Tensor LRR and sparse coding-based subspace clustering

Article

Accepted Version

Tensor LRR and Sparse Coding based Subspace Clustering

Yifan Fu, Junbin Gao, David Tien, Zhouchen Lin and Xia Hong

***Abstract*—Subspace clustering groups a set of samples from a union of several linear subspaces into clusters, so that samples in the same cluster are drawn from the same linear subspace. In the majority of existing work on subspace clustering, clusters are built based on the samples' feature information, while sample correlations in their original spatial structure are simply ignored. Besides, original high-dimensional feature vector contains noisy/redundant information, and the time complexity grows exponentially with the number of dimensions. To address these issues, we propose a tensor low-rank representation (TLRR) and sparse coding (SC) based subspace clustering method (TLRRSC) by simultaneously considering the samples' feature information and spatial structures. TLRR seeks a lowest-rank representation over original spatial structures along all spatial directions. Sparse coding learns a dictionary along feature spaces, so that each sample can be represented by a few atoms of the learned dictionary. The affinity matrix used for spectral clustering is built from the joint similarities in both spatial and feature spaces. TLRRSC can well capture the global structure and inherent feature information of data, and provide a robust subspace segmentation from corrupted data. Experimental results on both synthetic and real-world datasets show that TLRRSC outperforms several established state-of-the-art methods.**

***Index Terms*—Tensor LRR, Subspace Clustering, Sparse Coding, Dictionary Learning**

I. Introduction

In recent years we have witnessed a huge growth of multi-dimensional data due to technical advances in sensing, networking, data storage, and communications technologies. This prompts the development of a low-dimensional representation that best fits a set of samples in a high-dimensional space. *Linear subspace learning* is a type of traditional dimensionality reduction technique that finds an optimal linear mapping to a lower dimensional space. For example, Principle Component Analysis (PCA) [40] is essentially based on the hypothesis that the data are drawn from a low-dimensional subspace. However, in practice, a data set is not often well described by a *single* subspace. Therefore, it is more reasonable to consider data residing on a union of multiple low-dimensional subspaces, with each subspace fitting a subgroup of data. The objective of subspace clustering is to assign data to their relevant subspace clusters based on, for example, assumed models. In the last decade, subspace clustering has been widely applied to many real-world applications, including motion segmentation [14], [20], social community identification [9], and image clustering [3]. A famous survey on *subspace clustering* [44] classifies most existing subspace clustering algorithms into three categories: statistical methods [19], algebraic methods [38], [49] and spectral clustering-based methods [14], [29].

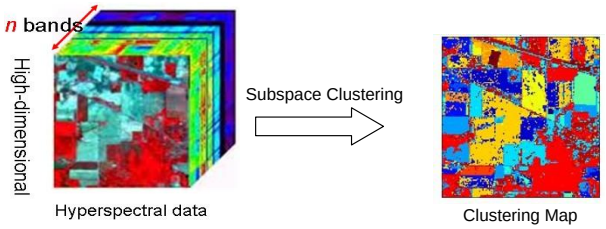


Fig. 1. Illustration of subspace clustering with high-dimensional data.

In existing traditional subspace clustering algorithms [44], one usually uses an "*unfolding*" process to re-arrange samples into a list of individual vectors, represented by a matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$, with each sample $\mathbf{x}_i$ $(1 \leq i \leq N)$ being denoted by a column vector. However, in many applications, samples may have multi-dimensional spatial structural forms, such as 2-dimension/mode hyperspectral images. In the 3-dimensional hyperspectral image case, one wishes to cluster all the pixels, each of which is represented as a spectrum vector consisting of many bands as shown in Fig. 1. As a result, the performance of traditional subspace clustering algorithms may be compromised in practical applications for two reasons: (1) they do not consider the inherent structure and correlations in the original data, and (2) building a model based on original high-dimensional features is not effective to filter the noisy/redundant information in the original feature spaces, and the time complexity grows exponentially with the number of dimensions.

For the first issue, *tensor* is a suitable representation for such multi-dimensional data like hyperspectral images, in a format of a multi-way array. The *order* of a tensor is the number of dimensions, also known as *ways* or *modes*. Thus, a set of hyperspectral images with a 2-dimension spatial structure can be denoted by an order-3 tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, with mode-$i$ $(1 \leq i \leq 2$) denoting the sample's position along its two spatial directions, and the mode-3 denoting the sample feature direction, e.g. a range of wavelengths in the spectral

Yifan Fu and David Tien are with School of Computing and Mathematics, Charles Sturt University, Bathurst, NSW 2795, Australia. E-mail: fuyf939@gmail.com, dtien@csu.edu.au

Junbin Gao is with Discipline of Business Analytics, The University of Sydney Business School, The University of Sydney, NSW 2006, Australia. E-mail: junbin.gao@sydney.edu.au

Zhouchen Lin is with Key Laboratory of Machine Perception (MOE), School of Electronics Engineering and Computer Science, Peking University, Beijing, China; Cooperative Medianet Innovation Center, Shanghai Jiaotong University, P. R. China. E-mail: zlin@pku.edu.cn. (Corresponding Author)

Xia Hong is with Department of Computer Science, School of Mathematical and Physical Sciences, University of Reading, Reading, RG6 6AY, UK. E-mail: x.hong@reading.ac.uk.

TABLE I
NOTATIONS USED IN THE PAPER

$\mathcal{X}$ and $\mathcal{E}$	an input order-N tensor and an error order-N tensor
$\mathbf{X}_{(n)}$ and $\mathbf{E}_{(n)}$	the mode-n matricization of tensors $\mathcal{X}$ and $\mathcal{E}$
S_k and $\mathcal{L}_k (k = 1, 2, ., , , K)$	the kth subspace and its corresponding orthogonal basis
$\mathbf{X}_k$ and Z_k	samples drawn from subspace S_k and corresponding low-dimensional representation
$\mathbf{S} = [S_1, S_2, \ldots, S_K]$	a set of multiple learnt subspaces
$\mathbf{Z} = [Z_1, Z_2, \ldots, Z_K]$	the low-dimensional representations of $\mathcal{X}$ with respect to $\mathbf{S}$
$\mathbf{U}_n \in \mathbb{R}^{I_n \times R_n} (1 \leq n \leq N)$	the factor matrices along mode-n
$\mathbf{D}$ and $\mathbf{A}$	a learned dictionary and a sparse representation on $\mathbf{X}_{(N)}$
r and R	the maximum number of non-zero elements in each instance and the matrix $\mathbf{A}$
$\Phi_{(I_1, \ldots, I_{N-1}, I_N)}$	a transformation of inverse matricization
$\mathbf{W}$	the structured coefficient matrix
λ, Y_n and $\mu_n > 0$	a balance parameter, the Lagrange multiplier and a penalty parameter, respectively
$\|\cdot\|_1$, $\|\cdot\|_0$, $\|\cdot\|_{2,1}$, $\|\cdot\|_*$ and $\|\cdot\|_F$	the l_1 norm, l_0 norm , $l_{2,1}$ norm, the nuclear and Frobenius norm

dimension. Fu. et al. [50] proposed a novel subspace clustering method called TLRR where the input data are represented in their original structural form as a tensor. It finds a lowest-rank representation for the input tensor, which can be further used to build an affinity matrix. The affinity matrix used for spectral clustering [54] records pairwise similarity along the row and column directions.

For the second issue, finding low-dimensional inherent feature spaces is a promising solution. Dictionary learning [51] is commonly used to seek the lowest rank [29], [53], [52] or sparse representation [14], [55] with respect to a given dictionary, which is often the data matrix $\mathbf{X}$ itself. Low-rank representation (LRR) and sparse representation/sparse coding (SC) take sparsity into account in different ways. The former defines a holistic sparsity on a whole data representation matrix, while the latter finds the sparsest representation of each data vector individually. SC has been widely used in numerous signal processing tasks, such as imaging denoising, texture synthesis and image classification [26], [48], [36]. Nevertheless, the performance of SC deteriorates when data are corrupted. Therefore, it is highly desirable to integrate spatial information into SC to improve clustering performance and reduce computational complexity as well.

Against this background, we propose a novel subspace clustering method where the input data are represented in their original structural form as a *tensor*. Our model finds a *lowest-rank representation* for each spatial mode of input *tensor*, and a *sparse representation* with respect to a learned *dictionary* in the feature mode. The combination of similarities in spatial and feature spaces is used to build an affinity matrix for spectral clustering. In summary, the contribution of our work is threefold:

- We propose a tensor low-rank representation to explore spatial correlations among samples. Unlike previous work which merely considers sample feature similarities and reorder original data into a matrix, our model takes sample spatial structure and correlations into account. Specifically, our method directly seeks a low-rank representation of samples' natural structural form — a high-order tensor.
- Our work integrates dictionary learning for sparse representation in the feature mode of tensor. This setting fits each individual sample with its sparsest representation with respect to the learned dictionary, consequently resolving exponential complexity and memory usage issues of some classical subspace clustering methods (e.g. statistical and algebraic based methods) effectively.
- The new subspace clustering algorithm based on our model is robust and capable of handling *noise* in the data. Since our model considers both feature and spatial similarities among samples, even if data are severely corrupted, it can still maintain a good performance since the spatial correlation information is utilized in order to cluster data into their respective subspaces correctly.

II. Related Work

The author of [44] classifies existing subspace clustering algorithms into three categories: statistical methods, algebraic methods, and spectral clustering-based methods.

Statistical models assume that mixed data are formed by a set of independent samples from a mixture of a certain distribution such as Gaussian. Each Gaussian distribution can be considered as a single subspace, then subspace clustering is transformed into a mixture of Gaussian model estimation problems. This estimation can be obtained by the Expectation Maximization (EM) algorithm in Mixture of Probabilistic PCA [41], or serial subspace searching in Random Sample Consensus (RANSAC) [17]. Unfortunately, these solutions are sensitive to noise and outliers. Some efforts have been made to improve algorithm robustness. For example, Agglomereative Lossy Compression (ALC) [31] finds the optimal segmentation that minimizes the overall coding length of the segmented data, subject to each subspace being modelled as a degenerate Gaussian. However, the optimization difficulty is still a bottleneck in solving this problem.

Generalized Principle Component Analysis (GPCA) [45] is an algebraic based method to estimate a mixture of linear subspaces from sample data. It factorizes a homogeneous polynomial whose degree is the number of subspaces and the factors (roots) represent normal vectors of each subspace. GPCA has no restriction on subspaces, and works well under certain conditions. Nevertheless, the performance of algebraic based methods in the presence of noise deteriorates as the number of subspaces increases. Robust Algebraic Segmentation (RAS) [38] is proposed to improve its robustness, but the complexity issue still exists. Iterative methods improve the performance of algebraic based algorithms to handle noisy

data in a repeated refinement. The *k*-subspace method [19], [6] extends the *k*-means clustering algorithm from data distributed around cluster centres to data drawn from subspaces of any dimensions. It alternates between assigning samples to subspaces and re-estimating subspaces. The *k*-subspace method can converge to a local optimum in a finite number of iterations. Nevertheless, the final solution depends on good initialization and is sensitive to outliers.

