1} = \left[\boldsymbol{B}_i - \frac{1}{L}\left(\boldsymbol{B}_i\boldsymbol{H} - \boldsymbol{F}\right)\right]_+,$$
$$\boldsymbol{B}_{i+1} = \boldsymbol{A}_{i+1} + \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}\left(\boldsymbol{A}_{i+1} - \boldsymbol{A}_i\right).$$

Until Convergence
Return $\hat{\boldsymbol{\Xi}}^{(n),t+1} = \boldsymbol{A}_i$.

References

[1] N. D. Sidiropoulos, L. D. Lathauwer, X. Fu, K. Huang, E. E. Papalexakis and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551-3582, 2017.

[2] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145-163, Feb. 2015.

[3] H. Chen, and J. Li, "DrugCom: synergistic discovery of drug combinations using tensor decomposition," in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pp. 899-904, Dec. 2018.

[4] M. M. Alamdari, N. L. Dang Khoa, Y. Wang, B. Samali, and X. Zhu, "A multi-way data analysis approach for structural health monitoring of a cable-stayed bridge," *Structural Health Monitoring*, vol. 18, no. 1, pp. 35-48, Nov. 2019.

[5] R. Bro, "Review on multiway analysis in chemistry 2000-2005," *Critical Reviews in Analytical Chemistry*, vol. 35, pp. 279-293, Jan. 2006.

[6] E. E. Papalexakis, N. D. Sidiropoulos, and R. Bro, "From K-means to higher-way co-clustering: multilinear decomposition with sparse latent factors," *IEEE Transactions on Signal Processing*, vol. 61, no. 2, pp. 493-506, Jan. 2013.

[7] B. W. Bader, R. A. Harshman, and T. G. Kolda, "Temporal analysis of social network using three-way DEDICOM," Sandia National Labs, TR SAND2006-2161, 2006.

[8] W. Peng and T. Li, "Temporal relating co-clustering on directional social network and author-topic evolution," *Knowledge and Information System*, pp. 1-20, Mar. 2010.

[9] A. Cichocki, R. Zdunek, A. H. Phan and S. I. Amari, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*, John Wiley & Sons, 2009.

[10] A. Cichocki, M. Mørup, P. Smaragdis, W. Wang and R. Zdunek, *Advances in Nonnegative Matrix and Tensor Factorization*, Computational Intelligence and Neuroscience, 2008.

[11] A. Cichocki, Z. Rafal, and A. Shunichi, "New algorithms for nonnegative matrix factorization in applications to blind source separation," in *Proceeding of 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings (ICASSP)*, vol. 5, pp. 621-624. May. 2006.

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

[13] Y. Xu and W. Yin, "A block coordinate descent method for regularized multi-convex optimization with applications to nonnegative tensor factorization and completion," *SIAM Journal on Imaging Sciences*, vol. 6, no. 3, pp. 1758-1789, 2013.

[14] A. P. Liavas, G. Kostoulas, G. Lourakis, K. Huang, and N. D. Sidiropoulos, "Nesterov-based alternating optimization for nonnegative tensor factorization: algorithm and parallel implementations," *IEEE Transactions on Signal Processing*, vol. 66, no. 4, Feb. 2018.

[15] K. Huang, N. D. Sidiropoulos, and A. P. Liavas, "A flexible and efficient algorithmic framework for constrained matrix and tensor factorization," *IEEE Transactions on Signal Processing*, vol. 64, no.19, pp. 5052-5065, Oct. 2016.

[16] X. Fu, S. Ibrahim, H.-T. Wai, C. Gao, and K. Huang, "Block-randomized stochastic proximal gradient for low-rank tensor factorization", *IEEE Transactions on Signal Processing*, accepted, Mar. 2020.

[17] L. Cheng, X. Tong, S. Wang, Y.-C. Wu, and H. V. Poor, "Learning nonnegative factors from tensor data: probabilistic modeling and inference algorithm," *IEEE Transactions on Signal Processing*, vol. 68, pp. 1792-1806, 2020.

[18] D. J. MacKay, "Probable networks and plausible predictions-a review of practical Bayesian methods for supervised neural networks," *Computation in Neural Systems*, vol. 6, no. 3, pp. 469-505, 1995.

[19] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211-244, Jun. 2001.

