2020 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, SEPT. 21–24, 2020, ESPOO, FINLAND

ONLINE TENSOR LOW-RANK REPRESENTATION FOR STREAMING DATA

Tong Wu

Department of Electrical and Computer Engineering, Rutgers University–New Brunswick
tong.wu.ee@rutgers.edu

ABSTRACT

This paper proposes a new streaming algorithm to learn low-rank structures of tensor data using the recently proposed tensor-tensor product (t-product) and tensor singular value decomposition (t-SVD) algebraic framework. It reformulates the tensor low-rank representation (TLRR) problem using the equivalent bifactor model of the tensor nuclear norm, where the tensor dictionary is updated based on the newly collected data and representations. Compared to TLRR, the proposed method processes tensor data in an online fashion and makes the memory cost independent of the data size. Experimental results on three benchmark datasets demonstrate the superior performance, efficiency and robustness of the proposed algorithm over state-of-the-art methods.

***Index Terms*—** Clustering, online learning, tensor low-rank representation

1. INTRODUCTION

Subspace clustering, whose objective is to segment the data points into different groups such that each cluster corresponds to one subspace, is one of the fundamental tasks in machine learning and computer vision [1]. In the past two decades, a number of subspace clustering algorithms have been developed in the literature [2, 3, 4, 5, 6], among which spectral clustering based approaches [4, 5, 6] have gained a lot of attention due to their promising performance; two typical such methods include sparse subspace clustering (SSC) [6] and low-rank representation (LRR) [5]. Both of them use the idea of self-expressiveness and seek a sparse or low-rank representation of the data. The clustering result can be obtained by means of spectral clustering (e.g., Ncut [7]). In order to enhance the computation and storage efficiency, an online LRR method for streaming data has been introduced in [8], which implements LRR based on stochastic optimization with guaranteed convergence.

Although subspace clustering has been well studied in the matrix domain, many real-world data are multiway, e.g., videos, hyperspectral images, and multichannel electroencephalography (EEG) signals. In traditional subspace clustering literature, multi-dimensional data are always flattened into vectors such that conventional matrix-based algorithms can be applied. This simplistic approach can degrade performance because the intrinsic multi-dimensional structure of data is destroyed [9, 10, 11]. To address this problem, some recent works have been proposed using tensor algebraic frameworks [12, 9], which exploit the multi-dimensional structure of tensor data [13, 14, 15, 16, 17]. In particular, suppose that we are given a third-order data tensor, where each lateral slice of this tensor corresponds to one data sample. Motivated by the t-product [12, 9], one type of the tensor-tensor products for third-order tensors, the samples can be assumed to be drawn from a union of tensor subspaces (see Definition 8). Similar to matrix-based subspace clustering methods, each sample can now be expressed as a t-linear combination of other samples and the representation tensor coefficients suggest the tensor subspace membership of the samples. Concretely, sparse submodule clustering (SSmC) [13] imposes a sparsity constraint on the representation tensor. In tensor LRR (TLRR) [15], the tensor nuclear norm [11] is used to recover the low-rank structure of tensor data. However, these methods are often computationally expensive for large-scale tensor data, and even storing these tensors is problematic since the memory requirements grow rapidly with the size of data.

In this paper, we propose an online TLRR learning method, termed online low-rank tensor subspace clustering (OLRTSC), to recover low-rank tensor data and cluster them from streaming multi-dimensional data. OLRTSC reduces the memory cost of TLRR and mitigates the computational bottleneck of the tensor nuclear norm regularizer in TLRR. Our work is closely related to that in [8], which performs online subspace clustering. The difference between this work and [8] lies in the fact that OLRTSC preserves the multi-dimensional structure in tensor data and directly seeks a low-rank representation of the tensor. Numerical experiments on real-world image datasets demonstrate the effectiveness of OLRTSC.

The rest of this paper is organized as follows. Section 2 provides some notations and preliminaries. In Section 3 we describe our proposed online clustering algorithm. We present experimental results in Section 4 to show efficacy of the proposed method and conclude in Section 5.