The works in [14], [29] and [38] are representative of spectral clustering-based methods. They aim to find a linear representation, $\mathbf{Z}$, for all the samples in terms of all other samples, which is solved by finding the optimal solution to the following objective function:

$$\begin{aligned} \min_{\mathbf{Z}} \|\mathbf{Z}\|_b + \frac{\lambda}{2}\|\mathbf{E}\|_q \\ \text{s.t. } \mathbf{X} = \mathbf{X}\mathbf{Z} + \mathbf{E} \end{aligned} \tag{1}$$

where $\|\cdot\|_q$ and $\|\cdot\|_b$ denote the norms for error and the new representation matrix $\mathbf{Z}$, respectively, λ is the parameter to balance the two terms. Using the resulting matrix $\mathbf{Z}$, an affinity matrix $|\mathbf{Z}|+|\mathbf{Z}^T|$ is built and used for spectral clustering. The Sparse Subspace Clustering (SSC) [14] uses the l_1 norm $\|\mathbf{Z}\|_1$ in favour of a sparse representation, with the expectation that within-cluster affinities are sparse (but not zero) and between-cluster affinities shrink to zero. However, this method is not designed to accurately capture the global structure of data and may not be robust to noise in data. The Low-Rank Representation (LRR) [29] employs the nuclear norm $\|\mathbf{Z}\|_*$ to guarantee a low-rank structure, and the $l_{2,1}$ norm is used in the error term to make it robust to outliers.

Dictionary learning for sparse representation aims at learning a dictionary $\mathbf{D}$ such that each sample in the dataset can be represented as a sparse linear combination of the atoms of $\mathbf{D}$. The problem of dictionary learning is formulated as

$$\begin{aligned} \min_{\mathbf{D},\mathbf{z}_i(i=1,2,\ldots,N)} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{D}\mathbf{z}_i\|_2^2 \\ \text{s.t. } \|\mathbf{z}_i\|_0 = r \end{aligned} \tag{2}$$

where $\|\cdot\|_2$ denotes the l_2 norm, $\|\mathbf{z}_i\|_0$ is the l_0-norm of the coefficient vector $\mathbf{z}_i$, which is defined as the number of non-zero elements, and r is a pre-defined sparsity integer for each sample. The optimization is carried out using an iterative algorithm that is formed by two alternative steps: (i) the sparse coding by fixing the dictionary $\mathbf{D}$ and (ii) the dictionary update with a fixed sparse representation.

With regard to sparse coding for a given dictionary, existing algorithms are divided into three categories: optimization methods, greedy methods and thresholding-based methods. Basis Pursuit (BP) is a commonly used optimization method, which uses a convex optimization method to minimize the l_1 norm $\|\mathbf{z}_i\|_1$ subject to the constraint $\mathbf{x}_i = \mathbf{D}\mathbf{z}_i$, if the vector $\mathbf{z}_i$ is sparse enough and the matrix $\mathbf{D}$ has sufficiently low coherence [12], [42]. The computational complexity of BP is very high, thus it is not suitable for large-scale problems. In comparison, the greedy algorithm Matching Pursuit (MP) has a significantly smaller complexity than BP, especially when the sparsity level is low [43]. A popular extension of MP is Orthogonal Matching Pursuit (OMP) [33], [34], which iteratively refines a sparse representation by successively identifying one component at a time that yields the greatest improvement in quality until an expected sparsity level is achieved or the approximation error is below the given threshold. The thresholding-based methods contains algorithms that do not require an estimation of the sparsity. In such algorihtms, the hard thresholding operator gives way to a soft thresholding operator with a positive threshold, such as the iterative hard thresholding algorithm (IHT) [5] and the hard thresholding pursuit (HTP)[18]. Another important method for sparse coding is the message-passing algorithm studied by Donoho, Maleki, and Montanari in [11].

The main differences in dictionary update algorithms are in the ways they update the dictionary. Sparsenet [35] and Method of Optimal Directions (MOD) [15] perform the dictionary update with fixed values of coefficients. Sparsenet updates each atom of dictionary iteratively with a projected fixed step gradient descent. MOD updates the whole dictionary in one step by finding a closed-form solution of an unconstrained least-square problem. Different from the above two algorithms, K-SVD [1] updates each dictionary atom and the values of its non-zero sparse coefficient simultaneously. The atom update problem then becomes a PCA problem. The K-SVD algorithm is flexible and can work with any pursuit methods.

III. Notations and Problem Formulation

A. Definition and Notations

Before formulating the subspace clustering problem, we first introduce some tensor fundamentals and notations. Please refer to [22] for more detailed definitions and notations.

Definition 1 (Tensor Matricization): Matricizaion is the operation of rearranging the entries of a tensor so that it can be represented as a matrix. Let $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ be a tensor of order-N, the mode-n matricization of $\mathcal{X}$ reorders the mode-n vectors into columns of the resulting matrix, denoted by $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times (I_{n+1} I_{n+2} \ldots I_N I_1 I_2 \ldots I_{n-1})}$.

Definition 2 (Kronecker Product [22]): The Kronecker product of matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{P \times L}$, denoted by $\mathbf{A} \otimes \mathbf{B}$, is a matrix of size $(IP) \times (JL)$ defined by

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \cdots & a_{IJ}\mathbf{B} \end{bmatrix} \tag{3}$$

Definition 3 (The n-mode Product): The n-mode product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ by a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$, denoted as $\mathcal{X} \times_n \mathbf{U}$, is a tensor with entries:

$$(\mathcal{X} \times_n \mathbf{U})_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_N} u_{j i_n} \tag{4}$$

The n-mode product is also denoted by each mode-n vector multiplied by the matrix $\mathbf{U}$. Thus, it can be expressed in terms of tensor matricization as well:

$$\mathcal{Y} = \mathcal{X} \times_n \mathbf{U} \qquad \Leftrightarrow \qquad \mathbf{Y}_{(n)} = \mathbf{U}\mathbf{X}_{(n)} \tag{5}$$

Definition 4 (Tucker Decomposition): Given an order-N tensor $\mathcal{X}$, its Tucker decomposition is an approximated tensor defined by,

$$\begin{aligned}\hat{\mathcal{X}} &\equiv [\![\mathcal{G}; \mathbf{U}_1, ..., \mathbf{U}_N]\!] = \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \ldots \times_N \mathbf{U}_N \\ &= \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \cdots \sum_{r_N=1}^{R_N} g_{r_1 r_2 \ldots r_N} \mathbf{u}^1_{r_1} \circ \mathbf{u}^2_{r_2} \ldots \circ \mathbf{u}^N_{r_N} \end{aligned} \quad (6)$$

where $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times \ldots R_N}$ is called a core tensor, $\mathbf{U}_n = [\mathbf{u}^n_1, \mathbf{u}^n_2, ..., \mathbf{u}^n_{R_n}] \in \mathbb{R}^{I_n \times R_n} (1 \leq n \leq N)$ are the factor matrices and the symbol $\circ$ represents the vector outer product.

For ease of presentation, key symbols used in this paper are listed in Table I.

B. Tensorial Datasets

Given an order-N tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, we consider a data set of all the I_N dimensional vectors/features along $\mathcal{X}$'s N-mode (also called N-mode fibres). The size of the data set is $(I_1 \times I_2 \times \cdots \times I_{N-1})$. Assume that these samples are drawn from a union of K independent subspaces $\{S_k\}_{k=1}^K$ of unknown dimensions, i.e., $\sum_{k=1}^K S_k = \bigoplus_{k=1}^K S_k$, where $\bigoplus$ is the direct sum. Our purpose is to cluster all the I_N-dimensional vectors from the tensor $\mathcal{X}$ into K subspaces by incorporating their relevant spatial and feature information in the tensor.

IV. Subspace Clustering via Tensor Low-Rank Representation and Sparse Coding

A. Tensor Low-Rank Representation on Spatial Modes

The new approach Low-Rank Representation (LRR) [29] is very successful in subspace clustering for even highly corrupted data, outliers or missing entries. Inspired by the idea used in LRR, we consider a model of low-rank representation for an input tensor $\mathcal{X}$ similar to problem (1). Specifically, we decompose the input tensor $\mathcal{X}$ into a Tucker decomposition in which the core tensor $\mathcal{G}$ is the input tensor itself along with a factor matrix $\mathbf{U}_n$ at each mode $n \leq N$. That is, the proposed data representation model is

$$\mathcal{X} = [\![\mathcal{X}; \mathbf{U}_1, \mathbf{U}_2, ..., \mathbf{U}_N]\!] + \mathcal{E}. \quad (7)$$

Here we are particularly interested in the case where $\mathbf{U}_N = \mathbf{I}_{I_N}$ (identity matrix of order I_N). If we define $\mathbf{Z} = \mathbf{U}_{N-1} \otimes \mathbf{U}_{N-2} \otimes \cdots \otimes \mathbf{U}_1$, then based on the above multiple linear model, we may interpret the entries of $\mathbf{Z}$ as the similarities between the pairs of all the vectors along the N-mode of the data tensor $\mathcal{X}$. These similarities are calculated based on the similarities along all the $N-1$ spatial modes through the factor matrices $\mathbf{U}_n$ $(n = 1, ..., N-1)$, each of which measures the similarity at the n-th spatial mode.

As in LRR, the model (7) uses the data to represent itself, therefore we can expect low-rank factor matrices $\mathbf{U}_n$. It is well known that it is very hard to solve an optimization problem with matrix rank constraints. A common practice is to relax the rank constraint by replacing it with the nuclear norm [32] as suggested by matrix completion methods [8], [21]. Thus, we finally formulate our model as follows,

$$\begin{aligned} &\min_{\mathbf{U}_1, ..., \mathbf{U}_{N-1}} \sum_{n=1}^{N-1} \|\mathbf{U}_n\|_* + \frac{\lambda}{2} \|\mathcal{E}\|_F^2 \\ &\text{s.t. } \mathcal{X} = [\![\mathcal{X}; \mathbf{U}_1, ..., \mathbf{U}_{N-1}, \mathbf{I}_{I_N}]\!] + \mathcal{E} \end{aligned} \quad (8)$$

where $\|\cdot\|_*$ denotes the *nuclear norm* of a matrix, defined as the sum of singular values of the matrix, $\|\cdot\|_F$ denotes the Frobenius norm of a tensor, i.e. the square root of the sum of the squares of all its entries, and $\lambda > 0$ is a parameter to balance the two terms, which can be tuned empirically. That is, TLRR seeks optimal low-rank solutions $\mathbf{U}_n (1 \leq n < N)$ of the structured data $\mathcal{X}$ using itself as a dictionary [50].

Remark 1: There is a clear link between LRR and TLRR in (8). If we consider the mode-N matricization in (8), we will see that it can be converted to an LRR model with $\mathbf{Z} = \mathbf{U}_{N-1} \otimes \mathbf{U}_{N-2} \otimes \cdots \otimes \mathbf{U}_1$. However, in the standard LRR, such an explicit Kronecker structure in $\mathbf{Z}$ has been ignored, so the number of unknown parameters in $\mathbf{Z}$ is $(I_1 \times I_2 \times \cdots I_{N-1})^2$. This will cause difficulty in LRR algorithm doing SVD. However, TLRR exploits the Kronecker structure with the number of unknown parameters reduced to $I_1^2 + I_2^2 + \cdots + I_{N-1}^2$. Our experiments demonstrate TLRR is much faster than LRR.

B. Dictionary Learning for Sparse Representation on Feature Mode

Dictionary learning for sparse representation has been proven to be very effective in machine learning, neuroscience, signal processing, and statistics [13], [37], [25]. Similar ideas have been proposed for subspace clustering. Sparse subspace clustering algorithm (SSC) [14] is an inspiring approach that uses data itself as a given dictionary, sparsely representing each sample as a linear combination of the rest of the data. However, such representation is computationally expensive for large-scale data. In addition, it is well known that such sparse coding techniques strongly rely on the internal coherence of the dictionary, and the performance depresses grossly as the number of cluster grows. In contrast, our sparse modeling framework exploits a sparse representation in the design of an optimization procedure dedicated to the problem of dictionary learning, with comparable memory consumption and a lower computational cost than SSC.

