Chapter 5
Related Work on Tensor Factorization

Abstract In this chapter, we provide a preliminary knowledge overview of tensors. Moreover, we provide the related work on tensor decomposition methods. The first method that is discussed is the Tucker Decomposition (TD) method, which is the underlying tensor factorization model of Higher Order Singular Value Decomposition. TD decomposes a tensor into a set of matrices and one small core tensor. The second one is the PARAFAC method (PARAllel FACtor analysis), which is the same as the TD method with the restriction that the core tensor should be diagonal. The third method is the Pairwise Interaction Tensor Factorization method, which is a special case of the TD method with linear runtime both for learning and prediction. The last method that is analyzed is the low-order tensor decomposition (LOTD) method. This method has low functional complexity, is uniquely capable of enhancing statistics, and avoids overfitting compared with traditional tensor decompositions such as TD and PARAFAC.

Keywords Tensor decomposition

5.1 Preliminary Knowledge of Tensors

Formally, a *tensor* is a multidimensional matrix. A N-order tensor $\mathcal{A}$ is denoted as $\mathcal{A} \in \mathbb{R}^{I_1 \ldots I_N}$, with elements $a_{i_1,\ldots,i_N}$. The higher order singular value decomposition [10] generalizes the singular value decomposition (SVD) computation to tensors. To apply the higher order singular value decomposition (HOSVD) technique on a third-order tensor $\mathcal{A}$, three *matrix unfolding*[1] operations are defined as follows [10]:

[1]We define as "matrix unfolding" of a given tensor the matrix representations of that tensor in which all column (row, …) vectors are stacked one after the other.

P. Symeonidis and A. Zioupos, *Matrix and Tensor Factorization Techniques for Recommender Systems*, SpringerBriefs in Computer Science,
DOI 10.1007/978-3-319-41357-0_5

$$A_1 \in \mathbb{R}^{I_1 \times (I_2 I_3)}, \qquad A_2 \in \mathbb{R}^{I_2 \times (I_1 I_3)}, \qquad A_3 \in \mathbb{R}^{(I_1 I_2) \times I_3} \tag{5.1}$$

where A_1, A_2, A_3 are called the mode-1, mode-2, mode-3 matrix unfolding of $\mathcal{A}$, respectively. The unfoldings of $\mathcal{A}$ in the three modes are illustrated in Fig. 5.1.

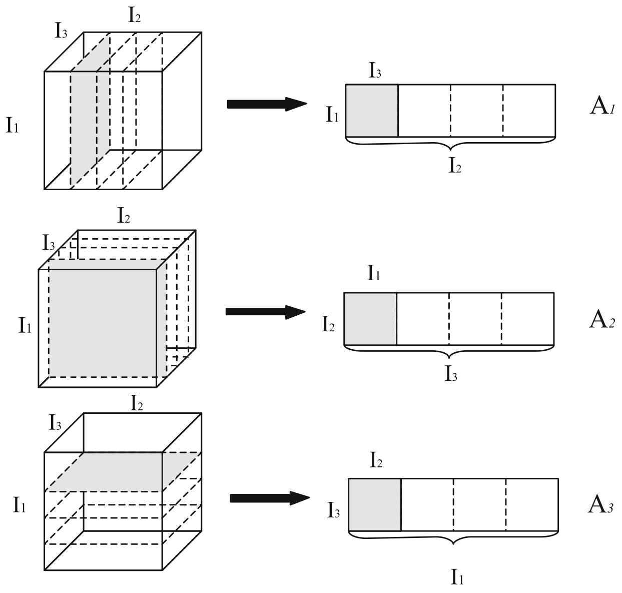


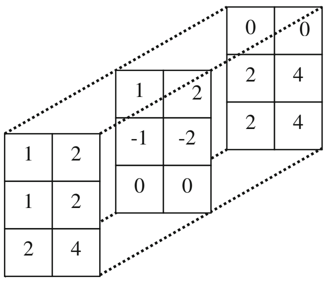
Fig. 5.1 Visualization of the three unfoldings of a third-order tensor

In the following, we will present an example of tensor decomposition adopted from [10]:

