Simple and Efficient Parallelization for Probabilistic Temporal Tensor Factorization

Guangxi Li, Zenglin Xu, Linnan Wang, Jinmian Ye
Big Data Res. Center, School of Com. Sci. and Engin.
University of Electronic Science and Technology of China
Chengdu, Sichuan, China
gxli@std.uestc.edu.cn;{zenglin,wangnan318,jinmian.y}@gmail.com

Irwin King, Michael Lyu
Department of Computer Science and Technology
The Chinese University of Hong Kong
Shatian, N.T., Hong Kong
{king, lyu}@cse.cuhk.edu.hk

***Abstract*—Probabilistic Temporal Tensor Factorization (PTTF) is an effective algorithm to model the temporal tensor data. It leverages a time constraint to capture the evolving properties of tensor data. Nowadays the exploding dataset demands a large scale PTTF analysis, and a parallel solution is critical to accommodate the trend. Whereas, the parallelization of PTTF still remains unexplored. In this paper, we propose a simple yet efficient *Parallel Probabilistic Temporal Tensor Factorization*, referred to as P^{2}T^{2}F, to provide a scalable PTTF solution. P^{2}T^{2}F is fundamentally disparate from existing parallel tensor factorizations by considering the probabilistic decomposition and the temporal effects of tensor data. It adopts a new tensor data split strategy to subdivide a large tensor into independent sub-tensors, the computation of which is inherently parallel. We train P^{2}T^{2}F with an efficient algorithm of stochastic Alternating Direction Method of Multipliers, and show that the convergence is guaranteed. Experiments on several real-word tensor datasets demonstrate that P^{2}T^{2}F is a highly effective and efficiently scalable algorithm dedicated for large scale probabilistic temporal tensor analysis.**

I. Introduction

Recent developments of tensor decomposition have great impacts on signal processing [1], computer vision [2], numerical analysis [3], [4], social network analysis [5], [6], recommendation systems [7], [8] and etc. A comprehensive overview can be found from the survey paper by [9]. In particular, automatic recommendation systems significantly benefit from tensor decomposition as it effectively extracts hidden patterns from the multi-way data.

Various tensor decomposition methods have been proposed. The CANDECOMP/PARAFAC decomposition, shorted as CP decomposition, is a direct extension of low-rank matrix decomposition to tensors; and it can be regarded as a special case of Tucker Decomposition by adding a super-diagonal constraint on the core tensor [10]. This method, however, fails to consider the fact that the real relational data is evolving over time and exhibits strong temporal patterns, especially in recommendation systems. To resolve this issue, *Probabilistic Temporal Tensor Factorization* (PTTF) [11], inspired by probabilistic latent factor models [12], [13], has been proposed by incorporating a time constraint. In contrast with Multi-HDP [14], Probabilistic Non-negative Tensor Factorization [15] and Probabilistic Polyadic Factorization [16], PTTF is the only one capturing the temporal effects of tensor data.

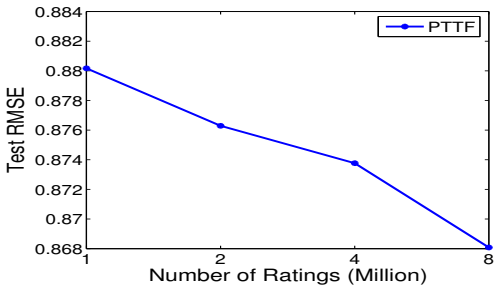


Fig. 1. The RMSE curve for PTTF on MovieLens. The prediction error reduces as the number of ratings grows.

The era of big data has also witnessed the explosion of tensor datasets, while the large scale PTTF analysis is important to accommodate the increasing datasets. Figure 1 demonstrates PTTF achieves better performance as the tensor size increases on the MovieLens data (Table I). The result directly sheds light on the necessity of a parallel PTTF solution. Nevertheless, there is a huge gap to be filled.

In this paper, we present *Parallel Probabilistic Temporal Tensor Factorization* (P^{2}T^{2}F) dedicated for large-scale temporal tensor factorization problems. The core concept of P^{2}T^{2}F is to reduce each sequential operation on a large tensor into a set of independent operations on smaller sub-tensors for parallel executions, while still retains the ability to model temporal effects of PTTF. In general, the main contributions of P^{2}T^{2}F are as follows:

- P^{2}T^{2}F allows parallel solutions to probabilistic tensor decomposition with temporal effects and thus makes PTTF model scalable. We demonstrate a new parallelization scheme in P^{2}T^{2}F to divide the large-scale problem into several sub-problems for concurrent executions.
- In P^{2}T^{2}F, we also design a novel stochastic learning algorithm for parallel ADMM framework to improve the PTTF model. Specifically, this algorithm calculates the latent feature factors using a substitutive objective function that is convex and can be viewed as an upper bound of the original problem.
- The convergence of P^{2}T^{2}F is theoretically guaranteed.

978-1-5090-6182-2/17/$31.00 ©2017 IEEE

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 03,2020 at 03:01:41 UTC from IEEE Xplore. Restrictions apply.

II. Related Work

Matrix or tensor factorization methods are useful tools in recommendation systems. One prominent representative factor-based method for recommendation systems is *Probabilistic Matrix Factorization* (PMF) [12], the latent factors of which can be learned by maximum likelihood estimation. Temporal modeling has been greatly ignored in the community of collaborative filtering until the timeSVD++ algorithm is proposed in [17]. This method demonstrates that the latent features consist of components evolving over time; and such features effectively capture local changes of user preferences. To extend the method to tensors, [11] proposes the PTTF model to capture the global effect of time shared among users and items.

The increasing demand of modeling data scalability incubates several parallel models for PMF problems, such as PPMF [18], Hogwild [19] and DSGD [20]. However, compared to large-scale matrix factorization [21], [22], there are fewer works devoted to large-scale tensors. In general, the existing large-scale tensor methods can be categorized into two classes. The first one consists in exploiting sparseness of tensors. For example, the GigaTensor algorithm in [23] and DFacTo method in [24] intend to minimize the number of floating point operations and to handle the size of intermediate data to avoid the intermediate data explosion problem, respectively. The other class of methods consists in distributing the computation load to a number of workers [25], [26], [27], [28]. Unfortunately, these algorithms do not concentrate on modeling the temporal effects of tensor data.

ADMM is an effective framework for accelerating the optimization of tensor factorization [29], [27]. However, they do not target for improving the scalability of tensor factorization algorithms; it also neglects to model the temporal effects in dynamic systems. For example, the parallel ADMM algorithm proposed in [27] computes tensor decomposition mode by mode; and it is hard to be deployed in online training or distributed training. As a response, we propose $\mathrm{P^2T^2F}$ that adapts for the online training or the distributed training. It also works in various computing environments, such as multi-core computers or clusters.

III. Preliminaries

A. Notations

A tensor is a multi-dimensional array that generalizes a vectors (1-dimensional tensor) and a matrix (2-dimensional tensor) to higher order. Like rows and columns in a matrix, an N-dimensional tensor has N modes whose lengths are I_1 through I_N, respectively. By convention, vectors and matrices are denoted by boldface lowercase letters or uppercase letters with a subscript, e.g., $\boldsymbol{a}$ or A_i, and boldface capital letters, e.g., $\boldsymbol{A}$, respectively. We denote higher-order tensors (order three or higher) by boldface Euler script letters, e.g., $\boldsymbol{\mathcal{X}}$. We also denote the entry of a tensor by the symbolic name of tensor with its indices in subscript. For example, the (i_1,i_2)th entry of $\boldsymbol{A}$ is denoted by $a_{i_1 i_2}$, and the $(i_1,\ldots,i_N)$th entry of $\boldsymbol{\mathcal{X}}$ is denoted by $x_{i_1,\ldots,i_N}$.

B. Tensor Decomposition

There are several ways to define tensor decomposition [9], [30], [31]. Our definition is based on CP (CANDECOMP/PARAFAC) decomposition, which is one of the most popular decomposition methods. Details about CP decomposition can be found in [9]. For ease of presentation, we only derive our model in the third-order case, but it can be easily generalized to N-way tensor.

