2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Dimensionality Reduction Assisted Tensor Clustering

Yanfeng Sun, Junbin Gao, Xia Hong, Yi Guo and Chris J. Harris

***Abstract*— This paper is concerned with tensor clustering with the assistance of dimensionality reduction approaches. A class of formulation for tensor clustering is introduced based on tensor Tucker decomposition models. In this formulation, an extra tensor mode is formed by a collection of tensors of the same dimensions and then used to assist a Tucker decomposition in order to achieve data dimensionality reduction. We design two types of clustering models for the tensors: PCA Tensor Clustering model and Non-negative Tensor Clustering model, by utilizing different regularizations. The tensor clustering can thus be solved by the optimization method based on the alternative coordinate scheme. Interestingly, our experiments show that the proposed models yield comparable or even better performance compared to most recent clustering algorithms based on matrix factorization.**

***Index terms*—** Tensor Tucker Decomposition, Tensor Clustering, Matrix Factorization, Tensor PCA.

I. Introduction

In the last two decades, the advance of modern sensing, networking, communication and storage technologies pave the way for the availability of multidimensional data with high dimensionality. For example, the remote sensing is producing massive multidimensional data that needs to be carefully analyzed. One of characteristics of these gigantic datasets is they often have a large amount of redundancies. This gives arise to the development of a low-dimensional representation that best a low-dimensional representation that best assists a range of learning tasks to avoid the so-called "curse of dimensionality" [1]. Many data processing tasks involve manipulating multi-dimension objects, such as video data [2], remote sensing data [3] and text documents analysis [4]. The multi-dimensional data are known as tensors [5], [6], where data elements are addressed by more than two indices. An Nth-order tensor is an element of the tensor product of N vector spaces. A 2D matrix is an example of the 2nd-order tensor. Similarly, hyperspectral imagery [7] is naturally a three-dimensional (3D) data cube containing both spatial and spectral dimensions. It is believed that spectral and spatial structures of hyperspectral data should be considered simultaneously in clustering or classification, so that the accuracy could be further improved. In this regard, we prefer to treat such 3D cube as a whole. Indeed, the data tensors are defined in a high-dimensional space, thus applying standard machine learning methods directly to such data gives arise to the problems of high demands in computational and memory costs, as well as poor model generalization. A fundamental strategy to address these issues is to compress the data while capturing the dominant trends or to find the most suitable "sparse" representation of the data, simultanenously.

Principal component analysis (PCA) [8] decomposes data object in terms of vectors, i.e., the principal components, while the singular value decomposition (SVD) [9] decomposes the 2nd-order tensors, i.e. matrices, in terms of principal components along two modes, the row and the column. One of trivial ways to deal with tensor data is to vectorize them before any analysis is applied. Obviously this strategy breaks the higher order dependencies presented in the natural data structure. For example, it is a common practice to convert image data, a typical 2nd-order tensorial structure, into vectors for processing. However the 2D spatial structural information that can potentially lead to more compact and useful representations vanishes in the process of vectorization. In the past ten to fifteen years, new approaches capable of possessing structural information of tensorial data have been proposed [6]. These are finding their ways into applications in computer vision [10], [11] and machine learning [12], [13].

There exist many different kinds of tensor decomposition models, amongest which the CANDECOMP (canonical decomposition)/PARAFAC (parallel factors) or in short CP decomposition [14], and the Tucker decomposition are two fundamental models for tensor decomposition (please refer to the survey paper [6] for details). It can be noted that the CP decomposition is a special case of Tucker decomposition [6] where factor matrices have the 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 CP or Tucker decomposition. As an extension of the classical SVD, the higher order SVD [15] is a special case of general Tucker decomposition in which the core tensor is of the same dimension as the tensor to be decomposed and all the mode matrices have orthonormal columns. The classic PCA has several extensions for a given set of 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 is also called the Higher-order Orthogonal Iteration (HOOI) in [16].

In applications where data are non-negative such as im-

Yanfeng Sun is with Beijing Municipal Key Lab of Multimedia and Intelligent Software Technology, Beijing University of Technology, Beijing 100124, China, (yfsun@bjut.edu.cn); Junbin Gao is with the School of Computing and Mathematics, Charles Sturt University, Bathurst, NSW 2795, Australia (jbgao@csu.edu.au); Xia Hong is with the School of Systems Engineering, University of Reading, Reading, RG6 6AY,UK (x.hong@reading.ac.uk); Yi Guo is with CSIRO Mathematics, Informatics and Statistics, North Ryde, NSW 1670, Australia (yi.guo@csiro.au); Chris J. Harris is with Electronics and Computer Science, University of Southampton, Southampton, UK, (cjh@ecs.soton.ac.uk).

This work was supported by Australian Research Council (ARC) under Grant DP130100364. Both Yanfeng Sun and Junbin Gao are supported the National Natural Science Foundation of China, under Grant 61370119.

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

ages, the non-negative matrix factorization (NFM) has proven to be a successful approach for detecting essential features in the data [17]. Several efficient algorithms have been proposed [18], [19]. Naturally NFM has been extended to non-negative tensor factorization (NTF) and NTF has been actively investigated [12], [20], [21], [22], [23].

Another trend in tensor decomposition research is to introduce more structures in decomposition models. Recently Zhang et al. [24] consider a new tensor decomposition model which they called Tri-ONTD (Tri-favctor orthogonal non-negative tensor decomposition). The fundamental aim is to discover the common characteristics of a series of matrix data. A straightforward application of Tri-ONTD is to identify cluster structures of the dataset. The core idea behind this model is based on the centroid-based clustering algorithms such as the well-known K-means algorithm. The idea of introducing new structures can also be seen in [25] under the different context with useful applications in image representation.