Based on the model in Eq. (8), we consider a dictionary learning model for sparse representation along the N-mode (feature mode) of $\mathcal{X}$. To be specific, we approximate the mode-N matricization of tensor $\mathbf{X}_{(N)}$ with a dictionary to be learnt $\mathbf{D} \in \mathbb{R}^{I_N \times m}$ and a sparse representation $\mathbf{A} \in \mathbb{R}^{m \times (I_1 \times I_2 \times \ldots \times I_{N-1})}$ on feature spaces over all the samples, so that the feature vectors of each sample can be represented by a few atoms of $\mathbf{D}$ (i.e. $||\mathbf{a}_i||_0 < r$ for $1 \leq i \leq I_1 \times I_2 \times \ldots \times I_{N-1}$). Thus, our sparse coding model in the feature direction has a similar formulation to problem (2)

$$\begin{aligned} &\min_{\mathbf{D}, \mathbf{A}} \|\mathbf{X}_{(N)} - \mathbf{D}\mathbf{A}\|_2^2 \\ &\text{s.t. } \|\mathbf{A}\|_0 = R \end{aligned} \quad (9)$$

where $R = r \times (I_1 \times I_2 \times \ldots \times I_{N-1})$ is maximum number of non-zero elements in the matrix $\mathbf{A}$. By solving problem (9), we can obtain an optimal dictionary on the feature space of the input tensor, and sparse factors for each sample with respect to the learnt dictionary.

C. Tensor Spatial Low Rank Representation and Feature Sparse Coding

By integrating the advantages of LRR and SC, we propose a spectral based subspace clustering method, which simultaneously considers sample feature information and spatial structures. More specifically, we first define a transformation of inverse matricization $\Phi_{(I_1,\ldots,I_{N-1},I_N)}$ which converts a matrix $\mathbf{M} \in \mathbb{R}^{I_N \times (I_1 I_2 \cdots I_{N-1})}$ back to an order-N tensor $\mathcal{M} \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_{N-1} \times I_N}$, i.e., $\Phi_{I_1,\ldots,I_{N-1},I_N}(M) = \mathcal{M}$, see [2].

Now we replace the fixed data tensor core in (8) with a tensor re-converted from a matrix $\mathbf{DA}$ by applying the inverse matricization operator Φ, i.e., the model is

$$\mathcal{X} = [\![\Phi_{(I_1,I_2,\ldots,I_{N-1},I_N)}(\mathbf{DA}); \mathbf{U}_1, \cdots, \mathbf{U}_{N-1}, \mathbf{I}_{I_N}]\!] + \mathcal{E}. \tag{10}$$

where $\mathbf{D} \in \mathbb{R}^{I_N \times m}$ is the feature mode dictionary to be learned and $\mathbf{A} \in \mathbb{R}^{m \times (I_1 I_2 \cdots I_{N-1})}$ is the sparse representation coefficient matrix. Finally, our proposed learning model can be formulated as follows,

$$\begin{aligned} \min_{\mathbf{U}_1,\ldots,\mathbf{U}_{N-1},\mathbf{D},\mathbf{A}} & \sum_{n=1}^{N-1} \|\mathbf{U}_n\|_* + \frac{\lambda}{2} \| \mathcal{E} \|_F^2 \\ \text{s.t. } & \mathcal{X} = [\![\Phi_{(I_1,I_2,\ldots,I_{N-1},I_N)}(\mathbf{DA}); \mathbf{U}_1, \cdots, \mathbf{U}_{N-1}, \mathbf{I}_{I_N}]\!] + \mathcal{E} \\ & \|\mathbf{A}\|_0 = R, \end{aligned} \tag{11}$$

where R is a given sparsity. Thus, our model (11) aims to find the lowest-rank representations along all the spatial modes, and learn a dictionary with its sparse representation over samples on the feature mode at the same time.

Unlike Tensor LRR model in [50] merely consider multiple space information, our model incorporates both spatial and feature information into consideration. The advantage of our model over the Tensor LRR model is illustrated in Figure 2. Taking the mode-N matricization of $\mathcal{X}$ as an example, Tensor LRR model uses $\mathbf{X}_{(N)}$ as a dictionary, the coefficient matrix $\mathbf{Z} = [\mathbf{U}_{(N-1)} \otimes \cdots \otimes \mathbf{U}_1]^T$. However, our model learn a dictionary $\mathbf{D}$ on the feature space, which makes

$$\mathbf{X}_{(N)} = \mathbf{DA}[\mathbf{U}_{(N-1)} \otimes \cdots \otimes \mathbf{U}_1]^T$$

as shown in Figure 2(b). Accordingly, the dictionary representation of data $\mathbf{X}_{(N)}$ is given by the structured coefficient matrix $\mathbf{Z} = \mathbf{A}[\mathbf{U}_{(N-1)} \otimes \cdots \otimes \mathbf{U}_1]^T$. Thus both spatial information encoded in $\mathbf{U}_n$'s and the feature relations encoded in $\mathbf{A}$ make a contribution to the final dictionary representation.

Remark 2: Problem (11) is ill-posed due to the scaling between variables $\mathbf{D}, \mathbf{A}$ and $\mathbf{U}$. To make a well-posed problem, we also require extra constraints, for example, the columns of D are unit vectors and the largest entry of $\mathbf{A}$ to be 1.

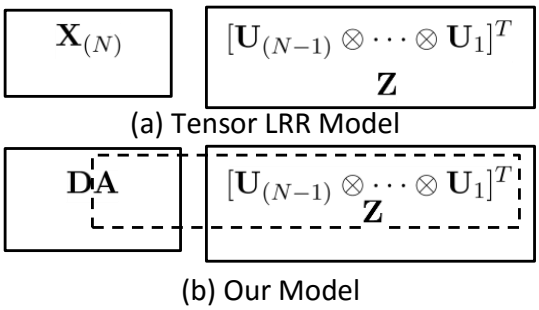


Fig. 2. the mode-N matricization of $\mathcal{X}$ decomposition for Tensor LRR model and our algorithm.

Remark 3: Using the Frobenius norm means we are dealing with Gaussian noises in the tensor data. If based on some domain knowledge, we know some noise patterns along a particular mode, for example, in multispectral imaging data, noises in some spectral bands are significant, we may adapt the so-called robust noise models like $l_{2,1}$-norm [10] instead.

Remark 4: As pointed out in Remark 1, TLRR improves the computational cost of the original LRR by introducing the Kronecker structure into the expression matrix. Although the new model looks more complicated than the traditional dictionary model, the computational complexity won't blow up due to the Kronecker structure. The cost added over the traditional dictionary learning is the overhead in handling low rank constraint over spatial modes. This is the price we have to pay for incorporating the spatial information of the data.

D. Solving the Optimization Problem

Optimizing (11) can be carried out using an iterative approach that solves the following two subproblems:

1) Solve Tensor LRR problem : Fix $\mathbf{D}$ and $\mathbf{A}$ to update $\mathbf{U}_n$, where $1 \le n < N$ by

$$\begin{aligned} \min_{\mathbf{U}_1,\ldots,\mathbf{U}_{N-1}} & \sum_{n=1}^{N-1} \| \mathbf{U}_n \|_* + \frac{\lambda}{2} \| \mathcal{E} \|_F^2 \\ \text{s.t. } & \mathcal{X} = [\![\Phi(\mathbf{DA}); \mathbf{U}_1, \cdots, \mathbf{U}_{N-1}, \mathbf{I}_{I_N}]\!] + \mathcal{E} \end{aligned} \tag{12}$$

2) Solve Dictionary learning for SC problem : Fix $\mathbf{U}_n (1 \le n < N)$ to update $\mathbf{D}$ and $\mathbf{A}$ by

$$\begin{aligned} & \min_{\mathbf{D},\mathbf{A}} \frac{\lambda}{2} \|\mathcal{X} - [\![\Phi(\mathbf{DA}); \mathbf{U}_1, \cdots, \mathbf{U}_{N-1}, \mathbf{I}_{I_N}]\!]\|_F^2 \\ & \text{s.t. } ||\mathbf{A}||_0 = R, \end{aligned} \tag{13}$$

We will employ the Block Coordinate Descent (BCD) [4] to solve the optimization problem (12) by fixing all the other mode variables to solve one variable at a time alternatively. For instance, TLRR fixes $\mathbf{U}_1, \ldots, \mathbf{U}_{n-1}, \mathbf{U}_{n+1}, \ldots, \mathbf{U}_{N-1}$ to minimize (12) with respect to the variable $\mathbf{U}_n (n = 1, 2, \ldots, N-1)$, which is equivalent to solve the following optimization subproblem:

$$\begin{aligned} \min_{\mathbf{U}_n} & \| \mathbf{U}_n \|_* + \frac{\lambda}{2} \| \mathcal{E} \|^2 \\ \text{s.t. } & \mathcal{X} = [\![\Phi(\mathbf{DA}); \mathbf{U}_1, \cdots, \mathbf{U}_{N-1}, \mathbf{I}_{I_N}]\!] + \mathcal{E} \end{aligned} \tag{14}$$

Using tensorial matricization, the problem (14) can be rewritten in terms of matrices as follows:

$$\begin{aligned} \min_{\mathbf{U}_n} \parallel \mathbf{U}_n \parallel_* + \frac{\lambda}{2} \parallel \mathbf{E}_{(n)} \parallel_F^2 \\ \text{s.t.} \quad \mathbf{X}_{(n)} = \mathbf{U}_n \mathbf{B}_{(n)} + \mathbf{E}_{(n)} \end{aligned} \tag{15}$$

where $\mathbf{B}_{(n)} = \Phi(\mathbf{DA})_{(n)}(\mathbf{I} \otimes \mathbf{U}_{N-1} \otimes \cdots \mathbf{U}_{n+1} \otimes \mathbf{U}_{n-1} \otimes \cdots \otimes \mathbf{U}_1)^T$.

Based on Eq.(15), each matrix $\mathbf{U}_n (1 \leq n < N)$ is optimized alternatively, while the other matrices are held fixed. All the matrices update iteratively until the change in fit drops below a threshold or when the number of iterations reaches a maximum, whichever comes first. The general process of BCD is illustrated by Algorithm 1.

Algorithm 1 Solving Problem (12) by BCD

Input: data tensor $\mathcal{X}$, dictionary $\mathbf{D}$ and $\mathbf{A}$ and the parameter λ
Output: factor matrices $\mathbf{U}_n$ $(n = 1, 2, \ldots, N-1)$

```
randomly initialize U_n ∈ R^{I_n × R_n} for n = 1, ..., N − 1
for n = 1, ..., N − 1 do
    X_(n) ← the mode-n matricization of the tensor X
    Φ(DA)_(n) ← the mode-n matricization the tensor Φ(DA)
end for
while reach maximum iterations or converge to stop do
    for n = 1, ..., N − 1 do
        B_(n) ← Φ(DA)_(n)(I⊗U_{N−1}⊗···U_{n+1}⊗U_{n−1}⊗···⊗U_1)^T
        U_n ← solve the subproblem (15)
    end for
end while
```

We use the Linearized Alternating Direction Method (LADM) [30] to solve the constrained optimization problem (15).