Example 1 Define a tensor $\mathcal{A} \in \mathbb{R}^{3\times2\times3}$ by $a_{1,1,1} = a_{1,1,2} = a_{2,1,1} = -a_{2,1,2} = 1$, $a_{2,1,3} = a_{3,1,1} = a_{3,1,3} = a_{1,2,1} = a_{1,2,2} = a_{2,2,1} = -a_{2,2,2} = 2$, $a_{2,2,3} = a_{3,2,1} = a_{3,2,3} = 4$, $a_{1,1,3} = a_{3,1,2} = a_{1,2,3} = a_{3,2,2} = 0$. The tensor and its mode-1 matrix unfolding $A_1 \in \mathbb{R}^{I_1 \times I_2 I_3}$ are illustrated in Fig. 5.2.

Next, we define the mode-n product of a N-order tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ by a matrix $U \in \mathbb{R}^{J_n \times I_n}$, which is denoted as $\mathcal{A} \times_n U$. The result of the mode-n product is an $(I_1 \times I_2 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N)$-tensor, the entries of which are defined as follows:

$$(\mathcal{A} \times_n U)_{i_1 i_2 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n} a_{i_1 i_2 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} u_{j_n, i_n} \tag{5.2}$$



$$
A_1 = \begin{bmatrix} 1 & 1 & 0 & 2 & 2 & 0 \\ 1 & -1 & 2 & 2 & -2 & 4 \\ 2 & 0 & 2 & 4 & 0 & 4 \end{bmatrix}
$$

Fig. 5.2 Visualization of tensor $\mathcal{A} \in \mathbb{R}^{3\times2\times3}$ and its mode-1 matrix unfolding

Since we focus on third-order tensors, $n \in \{1, 2, 3\}$, we use mode-1, mode-2, and mode-3 products.

In terms of mode-n products, SVD on a regular two-dimensional matrix (i.e., second-order tensor) can be rewritten as follows [10]:

$$
F = S \times_1 U^{(1)} \times_2 U^{(2)} \tag{5.3}
$$

where $U^{(1)} = (u_1^{(1)} u_2^{(1)} \dots u_{I_1}^{(1)})$ is a *unitary* $(I_1 \times I_1)$-matrix,[2] $U^{(2)} = (u_1^{(2)} u_2^{(2)} \dots u_{I_1}^{(2)})$ is a *unitary* $(I_2 \times I_2)$-matrix, and S is an $(I_1 \times I_2)$-matrix with the properties of:

i. pseudodiagonality: $S = \mathrm{diag}(\sigma_1, \sigma_2, \dots, \sigma_{\min\{I_1, I_2\}})$ and
ii. ordering: $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min\{I_1, I_2\}} \geq 0$.

By extending this form of SVD, HOSVD of a third-order tensor $\mathcal{A}$ can be written as follows [10]:

$$
\mathcal{A} = S \times_1 U^{(1)} \times_2 U^{(2)} \times_3 U^{(3)} \tag{5.4}
$$

where $U^{(1)}$, $U^{(2)}$, $U^{(3)}$ contain the orthonormal vectors (called the mode-1, mode-2, and mode-3 singular vectors, respectively) spanning the column space of the A_1, A_2, A_3 matrix unfoldings. S is called the core tensor and has the property of "all-orthogonality."[3] This decomposition also refers to a general factorization model known as Tucker decomposition [20].

[2]An $n \times n$ matrix U is said to be unitary if its column vectors form an orthonormal set in the complex inner product space $\mathbb{C}^n$. That is, $U^T U = I_n$.

[3]All-orthogonality means that the different "horizontal matrices" of S (the first index i_1 is kept fixed, while the two other indices, i_2 and i_3, are free) are mutually orthogonal with respect to the scalar product of matrices (i.e., the sum of products of corresponding entries vanishes); at the same time, different "frontal" matrices (i_2 fixed) and different "vertical" matrices (i_3 fixed) should be mutually orthogonal as well. For more information, see [10].

In the following, we will discuss several factorization models that have been proposed for tag recommendation. We investigate their model assumptions, complexity, and their relations among each other.

