Fast DTT – A Near Linear Algorithm for Decomposing a Tensor into Factor Tensors

Xiaomin Fang
Department of Computer Science
School of Information Science and Technology
Sun Yat-sen University
Guangzhou, China
fangxmin@mail2.sysu.edu.cn

Rong Pan*
Department of Computer Science
School of Information Science and Technology
Sun Yat-sen University
Guangzhou, China
panr@sysu.edu.cn

ABSTRACT

As tensors provide a natural representation for the higher-order relations, tensor factorization techniques such as Tucker decomposition and CANDECOMP/PARAFAC decomposition have been applied to many fields. Tucker decomposition has strong capacity of expression, but the time complexity is unpractical for the large-scale real problems. On the other hand, CANDECOMP/PARAFAC decomposition is linear in the feature dimensionality, but the assumption is so strong that it abandons some important information. Besides, both of TD and CP decompose a tensor into several factor matrices. However, the factor matrices are not natural for the representation of the higher-order relations. To overcome these problems, we propose a near linear tensor factorization approach, which decompose a tensor into factor tensors in order to model the higher-order relations, without loss of important information. In addition, to reduce the time complexity and the number of the parameters, we decompose each slice of the factor tensors into two smaller matrices. We conduct experiments on both synthetic datasets and real datasets. The experimental results on the synthetic datasets validate that our model has strong capacity of expression. The results on the real datasets show that our approach outperforms the state-of-the-art tensor factorization methods.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*; I.2.6 [**Artificial Intelligence**]: Learning

Keywords

Tensor decomposition; DTT

*Corresponding author

1. INTRODUCTION

Matrix factorization is used for modeling the second-order relations. For example, for the item recommendation problem, we can construct a user-item matrix. The user-item matrix is factorized into two factor matrices, representing the latent features of the users and the items, respectively. From the users' view, matrix factorization assumes the items are divided into D item-groups, where D denote the number of the elements in the vector. For user u, there is a corresponding vector of the factor matrix of the users, describing the relation between that user and the items. Each element of that vector represents the relevance between that user and a item-groups. From the items'view, the situation is similar.

Tensor is a higher-order extension of matrix and tensor factorization is a higher-order extension of matrix factorization. Since tensor decomposition can model the higher-order relations, tensor decomposition techniques are applied to many fields, such are psychology, chemometrics, computer vision and data mining. Traditional tensor decomposition techniques like Tucker decomposition (TD) [21] and CANDECOMP/PARAFAC (CP) [2, 9] decomposition factorize a tensor into several factor matrices. Take the problem of personalized tag recommendation for example. Traditional tensor decomposition techniques decompose a user-item-tag tensor into three factor matrices. Just like matrix factorization, from the users' view, for user u, there is a corresponding vector of the factor matrix of the users, indicating the relation between that user, the items and the tag, but we can not explain the meaning of each element in that vector by the same way we analyzing matrix factorization, since vector is not natural for the representation of the relations between that user, the item-groups and the tag-groups. Moreover, both of Tucker decomposition and CANDECOMP/PARAFAC have some drawbacks. Tucker decomposition has strong capacity of expression but it requires lots of computation power, which is unpractical for large-scale real problems. The other tensor factorization technique CANDECOMP/PARAFAC decomposition is linear in the feature dimensionality, but it makes a strong assumption that limits its capacity of expression.

To tackle all these problems, we propose a novel tensor factorization approach in this paper. We decompose a tensor into several factor tensors instead of factor matrices. From the users' view, for user u, there is a corresponding matrix of the factor tensor of the users, indicating the relation between that user, the items and the tag, and each element of that matrix can be seen as the relevance between that user, a item-group and a tag-group. Thus, the higher-order relations can be modeled by a more natural way. In addition, to reduce the time complexity and the number of the parameters, we decompose each slice of the factor tensors into two smaller matrices. Our proposed model is near linear and has strong capacity of expression.

The contributions of our work are summarized as follows.

- We propose a novel tensor decomposition approach, decomposing a tensor into factor tensors rather than factor matrices, which is more natural for the representation of the higher-order relations
- Our proposed approach has strong capacity of expression and is near linear, which is practical for the large-scale real problems.
- Our experimental results on the synthetic datasets verify that our proposed approach has strong capacity of expression and our proposed approach significantly outperforms other tensor factorization techniques on the real datasets.

The rest of the paper is organized as follows. The related work to tensor factorization techniques is depicted in the next section. The notations and preliminaries of tensor and tensor factorization are presented in section 3. Our proposed tensor decomposition approach is described in section 4. The experimental results on the synthetic datasets and the real datasets are showed and analyzed in section 5. Finally, we conclude our work in section 6.

2. RELATED WORK

Since tensors provide a natural representation for the higher-order relations, they are applied to many fields [12], including chemometrics [1], neuroscience [13], graph analysis [18, 6, 5, 24], personalized recommendation [19, 24, 11, 14, 17, 7, 20, 16, 23] and so on. Tucker decomposition (TD) [21] and CANDECOMP/PARAFAC decomposition (CP) [2, 9] are the most common techniques to decompose a tensor, where Tucker decomposition has stronger capacity of expression and CANDECOMP/PARAFAC decomposition can be trained in linear time.

Some researches are based on Tucker decomposition (TD) [3, 4, 19, 11, 14]. Higher-Order Singular Value Decomposition (HOSVD) [3, 4], a generation of the matrix Singular Value Decomposition (SVD), is introduced for computing Tucker decomposition. Sun et al. [19] construct a user-query-URL tensor from the clickthrough data in the search engines and employ HOSVD to capture the latent factors of users, queries and URLs for personalized web search. Karatzoglou et al. [11] integrate the context information into the traditional collaborative filtering models to improve the recommendation quality by making use of HOSVD as well. However, one of the drawbacks of HOSVD is that it can not deal with the missing data in the sparse tensors. It treats the values of all the missing elements in the tensors as zeros. To fix this problem, Rendle et al. [14] propose a method called RTF, which exploits Bayesian Personalized Ranking (BPR) [15] and learns from pairwise constraints, for personalized tag recommendation. Although TD has a strong capacity of expression for the higher-order relations, the time complexity of which is not feasible for large-scale real problems.

On the other hand, some studies employ CP decomposition (CANDECOMP/PARAFAC decomposition) [5, 23, 17]. Dunlavy, Kolda, and Acar [5] incorporate the time information for link predictions. The first two dimensions of the tensor represent the entities and the third dimension represents the time slices. They decompose the tensor by CP. Similarly, Xiong et al. [23] take the time into consideration and add a special constraint on the time dimension. Work by Rendle and Schmidt-Thieme [17] puts forward a new tensor factorization technique, Pairwise Interaction Tensor Factorization (PITF) for personalized tag recommendation. PITF can be seen as a special case of CP, modeling the pairwise interactions between the three dimensions of the user-item-tag tensor. The CP-based models can be trained in linear time, but it abandons some important information and restricts the capacity of expression.

