Knowl Inf Syst (2013) 34:243–265
DOI 10.1007/s10115-011-0460-y

REGULAR PAPER

Non-negative Tri-factor tensor decomposition with applications

Zhong-Yuan Zhang · Tao Li · Chris Ding

Received: 10 July 2010 / Revised: 24 August 2011 / Accepted: 15 November 2011 /
Published online: 1 March 2012

Abstract Non-negative matrix factorization (NMF) mainly focuses on the hidden pattern discovery behind a series of vectors for two-way data. Here, we propose a tensor decomposition model Tri-ONTD to analyze three-way data. The model aims to discover the common characteristics of a series of matrices and at the same time identify the peculiarity of each matrix, thus enabling the discovery of the cluster structure in the data. In particular, the Tri-ONTD model performs adaptive dimension reduction for tensors as it integrates the subspace identification (i.e., the low-dimensional representation with a common basis for a set of matrices) and the clustering process into a single process. The Tri-ONTD model can also be regarded as an extension of the Tri-factor NMF model. We present the detailed optimization algorithm and also provide the convergence proof. Experimental results on real-world datasets demonstrate the effectiveness of our proposed method in author clustering, image clustering, and image reconstruction. In addition, the results of our proposed model have sparse and localized structures.

Keywords Non-negative tensor decomposition · Non-negative matrix factorization

Z.-Y. Zhang
School of Statistics, Central University of Finance and Economics,
Beijing, People's Republic of China

T. Li (✉)
School of Computing and Information Sciences,
Florida International University,
11200 SW 8th Street, Miami, FL 33199, USA
e-mail: taoli@cs.fiu.edu

C. Ding
Computer Science and Engineering Department,
University of Texas at Arlington, Arlington, TX, USA

1 Introduction

1.1 Tensor data and tensor decomposition

Recently, analyzing three-way data (or three-way tensor) has attracted a lot of attention due to the intrinsic rich structures in real-world datasets [1,22,28,38,41,51]. Three-way data are generalizations of matrices and they appear in many applications. One typical type of three-way data is multiple two-way data/matrices with different time periods, for example, a series of 2-D images, 2-D text data (documents vs. terms), or 2-D microarray data (genes vs. conditions) are naturally represented as three-way data. In particular, in document clustering, the data over different time periods can be represented as a three-way dataset as author × terms × time; in email communications, the data can be represented as sender × receiver × time; in web page personalization, the data can be represented as user × query word × web-page; in high-order web link analysis, the data are represented as a three-way dataset as web page× web page × anchor text [37].

An example of three-way data is shown in Fig. 1 with three modes: data units, features, and occasions. The cth *frontal slice* of the three-way data is $X_c \in \mathbb{R}^{d_1,d_2}$ by holding the last mode of $\underline{\mathbf{X}}$ fixed at c. There are d_3 frontal slices lined up as shown on the left side of Fig. 1. The three-way data can be *matricized* in the first mode to form a flattened matrix as shown on the right side of Fig. 1.

One way to analyze three-way data is to convert the three-way data into two-way matrices, either by transforming the three-way data into the sum-up matrix $\left(\mathbf{X} = \sum_{\mathbf{i}}(\mathbf{X_i})\right)$ where X_i is the ith front slice or by matricization. After the conversion, traditional low-rank matrix factorization techniques including singular value decomposition (SVD) [17,18], principle component analysis (PCA) [6,44], factor analysis [23,34], independent component analysis [24], and non-negative matrix factorization (NMF) [29,31,32,36,45] can then be applied. However, the conversion may result in information loss and fail to capture the underlying structures in three-way datasets [1]. Many extensions of matrix computation techniques have been proposed to efficiently analyze three-way data for different purposes [3,13,20,26,27,40,54].

We note that there are generally two types of tensor decomposition models including: (1) **Parafac** (parallel factor analysis [21]): The Parafac model can be thought as a multilinear form of decomposition for the objective tensor. Specially for a 3-D tensor, each entry of the three-way tensor is approximated by a linear combination of three vectors [21]. It only allows the same number of factors in each mode, and the ith factor in one mode only interacts with the ith factors in other modes (e.g., one-to-one interactions). (2) **Tucker** [8,46]: The

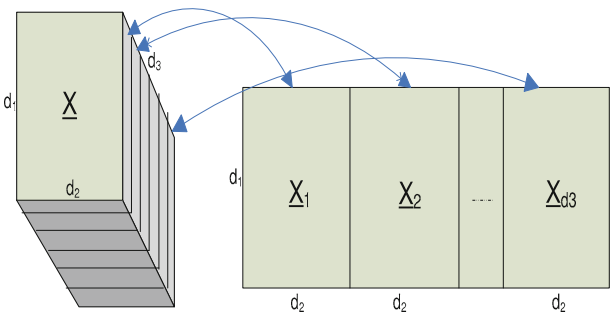

Fig. 1 Three-way data $\underline{\mathbf{X}} \in \mathbb{R}^{d_1,d_2,d_3}$ matricized in the first mode

Tucker model such as HOSVD (higher-order singular value decomposition) can be thought as the multi-way principal component analysis and aims to give the optimal low-rank approximation of a tensor in given dimensions. Many multi-way models can be considered as the extensions or modifications of the above two types [1,25,27,40,54]. These multi-way models have been applied to many applications such as web link analysis [28], webpage personalization [43], social network analysis [22], and others [2,41,48].

1.2 Content of the paper

In this paper, we present a new tensor decomposition model called Tri-ONTD (Tri-factor orthogonal non-negative tensor decomposition). The model aims to discover the common characteristics of a series of input matrices (the common characteristics will be represented using two basis matrices) and at the same time identify the peculiarity of each matrix (the peculiarity will be represented using a tensor, see Sect. 3). As a result, the cluster structure of the data can be discovered based on the specialties of the input matrices (samples), i. e., the samples that have similar specialties are grouped into one cluster. In other words, with the Tri-ONTD model, we obtain two basis matrices which are used to represent the common characteristics of the objective matrices (samples) and one 3-D tensor which is used to denote the peculiarities of the samples. The model can be used to perform hard clustering, soft clustering, and dimensional reduction. The non-negative constraints in the model can improve the interpretability of the decomposition results [29].

We emphasize two properties of our Tri-ONTD model: (1) The Tri-ONTD model is an extension of adaptive dimension reduction on tensors. Generally, the dimension reduction is carried out as a preprocessing step and is decoupled from the clustering process: once the subspace dimensions are selected, they stay fixed during the clustering process. Adaptive dimension reduction refers to the approach where the subspace is adaptively adjusted and integrated with the clustering process [4,5,10,16,33,49]. Our Tri-ONTD model integrates the subspace identification (i.e., the low-dimensional representation with a common basis for a set of matrices) and the clustering process into a single process. (2) The Tri-ONTD model can also be viewed as an extension of the Tri-ONMF (Tri-factor orthogonal non-negative matrix factorization) model [12]. In [12], Tri-ONMF has been applied to document co-clustering successfully, but it can only cope with static data, i.e., it cannot be used for analyzing tensor data such as a series of document data across several years. The Tri-ONTD model is an extension of the Tri-ONMF model on tensors.

Our Tri-ONTD model differs from the multi-linear form of tensor factorization in that Tri-ONTD explicitly maintains the 2-D nature of the original 2-D items (i.e., image and document term) while in the multi-linear case, different dimensions are of different natures. In other words, the Tri-ONTD model maintains the 2-D nature of the original 2-D items (e.g., 2-D image structures), while the multi-linear form of decompositions does not. In the multi-linear case, each entry of the original 3-D tensor is approximated by a linear combination of three vectors, each of which has its own meanings (e.g., unit, variable, or occasion).

Our Tri-ONTD model essentially belongs to the Tucker type but also differs from HOSVD in that additional constraints (e.g., non-negativity) are enforced in the model so that the Tri-ONTD model has better interpretability and is more intuitive [35,52]. We compared the Tri-ONTD model with 2DSVD (a special form of HOSVD with the form $X_{ijk} = \sum_{i'j'} U_{ii'} V_{jj'} S_{i'j'k}$, where U and V are matrices and $\underline{\mathbf{S}}$ is a 3-D tensor [13]) in our experiments. The experimental results demonstrated the effectiveness of our Tri-ONTD model.