5.2 Tucker Decomposition and HOSVD

The Tucker decomposition (TD) was first introduced by Tucker [20] in 1963. The Tucker I decomposition method is an important variation of the Tucker decomposition, which is later known as HOSVD [10]. HOSVD decomposes a tensor into a set of matrices and one small core tensor. In this section, we elaborate on how HOSVD can be employed for tensor factorization in social tagging systems (STSs).

The ternary relation of users, items, and tags in STSs can be represented as a third-order tensor $\mathcal{A}$, such that tensor factorization techniques can be employed in order to exploit the underlying latent semantic structure in $\mathcal{A}$. The idea of computing low-rank tensor approximations has already been used for many different purposes [3, 9, 10, 17, 18, 22]. The basic idea is to cast the recommendation problem as a third-order tensor completion problem by completing the nonobserved entries in $\mathcal{A}$.

Formally, a social tagging system is defined as a relational structure $\mathbb{F} := (U, I, T, Y)$ in which

- U, I, and T are disjoint nonempty finite sets, whose elements are called users, items, and tags, respectively, and
- Y is the set of observed ternary relations between them, i.e., $Y \subseteq U \times I \times T$, whose elements are called tag assignments.
- A post corresponds to the set of tag assignments of a user for a given item, i.e., a triple $(u, i, T_{u,i})$ with $u \in U$, $i \in I$, and a nonempty set $T_{u,i} := \{t \in T \mid (u, i, t) \in Y\}$.

Y which represents the ternary relation of users, items, and tags can be depicted by the binary tensor $\mathcal{A} = (a_{u,i,t}) \in \mathbb{R}^{|U|\times|I|\times|T|}$ where 1 indicates observed tag assignments and 0 missing values, i.e.,

$$a_{u,i,t} := \begin{cases} 1, & (u,i,t) \in Y \\ 0, & \text{else} \end{cases}$$

Now, we express the tensor decomposition as

$$\hat{\mathcal{A}} := \hat{C} \times_u \hat{U} \times_i \hat{I} \times_t \hat{T} \tag{5.5}$$

where $\hat{U}$, $\hat{I}$, and $\hat{T}$ are low-rank feature matrices representing a mode (i.e., user, items, and tags, respectively) in terms of its small number of latent dimensions k_U, k_I, k_T, and $\hat{C} \in \mathbb{R}^{k_U \times k_I \times k_T}$ is the core tensor representing interactions between the

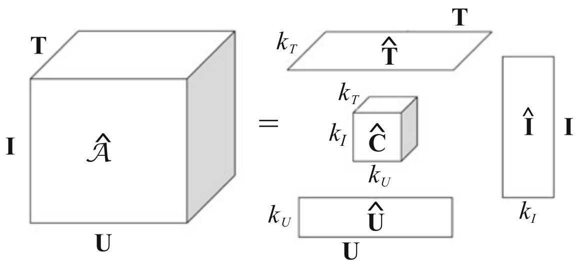


Fig. 5.3 Tensor decomposition in STS. Figure adapted from [15]

latent factors. The model parameters to be optimized are represented by the quadruple $\hat{\theta} := (\hat{C}, \hat{U}, \hat{I}, \hat{T})$ (see Fig. 5.3).

The basic idea of the HOSVD algorithm is to minimize an elementwise loss on the elements of $\hat{\mathcal{A}}$ by optimizing the square loss, i.e.,

$$\underset{\hat{\theta}}{\operatorname{argmin}} \sum_{(u,i,t)\in Y} (\hat{a}_{u,i,t} - a_{u,i,t})^2$$

After the parameters are optimized, predictions can be done as follows:

$$\hat{a}(u,i,t) := \sum_{\tilde{u}=1}^{k_U} \sum_{\tilde{i}=1}^{k_I} \sum_{\tilde{t}=1}^{k_T} \hat{c}_{\tilde{u},\tilde{i},\tilde{t}} \cdot \hat{u}_{u,\tilde{u}} \cdot \hat{i}_{i,\tilde{i}} \cdot \hat{t}_{t,\tilde{t}} \tag{5.6}$$

where $\hat{U} = [\hat{u}_{u,\tilde{u}}]_{\tilde{u}=1,\ldots,k_U}^{u=1,\ldots,U}$, $\hat{I} = [\hat{i}_{i,\tilde{i}}]_{\tilde{i}=1,\ldots,k_I}^{i=1,\ldots,I}$, $\hat{T} = [\hat{t}_{t,\tilde{t}}]_{\tilde{t}=1,\ldots,k_T}^{t=1,\ldots,T}$ and indices over the feature dimension of a feature matrix are marked with a tilde, and elements of a feature matrix are marked with a hat (e.g., $\hat{t}_{t,\tilde{t}}$).

