Heterogeneous Tensor Decomposition for Clustering via Manifold Optimization

Yanfeng Sun, Junbin Gao, Xia Hong, Bamdev Mishra, and Baocai Yin

Abstract—Tensor clustering is an important tool that exploits intrinsically rich structures in real-world multiarray or Tensor datasets. Often in dealing with those datasets, standard practice is to use subspace clustering that is based on vectorizing multiarray data. However, vectorization of tensorial data does not exploit complete structure information. In this paper, we propose a subspace clustering algorithm without adopting any vectorization process. Our approach is based on a novel heterogeneous Tucker decomposition model taking into account cluster membership information. We propose a new clustering algorithm that alternates between different modes of the proposed heterogeneous tensor model. All but the last mode have closed-form updates. Updating the last mode reduces to optimizing over the multinomial manifold for which we investigate second order Riemannian geometry and propose a trust-region algorithm. Numerical experiments show that our proposed algorithm compete effectively with state-of-the-art clustering algorithms that are based on tensor factorization.

Index Terms—Tensor clustering, multinomial manifold, Fisher metric, Riemannian optimization, trust-region

1 Introduction

In the last two decades, the advance of modern sensing, networking, communication and storage technologies have paved the way for the availability of multidimensional data with high dimensionality. For example, remote sensing is producing massive multidimensional data that need to be carefully analyzed. One of the characteristics of these gigantic datasets is that they often have a large amount of redundancies. This motivates the development of a low-dimensional representation that best assists a range of learning tasks in order to avoid the so-called "curse of dimensionality" [1]. Many data processing tasks involve manipulating multidimensional objects. For example, video data [2] can be regarded as an object of pixel location in two dimensions plus one dimension in time. Similar representation can be seen in analyzing remote sensing data [3]. In analyzing personalized webpages, the data is usually represented as a third order dataset with three dimensional modes of users, query words and webpages, respectively [4]. In document clustering, one presents the dataset in a three-way format of authors, terms and times [5], [6].

The multi-dimensional data are known as tensors [7], [8], where data elements are addressed by more than two indices. An N-order tensor is an element of the tensor product of N vector spaces. A 2D matrix is an example of the second-order tensor. Similarly, hyperspectral imagery [9] is naturally a three-dimensional (3D) data cube containing both spatial and spectral dimensions. Simultaneously considering both spectral and spatial structures of hyperspectral data in clustering or classification leads to superior results, e.g., in [10]. However, in order to circumvent the issue of high dimensionality, often data compression with "parsimonious" representation is used. Other approaches that directly process structural information of tensorial data have also been proposed, e.g., tensorial data structure is exploited in computer vision applications [11], [12] and machine learning [13], [14].

Tensorial data have a large amount of redundancies. It is desired to have a mechanism to reduce such redundancies for the sake of efficient application of learning algorithms. We use the general term dimensionality reduction to describe such techniques or models. There exist various tensor decomposition models, amongst which the CANDECOMP (canonical decomposition)/PARAFAC (parallel factors) or in short *CP decomposition* [15] and the *Tucker decomposition* are two fundamental models for tensor decomposition (refer to the survey paper [8] for details). It should be noted that the CP decomposition is a special case of the Tucker decomposition [8], where the factor matrices have same number of columns and the core tensor is superdiagonal, which means that every mode of the tensor is of the same size and its elements remain constant under any permutation of the indices. Many other decomposition algorithms/models can be viewed as special formats of the CP and Tucker decomposition. For example, the higher order

- *Y. Sun is with the Beijing Municipal Key Lab of Multimedia and Intelligent Software Technology, College of Metropolitan Transportation, Beijing University of Technology, Beijing 100124, China. E-mail: yfsun@bjut.edu.cn.*
- *B. Yin is with the Beijing Municipal Key Lab of Multimedia and Intelligent Software Technology, College of Metropolitan Transportation, Beijing University of Technology, Beijing 100124, China, and the School of Software Technology, Dalian University of Technology, Dalian 116024, China. E-mail: ybc@bjut.edu.cn.*
- *J. Gao is with the School of Computing and Mathematics, Charles Sturt University, Bathurst, NSW 2795, Australia. E-mail: jbgao@csu.edu.au.*
- *X. Hong is with the School of Systems Engineering, University of Reading, Reading RG6 6AY, United Kingdom. E-mail: x.hong@reading.ac.uk.*
- *B. Mishra is with the Department of Electrical Engineering and Computer Science, University of Liège, Liège, Belgium, and the Department of Engineering, University of Cambridge, Cambridge, United Kingdom. E-mail: b.mishra@ulg.ac.be.*

Manuscript received 7 May 2014; revised 15 July 2015; accepted 31 July 2015. Date of publication 6 Aug. 2015; date of current version 10 Feb. 2016. Recommended for acceptance by J. Ye.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2015.2465901

0162-8828 © 2015 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

SVD algorithm (HOSVD) [16], an extension of the classical SVD, is a special case of the general Tucker decomposition of a tensor in which the core tensor is of the same dimension as the tensor to be decomposed and all the mode matrices have *orthonormal columns*. Similarly, the classical PCA has several extensions for tensorial data. The generalized tensor PCA (GND-PCA) seeks a shared Tucker decomposition for all the given tensors in which the core tensors are different but the matrix factors along each mode are orthogonal. This decomposition procedure is also called the *higher-order orthogonal iteration* (HOOI) algorithm in [17].

In applications where data are non-negative such as images, the non-negative matrix factorization (NFM) has proven to be a successful approach for detecting essential features in the data [18]. Several efficient algorithms have been proposed [19], [20]. Recently, NFM has been extended to non-negative tensor factorization (NTF) and has been investigated in [13], [21], [22], [23], [24].

Another trend in tensor decomposition research is to introduce more structures in the decomposition model itself. Recently, Zhang et al. [10] considered Tri-factor orthogonal non-negative tensor decomposition (Tri-ONTD), a new tensor decomposition model. The fundamental aim of this model is to discover common characteristics of a series of matrix data. A straightforward application of Tri-ONTD is to identify cluster structures of a dataset. The core idea behind this model is based on the centroid-based clustering algorithms such as the *k-means* algorithm. The idea of introducing new structures in tensor decomposition can also be seen in [25] in the context of image representation.

Tensor decomposition is not the ultimate goal of tensor analysis, rather it is a process to condense information and reduce redundancies. A tensor decomposition model is usually employed in the process of conducting tensorial analysis such as tensor clustering and classification. Tensor clustering tasks, including those mentioned above, are often formulated as optimization problems on specific tensor decomposition models [26], [27], [28], [29]. For example, when imposing some specific constraints, like orthogonality in the HOOI algorithm [17], the resulting problems reduce to optimization over *matrix manifold constraints* [30]. In the case of the HOOI algorithm for the HOSVD decomposition, the optimization problem is on the *Stiefel manifold* [30, Section 3.3]).

While the optimization problem in HOSVD admits a closed-form solution under the least-squares error criterion, computing a closed-form solution is not possible for the tensor clustering problem considered in this paper. Most existing algorithms avoid this issue by reformulating it into an optimization problem over the "flat" euclidean space instead of a manifold with some additional treatment, e.g., by introducing a regularization term. On the other hand, recent years have witnessed significant development of Riemannian optimization algorithms on matrix manifolds such as the Stiefel manifold, the Grassmann manifold, and the manifold of positive definite matrices [30], [31], [32]. The Riemannian optimization framework endows a matrix manifold constraint with a *Riemannian manifold structure. Conceptually, it translates a constrained optimization problem into an*unconstrained optimization*problem on a Riemannian manifold*. Since the Riemannian optimization framework is directly based on nonlinear manifolds, one "eliminates" constraints such as orthogonality to obtain an unconstrained optimization problem that, by construction, only use feasible points. The recent successful applications of the Riemannian optimization framework in machine learning, computer vision and data mining, include low rank optimization [31], [32], [33], [34], estimation [34], Riemannian dictionary learning [35], [36], and computer vision tasks [37], to name a few.

In this paper, we propose a novel subspace clustering algorithm that exploits the tensorial structure of data. To this end, we introduce a novel *heterogeneous* Tucker decomposition model. The proposed clustering algorithm alternates between different modes of the proposed heterogeneous tensor model. All but the last mode have closed-form updates. Updating the last mode reduces to optimizing over the *multinomial manifold*, defined in Section 3, for which we investigate second order Riemannian geometry and propose a trust-region (TR) algorithm. The multinomial manifold is given a Riemannian manifold structure by endowing it with the *Fisher information metric* [38], [39]. The Fisher information metric gives the multinomial manifold a differentiable structure, i.e., it ensures that the boundary is "scaled" to infinity. Our numerical experiment results from the proposed algorithm demonstrate the benefits.

The contribution of this paper is twofold. First, we propose a heterogeneous Tucker decomposition model for tensor clustering. Second, we investigate the Riemannian geometry of the multinomial manifold and apply the Riemannian trust-region (RTR) algorithm to the resulting nonlinear clustering problem over the manifold.

The paper is organized as follows. We introduce the notations for tensor representation in Section 2.1. Section 2.2 introduces the novel clustering scheme based on the proposed heterogeneous Tucker decomposition model. The associated optimization problems are discussed in Section 2.3. Section 3 explores the Riemannian geometry for the multinomial manifold and develops all the necessary optimization-related ingredients. Section 4 presents the algorithm for the tensor clustering including the proposed Riemannian trust-region algorithm. Section 5 shows numerical experimental results on both synthetic tensorial data and real-world datasets. Finally, Section 6 concludes the paper.

2 Heterogeneous Tucker Decomposition Model for Clustering

In this section, starting with notation for tensors, we motivate the work, and finally propose the new heterogeneous Tucker decomposition model for clustering.

2.1 Tensor Notation and Operations

We follow the convention used in [8] to denote 1D vector by lowercase boldface symbols like $\mathbf{v}$, 2D matrix by uppercase boldface symbols like $\mathbf{U}$ and general tensors by calligraphy symbols like $\mathcal{X}$.

Let $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times \cdots \times I_N}$ be an N-order tensor with $x_{i_1 \cdots i_n \cdots i_N}$ as the $(i_1 \cdots i_n \cdots i_N)$th element. The n-mode product of an N-order tensor $\mathcal{X}$ with a matrix $\mathbf{U}_n \in \mathbb{R}^{I_n \times J_n}$ is denoted by $\mathcal{X} \times_n \mathbf{U}_n$. The result is an N-order tensor of

dimension $I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N$. Element wise, the n-mode product is expressed as

$$(\mathcal{X} \times_n \mathbf{U}_n)_{i_1 \cdots i_{n-1} j_n i_{n+1} \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \cdots i_{n-1} i_n i_{n+1} \cdots i_N} u_{j_n i_n}.$$

Given an N-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times \cdots \times I_N}$, its Tucker decomposition is

$$\mathcal{X} \approx \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \cdots \times_N \mathbf{U}_N \triangleq [\![\mathcal{G}; \mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_N]\!], \quad (1)$$

where $\mathcal{G}$ is the *core* N-order tensor of size $J_1 \times \cdots \times J_n \times \cdots \times J_N$ with $J_n \leq I_n$ and $\mathbf{U}_n \in \mathbb{R}^{I_n \times J_n}$ is a matrix applied along mode-n. The core tensor $\mathcal{G}$ is interpreted as a low dimensional representation of the tensor $\mathcal{X}$.