Furthermore, both TD and CP decompose a tensor into several factor matrices, but the factor matrices are not natural for the representation of the higher-order relations, since matrices can only represent the second-order relations. Therefore, we propose a novel tensor factorization approach in this paper, which decomposes a tensor into several factor tensors instead.

3. NOTATIONS AND PRELIMINARIES

In this section, we first introduce tensor and the notations. Then, we depict two most common tensor factorization techniques, Tucker decomposition and CANDECOMP/PARAFAC, and analyze the advantages and the disadvantages of them.

3.1 Tensor

A tensor is a multidimensional or multi-way array and the order of a tensor is the number of the dimensions or modes. Vectors are first-order tensors, matrices are second-order tensors and tensors of order three or higher are called higher-order tensors. Matrices have column and row, a column and a row of which are called mode-1 vector and mode-2 vector respectively. Similar to matrices, the third-order tensors have column, row and tube and a tube is a mode-3 vector. Since the constructions of the higher-order tensors are similar, we focus on the third-order tensors and the third-order tensors are called tensors for short in the following paper.

Matrices and tensors are denoted by boldface capital letters, e.g., $\mathbf{X}$. Element (i, j) of a matrix $\mathbf{X}$ is denoted by x_{ij} and element (i, j, k) of a tensor $\mathbf{A}$ is denoted by a_{ijk}.

3.2 Tensor Decomposition

Just like matrix factorization, tensor decomposition techniques factorize a tensor into several components. Tucker decomposition (TD) [21] and CANDECOMP/PARAFAC decomposition (CP) [2, 9] are the most common tensor decomposition techniques and can be seen as higher-order generations of the matrix factorization, Singular Value Decomposition (SVD) [8].

Tensor decomposition techniques have been applied to personalized tag recommendation. For the sake of convenience of analysis, let's take the personalized tag recommendation problem for example in the following paper. A user-item-tag tensor $\mathbf{A} \in \mathbb{R}^{I \times J \times K}$ is constructed from the social tagging data, where I, J, K denote the number of the users, the items and the tags, respectively. Element a_{ijk} of tensor $\mathbf{A}$ measures the probability of the i-th user annotat-

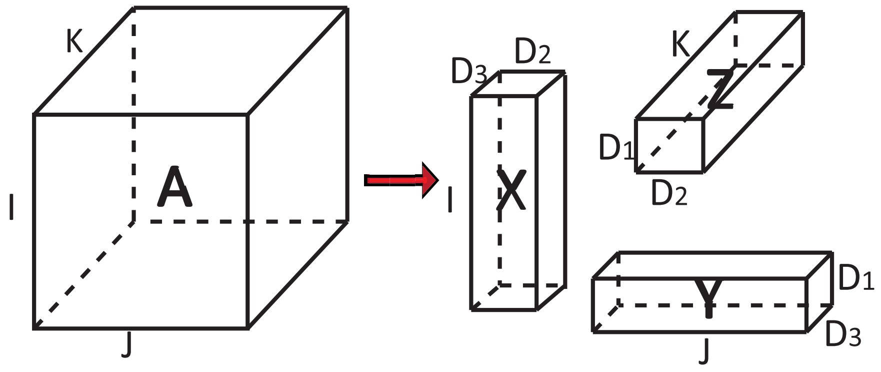


Figure 1: DTT: A novel tensor factorization model

ing the j-th item with the k-th tag. We treat all of the users, the items and the tags as three different types of entities.

Tucker decomposition (TD) [21] was first introduced by Tucker in 1966. It decomposes a tensor into a core tensor multiplied by a matrix along each mode. Element a_{ijk} of tensor $\mathbf{A} \in \mathbb{R}^{I \times J \times K}$ can be written as

$$a_{ijk} = \sum_{p=1}^{D_1}\sum_{q=1}^{D_2}\sum_{r=1}^{D_3} c_{pqr} \cdot x_{ip} \cdot y_{jq} \cdot z_{kr}, \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{I \times D_1}$, $\mathbf{Y} \in \mathbb{R}^{J \times D_2}$, $\mathbf{Z} \in \mathbb{R}^{K \times D_3}$ are the factor matrices and $\mathbf{C} \in \mathbb{R}^{D_1 \times D_2 \times D_3}$ is the core tensor. D_1, D_2 and D_3 are the numbers of the latent features. Given a factor matrix, a vector of that matrix represents the latent features of the corresponding entity. For user i, item j and tag k, c_{pqr} of the core tensor $\mathbf{C}$ indicates the relevance between the p-th feature of user i, the q-th feature of item j and the r-th feature of tag k.

CANDECOMP/PARAFAC decomposition (CP) [2, 9] decomposes a tensor into a sum of component rank-one tensors. $a_{i,j,k}$ can be written as

$$a_{ijk} = \sum_{p=1}^{D} x_{ip} \cdot y_{jp} \cdot z_{kp}, \tag{2}$$

where $\mathbf{X} \in \mathbb{R}^{I \times D}$, $\mathbf{Y} \in \mathbb{R}^{J \times D}$, $\mathbf{Z} \in \mathbb{R}^{K \times D}$ are the factor matrices and D is the number of latent features. For user i, item j and tag k, CP makes a strong assumption that the p-th feature of user i, x_{ip} is only relevant to the p-th feature of item j, y_{jp} and the p-th feature of tag k, z_{kp}. In other words, it assumes x_{ip} is independent of y_{jq} and z_{kr}, where $p \neq q$ and $p \neq r$. As generally $D << I, J, K$ on real problems, the assumption limits CP's capacity of expression.

TD is more flexible and can model the higher-order relations better, but it requires lots of computation power. Therefore, TD is not practical for large-scale real problems. On the other hand, CP is linear in the feature dimensionality, but it gives up some important information. To overcome the drawbacks, we propose a near linear tensor decomposition technique, which is practical for large-scale problems, without lost of information.

4. DECOMPOSING A TENSOR INTO FACTOR TENSORS

In this section, we explain our proposed tensor factorization approach and how to compute our model. Besides, we compare our proposed approach with other tensor factorization techniques on the time complexity, the space complexity and the capacity of expression.