Please notice that there are incremental solutions to update the tensor, as more data are accumulated to the system. However, please notice that the reason for the cubic complexity (i.e., $O(k^3)$ with $k := \min(k_U, k_I, k_T)$) of HOSVD is the core tensor.

5.3 AlsHOSVD

The reconstructed tensor of the previous subsection (also known as truncated HOSVD) is not optimal, but is a good starting point for an iterative alternating least squares (ALS) algorithm to best fit the reconstructed tensor to the original one [8]. The basic idea of the AlsHOSVD algorithm tries to minimize theerror between the initial and the predicted values of the tensor. The pseudocode of the approach is depicted in Algorithm 5.1.

Algorithm 5.1 AlsHOSVD

```
Require: The initial tensor $\mathcal{A}$ with user, tag, and item dimensions.
Ensure: The approximate tensor $\hat{\mathcal{A}}$ with $k_U$, $k_I$ and $k_T$ left leading eigenvectors of each dimension,
    respectively.

1: Initialize core tensor $\mathcal{C}$ and left singular vectors $U^{(1)}, U^{(2)}, U^{(3)}$ of $A_1$, $A_2$, and $A_3$, respectively.
2: repeat
3:     $\mathcal{C} = \mathcal{A} \times_1 {U_{k_U}}^{(1)^T} \times_2 {U_{k_I}}^{(2)^T} \times_3 {U_{k_T}}^{(3)^T}$
4:     $\hat{\mathcal{A}} = \mathcal{C} \times_1 {U_{k_U}}^{(1)} \times_2 {U_{k_I}}^{(2)} \times_3 {U_{k_T}}^{(3)}$
5:     ${U_{k_U}}^{(1)} \leftarrow k_U$ leading left singular vectors of $A_1$
6:     ${U_{k_I}}^{(2)} \leftarrow k_I$ leading left singular vectors of $A_2$
7:     ${U_{k_T}}^{(3)} \leftarrow k_T$ leading left singular vectors of $A_3$
8: until $\|\mathcal{A} - \hat{\mathcal{A}}\|^2$ ceases to improve OR maximum iterations reached
9: return $\mathcal{C}$, ${U_{k_U}}^{(1)}$, ${U_{k_I}}^{(2)}$, and ${U_{k_T}}^{(3)}$
```

As shown in line 8 of Algorithm 5.1, AlsHOSVD minimizes an objective function that computes the error among real and predicted values of original and approximate tensors. This is done cyclically until our objective function ceases to fit to the original values or the maximum number of user-defined iterations is reached. Please notice that values of leading left singular vectors in all three modes (lines 5–7) increased gradually in each repetition.

5.4 Parallel Factor Analysis (PARAFAC)

The PARAFAC [6] model a.k.a. canonical decomposition [2] (CANDECOMP) reduces the complexity of the TD model by assuming only a diagonal core tensor.

$$c_{\tilde{u},\tilde{i},\tilde{t}} \overset{!}{=} \begin{cases} 1, & \text{if } \tilde{u} = \tilde{i} = \tilde{t} \\ 0, & \text{else} \end{cases} \tag{5.7}$$

which allows to rewrite the model equation:

$$\hat{a}_{u,i,t} = \sum_{f=1}^{k} \hat{u}_{u,f} \cdot \hat{i}_{i,f} \cdot \hat{t}_{t,f}, \text{ for } u = 1, \ldots, U, i = 1, \ldots, I, t = 1, \ldots, T \tag{5.8}$$

In contrast to TD, model equation of PARAFAC can be computed in $O(k)$. In total, model parameters $\hat{\theta}$ of the PARAFAC model are as follows:

$$\hat{U} \in \mathbb{R}^{|U| \times k}, \quad \hat{I} \in \mathbb{R}^{|I| \times k}, \quad \hat{T} \in \mathbb{R}^{|T| \times k} \tag{5.9}$$

The assumption of a diagonal core tensor is a restriction of the TD model.