The rest of the paper is organized as follows: Sect. 2 presents an overview of various tensor decomposition models. Section 3 introduces the detailed model formulation of the

Tri-ONTD model. The corresponding algorithms and the convergence proof are introduced in Sect. 4. Section 5 shows the experiments on author clustering, image clustering, and image reconstruction. Finally, Sect. 6 concludes.

2 An overview on tensor decomposition

The notations used in the paper are summarized in Table 1. We will mainly use the matrix representations and occasionally explicit indexes to denote three-way tensors.

In this section, we give a brief overview of the related tensor decomposition models. Here, we focus on the model formulations.

1. **2DSVD** [13]: 2DSVD is an extension of the classic singular value decomposition (SVD) [17,18]. Different from SVD (which is computed for a set of vectors), 2DSVD is defined for a set of matrices. We can write the model as:

$$\text{2DSVD:} \quad X_{l\pm} \approx W_{\pm} M_{l\pm} R_{\pm}^T, \tag{1}$$

for $l = 1, \ldots, L$, where $W = (u_1, u_2, \ldots, u_k)$ and $R = (v_1, v_2, \ldots, v_s)$. u_i is the eigenvector of $F = \sum_i X_i R R^T X_i^T$, and v_i is the eigenvector of $G = \sum_i X_i^T W W^T X_i$. Note that $W^T W = I$, $R^T R = I$.

2. **NTD (Non-negative tensor decomposition** [40]): NTD is an extension of NMF where the input datum is a non-negative tensor. For a three-way tensor, the standard NTD can be written as:

$$\text{NTD:} \quad \underline{\mathbf{X}}_{+} \approx \sum_{i=1}^{K} \otimes_{j=1}^{3} W_{i+}^{j}, \tag{2}$$

where elements of X, and W^j are non-negative. $\otimes$ denotes the outer product and W^j is a matrix, each column of which is W_i^j. Given two vectors $a \in R^m, b \in R^n, a \otimes b = C \in R^{m \times n}$, where $C_{ij} = a_i b_j$.

3. **OTD (orthogonal tensor decomposition** [27,54]): Similar to NTD, the OTD model for a 3-D tensor can be formulated as:

$$\text{OTD:} \quad \underline{\mathbf{X}}_{\pm} \approx \sum_{i=1}^{K} \otimes_{j=1}^{3} W_{i\pm}^{j}. \tag{3}$$

Table 1 Notations used in the paper

$\underline{\mathbf{X}} = (X_1, \ldots, X_L)$	Tensor with L front slices
$(X_l)_{ij} = X_{ijl}$	The ij-th entry of X_l
$\underline{\mathbf{X}}_{+}$	Non-negative tensor $\underline{\mathbf{X}}$
$\underline{\mathbf{X}}_{\pm}$	Tensor with mixed signs
X_{l+}	Non-negative matrix $X_l \geq 0$
$X_{l\pm}$	Matrix with mixed signs
$X = (x_1, \ldots, x_m)$	Matrix with m column
x_i	Vector
I	Identity matrix

Instead of adding non-negative constraints, the columns of each matrix W^j are required to be mutually orthogonal.

4. **Tri-ONTD (Tri-factor orthogonal non-negative tensor decomposition)**: The Tri-ONTD model is an extension of the Tri-ONMF (Tri-Factor Orthogonal Non-negative matrix factorization) model introduced in [12]. Tri-ONMF is a variation of NMF and can be written as:

$$\text{Tri-ONMF:} \quad X_+ \approx F_+ S_+ G_+^T, \tag{4}$$

where $F^T F = I$ and $G^T G = I$. Tri-ONMF provides a framework for simultaneously clustering of rows and columns of an input matrix X: F and G indicate the cluster memberships for rows and columns of X, respectively. S, as an additional freedom degree, is used to absorb the different scales of X, F, and G. The Tri-ONMF model also bears similarity with double K-means [15,19,50] as they both perform simultaneous clustering of rows and columns.

Extending the Tri-ONMF model to a 3-D tensor, we have the Tri-ONTD model as follows:

$$\text{Tri-ONTD:} \quad X_{l+} \approx U_+ S_{l+} V_+^T, \tag{5}$$

for $l = 1, 2, \ldots, n$. U and V are the *common basis* for $\{X_l | l = 1, 2, \ldots, n\}$, and $\{S_l | l = 1, 2, \ldots, n\}$ indicates the peculiarity of each X_l. Note that the slices S_l's are not assumed to be diagonal as in the Parafac model [21] but are the slices of a general core array as in the Tucker3 model [46]. The size of U is $m \times t$, the size of V is $n \times s$, and both U and V are orthogonal. The common basis means the basis that represents the common characteristics of the slices in the input tensor. Each slice S_l is the projection of the original corresponding slice X_l in the input tensor onto the common basis.

3 Model formulation

3.1 Simultaneous subspace selection and data clustering

Consider a set of input data vectors $X = (x_1, \cdots, x_n)$. A long-held standard practice is to use PCA (principle component analysis) to project the data into a low-dimensional space:

$$Y = (y_1, \cdots, y_n) = (U^T x_1, \cdots, U^T x_n) = U^T X,$$

and then perform K-means clustering on Y. Recently, it has been shown that NMF is equivalent to K-means and provides a more versatile and often better clustering model than K-means [9,11]. Using NMF as the clustering model, the clustering after PCA subspace selection can be written as:

$$\min_{C,H} ||Y - CH^T||_F^2, \ s.t.\ C, H \geq 0, H^T H = I. \tag{6}$$

Here, H is the clustering membership indicator matrix, and each row of C is the cluster centroid.

We note that this common practice separates the subspace selection and data clustering as two unrelated procedures. Typically, dimension reduction is carried out as a preprocessing step and is decoupled from the clustering process: once the subspace dimensions are selected, they stay fixed during the clustering process. This would lead to poor clustering performance for high-dimensional data, where different clusters often exist in different low-dimensional subspaces [33].

Recently, some research efforts such as the adaptive dimension reduction techniques have integrated these two procedures into a single process, i.e., the clustering process is integrated with the subspace selection process, and the data are then simultaneously clustered while the feature subspaces are selected [4,5,10,16,33,49]. This can be written as:

$$\min_{C,H,U} ||U^T X - CH^T||_F^2, \ s.t.\ H \geq 0, H^T H = I, U^T U = I. \tag{7}$$

We call this SSC (simultaneous subspace identification and clustering) factorization. Assuming $UU^T = I$, the model can be written in a different form:

$$\min_{C,H,U} ||X - UCH^T||_F^2, \ s.t.\ H \geq 0, H^T H = I, U^T U = I. \tag{8}$$

The most important difference between Eqs. (7) and (8) is that in Eq. (7), data and the cluster centroids are all in the reduced subspace, while in Eq. (8), only cluster centroids are in the subspace. Note that: (1) $U^T U = I$ can be relaxed to $U^T U = D$, where D is a diagonal matrix; and (2) SSC can also be viewed as an extension of factorial K-means with a weighted matrix norm [49].

3.2 The Tri-ONTD model

Now, we extend the SSC factorization to tensors. Given a 3-D tensor, say a set of 2-D images, we denote them as

$$\underline{\mathbf{X}} = (X_1, \ldots, X_L),$$

where X_i is a $m \times n$ matrix.