Let $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$ be a third order tensor with observable entries $\{x_{ijk}|(i,j,k)\in\Omega\}$, we hope to find the factor matrices $\{\boldsymbol{A}\in\mathbb{R}^{I\times R},\boldsymbol{B}\in\mathbb{R}^{J\times R},\boldsymbol{C}\in\mathbb{R}^{K\times R}\}$ by minimizing the following loss function:

$$\min_{\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}} \quad \frac{1}{2}\sum_{(i,j,k)\in\Omega}(x_{ijk}- < A_i,B_j,C_k >)^2 + \frac{\lambda}{2}\left(\|\boldsymbol{A}\|_F^2+\|\boldsymbol{B}\|_F^2+\|\boldsymbol{C}\|_F^2\right), \tag{1}$$

where $< A_i,B_j,C_k >\equiv\sum_{r=1}^{R}a_{ir}b_{jr}c_{kr}$ denotes the inner product of three R-dimensional vectors, A_i denotes the ith row of $\boldsymbol{A}$, so does B_j and C_k. If we solve this model with *Stochastic Gradient Descent* (SGD), a drastic simplification[32], we can get a SGD-based CP decomposition model. The stochastic process depends on the examples randomly drawn at each iteration.

C. Probabilistic Temporal Tensor Factorization

Probabilistic Temporal Tensor Factorization (PTTF) model can be considered as the extension of PMF model [12] by adding a specially-constrained time dimension. For the third order tensor $\boldsymbol{\mathcal{X}}$ in (1), if the third dimension denotes the time corresponding to the factor matrix $\boldsymbol{C}$, we assume the following conditional prior for $\boldsymbol{C}$ [11]:

$$C_k\sim\mathcal{N}(C_{k-1},\sigma_C^2\boldsymbol{I}_R),\qquad k=1,\ldots,K.$$

For the initial time feature vector C_0, we assume

$$C_0\sim\mathcal{N}(\boldsymbol{\mu}_C,\sigma_0^2\boldsymbol{I}_R),$$

where $\boldsymbol{\mu}_C$ denotes a 1-by-R row vector and $\boldsymbol{I}_R$ denotes a R-by-R identity matrix. The PTTF model can be expressed to minimizing the following regularized sum of squared errors (the proof can be seen in [11]):

$$\frac{1}{2}\sum_{(i,j,k)\in\Omega}(x_{ijk}- < A_i,B_j,C_k >)^2+\frac{\lambda_0}{2}\|C_0-\mu_C\|_2^2 + \frac{\lambda_A}{2}\|\boldsymbol{A}\|_F^2+\frac{\lambda_B}{2}\|\boldsymbol{B}\|_F^2+\frac{\lambda_C}{2}\sum_{k=1}^{K}\|C_k-C_{k-1}\|_2^2, \tag{2}$$

where $\lambda_A,\lambda_B,\lambda_C,\lambda_0$ denote regularization parameters.

Obviously, we can also get a SGD-based PTTF model using SGD method, but it can not be easily parallelized because of the special constraint on the time dimension. In contrast, ADMM framework has the properties of flexibility and tractability and, the most important, can be naturally used to design parallel or distributed learning algorithms for large-scale problems. So, in the next section, we will address the

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 03,2020 at 03:01:41 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 Tensor Data Split

Input: Tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$,$P$.
Output: $\mathcal{X}^p \in \mathbb{R}^{\frac{I}{P}\times J\times K}$.

1) initialize $I_{first} = 0$;
2) **for** $p = 1, \ldots, P$ **do**
3) $\quad I_{last} = \lfloor \frac{pI}{P} \rfloor$;
4) $\quad \mathcal{X}^p \leftarrow \mathcal{X}_{(I_{first}+1:I_{last}),:,:}$;
5) $\quad I_{first} = I_{last}$;
6) return $\mathcal{X}^p$.

minimization problem in (2) in the framework of ADMM. In recent years, ADMM has occupied more and more attention and wide range of applications such as matrix completion [18], [33] and compressive sensing [34], but as far as we know, no works have yet been proposed to use parallel ADMM framework for probabilistic temporal tensor decomposition problems.

IV. Parallel Probabilistic Temporal Tensor Factorization

In this section, we elaborate *Parallel Probabilistic Temporal Tensor Factorization* (P^{2}T^{2}F). First, we introduce a new data split strategy to divide the whole tensor data into several sub-tensors, and we meticulously reduce a large tensor operation to a set of independent operations toward sub-tensors allowing for concurrent executions. We also extend ADMM to handle these sub-tensors in the training.

A. Data Split Strategy

The costs of tensor decomposition are closely contingent upon tensor sizes, it is natural for us to split a large tensor data into several independent sub-tensors. In general, we divide the tensor $\mathcal{X}$ into P sub-blocks (or sub-tensors) along the mode with the most dimensions (assuming the first mode without loss of the generality). In this case, each sub-block contains $\frac{I}{P}$ horizontal slices (e.g., $\mathcal{X}^p \in \mathbb{R}^{\frac{I}{P}\times J\times K}$, $p = 1, 2, \ldots, P$).

Assuming there are P corresponding local factor matrices for each mode denoted as $\boldsymbol{A}^p$, $\boldsymbol{B}^p$ and $\boldsymbol{C}^p$, respectively. Please note that $\boldsymbol{B}^p$ and $\boldsymbol{C}^p$ share the same size with $\boldsymbol{B}$ and $\boldsymbol{C}$. To better utilize the global variable consensus optimization method, we have a global item latent matrix denoted as $\overline{\boldsymbol{B}}$ and a global time latent matrix denoted as $\overline{\boldsymbol{C}}$. Since the local matrices $\boldsymbol{B}^p$ and $\boldsymbol{C}^p$ are only coupled with $\boldsymbol{A}^p$, it is feasible to independently update $\boldsymbol{A}^p$, $\boldsymbol{B}^p$ and $\boldsymbol{C}^p$ for each process. This split strategy enables the CP decomposition problem to fit in the parallel ADMM framework.

Algorithm 1 demonstrates the details of proposed tensor data split strategy. Please note that it is also possible to divide a 3-order tensor into P sub-blocks simultaneously along two modes, yet the approach is subject to significant complex loss functions and constraints.

B. Parallel Probabilistic Temporal Tensor Factorization Model

In this split setting, the minimization problem (e.g., PTTF model) in (2) can be reformulated as the following constrained optimization problem:

$$\begin{aligned}\min_{\substack{\boldsymbol{A}^p,\boldsymbol{B}^p,\boldsymbol{C}^p,\\ C_0^p,\overline{\boldsymbol{B}},\overline{\boldsymbol{C}}}} & \sum_{p=1}^{P}\left[f\left(\boldsymbol{A}^p,\boldsymbol{B}^p,\boldsymbol{C}^p\right)+g\left(\boldsymbol{A}^p,\boldsymbol{B}^p,\boldsymbol{C}^p,C_0^p\right)\right]\\ \text{s.t.}\quad & \boldsymbol{B}^p-\overline{\boldsymbol{B}}=\boldsymbol{0},\\ & \boldsymbol{C}^p-\overline{\boldsymbol{C}}=\boldsymbol{0};\forall p\in\{1,2,\ldots,P\}.\end{aligned} \tag{3}$$

where

$$f\left(\boldsymbol{A}^p,\boldsymbol{B}^p,\boldsymbol{C}^p\right)=\frac{1}{2}\sum_{(i,j,k)\in\Omega^p}(x_{ijk}^p-<A_i^p,B_j^p,C_k^p>)^2,$$

$$\begin{aligned}g(\boldsymbol{A}^p,\boldsymbol{B}^p,\boldsymbol{C}^p,C_0^p)=&\frac{\lambda_A}{2}\|\boldsymbol{A}^p\|_F^2+\frac{\lambda_B}{2}\|\boldsymbol{B}^p\|_F^2\\&+\frac{\lambda_C}{2}\sum_{k=1}^{K}\|C_k^p-C_{k-1}^p\|_2^2+\frac{\lambda_0}{2}\|C_0^p-\mu_C\|_2^2.\end{aligned}$$

Here, Ω^p denotes the (i,j,k) indices of the values located in process p. $\overline{\boldsymbol{B}}$ and $\overline{\boldsymbol{C}}$ denote the global factor matrices. If we want to generalize it to a N-way tensor ($N > 3$), we add additional constraints.