This paper is dedicated to developing a tensor factorization based clustering algorithm, referred to as Dimensionality Reductin Assisted Tensor Clustering (DRATC). In this algorithm, the tensor decompostion is used as a way to learn low dimensional representation of the given tensors and simultaneously clustering is conducted by coupling the approximation and clustering learning. The contributions of this paper are summarized as follows

1) We propose a Tucker tensor decomposition model with specific core structures aiming at tensor clustering. The centroid tensors are of lower dimensionality along each tensor mode. Depending on different constraints, two clustering models, i.e. PCA Tensor Clustering Model and Non-negative Tensor Clustering model, are proposed in the paper.
2) The standard centroid based clustering algorithms are unsupervised which are not able to incorporate any available label information. In this paper, as the second contribution, we extend the newly proposed tensor clustering algorithms to semi-supervised learning algorithms.
3) The idea behind our proposed models shares some similarity as the recent work by Adler et al. [26], yet with a key difference that the clustering in [26] is conducted in the separate post stage of learning a sparse representation under appropriate dictionary. In our proposed models and algorithms, both learning stages are integrated in one framework.

The paper is organized as follows. Section II is devoted to reviewing the concepts of tensors and introducing necessary notations based on which several tensorial clustering models are proposed. In Section III, we investigate an algorithm for each proposed model. To validate the proposed models and algorithms, a number of numerical experiments over synthetic data and real world data are conducted and presented in Section IV. Further analysis and conclusion are presented in the last section.

II. Tensor Clustering Model based on Dimensionality Reduction

A. Tensor Notation and Operations

In the sequel, we denote 1D vector by lowercase bold symbols like $\mathbf{v}$, 2D matrix by uppercase bold 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 vector $\mathbf{v} \in \mathbb{R}^{I_n}$, denoted by $\mathcal{X} \overline{\times}_n \mathbf{v}$, is defined elementwise,

$$(\mathcal{X} \overline{\times}_n \mathbf{v})_{i_1 \ldots i_{n-1} i_{n+1} \ldots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \ldots i_n \ldots i_N} v_{i_n}$$

Actually the result is a tensor of order $N-1$ with dimension $I_1 \times \cdots I_{n-1} \times I_{n+1} \cdots \times I_N$. Similarly the n-mode product of an N-order tensor $\mathcal{X}$ with a matrix $\mathbf{U}^{(n)} \in \mathbb{R}^{J_n \times I_n}$ is denoted by $\mathcal{X} \times_n U^{(n)}$. The result is an N-order tensor of dimension $I_1 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N$. Elementwise, the n-mode product can be expressed as

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

Note the difference between two n-mode products. Vectorial n-mode product reduces the order of the tensor while matrix n-mode product maintains the same order. Although a vector can be regarded as a one column matrix, the results of two n-mode products are different for the same vector.

Under the Tucker decomposition [6], for a given N-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times \cdots \times I_N}$, we seek a Tucker model, as defined below, to approximate the tensor

$$\begin{aligned} \mathcal{X} &\approx \mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \times_3 \cdots \times_N \mathbf{U}^{(N)} \\ &\triangleq [\![\mathcal{G}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, ..., \mathbf{U}^{(N)}]\!] \end{aligned} \tag{1}$$

where $\mathcal{G}$ is an N-order tensor of dimension $J_1 \times \cdots \times J_n \times \cdots \times J_N$ with $J_n \leq I_n$, called the core tensor, and $\mathbf{U}^{(n)} \in \mathbb{R}^{J_n \times I_n}$ is the matrix applied along mode-n. In this decomposition, we can say that the core tensor $\mathcal{G}$ is a lower dimensional representation of the tensor $\mathcal{X}$. The Tucker decomposition is a form of higher-order PCA [27] where all the matrices $\mathbf{U}^{(n)}$ are shared by a group of given tensors.

B. Tensor Clustering Model Formulation

In this section, we propose the problem of tensor clustering. Let $\mathcal{D} = \{\mathcal{X}_l\}_{l=1}^L \subset \mathbb{R}^{I_1 \times \cdots \times I_n \times \cdots \times I_N}$ be L N-order tensors of dimension $I_1 \times \cdots \times I_n \times \cdots \times I_N$. We wish to categorize $\mathcal{D}$ into K groups under some criterion. One typical way to deal with this problem is to reduce the dimension of tensors by using N-order PCA or the N-order SVD [15] as

$$\mathcal{X}_l \approx [\![\mathcal{G}_l; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, ..., \mathbf{U}^{(N)}]\!] \tag{2}$$

Then one can apply a clustering algorithm over $\{\mathcal{G}_l\}_{l=1}^L \subset \mathbb{R}^{J_1 \times \cdots \times J_n \times \cdots \times J_N}$. For example, an easy way is to convert each low-dimension tensor $\mathcal{G}_l$ into a vector $\text{vec}(\mathcal{G}_l)$ of dimension $\prod_{n=1}^N J_n$ and then to cluster $\{\text{vec}(\mathcal{G}_l)\}_{l=1}^L$

with an algorithm like K-means [8]. More generally, we seek K tensorial centroids $\mathcal{C}_k \in \mathbb{R}^{J_1 \times \cdots \times J_n \times \cdots \times J_N}$ and a membership matrix $\mathbf{H} = (h_{lk}) \in \mathbb{R}^{L \times K}$ such that the following error is minimized

$$\min_{H, \mathcal{C}_1, ..., \mathcal{C}_K} \sum_{l=1}^{L} \|\mathcal{G}_l - \sum_{k=1}^{K} h_{lk} \mathcal{C}_k\|_F^2. \quad (3)$$

If we restrict to the condition that each row of $\mathbf{H}$ is all zeros but only one 1, the above problem is a combinatorial optimization. In practical implementation, we relax this condition to $\sum_{k=1}^{K} h_{lk} = 1$ for each $l = 1, 2, ..., L$ with nonnegative entries $h_{lk} \geq 0$. In this framework, the dimensionality reduction (PCA or SVD) and clustering are separated in two independent stages. Recently some research such as the adaptive dimensionality reduction techniques have integrated these two procedures into a single process, see [28], [29].