First, we consider the subspace projection. In this case, we seek two sets of basis U and V, and the low-dimensional representation S_i, $i = 1, \ldots, L$:

$$\min_{U,V,S_l} \sum_{l=1}^{L} ||X_l - U S_l V^T||_F^2, s.t., V^T V = I, U^T U = I. \tag{9}$$

The size of U is $m \times t$, and the size of V is $n \times s$. In the subspace, we have $\underline{\mathbf{S}} = (S_1, \cdots, S_L)$, where S_l is a $t \times s$ matrix. $t \times s$ is the size of the low dimension (subspace) representation. We then perform K-means clustering on them. Let $\underline{\mathbf{C}} = (C_1, \cdots, C_K)$ represent the centroids of the K clusters, where C_i is also a $t \times s$ matrix. Viewing S_l, C_i as vectors in a $t \times s$-dimensional space, we can write down the K-means clustering objective function as follows:

$$\begin{aligned} O &= \sum_{k=1}^{K} \sum_{l \in cluster\ k} ||S_l - C_k||_F^2 \\ &= \sum_{l=1}^{L} \sum_{k=1}^{K} H_{lk} ||S_l - C_k||_F^2 \\ &= \sum_{l=1}^{L} \sum_{H_{lk} \neq 0} ||S_l - C_k||_F^2 \\ &= \sum_{l=1}^{L} ||S_l - \sum_{k=1}^{K} C_k H_{lk}||_F^2, \end{aligned} \tag{10}$$

where $H \in R^{L\times K}$ is the cluster membership indicator matrix, each row of which has and only has one nonzero element 1. Note that the above optimization is similar to the tandem analysis described in [39]. Equation (10) can be written in the complete index:

$$O = \sum_{l=1}^{L}\sum_{i=1}^{k}\sum_{j=1}^{s}\left(S_{ijl} - \sum_{k=1}^{K} C_{ijk}H_{lk}\right)^2.$$

Similar to SSC, we combine the two optimization problems in Eqs. (9) and (10) into one process, and thus, the Tri-ONTD model can be written as:

$$\begin{aligned} \min_{C,H,U,V} \sum_{l} \left\| X_l - U\sum_{k=1}^{K}(C_k H_{lk})V^T \right\|_F^2, \\ \text{s.t. } U^TU = I, V^TV = I, H^TH = I, \\ U, V, H, C \geq 0. \end{aligned} \tag{11}$$

If one only wants to identify the optimal subspace, solving Eq. (9) is enough for subspace identification. It should be pointed out that the results of our proposed model are essentially unique except for column/row permutations. The proof follows directly from the Proposition 1 in [9].

3.3 Extension of Tri-ONMF

The Tri-ONTD model can be regarded as an extension of the Tri-ONMF model. The two basis U and V identify the common characteristics of the series of matrices that are composing the tensor $\underline{\mathbf{X}}$, while the core tensor $\underline{\mathbf{S}}$ gives the peculiarity of each slice of $\underline{\mathbf{X}}$. Based on $\underline{\mathbf{S}}$, we can cluster each slice of $\underline{\mathbf{X}}$ into different groups. In other words, we factorize $\underline{\mathbf{S}}$ into $\underline{\mathbf{C}}$ and H, where $\underline{\mathbf{C}}$ is the centroid tensor and H is the cluster membership indicator. Figure 2 illustrates the relation between our Tri-ONTD model and the Tri-ONMF model.

3.4 An illustrative example

We close this section by a small example to show the effectiveness of our tensor decomposition model. We try to cluster twelve matrices into different classes. The first six slices of the tensor are the same as: $\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix}$; while the remaining six slices of the tensor are the same as $\begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$. Clearly, the twelve matrices can be separated into two clusters. We randomly flip some elements of the matrices(i.e., change some elements from 0 to 1 or from 1 to 0) to obtain a noisy version of the tensor. When applying NMF, we write the tensor as a matrix form, i.e., reshape each slice of the tensor into one column of a matrix

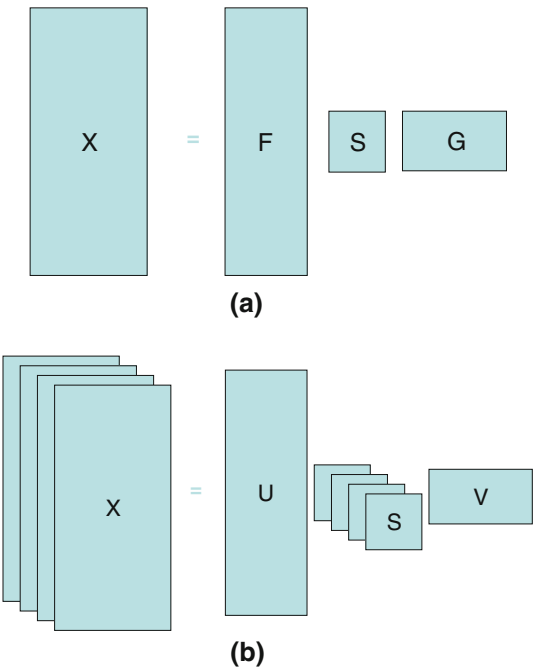

Fig. 2 An intuitive illustration of the differences between Tri-ONTD and Tri-ONMF: **a** Tri-ONMF; **b** Tri-ONTD

$$X = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}.$$

When we apply NMF to obtain the columns' cluster membership information, the second and the sixth columns, and the columns from 7th to 12th are of one cluster while the rest are of the other cluster. The tensor model can discover the correct cluster structure, while NMF cannot. The centroids of the two clusters are the following:

$$C_1 = \begin{pmatrix} 0.65 & 1.00 \\ 0.79 & 0.00 \end{pmatrix}, \quad C_2 = \begin{pmatrix} 0.066 & 1.59 \\ 0.054 & 1.79 \end{pmatrix}.$$

4 Algorithm procedure and analysis of convergence

4.1 Algorithm procedure

In most cases, we only want to identify the optimal basis U and V for a series of matrices in order to discover the common characteristics of the matrices. Under this consideration, the

model can be written as:

$$\min \ J = \sum_l ||X_l - U S_l V^T||_F^2, \tag{12}$$

$$\text{s. t. } U^T U = I, V^T V = I, \tag{13}$$

$$U, V, \underline{\mathbf{S}} \geq 0. \tag{14}$$

As we can see, all the elements of U, V, and S are variables that need to be determined. In other words, we need to find the optimal matrices U, S, and V so that the objective function J achieves its local minimum. It will be very time-consuming if we employ the standard optimization methods.

We then follow the multiplicative update strategy that is widely used in solving NMF. In other words, instead of iteratively searching for the optimal solutions of U, S, and V, we alternatively update one of the three matrices while fixing the other two. We derive the update rules from the Karush-Kuhn-Tucker (KKT) condition. Note that the KKT condition is the necessary condition for local optimal solutions in non-linear programming, provided that some constraint qualification such as LICQ (linear independence constraint qualification, the gradients of the active constraints are linearly independent at the optimal point) is satisfied.

From non-linear optimization, the Lagrangian function of the constrained programming is:

$$\begin{aligned} L &= tr\left[\sum_l (X_l - U S_l V^T)^T (X_l - U S_l V^T)\right] \\ &\quad + tr(\lambda(U^T U - I) + \xi(V^T V - I)) + \sum_{ij} (\delta_{ij} U_{ij} + \gamma_{ij} V_{ij} + \theta_{ij} S_{ij}) \\ &= \sum_l tr[X_l^T X_l - 2 X_l^T U S_l V^T + V S_l^T U^T U S_l V^T] \\ &\quad + tr(\lambda(U^T U - I) + \xi(V^T V - I)) + \sum_{ij} (\delta_{ij} U_{ij} + \gamma_{ij} V_{ij} + \theta_{ij} S_{ij}), \end{aligned}$$

where $\lambda, \xi, \delta, \gamma, \theta$ are the matrices of corresponding Lagrangian multipliers of the five constraints. We rewrite $V S_l^T$ as G_l, and then, the first part of Lagrangian function can be written as:

$$\sum_l tr\left[X_l^T X_l - 2 X_l^T U G_l^T + G_l^T U^T U G_l^T\right].$$

So, the gradient of L with respect to U is:

$$\frac{\partial L}{\partial U} = 2 \sum_l (-X_l G_l + U G_l^T G_l) + 2 U \lambda + \delta. \tag{15}$$

From the constraint $U \geq 0$, the complementary condition of U is $\delta_{ab} U_{ab} = 0$ for any a, b. Using this condition, we have:

$$\left(-\sum_l X_l G_l + \sum_l U G_l^T G_l + U \lambda\right)_{ab} U_{ab} = 0, \tag{16}$$

which gives the update rule of U while fixing V and S. Similarly, we can obtain the update rules of V and S. We summarize the rules of U, V, and S as follows:

$$U_{ab} := U_{ab}\sqrt{\frac{\sum_l [X_l V S_l^T]_{ab}}{\sum_l (U U^T X_l V S_l^T)_{ab}}}; \tag{17}$$

$$V_{ab} := V_{ab}\sqrt{\frac{\sum_l [X_l^T U S_l]_{ab}}{\sum_l (V V^T X^T U S_l)_{ab}}}; \tag{18}$$

$$S_{abl} := S_{abl}\sqrt{\frac{(U^T X_l V)_{ab}}{(U^T U S_l V^T) V_{ab}}}. \tag{19}$$

Consequently, we can prove that the update rules converge to a local minima of the Lagrangian function L. We have the following monotonic property theorem:

Theorem 1 (Convergence and Correctness) *The Lagrangian function L is non-increasing under the update rules in Eqs.* (17), (18) *and* (19).