A graphical representation of TD and PARAFAC is shown in Fig. 5.4. It is seen that any PARAFAC model can be expressed by a TD model (with diagonal core tensor).

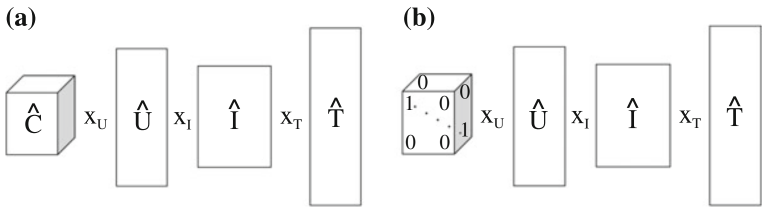


Fig. 5.4 Relationship between Tucker decomposition (TD) and parallel factor analysis (PARAFAC)

Let $\mathcal{M}$ be the set of models that can be represented by a model class. In [14], it is shown that for tag recommendation

$$\mathcal{M}^{\text{PARAFAC}} \subset \mathcal{M}^{\text{TD}} \tag{5.10}$$

This means that any PARAFAC model can be expressed with a TD model but there are TD models that cannot be represented with a PARAFAC model. In [14, 16] it was pointed out that this does not mean that TD is guaranteed to have a higher prediction quality than PARAFAC. On the contrary, as all model parameters are estimated from limited data, restricting the expressiveness of a model can lead to a higher prediction quality if the restriction is in line with true parameters.

5.5 Pairwise Interaction Tensor Factorization (PITF)

Rendle and Schmidt-Thieme [13] proposed the (PITF) model, which is a special case of the TD model with a linear runtime both for learning and prediction. PITF explicitly models pairwise interactions between users, items, and tags. Whereas TD and PARAFAC directly express a ternary relation, the idea of PITF is to model pairwise interactions instead. The motivation is that observations are typically very limited and sparse in tag recommendation data, and thus it is often easier to estimate pairwise interactions than ternary ones. This assumption is reflected in the model equation of PITF which reads:

$$\hat{a}_{u,r,t} = \sum_{f}^{k} \hat{u}_{u,f} \cdot \hat{t}^{U}_{t,f} + \sum_{f}^{k} \hat{i}_{i,f} \cdot \hat{t}^{I}_{t,f} \tag{5.11}$$

with model parameters $\hat{\theta}$

$$\hat{U} \in \mathbb{I}^{|U| \times k}, \quad \hat{I} \in \mathbb{I}^{|I| \times k}, \quad \hat{T^U} \in \mathbb{I}^{|T| \times k}, \quad \hat{T^I} \in \mathbb{I}^{|T| \times k} \tag{5.12}$$

Note that in contrast to PARAFAC, there are two factor matrices for tags: one (T^U) for the interaction of tags with users and a second one (T^I) for the interaction of tags with items.

5.6 PCLAF and RPCLAF Algorithms

In this section, we elaborate on how tensor decomposition techniques can be employed in location-based social networks (LBSNs). The ternary relation of users, locations, and activities in LBSNs can be represented as a third-order tensor.

Zheng et al. [23] introduced a personalized recommendation algorithm for LBSNs, which performs personalized collaborative location and activity filtering (PCLAF). PCLAF treats each user differently and uses a collective tensor and matrix factorization to provide personalized recommendations. As shown in Fig. 5.5, the novelty of PCLAF lies in the utilization of a user–location–activity tensor along with user–user, user–location, location–features, and activity–activity matrices.

As also shown in Fig. 5.5, to fill missing entries in the tensor $\mathcal{A}$, PCLAF decomposes $\mathcal{A}$ w.r.t. each tensor dimension (i.e., user, location, activity). Then, PCLAF forces latent factors to be shared with additional matrices to utilize their information. After such latent factors are obtained, PCLAF reconstructs an approximation tensor $\widehat{\mathcal{A}}$ by filling all missing entries. Notice that PCLAF uses a PARAFAC-style regularized tensor decomposition framework to integrate the tensor with additional matrices. In particular, Zheng et al. [23] construct a third-order tensor $\mathcal{A}$, which captures relations among users X, locations Y, activities Z, and location features U.