2.2 Heterogeneous Tucker Decomposition Model

Most Tucker decomposition models are of a homogeneous nature, by which we mean that all the factor matrices $\mathbf{U}_n$ satisfy the same constraint. For example, in the classical HOSVD [8], all the factor matrices are required to be orthogonal, i.e., $\mathbf{U}_n^T \mathbf{U}_n = \mathbf{I}_{J_n}$ (denoted by $\mathbf{I}$ for simplicity). In the nonnegative Tucker decomposition model [22], [23], all the factors are matrices with nonnegative entries, i.e., $\mathbf{U}_n \geq 0$.

However, the requirement for homogeneous factors $\mathbf{U}_n$ is not preferred in many cases. Especially, when the factor matrices in different modes have different interpretations. For example, consider the problem of clustering a set of images (two-order tensors). We can stack all the images onto a three-order tensor, where the third mode corresponds to the number of images. For clustering the images, it is of interest to decompose the entire three-order tensor in a way that, along the third mode, each image is represented by several cluster representatives, as done in fuzzy k-means algorithms [40]. This can be achieved by ensuring that the last mode factor matrix, i.e., $\mathbf{U}_3$ is nonnegative and the row sum of $\mathbf{U}_3$ is 1 to mimic the cluster probability of an image.

In general, we suppose that we are given a set of M $(N-1)$-order tensors, denoted by $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_M\}$ and we want cluster them into K clusters. This can be done by projecting the tensors along all the first $N-1$ modes, then cluster them. To this end, we stack all the M $(N-1)$-order tensors along the N mode, so that we have an N-order tensor $\mathcal{X}$ in a way that each slice of $\mathcal{X}$, along the last mode, is one of $(N-1)$-order tensors $\mathcal{X}_l$ $(l = 1, 2, \ldots, M)$. Following the general notation in Section 2.1, we have $I_N = M$.

Our proposed model is defined by the optimization problem

$$\min_{\mathcal{G}, \mathbf{U}_1^T \mathbf{U}_1 = \mathbf{I}, \ldots, \mathbf{U}_{N-1}^T \mathbf{U}_{N-1} = \mathbf{I}, \mathbf{U}_N \mathbf{1} = \mathbf{1}, \mathbf{U}_N \geq 0} f(\mathcal{G}, \mathbf{U}) \quad (2)$$

with

$$f(\mathcal{G}, \mathbf{U}) = \frac{1}{2} \|\mathcal{X} - \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \cdots \times_N \mathbf{U}_N\|_F^2,$$

where $\mathbf{1}$ is a column vector of all ones, $\mathbf{U}$ collects all the $\mathbf{U}_1, \ldots, \mathbf{U}_{N-1}$ matrices whose columns are orthogonal, $\|\cdot\|_F$ denotes the *Frobenius* norm, and $\mathbf{U}_N \in \mathbb{R}^{M \times K}$ is nonnegative and the sum of each row is 1. In (2), the dimension of $\mathcal{G}$ in mode N is K.

If each of the K slices of $\mathcal{G}$ is interpreted as *cluster centroids* in the projected space $\mathbb{R}^{J_1 \times J_2 \times \cdots \times J_{N-1}}$, then the rows of $\mathbf{U}_N$ has the interpretation of cluster indicators.

Given this heterogeneous Tucker decomposition that is specifically aimed for clustering, the search space is the product space of $(N-1)$ *Stiefel manifolds*, corresponding to the constraints $\mathbf{U}_1^T \mathbf{U}_1 = \mathbf{I}, \ldots, \mathbf{U}_{N-1}^T \mathbf{U}_{N-1} = \mathbf{I}$, and the *multinomial manifold* that corresponds to the constraints $\mathbf{U}_N \mathbf{1} = \mathbf{1}, \mathbf{U}_N \geq 0$. It should be noted that the problem (2) needs two different treatments in terms of optimizing the factor matrices, i.e., one for the first $(N-1)$ orthogonal matrices and the other for the factor $\mathbf{U}_N$.

2.3 Optimization

To reduce the number of optimization variables in problem (2), we optimize the core tensor $\mathcal{G}$ by assuming that all the factor matrices are fixed to their current values. This becomes a least-squares problem. It is straightforward to show that the solution is given by [8]

$$\mathcal{G} = \mathcal{X} \times_1 \mathbf{U}_1^T \times_2 \cdots \times_{N-1} \mathbf{U}_{N-1}^T \times_N \left[\left(\mathbf{U}_N^T \mathbf{U}_N\right)^{-1} \mathbf{U}_N^T\right], \quad (3)$$

where we have used the orthogonality of $\mathbf{U}_n$ $(n = 1, 2, \ldots, N-1)$. It should be noted that $\mathbf{U}_N$ is column full rank.

Using (3) in the objective function of (2) results in the following relation

$$\begin{aligned} f(\mathcal{G}, \mathbf{U}) &= \frac{1}{2} \|\mathcal{X} - \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \cdots \times_N \mathbf{U}_N\|_F^2 \\ &= \frac{1}{2} \|\mathcal{X}\|_F^2 - \frac{1}{2} \langle \mathcal{G} \times_N (\mathbf{U}_N^T \mathbf{U}_N), \mathcal{G} \rangle \\ &= \frac{1}{2} \|\mathcal{X}\|_F^2 - \frac{1}{2} \langle \mathcal{G} \times_N \mathbf{U}_N, \mathcal{G} \times_N \mathbf{U}_N \rangle \\ &= \frac{1}{2} \|\mathcal{X}\|_F^2 - \frac{1}{2} \|\mathcal{X} \times_1 \mathbf{U}_1^T \times_2 \cdots \times_{N-1} \mathbf{U}_{N-1}^T \times_N \\ &\quad \times_N [\mathbf{U}_N (\mathbf{U}_N^T \mathbf{U}_N)^{-1} \mathbf{U}_N^T]\|_F^2. \end{aligned}$$

Subsequently, minimizing (2) is equivalent to

$$\max_{\mathbf{U}_1^T \mathbf{U}_1 = \mathbf{I}, \ldots, \mathbf{U}_{N-1}^T \mathbf{U}_{N-1} = \mathbf{I}, \mathbf{U}_N \mathbf{1} = \mathbf{1}, \mathbf{U}_N \geq 0} h(\mathbf{U}), \quad (4)$$

where $h(\mathbf{U}) = 0.5 \|\mathcal{X} \times_1 \mathbf{U}_1^T \times_2 \cdots \times_{N-1} \mathbf{U}_{N-1}^T \times_N [\mathbf{U}_N (\mathbf{U}_N^T \mathbf{U}_N)^{-1} \mathbf{U}_N^T]\|_F^2$. It should be emphasized that the function h is *smooth* in the variables and the constraints are *separable*. This motivates to consider an *alternating optimization scheme* for (4), where we maximize (4) with respect to one variable while fixing others. This procedure is then cyclically repeated.

Let $\mathbf{V}_N = \mathbf{U}_N (\mathbf{U}_N^T \mathbf{U}_N)^{-1} \mathbf{U}_N^T$. If all but $\mathbf{U}_n$ $(n = 1, 2, \ldots, N-1)$ are fixed, then $\mathbf{U}_n$ is optimized by solving an *eigenvector problem*. Specifically, consider updating $\mathbf{U}_n$ $(n = 1, 2, \ldots, N-1)$ while all the others being fixed. Let $\mathbf{U}_{(-n)} = \mathbf{V}_N \otimes \cdots \otimes \mathbf{U}_{(n+1)} \otimes \mathbf{U}_{(n-1)} \otimes \cdots \otimes \mathbf{U}_1$. Subsequently, (4) can be rewritten as, after mode-n matricization,

$$\max_{\mathbf{U}_n^T \mathbf{U}_n = I_n} \quad \frac{1}{2} \|\mathbf{U}_n^T (\mathbf{X}_{(n)} \mathbf{U}_{(-n)})\|_F^2, \quad (5)$$

where $\mathbf{X}_{(n)}$ is the n-mode matricization of tensor $\mathcal{X}$. The problem (5) has a closed-form solution, i.e., if $\mathbf{B}_n = \mathbf{X}_{(n)} \mathbf{U}_{(-n)}$, then

$$\mathbf{U}_n = \mathrm{uf}\big(\mathbf{B}_n^T\big), \tag{6}$$

where $\mathrm{uf}(\cdot)$ extracts the orthogonal factor of the polar decomposition of its argument and is computed as $\mathrm{uf}(\mathbf{A}) = \mathbf{P}\mathbf{Q}^T$, where $\mathbf{P}\Sigma\mathbf{Q}^T = \mathbf{A}$ is the thin singular value decomposition of $\mathbf{A}$.

As discussed in Section 2.2, the nonnegativity constraint on $\mathbf{U}_N$ is specifically introduced for clustering, leading to the optimization problem

$$\begin{aligned}\max_{\mathbf{U}_N\mathbf{1}=\mathbf{1},\mathbf{U}_N\geq 0} \quad & \frac{1}{2}\|\mathbf{V}_N\mathbf{B}_N\|_F^2 = \frac{1}{2}\mathrm{tr}\big(\mathbf{B}_N^T\mathbf{V}_N^T\mathbf{V}_N\mathbf{B}_N\big) \\ & = \frac{1}{2}\mathrm{tr}\big(\mathbf{B}_N^T\mathbf{U}_N\big(\mathbf{U}_N^T\mathbf{U}_N\big)^{-1}\mathbf{U}_N^T\mathbf{B}_N\big),\end{aligned} \tag{7}$$

where $\mathrm{tr}(\cdot)$ is the trace of a square matrix and $\mathbf{B}_N = \mathbf{X}_{(N)}[\mathbf{U}_{N-1}^T \otimes \cdots \otimes \mathbf{U}_1^T]^T = \mathbf{X}_{(N)}[\mathbf{U}_{N-1} \otimes \cdots \otimes \mathbf{U}_1]$ with $\mathbf{X}_{(N)}$ as the N-mode matricization of $\mathcal{X}$ and $\otimes$ is the Kronecker product of matrices.

Problem (7) is a nonlinear optimization problem over the constraint $\{\mathbf{U}_N|\mathbf{U}_N\mathbf{1} = \mathbf{1}, \mathbf{U}_N \geq 0\}$, called the multinomial manifold, where $\mathbf{1}$ is the vector of all ones. An efficient numerical algorithm is needed. To this end, we propose an a trust-region algorithm in Section 4.1 based on the Riemannian structure of the multinomial manifold that is discussed in Section 3.

It should be noted that there exist several efficient algorithms for linearly constrained smooth convex optimization, e.g., [41]. However, it is not clear whether such approaches are efficiently implementable for a structured nonconvex optimization problem such as the one in (7).

3 The Multinomial Manifold

For the concepts of general abstract manifolds, we refer the readers to the textbook [42]. Each row of $\mathbf{U}_N$ is a discrete probability distribution which describes the membership over the tensor centroids. All the discrete probability distributions make up the multinomial manifold (also called a simplex) defined by

$$\mathbb{P}^K = \left\{\mathbf{u} = (u_1, \ldots, u_K)^T \in \mathbb{R}^K : \; u_k > 0, \sum_{k=1}^K u_k = 1\right\}.$$

It should be noted that the nonnegativity constraint $u_k \geq 0$ is replaced with strict positivity to ensure that the set $\mathbb{P}^K$ is differentiable. A possible use of the multinomial manifold is in proposing a classifier with kernels, e.g., in [38], [39], [43]. For the purpose of solving the optimization problem (7), we investigate the product manifold of multiple multinomial manifolds, still called the multinomial manifold, defined as

$$\mathbb{P}_M^K = \left\{\mathbf{U} = [U_{mk}] \in \mathbb{R}^{M\times K} : \; U_{mk} > 0, \sum_{k=1}^K U_{mk} = 1\right\}.$$

3.1 The Submanifold Structure

An element of $\mathbb{P}_M^K$ is represented by an $M \times K$ matrix $\mathbf{U}$. It is straightforward to show that the tangent space at $\mathbf{U} \in \mathbb{P}_M^K$ is given by

$$T_{\mathbf{U}}\mathbb{P}_M^K = \{\eta_{\mathbf{U}} \in \mathbb{R}^{M\times K} : \eta_{\mathbf{U}}\mathbf{1} = \mathbf{0}\},$$

where $\mathbf{1} \in \mathbb{R}^K$ is a column vector of all ones and $\mathbf{0} \in \mathbb{R}^M$ is a column vector of all zeros.