The proof is given in the Appendix. Note that as similar to most NMF algorithms, the algorithm of the Tri-ONTD model only has the monotonic property, and hence, the global optimal solution is not guaranteed. Though the algorithm often falls into a suboptimal stationary point, the experimental results of clustering and the image reconstruction are stable as demonstrated in Sect. 5.

4.2 Hard clustering

As we have discussed earlier, U and V are the common basis of a series of matrices X, and $\underline{\mathbf{S}}$ shows the specialty of each slice of X. We can cluster the series of matrices X into K classes based on S. Note that usually the number of clusters K is less than the number of slices L. To perform clustering, we factorize $\underline{\mathbf{S}}$ into $\underline{\mathbf{C}}$ and H so that $S_l = \sum_{k=1}^{K} C_k H_{lk}$ for $l = 1, \cdots, L$, where H is of size $L \times K$ and satisfies $H^T H = I$, and $\underline{\mathbf{C}}$ is a tensor of size $d_1 \times d_2 \times K$ which can be viewed as centroids of the clusters. Formally, the model can be written as:

$$\min \quad J = \sum_l \left\| X_l - U \sum_{k=1}^{K} (C_k H_{lk}) V^T \right\|_F^2, \tag{20}$$

$$\text{s. t.} \quad U^T U = I, V^T V = I, H^T H = I, \tag{21}$$

$$U, V, H, \underline{\mathbf{C}} \geq 0. \tag{22}$$

The updating rules of U and V are the same as those in Sect. 4.1.

The update rule of $\underline{\mathbf{C}}$ is:

$$C_{abc} = C_{abc} \frac{\sum_l (U^T X_l V)_{ab} H_{lc}}{\sum_l \left\{ U^T \left[U \left(\sum_{i=1}^{K} C_i H_{li} \right) V^T \right] V \right\}_{ab} H_{lc}}. \tag{23}$$

Since H is the cluster membership indicator, there is only one nonzero element 1 in each row of H under hard clustering. Hence, H can be updated via a greedy strategy: at each iteration, from $i = 1$ to L, $H_{ip^*} = 1$ if $p^* = argmin_p J$ for $H_{ip} = 1$ while $H_{ip'} = 0$, $p' \neq p$. The algorithm procedure for hard clustering is described in Table 2.

Table 2 The algorithm procedure of Tri-ONTD: hard clustering case

Input: The data tensor $\underline{\mathbf{X}}$, the positive integers d_1, d_2 and K.

Output: Left basis (feature basis) U,
Right Basis (sample basis) V,
The centroid tensor $\underline{\mathbf{C}}$,
The cluster membership indicator H.

Method:

Step 1. **Initialize** U, V, $\underline{\mathbf{C}}$ and H
with random positive numbers in [0,1].

Step 2. **Repeat:**

1. Fixing V, $\underline{\mathbf{C}}$ and H, update U;
2. Fixing U, $\underline{\mathbf{C}}$ and H, update V;
3. Fixing U, V and H, update $\underline{\mathbf{C}}$;
4. Fixing U, V and $\underline{\mathbf{C}}$, update H using a greedy strategy;

Until Convergence.

4.3 Soft clustering

The model can be naturally extended to perform soft clustering. Different from hard clustering, each slice of a tensor $\underline{\mathbf{X}}$ can be grouped into multiple clusters with a certain probability. The only modification of the algorithm is the updating rule of H. Instead of using a greedy strategy, H is updated based on the gradient descent similar to that of U, V, and $\underline{\mathbf{C}}$.

$$H_{ab} := H_{ab} \frac{\sum_{t,j} (X_a)_{tj} (U C_b V^T)_{tj}}{\sum_{t,j} [U(C_b H_{ab}) V^T]_{tj} (U C_b V^T)_{tj} + \sum_j H_{aj}}. \tag{24}$$

4.4 Time complexity

If the sizes of U, V, and $\underline{\mathbf{S}}$ are $m \times t$, $n \times s$, and $t \times s \times L$, respectively, then the time complexity for updating U in Eq. (17) is of order $mt + 2m^2t + Lmns + Lmst$, the time complexity for updating v in Eq. (18) is of order $ns + 2n^2s + Lmnt + Lnst$, and the time complexity for updating $\underline{\mathbf{S}}$ in Eq. (19) is $L(ts + mnt + nts + mt^2 + ns^2 + st^2 + ts^2)$. The time complexity for updating $\underline{\mathbf{C}}$ (of size $d_1 \times d_2 \times K$) in Eq. (23) is of order $K(d_1 d_2 + L(mnd_1 + nd_1 d_2 + 2d_1 d_2 + md_1^2 + nd_2^2 + d_1 d_2 K + d_1 d_2^2 + d_2 d_1^2))$. A similar analysis can also be found in [11]. In practice, the values of t (or d_1), s (or d_2), and K are often very small, and hence, these matrix multiplications can be computed efficiently on most cases.

5 Applications

In this section, we evaluate our Tri-ONTD model with different real-world applications: document applications (author clustering) and image applications (image clustering and reconstruction). For each application, we compare our proposed model with the popular and the state-of-the-art techniques used in the specific application. Note that documents and images are of different types with different data characteristics. Hence, the state-of-art techniques used for different applications might be different. For author clustering, we compare our proposed model with several tensor factorization methods such as Parafac and HOSVD which

can be directly used for data clustering. We also compare our proposed model with other popular two-way document clustering methods by matricizing the input tensor data. For image applications, since NMF, SVD, and 2D-SVD are the popular techniques used in image clustering and reconstruction, we compare our proposed model with them.

5.1 Clustering of authors

5.1.1 Dataset description

We performed author clustering on a dataset extracted from the DBLP record file that can be downloaded at *http://www.informatik.uni-trier.de/~ley/db/*. DBLP is a bibliography website that indexes up-to-date papers with the associated conferences or journals in the computer science field. We extracted author names, publication titles, and the corresponding years of the publications. Among these records, 1000 active researchers with their publication titles for the last 20 years (from 1988 to 2007) were chosen for our experiment. These researchers and their publications were divided into 9 different research areas: *Database*, *Data Mining*, *Software Engineering*, *Theory*, *Computer Vision*, *Operating System*, *Machine Learning*, *Networking*, and *Natural Language Processing* based on authors' major activities in these areas. These different areas were served as the ground-truth labels for our experimental comparison purpose. The data were preprocessed by using the standard text preprocessing techniques. For each year, a binary matrix with each entry denoting the co-occurrence of the corresponding author and the term in that year was constructed. The data were represented as a three-way tensor with the author, term, and year modes.

Each slice of the tensor is a author-by-term matrix, where the terms are words that used in the paper titles. Recall that in our Tri-ONTD model, the size of U is $m \times t$, the size of V is $n \times s$, and the size of $\underline{\mathbf{S}}$ is $t \times s \times L$. In the author clustering application, we set $m = 1{,}000$, $n = 500$, $L = 20$, $t = 9$, and $s = 9$. The goal of author clustering is to find the cluster information of authors from the basis matrix U, i.e., author i is of cluster j if the element u_{ij} of matrix U is the largest in ith row.

5.1.2 Performance measures

In order to compare the clustering performance, we used normalized mutual information (NMI) and accuracy(ACC) as our performance measures. These measures provide good insights on how the clustering results agree with the true label. Normalized mutual information measures how clustering results share the information with the ground-truth label [42]. Generally, the larger the NMI value, the better the clustering quality is. Its value is between [0, 1]. The NMI of the entire clustering solution is computed as:

$$\text{NMI} = \frac{\sum_{i,j} P(i,j) log_2 \frac{P(i,j)}{P(i)P(j)}}{\sqrt{\left(\sum_i -P(i) log_2 P(i)\right)\left(\sum_j -P(j) log_2 P(j)\right)}}, \tag{25}$$

where $P(i)$ is the probability that an arbitrary data point belongs to cluster i, and $P(j)$ is the probability that an arbitrary data point belongs to ground-truth class j. $P(i, j)$ is the joint probability that an arbitrary data point belongs to cluster i and also class j. Note that NMI is actually the mutual information between clustering and ground-truth class knowledge divided by the maximum value of clustering entropy and class entropy.