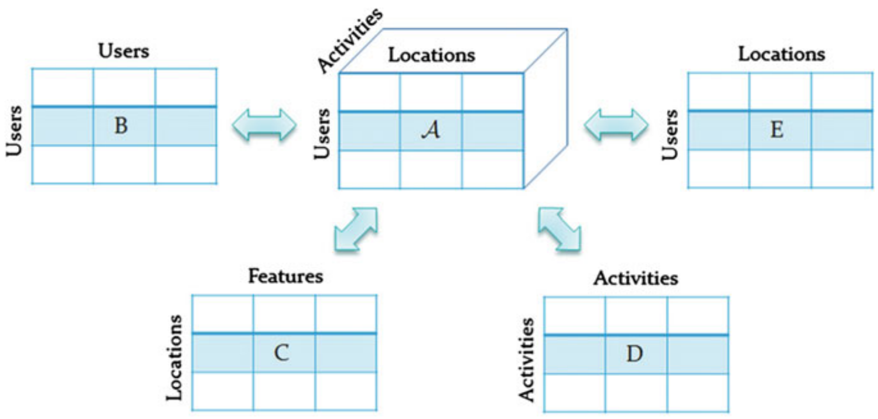


Fig. 5.5 Visual representation of a user–location–activity tensor along with user–user, user–location, location–features, and activity–activity matrices adapted from [23]

They initially decompose tensor $\mathcal{A}$ to three low-dimensional representations with respect to each tensor entity (i.e., users, locations, and activities). Then, they reconstruct the tensor trying to fill all its missing entries. To do so, they exploit additional information from user–user, location–feature, activity–activity, and location–activity matrices. They want to minimize the error between real and predicted values of the reconstructed tensor as shown in the following objective function Eq. 5.13:

$$\begin{aligned}\mathcal{L}(X, Y, Z, U) = &\frac{1}{2}\|A - [X, Y, Z]\|^2 + \frac{\lambda_1}{2} tr(X^T L_B X) \\ &+ \frac{\lambda_2}{2}\|C - YU^T\|^2 + \frac{\lambda_3}{2} tr(Z^T L_D Z) \\ &+ \frac{\lambda_4}{2}\|E - XY^T\|^2 + \frac{\lambda_5}{2}(\|X\|^2 + \|Y\|^2 + \|Z\|^2 + \|U\|^2)\end{aligned} \tag{5.13}$$

where B denotes the user–user matrix, C is the location–feature matrix, D is the activity–activity matrix, and E is the location–activity matrix. L_B and L_D are Laplacian matrices of B and D, respectively (i.e., $L_B = Q - B$ and $L_D = Q - D$, where Q is a diagonal matrix). tr denoted as the trace of a matrix. Finally, λ_i are model parameters.

In addition, Zheng et al. [24] proposed the ranking-based personalized collaborative location and activity filtering (RPCLAF). RPCLAF takes a direct way to solve the recommendation problem using a ranking loss objective function. That is, instead of minimizing the prediction error between the real and predicted user preference for an activity in a location, the RPCLAF method formulates the user's location–activity pairwise preferences by Eq. 5.14:

$$\theta_{u,l,a,a'} := \begin{cases} +1, & \textbf{if } \mathcal{A}_{u,l,a} > \mathcal{A}_{u,l,a'} \mid (u, l, a) \in I_i \wedge (u, l, a') \notin I_i; \\ 0, & \textbf{if } \mathcal{A}_{u,l,a} = \mathcal{A}_{u,l,a'} \mid (u, l, a) \in I_i \wedge (u, l, a') \in I_i; \\ -1, & \textbf{if } \mathcal{A}_{u,l,a} < \mathcal{A}_{u,l,a'} \mid (u, l, a) \notin I_i \wedge (u, l, a') \in I_i; \\ ?, & \textbf{if } (u, l, a) \notin I_i \vee (u, l, a') \notin I_i \end{cases} \tag{5.14}$$