It should also be noted that we can characterize the manifold $\mathbb{P}_M^K$ as an *embedded Riemannian submanifold* of the euclidean space $\mathbb{R}^{M\times K}$ equipped with the metric g (inner product), i.e.,

$$g_{\mathbf{U}}(\xi_{\mathbf{U}}, \eta_{\mathbf{U}}) = \sum_{m,k} \frac{(\xi_{\mathbf{U}})_{mk}(\eta_{\mathbf{U}})_{mk}}{U_{mk}}, \tag{8}$$

where $\xi_{\mathbf{U}}$ and $\eta_{\mathbf{U}}$ belong to the tangent space $T_{\mathbf{U}}\mathbb{P}_M^K$ at the point $\mathbf{U}$ on the manifold $\mathbb{P}_M^K$. The metric $g_{\mathbf{U}}$ defining the new Riemannian structure of the manifold is called the *Fisher information metric* [39]. The inner product defined in (8) determines the geometry such as distance, angle, curvature on $\mathbb{P}_M^K$. Despite the use of the Fisher information metric on the multinomial manifold in [38], [39], to the best of our knowledge, the derivations of the Riemannian gradient and Hessian formulas for a smooth objective function are new to the present paper.

The notion of *Riemannian immersion* helps in computing the Riemannian gradient and Hessian formulas on the manifold $\mathbb{P}_M^K$ in terms of their formulas in the embedding space $\mathbb{R}^{M\times K}$. The basis of this is the following orthogonal, in the sense of the proposed metric, projection operator.

To project a matrix $\mathbf{Z} \in \mathbb{R}^{M\times K}$ onto the tangent space $T_{\mathbf{U}}\mathbb{P}_M^K$, we define the linear operation $\Pi_{\mathbf{U}} : \mathbb{R}^{M\times K} \to T_{\mathbf{U}}\mathbb{P}_M^K : \mathbf{Z} \mapsto \Pi_{\mathbf{U}}(\mathbf{Z})$ as

$$\Pi_{\mathbf{U}}(\mathbf{Z}) = \mathbf{Z} - (\alpha\mathbf{1}^T) \odot \mathbf{U},$$

where $\alpha = \mathbf{Z}\mathbf{1} \in \mathbb{R}^M$ and $\odot$ is the element-wise matrix multiplication operation. This projection operation is computed by characterizing the tangent space and its complementary space in the sense of the metric (8).

Another important concept in the framework of Riemannian optimization is the notion of a *retraction* operation [30, Section 4.1]. The retraction mapping is used to locate the next iterate on the manifold along a specified tangent vector [30, Chapter 4]. Ideally, the *exponential map* is the canonical choice for the retraction mapping, which generalizes the notion of "following a straight line" in the euclidean space. However, in this paper we work with the following standard approximation that is easier to implement. Given a tangent vector $\xi_{\mathbf{U}} \in T_{\mathbf{U}}\mathbb{P}_M^K$, the proposed retracting mapping $R_{\mathbf{U}} : T_{\mathbf{U}}\mathbb{P}_M^K \to \mathbb{P}_M^K$ is

$$\begin{aligned}\mathbf{U}_+ &= R_{\mathbf{U}}(\xi_{\mathbf{U}}) \\ &:= (\mathbf{U} \odot \exp(\xi_{\mathbf{U}} \oslash \mathbf{U})) \oslash ((\mathbf{U} \odot \exp(\xi_{\mathbf{U}} \oslash \mathbf{U}))\mathbf{1}\mathbf{1}^T),\end{aligned}$$

where $\oslash$ is the element-wise matrix inversion and $\exp(\cdot)$ is the element-wise exponential operator on matrices.

3.2 The Riemannian Gradient Computation

Let $\mathrm{Grad}F(\mathbf{U})$ be the euclidean gradient of a smooth function $F : \mathbb{P}_M^K \longmapsto \mathbb{R}$ with the euclidean metric. The gradient in $\mathbb{R}^{M\times K}$ endowed with the metric g is scaled as $\mathrm{Grad}F(\mathbf{U}) \odot \mathbf{U}$. This can be attributed to the "change of basis" in the euclidean space $\mathbb{R}^{M\times K}$.

The expression of the Riemannian gradient $\mathrm{grad}F(\mathbf{U})$ on $\mathbb{P}_M^K$ is obtained by projecting the scaled-gradient $\mathrm{Grad}F(\mathbf{U})\odot\mathbf{U}$ to the tangent space $T_{\mathbf{U}}\mathbb{P}_M^K$, i.e.,

$$\mathrm{grad}F(\mathbf{U}) = \Pi_{\mathbf{U}}(\mathrm{Grad}F(\mathbf{U})\odot\mathbf{U}). \tag{9}$$

3.3 The Riemannian Hessian Computation

In order to compute the Riemannian Hessian of a smooth objective function, we need the notion of the *Riemannian connection* [30, Section 5.5]. The Riemannian connection, denoted as $\nabla_{\xi_{\mathbf{U}}}\eta_{\mathbf{U}}$, generalizes the *covariant-derivative* of the tangent vector $\eta_{\mathbf{U}}$ along the direction of the tangent vector $\xi_{\mathbf{U}}$ on the manifold $\mathbb{P}_M^K$. Since $\mathbb{P}_M^K$ is a Riemannian submanifold of $\mathbb{R}^{M\times K}$, the Riemannian connection on $\mathbb{P}_M^K$ is also characterized by the projection of the corresponding connection $\overline{\nabla}_{\xi_{\mathbf{U}}}\eta_{\mathbf{U}}$ in the embedding space $\mathbb{R}^{M\times K}$ endowed with the metric (8), i.e., $\nabla_{\xi_{\mathbf{U}}}\eta_{\mathbf{U}} = \Pi_{\mathbf{U}}(\overline{\nabla}_{\xi_{\mathbf{U}}}\eta_{\mathbf{U}})$ [30, Proposition 5.3.2].

The connection $\overline{\nabla}$ in the euclidean space $\mathbb{R}^{M\times K}$ endowed with the metric (8) is computed using the *Koszul formula* [30, Theorem 5.3.1], and after a few steps of computations, it admits the matrix characterization

$$\overline{\nabla}_{\xi_{\mathbf{U}}}\eta_{\mathbf{U}} = D\eta_{\mathbf{U}}[\xi_{\mathbf{U}}] - \frac{1}{2}(\xi_{\mathbf{U}}\odot\eta_{\mathbf{U}})\oslash\mathbf{U}.$$

Consequently, the Riemannian Hessian $\mathrm{Hess}F(\mathbf{U})[\xi_{\mathbf{U}}]$ of a smooth function $F:\mathbb{P}_M^K\longmapsto\mathbb{R}$ is the covariant-derivative of the Riemannian gradient $\mathrm{grad}F(\mathbf{U})$ in the direction $\xi_{\mathbf{U}}\in T_{\mathbf{U}}\mathbb{P}_M^K$, i.e.,

$$\begin{aligned}\mathrm{Hess}F(\mathbf{U})[\xi_{\mathbf{U}}] &= \Pi_{\mathbf{U}}(\overline{\nabla}_{\xi_{\mathbf{U}}}\mathrm{grad}F(\mathbf{U}))\\ &= \Pi_{\mathbf{U}}(D\mathrm{grad}F(\mathbf{U})[\xi_{\mathbf{U}}] - \frac{1}{2}(\xi_{\mathbf{U}}\odot\mathrm{grad}F(\mathbf{U}))\oslash\mathbf{U}),\end{aligned} \tag{10}$$

where $\mathrm{grad}F(\mathbf{U})$ is the Riemannian gradient and $D\mathrm{grad}F(\mathbf{U})[\xi_{\mathbf{U}}]$ is the euclidean directional derivative of the Riemannian gradient in the direction $\xi_{\mathbf{U}}\in T_{\mathbf{U}}\mathbb{P}_M^K$, i.e.,

$$\begin{aligned}D\mathrm{grad}F(\mathbf{U})[\xi_{\mathbf{U}}] &= D\Pi_{\mathbf{U}}(\mathrm{Grad}F(\mathbf{U})\odot\mathbf{U})[\xi_{\mathbf{U}}]\\ &= D\mathrm{Grad}F(\mathbf{U})[\xi_{\mathbf{U}}]\odot\mathbf{U} + \mathrm{Grad}F(\mathbf{U})\odot\xi_{\mathbf{U}}\\ &\quad - (\alpha\mathbf{1}^T)\odot\xi_{\mathbf{U}} - (D\alpha[\xi_{\mathbf{U}}]\mathbf{1}^T)\odot\mathbf{U},\end{aligned} \tag{11}$$

where α is equal to $(\mathrm{Grad}F(\mathbf{U})\odot\mathbf{U})\mathbf{1}$. Therefore, $D\alpha[\xi_{\mathbf{U}}] = (D\mathrm{Grad}F(\mathbf{U})[\xi_{\mathbf{U}}]\mathbf{U} + \mathrm{Grad}F(\mathbf{U})\odot\xi_{\mathbf{U}})\mathbf{1}$, and $D\mathrm{Grad}F(\mathbf{U})[\xi_{\mathbf{U}}]$ is the euclidean directional derivative of the euclidean gradient $\mathrm{Grad}F(\mathbf{U})$ along the direction $\xi_{\mathbf{U}}\in T_{\mathbf{U}}\mathbb{P}_M^K$, i.e., $D\mathrm{Grad}F(\mathbf{U})[\xi_{\mathbf{U}}] = \lim_{t\to 0}(\mathrm{Grad}F(\mathbf{U}+t\xi_{\mathbf{U}}) - \mathrm{Grad}F(\mathbf{U}))/t$.

4 The Proposed Algorithm

The optimization problem (7) in Section 2.3 is nonlinear, with linear constraints. To this end, we propose to use the Riemannian trust-region optimization algorithm [30, Chapter 7].

4.1 The Riemannian Trust-Region Algorithm

In order to solve (7), we define $F(\mathbf{U}_N) = -\frac{1}{2}\mathrm{tr}(\mathbf{B}_N^T\mathbf{U}_N(\mathbf{U}_N^T\mathbf{U}_N)^{-1}\mathbf{U}_N^T\mathbf{B}_N)$. The problem (7) boils down to the optimization problem of the form

$$\min_{\mathbf{U}\in\mathbb{P}_M^K} F(\mathbf{U}). \tag{12}$$

In order to simplify the exposition in the subsequent sections, we use $\mathbf{U}$ and $\mathbf{B}$ in (12) instead of $\mathbf{U}_N$ and $\mathbf{B}_N$, respectively. Similarly, we use F instead of $-F$.