2. NOTATIONS AND PRELIMINARIES

In this section, we introduce some notations and basic definitions for tensors. Throughout this paper, we use calligraphy letters for tensors, e.g., $\mathcal{A}$, bold uppercase letters for matrices, e.g., $\mathbf{A}$, bold lowercase letters for vectors, e.g., $\mathbf{a}$, and non-bold letters for scalars, e.g., a. The (i, j)-th element of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}_{i,j}$ and its i-th column is denoted by $\mathbf{a}_i$. The identity matrix is denoted by $\mathbf{I}$ of appropriate dimension. We denote a third-order tensor as $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and its (i, j, k)-th entry as $\mathcal{A}_{i,j,k}$. We use $\mathcal{A}(:, :, i)$ or $\mathbf{A}^{(i)}$ to denote its i-th frontal slice. Any lateral slice of size $n_1 \times 1 \times n_3$ is denoted as $\overrightarrow{\mathcal{A}}$. In particular, we use $\overrightarrow{\mathcal{A}}_i$ to denote the i-th lateral slice of tensor $\mathcal{A}$. The (i, j)-th mode-1, mode-2 and mode-3 fibers of $\mathcal{A}$ are represented by $\mathcal{A}(:, i, j)$, $\mathcal{A}(i, :, j)$ and $\mathcal{A}(i, j, :)$, respectively. We denote $\widehat{\mathcal{A}}$ the discrete Fourier transform (DFT) along mode-3 of $\mathcal{A}$, that is, $\widehat{\mathcal{A}} = \text{fft}(\mathcal{A}, [\,], 3)$. Similarly, $\mathcal{A}$ can be computed from $\widehat{\mathcal{A}}$ via inverse DFT, i.e., $\mathcal{A} = \text{ifft}(\widehat{\mathcal{A}}, [\,], 3)$. The ℓ_1 and Frobenius norms of $\mathcal{A}$ are defined as $\|\mathcal{A}\|_1 = \sum_{i,j,k} |\mathcal{A}_{i,j,k}|$ and $\|\mathcal{A}\|_F = \sqrt{\sum_{i,j,k} |\mathcal{A}_{i,j,k}|^2}$, respectively. Let $\widehat{\mathbf{A}} \in \mathbb{C}^{n_1 n_3 \times n_2 n_3}$ be a block

978-1-7281-6662-9/20/$31.00 ©2020 IEEE

diagonal matrix with its i-th block on diagonal as the i-th frontal slice $\widehat{\mathbf{A}}^{(i)}$ of $\widehat{\mathcal{A}}$, i.e.,

$$\widehat{\mathbf{A}} = \text{bdiag}(\widehat{\mathcal{A}}) = \begin{bmatrix} \widehat{\mathbf{A}}^{(1)} & & & \\ & \widehat{\mathbf{A}}^{(2)} & & \\ & & \ddots & \\ & & & \widehat{\mathbf{A}}^{(n_3)} \end{bmatrix}.$$

Also, we define the block circulant matrix $\text{bcirc}(\mathcal{A}) \in \mathbb{R}^{n_1 n_3 \times n_2 n_3}$ of $\mathcal{A}$ as

$$\text{bcirc}(\mathcal{A}) = \begin{bmatrix} \mathbf{A}^{(1)} & \mathbf{A}^{(n_3)} & \dots & \mathbf{A}^{(2)} \\ \mathbf{A}^{(2)} & \mathbf{A}^{(1)} & \dots & \mathbf{A}^{(3)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^{(n_3)} & \mathbf{A}^{(n_3-1)} & \dots & \mathbf{A}^{(1)} \end{bmatrix}.$$

Then we define the block vectorizing and its opposite operation as

$$\text{bvec}(\mathcal{A}) = \begin{bmatrix} \mathbf{A}^{(1)} \\ \mathbf{A}^{(2)} \\ \vdots \\ \mathbf{A}^{(n_3)} \end{bmatrix} \in \mathbb{R}^{n_1 n_3 \times n_2}, \ \text{bvfold}(\text{bvec}(\mathcal{A})) = \mathcal{A}.$$

Definition 1 (t-product [9])**.** *The t-product between two tensors $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2 \times n_4 \times n_3}$ is defined as $\mathcal{A} * \mathcal{B} = bvfold(bcirc(\mathcal{A}) \cdot bvec(\mathcal{B})) \in \mathbb{R}^{n_1 \times n_4 \times n_3}$.*

Definition 2 (Tensor transpose)**.** *Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, then $\mathcal{A}^T$ is an $n_2 \times n_1 \times n_3$ tensor obtained by transposing each of $\mathcal{A}$'s frontal slices and then reversing the order of the transposed frontal slices 2 through n_3.*

Definition 3 (Identity tensor)**.** *The identity tensor $\mathcal{I} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$ is a tensor whose first frontal slice is the $n_1 \times n_1$ identity matrix and all other frontal slices are zeros.*

Definition 4 (Orthogonal tensor)**.** *A tensor $\mathcal{Q} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$ is orthogonal if $\mathcal{Q}^T * \mathcal{Q} = \mathcal{Q} * \mathcal{Q}^T = \mathcal{I}$.*

Definition 5 (Tensor singular value decomposition (t-SVD) [9])**.** *Let $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, then it can be factorized as*

$$\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T,$$

where $\mathcal{U} \in \mathbb{R}^{n_1 \times n_1 \times n_3}$, $\mathcal{V} \in \mathbb{R}^{n_2 \times n_2 \times n_3}$ are orthogonal, and $\mathcal{S} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ is a tensor whose frontal slices are diagonal matrices.