where I_i denotes location–activity pairwise preferences of user i in tensor $\mathcal{A}$, $\mathcal{A}_{u,l,a}$ denotes the preference of user u on the activity a that she performed in location l, whereas $\mathcal{A}_{u,l,a'}$ denotes the preference for user u on the activity k' that she performed in location l. Based on Eq. 5.14, RPCLAF distinguishes between positive and negative location–activity pairwise preferences and missing values to learn a personalized ranking of activities/locations. The idea is that positive $(+1)$ and negative examples (-1) are only generated from observed location–activity pairwise preferences. Observed location–activity pairwise preferences are interpreted as positive feedback $(+1)$, whereas nonobserved location–activity pairwise preferences are marked as negative (-1) feedback. All other entries are assumed to be either missing (?) or zero values.

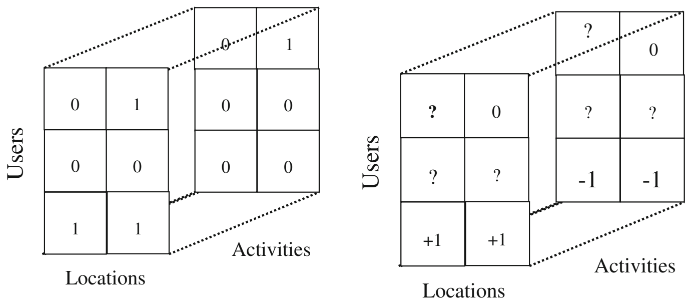


Fig. 5.6 *Left* Tensor's data representation of the HOSVD algorithm (*left*) and the RPCLAF algorithms (*right*)

To give a more clear view of tensor representation based on ranking, in Fig. 5.6, we compare tensor representation of the HOSVD [19] algorithm, with the tensor representation of RPCLAF.

The left-hand side of Fig. 5.6 shows the tensor representation of the HOSVD algorithm [19], where the positive feedback is interpreted as 1 and the rest as 0. The right-hand side of Fig. 5.6 shows the tensor representation of the RPCLAF algorithm where observed location–activity pairwise preferences are considered positive feedback (+1), while nonobserved location–activity pairwise preferences are marked as negative feedback (−1). All other entries are either missing (?) or zero values. For example, in the right-hand side of Fig. 5.6, the value of tensor element $\mathcal{A}_{3,1,1}$ is $+1$, because it holds $\mathcal{A}_{3,1,1} > \mathcal{A}_{3,1,2}$, whereas the value of tensor element $\mathcal{A}_{3,1,2} = -1$ because $\mathcal{A}_{3,1,2} < \mathcal{A}_{3,1,1}$.

5.7 Other Tensor Decomposition Methods

A drawback of TD models such as HOSVD is the fact that the construction of the core tensor requires cubic runtime in factorization dimension for both prediction and learning. Moreover, they suffer from sparsity that incurs in STSs and LBSNs. To overcome the aforementioned problem, HOSVD can be performed efficiently following the approach of Sun and Kolda [7]. Other approaches to improve the scalability to large data sets are through slicing [21] or approximation [4]. Rendle et al. [12] proposed ranking with tensor factorization (RTF), a method for learning optimal factorization of a tensor for a specific problem of tag recommendations. Moreover, Cai et al. [1] proposed LOTD, which also targets the very sparse data problem for tag recommendation. Their LOTD method is based on low-order polynomials that present low functional complexity. LOTD is capable of enhancing statistics and avoids overfitting, which is a problem of traditional tensor decompositions such as Tucker and PARAFAC decompositions. It has been experimentally shown [1] with extensive experiments on several data sets that LOTD outperforms PITF and other methods in terms of efficiency and accuracy. Another method which outperformed PITF is proposed by Gemmell et al. [5]. Their method builds a weighted hybrid tag recommender that blends multiple recommendation components drawing separately on complementary dimensions. Moreover, Leginus et al. [11] improved tensor-based recommenders with clustering. They reduced the tag space by exploiting clustering techniques so that both the quality of recommendations and the execution time are improved.