The RTR algorithm is a generalization of the classical unconstrained trust-region method to Riemannian manifolds. It is a matrix-free and globally convergent second-order method suitable for large-scale optimization on Riemannian manifolds. Each iteration consists of two steps: (1) approximating the solution of the *trust-region subproblem* and (2) computing a new iterate based on the retracting mapping, defined in Section 3.1. The trust-region subproblem at $\mathbf{U}\in\mathbb{P}_M^K$ is defined as

$$\begin{aligned}\min_{\xi_{\mathbf{U}}\in T_{\mathbf{U}}\mathbb{P}_M^K,\|\xi_{\mathbf{U}}\|\leq\Delta} & F(\mathbf{U}) + g_{\mathbf{U}}(\mathrm{grad}F(\mathbf{U}),\xi_{\mathbf{U}})\\ & + \frac{1}{2}g_{\mathbf{U}}(\mathrm{Hess}F(\mathbf{U})[\xi_{\mathbf{U}}],\xi_{\mathbf{U}}),\end{aligned} \tag{13}$$

where g is the Riemannian metric (8), Δ is the trust-region radius, and $\|\xi_{\mathbf{U}}\| = \sqrt{g_{\mathbf{U}}(\xi_{\mathbf{U}},\xi_{\mathbf{U}})}$. (13) amounts to minimizing a *quadratic model* of the objective function within a trust-region radius of Δ. Here, $\mathrm{grad}F(\mathbf{U})$ is the Riemannian gradient of F and $\mathrm{Hess}F(\mathbf{U})[\xi_{\mathbf{U}}]$ is the Riemannian Hessian of F along $\xi_{\mathbf{U}}$.

The Riemannian gradient and Hessian of an objective function on the manifold can be computed using the expressions in (9), (10) and (11) from the euclidean gradient $\mathrm{Grad}F(\mathbf{U})$ and euclidean Hessian $D\mathrm{Grad}F(\mathbf{U})[\xi_{\mathbf{U}}]$ counterparts.

For our objective function

$$F(\mathbf{U}) = -\frac{1}{2}\mathrm{tr}(\mathbf{B}^T\mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{B}),$$

it is straightforward to check, using matrix calculus [44], that

$$\begin{aligned}&\mathrm{Grad}F(\mathbf{U})\\ &\quad = -\mathbf{B}\mathbf{B}^T\mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1} + \mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{B}\mathbf{B}^T\mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\end{aligned}$$

and

$$\begin{aligned}&D\mathrm{Grad}F(\mathbf{U})[\xi_{\mathbf{U}}]\\ &\quad = -\mathbf{B}\mathbf{B}^T\xi_{\mathbf{U}}(\mathbf{U}^T\mathbf{U})^{-1} + \xi_{\mathbf{U}}(\mathbf{U}^T\mathbf{U})^{-1}(\mathbf{U}^T\mathbf{B}\mathbf{B}^T\mathbf{U})(\mathbf{U}^T\mathbf{U})^{-1}\\ &\qquad + 2\mathbf{B}\mathbf{B}^T\mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathrm{sym}(\xi_{\mathbf{U}}^T\mathbf{U})(\mathbf{U}^T\mathbf{U})^{-1}\\ &\qquad + 2\mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathrm{sym}(\xi_{\mathbf{U}}^T\mathbf{B}\mathbf{B}^T\mathbf{U})(\mathbf{U}^T\mathbf{U})^{-1}\\ &\qquad - 2\mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathrm{sym}(\mathbf{U}^T\xi_{\mathbf{U}})(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{B}\mathbf{B}^T\mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\\ &\qquad - 2\mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{U}^T\mathbf{B}\mathbf{B}^T\mathbf{U}(\mathbf{U}^T\mathbf{U})^{-1}\mathrm{sym}(\mathbf{U}^T\xi_{\mathbf{U}})(\mathbf{U}^T\mathbf{U})^{-1},\end{aligned}$$

where $\mathrm{sym}(\mathbf{A}) = (\mathbf{A}+\mathbf{A}^T)/2$ extracts the symmetric part of a square matrix $\mathbf{A}$.

Once the expressions of the Riemannian gradient and Hessian are obtained, we use the Riemannian trust-region algorithm implemented in the Manopt toolbox [45], [46]. With all the ingredients summarized in Table 1, the main steps of the RTR algorithm are presented in Fig. 1 for the paper to be self-contained.

4.2 Overall Algorithm

The overall algorithm for (4) is based on an alternating optimization scheme in which one matrix variable is updated

TABLE 1
Optimization-Related Ingredients for (12)

Matrix representation of an element in the multinomial manifold $\mathbb{P}_M^K$	A matrix $\mathbf{U}$ of size $M \times K$.
$\mathbb{P}_M^K$	$\mathbb{P}_M^K := \left\{\mathbf{U} := [U_{mk}] \in \mathbb{R}^{M\times K} : \ U_{mk} > 0, \sum_{k=1}^K U_{mk} = 1\right\}$.
Tangent vectors in $T_{\mathbf{U}}\mathbb{P}_M^K$	$\{\xi_{\mathbf{U}} \in \mathbb{R}^{M\times K} : \xi_{\mathbf{U}}\mathbf{1} = \mathbf{0}\}$, where $\mathbf{1} \in \mathbb{R}^K$ is a column vector of ones.
Metric $g_{\mathbf{U}}(\xi_{\mathbf{U}}, \eta_{\mathbf{U}})$ for any $\xi_{\mathbf{U}}, \eta_{\mathbf{U}} \in T_{\mathbf{U}}\mathbb{P}_M^K$	$g_{\mathbf{U}}(\xi_{\mathbf{U}}, \eta_{\mathbf{U}}) = \sum_{l,k} \frac{(\xi_{\mathbf{U}})_{mk}(\eta_{\mathbf{U}})_{mk}}{U_{mk}}$.
Projection of $\mathbf{Z} \in \mathbb{R}^{M\times K}$ onto the tangent space $T_{\mathbf{U}}\mathbb{P}_M^K$ with $\Pi_{\mathbf{U}}$	$\Pi_{\mathbf{U}}(\mathbf{Z}) := \mathbf{Z} - (\alpha\mathbf{1}^T) \odot \mathbf{U}$, where $\alpha = \mathbf{Z}\mathbf{1}^T$ and $\odot$ is the element-wise matrix multiplication operation.
Retraction $R_{\mathbf{U}}(\xi_{\mathbf{U}})$ that maps a search direction $\xi_{\mathbf{U}}$ onto $\mathbb{P}_M^K$	$(\mathbf{U} \odot \exp(t(\xi_{\mathbf{U}} \oslash \mathbf{U}))) \oslash (\mathbf{U} \odot \exp(t(\xi_{\mathbf{U}} \oslash \mathbf{U}))\mathbf{1}\mathbf{1}^T)$, where $\oslash$ is the element-wise matrix inversion and $\exp(\cdot)$ is the element-wise exponential operator.
Riemannian gradient $\mathrm{grad}F(\mathbf{U})$	$\Pi_{\mathbf{U}}(\mathrm{Grad}F(\mathbf{U}) \odot \mathbf{U})$, where $\mathrm{Grad}F(\mathbf{U})$ denotes the euclidean gradient of function F.
Riemannian Hessian $\mathrm{Hess}F(\mathbf{U})[\xi_{\mathbf{U}}]$ along $\xi_{\mathbf{U}} \in T_{\mathbf{U}}\mathbb{P}_M^K$	$\Pi_{\mathbf{U}}(D\mathrm{grad}F(\mathbf{U})[\xi_{\mathbf{U}}] - \frac{1}{2}(\xi_{\mathbf{U}} \odot \mathrm{grad}F(\mathbf{U})) \oslash \mathbf{U})$, where $D\mathrm{grad}F(\mathbf{U})[\xi_{\mathbf{U}}]$ is defined in (11).

while fixing the other matrix variables. In Section 2.3, the updates of the matrix variables $\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_{N-1}$ are shown in closed form. The update of the last mode $\mathbf{U}_N$ is computed by solving (7) with the Riemannian trust-region algorithm discussed in Section 4.1. The procedure of cyclically updating the variables is repeated until a convergence criterion is satisfied. Finally, the membership information $\mathbf{U}_N$ is used with a clustering algorithm, such as the k-means, to learn the final clustering parameters.

The overall algorithm is summarized in Fig. 2.

Remark 1: The tensor clustering model and algorithm proposed in [28] is a special version of the HOSVD model. The idea is to combine tensor projection and k-means by minimizing the distance of projected tensors and the one of K tensor centroids. The objective function is different from our formulation. The algorithm in [28] uses a *heuristic* method to approximately solve the relevant optimization problem.

Remark 2: Our model is also different from the fuzzy k-means algorithm in which the fuzzy cluster membership parameters are applied over the distance between data and centroids. In our model, we regard each data as a linear combination of centroids under relevant membership coefficients.

Remark 3: The algorithm in Fig. 2 starts the *for* loop with updating the membership information $\mathbf{U}_N$. We can also start the for loop with updating factor matrix $\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_{N-1}$. In this case, we can use k-means to make an estimate for $\mathbf{U}_N$.

Remark 4: The overall algorithm is an alternating optimization scheme for (4). The general convergence analysis of alternating optimization schemes is discussed in [47, Section 8.9]. The convergence analysis of RTR is discussed in [30, Chapter 7]. It should be emphasized that the objective function in (7) is nonlinear and RTR, in general, converges to a critical point. Nonetheless, this suffices for the scheme in Fig. 2 for solving (4), which is a challenging optimization problem.

Remark 5: The complexity per iteration of the algorithm in Fig. 2 is dominated by Steps 4 - 7. Computing Step 5 is

Require: An initial guess $\mathbf{U}_0$ on the manifold $\mathbb{P}_M^K$.
Ensure: The minimum $\mathbf{U}$ for the objective function F.
1: Continue the following for loop until a convergence criterion is satisfied.
2: **for** $i = 1, 2, \ldots$ **do**
3: Approximately minimize the trust-region subproblem (13) for a new direction $\xi_{\mathbf{U}}$.
4: Construct the new trial iterate by using retraction mapping $\mathbf{U}_+ = R_{\mathbf{U}}(\xi_{\mathbf{U}})$.
5: Update the iterate by rejecting or accepting $\mathbf{U}_+$ depending on its quality of decrease in the objective function.
6: Update the trust-region radius Δ.
7: **end for**

Fig. 1. The Riemannian trust-region algorithm.

Require: Tensorial data $\mathcal{X} = \{\mathcal{X}_1, ..., \mathcal{X}_M\}$, dimensions $I_1, I_2, \ldots, I_{N-1}$ and the number of clusters K.
Ensure: Factor Matrices $\mathbf{U}_1, \mathbf{U}_2, ..., \mathbf{U}_{N-1}$ and Clusters.
1: Initialize $\mathbf{U}_1, \mathbf{U}_2, ..., \mathbf{U}_{N-1}$.
2: Continue the following for loop until a convergence criterion is satisfied.
3: **for** $i = 1, 2, \ldots$ **do**
4: Call the Riemannian trust-region algorithm (RTR) in Fig. 1 to compute $\mathbf{U}_N$.
5: **for** $n = 1, 2, ..., N-1$ **do**
6: Solve (5) for $\mathbf{U}_n$ by using (6).
7: **end for**
8: **end for**
9: Do k-means over $\mathbf{U}_N$ for the final clustering.

Fig. 2. The overall algorithm for (4).

equivalent to computing SVD for a matrix of size $J_n \times (I_N \times J_{N-1} \times \cdots \times J_{n+1} \times J_{n-1} \times \cdots \times J_1)$, which costs $O(I_N^2) = O(M^2)$ (the number of data) while J_ns are small. As for Step 4, i.e., the RTR algorithm, we only use a fixed number of iterations for the RTR algorithm (see Section 5.3). Furthermore, the computational cost of each iteration within the RTR algorithm depends on objective function, gradient, Hessian, retraction, and projection evaluations. All these involve matrix-vector multiplications (refer to Table 1) which cost $O(M^2)$ and $(\mathbf{U}_N^T\mathbf{U}_N)^{-1}$ which costs $O(K^3)$.