First of all, the augmented Lagrange function of (15) can be written as

$$\begin{aligned} L(\mathbf{E}_{(n)}, \mathbf{U}_n, \mathbf{Y}_n) = \parallel \mathbf{U}_n \parallel_* + \frac{\lambda}{2} \parallel \mathbf{E}_{(n)} \parallel_F^2 \\ + \text{tr}[\mathbf{Y}_n^T(\mathbf{X}_{(n)} - \mathbf{U}_n\mathbf{B}_{(n)} - \mathbf{E}_{(n)})] \\ + \frac{\mu_n}{2} \parallel \mathbf{X}_{(n)} - \mathbf{U}_n\mathbf{B}_{(n)} - \mathbf{E}_{(n)} \parallel_F^2 . \end{aligned} \tag{16}$$

where Y_n is the Lagrange multiplier and $\mu_n > 0$ is a penalty parameter.

Then the variables are updated by minimizing the augmented Lagrangian function L alternately, i.e., minimizing one variable at a time while the other variables are fixed. The Lagrange multiplier is updated according to the feasibility error. More specifically, the iterations of LADM go as follows

1) Fix all others to update $\mathbf{E}_{(n)}$ by

$$\min_{\mathbf{E}_{(n)}} \left\| \mathbf{E}_{(n)} - \left(\mathbf{X}_{(n)} - \mathbf{U}_n\mathbf{B}_{(n)} + \frac{\mathbf{Y}_n}{\mu_n} \right) \right\|_F^2 + \frac{\lambda}{\mu_n} \|\mathbf{E}_{(n)}\|_F^2 \tag{17}$$

which is equivalent to a least square problem. The solution is given by

$$\mathbf{E}_n = \frac{\lambda}{\lambda + \mu_n} \left(\mathbf{X}_{(n)} - \mathbf{U}_n\mathbf{B}_{(n)} + \frac{\mathbf{Y}_n}{\mu_n} \right) \tag{18}$$

2) Fix all others to update $\mathbf{U}_n$ by

$$\begin{aligned} \min_{\mathbf{U}_n} \|\mathbf{U}_n\|_* - \text{tr}(\mathbf{Y}_n^T\mathbf{U}_n\mathbf{B}_{(n)}) \\ + \frac{\mu_n}{2} \parallel (\mathbf{X}_{(n)} - \mathbf{E}_{(n)}) - \mathbf{U}_n\mathbf{B}_{(n)} \parallel_F^2 \end{aligned} \tag{19}$$

3) Fix all others to update $\mathbf{Y}_n$ by

$$\mathbf{Y}_n \leftarrow \mathbf{Y}_n + \mu_n(\mathbf{X}_{(n)} - \mathbf{U}_n\mathbf{B}_{(n)} - \mathbf{E}_{(n)}) \tag{20}$$

However, there is no closed-form solution to problem (19) because of the coefficient $\mathbf{B}_{(n)}$ in the third term. We propose to use the linearized approximation with an added proximal term to approximate the objective in (19) as described in [27]. Suppose that $\mathbf{U}_{(n)}^k$ is the current approximated solution to (19) and the sum of the last two terms is denoted by L, then the first order Taylor expansion at $\mathbf{U}_{(n)}^k$ plus a proximal term is given by

$$\begin{aligned} L \approx & \mu_n \langle (\mathbf{U}_n^k\mathbf{B}_{(n)} + \mathbf{E}_n - \mathbf{X}_{(n)} - \frac{\mathbf{Y}_n}{\mu_n})\mathbf{B}_{(n)}^T, \mathbf{U}_n - \mathbf{U}_n^k \rangle \\ & + \frac{\mu_n\eta_n}{2} \|\mathbf{U}_n - \mathbf{U}_n^k\|_F^2 + \text{consts} \end{aligned}$$

Thus, solving (19) can be converted to iteratively solve the following problem

$$\min_{\mathbf{U}_n} \|\mathbf{U}_n\|_* + \frac{\mu_n\eta_n}{2} \|\mathbf{U}_n - \mathbf{U}_n^k + \mathbf{P}_n\|_F^2$$

where $\mathbf{P}_n = \frac{1}{\eta_n}(\mathbf{U}_n^k\mathbf{B}_{(n)} + \mathbf{E}_n - \mathbf{X}_{(n)} - \frac{\mathbf{Y}_n}{\mu_n})\mathbf{B}_{(n)}^T$. The above problem can be solved by applying the SVD thresholding operator to $\mathbf{M}_n = \mathbf{U}_n^k - \mathbf{P}_n$. Take SVD for $\mathbf{M}_n = \mathbf{W}_n\Sigma_n\mathbf{V}_n^T$, then the new iteration is given by

$$\mathbf{U}_n^{k+1} = \mathbf{W}_n \text{soft}(\Sigma_n, \eta_n\mu_n)\mathbf{V}_n^T \tag{21}$$

where $\text{soft}(\Sigma, \sigma) = \max\{0, (\Sigma)_{ii} - \frac{1}{\sigma}\}$ is the soft thresholding operator for a diagonal matrix, see [7].

Algorithm 2 Solving Problem (15) by LADM

Input: matrices $\mathbf{X}_{(n)}$and $\mathbf{B}_{(n)}$, parameter λ
Output: : factor matrices $\mathbf{U}_n$

```
initialize: U_n = 0, E_(n) = 0, Y_n = 0, μ_n = 10^{−6}, max_u = 10^{10}, ρ = 1.1, ε = 10^{−8} and η_n = ‖B_(n)‖^2.
while ‖ X_(n) − U_n B_(n) − E_(n) ‖_∞ ≥ ε do
    E_(n) ← the solution (18) to the subproblem (17);
    U_n ← the iterative solution by (21) by for example five iterations;
    Y_n ← Y_n + μ_n(X_(n) − U_n B_(n) − E_(n))
    μ_n ← min(ρμ_n, max_u)
end while
```

Now we consider solving Dictionary learning for SC problem (13). Using tensorial matricization, the problem (13) can be equivalently written in terms of matrices as follows:

$$\begin{aligned} \min_{\mathbf{D},\mathbf{A}} \frac{\lambda}{2} \|\mathbf{E}_{(N)}\|_F^2 \\ \text{s.t. } \mathbf{X}_{(N)} = \mathbf{DAC}^T + \mathbf{E}_{(N)}, \\ ||\mathbf{A}||_0 = R, \end{aligned} \tag{22}$$

where $\mathbf{C} = (\mathbf{U}_{N-1} \otimes \cdots \otimes \mathbf{U}_1)$. The above problem (22) can be solved by using a two-phase BCD approach. In the first phase, we optimize $\mathbf{A}$ by fixing $\mathbf{D}$; in the second phase, we

update $\mathbf{D}$ by fixing $\mathbf{A}$. The process repeats until some stop criterion is satisfied.

When the dictionary $\mathbf{D}$ is given, the sparse representation $\mathbf{A}$ can be obtained by solving (22) with fixed $\mathbf{D}$.

The resulting problem becomes a 2D sparse coding problem, which can be solved by the 2D-OMP [16].

Remark 5: 2D-OMP is in fact equivalent to 1D-OMP, with exactly the same results. However, the memory usage of 2D-OMP is much lower than 1D-OMP. Note that 2D-OMP only need the memory usage of size $I_N \times m + (I_1 \times I_2 \ldots \times I_{(N-1)})^2$. However, the 1D-OMP need $(I_1 \times I_2 \ldots \times I_{(N)}) \times (m \times I_1 \times I_2 \ldots \times I_{(N-1)})$.

Given the sparse coefficient matrix $\mathbf{A}$, we define $\mathbf{F} = \mathbf{A}\mathbf{C}^T$, then the dictionary $\mathbf{D}$ can be updated by

$$\begin{aligned} &\min_{\mathbf{D}} \frac{\lambda}{2} \|\mathbf{E}_{(N)}\|_F^2 \\ &\text{s.t. } \mathbf{X}_{(N)} = \mathbf{D}\mathbf{F} + \mathbf{E}_{(N)}, \end{aligned} \tag{23}$$

Actually, (23) is a least squares problem. As it is large scale, a direct closed-form solution will cost too much overhead. Here we propose an iterative way alternatively on the columns of $\mathbf{D}$ based on the spare structures in $\mathbf{F}$. Let us consider only one column $\mathbf{d}_j$ in the dictionary and its corresponding coefficients, the j-th row in $\mathbf{F}$, denoted as $\mathbf{f}^j$. Eq. (23) can be rewritten as:

$$\begin{aligned} \|\mathbf{E}_{(N)}\|_F^2 &= \|\mathbf{X}_{(N)} - \sum_{j=1}^{m} \mathbf{d}_j \mathbf{f}^j\|_F^2 \\ &= \|(\mathbf{X}_{(N)} - \sum_{j \neq l} \mathbf{d}_j \mathbf{f}^j) - \mathbf{d}_l \mathbf{f}^l\|_F^2 \\ &= \|\mathbf{E}_{(N)}^l - \mathbf{d}_l \mathbf{f}^l\|_F^2 \end{aligned} \tag{24}$$

We have decomposed the multiplication $\mathbf{DF}$ into the sum of m rank-1 matrices, where m is the number of atoms in $\mathbf{D}$. The matrix $\mathbf{E}_{(N)}^l$ represents the error for all the m examples when the l-th atom is removed. Indeed, we are using K-SVD strategy [1] to update each atom $\mathbf{d}_l$ and $\mathbf{f}^l$ $(1 \leq l \leq m)$ by fixing all the other terms. However, the sparsity constraint is enforced in such an update strategy.

The general process of dictionary learning for SC is listed in Algorithm 3

Algorithm 3 Solving problem (13) by BCD

Input: matrices: $\mathbf{X}_{(N)}$ and $\mathbf{C}$
Output: dictionary $\mathbf{D}$ and sparse representation matrix $\mathbf{A}$

1: initialize the dictionary $\mathbf{D}$ with a random strategy.
2: **while** reach maximum iterations **do**
3: sparse representation $\mathbf{A} \leftarrow$ solve the problem (22) with fixed $\mathbf{D}$;
4: dictionary $\mathbf{D} \leftarrow$ solve the problem (23) with K-SVD strategy;
5: **end while**

E. The Complete Subspace Clustering Algorithm

After iteratively solving two subproblems (12) and (13), we finally obtain the low-rank and sparse representations given by $\mathbf{U}_i (i = 1, 2, \ldots, N-1))$ and $\mathbf{A}$ for the data $\mathcal{X}$. We create a similarity matrix on the spatial spaces $\mathbf{Z}^s = \mathbf{U}_{N-1} \otimes \mathbf{U}_{N-2} \otimes \cdots \otimes U_1$. The affinity matrix is then defined by $|\mathbf{Z}^s| + |\mathbf{Z}^{sT}| + |\mathbf{A}^T\mathbf{A}|$[①]. Each element of the affinity matrix is the joint similarity between a pair of mode-N vectorial samples across all the $N-1$ spatial modes/directions and the N-th feature mode. Finally, we employ the Normalized Cuts clustering method [39] to divide the samples into their respective subspaces. Algorithm 4 outlines the whole subspace clustering method of TLRRSC.