4.1 Tensor Factorization Model DTT

Singular Value Decomposition (SVD) factorize a matrix into two smaller factor matrices and the factor matrices present the the second-order relations. Tensor factorization is a higher-order generation of matrix factorization. Traditional tensor decomposition techniques like TD and CP, decomposing a tensor $\mathbf{A} \in \mathbb{R}^{I \times J \times K}$ into factor matrices, but the factor matrices are not proper for presenting the higher-order relations. Different from TD and CP, we decompose the tensor $\mathbf{A}$ into factor tensors, since the representation of a factor tensor is more natural for the higher-order relations, compared with the representation of a factor matrix. We refer our proposed tensor factorization approach to decomposing a tensor into tensors (DTT). The framework of our proposed tensor factorization model is illustrated in Figure 1. From the figure, we can see that tensor $\mathbf{A}$ is factorized into three smaller factor tensors $\mathbf{X} \in \mathbb{R}^{I \times D_2 \times D_3}$, $\mathbf{Y} \in \mathbb{R}^{J \times D_3 \times D_1}$ and $\mathbf{Z} \in \mathbb{R}^{K \times D_1 \times D_2}$, where D_1, D_2 and D_3 are parameters of the DTT model (e.g., taking personalized social tagging as an example, they can been seen as the numbers of the user groups, the item groups and the tag groups, respectively).

Since we decompose a tensor into factor tensors rather than factor matrices, the latent features of an entity are represented by a matrix rather than a vector. $D_2 \times D_3$ is the size of the factor matrix of a user, $D_3 \times D_1$ is the size of the factor matrix of an item and $D_1 \times D_2$ is the size of the factor matrix of a tag. From the users' view, the factor matrix of user i, the i-th slice in $\mathbf{X}$, which is denoted by $\mathbf{X}_{i::}$, contains D_2 rows and D_3 columns. It can be assumed that for user i, all the items are divided into D_2 item-groups and the q-th row of $\mathbf{X}_{i::}$ represents the characters of the q-th item-group. Similarly, for user i, all the tags are divided into D_3 tag-groups and the r-th column of $\mathbf{X}_{i::}$ represents the characters of the r-th tag-group. Thus, x_{iqr} means the relevance between user i, the items in the q-th item-group and the tags in the r-th tag-group. That is to say, matrix $\mathbf{X}_{i::}$ indicates the preference of user i under different conditions. The analysis from the views of the items and the tags is similar.

Our proposed approach DTT models the higher-order relations by modeling the relations between the user-groups,

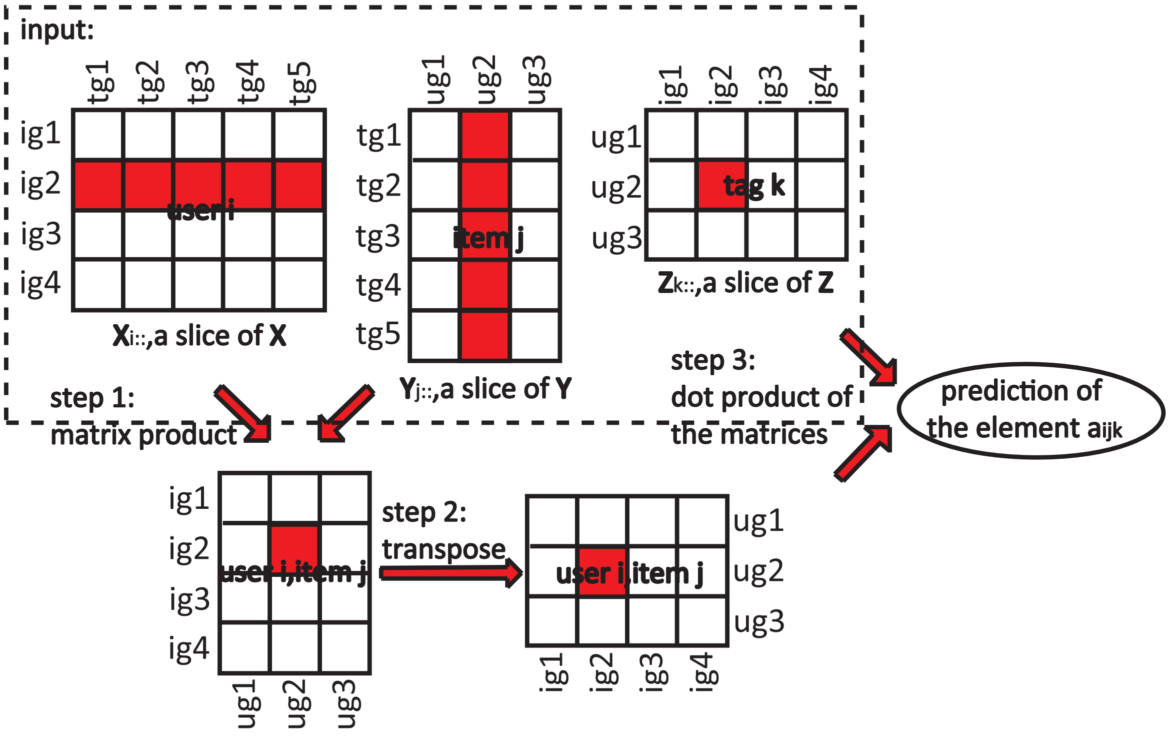


Figure 2: Reconstruction of the tensor elements from the three slices of the factor tensors

the item-groups and the tag-groups. For triplet (i, j, k), the way how we predict the value of that triplet is illustrated in Figure 2, where ug, ig and tg are short for the user-group, the item-group and the tag-group.

- In step 1, we compute the matrix product of the factor matrix of user i, $\mathbf{X}_{i::} \in \mathbb{R}^{D_2 \times D_3}$ and the factor matrix of item j, $\mathbf{Y}_{j::} \in \mathbb{R}^{D_3 \times D_1}$. Thus, the factor matrices of user i and item j are integrated into a new factor matrix $\mathbf{G}^{(ij)} \in \mathbb{R}^{D_2 \times D_1}$ by their multiplication. That is, $\mathbf{G}^{(ij)} = \mathbf{X}_{\mathbf{i}::}\mathbf{Y}_{\mathbf{j}::}$.

- In step 2, we transpose the new factor matrix $\mathbf{G}^{(ij)} \in \mathbb{R}^{D_2 \times D_1}$ to make its dimension be the same as the dimension of the factor matrix of tag k, $\mathbf{Z}_{k::} \in \mathbb{R}^{D_1 \times D_2}$.