We transform the constrained optimization problem in (3) to an unconstrained problem with Augmented Lagrangian Method, and yield the following local objective function:

$$\begin{aligned}&L^p(\boldsymbol{A}^p,\boldsymbol{B}^p,\boldsymbol{C}^p,C_0^p,\boldsymbol{\Theta}_B^p,\boldsymbol{\Theta}_C^p,\overline{\boldsymbol{B}},\overline{\boldsymbol{C}})=f\left(\boldsymbol{A}^p,\boldsymbol{B}^p,\boldsymbol{C}^p\right)\\&+g\left(\boldsymbol{A}^p,\boldsymbol{B}^p,\boldsymbol{C}^p,C_0^p\right)+l(\boldsymbol{B}^p,\boldsymbol{C}^p,\boldsymbol{\Theta}_B^p,\boldsymbol{\Theta}_C^p,\overline{\boldsymbol{B}},\overline{\boldsymbol{C}}),\end{aligned} \tag{4}$$

where

$$\begin{aligned}&l(\boldsymbol{B}^p,\boldsymbol{C}^p,\boldsymbol{\Theta}_B^p,\boldsymbol{\Theta}_C^p,\overline{\boldsymbol{B}},\overline{\boldsymbol{C}})\\=\;&\operatorname{tr}\left([\boldsymbol{\Theta}_B^p]^\top(\boldsymbol{B}^p-\overline{\boldsymbol{B}})\right)+(\rho_B/2)\|\boldsymbol{B}^p-\overline{\boldsymbol{B}}\|_F^2\\&+\operatorname{tr}\left([\boldsymbol{\Theta}_C^p]^\top(\boldsymbol{C}^p-\overline{\boldsymbol{C}})\right)+(\rho_C/2)\|\boldsymbol{C}^p-\overline{\boldsymbol{C}}\|_F^2.\end{aligned}$$

Here, $\boldsymbol{\Theta}_B^p$ and $\boldsymbol{\Theta}_C^p$ denote the Lagrangian multipliers, ρ_B and ρ_C are the penalty parameters.

The global objective function is then as follows:

$$\begin{aligned}&L(\boldsymbol{\mathcal{A}},\boldsymbol{\mathcal{B}},\boldsymbol{\mathcal{C}},\mathcal{C}_0,\boldsymbol{\Theta}_B,\boldsymbol{\Theta}_C,\overline{\boldsymbol{B}},\overline{\boldsymbol{C}})\\=\;&\sum_{p=1}^{P}L^p(\boldsymbol{A}^p,\boldsymbol{B}^p,\boldsymbol{C}^p,C_0^p,\boldsymbol{\Theta}_B^p,\boldsymbol{\Theta}_C^p,\overline{\boldsymbol{B}},\overline{\boldsymbol{C}}),\end{aligned} \tag{5}$$

where, $\boldsymbol{\mathcal{A}} = \{\boldsymbol{A}^p\}_{p=1}^P$, $\boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}, \mathcal{C}_0, \boldsymbol{\Theta}_B$ and $\boldsymbol{\Theta}_C$ are similarly defined. The ADMM method will solve this problem by repeating the following steps:

$$\boldsymbol{A}_{t+1}^p,\boldsymbol{B}_{t+1}^p,\boldsymbol{C}_{t+1}^p,(C_0^p)_{t+1}\leftarrow\operatorname*{argmin}_{\boldsymbol{A}^p,\boldsymbol{B}^p,\boldsymbol{C}^p,C_0^p}F_{local}, \tag{6}$$

$$\overline{\boldsymbol{B}}_{t+1},\overline{\boldsymbol{C}}_{t+1}\leftarrow\operatorname*{argmin}_{\overline{\boldsymbol{B}},\overline{\boldsymbol{C}}}F_{global}, \tag{7}$$

$$(\boldsymbol{\Theta}_B^p)_{t+1}\leftarrow(\boldsymbol{\Theta}_B^p)_t+\rho_B(\boldsymbol{B}_{t+1}^p-\overline{\boldsymbol{B}}_{t+1}), \tag{8}$$

$$(\boldsymbol{\Theta}_C^p)_{t+1}\leftarrow(\boldsymbol{\Theta}_C^p)_t+\rho_C(\boldsymbol{C}_{t+1}^p-\overline{\boldsymbol{C}}_{t+1}), \tag{9}$$

$$\forall p\in\{1,2,\ldots,P\},$$

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 03,2020 at 03:01:41 UTC from IEEE Xplore. Restrictions apply.

where

$$F_{local} = L^p(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p, C_0^p, (\boldsymbol{\Theta}_B^p)_t, (\boldsymbol{\Theta}_C^p)_t, \overline{\boldsymbol{B}}_t, \overline{\boldsymbol{C}}_t),$$
$$F_{global} = \sum_{p=1}^{P} l(\boldsymbol{B}_{t+1}^p, \boldsymbol{C}_{t+1}^p, (\boldsymbol{\Theta}_B^p)_t, (\boldsymbol{\Theta}_C^p)_t, \overline{\boldsymbol{B}}, \overline{\boldsymbol{C}}).$$

These update rules suggest that $\boldsymbol{A}^p$, $\boldsymbol{B}^p$, $\boldsymbol{C}^p$, C_0^p, $\boldsymbol{\Theta}_B^p$ and $\boldsymbol{\Theta}_C^p$ can be locally updated in an independent process. In this case, we dissect the whole tensor factorization problem into P independent sub-problems allowing for concurrent executions. Since we solve the problem under the ADMM framework, P^{2}T^{2}F can be viewed as a parallel extension of ADMM applied in PTTF problem.

C. Stochastic Learning Algorithm for P^{2}T^{2}F

To learn the parameters in (3), we need to solve the above several steps from (6) to (9). If the optimal $\boldsymbol{A}^p$, $\boldsymbol{B}^p$ and $\boldsymbol{C}^p$ have been obtained, it is easy to calculate C_0^p, $\overline{\boldsymbol{B}}$ and $\overline{\boldsymbol{C}}$. By setting the partial derivative of L^p w.r.t C_0^p to zero, we acquire the update rule of C_0^p:

$$C_0^p \leftarrow \frac{1}{\lambda_0 + \lambda_C} \left(\lambda_C C_1^p + \lambda_0 \mu_C\right). \tag{10}$$

Since $\overline{\boldsymbol{B}}$ is a global variable, we need to take the partial derivative of L in (5) w.r.t $\overline{\boldsymbol{B}}$, and set the derivative to zero, then we can get $\overline{\boldsymbol{B}}$. If we set $(\boldsymbol{\Theta}_B^p)_0 = \boldsymbol{0}, p = 1, \ldots, P$, we can prove that $\sum_{p=1}^{P} (\boldsymbol{\Theta}_B^p)_t = \boldsymbol{0}, t = 1, 2, \ldots$. Then, the update rules for $\overline{\boldsymbol{B}}$ and $\overline{\boldsymbol{C}}$ ($\overline{\boldsymbol{C}}$ is similar to $\overline{\boldsymbol{B}}$) can be concisely written as:

$$\overline{\boldsymbol{B}} \leftarrow \frac{1}{P} \sum_{p=1}^{P} \boldsymbol{B}^p, \qquad \overline{\boldsymbol{C}} \leftarrow \frac{1}{P} \sum_{p=1}^{P} \boldsymbol{C}^p. \tag{11}$$

$\boldsymbol{\Theta}_B^p$ and $\boldsymbol{\Theta}_C^p$ can be directly updated by formulae (8) and (9). Therefore, how to efficiently compute the factor matrices becomes the key learning part. In the following content of this subsection, we will design a stochastic learning algorithm to solve it.