Algorithm 4 Subspace Clustering by TLRRSC

Input: structured data: tensor $\mathcal{X}$, number of subspaces K
Output: : the cluster indicator vector $\mathbf{l}$ with terms of all samples

1: **while** reach maximum iterations or converge to stop **do**
2: lowest-rank representation $\mathbf{U}_n (n = 1, 2, \ldots, N-1) \leftarrow$ solve the problem (12)
3: sparse representation $\mathbf{A}$ and the dictionary $\mathbf{D}$ $\leftarrow$solve the problem (13)
4: **end while**
5: $\mathbf{Z}^s \leftarrow \mathbf{U}_{N-1} \otimes \mathbf{U}_{N-2} \otimes \cdots \otimes \mathbf{U}_1$
6: $\mathbf{l} \leftarrow$ Normalized Cuts$(|\mathbf{Z}^s| + |\mathbf{Z}^{sT}| + |\mathbf{A}^T\mathbf{A}|)$

F. Computational Complexity

The TLRRSC algorithm composes of two iterative updating parameters steps followed by an normalized cut on an affinity matrix. Assuming the iteration times is t, $I_N = N$, low rank value is r and $I_n = d$,$(1 \leq n \leq N-1)$.

In the process of updating lowest-rank representation $\mathbf{U}_n (n = 1, 2, \ldots, N-1)$, the complexity of computing $\mathbf{DA}$ is O($N^2 d^{N-1}$), the computational costs regarding updating $\mathbf{B}_n$, $\mathbf{U}_n$ and $\mathbf{Y_n}$ to solve Problem (15) are O($N^2 d^{N-1}$), O(Nrd^2) and O($N^2 d^{N-1}$). Accordingly, the computational complexity of $\mathbf{U}_n (n = 1, 2, \ldots, N-1)$ is approximately O($N^2 d^{N-1}$)+ O(Nrd^2).

In the dictionary learning process, the costs of updating $\mathbf{A}$ and $\mathbf{D}$ are O(md^{N-1}) and O($N(k^2 m + 2Nd^{N-1})$) respectively, where k is the sparsity value in the KSVD algorithm.

After obtaining the final optimal $\mathbf{U}_n (n = 1, 2, \ldots, N-1)$, $\mathbf{A}$ and $\mathbf{D}$, the time complexity of creating an affinity matrix is O(d^{N-1}). With the affinity matrix, the normalized cut can be solved with a complexity of O($NlogN + d^{2(N-1)}$).

With above analysis, the total complexity of TLRRSC is

$$\begin{aligned} &O(N^2 d^{N-1}) + O(Nrd^2) + O(md^{N-1}) + \\ &O(N(k^2 m + 2Nd^{N-1})) + O(d^{N-1}) + O(NlogN + d^{2(N-1)}) \end{aligned} \tag{25}$$

As $k, r \ll d$, therefore the approximate complexity is O($(N^2 + m)d^{N-1}$) +O($NlogN + d^{2(N-1)}$).

V. Experimental Results

In this section, we present a set of experimental results on some synthetic and real data sets with multi-dimensional spatial structures. The intention of these experiments is to demonstrate our new method TLRRSC's superiority over

[①]To maintain scaling, we may use $|(A^T A)^{\frac{1}{2}}|$, but the experiments show that the simple definition $|\mathbf{Z}^s| + |\mathbf{Z}^{sT}| + |\mathbf{A}^T\mathbf{A}|$ works well. Other possible choices are $|Z^T Z| + |A^T A|$ and $(|Z| + |Z^T|) \odot |A^T A|$.

the state-of-art subspace clustering methods in prediction accuracy, computation complexity, memory usage, and noise robustness. To analyze the clustering performance, the Hungarian algorithm [23] is applied to measure the accuracy by comparing the predicted clustering results with the ground truth .

A. Baseline Methods

Because our proposed method is closely related to LRR and SSC, we choose LRR, TLRR and SSC methods as the baselines. Moreover, some previous subspace clustering methods are also considered.

1) LRR: The LRR methods have been successfully applied to subspace clustering for even highly corrupted data, outliers or missing entries. In this paper, we consider an LRR method introduced in [29], which is based on minimizing

$$\begin{aligned} \min_{\mathbf{Z}} \|\mathbf{Z}\|_* + \frac{\lambda}{2}\|\mathbf{E}\|_{2,1} \\ \text{s.t. } \mathbf{X} = \mathbf{X}\mathbf{Z} + \mathbf{E} \end{aligned} \tag{26}$$

However, this method conducts subspace clustering on a rearranged matrix, ignoring data spatial correlations. Thus, the entries of affinity matrix $|\mathbf{Z}|+|\mathbf{Z}^T|$ denote the pairwise similarity in the low-dimensional feature spaces.

2) TLRR: As an improvement over LRR, TLRR finds a low-rank representation for an input tensor by exploring factors along each spatial dimension/mode, which aims to solve the problem (8). An affinity matrix built for spectral clustering records the pairwise similarity along all the spatial modes.

3) SSC: SSC has a similar formulation to LRR, except for the employment of the l_1 norm $\|\mathbf{Z}\|_1$ in favour of a sparse representation. For fair comparisons, we implement two versions of SSC, i.e., SSC_1 is a l_1-norm version ($q = 2$ and $b = 1$ in (1)) and $SSC_{2,1}$ is a $l_{2,1}$-norm version ($q = 2, 1$ and $b = 1$ in (1)). SSC denotes SSC_1 if not specified in the following experiments.

4) Some Other Methods: We also consider for comparison some previous subspace clustering methods, including GPCA [45], Local Subspace Analysis (LSA) [47], and RANSAC [17].

In the following experiments, the parameter setting is as follows: a balance parameter $\lambda = 0.1$, a penalty parameter $\mu_n = 10^{-6}$, the convergence threshold $\varepsilon = 10^{-8}$.

B. Results on Synthetic Datasets

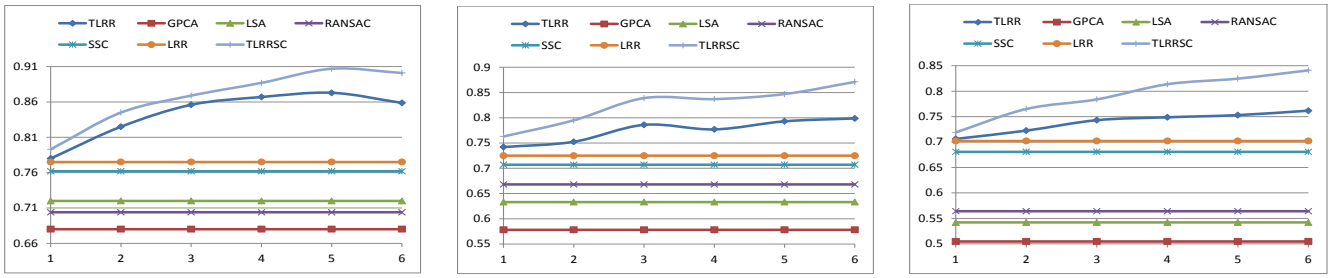
In this section, we evaluate TLRRSC against state-of-the-art subspace clustering methods on synthetic datasets. We use 3 synthetic data sets containing 3 subspaces, each of which is formed by N_k samples of d dimension feature respectively, where $d \in \{5, 10, 20\}$, $k \in \{1, 2, 3\}, N_1 = 30$, $N_2 = 24$, and $N_3 = 10$. The generation process is as follows: 1) Select 3 cluster centre points $c_i \in \mathbb{R}^d$ for above subspaces respectively, which are far from each other. 2) Generate a matrix $\mathbf{C}^k \in \mathbb{R}^{d\times N_k}$, each column of which is drawn from a Gaussian distribution $\mathcal{N}(\cdot|c_k, \Sigma^k)$, where $\Sigma^k \in \mathbb{R}^{d\times d}$ is a diagonal matrix such that the k-th element is 0.01 and others 1s. This setting guarantees each cluster lies roughly in a $d-1$ dimension subspace. 3) Combine samples in each subspace to form an entire data set $\mathbf{X} = \cup\mathbf{C}^k$.

1) Performance with High Order Tensorial Data: To show the TLRRSC's advantage of handling high order tensorial data over other baseline methods, we create 5 other synthetic datasets from the above data $\mathbf{X}$ by reshaping it into a higher j-mode tensor ($3 \le j \le 7$). Since all other baseline methods except TLRR conduct subspace clustering on an input matrix, i.e. a 2-mode tensor, we use $\mathbf{X}$ on all these baseline methods for the purpose of fair comparisons. Fig. 3 reports the results on all the baseline methods with different dimensions of feature spaces.

As we can see, TLRRSC and TLRR perform much better than other methods in the higher mode of tensor. This observation suggests that incorporating data structure information into subspace clustering can boost clustering performance, while the performance of other methods always stays still because these methods treat each sample independently, ignoring inherent data spatial structure information. TLRRSC is always superior to TLRR, which demonstrates that incorporating feature similarity can further boost clustering performance. Another interesting observation is that the performance gap between TLRR and TLRRSC is enlarged with growth of feature dimensions, which suggests that seeking the inherent sparse representation in the high dimensional feature space does help improve the clustering performance by filtering redundant information.

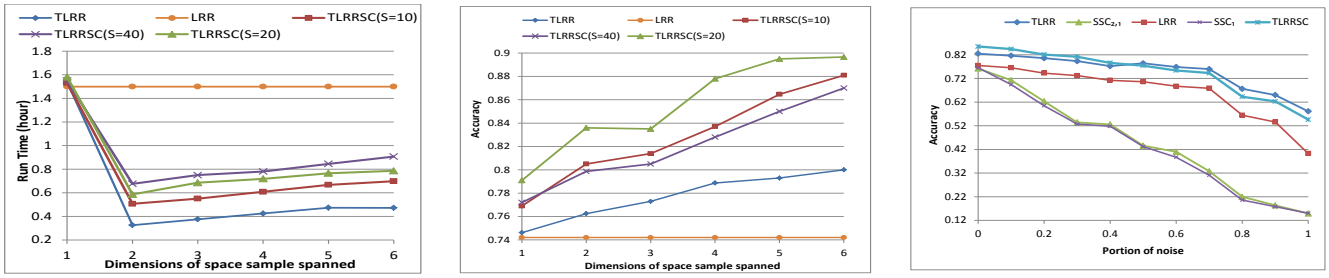
To compare the accuracy and running time among LRR based algorithms, we create a matrix $\widetilde{\mathbf{X}} \in \mathbb{R}^{200\times 640}$ containing 3 subspaces, each of which is formed by containing 3 subspaces, each of which is formed by N_k samples of 200 dimension features, where $k \in \{1, 2, 3\}, N_1 = 300$, $N_2 = 240$, and $N_3 = 100$. The generation process is similar to the construction of $\mathbf{X}$. Then we we create 5 other synthetic datasets from the new data matrix $\widetilde{\mathbf{X}}$ by reshaping it into a higher j-mode tensor ($3 \le j \le 7$). Fig. 4(a) and Fig. 4(b) compare the accuracy and running time among LRR based algorithms on the data set with 200-dimensional feature spaces. To investigate proposed algorithm's performance with different sparsity values S used in the dictionary learning along the feature direction, we use three sparsity values $S \in \{10, 20, 40\}$. We observe that as the order of tensor increases, the running time of TLRR and TLRRSC are significantly reduced compared with LRR (as shown in Fig. 4(a)), and the clustering accuracy of TLRR and TLRRSC is superior to its vectorized counterpart LRR (as shown in Fig. 4(b)). These observations suggest that the structural information has an important impact on speeding up the subspace clustering process and improving clustering accuracy.