Denote the error term by

$$\mathcal{E}_l = \mathcal{X}_l - \left(\sum_{k=1}^{K} h_{lk} \mathcal{C}_k \right) \times_1 U^{(1)} \times_2 \cdots \times_N U^{(N)}.$$

Now we propose the following clustering problem: For a given set of N-order tensors $\mathcal{D} = \{\mathcal{X}_l\}_{l=1}^{L}$ of dimension $I_1 \times \cdots \times I_n \times \cdots \times I_N$, we seek K N-order tensorial centroids $\{\mathcal{C}_k\}_{k=1}^{K}$ of low dimension $J_1 \times \cdots \times J_n \times \cdots \times J_N$, a membership matrix $\mathbf{H}$ and a set of projection matrices $U^{(n)}$ $(n = 1, 2, ..., N)$ by solving the following optimization problem

$$\min_{H, \mathcal{C}_1, ..., \mathcal{C}_K, U^{(1)}, ..., U^{(N)}} \sum_{l=1}^{L} \phi\left(\mathcal{E}_l\right). \quad (4)$$

where $\phi(\cdot)$ is a loss function. The popular loss function in tensor decomposition is the squared tensorial Frobenius norm $\|\cdot\|_F^2$, see [5].

Now we stack K N-order tensorial centroids into an $(N+1)$-order tensor $\mathcal{C} = [\mathcal{C}_1; ...; \mathcal{C}_K] \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N \times K}$. According to the definition of $(N+1)$-mode product of a tensor with a vector, the above problem can be re-defined with

$$\mathcal{E}_l = \mathcal{X}_l - (\mathcal{C} \overline{\times}_{N+1} \mathbf{h}_l) \times_1 U^{(1)} \times_2 \cdots \times_N U^{(N)}. \quad (5)$$

where $\mathbf{h}_l$ is the l-th row vector of $\mathbf{H}$ with property $\mathbf{1}^T \mathbf{h}_l = 1$.

C. Regularization for Tensor Clustering

Problem defined by (4) and (5) is not well-posed. To see this, suppose that $U^{(1)}, ..., U^{(N)}$ are part of a solution, then for any orthonormal matrices $Q^{(1)}, ..., Q^{(N)}$ in appropriate dimensions, $Q^{(1)}U^{(1)}, ..., Q^{(1)}U^{(N)}$ are also part of the solution according to Proposition 3.12 in [5] when the loss function is the squared Frobenius norm. Hence an appropriate regularization is needed. Here we propose two regularized problems of (4) and (5):

PCA Tensor Clustering Model (PCA-TC): Let $\phi(\cdot) = \|\cdot\|_F^2$. We use the regularization used in higher-order tensorial PCA by specifying the conditions $U^{(n)T} U^{(n)} = \mathbf{I}_{J_n}$, i.e., $U^{(n)}$ consists of orthonormal columns. Then problem in (4) and (5) becomes

$$\min_{H, \mathcal{C}_1, ..., \mathcal{C}_K, U^{(1)T} U^{(1)} = \mathbf{I}_{J_1}, ..., U^{(N)T} U^{(N)} = \mathbf{I}_{J_N}} \sum_{l=1}^{L} \|\mathcal{E}_l\|_F^2. \quad (6)$$

In general, the orthonormality can be relaxed to orthogonality by only requiring $U^{(n)T} U^{(n)} = \mathbf{D}_{J_n}$ to be diagonal.

Non-negative Tensor Clustering Models (NNTC): Non-negative matrix factorization (NMF) [17] has proven a successful approach in many application areas [30]. This has been extended to the tensorial cases. Kim and Choi [31] proposed the non-negative Tucker decomposition for a single tensor while Chi and Kolda [32] focuses on non-negative CANDECOMP. Here we propose its clustering version with non-negativity. We assume that the given tensor data are also non-negative, i.e., $\mathcal{X}_l \geq 0$. There are two possible choices for the loss function in non-negative cases. We first present the model under the tensorial norm as the loss function,

$$\min_{H, \mathcal{C}_1, ..., \mathcal{C}_K, U^{(1)} \geq 0, ..., U^{(N)} \geq 0} \sum_{l=1}^{L} \|\mathcal{E}_l\|_F^2. \quad (7)$$

For the counting data $\mathcal{X}_l$, it is more appropriate to minimize the generalized Kullback-Leibler (KL) divergence. For the sake of notation simplicity, denote

$$\mathcal{M}_l = (\mathcal{C} \overline{\times}_{N+1} \mathbf{h}_l) \times_1 U^{(1)} \times_2 \cdots \times_N U^{(N)}$$

the clustering Tucker model. Then we propose a second non-negative tensor clustering model as

$$\min_{H, \mathcal{C}_1, ..., \mathcal{C}_K, U^{(1)} \geq 0, ..., U^{(N)} \geq 0} \sum_{l=1}^{L} \text{sum}(\mathcal{M}_l - \mathcal{X}_l \odot \log(\mathcal{M}_l)). \quad (8)$$

where sum operator means the sum of all the elements of a tensor and $\odot$ means the element-wise multiplication.