Accuracy (ACC) discovers the one-to-one relationship between clusters and classes and measures the extent to which each cluster contains data points from the corresponding class. It sums up the whole matching degree between all class-cluster pairs. Its value is also between [0, 1]. Accuracy can be represented as:

$$\mathrm{ACC} = \max\left(\sum_{C_k, L_m} T(C_k, L_m)\right) \Big/ N, \tag{26}$$

where C_k denotes the kth cluster, and L_m is the mth class. $T(C_k, L_m)$ is the number of entities that belong to class m, but are assigned to cluster k. Accuracy computes the maximum sum of $T(C_k, L_m)$ for all pairs of clusters and classes, and these pairs have no overlaps. Generally, the greater the accuracy, the better the clustering performance.

5.1.3 Comparison methods

The clustering performance of Tri-ONTD was compared with a wide range of clustering algorithms, and we expect these comparisons would provide us with enough insights into the performance of Tri-ONTD. The comparison methods include:

- Tensor factorization methods: We compared Tri-ONTD with **Parafac** [20] and **HOSVD** [46], two commonly used multi-way analysis models. The clustering results were derived as follows: given the component matrix W from Parafac and HOSVD, we discretize it by solving $\arg\min_{H,R} ||W - HR||$, where H is fixed to indicator matrix and R is rotation matrix, as in [47].
- Two-way data clustering methods
  - (a) **KMeans(sum)**: Run K-Means algorithm [53] on the sum-up matrix of 20 years author $\times$ terms matrices;
  - (b) **KMeans(ext)**: Run K-Means algorithm on the unfolded matrix in the first mode of the three-way array;
  - (c) **KMeans(PCA)**: Perform PCA first to reduce the dimensionality of the unfolded matrix in the first mode and then use K-Means algorithm;
  - (d) **InfoCo**: Run information theoretic co-clustering algorithm [14] on the sum-up matrix of 20 years author$\times$terms matrices;
  - (e) **EuclCo**: Run Euclidean co-clustering algorithm [7] on the sum-up matrix;
  - (f) **MinSqCo**: Performs minimum squared residue co-clustering algorithm [7] on the sum-up matrix.
- **ClusterAgg**: Run K-Means clustering on each frontal slice of the three-way array, and combine them using clustering aggregation [42]. We used the hypergraph partitioning algorithm for cluster aggregation.

5.1.4 Analysis of results

In the experiments, the clustering quality was obtained by averaging ten trials. The accuracy and NMI results are presented in Figs. 3 and 4, respectively. We observed that Tri-ONTD achieved the obviously better clustering performance than other algorithms. **KMeans(PCA)** and three co-clustering algorithms performed relatively better since the dimensionality reduction step made them viable on clustering high-dimensional data. **KMeans(ext)** obtained very poor clustering results because it was easily affected by noise dimensions.

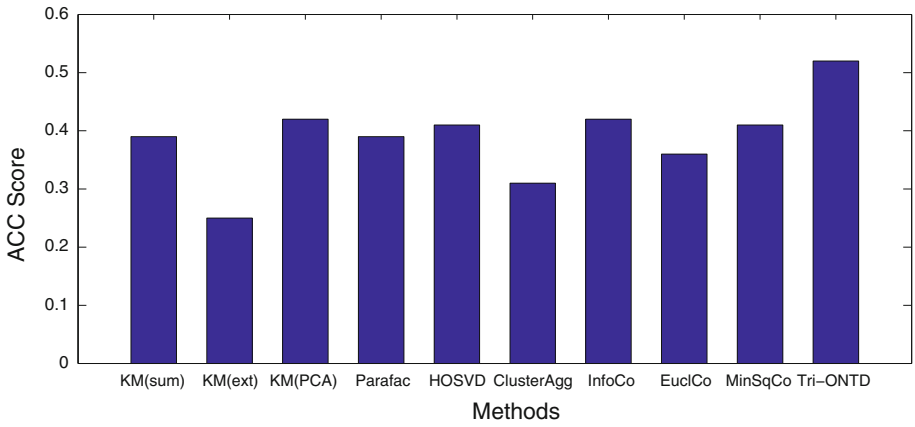

Fig. 3 Accuracy comparison for author clustering

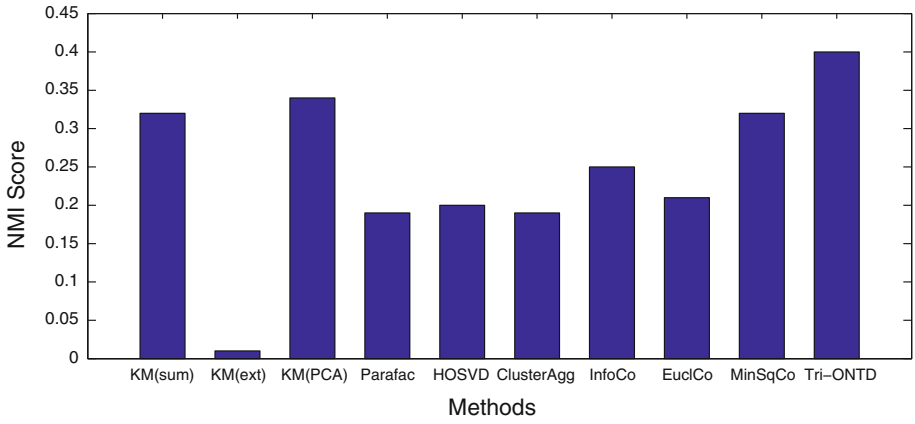

Fig. 4 NMI comparison for author clustering

To gain more insights into the Tri-ONTD model, Table 3 lists the terms whose corresponding coefficients are the largest in the columns that are highly related to the factors (i.e., columns) in U (such as Factor 1, Factor 2 and Factor 6). After checking the author information, we know that Factor 1 is dominated by authors in data mining, Factor 2 is dominated by authors in database, and Factor 6 is dominated by authors in theory. We observe that the discovered terms are highly related to the author clusters.

5.2 Image clustering

We used the **CBCL face** dataset[1] to evaluate the clustering performance of Tri-ONTD on image datasets. The CBCL dataset contains two classes of data: face and non-face. The size of each image is 19×19 (i.e., $m = 19$ and $n = 19$). The goal of image clustering is to cluster the images into two different classes: face and non-face.

[1] http://cbcl.mit.edu/software-datasets/FaceData2.html.

Table 3 The most highly related terms to Factor 1 (Data Mining), Factor 2 (Database), and Factor 6 (Theory)

Authors	Terms		
Factor 1	Mining	Data	Cluster
Data mining	Pattern	Discovery	Rule
	Frequent		
Factor 2	Web	Xml	Database
Database	Management	System	Access
	Document	Service	
Factor 6	Algorithm	Problem	Approximation
Theory	Time	Complexity	Graph

The results are based on the DBLP dataset

Table 4 Performance comparison of Non-negative matrix factorization (NMF) and Non-negative Orthogonal Tensor Decomposition (Tri-ONTD) on the CBCL dataset

	Accuracy	NMI	Memory storage
Data size = 50			
NMF	0.68	0.01	822 ($19 \times 19 \times 2 + 50 \times 2$)
Tri-ONTD ($t = s = 2$)	0.90	0.53	184 ($19 \times 2 \times 2 + 2 \times 2 \times 2 + 50 \times 2$)
Tri-ONTD ($t = s = 3$)	0.90	0.53	194 ($19 \times 2 \times 2 + 3 \times 3 \times 2 + 50 \times 2$)
Tri-ONTD ($t = s = 5$)	0.90	0.53	226 ($19 \times 2 \times 2 + 5 \times 5 \times 2 + 50 \times 2$)
Data size = 100			
NMF	0,56	0	922
Tri-ONTD ($t = s = 2$)	0.84	0.37	284
Tri-ONTD ($t = s = 3$)	0.84	0.37	294
Tri-ONTD ($t = s = 5$)	0.84	0.37	326
Data size = 200			
NMF	0.62	0.04	1,122
Tri-ONTD ($t = s = 2$)	0.74	0.19	484
Tri-ONTD ($t = s = 3$)	0.74	0.19	494
Tri-ONTD ($t = s = 5$)	0.74	0.19	526
Data size = 300			
NMF	0.65	0.07	1,322
Tri-ONTD ($t = s = 2$)	0.74	0.12	684
Tri-ONTD ($t = s = 3$)	0.74	0.12	694
Tri-ONTD ($t = s = 5$)	0.74	0.12	726

We compared our method with NMF (which has been successfully applied to image clustering). To apply NMF, we converted the tensor into matrices by re-arranging each slice of the tensor into a column vector. In addition to the clustering performance comparison, we also compared the memory usage of different methods. Note that for image clustering, we usually chose $t = s$ in low-dimensional subspace representations [13]. In our experiments, three different values (e.g., 2,3, and 5) were used for t and s.