4.3 Initialization

Since we have a nonlinear objective function that is to be optimized over the "curved" multinomial manifold, an initialization has an impact on the final result. In our experiments, shown in the next section, we consider the following strategies for initialization.

Random initialization: This suggests using a random orthogonal basis as each of the factor matrices $\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_{N-1}$.

HOSVD Initialization I: Given the data tensor $\mathcal{X}$, we conduct the HOSVD or HOOI. Keeping the orthogonal matrix factors $\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_{N-1}$ fixed to the initial values, we update $\mathbf{U}_N$ by using the Riemannian trust-region algorithm. However, the orthogonal matrix factors generated by the HOSVD algorithm may not accurately represent data for clustering. In other words, the initialization for $\mathbf{U}_N$ from the HOSVD is far from a good membership representation of clusters. As a result, the the first call to the RTR algorithm requires a larger number of iterations, e.g., 1,000.

HOSVD Initialization II: The HOSVD or HOOI decomposition is designed for a single tensor. As we have to single out the last mode for the purpose of clustering, we design a similar HOSVD algorithm for the first $(N-1)$ modes. In other words, the resulting problem is over a group of $(N-1)$-order tensors $\{\mathcal{X}_i\}_{i=1}^M$ for which we learn their Tucker decompositions under a fixed set of factor matrices $\{\mathbf{U}_1, \ldots, \mathbf{U}_{N-1}\}$ such that $\sum_{i=1}^M \|\mathcal{X}_i - \mathcal{G}_i \times_1 \mathbf{U}_1 \times_2 \cdots \times_{N-1} \mathbf{U}_{N-1}\|_F^2$ is minimized. This problem can be solved in a way similar to the problem of computing the single tensor HOSVD. However, this initialization may not provide any information relating to the tensor representation in terms of the K centroids along the Nth mode.

5 Numerical Experiments

In this section, we present a set of experimental results on synthetic and real-world datasets with high dimensional spatial structures. The intention of these experiments is to demonstrate the performance of our proposed model in comparison to a number of state-of-the-art clustering methods. The algorithm proposed in Fig. 2 compete effectively with the benchmark methods in terms of prediction accuracy.

5.1 Evaluation Metrics

To quantitatively evaluate the clustering results, we adopt two evaluation metrics, the *accuracy* (AC) and the *normalized mutual information* (NMI) metrics [48]. Given a data point $\mathbf{x}_i$, let L and $\hat{L}$ be the ground truth label and the cluster label provided by the clustering approaches, respectively. The AC measure is defined by

$$\mathrm{AC} = \frac{\sum_{i=1}^M \delta(\hat{L}(i), \mathrm{Map}_{(\hat{L},L)}(i))}{M},$$

where M is the total number of samples and the function $\delta(\mathbf{a}, \mathbf{b})$ is set to 1 if and only if $\mathbf{a} = \mathbf{b}$, and 0 otherwise. The operator $\mathrm{Map}(\cdot)$ is the *best mapping function* that permutes $\hat{L}$ to match L, which is usually implemented by the Kuhn-Munkres algorithm [49].

The other metric is the normalized mutual information measure between two index sets L and $\hat{L}$, defined as

$$\mathrm{NMI}(L, \hat{L}) = \frac{\mathrm{MI}(L, \hat{L})}{\max\big(\mathrm{H}(L), \mathrm{H}\big(\hat{L}\big)\big)},$$

where $\mathrm{H}(L)$ and $\mathrm{H}(\hat{L})$ denote the entropy of L and $\hat{L}$, respectively, and

$$\mathrm{MI}\big(L, \hat{L}\big) = \sum_{y \in L} \sum_{x \in \hat{L}} p(x, y) \log_2 \left(\frac{p(x, y)}{p(x)p(y)} \right).$$

$p(y)$ and $p(x)$ denote the marginal probability distribution functions of L and $\hat{L}$, respectively, and $p(x, y)$ is the joint probability distribution function of L and $\hat{L}$. $\mathrm{NMI}(L, \hat{L})$ ranges from 0 to 1, for which the value 1 means that the two sets of clusters are identical and the value 0 means that the two are independent. Different from AC, NMI is *invariant* to permutation of labels, i.e., it does not require the matching processing in advance.

5.2 Dataset Description

This section describes the real-world datasets that we use for assessing the performance of various clustering algorithms.

CBCL face dataset: This face dataset[1] contains images of size 19×19. The goal of clustering for this dataset is to cluster the images into two different classes: face and non-face.

MNIST dataset: This handwritten digits dataset[2] consists of 70,000 handwritten digit images, including a training set of 60,000 examples and a test set of 10,000 examples. It is a subset extracted from a larger set available from NIST. The digits have been size-normalized and centered in the fixed-size 28×28. The images are in grey scale and each image can be treated as a 784-dimension feature vector or a 28×28 second order tensor. This dataset has 10 classes corresponding to the digits 0 to 9, with all the images being labeled.

PIE dataset, CMU: The CMU Pose, Illumination, and Expression (PIE) dataset[3] consists of 41,368 images of 68 people. Each subject was imaged under 13 different poses, 43 different illumination conditions, and with four different expressions. In this paper, we test the algorithms (ours and the benchmarks) on the *Pose27 sub-dataset* as described in [50]. The image size is 32×32.

1. cbcl.mit.edu/projects/cbcl/software-datasets/FaceData2
2. yann.lecun.com/exdb/mnist/
3. www.ri.cmu.edu/research_project_detail.html?project_id=418&menu_id=261

TABLE 2
Results for Random Initialization

Evaluation metric	Mean	Std Var	Best	Worst
AC	0.7120	0.0394	0.7630	0.6150
NMI	0.8189	0.0210	0.8467	0.7717

ORL dataset: The AT&T ORL dataset[4] consists of 10 different images for each of 40 distinct subjects, thus 400 images in total. All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position, under varying lighting, facial expressions (open/closed eyes, smiling/not smiling), and facial details (glasses/no glasses). For our experiments, each image is resized to 32×32 pixels.

Extended Yale B dataset [51]: For this dataset,[5] we use the cropped images and resize them to 32×32 pixels. This dataset has 38 individuals and around 64 near frontal images under different illuminations per individual.

Dynamic texture dataset [52]: The DynTex++ dataset[6] consists of video sequences that contains dynamic river water, fish swimming, smoke, cloud and so on. These videos are labeled with 36 classes and each class has 100 subsequences (a total of $3,600$ subsequences) with a fixed size of $50 \times 50 \times 50$ (50 gray frames). This is a challenging dataset for clustering because most of texture from different class is fairly similar.

5.3 Assessing Initialization Strategies

In Section 4.3 we propose three different initialization schemes for the proposed tensor clustering algorithm. In this experiment, we assess the influence of those initialization strategies on the final clustering accuracy on the sub-dataset Pose27 from the PIE dataset.

Before we present the experimental results, we describe the parameter settings used in the experiments. We randomly select 1,000 data from the sub-dataset Pose27. There are 24 different classes in this chosen subset. The image size is 32×32. We choose the size of core tensors to be 12×12. In the overall algorithm described in Fig. 2, we fix the number of iterations to 250 as the recovered error does not vary much after 250 iterations in most testing cases. For the RTR algorithm in line 4 of Fig. 2, we use the default parameters proposed in the toolbox Manopt with the RTR maximal inner iteration number 30. For the first call to the RTR algorithm, we set the number of maximum outer iterations to 1,000 with Manopt's default initialization for $\mathbf{U}_N$ ($N = 3$ in this case). For subsequent calls to RTR, we use five iterations with the current $\mathbf{U}_N$ as the initial values for memberships. In each overall iteration, we repeat the calls to Lines 5-7 in Fig. 2 twice to make sure that the factors $\mathbf{U}_n$ stable.

We conduct 20 runs for randomly initializing $\mathbf{U}_1$ and $\mathbf{U}_2$. The statistics are reported in Table 2. Under similar parameter settings, $\mathrm{AC} = 0.7620$ and $\mathrm{NMI} = 0.8199$ for HOSVD Initialization I and $\mathrm{AC} = 0.7240$ and $\mathrm{NMI} = 0.8132$ for HOSVD Initialization II, respectively. Both the random initialization and HOSVD initialization II strategies give comparable results and are are shown in Fig. 3b. It should be noted that the reconstructed error does not improve with iterations. On the other hand, Fig. 3a shows the error curve for the HOSVD Initialization I strategy. As HOSVD gives the best estimate in terms of the model error, the error goes up initially when we move away from the best orthogonal factor $\mathbf{U}_3$ to a membership factor. Finally, it stabilizes as shown in Fig. 3a. This leads to the conclusion that among the three initialization strategies discussed in Section 4.3, HOSVD initialization I is the initialization of choice. In subsequent experiments, we initialize all the compared algorithms with HOSVD initialization I.

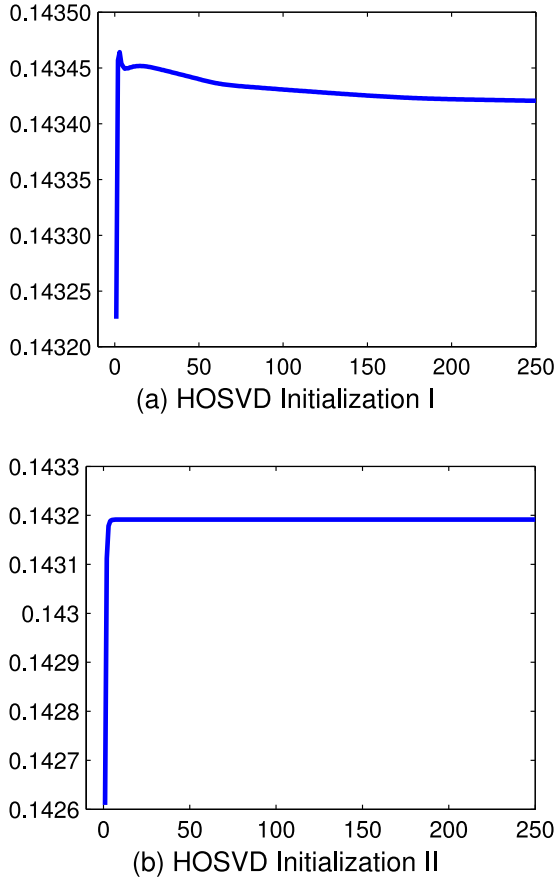

Fig. 3. Errors with iterations. The low errors in the initial phase are due to non satisfiability of constraints.

5.4 Algorithms and Comparisons

There exist many different clustering algorithms such as *spectral clustering approaches* [53] and the recent popular *matrix factorization* approaches [19], [54]. Since the Tucker decomposition is a multidimensional generalization of matrix factorization, we focus on the comparison between our proposed heterogeneous Tucker model with recent matrix factorization based clustering algorithms. The following methods are chosen as the benchmark for numerical comparisons. All the experiments are conducted on a desktop with an Intel Core i5-4670 CPU at 3.40 GHz and with RAM of 8.00 GB.