III. Algorithms

As there are a number of tensor and matrix variables in models (6) - (8), we solve these problems via an alternating approach in which updating one variable while holding all other variables fixed.

A. Algorithm for PCA Tensor Clustering Model

First, we focus on problem (6). We solve (6) by alternating matrix variables $U^{(n)}, \mathcal{C}_k, \mathbf{H}$. Denote the objective function by

$$F(U^{(1)}, ..., U^{(N)}, \mathcal{C}_1, ..., \mathcal{C}_K, \mathbf{H}) = \sum_{l=1}^{L} \|\mathcal{E}_l\|_F^2.$$

In order to simplify notation, let us denote

$$U_{\otimes}^{-n} = U^{(N)} \otimes \cdots \otimes U^{(n+1)} \otimes U^{(n-1)} \otimes \cdots \otimes U^{(1)}.$$

where $\otimes$ is the Kronecker operation of matrices. To optimize F with respect to matrix variable $U^{(n)}$ while holding all

others constants, from [6], we can take mode-n matricization of the tensor terms in F, i.e.,

$$F(U^{(n)}) = \sum_{l=1}^{L} \|X_{l(n)} - U^{(n)} \left(\mathcal{C}\overline{\times}_{N+1}\mathbf{h}_l\right)_{(n)} \left(U_{\otimes}^{-n}\right)^T \|_F^2$$

where the subscript (n) means mode-n matricization of a tensor. Denote

$$B^{l(n)} = \left(\mathcal{C}\overline{\times}_{N+1}\mathbf{h}_l\right)_{(n)} \left(U_{\otimes}^{-n}\right)^T,$$

then the solution $U^{(n)*}$ shall be given by

$$\min_{U^{(n)T}U^{(n)}=\mathbf{I}} F = \sum_{l=1}^{L} \|X_{l(n)} - U^{(n)}B^{l(n)}\|_F^2.$$

which is a generalized orthogonal Procrustes problem [33]. Hence the columns of the optimal solution $U^{(n)*}$ are the J_n (leading) eigenvectors of the matrix $\sum_{l=1}^{L} B^{l(n)}X_{l(n)}^T$.

Now we optimize F with respect to the k-th tensorial centroid $\mathcal{C}_k$. Denote the new tensor by $\mathcal{Y}_l^{(-k)} = \mathcal{X}_l - \left(\sum_{\mu\neq k} h_{l\mu}\mathcal{C}_\mu\right) \times_1 U^{(1)} \times_2 \cdots \times_N U^{(N)}$, we can have

$$F(\mathcal{C}_k) = \sum_{l=1}^{L} \|\mathcal{Y}_l^{(-k)} - h_{lk}\mathcal{C}_k \times_1 U^{(1)} \times_2 \cdots \times_N U^{(N)}\|_F^2.$$

To isolate the elements in tensor $\mathcal{C}_k$, according to Propositions 3.7 and 3.8 in [5], we can re-write the above objective in vectorial form,

$$F(\mathcal{C}_k) = \sum_{l=1}^{L} \|\text{vec}(\mathcal{Y}_{l(1)}^{(-k)}) - h_{lk}(U^{(N)}\otimes\cdots\otimes U^{(1)})\text{vec}(\mathcal{C}_{k(1)})\|_F^2.$$

That is, we unfold the tensor along mode-1, then vectorize the mode-1 matrices. This becomes a standard least square problem and its solution is

$$\text{vec}(\mathcal{C}_{k(1)}) = \frac{\sum_{l=1}^{L} h_{lk}(U^{(N)} \otimes \cdots \otimes U^{(1)})^T \text{vec}(\mathcal{Y}_{l(1)}^{(-k)})}{\sum_{l=1}^{L} h_{lk}^2},$$

where we have used the fact that $U^{(n)T}U^{(n)} = \mathbf{I}$.

It is easy to see that optimizing F with respect to $\mathbf{H}$ while holding others fixed is equivalent to L independent subproblems

$$\min_{\mathbf{h}_l\geq 0, \sum_{k=1}^{K} h_{lk}=1} \|\text{vec}(X_{(1)}) - (U^{(N)} \otimes \cdots \otimes U^{(1)})\mathbf{C}\mathbf{h}_l\|^2$$

where $\mathbf{C} = [\text{vec}(\mathcal{C}_{1(1)}), ..., \text{vec}(\mathcal{C}_{K(1)})]$. This is a non-negative least square problem which can be solved by, for example, Matlab function `lsqlin.m`.

B. Algorithm Non-negative Tensor Clustering Models

Here we consider model (7) and we will discuss algorithms for model (8) in a separate paper. The difference between (6) and (7) is the non-negativity constraints $U^{(n)} \geq 0$. Hence when we adopt an alternating method, we only need to change the subproblem for optimizing $U^{(n)}$ in the algorithm introduced in the last subsection. Specifically we consider the following problem

$$\min_{U^{(1)}\geq 0,\ldots,U^{(N)}\geq 0} F(U^{(1)}, ..., U^{(N)}) = \sum_{l=1}^{L} \|\mathcal{E}_l\|_F^2. \quad (9)$$

Now we consider optimizing (9) with respect to $U^{(n)}$ in turn under the alternative fashion by using an iterative procedure. Suppose we are at iteration k, then we update $U_k^{(n)}$ around its current estimate $U_{k-1}^{(n)}$ while fixing $U^1, ..., U^{(n-1)}$ to their new values $U_k^1, ..., U_k^{(n-1)}$ at iteration k and $U^{(n+1)}, ..., U^{(N)}$ to $U_{k-1}^{(n+1)}, ..., U_{k-1}^{(N)}$ at the last iteration $k-1$. That is, the sub-optimization problem is given by

$$\min_{U^{(n)}\geq 0} F^k = F(U_k^{(1)}, ..., U_k^{(n-1)}, U^{(n)}, U_{k-1}^{(n+1)}, ..., U_{k-1}^{(N)}).$$