As TLRRSC needs extra time to solve the sparse representation along the feature mode, the time cost of TLRRSC is a little more expensive than TLRR. Moreover, when the sparsity value is 20, TLRRSC performs best compared to other sparsity values, which suggests that our method can accurately cluster data with a small sparsity value. To sum up, our new method TLRRSC can achieve better performance with a comparable time cost in the higher mode of tensor.



(a) 5-dimension feature (b) 10-dimension feature (c) 20-dimension feature

Fig. 3. Accuracy comparisons *w.r.t.* different orders of a tensor.(a) high order tensorial data with 5-dimensional feature spaces. (b) high order tensorial data with 10-dimensional feature spaces. (c) high order tensorial data with 20-dimensional feature spaces.



(a) (b) (c)

Fig. 4. Time and Accuracy Comparison on the synthetic datasets with 20-dimension feature spaces. (a) Run time comparisons *w.r.t.* different orders of a tensor. (b) Accuracy comparisons *w.r.t.* different orders of a tensor. (c) Accuracy comparisons *w.r.t.* different potions of noisy samples.

2) Performance with Different Portions of Noisy Samples: Consider the cases where there exist noisy samples in the data. We randomly choose 0%, 10%,..., 100% of the samples of the above $\mathbf{C}^k$ respectively, and add Gaussian noises $\mathcal{N}(\cdot|c_k, 0.3\Sigma^k)$ to these samples. Then a noisy data set $\mathbf{X}'$ is generated by combining the corrupted $\mathbf{C}^k$ to one. The performances on $SSC_{2,1}, SSC_1$, LRR, TLRR and TLRRSC are listed in Fig. 4(c). Obviously, low-rank representation based subspace clustering methods TLRRSC, TLRR and LRR maintain their accuracies even though 70% of samples are corrupted by noise. Moreover, three LRR based methods significantly outperform both $SSC_{2,1}$ and SSC_1, as shown in Fig. 4(c), which suggests that low-rank representation is good at handling noisy data, while SSC is not because it solves the columns of the representation matrix independently. For low-rank based methods, LRR method is inferior to the structure based TLRR and TLRRSC. This is mainly because TLRR and TLRRSC integrate data spatial information into subspace clustering, resulting in a good performance even when 90% of data are corrupted. Another interesting result is that TLRRSC is marginally superior to TLRR when the noise rate is less than 50%, but its performance becomes inferior to TLRR as the noise rate continually increases to 100%. This again proves that sparse coding is sensitive to noise. Although TLRRSC maintains a good performance by exploring the spatial correlations among samples, sparse representation along the feature spaces induces more noises as the noise portion increase. Therefore, the clustering performance depresses with noisy feature similarities integrated in the affinity matrix.

3) Performance with Dictionary Learning for Sparse Coding: Like LRR and SSC, our model TLRRSC considers sparsity regarding low-dimensional representation on the feature space. In contrast to LRR and SSC, using the input data as a dictionary, TLRRSC learns a dictionary and its corresponding sparse representation. In this section, we compare the performances of different sparse strategies.

First of all, we create a matrix $\widehat{\mathbf{X}} \in \mathbb{R}^{30\times 64}$ containing 3 subspaces, each of which is formed by N_k samples of 30 dimensions, where $k \in \{1, 2, 3\}, N_1 = 30$, $N_2 = 24$, and $N_3 = 10$. The generation process is similar to the construction of $\mathbf{X}$, except each cluster centre points $c_k \in \mathbb{R}^{30}$ and the last 20 diagonal elements in $\Sigma^k \in \mathbb{R}^{30\times 30}$ is 0.01 and others are 1s. This setting guarantees that each cluster lies roughly in a 10 dimension subspace. Fig. 5 illustrates the evaluated mean of each band on the reconstructed data matrix denoted by the product of a dictionary and its sparse representation. Obviously, the evaluated mean of our model TLRRSC is the closest to the true value, compared to LRR and SSC. This suggests that TLRRSC finds a better dictionary to fit the data, instead of a fixed dictionary $\widehat{\mathbf{X}}$ in the other two methods. Moreover, Fig. 6 depicts the sparse representations obtained for $\widehat{\mathbf{X}}$. In our algorithm, we learn a dictionary $\mathbf{D}^{30\times 200}$ in the feature space with 200 atoms, while the other two models use given data as a dictionary, the corresponding sparse representation under the dictionary for baselines are illustrated in the black blocks of Fig. 6. Each line in the white block statistics of the total number of each atom used in the new sparse representation of the given dataset (i.e.the relative magnitude). For LRR algorithm, each atom in the dictionary is activated

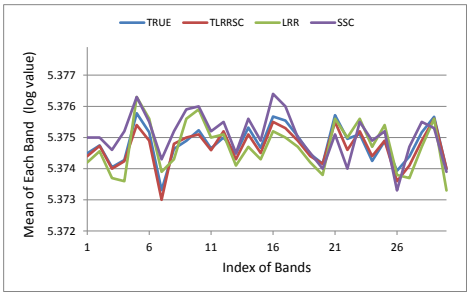


Fig. 5. Comparison between the true band mean and evaluated one on the synthetic data.

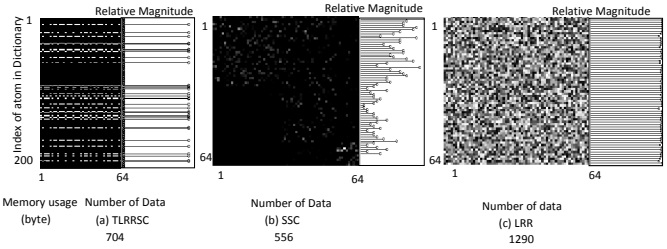
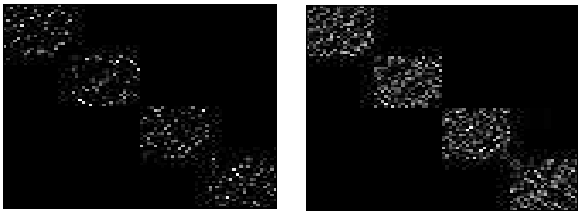


Fig. 6. Sparse coefficients, their relative magnitudes and memory usage of TLRRSC, LRR and SSC on the synthetic data

with almost the same relative magnitude, whereas in Fig. 6(b), far fewer atoms are activated with a higher magnitudes. This is mainly because LRR uses a holistic sparsity defined by low rank, where in SSC, sparsity is represented individually. The original high-dimensional data matrix $\widehat{\mathbf{X}}$ needs 1290 byte memory spaces, while all sparsity involved methods reduce space costs to some extent as shown in Fig. 6. In Fig. 6(a), our sparse representation only activates a few atoms with almost the same high magnitude. We can clearly see the number of lines in the white part of Fig. 6 (a) is fewer than that of Fig. 6(b). Although the memory usage of our model TLRRSC is 26% more than SSC, our sparse representation activates a far fewer number of atoms (Fig. 6(a)) than for SSC (Fig. 6(b)).

4) Performance Comparisons with Other LRR+SC Subspace Clustering Algorithm: We compare our algorithm's performance with another state-of-art algorithm LRSSC in [28], which also takes the advantage of SC and LRR. LRSSC minimizes a weighted sum of nuclear norm and vector 1-norm of the representation matrix simutanously, so as to preserve the properties of interclass separation and intra-class connectivity at the same time. Therefore, it works well in the matrices where data distribution is skewed and subspaces are not independent. Unlike LRSSC explicitly satisfies LRR and SC property simultaneously, our model updates the parameters for LRR and SC alternatively, and our model focuses on multidimensional data with a high dimensional feature space.

In the experiments, we randomly generate 4 disjoint subspaces of dimension 10 from $\mathbb{R}^{50}$, each sampled 20 data points. 50 unit length random samples are drawn from each subspace and we concatenate into a $\mathbb{R}^{50\times 80}$ data matrix. The clustering results are illustrated in Fig. 7. As we can see, our algorithm performs better than LRSSC, this is maybe because the alterative update LRR parameters and SC parameters in the iterations can help find better solution for each setting.



(a) LRSSC (b) TLRRSC

Fig. 7. Subspace Clustering Algorithm Performance Comparisons

C. Results on Real Datasets

We evaluate our model on a clean dataset called the Indianpines [24] and a corrupted dataset called Pavia University database [46].

The Indianpines dataset is gathered by AVIRIS sensor over the Indian Pines test site in North-western Indiana, and consists of 145×145 pixels and 224 spectral reflectance bands in the wavelength range 0.4-2.5 micrometers. The whole data set is formed by 16 different classes having an available ground truth. In our experiments, 24 bands covering the region of water absorption are discarded. The task is to group pixels into clusters according to their spectral reflectance bands information.

The Pavia University database is acquired by the ROSIS sensor with a geometric resolution of 1.3 meters, during a flight campaign over Pavia, nothern Italy. Pavia University consists of 610×340 pixels, each of which has 103 spectral bands covering 0.43 to 0.86 μm. The data set contains 9 different classes with available groundtruths. We examine the noise robustness of the proposed model by adding Guassian white noises with intensities ranging from 20 to 60 with a step of 20 to the whole database.

1) Subspace Clustering Performance: In this section, we show TLRRSC's performance in subspace clustering with the subspace number given. Table II shows the results of all baseline methods on both datasets. Clearly, our method TLRRSC outperforms the other six baselines on this dataset. The advantage of TLRRSC mainly comes from its ability to incorporate 2 dimensional data structure information and 200 dimensional bands information into the low-rank representation and sparse representation.

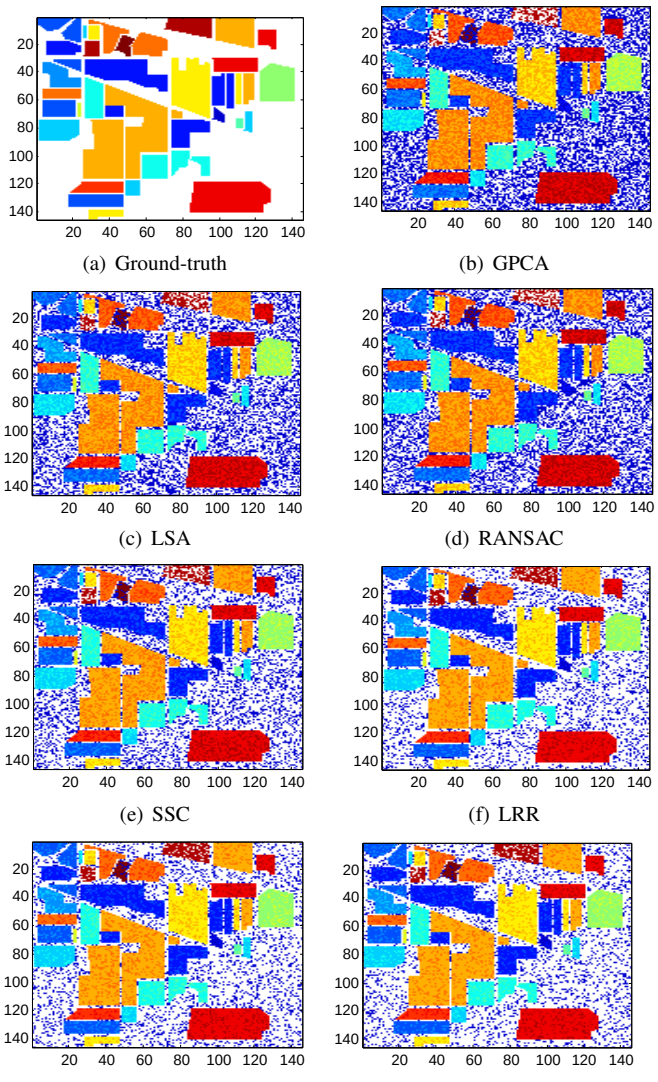
Besides, the efficiency (in terms of running time) of TLRRSC is comparable to TLRR, GPCA and RANSAC methods. TLRR costs more computational time because its optimization procedure needs more iterations than GPCA and RANSAC to converge. The results regarding time cost on TLRR and LRR are consistent with Remark 1 in Section IV-A, which shows that TLRR significantly reduces time cost by exploiting the Kronecker structure along each space dimension. Although GPCA and RANSAC are faster than LRR and TLRR, their accuracy is much lower than those of LRR and TLRR. Even though TLRRSC uses 22 more minutes than TLRR for a dictionary learning task in the feature space, its overall performance is better than TLRR.