- Finally, in step 3, we define the prediction of the tensor element a_{ijk} as the inner product of two matrices $(\mathbf{G}^{(ij)})^{\mathbf{T}} \in \mathbb{R}^{D_1 \times D_2}$ and the factor matrix of tag k, $\mathbf{Z}_{k::} \in \mathbb{R}^{D_1 \times D_2}$. That is,

$$a_{ijk} = \left\langle (\mathbf{G}^{(ij)})^{\mathbf{T}}, \mathbf{Z}_{k::} \right\rangle = \left\langle (\mathbf{X}_{\mathbf{i}::}\mathbf{Y}_{\mathbf{j}::})^{\mathbf{T}}, \mathbf{Z}_{k::} \right\rangle, \quad (3)$$

where the inner product of matrices $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{\mathbf{m} \times \mathbf{n}}$ is defined as

$$\langle \mathbf{X}, \mathbf{Y} \rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} y_{ij}.$$

Note that Eq. (3) is equivalent to

$$a_{ijk} = \sum_{p=1}^{D_1} \sum_{q=1}^{D_2} \sum_{r=1}^{D_3} x_{iqr} \cdot y_{jrp} \cdot z_{kpq}, \quad (4)$$

where $\mathbf{X} \in \mathbb{R}^{I \times D_2 \times D_3}$, $\mathbf{Y} \in \mathbb{R}^{J \times D_3 \times D_1}$ and $\mathbf{Z} \in \mathbb{R}^{K \times D_1 \times D_2}$ are the factor tensors. As the order of x_{iqr}, y_{jrp} and z_{kpq} has no influence on computing a_{ijk} in Eq. (4), the order of the factor matrices of user i, item j and tag k in Figure 2 has no influence on computing the relevance score, which is a good property as follows.

Property 1. The prediction of tensor element a_{ijk} is well defined. That is,

$$\begin{aligned} a_{ijk} &= \left\langle (\mathbf{X}_{\mathbf{i}::}\mathbf{Y}_{\mathbf{j}::})^{\mathbf{T}}, \mathbf{Z}_{k::} \right\rangle && (5)\\ &= \left\langle (\mathbf{Y}_{\mathbf{j}::}\mathbf{Z}_{\mathbf{k}::})^{\mathbf{T}}, \mathbf{X}_{i::} \right\rangle && (6)\\ &= \left\langle (\mathbf{Z}_{\mathbf{k}::}\mathbf{X}_{\mathbf{i}::})^{\mathbf{T}}, \mathbf{Y}_{j::} \right\rangle. && (7) \end{aligned}$$

4.2 Fast DTT

Although DTT can capture the higher-order relations, the time complexity to calculate Eq. (4) is $\mathcal{O}(D_1D_2D_3)$, which is unacceptable for the large-scale real problems. Besides, there are too many parameters in DTT that it may lead to overfitting, the space complexity of which is $\mathcal{O}(ID_2D_3 + JD_3D_1 + KD_1D_2)$, where I, J and K denote the numbers of the users, the items and the tags, repectively. Note that, for each entity, there is a factor matrix describe the features of that entity. To reduce the number of parameters and the time complexity, we decompose the factor matrix of each entity into two smaller matrices. For user i, the factor matrix $\mathbf{X}_{i::} \in \mathbb{R}^{D_2 \times D_3}$ are decomposed into two smaller matrices $\mathbf{X}^{(l)}_{i::} \in \mathbb{R}^{D_2 \times d_1}$ and $\mathbf{X}^{(r)}_{i::} \in \mathbb{R}^{D_3 \times d_1}$. For item j, the factor matrix $\mathbf{Y}_{j::} \in \mathbb{R}^{D_3 \times D_1}$ are decomposed into two smaller matrices $\mathbf{Y}^{(l)}_{j::} \in \mathbb{R}^{D_3 \times d_2}$ and $\mathbf{Y}^{(r)}_{j::} \in \mathbb{R}^{D_1 \times d_2}$. For tag k, the factor matrix $\mathbf{Z}_{k::} \in \mathbb{R}^{D_1 \times D_2}$ are decomposed into two

Table 1: Comparison between DTT, Tucker decomposition (TD) and CANDECOMP/PARAFAC (CP)

Algorithm	Time Complexity		Space Complexity
	Train	Predict	
DTT	$\mathcal{O}(Dd^2NT)$	$\mathcal{O}(Dd^2)$	$\mathcal{O}(Dd^2I)$
TD	$\mathcal{O}(D^3NT)$	$\mathcal{O}(D^3)$	$\mathcal{O}(D^3I)$
CP	$\mathcal{O}(DNT)$	$\mathcal{O}(D)$	$\mathcal{O}(DI)$

smaller matrices $\mathbf{Z}^{(l)}_{k::} \in \mathbb{R}^{D_1 \times d_3}$ and $\mathbf{Z}^{(r)}_{j::} \in \mathbb{R}^{D_2 \times d_3}$. d_1, d_2 and d_3 are the numbers of latent features of the factor matrices $\mathbf{X}_{i::}$, $\mathbf{Y}_{j::}$ and $\mathbf{Z}_{k::}$, respectively. Additionally, d_1, d_2 and d_3 should satisfy $d_1 \le min(D_2, D_3)$, $d_2 \le min(D_3, D_1)$ and $d_3 \le min(D_1, D_2)$. d_1, d_2 and d_3 depend on the complexity of the pairwise relations. For example, if the relation between the items and the tags is complex for the users, d_1 should be large so as to reconstruct the factor matrices of the users and if the relation between the items and the tags is simple, d_1 is small. Concretely, from the users' view, for user i, matrix $\mathbf{X}^{(l)}_{i::}$ indicates the relations between user i and the item-groups and matrix $\mathbf{X}^{(r)}_{i::}$ indicates the relations between user i and the tag-groups. Thus, the multiplication of $\mathbf{X}^{(l)}_{i::}$ and $\mathbf{X}^{(r)}_{i::}$, $\mathbf{X}_{i::}$, indicates the relations between user i, the item-groups and the tag-groups.

After decomposing each factor matrix into two smaller matrices, a_{ijk} can be rewritten as

$$a_{ijk} = \sum_{p=1}^{D_1}\sum_{q=1}^{D_2}\sum_{r=1}^{D_3}\sum_{u=1}^{d_1} x^{(l)}_{iqu}x^{(r)}_{iru} \sum_{v=1}^{d_2} y^{(l)}_{jrv}y^{(r)}_{jpv} \sum_{w=1}^{d_3} z^{(l)}_{kpw}z^{(r)}_{kqw}, \tag{8}$$

where

$$x_{iqr} = \sum_{u=1}^{d_1} x^{(l)}_{iqu}x^{(r)}_{iru},$$

$$y_{jrp} = \sum_{v=1}^{d_2} y^{(l)}_{jrv}y^{(r)}_{jpv},$$

$$z_{kpq} = \sum_{w=1}^{d_3} z^{(l)}_{kpw}z^{(r)}_{kqw}.$$

Note that, Eq. (8) is equivalent to

$$a_{ijk} = \sum_{u=1}^{d_1}\sum_{v=1}^{d_2}\sum_{w=1}^{d_3} \psi^{(1)}_{jk}(v,w)\cdot\psi^{(2)}_{ki}(w,u)\cdot\psi^{(3)}_{ij}(u,v), \tag{9}$$