Then we apply the linearized proximal gradient method to the above problem so that the new updated is defined as

$$U_k^{(n)} = \underset{U^{(n)}\geq 0}{\text{argmin}} \langle G_k^{(n)}, U^{(n)} - U_{k-1}^{(n)}\rangle + \frac{L_{k-1}^{(n)}}{2}\|U^{(n)} - U_{k-1}^{(n)}\|^2 \quad (10)$$

where $G_k^{(n)} = \frac{\partial F^k}{\partial U^{(n)}}(U_{k-1}^{(n)})$ is the block-partial gradient of F at $U_{k-1}^{(n)}$ and $L_{k-1}^{(n)} > 0$ is an appropriate constant, which can be chosen as the Lipschitz constant of the derivative $G_k^{(n)}$.

The derivative $G_k^{(n)}$ can be calculated in the following way. For the sake of notation simplicity, denote the new N-order tensors $\mathcal{C}^l = \mathcal{C}\overline{\times}_{N+1}\mathbf{h}_l$ with mode-n matrix $C_{(n)}^l$. Then using Proposition 4.3b and Section 4.5 of [5] and applying the chain rule give

$$\frac{\partial F}{\partial U^{(n)}} = \sum_{l=1}^{L} \left(X_{(n)}^l - U^{(n)}C_{(n)}^l U_{\otimes}^{-n^T}\right) U_{\otimes}^{-n} C_{(n)}^{lT}$$

Note that in the definition of $U_{\otimes}^{-n}$ we shall replace those $U^{(m)}$ with $U_k^{(m)}$ ($m \leq n-1$) and $U_{k-1}^{(m)}$ ($m \geq n+1$) respectively. Given the above derivative, in our experiments, we set

$$L_{k-1}^{(n)} = \|\sum_{l=1}^{L} C_{(n)}^l U_{\otimes}^{-n^T} U_{\otimes}^{-n} C_{(n)}^{lT}\|.$$

It is a common practice to use an linear extrapolation to speed up convergence. In our experiments, we adopt Nesterov's extrapolation [34]. We replace the current estimate $U_{k-1}^{(n)}$ in (10) by its linear combination of previous estimates defined as

$$\widehat{U}_k^{(n)} = U_{k-1}^{(n)} + \omega_n^{k-1}(U_{k-1}^{(n)} - U_{k-2}^{(n)})$$

where the weights are given by

$$\omega_n^{k-1} = \min\left\{\widehat{w}_{k-1}, \delta_w\sqrt{\frac{L_{k-2}^{(n)}}{L_{k-1}^{(n)}}}\right\}$$

where $\delta_w < 1$ is a preselected constant and $\widehat{w}_{k-1} = \frac{t_{k-1}-1}{t_k}$ with

$$t_0 = 1,\ t_k = \frac{1}{2}(1 + \sqrt{1 + 4t_{k-1}^2}).$$

The above algorithm is actually a generalization of the alternative algorithm for non-negative matrix factorization in [35].

IV. Simulation Study

In this section, we present a set of experimental results on some real world data sets with high dimensional spatial structures. The intention of these experiments is to demonstrate our new models' superiority over some state-of-the-art clustering methods in prediction accuracy.

A. Recent Algorithms and Evaluation Metrics

Data clustering is indeed a classification task of assigning samples to different groups. There are many different types of clustering algorithms such as spectral clustering approaches [36] and the recent popular matrix factorization approaches. As the Tucker decomposition is a kind of multidimensional generalization of matrix factorization, in this section, we will focus on the comparison between the proposed models with some recent matrix factorization clusterings. The following methods for clustering are compared

1) Multiplicative Method for Nonnegative Matrix Factorization (MM) [18]
2) Alternating Least Square (ALS) algorithm [37];
3) Bayesian Nonnegative Matrix Factorization (B-NFM) [38];
4) Projected Gradient Method for Nonnegative Matrix Factorization (PGM) [19];
5) Dual Regularized Co-Clustering (DRCC) [39];
6) Nonnegative tri-Factor tensor decomposition with applications (Tri-ONTD) [24].

To quantitatively and effectively evaluate the clustering results, we adopt two quantity metrics, the accuracy (AC) and the normalized mutual information (NMI) [40], in our experiments. 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, then the AC measure is defined by

$$AC = \frac{\sum_{i=1}^{n} \delta\left(\hat{L}(i), \text{Map}_{(\hat{L},L)}(i)\right)}{n}$$

where n is the number of samples in total and function $\delta(\mathbf{a}, \mathbf{b})$ is set to 1 if and only if $\mathbf{a} = \mathbf{b}$, and 0 otherwise. $\text{Map}(\cdot)$ is the best mapping function that permutes $\hat{L}$ to match L, which is usually implemented by the Kuhn-Munkres algorithm [41].

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

$$\text{NMI}(L, \hat{L}) = \frac{\text{MI}(L, \hat{L})}{\max\left(\text{H}\left(L\right), \text{H}\left(\hat{L}\right)\right)}$$

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

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

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

B. Dataset Description

1) CBCL face database: This face database[1] contains two classes of data: face and non-face. The size of each image is 19x19. The goal of clustering for this database is to cluster the images into two different classes: face and non-face.

2) Entended Yale B dataset: For this database[2], we simply use the cropped images and resize them to 32x32 pixels. This dataset now has 38 individuals and around 64 near frontal images under different illuminations per individual.