Definition 6 (Tensor tubal rank [9])**.** *The tensor tubal rank, denoted as $rank_t(\mathcal{A})$, is defined as the number of nonzero singular tubes of $\mathcal{S}$, where $\mathcal{S}$ is from the t-SVD of $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$. That is, $rank_t(\mathcal{A}) = \#\{i : \mathcal{S}(i,i,:) \neq 0\}$.*

Definition 7 (Tensor nuclear norm [11])**.** *Let $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^T$ be the t-SVD of $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. The tensor nuclear norm $\|\mathcal{A}\|_*$ of $\mathcal{A}$ is defined as $\|\mathcal{A}\|_* = \sum_{i=1}^{r} \mathcal{S}_{i,i,1}$, where $r = rank_t(\mathcal{A})$.*

Let $\mathbb{K}_{n_3}^{n_1}$ denote the set of all $n_1 \times 1 \times n_3$ lateral slices. Then a set of tensors $\{\overrightarrow{\mathcal{D}}_1, \dots, \overrightarrow{\mathcal{D}}_p\} \subseteq \mathbb{K}_{n_3}^{n_1}$, stored as lateral slices of a tensor $\mathcal{D} \in \mathbb{R}^{n_1 \times p \times n_3}$, is said to be *linearly independent* if there is not a nonzero tensor $\mathcal{C} \in \mathbb{R}^{p \times 1 \times n_3}$ such that $\mathcal{D} * \mathcal{C} = 0$ [15].

Definition 8 (Tensor subspace [15])**.** *Given a set $\{\overrightarrow{\mathcal{D}}_1, \dots, \overrightarrow{\mathcal{D}}_p\} \subseteq \mathbb{K}_{n_3}^{n_1}$ in which the elements $\overrightarrow{\mathcal{D}}_i$'s are linearly independent, the set $\mathcal{S}_{n_3}^{n_1} = \{\mathcal{Y} | \mathcal{Y} = \mathcal{D} * \mathcal{C}, \forall \mathcal{C} \in \mathbb{R}^{p \times 1 \times n_3}\}$ is called a tensor subspace of dimension $dim(\mathcal{S}_{n_3}^{n_1}) = p$. Here, $\overrightarrow{\mathcal{D}}_1, \overrightarrow{\mathcal{D}}_2, \dots, \overrightarrow{\mathcal{D}}_p$ are called the spanning basis of $\mathcal{S}_{n_3}^{n_1}$.*

3. ONLINE LOW-RANK TENSOR SUBSPACE CLUSTERING

3.1. Problem Formulation

Consider a collection of N 2-D data samples $\overrightarrow{\mathcal{Z}}_1, \overrightarrow{\mathcal{Z}}_2, \dots, \overrightarrow{\mathcal{Z}}_N$ of size $n_1 \times n_3$, that are drawn from a union of L low-dimensional tensor subspaces $\{{}^{\ell}\mathcal{S}_{n_3}^{n_1}\}_{\ell=1}^{L}$. We arrange them as lateral slices to make a third-order tensor $\mathcal{Z}$ of size $n_1 \times N \times n_3$. Tensor low-rank representation (TLRR) seeks a low-rank representation of the tensor data using a predefined dictionary to infer the clusters of samples. Mathematically, TLRR can be formulated as the following optimization problem [15]:

$$\min_{\mathcal{X},\mathcal{E}} \frac{\lambda_1}{2} \|\mathcal{Z} - \mathcal{Y} * \mathcal{X} - \mathcal{E}\|_F^2 + \|\mathcal{X}\|_* + \lambda_2 \|\mathcal{E}\|_1, \tag{1}$$

where $\mathcal{Y} \in \mathbb{R}^{n_1 \times N \times n_3}$ is a predefined *atom dictionary*, $\mathcal{X} \in \mathbb{R}^{N \times N \times n_3}$ is the low-rank representation of the data over $\mathcal{Y}$, $\mathcal{E}$ is a tensor representing the approximation errors of the data, and λ_1 and λ_2 are penalty parameters. Now suppose the 2-D samples $\overrightarrow{\mathcal{Z}}_1, \overrightarrow{\mathcal{Z}}_2, \dots, \overrightarrow{\mathcal{Z}}_N$ are revealed sequentially, our goal is to efficiently learn the representation tensor $\mathcal{X}$ and the error tensor $\mathcal{E}$ in an online fashion. Assume the tubal rank of $\mathcal{X}$ is at most d, we use an equivalent form of the tensor nuclear norm of $\mathcal{X}$ as follows [18]:

$$\|\mathcal{X}\|_* = \min_{\substack{\mathcal{U} \in \mathbb{R}^{N \times d \times n_3} \\ \mathcal{V} \in \mathbb{R}^{N \times d \times n_3} \\ \mathcal{X} = \mathcal{U} * \mathcal{V}^T}} \frac{n_3}{2} (\|\mathcal{U}\|_F^2 + \|\mathcal{V}\|_F^2). \tag{2}$$