1) Batch Learning: The update step in (6) is actually a PTTF problem with $\boldsymbol{\Theta}_B^p$, $\boldsymbol{\Theta}_C^p$, $\overline{\boldsymbol{B}}$ and $\overline{\boldsymbol{C}}$ fixed. Since $\boldsymbol{A}^p$, $\boldsymbol{B}^p$ and $\boldsymbol{C}^p$ are coupled together and the objective function of the PTTF problem is non-convex, it is not easy to get a satisfied solution. We employ a technique similar to that in [35] to resolve this issue by constructing a substitutive objective function, where the factor matrices $\boldsymbol{A}^p$, $\boldsymbol{B}^p$ and $\boldsymbol{C}^p$ are decoupled and can be simultaneously calculated. The convexity of the constructed function, in each iteration, enables us to get the analytical solution of $\boldsymbol{A}^p$, $\boldsymbol{B}^p$ and $\boldsymbol{C}^p$ by setting their gradients to zero.

The substitutive objective function is defined as follows:

$$\begin{aligned} &H^p(\boldsymbol{\mathcal{M}}^p, (C_0^p)_t, (\boldsymbol{\Theta}_B^p)_t, (\boldsymbol{\Theta}_C^p)_t, \overline{\boldsymbol{B}}_t, \overline{\boldsymbol{C}}_t, \tau_t | \boldsymbol{\mathcal{M}}_t^p) \\ =&h\left(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p, \tau_t | \boldsymbol{A}_t^p, \boldsymbol{B}_t^p, \boldsymbol{C}_t^p\right) + g\left(\boldsymbol{\mathcal{M}}^p, (C_0^p)_t\right) \\ &+l(\boldsymbol{B}^p, \boldsymbol{C}^p, (\boldsymbol{\Theta}_B^p)_t, (\boldsymbol{\Theta}_C^p)_t, \overline{\boldsymbol{B}}_t, \overline{\boldsymbol{C}}_t), \end{aligned} \tag{12}$$

where

$$\begin{aligned} &h(\boldsymbol{\mathcal{M}}^p, \tau_t | \boldsymbol{\mathcal{M}}_t^p) = f(\boldsymbol{\mathcal{M}}_t^p) + \mathrm{tr}[\nabla_{\boldsymbol{A}^p}^\top f(\boldsymbol{\mathcal{M}}_t^p)(\boldsymbol{A}^p - \boldsymbol{A}_t^p)] \\ &+\mathrm{tr}[\nabla_{\boldsymbol{B}^p}^\top f(\boldsymbol{\mathcal{M}}_t^p)(\boldsymbol{B}^p - \boldsymbol{B}_t^p)] + \mathrm{tr}[\nabla_{\boldsymbol{C}^p}^\top f(\boldsymbol{\mathcal{M}}_t^p)(\boldsymbol{C}^p - \boldsymbol{C}_t^p)] \\ &+\frac{1}{2\tau_t}(\|\boldsymbol{A}^p - \boldsymbol{A}_t^p\|_F^2 + \|\boldsymbol{B}^p - \boldsymbol{B}_t^p\|_F^2 + \|\boldsymbol{C}^p - \boldsymbol{C}_t^p\|_F^2). \end{aligned} \tag{13}$$

Here, $\boldsymbol{\mathcal{M}}^p = \{\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p\}$, $\boldsymbol{\mathcal{M}}_t^p = \{\boldsymbol{A}_t^p, \boldsymbol{B}_t^p, \boldsymbol{C}_t^p\}$ are used for simple writing, τ_t is a value that will be related to the learning rate and $f(\boldsymbol{\mathcal{M}}_t^p)$ is already defined in (3).

Theorem 1. *Let* $D = \{\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p | \|A_i^p - (A_i^p)_t\|_2^2 \le \delta^2, \|B_j^p - (B_j^p)_t\|_2^2 \le \delta^2, \|C_k^p - (C_k^p)_t\|_2^2 \le \delta^2\}$, $\delta^2 > 0$. *Then,* $\forall \boldsymbol{\mathcal{M}}^p \in D$, *a suitable* τ_t *can always be found to make* $H^p(\cdot)$ *satisfy the following two properties:*

$$\begin{aligned} &H^p(\boldsymbol{\mathcal{M}}^p, (C_0^p)_t, (\boldsymbol{\Theta}_B^p)_t, (\boldsymbol{\Theta}_C^p)_t, \overline{\boldsymbol{B}}_t, \overline{\boldsymbol{C}}_t, \tau_t | \boldsymbol{\mathcal{M}}_t^p) \\ \ge &L^p(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p, (C_0^p)_t, (\boldsymbol{\Theta}_B^p)_t, (\boldsymbol{\Theta}_C^p)_t, \overline{\boldsymbol{B}}_t, \overline{\boldsymbol{C}}_t), \\ &H^p(\boldsymbol{\mathcal{M}}_t^p, (C_0^p)_t, (\boldsymbol{\Theta}_B^p)_t, (\boldsymbol{\Theta}_C^p)_t, \overline{\boldsymbol{B}}_t, \overline{\boldsymbol{C}}_t, \tau_t | \boldsymbol{\mathcal{M}}_t^p) \\ = &L^p(\boldsymbol{A}_t^p, \boldsymbol{B}_t^p, \boldsymbol{C}_t^p, (C_0^p)_t, (\boldsymbol{\Theta}_B^p)_t, (\boldsymbol{\Theta}_C^p)_t, \overline{\boldsymbol{B}}_t, \overline{\boldsymbol{C}}_t). \end{aligned}$$

The proof of Theorem 1 can be found in the supplemental material. From Theorem 1, we can find that $H^p(\cdot)$ is an upper bound of $L^p(\cdot)$, and $H^p(\cdot) = L^p(\cdot)$ at the point $(\boldsymbol{A}_t^p, \boldsymbol{B}_t^p, \boldsymbol{C}_t^p)$. Fortunately, $H^p(\cdot)$ is convex in $(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p)$, and $\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p$ are decoupled in $H^p(\cdot)$. Hence, we can optimize the easily-solved constructed function $H^p(\cdot)$ instead of $L^p(\cdot)$ in (6) by setting the gradients to zero, the optimal results are computed as follows:

$$\boldsymbol{A}^p \leftarrow \frac{1}{1 + \lambda_A \tau_t} [\boldsymbol{A}_t^p - \tau_t * \nabla_{\boldsymbol{A}^p} f(\boldsymbol{\mathcal{M}}_t^p)], \tag{14}$$

$$\begin{aligned} \boldsymbol{B}^p \leftarrow &\frac{1}{1/\tau_t + \lambda_B + \rho_B} \big[\boldsymbol{B}_t^p/\tau_t + \rho_B \overline{\boldsymbol{B}}_t \\ &- (\boldsymbol{\Theta}_B^p)_t - \nabla_{\boldsymbol{B}^p} f(\boldsymbol{A}_t^p, \boldsymbol{B}_t^p, \boldsymbol{C}_t^p)\big], \end{aligned} \tag{15}$$

$$\begin{aligned} \boldsymbol{C}^p \leftarrow &\boldsymbol{Q}^{-1} \big[\boldsymbol{C}_t^p/\tau_t + \rho_C \overline{\boldsymbol{C}}_t + \lambda_C \boldsymbol{S}_C \\ &- (\boldsymbol{\Theta}_C^p)_t - \nabla_{\boldsymbol{C}^p} f(\boldsymbol{A}_t^p, \boldsymbol{B}_t^p, \boldsymbol{C}_t^p)\big], \end{aligned} \tag{16}$$

where $\boldsymbol{S}_C = \begin{bmatrix} (C_0^p)_t \\ \boldsymbol{0} \end{bmatrix}$ is a K-by-R matrix and $\boldsymbol{Q} = (1/\tau_t + \rho_C)\boldsymbol{I}_K + \lambda_C \boldsymbol{S}$ is a K-by-K matrix. Here, $\boldsymbol{S}$ denotes a coefficient matrix which shows relationships on the time dimension. Actually, $\boldsymbol{S}$ is a tridiagonal matrix and can be described as follows:

$$\boldsymbol{S} = \begin{bmatrix} 2\boldsymbol{I}_{K-1} & \boldsymbol{0} \\ \boldsymbol{0} & 1 \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} & -\boldsymbol{I}_{K-1} \\ 0 & \boldsymbol{0} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} & 0 \\ -\boldsymbol{I}_{K-1} & \boldsymbol{0} \end{bmatrix}.$$

A batch learning algorithm for the problem in (3) can be got by combining (8)-(11) and (14)-(16).