3) MNIST database: This handwritten digits database[3] has a training set of 60,000 examples, and a test set of 10,000 examples. The digits have been size-normalized and centered in a fixed-size image

4) ORL Database: The AT&T ORL database[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).

5) PIE database, CMU: This database[5] consists of 41,368 images of 68 people. Each subject was imaged under 13 different poses, 43 different illumination conditions, and with 4 different expressions. It is called CMU Pose, Illumination, and Expression (PIE) database. In this paper, we test the algorithms on Pose27 subdataset as described in [42].

C. Experimental Results

We ran nonnegative matrix factorization algorithms MM, ALS, B-NFM, PHM based on MATLAB NMF Toolbox provided by Kasper Winther Joergensen. Matlab code for Algorithm DRCC was provided by the authors of [39] and we implemented all the other algorithms. The tri-FTM algorithm is based on the description in [24].

[1] http://cbcl.mit.edu/projects/cbcl/software-datasets/FaceData2.html
[2] http://vision.ucsd.edu/~leekc/ExtYaleDatabase/ExtYaleB.html
[3] http://yann.lecun.com/exdb/mnist/
[4] http://www.uk.research.att.com/facedatabase.html.
[5] http://www.ri.cmu.edu/research_project_detail.html?project_id=418&menu_id=261

TABLE I

AC AND NMI RESULTS FOR THE TWO CLASSES CBCL FACE DATABASE WITH 1200 RANDOMLY CHOSEN DATA (600 FROM EACH CLASS)

Models:	MM	ALS	B-NFM	PGM	DRCC	Tri-ONTD	PCA-TC	NNTC
AC (%)	0.6888	0.6863	0.6738	0.6338	0.7850	0.5800	0.5013	**0.8450**
NMI	0.1056	0.1029	0.0897	0.0530	0.2545	0.0788	0.0106	**0.4396**

Experiment I: We first tested all the algorithms on the CBCL face database as this is the simplest case where there are only two clusters of images. We chose the datasizes from 200 to 1600 by 100 for each class and the experiment results indicate that when size = 600 almost all the algorithms achieved the best accurary. Under this size, we also tested different tensor core sizes from 4 to 10 and found both PCA Tensor Clustering Model (PCA-TC) and Non-negative Tensor Clustering model (NNTC) gave better results for the tensor core size 7. Table I compares all the results. We can see that at two class cases all the algorithms are comparable. Clearly the NNTC model is the winner while the performance of the PCA-TC model is a bit disappointing in this case, although PCA-TC had better results AC = 0.7358 and NMI = 0.1994 when the tensor core size is 8.

Experiment II: In the next experiment, we tested the impact of the number of classes on the performance of all the algorithms. We chose the NMIST handwritten digits dataset for this purpose. There are ten different classes. We randomly picked digits for training data according to the class number $K = 3, 4, ..., 10$ with 100 images for each class. We report the experimental results in Table II, however for most of cases Tri-ONTD failed, so no results reported. From the results we can see that in lower class number cases NNTC performs much better than all the others while in the case of higher class numbers PCA-TC takes over.

Experiment III: Then we did a similar test as Experiment II on PIE database to further confirm the observation in Experiment II. As the tri-ONTD algorithm provides meaningful results in our experiments, we abandoned further testing on the tri-ONTD algorithm. In this experiment, we randomly chose 600 images and tested on the cluster numbers 8, 18, 28, 38, 48, 58 and 68 where 68 is the maximal cluster number in the database. The results are collected in Table III. Obviously NNTC performs much better than all the others, except for several occasions where the classical ALS gave slightly better results.

Experiment IV: Finally we tested all the algorithms on the ORL database (40 subjects with 400 images) and YaleB extended database (38 subjects with 2414 images), respectively. For the ORL database we randomly chose 10 subjects with five runs while for the YaleB database we used all the images. The results are summarized in Table IV. We noted that almost all the algorithms failed over the YaleB database although NNTC manages to achieve the best result among them. Part of the reason is due to the larger database size and there are more variants among the images. For the ORL database, it seems NNTC is comparable to all the other classical algorithm while its NMI score is much better than others.

V. CONCLUSIONS

We have proposed two tensor clustering models based on tensor Tucker decomposition with different regularization. Two relevant algorithms have been proposed to solve the clustering problems based on alternative optimization strategy. We implemented the clustering algorithms based on different linearized proximal gradient method. We have also used a number of real world datasets to test the proposed algorithms and compared them with some existing nonnegative matrix factorization methods. Numerical results illustrate the high efficiency of the proposed algorithms. Based on the experiment results, we highly recommend the NNTC for tensor clustering.

Further work can be carried out in several different directions. For example, the general tensor clustering model (5) is an unsupervised learning task. It cannot be applied directly to the situation when some label information is available. We will further explore a semi-supervised extension of model (5) in which label information can be utilized. Consider a tensorial set of L N-order tensors $\{\mathcal{X}_l\}_{l=1}^{L}$, among which the label information is available for the first l_0 tensors $\mathcal{X}_1, ..., \mathcal{X}_{l_0}$, and the rest of $L - l_0$ tensors $\mathcal{X}_{l_0+1}, ..., \mathcal{X}_L$ are unlabelled. Given this information, we can design the membership matrix $\mathbf{H}$ as follows: Let $\mathbf{h}_l = \mathbf{e}_k$ if the l-th tensor is in class k ($l = 1, 2, ..., l_0$ and $k = 1, 2, ..., K$). Thus the membership matrix $\mathbf{H} = [\mathbf{H}_0; \mathbf{H}_1]$ with $\mathbf{H}_0$ having been determined. Under this framework, problems (6) to (8) can be easily extended to semi-supervised cases. For example, the Semi-supervised Non-negative Tensor Cluster Model related to (7) can be defined as

$$\min_{\mathbf{H}_1, \mathcal{C}_1, ..., \mathcal{C}_K, U^{(1)} > 0, ..., U^{(N)} > 0} \sum_{l=1}^{L} \|\mathcal{E}_l\|_F^2. \tag{11}$$

Due to the limitation of space, we leave this for another paper.