To systematically study the clustering performance under different parameters, we performed the experiments on four subsets of the CBCL dataset. The four subsets have different sizes ranging from 50 to 300, in other words, $L = 50, 100, 200, 300$. Table 4 shows the performance results and the memory storage of the corresponding methods. The memory

Table 5 Performance comparison of non-negative matrix factorization (NMF) and non-negative orthogonal tensor decomposition (Tri-ONTD) on a subset of the ORL dataset (including 80 images)

	Accuracy	NMI	Storage
NMF	0.8750	0.8271	83072
Tri-ONTD	0.8960	0.8305	5,500

For Tri-ONTD, $t = s = 15$

storage was calculated as the number of matrix elements used during the computation. From Table 4, we observed that the clustering performance of Tri-ONTD was consistently better than that of NMF. For example, the accuracy value was significantly improved from 0.66 to 0.9 when the dataset size is 50. In general, the performance of Tri-ONTD was about 10% better than that of NMF. In addition, the clustering performance of Tri-ONTD did not vary too much when the dimensions of the subspaces changed. Table 5 presents the experimental results on ORL dataset.

Tri-ONTD is able to capture the 2-D nature of the images while converting the tensor into a matrix has changed the intrinsic structure of the image. Hence, Tri-ONTD outperforms NMF in the experiments. Moreover, Tri-ONTD is able to discover the common characteristics of the series of matrices and thus has the smaller memory usage than NMF (as it only needs to store the specialty for each image).

5.3 Image reconstruction

We used the **ORL Database of Faces**[2] to evaluate Tri-ONTD's ability of image reconstruction. The ORL dataset is a well-known dataset in image processing society. It contains 400 face images of 40 persons, each for 10 images including different poses. The size of each image is 112×92 (i.e., $m = 112$ and $n = 92$).

In this experiment, we compared Tri-ONTD with 2DSVD.[3] We also compared Tri-ONTD with two other methods: (1) One is SVD: each image X_i was factorized into U_i, $\sum_i$ and V_i so that $X_i = U_i \sum_i V_i^T$, the top r singular values of each $\sum_i$ were selected to reconstruct the image X_i, the cost function is $\sum_{i=1}^{L} \left(X_i - \left(\sigma_i^1 U_i^1 V_i^{1T} + \cdots + \sigma_i^r U_i^r V_i^{rT}\right)\right)$ where U_i^r and V_i^r are the rth columns of the matrices U_i and V_i, respectively. (2) Another one is matrix_SVD: a new matrix X, each column of which is a image, was constructed first, then SVD was applied to X to obtain U, Σ, and V, the top d singular values of Σ were selected to produce the reconstructed matrix $\bar{X}$, and finally each column of $\bar{X}$ was rearranged back as a matrix to approximate the corresponding image. The cost function is $||X - \bar{X}||_F^2$.

Figure 5 shows the comparisons of reconstruction errors for SVD, matrix_SVD, 2DSVD, and Tri-ONTD on ORL dataset. The size of each slice in the core tensor of 2DSVD is $p \times q$ and that of the Tri-ONTD model is $r \times s$. We observed that SVD had the largest reconstruction errors and 2DSVD had the smallest. In addition, the errors of Tri-ONTD were between those of SVD and 2DSVD. This was because we imposed non-negative constraints on the Tri-ONTD model. In fact, we also observed that the reconstruction error difference between 2DSVD and Tri-ONTD was small.

Table 6 shows the comparison of the memory storage. The number of top r singular values of SVD was selected from 5 to 15, the number of top d singular values of matrix_SVD was

[2] http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

[3] Note that the cost function of 2DSVD is $\sum_l ||X_l - L M_l R^T||_F^2$.

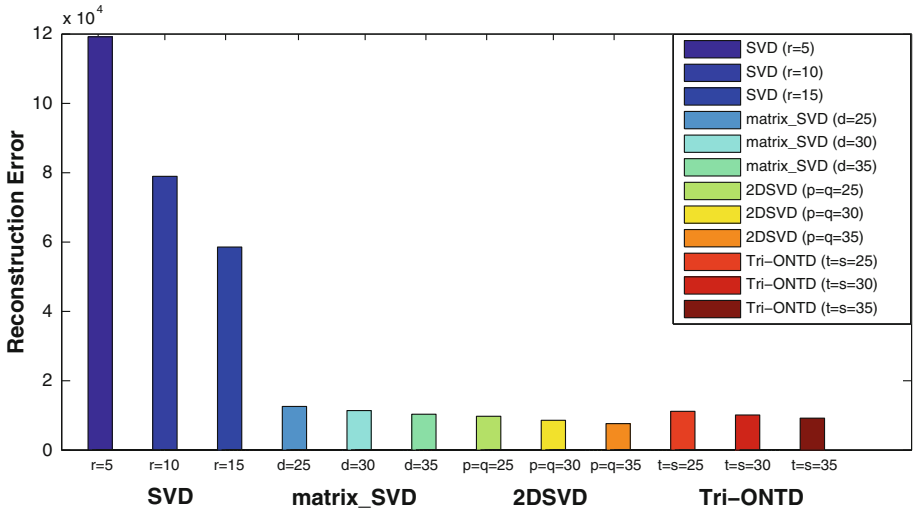

Fig. 5 Comparison of reconstruction errors on the ORL dataset. Note that the sizes of the reduced matrices (tensors) are different. Here, r and d are the numbers of selected singular values in SVD and matrix_SVD, respectively. k is the size of each slice of the resulted core tensors in 2DSVD and the Tri-ONTD, respectively, i.e., $p = q = k, r = s = k$

Table 6 Memory storage comparison of SVD, matrix_SVD, 2DSVD, and Tri-ONTD

SVD	$r = 5$	$r = 10$	$r = 15$
	83,600	171,200	262,800
Matrix_SVD	$d = 25$	$d = 30$	$d = 35$
	260,225	312,420	364,665
2DSVD	$p = q = 25$	$p = q = 30$	$p = q = 35$
	55,100	78,120	105,140
Tri-ONTD	$t = s = 25$	$t = s = 30$	$t = s = 35$
	55,100	78,120	105,140

The entries shown are the number of matrix elements used during the computation

selected from 25 to 35, the size of matrices S_l in Tri-ONTD (e.g., the dimensions of the subspace) ranged from 25×25 to 35×35, and the size of matrices M_l in 2DSVD ranged from 25×25 to 35×35. Note that 2DSVD and Tri-ONTD had similar memory usage as both methods made use of the common basis for the low-dimensional representation. We observed that matrix_SVD had the largest memory storage while Tri-ONTD and 2DSVD used the smallest memory storage.

Figure 6 illustrates eight reconstructed images selected from the ORL database. The first row was reconstructed by Tri-ONTD with $t = s = 35$, and the second row was reconstructed by Tri-ONTD with $t = s = 25$. The third row was reconstructed by 2DSVD when the size of matrices M_l is 35×35, and the fourth row was reconstructed by 2DSVD when the size of matrices M_l is 25×25. The fifth and the sixth rows were obtained by SVD with $r = 15$ and $r = 5$, respectively. The seventh row was reconstructed by matrix_SVD with $d = 25$. From Fig. 6, we observed: (1) If all the methods used nearly the same memory storage (e.g., the second row: Tri-ONTD; the fourth row: 2DSVD; and the sixth row: SVD), Tri-ONTD and 2DSVD produced better reconstruction results. (2) When all the methods gave comparable

Fig. 6 Eight images reconstructed by Tri-ONTD (the *first row*, $t = s = 35$; the *second row*, $t = s = 25$), by 2DSVD (the *third row*, 35×35; the *fourth row*, 25×25), by SVD (the *fifth row*, $r = 15$; the *sixth row*, $r = 5$), and by matrix_SVD (the *seventh row*, $r = 25$). The tensor of Tri-ONTD is composed of 80 images ($L = 80$), each of which is one pose of a person

Table 7 Sparsity comparison of 2DSVD and Tri-ONTD

Subspace size	Methods	Left basis	Right basis	Core tensor
35×35	2DSVD	L 100%	R 100%	**M** 100%
	TRi-ONTD	U **10.26%**	V **27.5%**	**S** 100%
30×30	2DSVD	L 100%	R 100%	**M** 100%
	Tri-ONTD	U **8.75%**	V **10.58%**	**S** 100%
25×25	2DSVD	L 100%	R 100%	**M** 100%
	Tri-ONTD	U **8.9%**	V **8.6%**	**S** 100%

The entries shown are the percentage of the elements whose absolute values are larger than 10^{-10}. Hence, the smaller percentage values imply the better sparsity of the results

reconstruction results (e.g., the first row: Tri-ONTD; the third row: 2DSVD; the fifth row: SVD), SVD used more memory storage.