Theorem 2. *The batch learning algorithm enables P^{2}T^{2}F to converge.*

Proof. From Theorem 1, we can get

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 03,2020 at 03:01:41 UTC from IEEE Xplore. Restrictions apply.

$$
\begin{aligned}
&L^p(\mathcal{M}^p_{t+1}, (C^p_0)_t, (\Theta^p_B)_t, (\Theta^p_C)_t, \overline{B}_t, \overline{C}_t) \\
\leq &H^p(\mathcal{M}^p_{t+1}, (C^p_0)_t, (\Theta^p_B)_t, (\Theta^p_C)_t, \overline{B}_t, \overline{C}_t, \tau_t | \mathcal{M}^p_t) \\
\leq &H^p(\mathcal{M}^p_t, (C^p_0)_t, (\Theta^p_B)_t, (\Theta^p_C)_t, \overline{B}_t, \overline{C}_t, \tau_t | \mathcal{M}^p_t) \\
= &L^p(A^p_t, B^p_t, C^p_t, (C^p_0)_t, (\Theta^p_B)_t, (\Theta^p_C)_t, \overline{B}_t, \overline{C}_t).
\end{aligned}
$$

That is to say, the global objective function $L(\cdot)$ in (5) will not increase in each iteration. Furthermore, $L(\cdot)$ is non-convex and has the lower bound $-\frac{\sum_{p=1}^P \|\Theta^p_B\|_F^2}{2\rho_B} - \frac{\sum_{p=1}^P \|\Theta^p_C\|_F^2}{2\rho_C}$. Hence, our batch learning algorithm will converge. □

2) Stochastic Learning: From the batch learning algorithm, we can find that the update rules for $\mathcal{M}^p$ will need all ratings related to $\mathcal{M}^p$. If the number of ratings become very large, the batch learning algorithm will not be efficient. So, we propose a stochastic learning algorithm to further improve the efficiency, and the update rules for $\mathcal{M}^p$ are as follows:

$$
\begin{aligned}
A^p_i \leftarrow &\frac{(A^p_i)_t + \tau_t \epsilon_{ijk}((B^p_j)_t * (C^p_k)_t)}{1 + \lambda_A \tau_t}, \\
B^p_j \leftarrow &\frac{1}{1/\tau_t + \lambda_B + \rho_B} \big[(B^p_j)_t/\tau_t + \rho_B(\overline{B}_j)_t \\
&- ((\Theta^p_B)_j)_t + \epsilon_{ijk}((A^p_i)_t * (C^p_k)_t)\big], \\
C^p_k \leftarrow &\frac{1}{1/\tau_t + 2\lambda_C + \rho_C} \big[(C^p_k)_t/\tau_t + \rho_C(\overline{C}_k)_t \\
&+ \lambda_C((C^p_{k-1})_t + (C^p_{k+1})_t) \\
&- ((\Theta^p_C)_k)_t + \epsilon_{ijk}((A^p_i)_t * (B^p_j)_t)\big], \qquad (17)
\end{aligned}
$$

where $\epsilon_{ijk} = x^p_{ijk} - < (A^p_i)_t, (B^p_j)_t, (C^p_k)_t >$. Hence, the stochastic learning algorithm is a variant of the batch learning algorithm.

By combining the tensor data split strategy and the stochastic update rules stated above, we get a stochastic learning algorithm for our P^{2}T^{2}F model. The whole procedure of P^{2}T^{2}F is briefly listed in Algorithm 2, where $\boldsymbol{A} = [(\boldsymbol{A}^1)^\top \ (\boldsymbol{A}^2)^\top \cdots (\boldsymbol{A}^P)^\top]^\top$. Note that the convergence criterion is met when the difference between the train RMSEs of two successive iterations less than some threshold, e.g., 10^{-4}.

D. Complexity Analysis

P^{2}T^{2}F mainly needs two steps to update all variables once. The first step updates $\boldsymbol{A}^p$, $\boldsymbol{B}^p$ and $\boldsymbol{C}^p$. For each value x_{ijk}, the time complexity of update A^p_i, B^p_j and C^p_k is $O(R)$. Because the total number of observed entries in each process is about $|\Omega|/P$, the time complexity of step one is $O(|\Omega|R/P)$. The second step needs to update a matrix of size $J \times R$ and a matrix of size $K \times R$ in each process, so the time complexity is $O(\max\{J, K\}R)$. In total, the time complexity of P^{2}T^{2}F for each iteration can come down to $O(|\Omega|R/P + \max\{J, K\}R)$.

V. Experimental Results

Our experiments are designed to study the accuracy and efficiency of the P^{2}T^{2}F and baselines on the publicly available real-word datasets. All the experiments are run on a 12-core server with 2.60GHz Intel(R) Xeon(R) E5-2630 processor and 64GB of RAM.

Algorithm 2 Our P^{2}T^{2}F model

Input: Tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$, Rank R, $MaxIter$, P.
Output: $\boldsymbol{A} \in \mathbb{R}^{I \times R}, \overline{\boldsymbol{B}} \in \mathbb{R}^{J \times R}, \overline{\boldsymbol{C}} \in \mathbb{R}^{K \times R}$.

```
use Algorithm 1 to get X^p;
initialize λ_A, λ_B, λ_C, λ_0, ρ_B, ρ_C, A^p, B^p, C^p, C^p_0, μ_C;
set Θ^p_B (and Θ^p_C)= 0,    p = 1, 2, ..., P;
calculate B̄, C̄ by (11);
for iter = 1, 2, ..., MaxIter do
    for p = 1, 2, ..., P parallel do
        update C^p_0 by (10);
        for each x_ijk in process p do
            update A^p_i, B^p_j, C^p_k by (17);
    update B̄, C̄ by (11);
    for p = 1, 2, ..., P parallel do
        update Θ^p_B, Θ^p_C by (8) and (9);
    if convergence criterion is met then
        break;
    update τ_t;
return A, B̄, C̄.
```

TABLE I
Summary of real-world datasets

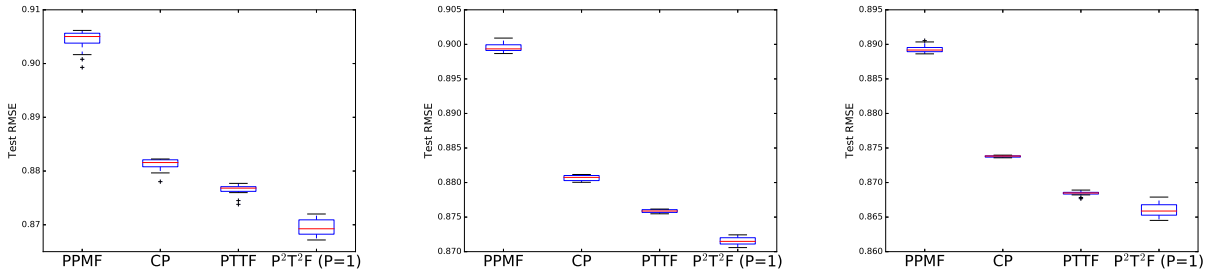
	S1	S2	S3
I_1	14,012	28,060	56,361
I_2	19,527	24,981	28,444
I_3	242	242	242
$\#Train$	1,851,291	3,739,047	7,566,903
$\#Test$	205,699	415,449	840,766

a) Datasets and Parameter Settings: The real-word tensor data used in our experiments are public collaborative filtering datasets: Movielens ml-latest [1], which is movie rating data from MovieLens, an online movie recommender service. In order to study P^{2}T^{2}F's parallel performance, we process it into three 3-order tensors, where each mode correspond to users, movies and calendar month, respectively, with the restriction of 20 ratings per user at a minimum. The rates range from 0.5 to 5, and the details are summarized in Table I. We set $\lambda_A = \lambda_B = \lambda_C = 0.01$ and $\rho_B = \rho_C = 0.5$. Since it is difficult to compute the exact value for τ_t, we approximately update it as $\tau_{t+1} = \tau_t * \beta (0 < \beta < 1)$ for the t-th iteration. We also set a threshold α. When $\tau_t \leq \alpha$, we stop decreasing τ_t. We set $\tau_0 = 0.0005$, $\beta = 0.9$ and $\alpha = 0.0001$. The hyperparameters are all determined by cross validation. We choose $R = 20$ here.