ACKNOWLEDGMENTS

The authors thank Dr. Quanquan Gu for sharing his DRCC code with us.

REFERENCES

[1] B. Schölkopf and A. Smola, *Learning with Kernels*. Cambridge, Massachusetts: The MIT Press, 2002.

[2] H. Lua, K. N. Plataniotisb, and A. N. Venetsanopoulos, "A survey of multilinear subspace learning for tensor data," *Pattern Recognition*, 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 Journal of Photogrammetry and Remote Sensing*, vol. 83, pp. 50–63, 2013.

TABLE II

AC AND NMI RESULTS FOR THE TEN CLASS NMIST HANDWRITTEN DIGITS WITH 1000 RANDOMLY CHOSEN IMAGES (100 FROM EACH CLASS)

	Models:	MM	ALS	B-NFM	PGM	DRCC	PCA-TC	NNTC
$K = 3$	AC (%)	0.6063	0.6067	0.6200	0.5967	0.5900	0.7067	**0.7900**
	NMI	0.3273	0.3466	0.3789	0.3164	0.3168	0.4426	**0.4576**
$K = 4$	AC (%)	0.7950	0.7975	0.7925	0.7900	0.7375	0.7900	**0.8425**
	NMI	0.5406	0.5549	0.5613	0.5361	0.4568	0.5484	**0.6006**
$K = 5$	AC (%)	0.6000	0.5540	0.6360	0.5360	0.5180	0.5820	**0.6440**
	NMI	0.4247	0.4324	0.4373	0.3733	0.3225	0.4365	**0.4508**
$K = 6$	AC (%)	0.6133	**0.6117**	0.5867	0.5933	0.4850	0.6100	0.6033
	NMI	0.4578	0.4623	0.4164	0.4532	0.4264	0.4452	**0.4919**
$K = 7$	AC (%)	0.5529	0.5486	0.4829	0.5300	0.6014	**0.6443**	0.5900
	NMI	0.4351	0.4835	0.4258	0.4479	0.4486	**0.5071**	0.4628
$K = 8$	AC (%)	0.5463	0.6350	0.5288	**0.6750**	0.6475	0.6200	0.5962
	NMI	0.4361	0.5010	0.4538	**0.5282**	0.4987	0.4644	0.4445
$K = 9$	AC (%)	0.4989	0.4822	0.4189	0.4644	0.5111	**0.5522**	0.4556
	NMI	0.4249	0.4197	0.3771	0.4272	**0.4324**	0.4219	0.3524
$K = 10$	AC (%)	0.4810	0.4750	0.4330	0.3910	0.5140	0.5150	**0.5210**
	NMI	0.4367	0.4170	0.4038	0.4243	0.3978	0.4438	**0.4481**

TABLE III

AC AND NMI RESULTS FOR MULTIPLE CLASS PIE FACE DATABASE WITH 600 RANDOMLY CHOSEN DATA

	Models:	MM	ALS	B-NFM	PGM	DRCC	PCA-TC	NNTC
$K = 8$	AC (%)	0.6575	0.7123	0.5753	0.6301	0.4384	0.6438	**0.8219**
	NMI	0.6590	0.7258	0.6402	0.5989	0.4206	0.5891	**0.8764**
$K = 18$	AC (%)	0.7200	0.6933	0.5267	0.6733	0.4000	0.5733	**0.8067**
	NMI	0.8250	0.7972	0.7186	0.7790	0.5253	0.7314	**0.8822**
$K = 28$	AC (%)	0.6104	0.5542	0.5783	0.5944	0.3052	0.4859	**0.7430**
	NMI	0.7819	0.7369	0.7655	0.7669	0.5335	0.6615	**0.8421**
$K = 38$	AC (%)	0.6170	0.5805	0.6444	0.5623	0.3100	0.4711	**0.6748**
	NMI	0.7821	0.7688	0.8066	0.7475	0.5536	0.6708	**0.8185**
$K = 48$	AC (%)	**0.6458**	0.5759	0.6289	0.5133	0.3108	0.4916	0.6361
	NMI	0.8079	0.7714	0.7983	0.7840	0.5687	0.7117	**0.8260**
$K = 58$	AC (%)	**0.6257**	0.5901	0.6000	0.6059	0.2733	0.4891	0.6198
	NMI	**0.8207**	0.7957	0.8089	0.7973	0.5616	0.7071	0.8132
$K = 68$	AC (%)	0.6033	0.5550	0.5900	0.5933	0.2483	0.4917	**0.6133**
	NMI	0.8064	0.7849	0.8073	0.8220	0.5444	0.7507	**0.8069**

TABLE IV

AC AND NMI RESULTS FOR BOTH ORL AND YALEB FACE DATABASES

	Models:	MM	ALS	B-NFM	PGM	DRCC	PCA-TC	NNTC
ORL	AC (%)	0.6440	**0.6820**	0.6280	0.6300	0.5580	0.6360	0.6680
	NMI	0.7308	0.7269	0.7107	0.7236	0.6498	0.6968	**0.7362**
YaleB	AC (%)	0.2175	0.2270	0.2104	0.2084	0.1002	0.2117	**0.2709**
	NMI	0.3534	0.3470	0.3508	0.3477	0.1455	0.3402	**0.3995**

[4] D. Cai, X. He, and J. Han, "Tensor space model for document analysis," in *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 2006, pp. 625 – 626.