TABLE II
SUBSPACE CLUSTERING RESULTS ON THE REAL DATASETS

			Subspace clustering accuracy(%)						
			GPCA	LSA	RANSAC	SSC	LRR	TLRR	TLRRSC
Indianpines		Mean	47.6	58.3	53.2	69.8	77.6	78.6	**80.5**
		Std.	10.45	10.56	9.98	7.02	5.45	4.67	**4.08**
		Max	70.9	81.5	78.3	80.7	85.4	89.7	**92.4**
		Time (min.)	6.87	177.84	**5.90**	745.73	380.07	51.23	73.48
Pavia University	Intensity=20	Mean	30.2	51.65	46.7	61.9	72.3	74.6	**76.8**
		Std.	12.83	10.69	8.76	8.95	5.38	4.54	**3.97**
		Max	62.5	70.1	73.8	80.6	85.7	87.6	**91.2**
		Time (hr.)	1.84	27.31	**0.76**	95.17	41.02	7.69	11.05
	Intensity=40	Mean	25.68	48.7	44.2	57.7	69.1	70.4	**73.5**
		Std.	11.79	13.69	7.68	9.75	6.74	6.01	**3.08**
		Max	60.2	65.17	69.8	76.8	80.1	82.5	**87.6**
		Time (hr.)	1.69	28.76	**0.79**	96.12	40.87	7.15	10.98
	Intensity=60	Mean	23.2	44.12	42.07	52.78	66.8	67.8	**69.8**
		Std.	10.68	15.09	8.67	8.99	4.96	4.02	**3.17**
		Max	54.2	60.2	63.8	71.6	78.7	79.3	**85.9**
		Time (hr.)	1.58	29.33	**0.97**	94.15	43.07	7.05	9.99

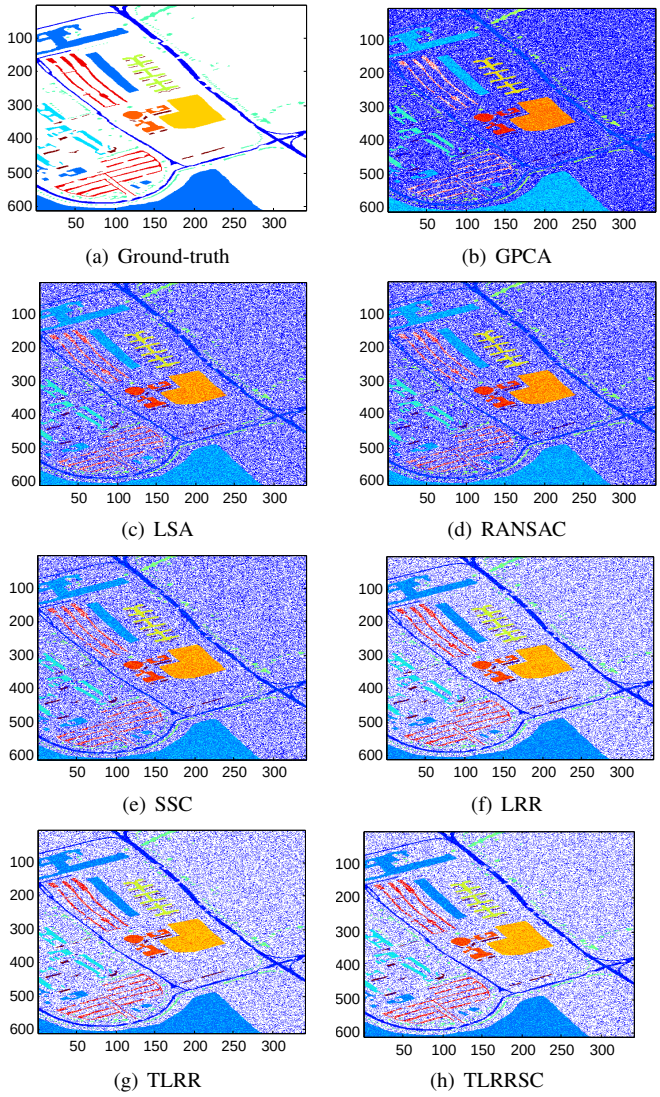
When data are corrupted, the performance of SSC is inferior to all LRR based methods, which shows that sparse representation is not good at handling corrupted data like LRR. Alt hough our model employs a sparse representation on the feature space, our model TLRRSC still performs best among all the methods on the corrupted data set. This is because TLRRSC explores data spatial correlation information with a low-rank representation, which guarantees accurately clustering data into different subgroups. Fig. 8 and Fig. 9 visualize the subspace clustering results on both datasets. In the above two figures, each cluster is represented by a particular color. Obviously, we can see many blue points scattered in Fig. 8 and Fig. 9, which originally belonging to other class are wrongly classified to the blue class. Similarly, there are a few other colors scattered in the green class in Fig.8 and the orange class in Fig. 9. Accordingly, it is easy to see that the clustering results on TLRRSC are closest to the groundtruths.

2) Choosing the Parameter λ: The parameter $\lambda > 0$ is used to balance the effects of the two parts in problem (11). Generally speaking, the choice of this parameter depends on the prior knowledge of the error level of data. When the errors are slight, a relatively larger λ should be used; while when the errors are heavy, we should set a smaller value. The blue curve in Fig. 10 is the evaluation results on the Indianpines data set. While λ ranges from 0.04 to 0.2, the clustering accuracy slightly varies from 80.34 % to 81.98 %. This phenomenon is mainly because TLRRSC employs LRR representation to explore data structure information. It has been proved that LRR works well on clean data (the indianpines is a clean data set), and there is an "invariance" in LRR that implies that it can be partially stable while λ varies (For the proof of this property see Theorem 4.3 in [29]). Notice that TLRRSC is more sensitive to λ on the Pavia University data set than on the Indianpines data set. This is because the samples in the Indianpines data set are clean, whereas the Pavia University data set contains some corrupted information. The more heavily data are corrupted, the performance of our new mehtod is more influenced by the λ value.



(a) Ground-truth (b) GPCA (c) LSA (d) RANSAC (e) SSC (f) LRR (g) TLRR (h) TLRRSC

Fig. 8. Subspace Clustering results on Indianpines, each color represents a class



(a) Ground-truth (b) GPCA (c) LSA (d) RANSAC (e) SSC (f) LRR (g) TLRR (h) TLRRSC

Fig. 9. Subspace Clustering results on Pavia University(Noise Intensity=60), each color represents a class

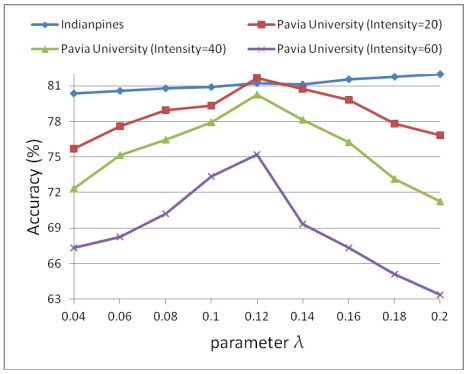


Fig. 10. The influences of the parameter λ of TLRRSC. These results are collected from the Indianpines data set and the Pavia University data set.

TABLE III
MEMORY USAGE COMPARISON ON REAL DATASETS

	Memory Usage for Sparse Representation (MB)			
	Original	TLRRSC	SSC	LRR
Indianpines	4.20	1.54	**1.21**	2.48
Pavia University	21.36	7.83	**6.18**	11.87

3) Memory Usage w.r.t. Different Sparsity Strategies: In TLRRSC, we learn a sparse representation on the feature mode (i.e. the 3rd mode) through a dictionary learning task. In this section, we compare the memory usage among TLRRSC, SSC and LRR on the mode-3 matricization of Indianpine database and Pavia University database. For an order-3 tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the memory complexity of our model TLRRSC and SSC are $O(r \times (I_1 \times I_2))$ at most, where r is the maximum sparsity for each instance. While LRR requires $O(m \times (I_1 \times I_2))$, where m is the rank of the sparse representation. Usually $(m \gg r)$. As shown in the first row of Table III, SSC has the least memory cost, and TLRRSC takes the second place. The reason behind this phenomenon is that our dictionary learning model based on both spatial information and feature relation involves more structured sparse representation than feature based SSC model. However, TLRRSC has an advantage over SSC in accuracy and running time as shown in Table II. Accordingly, our model maintains good performance with a comparable memory cost. The second row in Table III shows the memory cost of TLRRSC, LRR and SSC on Pavia University. The memory usage of SSC is the lowest, which is consistent with the result on Indianpines. However, the performance of SSC depresses on the corrupted data set as shown in Table II, while LRR is very effective in handling noise. Moreover, our model's memory usage is comparable. Therefore, we assert that TLRRSC is a noise robust method with low memory usage.

VI. CONCLUSIONS

We propose a tensor based low-rank representation (TLRR) and sparse coding (SC) for subspace clustering in this paper. Unlike existing subspace clustering methods work on an unfolded matrix, TLRRSC builds a model on data original structure form (i.e. tensor) and explores data similarities along all spatial dimensions and feature dimension. On the synthetic higher mode tensorial datasets, we show that our model considering data structure maintains a good performance. Moreover, the experimental results with different noise rates show our model maintains a good performance on highly corrupted data. On the real-world dataset, our method shows promising results, with low computation gains and memory usage. Moreover, our model is robust to noises, and capable of recovering corrupted data.

ACKNOWLEDGMENT

This work is supported by the Australian Research Council (ARC) through Discovery Project Grant DP130100364. Zhouchen Lin is supported by National Basic Research Program of China (973 Program) (grant no. 2015CB352502), National Natural Science Foundation (NSF) of China (grant

nos. 61272341 and 61231002), and Microsoft Research Asia Collaborative Research Program.

REFERENCES

[1] M. Aharon, M. Elad, and A. Brucktein, "K-SVD: An algorithm for desdesign overcomplete dictionaries for sparse representation," *IEEE Trans. on Signal Processing*, vol. 54, no. 2, pp. 4311–4322, 2006.

[2] B. W. Bader and T. G. Kolda, "Efficient MATLAB computations with sparse and factored tensors," *SIAM Journal of Scientific Computing*, vol. 30, no. 1, pp. 205–231, 2008.

[3] R. Basri and D. Jacobs, "Lambertian reflectance and linear subspaces," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 218–233, 2003.

[4] M. Blondel, K. Seki, and K. Uehara, "Block coordinate descent algorithms for large-scale sparse multiclass classification," *Machine Learning*, vol. 93, no. 1, pp. 31–52, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10994-013-5367-2

[5] T. Blumensath and M. Davies, "Normalised iterative hard thresholding; guaranteed stability and performance," *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, pp. 298–309, 2010.