b) RMSE and Efficiency: We use the root mean squared error (RMSE) to evaluate our P^{2}T^{2}F model and baselines and their cocnvergence criteria are the same. We first examine the significance of the improvement of P^{2}T^{2}F over the CP, PTTF and PPMF model on these datasets in one core by repeating the prediction tasks 12 times using different random

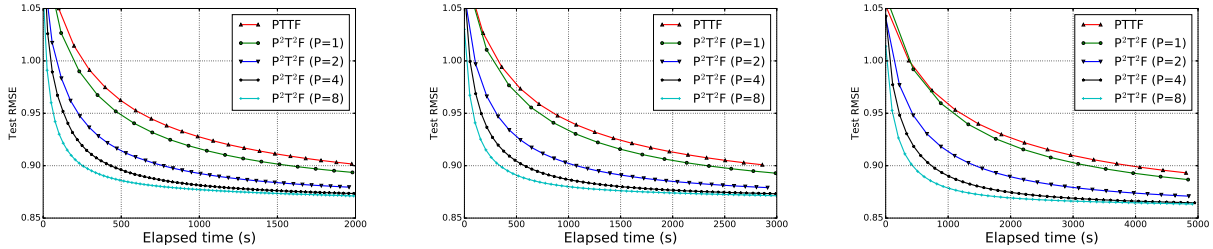
[1] http://grouplens.org/datasets/movielens/

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 03,2020 at 03:01:41 UTC from IEEE Xplore. Restrictions apply.



(a) S1 (b) S2 (c) S3

Fig. 2. Box plot of the RMSEs from PPMF, CP, PTTF and P^{2}T^{2}F on three datasets. P^{2}T^{2}F can outperform others in one core.



(a) S1 (b) S2 (c) S3

Fig. 3. The test RMSE curves for PTTF and P^{2}T^{2}F with different P on three datasets.

initializations. Figure 2 shows the resulting box plot of test RMSEs on S1-S3 at the moment when the convergence criteria of all these methods are satisfied. We can see that P^{2}T^{2}F model outperforms the baselines in all runs. P^{2}T^{2}F takes advantages of the stochastic learning algorithm, where it handles the latent factors using a surrogate objective function and makes them decoupled and easily computed. Therefore, it reaches a better RMSE than the conventional SGD-based PTTF method. In particular, the PTTF model outperforms CP model because of the temporal effects in the probabilistic decomposition. The PPMF model only considers the users and movies get the worst result. CP and PTTF model have a smaller degree of dispersion because they have fewer parameters than PPMF and P^{2}T^{2}F methods. Therefore, they are less sensitive random initializations.

c) Scalability: Another metric used to measure a parallel algorithm is the scalability. To study the scalability of P^{2}T^{2}F, we test our model on three datasets by varying the number of cores from 1 to 8. Figure 3 shows the test RMSE versus the running time for P^{2}T^{2}F model with different number of cores (or sub-tensors) P. The result demonstrates that the running time is approximately reduced to a half when the number of cores gets doubled. Note that the different curves of P^{2}T^{2}F eventually converge to the same solution. To see more clearly, we compute the speedup relative to the running time with 1 core ($P = 1$) by varying the number of cores from 1 to 8. Here, we set RMSE = 0.90 as a baseline. The results on S1-S3 are shown in Figure 4. We can intuitively see that P^{2}T^{2}F achieves nearly linear speedup and the speedup ratio increases

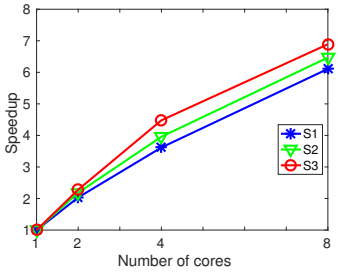


Fig. 4. The speedup of P^{2}T^{2}F w.r.t the number of cores on three datasets.

with the number of ratings. The increased speedup from S1 to S3 is probably caused by the increase of data density.

VI. Conclusion

In this paper, we present P^{2}T^{2}F by deriving a stochastic ADMM algorithm to calculate the latent factors of probabilistic temporal tensors. We propose a new data split strategy to divide the large-scale problem into several independent sub-problems along the user dimension. Then we use the parallel ADMM framework to decompose these sub-tensors in parallel. Experiments on real world data sets demonstrate that our P^{2}T^{2}F model outperforms the traditional CP decomposition, PTTF and PPMF model in terms of efficiency and scalability.

Acknowledgment

This paper was in part supported by Grants from the Natural Science Foundation of China (No. 61572111),

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 03,2020 at 03:01:41 UTC from IEEE Xplore. Restrictions apply.

the National High Technology Research and Development Program of China (863 Program) (No. 2015AA015408), the joint Foundation of the Ministry of Education of China and China Mobile Communication Corporation (No. MCM20150505), a Project funded by China Postdoctoral Science Foundation(No. 2016M602674), a 985 Project of UESTC (No.A1098531023601041) and two Fundamental Research Funds for the Central Universities of China (Nos. ZYGX2014J058, A03012023601042).

REFERENCES

[1] L. De Lathauwer, J. Castaing, and J.-F. Cardoso, "Fourth-order cumulant-based blind identification of underdetermined mixtures," *IEEE Transactions on Signal Processing*, vol. 55, no. 6, pp. 2965–2973, 2007.

[2] A. Shashua and A. Levin, "Linear image coding for regression and classification using the tensor-rank principle," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1. IEEE, 2001, pp. I–42.

[3] G. Beylkin and M. J. Mohlenkamp, "Numerical operator calculus in higher dimensions," *Proceedings of the National Academy of Sciences*, vol. 99, no. 16, pp. 10 246–10 251, 2002.

[4] L. Wang, W. Wu, G. Bosilca, R. Vuduc, and Z. Xu, "Efficient communications in training large scale neural networks," *arXiv preprint arXiv:1611.04255*, 2016.

[5] Z. Xu, F. Yan, and Y. Qi, "Bayesian nonparametric models for multiway data analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 2, pp. 475–487, 2015.

[6] S. Zhe, Z. Xu, X. Chu, Y. A. Qi, and Y. Park, "Scalable nonparametric multiway data analysis." in *AISTATS*, 2015.

[7] S. Chen, M. R. Lyu, I. King, and Z. Xu, "Exact and stable recovery of pairwise interaction tensors," in *Advances in Neural Information Processing Systems*, 2013, pp. 1691–1699.

[8] Z. Xu, F. Yan, and Y. A. Qi, "Infinite tucker decomposition: Nonparametric bayesian models for multiway data analysis," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

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

[10] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[11] L. Xiong, X. Chen, T.-K. Huang, J. G. Schneider, and J. G. Carbonell, "Temporal collaborative filtering with bayesian probabilistic tensor factorization." in *SDM*, vol. 10. SIAM, 2010, pp. 211–222.