[5] T. G. Kolda, "Multilinear operators for higher-order decompositions," Sandia National Laboratories, Tech. Rep., 2006.

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

[7] G. A. Shaw and H. hua K. Burke, "Spectral imaging for remote sensing," *Lincoln Laboratory Journal*, vol. 14, no. 1, pp. 3–28, 2003.

[8] C. Bishop, *Pattern Recognition and Machine Learning*, ser. Information Science and Statistics. Springer, 2006.

[9] G. Golub and C. van Loan, *Matrix Computations*, 3rd ed. Maryland: The Johns Hopkins University Press, 1996.

[10] S. Aja-Fernández, G. R. Luis, D. Tao, and X. Li, *Tensors in Image Processing and Computer Vision*. Springer, 2009.

[11] Y. Tang, R. Salakhutdinov, and G. Hinton, "Tensor analyzers," in *Proceedings of the 30th International Conference on Machine Learning*, Atlanta, USA, 2013.

[12] A. Shashua and T. Hazan, "Non-negative tensor factorization with applications to statistics and computer vision," in *Proceedings of International Conference on Machine Learning (ICML)*, 2005, pp. 792–799.

[13] M. Morup, "Applications of tensor (multiway array) factorizations and decompositions in data mining," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 24–40, 2011.

[14] H. Kiers, "Towards a standardized notation and terminology in multiway analysis," *Journal of Chemometrics*, vol. 14, no. 3, pp. 105–122, 2000.

[15] L. DeLathauwer, B. DeMoor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM Journal of Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2001.

[16] ——, "On the best rank-1 and rank-$(r_1, r_2, \cdots, r_n)$ approximation of higherorder tensors," *SIAM Journal of Matrix Analysis and Applications*, vol. 21, p. 13241342, 2000.

[17] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, p. 788791, 1999.

[18] ——, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems (NIPS)*, T. K. Leen, T. G. Diettercih, and V. Tresp, Eds., vol. 13. MIT Press, 2001, pp. 556–562.

[19] C. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural Computation*, vol. 19, no. 10, p. 27562779, 2007.

[20] M. Welling and M. Weber, "Positive tensor factorization," *Pattern Recognition Letters*, vol. 22, pp. 1255–1261, 2001.

[21] M. P. Friedlander and K. Hatz, "Computing non-negative tensor factorizations," *Optimization Methods and Software*, vol. 23, no. 4, pp. 631–647, 2008.

[22] J. Liu, J. Liu, P. Wonka, and J. Ye, "Sparse non-negative tensor factorization using columnwise coordinate descent," *Pattern Recognition*, vol. 45, pp. 649–656, 2012.

[23] F. Wu, X. Tan, Y. Yang, D. Tao, S. Tang, and Y. Zhuang, "Supervised nonnegative tensor factorization with maximum-margin constraint," in *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013, pp. 962–968.

[24] Z. Y. Zhang, T. Li, and C. Ding, "Non-negative tri-factor tensor decomposition with applications," *Knowledge Information Systems*, vol. 34, pp. 243–265, 2013.

[25] H. Liu, Z. Wu, X. Li, D. Cai, and T. S. Huang, "Constrained nonnegative matrix factorization for image representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 7, pp. 1299–1311, 2012.

[26] A. Adler, M. Elad, and Y. Hel-Or, "Probabilistic subspace clustering via sparse representations," *IEEE Signal Processing Letters*, vol. 20, no. 1, pp. 63–66, 2013.

[27] A. Kapteyn, H. Neudecker, and T. Wansbeek, "An approach to n-mode components analysis," *Psychometrika*, vol. 51, pp. 269–275, 1986.

[28] C. Ding and T. Li, "Adaptive dimension reduction using discriminant analysis and K-means clustering," in *Proceedings of International Conference on Machine Learning*, 2007, pp. 521–528.

[29] M. Vichi and H. A. L. Kiers, "Factorial K-means analysis for two-way data," *Computational Statistics & Data Analysis*, vol. 37, pp. 49–64, 2001.

[30] N. D. Ho, "Non-negative matrix factorization algorithms and applications," Ph.D. dissertation, Department of Applied Mathematics, Universite Catholique de Louvain, 2008.

[31] Y.-D. Kim and S. Choi, "Nonnegative Tucker decomposition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[32] E. C. Chi and T. G. Kolda, "On tensors, sparsity, and nonnegative factorizations," *SIAM Journal on Matrix Analysis and Applications*, vol. 33, no. 4, pp. 1272–1299, 2012.

[33] P. Schonemann, "A generalized solution of the orthogonal Procrustes problem," *Psychometrika*, vol. 31, pp. 1–10, 1966.

[34] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Kluwer Academic Publishers, 2003.

[35] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM J. of Imaging Sciences*, vol. 6, no. 3, pp. 1758–1789, 2013.

[36] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 888–905, 1997.

[37] 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.

[38] M. Schmidt, O. Winther, and L. Hansen, "Bayesian non-negative matrix factorization," in *Lecture Notes in Computer Science (LNCS): Proc. of International Conf on Independent Component Analysis and Signal Separation*, vol. 5541, 2009, pp. 540–547.

[39] Q. Gu and J. Zhou, "Co-clustering on manifolds," in *The 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, Paris, France, 2009.

[40] D. Cai, X. He, X. Wang, H. Bao, and J. Han, "Locality preserving nonnegative matrix factorization," in *Proc. of Int. Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[41] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Review*, vol. 43, no. 1, pp. 129–159, Jan. 2001.

[42] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation. , 33(8):, 2011." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1548–1560, 2011.