[20] W. Guo and W. Yu, "Variational Bayesian PARAFAC decomposition for multidimensional harmonic retrieval," in *Proceedings of 2011 IEEE CIE International Conference on Radar*, vol. 2, pp.1864-1867, 2011.

[21] Q. Zhao, L. Zhang, and A. Cichocki, "Bayesian CP factorization of incomplete tensors with automatic rank determination," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1751-1763, Sep. 2015.

[22] L. Cheng, Y.-C. Wu, and H. V. Poor, "Probabilistic tensor canonical polyadic decomposition with orthogonal factors," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 663-676, Feb. 2017.

[23] L. Cheng, Y.-C. Wu, and H. V. Poor, "Scaling probabilistic tensor canonical polyadic decomposition to massive data," *IEEE Transactions on Signal Processing*, vol. 66, no. 21, pp. 5534-5548, Nov. 2018.

[24] L. Cheng, Z. Chen, Q. Shi, Y.-C. Wu and S. Theodoridis, "Towards Flexible Sparsity-Aware Modeling: Automatic Tensor Rank Learning Using the Generalized Hyperbolic Prior," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1834-1849, 2022.

[25] L. Grippo and M. Sciandrone, "On the convergence of the block nonlinear Gauss-Seidel method under convex constraints," *Operation Research Letter*, vol. 26, pp. 127–136, 2000.

[26] H. Shi, S. Tu, Y. Xu and W. Yin, "A primer on coordinate descent algorithms," 2016, arXiv preprint arXiv:1610.00040.

[27] M. Razaviyayn, M. Hong and Z.-Q. Luo, "A unified convergence analysis of block successive minimization methods for nonsmooth optimization," *SIAM Journal on Optimization*, vol. 23, no. 2, pp.1126–1153, 2013.

[28] D. P. Bertsekas, *Nonlinear Programming*, 3rd edition, Athena Scientific, 2016.

[29] P. Tseng and S. Yun, "A coordinate gradient descent method for nonsmooth separable minimization," *Mathematical Programming*, vol. 117, pp. 387–423, 2009.

[30] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Foundations and Trends in Machine Learning*, vol. 1, no. 102, pp. 1-305, Jan. 2008.

[31] C. Zhang, J. Butepage, H. Kjellstrom and S. Mandt, "Advances in variational inference," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8. pp. 2008-2026, Aug. 2019.

[32] L. Cheng, F. Yin, S. Theodoridis, S. Chatzis and T. -H. Chang, "Rethinking Bayesian Learning for Data Analysis: The art of prior and inference in sparsity-aware modeling," *IEEE Signal Processing Magazine*, vol. 39, no. 6, pp. 18-52, Nov. 2022.

[33] M. Hoffman, D. Blei, J. Paisley, and C. Wang, "Stochastic variational inference," *Journal of Machine Learning Research*, vol. 14, pp. 1303-1347, 2013.

[34] J. C. Arismendi, "Multivariate truncated moments," *Journal of Multivariate Analysis*, vol. 117, pp. 41-75, 2013.

[35] N. D. Sidiropoulos, G. B. Giannakis, and R. Bro, "Blind PARAFAC receivers for DS-CDMA systems," *IEEE Transaction on Signal Processing*, vol. 48, no. 3, pp. 810-823, Mar. 2000.

[36] H. A. L. Kiers, "A three-step algorithm for Candecomp/Parafac analysis of large data sets with multicollinearity," *Journal of Chemometrics*, vol. 12, pp. 155-171, Jun. 1998.

[37] M. J. Beal, *Variational Algorithms for Approximate Bayesian Inference*, London, University of London, 2003.

[38] C. Fernandez, and M. F. Steel, "Multivariate student-t regression models: Pitfalls and inference," *Biometrika*, vol. 86, no. 1, pp. 153-167, 1999.

[39] X. Vu, C. Chaux, N. Thirion-Moreau, S. Maire, and E. M. Carstea, "A new penalized nonnegative third-order tensor decomposition using a block coordinate proximal gradient approach: Application to 3D fluorescence spectroscopy," Journal of Chemometrics, vol. 31, no. 4, 2017.

[40] C. M. Bishop, *Pattern recognition and machine learining*, New York: springer, 2006.