We also compared the sparsity of 2DSVD and Tri-ONTD as shown in Table 7. We observed that the results of Tri-ONTD had sparse structures, while the results of 2DSVD did not.

Hence, Tri-ONTD identified more localized information for image reconstruction. From above analysis, we have the following observations: (1) Tri-ONTD and 2DSVD give the best reconstruction images if different methods use same memory storage; (2) Tri-ONTD and 2DSVD need the smallest memory storage, while matrix_SVD needs the largest in the case of comparatively good image reconstruction ability; (3) the results of Tri-ONTD have sparse structure, while those of 2DSVD have not, so Tri-ONTD can identify the more localized structure and save the memory storage.

6 Conclusion

In this paper, we present the Tri-ONTD model to analyze 3-way tensors. The Tri-ONTD model performs adaptive dimension reduction on tensors as it integrates the subspace identification and the clustering process into a single process. Thus, the Tri-ONTD model is able to discover the common characteristics of a series of matrices while identifying the peculiarity of each matrix at the same time. The Tri-ONTD model can be regarded as an extension of the Tri-factor non-negative matrix factorization model. Three real-world applications: author clustering, image clustering, and image reconstruction were conducted to demonstrate the effectiveness of the proposed model.

Acknowledgments The work of Z. Zhang is supported by the Foundation of Academic Discipline Program at Central University of Finance and Economics. The work of T. Li is partially supported by NSF grants IIS-0546280, DMS-0915110, and CCF-0830659. The work of C. Ding is partially supported by NSF grants DMS-0915228 and CCF-0830780.

Appendix: Proof of convergence

We adopt the auxiliary function method that was introduced by [30] into the convergence proof of NMF algorithms.

Definition 6.1 (*Auxiliary Function*) Function $Z(H, H')$ is called an auxiliary function of $L(H)$ if it satisfies

$$Z(H, H') \geq L(H), Z(H, H) = L(H)$$

for any H, H'.

The following Lemma 1 establishes the key property of auxiliary function that is useful in convergence proof.

Lemma 1 (Lee & Seung) *If $Z(H, H')$ is an auxiliary function of $L(H)$, then $L(H)$ is non-increasing under the update*

$$H^{t+1} = arg \min_{H} Z(H, H^t).$$

We first prove the convergence of update rule in Eq. (17) by constructing an auxiliary function of $L(U)$. The convergence of the update rules in Eqs. (18) and (19) can be proved similarly.[4] To do this, we follow the proving steps used in [12]. We show that the conclusion can be reached as an extension of the convergence results of [12].

[4] In fact, only the convergence of rules in Eqs. (17) and (18) need to be proved. The proof of the rule in Eq. (19) is the same as that in [12] since S_l is only related to X_l when U and V are fixed.

Proposition 1 *For any matrices $A \in R_+^{n\times n}$, $B \in R_+^{k\times k}$, $S \in R_+^{n\times k}$, $S' \in R_+^{n\times k}$, where A, B are symmetric, the following inequality holds*

$$\sum_{i=1}^{n}\sum_{p=1}^{k}\frac{(AS'B)_{ip}S_{ip}^2}{S'_{ip}} \geq Tr(S^TASB).$$

If we regard the Lagrangian function L as a function of U, $\sum_l tr(X_l^T X_l)$, $tr(\lambda D)$ and $\xi(V^TV - D)$ are all constants and can be ignored. Thus, the Lagrangian function L can be written as:

$$\begin{aligned} L(U) &= \sum_l tr[-2U^T X_l G_l + G_l^T G_l U^T U] + tr\lambda U^T U \\ &= \sum_l tr(-2U^T X_l G_l) + tr\left(\sum_l G_l^T G_l + \lambda\right)U^T U. \end{aligned}$$

Theorem 2 *The following function is an auxiliary function of $L(U)$.*

$$\begin{aligned} Z(U,U') = &-\sum_{lik} 2(X_l G_l)_{ik}U'_{ik}\left(1 + log\frac{U_{ik}}{U'_{ik}}\right) \\ &+\sum_{ik}\frac{\left[U'\left(\sum_l G_l^T G_l + \lambda\right)\right]_{ik}U_{ik}^2}{U'_{ik}}. \end{aligned}$$

Proof Firstly, when $F = F'$, the equality $Z(U,U') = L(U)$ holds. Secondly, when $F \neq F'$, the first part of $Z(U,U')$ is always not bigger than that of $L(U)$ from the inequality $z \geq 1 + log(z)$ for any $z > 0$; the second part of $Z(U,U')$ is always not smaller than that of $L(U)$. This can be obtained from Proposition 1 where we let $A = I$, $B = \sum_l G_l^T G_l + \lambda$, $S = U$ and $S' = U'$. To summarize, $Z(U,U') \geq L(U)$ when $F \neq F'$. Thus, the theorem is proved. □

Theorem 2 gives an auxiliary function of $L(U)$. We can observe that $Z(U,U')$ is a convex function of U, so the gradient of U is zero if and only if at the global minimum point:

$$\begin{aligned} \frac{\partial Z(U,U')}{\partial U_{ab}} &= 0 \\ &= -2\sum_l \frac{U'_{ab}}{U_{ab}}(X_l G_l)_{ab} + 2\frac{\left[U'\left(\sum_l G_l^T G_l + \lambda\right)\right]_{ab}U_{ab}}{U'_{ab}}. \end{aligned}$$

Solving for the global minimum solution of U, we have:

$$U_{ab} := U'_{ab}\sqrt{\frac{\sum_l (X_l G_l)_{ab}}{\left[U'\left(\sum_l G_l^T G_l + \lambda\right)\right]_{ab}}}.$$

Now, we try to determine the Lagrangian multiplier matrix λ. The diagonal elements of λ can be obtained by summing over index a of Eq. (16):

$$\left[-U^T\left(\sum_l X_l G_l\right)\right]_{bb} + \left[U^T U\sum_l G_l^T G_l\right]_{bb} + (U^T U)_{bb}\lambda_{bb} = 0.$$

Using the constraint $U^T U - D = 0$, we get:

$$\left[-U^T\left(\sum_l X_l G_l\right)\right]_{bb} + \left[D\sum_l G_l^T G_l\right]_{bb} + D_{bb}\lambda_{bb} = 0,$$

so $\lambda_{bb} = [D^{-1}U^T(\sum_l X_l G_l) - \sum_l (G_l^T G_l)]_{bb}$ for any b. The off-diagonal elements of λ can be approximately computed by eliminating the non-negative constraint of U. Under this relaxed condition, by computing the gradient of the Lagrangian function with respect of U, we have:

$$\lambda_{ab} = \left(D^{-1}U^T\sum_l X_l G_l - G_l^T G_l\right)_{ab}, \quad a \neq b.$$

So, the Lagrangian multiplier λ has an uniform formulation. The update rule of U can then be written as a more compact formulation:

$$U_{ab} := U_{ab}\sqrt{\frac{\sum_l [X_l V S_l^T]_{ab}}{\sum_l (U U^T X_l V S_l^T)_{ab}}}. \tag{27}$$

From Lemma 1 and the above analysis, we have the conclusion that the Lagrangian function L is non-increasing under the update rules in Eqs. (17), (18) and (19).