[12] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," in *Advances in neural information processing systems*, 2007, pp. 1257–1264.

[13] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using markov chain monte carlo," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 880–887.

[14] I. Porteous, E. Bart, and M. Welling, "Multi-hdp: A non parametric bayesian model for tensor factorization." in *Aaai*, vol. 8, 2008, pp. 1487–1490.

[15] M. N. Schmidt and S. Mohamed, "Probabilistic non-negative tensor factorization using markov chain monte carlo," in *Signal Processing Conference, 2009 17th European*. IEEE, 2009, pp. 1918–1922.

[16] Y. Chi, S. Zhu, Y. Gong, and Y. Zhang, "Probabilistic polyadic factorization and its application to personalized recommendation," in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 941–950.

[17] Y. Koren, "Collaborative filtering with temporal dynamics," *Communications of the ACM*, vol. 53, no. 4, pp. 89–97, 2010.

[18] Z.-Q. Yu, X.-J. Shi, L. Yan, and W.-J. Li, "Distributed stochastic admm for matrix factorization," in *Proceedings of the 23rd ACM Int. Conf. on Inform. and Knowl. Manag.* ACM, 2014, pp. 1259–1268.

[19] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Advances in Neural Information Processing Systems*, 2011, pp. 693–701.

[20] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th ACM conference on Knowledge discovery and data mining*. ACM, 2011, pp. 69–77.

[21] L. Wang, W. Wu, Z. Xu, J. Xiao, and Y. Yang, "Blasx: A high performance level-3 blas library for heterogeneous multi-gpu computing," in *Proceedings of the 2016 International Conference on Supercomputing*. ACM, 2016, p. 20.

[22] L. Wang, W. Wu, J. Xiao, and Y. Yi, "Large scale artificial neural network training using multi-gpus," *arXiv preprint arXiv:1511.04348*, 2015.

[23] U. Kang, E. Papalexakis, A. Harpale, and C. Faloutsos, "Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries," in *Proc. of the 18th ACM conf. on Knowl. disc. and dat. min.* ACM, 2012, pp. 316–324.

[24] J. H. Choi and S. Vishwanathan, "Dfacto: Distributed factorization of tensors," in *Advances in Neural Information Processing Systems*, 2014, pp. 1296–1304.

[25] A. P. Liavas and N. D. Sidiropoulos, "Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 63, no. 20, pp. 5450–5463, 2015.

[26] S. Zhe, K. Zhang, P. Wang, K.-c. Lee, Z. Xu, Y. Qi, and Z. Ghahramani, "Distributed flexible nonlinear tensor factorization," in *Advances in Neural Information Processing Systems*, 2016, pp. 920–928.

[27] F. Shang, Y. Liu, and J. Cheng, "Generalized higher-order tensor decomposition via parallel admm," in *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014, pp. 1279–1285.

[28] S. Zhe, Y. Qi, Y. Park, Z. Xu, I. Molloy, and S. Chari, "Dintucker: Scaling up gaussian process models on large multidimensional arrays." in *AAAI*, 2016, pp. 2386–2392.

[29] R. Tomioka, K. Hayashi, and H. Kashima, "Estimation of low-rank tensors via convex optimization," *ArXiv e-prints*, Oct. 2010.

[30] K. Shin and U. Kang, "Distributed methods for high-dimensional and large-scale tensor factorization," in *Data Mining (ICDM), 2014 IEEE International Conference on*. IEEE, 2014, pp. 989–994.

[31] L. De Lathauwer, "A survey of tensor methods," in *Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on*. IEEE, 2009, pp. 2773–2776.

[32] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[33] D. Goldfarb, S. Ma, and K. Scheinberg, "Fast alternating linearization methods for minimizing the sum of two convex functions," *Mathematical Programming*, vol. 141, no. 1-2, pp. 349–382, 2013.

[34] J. Yang and Y. Zhang, "Alternating direction algorithms for \ell_1-problems in compressive sensing," *SIAM journal on scientific computing*, vol. 33, no. 1, pp. 250–278, 2011.

[35] H. Ouyang, N. He, L. Tran, and A. G. Gray, "Stochastic alternating direction method of multipliers." *ICML (1)*, vol. 28, pp. 80–88, 2013.

APPENDIX

We show the proof to Theorem 1 in the following.

A. Proof of Theorem 1

Proof. The constructed function $h(\cdot)$ in (13) can be written as

$$h(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p, \tau_t | \boldsymbol{A}_t^p, \boldsymbol{B}_t^p, \boldsymbol{C}_t^p) = \sum_{(i,j,k)\in\Omega^p} \hat{h}_{i,j,k}(A_i^p, B_j^p, C_k^p, \tau_t | \boldsymbol{A}_t^p, \boldsymbol{B}_t^p, \boldsymbol{C}_t^p),$$

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 03,2020 at 03:01:41 UTC from IEEE Xplore. Restrictions apply.

where

$$\begin{aligned}
&\hat{h}_{i,j,k}(A_i^p, B_j^p, C_k^p, \tau_t | \boldsymbol{A}_t^p, \boldsymbol{B}_t^p, \boldsymbol{C}_t^p) \\
&\quad = \hat{f}_{i,j,k}((A_i^p)_t, (B_j^p)_t, (C_k^p)_t) \\
&\quad + \nabla_{A_i^p} \hat{f}_{i,j,k}((A_i^p)_t, (B_j^p)_t, (C_k^p)_t)(A_i^p - (A_i^p)_t)^T \\
&\quad + \nabla_{B_j^p} \hat{f}_{i,j,k}((A_i^p)_t, (B_j^p)_t, (C_k^p)_t)(B_j^p - (B_j^p)_t)^T \\
&\quad + \nabla_{C_k^p} \hat{f}_{i,j,k}((A_i^p)_t, (B_j^p)_t, (C_k^p)_t)(C_k^p - (C_k^p)_t)^T \\
&\quad + [1/(2m_i\tau_t)]\|A_i^p - (A_i^p)_t\|_2^2 \\
&\quad + [1/(2n_j\tau_t)]\|B_j^p - (B_j^p)_t\|_2^2 \\
&\quad + [1/(2z_k\tau_t)]\|C_k^p - (C_k^p)_t\|_2^2.
\end{aligned}$$

Here, m_i denotes the number of ratings related to A_i^p in $\mathcal{X}^p$, n_j and z_k are similarly defined.

Then, we have

$$\begin{aligned}
&L^p(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p, (C_0^p)_t, (\boldsymbol{\Theta}_B^p)_t, (\boldsymbol{\Theta}_C^p)_t, \overline{\boldsymbol{B}}_t, \overline{\boldsymbol{C}}_t) \\
&\quad - H^p(\boldsymbol{\mathcal{M}}^p, (C_0^p)_t, (\boldsymbol{\Theta}_B^p)_t, (\boldsymbol{\Theta}_C^p)_t, \overline{\boldsymbol{B}}_t, \overline{\boldsymbol{C}}_t, \tau_t | \boldsymbol{\mathcal{M}}_t^p) \\
&= f(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p) - h(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p, \tau_t | \boldsymbol{A}_t^p, \boldsymbol{B}_t^p, \boldsymbol{C}_t^p) \\
&= \sum_{(i,j,k)\in\Omega^p} [\hat{f}_{i,j,k}(A_i^p, B_j^p, C_k^p) \\
&\quad - \hat{h}_{i,j,k}(A_i^p, B_j^p, C_k^p, \tau_t | \boldsymbol{A}_t^p, \boldsymbol{B}_t^p, \boldsymbol{C}_t^p)].
\end{aligned}$$