where we define

$$\psi^{(1)}_{jk}(v,w) = \sum_{p=1}^{D_1} y^{(r)}_{jpv}\cdot z^{(l)}_{kpw},$$

$$\psi^{(2)}_{ki}(w,u) = \sum_{q=1}^{D_2} z^{(r)}_{kqw}\cdot x^{(l)}_{iqu},$$

$$\psi^{(3)}_{ij}(u,v) = \sum_{r=1}^{D_3} x^{(r)}_{iru}\cdot y^{(l)}_{jrv}.$$

Eq. (9) can be calculated in $\mathcal{O}(D_1d_2d_3 + D_2d_3d_1 + D_3d_1d_2)$ and generally, $d_1, d_2, d_3 << D_1, D_2, D_3$. Therefore, DTT is feasible for large-scale real problems. Besides, the number of parameters is reduced from $\mathcal{O}(ID_2D_3 + JD_3D_1 + KD_1D_2)$ to $\mathcal{O}(Id_1(D_2+D_3) + Jd_2(D_3+D_1) + Kd_3(D_1+D_2))$.

Tensor decomposition techniques are employed to many applications. As for different applications, the objective function are different, let's assume the objective function is F. Gradient-base approaches are common to learn the parameters of the models and we use gradient-based approaches to optimize the objective function F. The gradients are

$$\frac{\partial a_{ijk}}{\partial x^{(l)}_{iqu}} = \sum_{w=1}^{d_3} \phi^{(2)}_{ijk}(w,u)\cdot z^{(r)}_{kqw},$$

$$\frac{\partial a_{ijk}}{\partial x^{(r)}_{iru}} = \sum_{v=1}^{d_2} \phi^{(3)}_{ijk}(u,v)\cdot y^{(l)}_{jrv},$$

$$\frac{\partial a_{ijk}}{\partial y^{(l)}_{jrv}} = \sum_{v=1}^{d_2} \phi^{(3)}_{ijk}(u,v)\cdot x^{(r)}_{iru},$$

$$\frac{\partial a_{ijk}}{\partial y^{(r)}_{jpv}} = \sum_{u=1}^{d_1} \phi^{(1)}_{ijk}(v,w)\cdot z^{(l)}_{kpw},$$

$$\frac{\partial a_{ijk}}{\partial z^{(l)}_{kpw}} = \sum_{u=1}^{d_1} \phi^{(1)}_{ijk}(v,w)\cdot y^{(r)}_{jpv},$$

$$\frac{\partial a_{ijk}}{\partial z^{(r)}_{kqw}} = \sum_{w=1}^{d_3} \phi^{(2)}_{ijk}(w,u)\cdot x^{(l)}_{iqu},$$

where we define

$$\phi^{(1)}_{ijk}(v,w) = \sum_{u=1}^{d_1} \psi^{(2)}_{ki}(w,u)\cdot\psi^{(3)}_{ij}(u,v),$$

$$\phi^{(2)}_{ijk}(w,u) = \sum_{v=1}^{d_2} \psi^{(3)}_{ij}(u,v)\cdot\psi^{(1)}_{jk}(v,w),$$

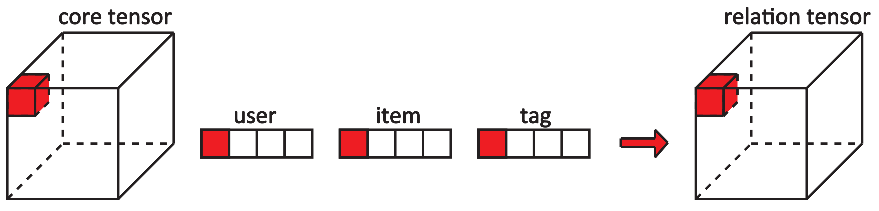
$$\phi^{(3)}_{ijk}(u,v) = \sum_{w=1}^{d_3} \psi^{(1)}_{jk}(v,w)\cdot\psi^{(2)}_{ki}(w,u).$$

Thus, given a_{ijk} in $\mathbf{A}$, the gradients can be calculated in $\mathcal{O}(D_1d_2d_3 + D_2d_3d_1 + D_3d_1d_2)$. With the gradients, we can optimize the objective function F.

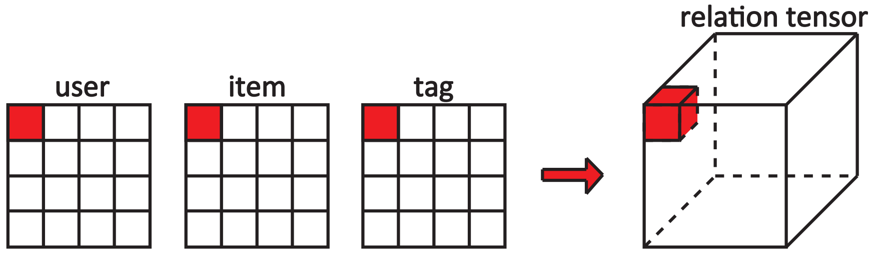
4.3 Comparisons with TD and CP

For the simplicity of the analysis, we assume $D_1 = D_2 = D_3 = D$, $d_1 = d_2 = d_3 = d$ and $I = J = K$, where I, J and K denote the number of the users, the items and the tags. Let N denote the number of the triplets in the training set and T denote the number of iterations for training. The comparison of the time complexity and space complexity is summarized in Table 1.

For TD, it makes the assumption that the users, the items and the tags are divided into several groups and uses a core tensor to model the higher-order relations between the user-groups, the item-groups and the tag-groups. Given user i, item j, tag k, the higher-order relation between them is or-



(a) Tucker decomposition



(b) DTT

Figure 3: Comparison of Tucker decomposition and DTT

ganized into a relation tensor. The framework of Tucker decomposition is illustrated in Figure 3(a). The three vectors on the left of the arrow describe the factors of the user, the item and the tag, respectively. The core tensor and the three vectors are integrated into a relation tensor, which is on the right of the tensor. Although Tucker decomposition is flexible and expressive, it takes $\mathcal{O}(D^3NT)$ to train the data and $\mathcal{O}(D^3)$ to predict a triplet. Besides, since the core tensor is shared by all the triplets, it is not suitable to apply parallel computing to training the data. Therefore, TD is unpractical on large-scale datasets.