Multiplicative method for nonnegative matrix factorization (MM) [19]: This nonnegative matrix factorization algorithm is designed for vectorial data that are organized into a data

4. www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html
5. vision.ucsd.edu/leekc/ExtYaleDatabase/ExtYaleB.html
6. www.bernardghanem.com/datasets

TABLE 3
Results on the CBCL Face Dataset with 1,200 Randomly Chosen Data (600 from Each Class)

Evaluation metric	MM	ALS	B-NFM	PGM	DRCC	Tri-ONTD	PCA-B	PCA-Mulitnomial
AC	0.6755 (5.494e-4)	0.6424 (3.255e-4)	0.6305 (3.594e-4)	0.5976 (2.073e-3)	**0.7352** (1.367e-3)	0.5764 (3.503e-5)	0.6394 (3.301e-3)	0.6208 (1.018e-4)
NMI	0.0928 (6.351e-4)	0.0604 (2.505e-4)	0.0509 (2.296e-4)	0.0334 (7.794e-4)	**0.1766** (2.777e-3)	0.0824 (1.836e-4)	0.0669 (1.181e-3)	0.0430 (5.524e-5)

matrix $\mathbf{X}$, i.e., each column represents an image. The matrix $\mathbf{X}$ is factorized into the product of two nonnegative matrices such that $\mathbf{X} = \mathbf{WH}$. The columns of $\mathbf{W}$ are regarded as the *basis vectors* and $\mathbf{H}$ is interpreted as the *weights*, i.e., each data vector is obtained by a weighted linear combination of the the basis vectors. The matrix $\mathbf{H}$ is also interpreted as containing clustering membership information.

Alternating least-squares algorithm (ALS) [54]: This algorithm is designed to solve the nonnegative matrix factorization problem of factorizing the nonnegative data matrix $\mathbf{X}$ as $\mathbf{WH}$ by *alternating optimization* over the factor matrices $\mathbf{W}$ and $\mathbf{H}$ with the nonnegativity constraint.

Bayesian nonnegative matrix factorization (B-NFM) [55]: The model is based on the probabilistic framework in which the matrix factors $\mathbf{W}$ and $\mathbf{H}$ are regarded as random variables characterized by relevant statistical density functions. A *Bayesian algorithm* is proposed for solving the problem.

Projected gradient method for nonnegative matrix factorization (PGM) [20]: This algorithm results from simultaneously optimizing over the nonnegative factors $\mathbf{W}$ and $\mathbf{H}$ to approximately decompose a nonnegative data matrix $\mathbf{X}$ as $\mathbf{WH}$. Lin [20] proposes an efficient *Newton* algorithm.

Dual regularized co-clustering method (DRCC) [56]: The DRCC algorithm is a data clustering algorithm that is based on *semi-nonnegative matrix tri-factorization* with dual regularization over data graph and feature graph by exploiting the geometric structure of the data manifold and the feature manifold.

Nonnegative tri-factor tensor decomposition (Tri-ONTD) [10]: This algorithm is designed for 3-order tensorial data. It proposes a tensor decomposition with nonnegative factor matrices. Since it directly deals with tensor data, it can be considered as a benchmark algorithm for comparison with our proposed method.

Benchmark PCA-B: To demonstrate the benefits of the proposed model and, particularly, the Riemannian algorithm, we take the following revised algorithm as another benchmark algorithm. Instead of the problem (7), we propose to consider the problem (2) for a nonnegative membership matrix $\mathbf{U}_N$ only, i.e., the constraint on the sum of each row to 1 is lifted. This makes the problem formulation simpler, which can be solved by alternatively optimizing of each factor matrices, similar to the procedure in Fig. 2. In this case, optimizing $\mathbf{U}_N$ becomes a *quadratic programming problem with nonnegative constraints*, which can be solved, e.g., using the Matlab function `lsqlin.m`.

PCA-multinomial: The algorithm proposed in Fig. 2, which is based on our proposed heterogeneous Tucker decomposition model, discussed in Section 2.2.

It should be noted that most above described algorithms perform better than (or comparable in some cases) to classical clustering algorithms, like the k-means [10], [56]. Hence, we do not compare with clustering algorithms that deal with vectorized data.

The implementations of the nonnegative matrix factorization algorithms MM, ALS, B-NFM, and PGM are in the Matlab NMF toolbox.[7] The Matlab code for DRCC is provided by the authors of [56]. We implement all the other algorithms. Our proposed algorithm, PCA-Multinomial, is implemented in Matlab using the Manopt toolbox [46] and the MATLAB Tensor Toolbox Version 2.6[8] [8].

5.4.1 Experiment I

We first test all the algorithms on the CBCL face dataset as this is the simplest case where there are only two clusters of images to be learned. The datasizes are chosen from 200 to $1{,}600$ by 100 for each class and the experiment results indicate that when size $= 600$ almost all the algorithms achieved similar accuracy. Under this size, we also test different tensor core sizes varying from 4 to 10 and find better results for our proposed tensor clustering method in the case of tensor core size 8.

Table 3 shows the means and variances (in brackets) for all the algorithms after 10 runs. In Table 3 we see that all the algorithms are comparable for the case when the number of clusters is 2. For this experiment, DRCC has shown better performance than all the other algorithms. However, we also observe that in experiments when $K = 2$ the performance of DRCC degrades. The benchmark algorithm PCA-B is slightly better than the proposed PCA-Multinomial by 1.8 percent in AC. However, the variance of PCA-Multinomial is better in this experiment showing robustness of the Riemannian optimization algorithm.

5.4.2 Experiment II

In this experiment, we assess the impact of the number of classes on the performance of all the algorithms. For this purpose, we conduct all the algorithms on the MNIST handwritten digits dataset. There are 10 different classes, corresponding to the digits 0 to 9. Most machine learning algorithms work well for the case of two clusters, i.e., $K = 2$. For $K > 2$, the performance of different algorithms varies, observing which is the main focus of the present experiment.

We randomly pick digits for training data according to the class number $K = 3, 4, \ldots, 10$ with 100 images for each class. Once the classes are decided randomly, we run all the algorithms five times each with randomly chosen digits in the classes. The experimental results are reported in Table 4 in terms of AC and NMI evaluation metrics. It should be

7. http://cogsys.imm.dtu.dk/toolbox/nmf/index.html.
8. http://www.sandia.gov/tgkolda/TensorToolbox/index-2.6.html

TABLE 4
Results for the MNIST Handwritten Digits with 1,000 Randomly Chosen Images (100 from Each Class)

	Evaluation metric	MM	ALS	B-NFM	PGM	DRCC	PCA-B	PCA-Multinomial
$K=3$	AC	**0.9173** (2.800e-4)	0.9133 (2.6111e-4)	0.8820 (1.644e-4)	0.9160 (1.800e-4)	0.9013 (3.589e-4)	0.8506 (5.274e-3)	0.9114 (1.700e-4)
	NMI	**0.7351** (2.488e-3)	0.7344 (1.654e-3)	0.6806 (6.069e-4)	0.7307 (1.792e-3)	0.7115 (1.704e-3)	0.6231 (1.492e-2)	0.7252 (9.761e-4)
$K=4$	AC	0.7140 (8.518e-4)	0.6520 (8.611e-3)	0.6780 (4.138e-4)	**0.7195** (5.325e-4)	0.6575 (3.237e-3)	0.6670 (4.576e-3)	0.6800 (4.906e-4)
	NMI	0.4654 (1.726e-3)	0.4181 (3.279e-3)	0.4446 (5.964e-4)	**0.4680** (1.181e-3)	0.4071 (3.278e-3)	0.4184 (2.588e-3)	0.4360 (7.243e-4)
$K=5$	AC	0.6216 (7.731e-3)	0.7112 (3.292e-4)	0.6104 (3.839e-3)	0.6764 (1.533e-3)	0.6632 (2.792e-4)	0.6960 (1.370e-3)	**0.7196** (1.334e-3)
	NMI	0.5201 (2.080e-3)	**0.5590** (1.018e-3)	0.4992 (2.921e-3)	0.5373 (9.196e-4)	0.5198 (2.156e-4)	0.5242 (1.130e-3)	**0.5547** (5.525e-4)
$K=6$	AC	0.5370 (4.249e-3)	0.5337 (3.626e-3)	0.5156 (3.812e-3)	0.5466 (4.257e-3)	0.4983 (1.061e-3)	0.5283 (5.234e-3)	**0.5540** (6.386e-3)
	NMI	0.4007 (9.891e-4)	0.3852 (6.482e-4)	0.3980 (1.197e-3)	0.3962 (1.507e-3)	0.3670 (1.348e-3)	0.3850 (3.077e-3)	**0.4083** (1.945e-3)
$K=7$	AC	0.5988 (8.816e-5)	0.5974 (3.295e-3)	0.5762 (1.051e-3)	0.5940 (3.704e-3)	**0.6585** (1.789e-3)	0.6074 (3.704e-3)	0.6231 (5.618e-3)
	NMI	0.5382 (7.593e-4)	0.5378 (1.607e-3)	0.4972 (7.238e-4)	0.5342 (9.776e-4)	**0.5629** (9.366e-4)	0.5097 (1.204e-3)	0.5401 (3.828e-4)
$K=8$	AC	0.6455 (3.217e-3)	0.6162 (8.523e-4)	0.5930 (3.880e-3)	0.5942 (3.267e-3)	0.6270 (2.312e-3)	0.5923 (4.726e-3)	**0.6485** (4.496e-3)
	NMI	0.5427 (5.694e-4)	**0.5496** (6.968e-4)	0.4979 (1.329e-3)	0.5152 (2.243e-3)	0.5033 (3.711e-3)	0.4944 (2.286e-3)	0.5254 (2.367e-3)
$K=9$	AC	0.5211 (1.251e-3)	0.5567 (8.173e-4)	0.5137 (5.528e-4)	0.5302 (1.816e-3)	0.5240 (7.867e-4)	0.5224 (8.213e-4)	**0.5756** (3.569e-3)
	NMI	0.4621 (3.209e-4)	0.4716 (3.473e-4)	0.4343 (3.890e-4)	0.4720 (5.279e-4)	0.4493 (8.811e-4)	0.4449 (1.130e-3)	**0.4774** (5.103e-4)
$K=10$	AC	0.4980 (1.154e-3)	0.4800 (9.220e-4)	0.4474 (5.823e-4)	0.4958 (1.833e-3)	0.4668 (2.846e-3)	0.4800 (2.761e-3)	**0.5126** (1.785e-3)
	NMI	0.4470 (7.543e-4)	0.4505 (5.267e-4)	0.4089 (3.474e-4)	**0.4516** (2.051e-3)	0.4053 (3.402e-3)	0.4142 (8.642e-4)	0.4461 (1.146e-3)

noted that Tri-ONTD algorithm fails for most of cases, and therefore, no results are reported. The results in Table 4 show that the proposed PCA-Mulitnomial performs better in the cases of more than four classes. This is consistent with the results from the first experiment, where we observe a similar behavior. For $K = 3, 4$, the performance of PCA-Multinomial is comparable with that of MM and PGM in both AC and NMI metrics.

Fig. 4 shows the reconstructed representatives from the learned centroids for $K = 4$, given by $\mathcal{X}^* = \mathcal{G}^* \times_1 \mathbf{U}_1 \times_2 \cdots \times_{N-1} \mathbf{U}_{N-1}$ with the learned low dimensional centroids $\mathcal{G}^*$.

5.4.3 Experiment III

A test similar to Experiment II is conducted on the PIE dataset to further confirm the main conclusion form Experiment II, that the performance of PCA-multinomial is better when are a larger number of clusters are sought. We do not report the performance of the Tri-ONTD algorithm as it does not provide meaningful results in our experiments. The PIE dataset contains 68 clusters, each with 42 objects. In this experiment, we test the algorithms on the data of 68, 58, 48, 38, 28, 18, and eight clusters, respectively. To maintain the data size to be around 600, we randomly pick nine images for the case of cluster 68, 11 for 58, 13 for 48, 16 for 38, 21 for 28, 33 for 18, and 42 for 8. The procedure is repeated for five runs. In the last case, the data size is 336 and each run is with eight different clusters.