[6] M. Bouguessa, S. Wang, and Q. Jiang, "A k-means-based algorithm for projective clustering," in *ICPR*, vol. 1, 2006, pp. 888–891.

[7] J. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. on Optimization*, vol. 20, no. 4, pp. 1956–1982, 2010. [Online]. Available: http://dx.doi.org/10.1137/080738970

[8] E. Candes and B. Recht, "Exact matrix completion via convex optimization," *Commun. ACM*, vol. 55, no. 6, pp. 111–119, 2012. [Online]. Available: http://doi.acm.org/10.1145/2184319.2184343

[9] Y. Chen, A. Jalali, S. Sanghavi, and H. Xu, "Clustering partially observed graphs via convex optimization," in *ICML*, 2011.

[10] C. Ding, D. Zhou, X. He, and H. Zha, "R1-PCA: Rotational invariant l1-norm principal component analysis for robust subspace factorization," in *ICML*, 2006.

[11] A. Donoho, D. L.and Maleki and A. Montanari, "Message passing algorithms for compressed sensing: I. motivation and construction," in *Information Theory Workshop(ITW), IEEE*, 2010.

[12] M. Donoho, D. L.and Elad, "Optimally sparse representation in general (non-orthogonal) dictionaries via l1 minimization," in *Proc. Natl Acad. Sci.*, 2003, pp. 2197–2202.

[13] M. Elad, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transactoins on Image Processing*, vol. 54, pp. 3736–3745, 2006.

[14] E. Elhamifar and R. Vidal, "Sparse subspace clustering," in *CVPR*, 2009, pp. 2790–2797.

[15] K. Engan, S. Aase, and J. Husoy, "Method of optimal directions for frame design," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 1, 1999, pp. 2443–2446.

[16] Y. Fang, J. Wu, and B. Huang, "2d sparse signal recovey via 2d orthogonal matching pursuit," *Science China Information Sciences*, vol. 55, pp. 889–897, 2012.

[17] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[18] S. Foucart, "Hard thresholding pursuit: An algorithm for compressive sensing," *SIAM J. on Numerical Analysis*, vol. 49, pp. 2543–2563, 2011.

[19] J. Ho, M. Yang, J. Lim, K. Lee, and D. Kriegman, "Clustering appearances of objects under varying illumination conditions," in *CVPR*, vol. 1, 2003, pp. 11–18.

[20] K. Kanatani, "Motion segmentation by subspace separation and model selection," in *ICCV*, vol. 2, 2001, pp. 586–591.

[21] R. Keshavan, A. Montanari, and S. Oh, "Matrix completion from noisy entries," *J. Mach. Learn. Res.*, vol. 11, pp. 2057–2078, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1859920

[22] G. Kolda and B. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51(3), pp. 455–500, 2009.

[23] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistic Quarterly*, vol. 2, pp. 83–97, 1955.

[24] D. Landgrebe, "Multispectral data analysis: A signal theory perspective," Purdue Univ., West Lafayette, IN, Tech. Rep., 1998.

[25] Y. LeCun, L. Bottou, G. Orr, and K. Muller, *Neutral Networks: Tricks of the trade*, Springer, 1998.

[26] S. Li, "Non-negative sparse coding shrinkage for image denoising using normal inverse gaussian density model," *Image Vision Comput.*, vol. 26, no. 8, pp. 1137–1147, Aug. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.imavis.2007.12.006

[27] Z. Lin, R. Liu, and Z. Su, "Linearized alternating direction method with adaptive penalty for low-rank representation," in *NIPS*, 2011.

[28] Y. Wang, H, Xu and C. Leng, "Provable Subspace Clustering: When LRR meets SSC, " in *NIPS*, 2013.

[29] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma, "Robust recovery of subspace structures by low-rank representation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 171–184, 2013.

[30] R. Liu, Z. Lin, and Z. Su, "Linearized alternating direction method with parallel splitting and adaptive penalty for separable convex programs in machine learning,," in *Proceedings of ACML*, 2013.

[31] Y. Ma, H. Derksen, W. Hong, and J. Wright, "Segmentation of multivariate mixed data via lossy data coding and compression," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 9, pp. 1546–1562, 2007.

[32] J. Martin and S. Marek, "A simple algorithm for nuclear norm regularized problems," in *ICML*, 2010, pp. 471–478. [Online]. Available: http://www.icml2010.org/papers/196.pdf

[33] D. Needell, J. Tropp, and R. Vershynin, "Greedy signal recovery review," in *Signals, Systems and Computers, 42nd Asilomar Conference on*. IEEE, 2008, pp. 1048–1050.

[34] D. Needell and J. Tropp, "Cosamp: Iterative signal recovery from incomplete and inaccurate samples," *Commun. ACM*, vol. 53, no. 12, pp. 93–100, Dec. 2010. [Online]. Available: http://doi.acm.org/10.1145/1859204.1859229

[35] B. Olshausen and D. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural image," *Nature*, vol. 381, no. 2, pp. 607–609, 1996.

[36] G. Peyré, "Sparse Modeling of Textures," *Journal of Mathematical Imaging and Vision*, vol. 34, no. 1, pp. 17–31, May 2009. [Online]. Available: http://dx.doi.org/10.1007/s10851-008-0120-3

[37] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng, "Self-taught learning: tranfer learning from unlabeled data," in *the International Conference on Machine Learning (ICML)*, 2007.

[38] S. Rao, A. Yang, S. Sastry, and Y. Ma, " Robust algebraic segmentation of mixed rigid-body and planar motions from two views," *International Journal of Computer Vision*, vol. 88, no. 3, pp. 425–446, 2010. [Online]. Available: http://dx.doi.org/10.1007/s11263-009-0314-1

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

[40] J. Shlens, "A tutorial on principal component analysis," in *Systems Neurobiology Laboratory, Salk Institute for Biological Studies*, 2005.

[41] M. Tipping and C. Bishop, "Mixtures of probabilistic principal component analyzers," *Neural Comput.*, vol. 11, no. 2, pp. 443–482, Feb. 1999. [Online]. Available: http://dx.doi.org/10.1162/089976699300016728

[42] J. Tropp, "Greed is good: algorithmic results for sparse approximation," *Information Theory, IEEE Transactions on*, vol. 50, no. 10, pp. 2231–2242, 2004.

[43] J. Tropp and A. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *Information Theory, IEEE Transactions on*, vol. 53, no. 12, pp. 4655–4666, 2007.

[44] R. Vidal, "Subspace clustering," *Signal Processing Magazine, IEEE*, vol. 28, pp. 52–68, 2011.

[45] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (GPCA)," in *CVPR*, vol. 1, 2003, pp. 621–628.

[46] L. Wei, S. Li, M. Zhang, Y. Wu, S. Su, and R. Ji, "Spectral-spatial classification of hyperspectral imagery based on random forests," in *Proceedings of the Fifth International Conference on Internet Multimedia Computing and Service*, New York, NY, USA: ACM, 2013, pp. 163–168. [Online]. Available: http://doi.acm.org/10.1145/2499788.2499853

[47] J. Yan and M. Pollefeys, "A general framework for motion segmentatise: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate." in *ECCV*, 2006.

[48] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *in IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2009.

[49] T. Zhang, A. Szlam and G. Lerman, "Median k-flats for hybrid linear modeling with many outliers," in *ICCV*, 2009, pp. 234–241.

[50] Y. Fu, J. Gao, D. Tien and Z. Lin, "Tensor LRR Based Subspace Clustering," in *IJCNN*, 2014, pp. 1877–1884.

[51] L. Jing, M.K. Ng and T. Zeng, "Dictionary Learning-Based Subspace Structure Identification in Spectral Clustering," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 24, no. 8, pp. 1188–1199, 2013.

[52] K. Tang, R.Liu and Z. Su, "Structure-Constrained Low-Rank Representation," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–14, 2014.

[53] Y. Deng, Q. Dai, R. Liu, Z. Zhang and S. Hu, "Low-rank structure learning via nonconvex heuristic recovery," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 24, no. 3, pp. 383–396, 2013.
[54] W. Yong, J. Yuan, W. Yi and Z. Zhou, "Spectral clustering on multiple manifolds," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 22, no. 7, pp. 1149–1161, 2011.
[55] R. He, W. Zheng, B. Hu and X. Kong, "Two-Stage Nonnegative Sparse Representation for Large-Scale Face Recognition," *Neural Networks and Learning Systems, IEEE Transactions on*, vol. 24, no. 1, pp. 35–46, 2013.

Yifan Fu received her PhD degree in Computer Science from University of Technology Sydney, Sydney Australia, in 2013. She received her M.E. degree in Software Engineering from Northeast Normal University, Changchun China, in 2009. She is currently a research associate in the school of Computing and Mathematics, Charles Sturt University, Australia (since August 2013). Her research interests lie in Machine Learning and Data Mining; more specifically, she is interested in active learning, ensemble methods, graph mining and tensor decompostion.

Junbin Gao graduated from Huazhong University of Science and Technology (HUST), China in 1982 with BSc. degree in Computational Mathematics and obtained PhD from Dalian University of Technology, China in 1991. He is a Professor of Big Data Analytics in the University of Sydney Business School at the University of Sydney and was a Professor in Computer Science in the School of Computing and Mathematics at Charles Sturt University, Australia. He was a senior lecturer, a lecturer in Computer Science from 2001 to 2005 at University of New England, Australia. From 1982 to 2001 he was an associate lecturer, lecturer, associate professor and professor in Department of Mathematics at HUST. His main research interests include machine learning, data analytics, Bayesian learning and inference, and image analysis.

David Tien received his undergraduate, master's and PhD degrees in Computer Science, Pure Mathematics and Electrical Engineering from the Heilongjiang University, Chinese Academy of Sciences, Ohio State University, USA, and the University of Sydney, Australia, respectively. Dr Tien's research interests are in the areas of image and signal processing, artificial intelligent, telecommunication coding theory and biomedical engineering. He is currently teaching computer science at the Charles Sturt University, Australia and serves as the Chairman of IEEE NSW Computer Chapter.

Zhouchen Lin (M'00-SM'08) received the Ph.D. degree in Applied Mathematics from Peking University, in 2000. He is currently a Professor at Key Laboratory of Machine Perception (MOE), School of Electronics Engineering and Computer Science, Peking University. Before March 2012, he was a Lead Researcher at Visual Computing Group, Microsoft Research Asia. His research interests include computer vision, image processing, computer graphics, machine learning, pattern recognition, and numerical computation and optimization. He is an Associate Editor of IEEE Trans. Pattern Analysis and Machine Intelligence and International J. Computer Vision, an area chair of CVPR 2014, ICCV 2015, NIPS 2015, AAAI 2016, CVPR 2016, and IJCAI 2016.

Xia Hong received her university education at National University of Defense Technology, P.R. China (B.Sc., 1984, M.Sc., 1987), and University of Sheffield, UK (Ph.D.,1998), all in automatic control. She worked as a Research Assistant in Beijing Institute of Systems Engineering, Beijing, China from 1987 to 1993. She worked as a Research Fellow in the Department of Electronics and Computer Science at University of Southampton from 1997 to 2001. She is currently a Professor at Department of Computer Science, School of Mathematical and Physical Sciences, 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 over 100 research papers, and coauthored a research book. She was awarded a Donald Julius Groen Prize by IMechE in 1999.