On the other hand, it takes $\mathcal{O}(DNT)$ to trained a CP-based model and $\mathcal{O}(D)$ to predict a triplet. Thus, CP is efficient. Moreover, CP divides the users, the items and the tags into several groups as well. However, it assumes the users in the p-th user-group are independent of the items in the q-th item-group and the tags in the r-th tag-group, where $p \neq q$ and $p \neq r$. In fact, $D << I$ when solving real problems and it is possible that a user in the first user-group might annotate an item in the first item-group with a tag in the first tag-group and annotate another item in the second item-group with another tag in the third tag-group. Therefore, The assumption is too strong that it reduce the capacity of expression.

For user i, item j, tag k, DTT exploits a special multiplication $x_{iqr}y_{jrp}z_{kpq}$ to model the higher-order relation between them. Just like TD, it organizes the relation into a relation tensor as well, the framework of which is showed in Figure 3(b). The three matrices on the left of the arrow are the factor matrices of the user, the item and the tag and the factor matrices are integrated into a relation tensor, which is on the right of the arrow. Since DTT doesn't make the strong independent assumption like CP, it is more expressive. Moreover, DTT requires $\mathcal{O}(Dd^2NT)$ to train and $\mathcal{O}(Dd^2)$ to predict, where $d << D$. Thus, DTT is near linear. According to the experimental results on the real datasets, $d = 1$ is enough to capture the higher-order relations. Thus, DTT is efficient for large-scale problems. Furthermore, compared with TD and CP, DTT exploits a more natural way to represent the features of an entity, by using a matrix to describe the features of the entity rather than a vector.

5. EXPERIMENTAL RESULTS

In this section, we compare our proposed model with other tensor factorization techniques on the synthetic datasets and the real datasets. The experiments on the synthetic datasets are used to compare the capacity of expression and the experiments on the real datasets are used to compare the accuracy for predicting a triplet of a tensor.

5.1 Experimental Setup

As tensor decomposition techniques are applied to personalized tag recommendation, we use real tag annotation datasets to evaluate the performance of DTT. In addition, we generate synthetic datasets to compare different factorization models' capacity of expression.

Let $\mathbf{A}$ denote the tensor constructed from the training set and P denote the observed user-item posts in the training set. Given post (u, i), $T^+_{(u,i)}$ and $T^-_{(u,i)}$ denote the set of the positive tags and the set of the negative tags for that post, respectively. We construct a candidate set for each post (u, i). If triplet (u, i', t) is observed for some item i' or triplet (u', i, t) is observed for some user u', t is treated as the candidate tag for post (u, i). For tag t in the candidate set of post (u, i), if triplet (u, i, t) is observed, t is treated as the positive tag for post (u, i), otherwise, the negative tag. Following the work of [14], we optimize the pair-wise ranking function Bayesian Personalized Ranking (BPR) [15] for personalized tag recommendation. The objective function

F is defined as

$$F = \sum_{(u,i)\in P} \frac{1}{|T^+_{u,i}||T^-_{u,i}|} \sum_{t^+ \in T^+_{u,i}} \sum_{t^- \in T^-_{u,i}} \sigma(a_{uit^+} - a_{uit^-}) + \lambda \sum_{\theta\in\Theta} \|\theta - \mu\|^2_F, \quad (10)$$

where λ is the weight of the regularization term, Θ are the model parameters and μ denotes the mean of the parameters. We use Stochastic Gradient Descent (SGD) to optimize the objective function F.

5.2 Evaluation Methodology

We compare our approach with three categories of baselines, CP-based, TD-based and popularity-based. PITF [17], CP are CP-based, RTF [14] and HOSVD [3] are TD-based. For popularity-based, given a post (u, i), method *popularity* ranks the tags based on the frequency item i and the tags co-occur in the training set.

For convenience, we set $D_1 = D_2 = D_3 = D$ and $d_1 = d_2 = d_3 = d$. We performed all the experiments multiple times and tuned the hyper-parameters of all models to achieve the best performance on the first training split for each dataset. Most of the parameters of the tensor factorization models are drawn from normal distribution $N(\mu, 0.01^2)$. For all the parameters in DTT, $\mu = \frac{1}{\sqrt{Dd}}$. For all the parameters in PITF, $\mu = 0.0$. For all the parameters in CP, $\mu = \frac{1}{\sqrt[3]{D}}$. For the parameters of the factor matrices in RTF, $\mu = \frac{1}{D}$ and the parameters of the core tensor in RTF are drawn from normal distribution $N(0, 0.1)$. Besides, the weight of the regularization term $\lambda = 0.0001$ on the synthetic datasets and $\lambda = 0.002$ on the real datasets for DTT and CP, $\lambda = 10^{-6}$ on all the datasets for RTF, and $\lambda = 0.001$ on synthetic datasets and $\lambda = 0.01$ on the real datasets for PITF.

We exploit two widely used metrics in information retrieval, Normalized Discounted Cumulative Gain (NDCG) and Mean Average Precision (MAP), to evaluate the performance of different models.

Discounted Cumulative Gain (DCG) evaluates the gain of a recommendation list based on the positions of the tags in that list. The NDCG accumulated at a particular position p is defined as

$$DCG@p = \sum_{i=1}^{p} \frac{2^{rel(i)} - 1}{log_2(i+1)}, \quad (11)$$

where $rel(i)$ is 1 if the tag at position i i relevant and 0, otherwise. Normalized Discounted Cumulated Gain (NDCG) is defined as

$$NDCG@p = \frac{DCG@p}{IDCG@p} \quad (12)$$

where IDCG (Ideal Discounted Cumulated Gain) is the DCG of the ideal ranked list, which is introduced to normalize the DCG.

Mean Average Precision (MAP) computes the mean of average precision (AP) over all posts in the test set, where AP is the average of precisions computed at all positions with a preferred tag and defined as

$$AP = \frac{\sum_{i=1}^{N} prec(i) \times rel(i)}{\sum_{i=1}^{N} rel(i)}, \quad (13)$$

where $prec(i)$ is the precision of the cutoff rank list from the first tag to the i-th tag and N is the size of the corresponding candidate set.

5.3 Performance on Synthetic Datasets

To evaluate different tensor factorization models' capacity of expression, we generate 10 synthetic datasets. There are 100 users, 100 items and 100 tags in each synthetic dataset. For each dataset, We first randomly divided the users into 10 user-groups, divide the items into 10 item-groups and divide the tags into 10 tag-groups. We assume the users in the i-th user-group always annotate the items in the j-th item-groups with the tags in the $g(i, j)$-th tag-group, where $g(i, j)$ is randomly generated in the range 1 - 10. Thus, to sample a triplet, we first randomly sample a user and an item, and then sample a tag in the $g(i, j)$-th tag group, where i, j denote the IDs of the user-group that user belongs to and the item-group that item belongs to, respectively. For each dataset, we randomly sample 6000 triplets and split the triplets into a training set with 5000 triplets and a test set with 1000 triplets. By this way, we can compare different tensor factorization techniques' capacity of expression and verify whether the independent assumption made by CP limits its capacity.