The results are collected in Table 5. PCA-Multinomial performs better than the others, except for the case $K = 8$, where MM takes the lead. Part of this behavior is due to *lack of sufficient data* in the case of cluster $K = 8$. However NMI scores are all over 80 percent, except for the case $K = 8$. Both DRCC and PCA-B do not show meaningful results on this dataset.

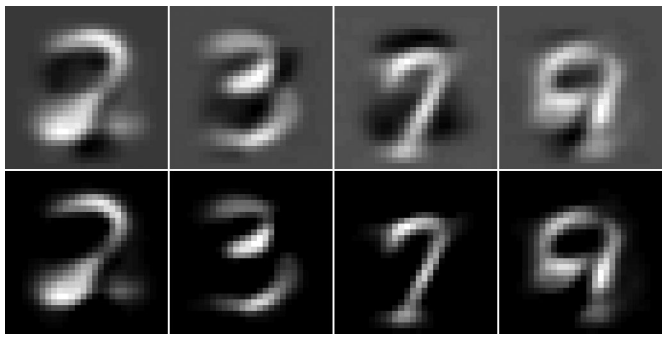

Fig. 4. The first row shows the Reconstructed Digits 2, 3, 7 and 9 from the learned centroids. The second row shows the reconstructed digits with positive values.

TABLE 5
Results on PIE Face Dataset with about 600 Randomly Chosen Data

	Evaluation metric	MM	ALS	B-NFM	PGM	DRCC	PCA-B	PCA-Multinomial
$K = 68$	AC	0.6088 (4.563e-4)	0.5633 (2.101e-4)	0.6192 (4.311e-4)	0.5120 (7.590e-4)	0.2745 (2.736e-4)	0.4314 (1.018e-3)	**0.6533** (8.749e-4)
	NMI	0.8066 (2.222e-4)	0.7791 (4.083e-5)	0.8099 (9.111e-5)	0.7540 (9.963e-5)	0.5652 (3.231e-4)	0.6828 (3.197e-4)	**0.8634** (2.484e-4)
$K = 58$	AC	0.6182 (5.719e-4)	0.5730 (3.549e-3)	0.5968 (1.884e-4)	0.5482 (1.362e-3)	0.2504 (2.215e-4)	0.4125 (1.318e-3)	**0.6781** (9.944e-4)
	NMI	0.8027 (2.504e-4)	0.7790 (6.398e-4)	0.7955 (2.789e-5)	0.7580 (2.096e-4)	0.5140 (2.995e-4)	0.6555 (8.198e-4)	**0.8742** (1.483e-4)
$K = 48$	AC	0.6112 (1.625e-4)	0.5964 (6.027e-4)	0.6147 (1.777e-3)	0.5951 (5.105e-4)	0.2875 (4.925e-4)	0.4228 (6.271e-4)	**0.6990** (7.863e-4)
	NMI	0.7899 (1.084e-4)	0.7812 (4.319e-4)	0.7896 (1.928e-4)	0.7775 (1.633e-4)	0.5266 (2.659e-4)	0.6491 (1.115e-4)	**0.8480** (8.585e-5)
$K = 38$	AC	0.5799 (2.292e-3)	0.6197 (9.405e-4)	0.6138 (9.892e-4)	0.5930 (3.646e-4)	0.2621 (3.578e-4)	0.3911 (1.227e-3)	**0.7118** (1.384e-3)
	NMI	0.7552 (3.956e-4)	0.7769 (6.884e-4)	0.7717 (5.941e-4)	0.7553 (1.132e-4)	0.4461 (1.461e-4)	0.5949 (5.438e-4)	**0.8516** (1.716e-4)
$K = 28$	AC	0.6224 (2.564e-3)	0.6214 (1.448e-3)	0.6329 (4.329e-4)	0.6408 (8.523e-4)	0.2619 (2.704e-4)	0.4212 (4.090e-3)	**0.6877** (4.106e-3)
	NMI	0.7470 (4.750e-4)	0.7610 (5.707e-4)	0.7586 (1.975e-4)	0.7519 (3.880e-4)	0.4033 (3.592e-4)	0.5669 (2.179e-3)	**0.8153** (5.873e-4)
$K = 18$	AC	0.6367 (6.368e-4)	0.6387 (1.002e-2)	0.5892 (5.057e-3)	0.6511 (2.232e-3)	0.2962 (2.326e-3)	0.4865 (6.932e-3)	**0.7329** (1.352e-3)
	NMI	0.7290 (5.039e-4)	0.7433 (2.437e-3)	0.7089 (8.551e-4)	0.7377 (4.477e-4)	0.3836 (1.570e-3)	0.5917 (3.972e-3)	**0.8010** (7.787e-4)
$K = 8$	AC	**0.6690** (1.073e-3)	0.6256 (2.698e-3)	0.6321 (9.592e-4)	0.6285 (1.396e-3)	0.3809 (9.553e-3)	0.4785 (3.036e-2)	0.6523 (1.634e-2)
	NMI	0.6512 (4.287e-3)	0.6797 (5.646e-4)	0.6382 (6.623e-4)	0.6289 (1.509e-3)	0.3110 (1.269e-2)	0.4403 (3.941e-2)	**0.6803** (1.057e-2)

5.4.4 Experiment IV

In this experiment, we test the algorithms on both the ORL dataset of 40 subjects with 400 images and the YaleB extended dataset of 38 subjects with $2{,}414$ images.

For the ORL dataset, we randomly choose 10 subjects for five runs of the algorithm. The results in Table 6 show that the PCA-Multinomial algorithm outperforms all the other algorithms on the ORL dataset in terms of both AC and NMI scores.

For the YaleB extended dataset, we use all the images. Hence, only one test is conducted and the results are summarized in Table 6. It should be noted that all the algorithms show poor performance on the YaleB dataset although the PCA-Multinomial algorithm has relatively better performance. This is due to the large variants among this dataset.

5.4.5 Experiment V

In this experiment, we test the PCA-Multinomial algorithm against the benchmark PCA-B algorithm on the DynTex++ dataset. The other algorithms are not appropriate for this dataset. Tri-ONTD works for matrix data, i.e., it can handle a set of images or 3-order tensors, whereas the rest work with vectorial data, i.e., can handle only matrices. The core tensor size is fixed to $(30, 30, 30)$. We conduct tests on the cases of up to 12 classes. In each test, the total number of training video *subsequences* is determined to 200 for 2 classes, 270 for three classes, 360 for four classes, 480 for six classes, 600 for eight classes and 10 classes, and 660 for 12 classes. The classes are randomly chosen. For each case, five runs are conducted by randomly choosing video subsequences in the chosen classes, except for the case of two classes, where the only 200 subsequences are all used in a single run.

TABLE 6
Results on ORL and YaleB Face Datasets

	Evaluation metric	MM	ALS	B-NFM	PGM	DRCC	PCA-B	PCA-Multinomial
ORL	AC	0.6440 (3.730e-3)	0.6820 (9.070e-3)	0.6280 (8.070e-3)	0.6300 (3.650e-3)	0.5580 (4.870e-3)	0.6360 (5.130e-3)	**0.7340** (2.230e-3)
	NMI	0.7308 (1.842e-3)	0.7269 (5.946e-3)	0.7107 (2.688e-3)	0.7236 (1.668e-3)	0.6498 (8.727e-4)	0.6968 (5.467e-3)	**0.7996** (7.283e-4)
YaleB	AC	0.2175	0.2270	0.2104	0.2084	0.1002	0.2117	**0.3293**
	NMI	0.3534	0.3470	0.3508	0.3477	0.1455	0.3402	**0.4772**

TABLE 7
Results on the DynTex++ Dataset

Models	Evaluation metric	2 Classes	3 Classes	4 Classes	6 Classes	8 Classes	10 Classes	12 Classes
PCA-B	AC	0.5550	0.4230	0.3727	0.2983	0.2507	0.1937	0.2024
		-	(4.069e-3)	(1.429e-3)	(4.088e-4)	(5.244e-4)	(8.527e-5)	(3.416e-4)
	NMI	0.1034	0.1849	0.2261	0.2079	0.2259	0.2451	0.2520
		-	(8.471e-3)	(1.667e-3)	(4.817e-4)	(2.926e-4)	(1.057e-3)	(9.549e-5)
PCA-Multinomial	AC	**0.7901**	**0.5827**	**0.5767**	**0.3842**	**0.4007**	**0.3300**	**0.2980**
		-	(5.522e-4)	(6.350e-3)	(1.361e-3)	(1.325e-3)	(1.272e-3)	(2.519e-4)
	NMI	**0.4056**	**0.2047**	**0.3360**	**0.3012**	**0.3538**	**0.3285**	**0.3444**
		-	(5.924e-4)	(6.116e-4)	(9.165e-4)	(5.356e-4)	(4.801e-4)	(1.563e-4)

The results are summarized in Table 7. Particularly, the results show the challenging nature of clustering dynamic textures for $K \geq 6$ classes. In all the cases, PCA-Multinomial algorithm consistently performs better than the benchmark PCA-B. This once again confirms the benefits of using the RTR algorithm based on the Riemannian geometry of the multinomial manifold.

6 Conclusion

We proposed a heterogeneous Tucker decomposition model for tensor clustering. The model simultaneously conducts dimensionality reduction and optimizes membership representation. The dimensionality reduction is conducted along the first $(N-1)$-modes while optimizing membership on the last mode of tensorial data. The resulting optimization problem is a nonlinear optimization problem over the special matrix multinomial manifold. To apply the Riemannian trust region algorithm to the problem, we explored the Riemannian geometry of the manifold endowed with the Fisher information metric. Finally, we implemented the clustering algorithm based on Riemannian optimization. We used a number of real-world datasets to compare the proposed algorithm with existing state-of-the-art nonnegative matrix factorization methods. Numerical results illustrate the good clustering performance of the proposed algorithm particularly for higher class numbers.

Further work can be carried out in several different directions. For example, in the current work we use an alternating procedure to optimize over all the factor matrices $\mathbf{U}_1$, $\mathbf{U}_2$, …, $\mathbf{U}_N$. However, this is not the only way to solve the problem. As the constraint $\mathbf{U}_n^T\mathbf{U}_n = \mathbf{I}$ for $n = 1, 2, \ldots, N-1$ defines the Stiefel manifold, the overall optimization problem (4) is defined over the product manifold of $(N-1)$ Stiefel manifolds and one on the multinomial manifold. This can be directly optimized over the entire product of N manifolds, e.g., using Manopt. Another issue that needs to be further investigated is how to scale the computation of (6) to higher-dimensional data.

Acknowledgments

The authors wish to thank all the anonymous reviewers for their comments and suggestions to improve the quality of the paper. They also thank Kasper Winther Jørgensen for providing the NMF toolbox. This research was supported under Australian Research Council's Discovery Projects funding scheme (project number DP130100364). This work was also partially supported by the National Natural Science Foundation of China (No. 61370119, 61133003 and 61390510) and the Natural Science Foundation of Beijing (No. 4132013). Bamdev Mishra is supported as an F.R.S-FNRS research fellow (Belgian Fund for Scientific Research).