For clarity, we denote $A_i^p, B_j^p, C_k^p, (A_i^p)_t, (B_j^p)_t$ and $(C_k^p)_t$ as $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{a}_t, \boldsymbol{b}_t$ and $\boldsymbol{c}_t$, respectively. Then we have

$$\begin{aligned}
\hat{f}_{i,j,k}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) =& \frac{1}{2}(x_{ijk}^p - < \boldsymbol{a} - \boldsymbol{a}_t + \boldsymbol{a}_t, \\
& \boldsymbol{b} - \boldsymbol{b}_t + \boldsymbol{b}_t, \boldsymbol{c} - \boldsymbol{c}_t + \boldsymbol{c}_t >)^2 \\
=& \hat{f}_{i,j,k}(\boldsymbol{a}_t, \boldsymbol{b}_t, \boldsymbol{c}_t) \\
& + \nabla_{\boldsymbol{a}} \hat{f}_{i,j,k}(\boldsymbol{a}_t, \boldsymbol{b}_t, \boldsymbol{c}_t)(\boldsymbol{a} - \boldsymbol{a}_t)^T \\
& + \nabla_{\boldsymbol{b}} \hat{f}_{i,j,k}(\boldsymbol{a}_t, \boldsymbol{b}_t, \boldsymbol{c}_t)(\boldsymbol{b} - \boldsymbol{b}_t)^T \\
& + \nabla_{\boldsymbol{c}} \hat{f}_{i,j,k}(\boldsymbol{a}_t, \boldsymbol{b}_t, \boldsymbol{c}_t)(\boldsymbol{c} - \boldsymbol{c}_t)^T \\
& + \boldsymbol{o}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}),
\end{aligned}$$

where $\boldsymbol{o}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c})$ contains all the second to sixth order terms.

We have the following properties by mainly using Cauchy inequality and the hypothesis $\|\boldsymbol{a} - \boldsymbol{a}_t\|_2^2 \le \delta^2, \|\boldsymbol{b} - \boldsymbol{b}_t\|_2^2 \le \delta^2, \|\boldsymbol{c} - \boldsymbol{c}_t\|_2^2 \le \delta^2$:

$$\begin{aligned}
&-(x_{ijk}^p - < \boldsymbol{a}_t, \boldsymbol{b}_t, \boldsymbol{c}_t >) < \boldsymbol{a} - \boldsymbol{a}_t, \boldsymbol{b} - \boldsymbol{b}_t, \boldsymbol{c}_t > \\
&\quad \le |\epsilon_{ijk}|(\boldsymbol{a} - \boldsymbol{a}_t)((\boldsymbol{b} - \boldsymbol{b}_t) * \boldsymbol{c}_t)^T \\
&\quad \le \frac{1}{2}|\epsilon_{ijk}|(\|\boldsymbol{a} - \boldsymbol{a}_t\|_2^2 + \|\boldsymbol{b} - \boldsymbol{b}_t\|_2^2 \|\boldsymbol{c}_t\|_2^2);
\end{aligned}$$

$$\begin{aligned}
&-(x_{ijk}^p - < \boldsymbol{a}_t, \boldsymbol{b}_t, \boldsymbol{c}_t >) < \boldsymbol{a} - \boldsymbol{a}_t, \boldsymbol{b} - \boldsymbol{b}_t, \boldsymbol{c} - \boldsymbol{c}_t > \\
&\quad \le |\epsilon_{ijk}|(\boldsymbol{a} - \boldsymbol{a}_t)((\boldsymbol{b} - \boldsymbol{b}_t) * (\boldsymbol{c} - \boldsymbol{c}_t))^T \\
&\quad \le \frac{1}{2}|\epsilon_{ijk}|(\|\boldsymbol{a} - \boldsymbol{a}_t\|_2^2 + \|\boldsymbol{b} - \boldsymbol{b}_t\|_2^2 \|\boldsymbol{c} - \boldsymbol{c}_t\|_2^2) \\
&\quad \le \frac{1}{2}|\epsilon_{ijk}|(\|\boldsymbol{a} - \boldsymbol{a}_t\|_2^2 + \|\boldsymbol{b} - \boldsymbol{b}_t\|_2^2 \delta^2);
\end{aligned}$$

where $\epsilon_{ijk} = x_{ijk}^p - < \boldsymbol{a}_t, \boldsymbol{b}_t, \boldsymbol{c}_t >$.

$$\begin{aligned}
&(a_1 + a_2 + \ldots + a_n)^2 \le n(a_1^2 + a_2^2 + \ldots + a_n^2); \\
&(< \boldsymbol{a} - \boldsymbol{a}_t, \boldsymbol{b} - \boldsymbol{b}_t, \boldsymbol{c} - \boldsymbol{c}_t >)^2 \\
&\quad \le \frac{1}{4}(\|\boldsymbol{a} - \boldsymbol{a}_t\|_2^2 + \|\boldsymbol{b} - \boldsymbol{b}_t\|_2^2 \delta^2)^2 \\
&\quad \le \frac{1}{4}(\delta^2 + \delta^4)(\|\boldsymbol{a} - \boldsymbol{a}_t\|_2^2 + \|\boldsymbol{b} - \boldsymbol{b}_t\|_2^2 \delta^2);
\end{aligned}$$

$$\begin{aligned}
&(< \boldsymbol{a} - \boldsymbol{a}_t, \boldsymbol{b} - \boldsymbol{b}_t, \boldsymbol{c}_t >)^2 \\
&\le \frac{1}{4}(\|\boldsymbol{a} - \boldsymbol{a}_t\|_2^2 + \|\boldsymbol{b} - \boldsymbol{b}_t\|_2^2 \|\boldsymbol{c}_t\|_2^2)^2 \\
&\le \frac{1}{4}(\delta^2 + \delta^2 \|\boldsymbol{c}_t\|_2^2)(\|\boldsymbol{a} - \boldsymbol{a}_t\|_2^2 + \|\boldsymbol{b} - \boldsymbol{b}_t\|_2^2 \|\boldsymbol{c}_t\|_2^2);
\end{aligned}$$

$$\begin{aligned}
(< \boldsymbol{a} - \boldsymbol{a}_t, \boldsymbol{b}_t, \boldsymbol{c}_t >)^2 &\le (\|\boldsymbol{a} - \boldsymbol{a}_t\|_2 \|\boldsymbol{b}_t * \boldsymbol{c}_t\|_2)^2 \\
&= \|\boldsymbol{b}_t * \boldsymbol{c}_t\|_2^2 \|\boldsymbol{a} - \boldsymbol{a}_t\|_2^2.
\end{aligned}$$

Using the above six properties, we can prove that

$$\begin{aligned}
\boldsymbol{o}(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}) \le& \pi_A \|\boldsymbol{a} - \boldsymbol{a}_t\|_2^2 + \pi_B \|\boldsymbol{b} - \boldsymbol{b}_t\|_2^2 \\
& + \pi_C \|\boldsymbol{c} - \boldsymbol{c}_t\|_2^2,
\end{aligned}$$

where π_A, π_B, π_C are constants which depend on $\boldsymbol{a}_t$, $\boldsymbol{b}_t$, $\boldsymbol{c}_t$ and δ^2.

If we let

$$\frac{1}{\tau_t} \ge \max\{2m_i\pi_A, 2n_j\pi_B, 2z_k\pi_C\},$$

then we can prove that

$$\begin{aligned}
&H^p(\boldsymbol{\mathcal{M}}^p, (C_0^p)_t, (\boldsymbol{\Theta}_B^p)_t, (\boldsymbol{\Theta}_C^p)_t, \overline{\boldsymbol{B}}_t, \overline{\boldsymbol{C}}_t, \tau_t | \boldsymbol{\mathcal{M}}_t^p) \\
&\quad \ge L^p(\boldsymbol{A}^p, \boldsymbol{B}^p, \boldsymbol{C}^p, (C_0^p)_t, (\boldsymbol{\Theta}_B^p)_t, (\boldsymbol{\Theta}_C^p)_t, \overline{\boldsymbol{B}}_t, \overline{\boldsymbol{C}}_t).
\end{aligned}$$

The second property in Theorem 1 can be easily proved. □

Authorized licensed use limited to: Chinese University of Hong Kong. Downloaded on December 03,2020 at 03:01:41 UTC from IEEE Xplore. Restrictions apply.