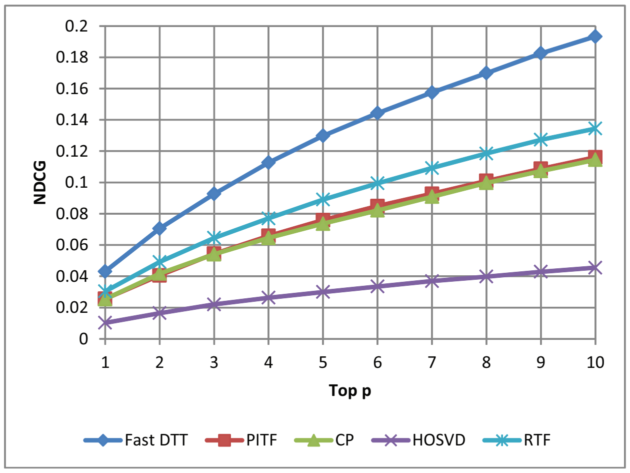


Figure 4: Performance on synthetic datasets

The experiments are repeated 10 times. We compare DTT to other tensor factorization models and for all tensor factorization models, $D = 2$, and for DTT $d = 1$. The results on the synthetic datasets are showed in Figure 4. HOSVD works poorly, since it doesn't handle the missing value in the tensors. As the other TD-based model, RTF, exploits the pair-wise ranking function Bayesian Personalized Ranking (BPR) [15], the performance is much better. Moreover, as we expected, DTT and RTF outperforms the CP-based models, PITF and CP, because the independent assumption constraints CP-based models' capacity. CP-based models assumes the users in the i-th user group will only annotate the items in the i-th item groups and annotate the items with the tags in the i-th tag group, which is not always the truth. Thus, DTT and TD-based models has stronger capacity of expression than CP-based models. Besides, DTT and TD-based models like RTF model the higher-order relation by a relation tensor, but it seems the method DTT exploiting to construct the relation tensor is better.

Table 2: Data statistics

Dataset	Users	Items	Tags	Posts	Triplets	Density	Average Candidates
Delicious	1,681	29,540	7,251	58,114	283,030	7.89×10^{-7}	131.79
Last.fm	1,348	6,927	2,132	59,849	162,047	8.14×10^{-6}	58.90
Movielens	456	1,973	1,222	15,210	27,0l26	2.46×10^{-5}	85.96
Delicious-large	433,791	1,241,772	125,455	30,538,733	71,709,640	1.06×10^{-9}	257.10

Table 3: Performance on the smaller real datasets

category	algorithm	#feature		Delicious		Last.fm		Movielens	
		D	d	MAP	NDCG@5	MAP	NDCG@5	MAP	NDCG@5
DTT	DTT	32	1	0.1042	0.2317	0.4184	0.6446	0.4608	0.6028
		64	1	**0.1111**	0.2395	**0.4300**	**0.6561**	**0.4764**	**0.6188**
TD	RTF	32	-	-	-	-	-	0.3285	0.4846
		64	-	-	-	-	-	0.3474	0.5004
	HOSVD	32	-	-	-	-	-	0.0685	0.1442
		64	-	-	-	-	-	0.0790	0.1679
CP	PITF	32	-	0.1050	0.2381	0.4126	0.6463	0.4464	0.6015
		64	-	0.1059	0.2398	0.4146	0.6482	0.4503	0.6058
	CP	32	-	0.1054	0.2419	0.3808	0.6126	0.3920	0.5380
		64	-	0.1040	0.2382	0.3782	0.6115	0.3944	0.5389
popularity	popularity	-	-	0.1004	0.1581	0.3151	0.4995	0.2195	0.3405

5.4 Performance on Real Datasets

In addition to the synthetic datasets, we evaluate the performance of DTT on real datasets *Delicoius*[1], *Last.fm*[2], *Movielens*[3] and *Delicious-large*[4] [22]. The first three smaller datasets come from the 2-nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011). The larger dataset *Delicious-large* [22] contains the complete bookmarking activity for almost 2 million users from the launch of the social bookmarking website in 2003 to the end of March 2011. For each real dataset, we removed the infrequent users, items and tags by using a core-based approach [10]. For the smaller datasets, we perform the removal until every user, item and tag occurred in at least 5 triplets. For *Delicious-large*, as the original dataset is too large, we first randomly sample 50% of the users, the items and the tags, and then perform the core-based approach until every user, item and tag occurred in at least 10 triplets. The statistics of the datasets after removal are described in Table 2. *Average Candidates* means the average number of the candidate tags for each post in each dataset.

We perform 10-fold cross-validation for the smaller datasets. The comparison between DTT, PITF, CP, HOSVD, RTF and *popularity* on the smaller real datasets are showed in Table 3. As TD-based models require a lot of computation power, we conducted experiments only on the smallest real dataset *Movielens* for TD-based models, HOSVD and RTF. Although TD-based models have stronger capacity of expression than CP-based models, they work poorer than the CP-based models on the real datasets. Maybe the core tensor of TD is too complex that it leads to overfitting. Both of TD and DTT model the higher-order relations into relation tensors, but it seems DTT works much better. It can be seen from the table that the MAP of DTT is significantly higher than that of the other approaches on all the real dataset and the NDCG@5 of DTT is significantly higher than that of the other approaches on the real datasets *Last.fm* and *Movielens*.

[1]http://www.delicious.com
[2]http://www.lastfm.com
[3]http://www.grouplens.org
[4]http://www.zubiaga.org/resources/socialbm0311

In the next experiment, we split *Delicious-large* into training (90%) and test (10%). As PITF is the most competitive approach on the smaller datasets, we only compare DTT with PITF on *Delicious-large*. The results can be found in Table 4, which show that DTT outperform PITF on the larger dataset as well.