References

[1] B. Schölkopf and A. Smola, *Learning with Kernels*. Cambridge, MA, USA: MIT Press, 2002.
[2] H. Lua, K. N. Plataniotisb, and A. N. Venetsanopoulos, "A survey of multilinear subspace learning for tensor data," *Pattern Recognit.*, vol. 44, pp. 1540–1551, 2011.
[3] X. Guo, L. Zhang, and X. Huang, "Hyperspectral image noise reduction based on rank-1 tensor decomposition," *ISPRS J. Photogrammetry Remote Sensing*, vol. 83, pp. 50–63, 2013.
[4] J. Sun, H. Zeng, H. Liu, Y. Lu, and Z. Chen, "CubeSVD: A novel approach to personalized web search," in *Proc. 14th Int. Conf. World Wide Web*, 2005, pp. 382–390.
[5] D. Cai, X. He, and J. Han, "Tensor space model for document analysis," in *Proc. 29th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2006, pp. 625–626.
[6] R. Forsati, M. Mahdavi, M. Shamsfard, and M. R. Meybodi, "Efficient stochastic algorithms for document clustering," *Inf. Sci.*, vol. 220, pp. 269–291, 2013.
[7] T. G. Kolda, "Multilinear operators for higher-order decompositions," Sandia Nat. Lab., Tech. Rep. Sandai Report, SAND2006-281, Livermore, CA 94551, pp. 1–28, Apr. 2006.
[8] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, 2009.
[9] G. A. Shaw and H. Burke, "Spectral imaging for remote sensing," *Lincoln Lab. J.*, vol. 14, no. 1, pp. 3–28, 2003.
[10] Z. Y. Zhang, T. Li, and C. Ding, "Non-negative tri-factor tensor decompositions with applications," *Knowl. Inf. Syst.*, vol. 34, pp. 243–265, 2013.
[11] S. Aja-Fernández, G. R. Luis, D. Tao, and X. Li, *Tensors in Image Processing and Computer Vision*. New York, NY, USA: Springer, 2009.
[12] Y. Tang, R. Salakhutdinov, and G. Hinton, "Tensor analyzers," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 163–171.
[13] A. Shashua and T. Hazan, "Non-negative tensor factorization with applications to statistics and computer vision," in *Proc. Int. Conf. Mach. Learn.*, 2005, pp. 792–799.
[14] M. Morup, "Applications of tensor (multiway array) factorizations and decompositions in data mining," *Wiley Interdisciplinary Rev.: Data Mining Knowl. Discovery*, vol. 1, no. 1, pp. 24–40, 2011.
[15] H. Kiers, "Towards a standardized notation and terminology in multiway analysis," *J. Chemometrics*, vol. 14, no. 3, pp. 105–122, 2000.
[16] L. DeLathauwer, B. DeMoor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Anal. Appl.*, vol. 21, no. 4, pp. 1253–1278, 2001.
[17] L. DeLathauwer, B. DeMoor, and J. Vandewalle, "On the best rank-1 and rank-$(r_1, r_2, \cdots, r_n)$ approximation of higherorder tensors," *SIAM J. Matrix Anal. Appl.*, vol. 21, pp. 1324–1342, 2000.
[18] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.
[19] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, pp. 556–562.

[20] C. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural Comput.*, vol. 19, no. 10, pp. 2756–2779, 2007.
[21] M. Welling and M. Weber, "Positive tensor factorization," *Pattern Recognit. Lett.*, vol. 22, pp. 1255–1261, 2001.
[22] M. P. Friedlander and K. Hatz, "Computing non-negative tensor factorizations," *Optim. Methods Softw.*, vol. 23, no. 4, pp. 631–647, 2008.
[23] J. Liu, J. Liu, P. Wonka, and J. Ye, "Sparse non-negative tensor factorization using columnwise coordinate descent," *Pattern Recognit.*, vol. 45, pp. 649–656, 2012.
[24] F. Wu, X. Tan, Y. Yang, D. Tao, S. Tang, and Y. Zhuang, "Supervised nonnegative tensor factorization with maximum-margin constraint," in *Proc. 27th AAAI Conf. Artif. Intell.*, 2013, pp. 962–968.
[25] H. Liu, Z. Wu, X. Li, D. Cai, and T. S. Huang, "Constrained nonnegative matrix factorization for image representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1299–1311, Jul. 2012.
[26] M. Vichi, R. Rocci, and H. Kiers, "Simultaneous component and clustering models for three-way data: Within and between approaches," *J. Classification*, vol. 24, no. 1, pp. 71–98, 2007.
[27] E. Acar and B. Yener, "Unsupervised multiway data analysis: A literature survey," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 1, pp. 6–20, Jan. 2009.
[28] W. Peng and T. Li, "Tensor clustering via adaptive subspace iteration," *Intell. Data Anal.*, vol. 15, pp. 695–713, 2011.
[29] Y. Sun, J. Gao, X. Hong, Y. Guo, and C. J. Harris, "Dimensionality reduction assisted tensor clustering," in *Proc. Int. Joint Conf. Neural Netw.*, 2014, pp. 1565–1572.
[30] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton, NJ, USA: Princeton Univ. Press, 2008.
[31] B. Vandereycken, "Riemannian and multilevel optimization for rank constrained matrix problems," PhD dissertation, Faculty of Eng., Katholieke Universiteit Leuven, B-3001 Leuven, Belgium, 2010.
[32] G. Meyer, "Geometric optimization algorithms for linear regression on fixed-rank matrices," PhD dissertation, Univ. of Liège, B-4031 Liège, Belgium, 2011.
[33] B. Mishra, G. Meyer, F. Bach, and R. Sepulchre, "Low-rank optimization with trace norm penalty," *SIAM J. Optim.*, vol. 23, no. 4, pp. 2124–2149, 2013.
[34] N. Boumal, "Optimization and estimation on manifolds," PhD dissertation, Université Catholique de Louvain, Belgium, 2014.
[35] Y. Xie, J. Ho, and B. Vemuri, "On a nonlinear generalization of sparse coding and dictionary learning," *J. Mach. Learn. Res.: Workshop Conf. Proc.*, vol. 28, no. 3, pp. 1480–1488, 2013.
[36] M. Harandi, R. Hartley, C. Shen, B. Lovell, and C. Sanderson, "Extrinsic methods for coding and dictionary learning on Grassmann manifolds," *Int. J. Comput. Vis.*, pp. 1–41, 2015, Doi: 10.1007/s11263-015-0833-x.
[37] Y. M. Lui, "Advances in matrix manifolds for computer vision," *Image Vis. Comput.*, vol. 30, pp. 380–388, 2012.
[38] G. Lebanon and J. Lafferty, "Hyperplane margin classifiers on the multinomial manifold," in *Proc. 21st Int. Conf. Mach. Learn.*, Banff, Canada, 2004, p. 66.
[39] R. Inokuchi and S. Miyamoto, "c-means clustering on the multinomial manifold," in *Proc. 4th Int. Conf. Model. Decisions Artif. Intell.*, 2007, vol. 4617, pp. 261–268.
[40] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York, NY, USA: Plenum Press, 1981.
[41] P. Tseng and S. Yun, "A coordinate gradient descent method for linearly constrained smooth optimization and support vector machines training," *Comput. Optim. Appl.*, vol. 47, pp. 179–206, 2010.
[42] J. M. Lee, *Introduction to Smooth Manifolds* (series Graduate Texts in Mathematics), vol. 218. New York, NY, USA: Springer, 2002.
[43] D. Zhang, X. Chen, and W. S. Lee, "Text classification with kernels on the multinomial manifold," in *Proc. 28th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2005, pp. 266–273.
[44] K. B. Petersen and M. S. Pedersen. (2008, Oct.). The matrix cookbook. version 20081110 [Online]. Available: http://www2.imm.dtu.dk/pubdb/p.php?3274
[45] P.-A. Absil, C. G. Baker, and K. A. Gallivan, "Trust-region methods on Riemannian manifolds," *Found. Comput. Math.*, vol. 7, pp. 303–330, 2007.
[46] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre, "Manopt, a Matlab toolbox for optimization on manifolds," *J. Mach. Learn. Res.*, vol. 15, pp. 1455–1459, 2014.
[47] D. G. Luenberger, and Y. Ye, *Linear and Nonlinear Programming*. New York, NY, USA: Springer, 2008.
[48] D. Cai, X. He, X. Wang, H. Bao, and J. Han, "Locality preserving nonnegative matrix factorization," in *Proc. Int. Joint Conf. Artif. Intell.*, 2009, pp. 1010–1015.
[49] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, Jan. 2001.
[50] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1548–1560, Aug. 2011.
[51] K. C. Lee, J. Ho, and D. Kriegman, "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 5, pp. 684–698, May 2005.
[52] B. Ghanem and N. Ahuja, "Maximum margin distance learning for dynamic texture recognition," in *Proc. Eur. Conf. Comput. Vis.*, Part II, LNCS 6312, pp. 223–236, 2010.
[53] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.
[54] H. Kim and H. Park, "Sparse non-negative matrix factorization via alternating non-negativity-constrained least squares for microarray data analysis," *Bioinformatics*, vol. 23, no. 12, pp. 1495–1502, 2007.
[55] M. Schmidt, O. Winther, and L. Hansen, "Bayesian non-negative matrix factorization," in *Proc. Int. Conf. Independent Component Anal. Signal Separation*, 2009, vol. 5541, pp. 540–547.
[56] Q. Gu and J. Zhou, "Co-clustering on manifolds," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Paris, France, 2009, pp. 359–368.

Yanfeng Sun received the PhD degree from the Dalian University of Technology, P.R. China, in 1993. She is a professor in the College of Metropolitan Transportation, Beijing University of Technology. Her research interest covers pattern recognition and image processing.

Junbin Gao received the BSc degree in computational mathematics from the Huazhong University of Science and Technology (HUST), China, in 1982, and the PhD degree from the Dalian University of Technology, China, in 1991. He is a professor in computing science in the School of Computing and Mathematics, Charles Sturt University, Australia. He was a senior lecturer, a lecturer in computer science from 2001 to 2005 at the University of New England, Australia. From 1982 to 2001, he was an associate lecturer, lecturer, associate professor, and a professor in the Department of Mathematics at HUST. His main research interests include machine learning, data mining, Bayesian learning and inference, and image analysis.

Xia Hong received the BSc and Msc degree from the National University of Defense Technology, P. R. China, in 1984 and 1987, respectively, and the PhD degree from the University of Sheffield, United Kingdom, all in automatic control. She was a research assistant in the Beijing Institute of Systems Engineering, Beijing, China, from 1987-1993. She was a research fellow in the Department of Electronics and Computer Science, University of Southampton from 1997-2001. She is currently a professor at the School of Systems Engineering, University of Reading. She is actively engaged in research into nonlinear systems identification, data modelling, estimation and intelligent control, neural networks, pattern recognition, learning theory, and their applications. She has published more than 100 research papers, and coauthored a research book. She was awarded the Donald Julius Groen Prize by IMechE in 1999.

Bamdev Mishra received the BTech and MTech degrees in electrical engineering from the Indian Institute of Technology Bombay, India, in 2010, and the PhD degree from the Department of Electrical Engineering and Computer Science, University of Liège, Belgium, in 2014. He is currently a postdoctoral fellow at the University of Liège and a visiting research associate at the University of Cambridge. He is a research fellow (aspirant) of the Belgian National Fund for Scientific Research (FNRS). His current research is on nonlinear optimization on matrix manifolds and machine learning.

Baocai Yin received the PhD degree from the Dalian University of Technology, P.R. China, in 1993. He is a professor in the College of Metropolitan Transportation, Beijing University of Technology. His research interest covers digital multimedia, multi-functional perception, virtual reality, and computer graphics.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.