References

1. Cai, Y., Zhang, M., Luo, D., Ding, C., Chakravarthy, S.: Low-order tensor decompositions for social tagging recommendation. In: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining (WSDM'11), pp. 695–704. ACM (2011)
2. Carroll, J.D., Chang, J.J.: Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. Psychometrika **35**, 283–319 (1970)
3. Chen, S., Wang, F., Zhang, C.: Simultaneous heterogeneous data clustering based on higher order relationships. In: Proceedings of the Seventh IEEE International Conference on Data Mining Workshops, ICDMW'07, pp. 387–392, Washington, DC, USA. IEEE Computer Society (2007)
4. Drineas, P., Mahoney, M.W.: A randomized algorithm for a tensor-based generalization of the singular value decomposition. Linear Algebra Appl. **420**(2–3), 553–571 (2007)
5. Gemmell, J., Schimoler, T., Mobasher, B., Burke, R.: Hybrid tag recommendation for social annotation systems. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pp. 829–838. ACM (2010)
6. Harshman, R.A.: Foundations of the parafac procedure: models and conditions for an 'exploratory' multimodal factor analysis. In: UCLA Working Papers in Phonetics, pp. 1–84 (1970)
7. Kolda, T., Sun, J.: Scalable tensor decompositions for multi-aspect data mining. In: Proceedings of the 8th IEEE International Conference on Data Mining (ICDM'08), pp. 363–372 (2008)
8. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009)
9. Kolda, T.G., Sun, J.: Scalable tensor decompositions for multi-aspect data mining. In: ICDM'08: Proceedings of the 8th IEEE International Conference on Data Mining, pp. 363–372. IEEE Computer Society, Dec 2008
10. De Lathauwer, L., De Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl. **21**(4), 1253–1278 (2000)
11. Leginus, M., Dolog, P., Žemaitis, V.: Improving tensor based recommenders with clustering. In: User Modeling, Adaptation, and Personalization Conference (UMAP 2012), pp. 151–163. Springer (2012)
12. Rendle, S., Marinho, L.B., Nanopoulos, A., Thieme, L.S.: Learning optimal ranking with tensor factorization for tag recommendation. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09), pp. 727–736 (2009)
13. Rendle, S., Thieme, L.S.: Pairwise interaction tensor factorization for personalized tag recommendation. In: Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM'10), pp. 81–90 (2010)
14. Rendle, S.: Context-Aware Ranking with Factorization Models, 1st edn. Springer, Berlin, Heidelberg (2010)
15. Rendle, S., Marinho, L.B., Nanopoulos, A., Schimdt-Thieme, L.: Learning optimal ranking with tensor factorization for tag recommendation. In: KDD'09: Proceedings of the 15th ACM

SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 727–736. ACM (2009)
16. Rendle, S., Schmidt-Thieme, L.: Pairwise interaction tensor factorization for personalized tag recommendation. In: WSDM'10: Proceedings of the Third ACM International Conference on Web Search and Data Mining. ACM (2010)
17. Shashua, A., Hazan, T.: Non-negative tensor factorization with applications to statistics and computer vision. In: ICML'05: Proceedings of the 22nd International Conference on Machine Learning, pp. 792–799. ACM (2005)
18. Sun, J., Shen, D., Zeng, H., Yang, Q., Lu, Y., Chen, Z.: Cubesvd: a novel approach to personalized web search. In: World Wide Web Conference, pp. 382–390 (2005)
19. Symeonidis, P., Papadimitriou, A., Manolopoulos, Y., Senkul, P., Toroslu, I.: Geo-social recommendations based on incremental tensor reduction and local path traversal. In: Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Location-Based Social Networks (LBSN), Chicago, IL, pp. 89–96 (2011)
20. Tucker, L.: Some mathematical notes on three-mode factor analysis. Psychometrika 279–311 (1966)
21. Turney, P.: Empirical evaluation of four tensor decomposition algorithms. Technical report (NRC/ERB-1152) (2007)
22. Wang, H., Ahuja, N.: A tensor approximation approach to dimensionality reduction. Int. J. Comput. Vis. 217–229 (2007)
23. Zheng, V., Cao, B., Zheng, Y., Xie, X., Yang, Q.: Collaborative filtering meets mobile recommendation: a user-centered approach. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI), Atlanta, GA (2010)
24. Zheng, V., Zheng, Y., Xie, X., Yang, Q.: Towards mobile intelligence: learning from GPS history data for collaborative recommendation. Artif. Intell. **184–185**, 17–37 (2012)