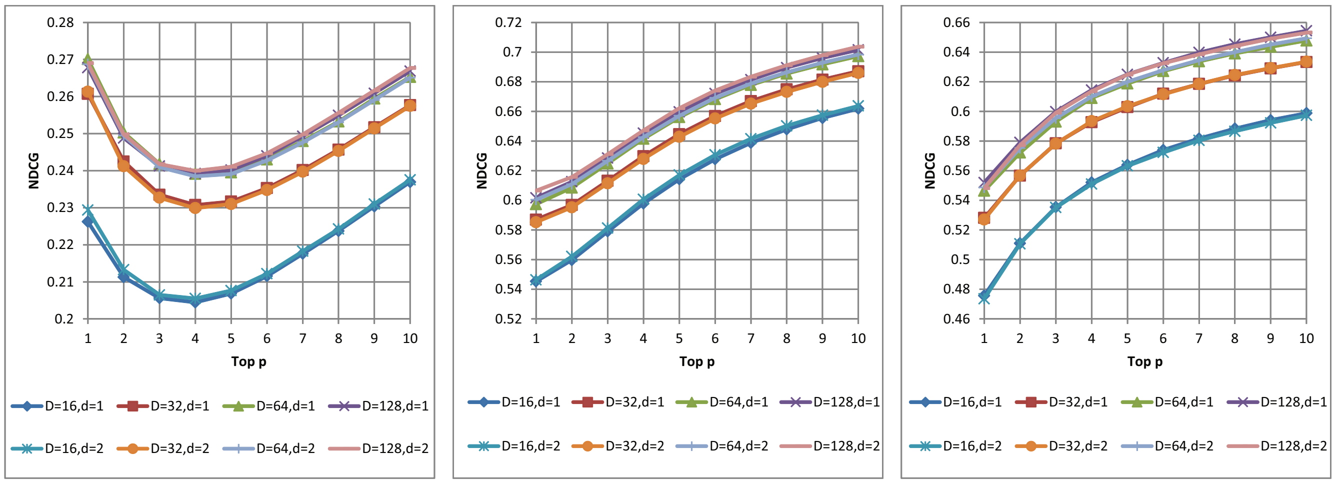
Table 4: Performance on ***Delicious-large***

algorithm	#feature		Delicious-large	
	D	d	MAP	NDCG@5
DTT	64	1	0.4674	0.6847
PITF	64	-	0.4612	0.6803

Finally, we investigate the influence of the number of features to DTT. The results with different number of features on the real datasets are showed in Figure 5. It can be seen from the figure that with the same D, the prediction quality of the models with $d = 1$ is comparable with the models with $d = 2$ on all the real datasets. d has no significant influence to the accuracy of the models on the real datasets. Thus, $d = 1$ is sufficient to capture the higher-order relations on the personalized tag recommendation datasets. As the time complexity of DTT to predict a triplet is $\mathcal{O}(Dd^2)$ and at most of the time $d = 1$, the time complexity is reduced to $\mathcal{O}(D)$. Thus, DTT is near linear and is efficient for the large-scale real problems.

6. CONCLUSION AND FUTURE WORK

We propose a novel tensor factorization technique in this paper. Different from the traditional tensor factorization techniques which decompose a tensor into factor matrices,



(a) Delicious (b) Last.fm (c) Movielens

Figure 5: Performance with different number of features on the real datasets

we decompose a tensor into factor tensors, since factor tensors can provide a more natural representation for the higher-order relations. Besides, we compare our proposed approach with two commonly used tensor factorization techniques, Tucker decomposition and CANDECOMP/PARAFAC. Our proposed approach overcome some of the drawbacks of Tucker decomposition and CANDECOMP/PARAFAC. It not only has strong capacity of expression, but also is near linear, which is feasible for the large-scale real problems. The experimental results on the synthetic datasets validate that our approach has strong capacity of expression. The results on the real datasets show that our approach outperforms other tensor factorization techniques.

In the future, we plan to further reduce the time complexity and space complexity to compute our proposed approach, by investigating its special case.

Acknowledgements

We would like to thank the many referees of the previous version of this paper for their extremely useful suggestions and comments. This work was supported by Huawei Innovation Research Program (HIRP) and National Science Foundation of China (61033010).

7. REFERENCES

[1] Carl J Appellof and ER Davidson. Strategies for analyzing data from video fluorometric monitoring of liquid chromatographic effluents. *Analytical Chemistry*, 53(13):2053–2056, 1981.

[2] J.D. Carroll and J.J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[3] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[4] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. On the best rank-1 and rank-(r 1, r 2,..., rn) approximation of higher-order tensors. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1324–1342, 2000.

[5] Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(2):10, 2011.

[6] Beyza Ermiş, Evrim Acar, and A Taylan Cemgil. Link prediction via generalized coupled tensor factorisation. *arXiv preprint arXiv:1208.6231*, 2012.

[7] Dehong Gao, Renxian Zhang, Wenjie Li, and Yuexian Hou. Twitter hyperlink recommendation with user-tweet-hyperlink three-way clustering. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2535–2538. ACM, 2012.

[8] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420, 1970.

[9] Richard A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA working papers in phonetics*, 16:1, 1970.

[10] Robert Jäschke, Leandro Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. Tag recommendations in social bookmarking systems. *Ai Communications*, 21(4):231–247, 2008.

[11] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.

[12] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[13] Fumikazu Miwakeichi, Eduardo Martınez-Montes, Pedro A Valdés-Sosa, Nobuaki Nishiyama, Hiroaki Mizuhara, and Yoko Yamaguchi. Decomposing eeg data into space–time–frequency components using

parallel factor analysis. *NeuroImage*, 22(3):1035–1045, 2004.

[14] Steffen Rendle, Leandro Balby Marinho, Alexandros Nanopoulos, and Lars Schmidt-Thieme. Learning optimal ranking with tensor factorization for tag recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 727–736. ACM, 2009.

[15] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 452–461. AUAI Press, 2009.

[16] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 811–820. ACM, 2010.

[17] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90. ACM, 2010.

[18] Stephan Spiegel, Jan Clausen, Sahin Albayrak, and Jérôme Kunegis. Link prediction on evolving data using tensor factorization. In *New Frontiers in Applied Data Mining*, pages 100–110. Springer, 2012.

[19] Jian-Tao Sun, Hua-Jun Zeng, Huan Liu, Yuchang Lu, and Zheng Chen. Cubesvd: a novel approach to personalized web search. In *Proceedings of the 14th international conference on World Wide Web*, pages 382–390. ACM, 2005.

[20] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. Tag recommendations based on tensor dimensionality reduction. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 43–50. ACM, 2008.

[21] Ledyard R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31:279–311, 1966.

[22] Robert Wetzker, Carsten Zimmermann, and Christian Bauckhage. Analyzing social bookmarking systems: A del.icio.us cookbook. In *Mining Social Data (MSoDa) Workshop Proceedings*, pages 26–30, 2008.

[23] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G Schneider, and Jaime G Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *SDM*, volume 10, pages 211–222, 2010.

[24] Nan Zheng, Qiudan Li, Shengcai Liao, and Leiming Zhang. Flickr group recommendation based on tensor decomposition. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 737–738. ACM, 2010.