References

1. Acar E, Yener B (2007) Unsupervised multiway data analysis: a literature survey. Technical report, Computer Science Department, Rensselaer Polytechnic Institute
2. Acar E, Camtepe SA, Krishnamoorthy M, Yener B (2005) Modeling and multiway analysis of chatroom tensors. In: Proceedings of IEEE international conference on intelligence and security informatics. Lecture Notes in Computer Science
3. Bader B, Harshman R, Kolda T (2006) Analysis of latent relationships in semantic graphs using DEDICOM invited talk at the workshop on Algorithms for Modern Massive Data Sets
4. Bock HH (1986) On the interface between cluster analysis, principal components, and multidimensional scaling. In: Proceedings of advances symposium on multivariate modelling and data analysis. Reidel Publishing Co., Dordrecht, pp 17–34
5. Bolton RJ, Krzanowski WJ (2003) Projection pursuit clustering for exploratory data analysis. J Comput Graph Stat 12:121–142
6. Buntine W, Perttu S (2003) Is multinomial pca multi-faceted clustering or dimensionality reduction. In: Proceedings of 9th international workshop on artificial intelligence and statistics, pp 300–307
7. Cho H, Dhillon I, Guan Y, Sra S (2004) Minimum sum squared residue co-clustering of gene expression data. In: Proceedings of SIAM data mining conference
8. De Lathauwer L, De Moor B, Vandewalle J (2000) A multilinear singular value decomposition. SIAM J Matrix Anal Appl 21(4):1253–1278
9. Ding C, He X, Simon H (2005) On the equivalence of nonnegative matrix factorization and spectral clustering. In: Proceedings of SIAM data mining conference
10. Ding C, Li T (2007) Adaptive dimension reduction using discriminant analysis and k-means clustering. In: ICML, pp 521–528
11. Ding C, Li T, Jordan Michael I (2010) Convex and semi-nonnegative matrix factorizations. IEEE Trans Pattern Anal Mach Intell 32(1):45–55
12. Ding C, Li T, Peng W, Park H (2006) Orthogonal nonnegative matrix tri-factorizations for clustering. In: SIGKDD, pp 126–135
13. Ding C, Ye JP (2005) 2-Dimensional singular value decomposition for 2D maps and images. In: Proceedings of SIAM data mining conference
14. Dhillon IS, Mallela S, Modha DS (2003) Information-theoretical co-clustering. In: SIGKDD, pp 89–98
15. DeSarbo WS (1982) GENNCLUS: new models for general non-hierarchical clustering analysis. Psychometrika 47:449–475

16. De Soete G, Carroll JD (1994) K-means Clustering in a Low-dimensional Euclidean Space. In: New approaches in classification and data analysis. Springer, Heidelberg, pp 212–219
17. Eckart C, Young G (1936) The approximation of one matrix by another of lower rank. Psychometrika 1:183–187
18. Golub G, Van Loan C (1996) Matrix computations, 3rd edn. Johns Hopkins, Baltimore
19. Govaert G (1995) Simultaneous clustering of rows and columns. Control Cybern 24:437–458
20. Harshman RA (1978) Models for analysis of asymmetrical relationships among N objects or stimuli. In: First joint meeting of the psychometric society for mathematical psychology
21. Harshman RA (1970) Foundations of the parafac procedure: models and conditions for an 'explanatory' multi-modal factor analysis. UCLA working papers in phonetics 16, pp 1–84
22. Harshman RA, Kolda TG, Bader BW (2007) Temporal analysis of semantic graphs using asalsan. In: Proceedings of IEEE international conference on data mining (ICDM 2007)
23. Hastie T, Tibshirani R, Friedman JH (2001) The elements of statistical learning. Springer, Berlin
24. Hyvarinen A, Karhunen J, Oja E (2001) Independent component analysis. Wiley, London
25. Kim Y, Choi S (2007) Nonnegative tucker decomposition. In: Proceedings of IEEE conference on computer vision and pattern recognition
26. Kroonenberg PM, De Leeuw J (1980) Principal component analysis of three-mode data by means of alternating least squares algorithms. Psychometrika 45:69–97
27. Kolda T (2001) Orthogonal tensor decomposition. SIAM J Matrix Anal Appl 23:243–255
28. Kolda T, Bader B (2006) The TOPHITS model for higher-order web link analysis. In: Workshop on link analysis, counter terrorism and security
29. Lee D, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401:788–791
30. Lee D, Seung HS (2001) Algorithms for non-negative matrix factorization. In: NIPS
31. Li T (2008) Clustering based on matrix approximation: a unifying view. Knowl Inf Syst (KAIS) 17(1): 1–15
32. Li T, Ding C (2006) The relationships among various nonnegative matrix factorization methods for clustering. In: ICDM, pp 362–371
33. Li T, Ma S, Ogihara M (2004) Document clustering via adaptive subspace iteration. In: SIGIR, pp 218–225
34. Mardia KV, Kent JT, Bibby JM (1979) Multivariate analysis. Academic Press, London
35. Paatero P (1999) The multilinear engine: a table-driven, least squares program for solving multilinear problems, including the n-way parallel factor analysis model. J Comput Graph Stat 8(4):854–888
36. Paatero P, Tapper U (1994) Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. Environmetrics 5:111–126
37. Peng W, Li T (2011) Temporal relation co-clustering on directional social network and author-topic evolution. Knowl Inf Syst (KAIS) 26(3):467–486
38. Peng W, Li T (2011) On the equivalence between nonnegative tensor factorization and tensorial probabilistic latent semantic analysis. Appl Intell 35(2):285–295
39. Rocci R, Vichi M (2005) Three-mode component analysis with crisp or fuzzy partition of units. Psychometrika 70(4):715–736
40. Shashua A, Hazan T (2005) Non-negative tensor factorization with applications to statistics and computer vision. ICML'05
41. Smilde A, Bro R, Geladi P (2004) Multi-way analysis: applications in the chemical sciences. Wiley, London
42. Strehl A, Ghosh J (2002) Cluster ensembles—a knowledge reuse framework for combining multiple partitions. J Mach Learn Res (JMLR) 3:583–617
43. Sun J, Zeng H, Liu H, Lu Y, Chen Z (2005) Cubesvd: a novel approach to personalized web search. In: Proceedings of the 14th international conference on World Wide Web
44. Tipping M, Bishop C (1999) Probabilistic principal component analysis. J R Stat Soc Ser B 21(3): 611–622
45. Thurau C, Kersting K, Wahabzada M, Bauckhage C (2011) Convex non-negative matrix factorization for massive datasets. Knowl Inf Syst (KAIS)
46. Tucker LR (1966) Some mathematical notes on three-mode factor analysis. Psychometrika 31(3): 279–311
47. Yu SX, Shi J (2003) Multiclass spectral clustering. In: Proceedigns of the 9th IEEE international conference on computer vision (ICCV 2003), pp 313–319
48. Vasilescu MAO, Terzopoulos D (2002) Multilinear analysis of image ensembles: Tensorfaces. In: Proceedings of the 7th European conference on computer vision-part I (ECCV'02), pp 447–460
49. Vichi M, Kiers HAL (2001) Factorial k-means analysis for two-way data. Comput Stat Data Anal 37: 49–64
50. Vichi M, Rocci R (2008) Two-mode multi-partitioning. Comput Stat Data Anal 52:1984–2003

51. Vichi M, Rocci R, Kiers HAL (2007) Simultaneous component and clustering models for three-way data: within and between approaches. J Classif 24(1):71–98
52. Welling M, Weber M (2001) Positive tensor factorization. Pattern Recogn Lett 22(12):1255–1261
53. Wu X, Kumar V, Quinlan JR, Ghosh J, Yang Q, Motoda H, McLachlan GJ, Ng A, Liu B, Yu PS, Zhou Z, Steinbach M, Hand DJ, Steinberg D (2007) Top 10 algorithms in data mining. Knowl Inf Syst (KAIS) 14(1):1–37
54. Zhang T, Golub GH (2001) Rank-one approximation to high order tensor. SIAM J Matrix Anal Appl 23:534–550

Author Biographies

Zhong-Yuan Zhang is an Associate Professor with School of Statistics, Central University of Finance and Economics, P.R. China. He received his PhD degree in Academy of Mathematics and Systems Science, Chinese Academy of Sciences in 2008. His current research interests include data mining, complex social network analyzing, and image processing.

Tao Li is currently an Associate Professor in the School of Computer Science at Florida International University. He received his PhD in Computer Science in 2004 from the University of Rochester. His research interests are data mining, machine learning, and information retrieval. He is a recipient of NSF CAREER Award and multiple IBM Faculty Research Awards.

Chris Ding received his PhD from Columbia University. He did research at California Institute of Technology, Jet Propulsion Laboratory, and Lawrence Berkeley National Laboratory, before joining University of Texas at Arlington in 2007 as a professor of Computer Science. His main research areas are machine learning and bioinformatics.