Thus (1) can be reformulated as the following non-convex optimization problem:

$$\min_{\mathcal{U},\mathcal{V},\mathcal{E}} \frac{\lambda_1}{2} \|\mathcal{Z} - \mathcal{Y} * \mathcal{U} * \mathcal{V}^T - \mathcal{E}\|_F^2 + \frac{n_3}{2} (\|\mathcal{U}\|_F^2 + \|\mathcal{V}\|_F^2) + \lambda_2 \|\mathcal{E}\|_1.$$

Unlike [18], however, the sizes of $\mathcal{U}$ and $\mathcal{V}$ are both proportional to N and the dictionary $\mathcal{Y}$ is partially observed at each time t, thus making the method in [18] infeasible to TLRR. Similar to [8], we introduce an auxiliary variable $\mathcal{D} = \mathcal{Y} * \mathcal{U} \in \mathbb{R}^{n_1 \times d \times n_3}$, which we refer to as a *basis dictionary*, and get an equivalent reformulation given by

$$\begin{aligned} \min_{\mathcal{D},\mathcal{U},\mathcal{V},\mathcal{E}} \frac{\lambda_1}{2} \|\mathcal{Z} - \mathcal{Y} * \mathcal{U} * \mathcal{V}^T - \mathcal{E}\|_F^2 + \frac{n_3}{2} (\|\mathcal{U}\|_F^2 + \|\mathcal{V}\|_F^2) \\ + \lambda_2 \|\mathcal{E}\|_1 \quad \text{s.t. } \mathcal{D} = \mathcal{Y} * \mathcal{U}. \end{aligned} \tag{3}$$

We incorporate the constraint into the objective function and pose the goal of online TLRR in terms of the following optimization problem:

$$\begin{aligned} \min_{\mathcal{D},\mathcal{U},\mathcal{V},\mathcal{E}} \frac{\lambda_1}{2} \|\mathcal{Z} - \mathcal{D} * \mathcal{V}^T - \mathcal{E}\|_F^2 + \frac{n_3}{2} (\|\mathcal{U}\|_F^2 + \|\mathcal{V}\|_F^2) \\ + \lambda_2 \|\mathcal{E}\|_1 + \frac{\lambda_3}{2} \|\mathcal{D} - \mathcal{Y} * \mathcal{U}\|_F^2, \end{aligned} \tag{4}$$

where λ_3 is used to control the reconstruction quality of the basis dictionary $\mathcal{D}$.

3.2. Online Optimization for OLRTSC

In this section, we describe our online algorithm for optimizing the objective function in (4). Let $\overrightarrow{\mathcal{Z}}_t$, $\overrightarrow{\mathcal{Y}}_t$, $\overrightarrow{\mathcal{E}}_t$, $\overrightarrow{\mathcal{U}}_t$ and $\overrightarrow{\mathcal{V}}_t$ denote the

Algorithm 1 Online Low-Rank Tensor Subspace Clustering

Input: Sequentially observed data $\mathcal{Z} \in \mathbb{R}^{n_1 \times N \times n_3}$, $\mathcal{Y} \in \mathbb{R}^{n_1 \times N \times n_3}$, parameters λ_1, λ_2 and λ_3.
Initialize: Random initial basis $\mathcal{D}_0 \in \mathbb{R}^{n_1 \times d \times n_3}$, $\mathcal{M}_0 = 0$, $\mathcal{A}_0 = 0$, and $\mathcal{B}_0 = 0$.

1: **for** $t = 1, 2, \ldots, N$ **do**
2: Access the t-th sample $\overrightarrow{\mathcal{Z}}_t$ and the t-th atom $\overrightarrow{\mathcal{Y}}_t$.
3: Compute coefficient and error tensors (See Algorithm 2): $\{\overrightarrow{\mathcal{V}}_t, \overrightarrow{\mathcal{E}}_t\} = \arg\min_{\overrightarrow{\mathcal{V}}, \overrightarrow{\mathcal{E}}} \widetilde{\ell}(\mathcal{D}_{t-1}, \overrightarrow{\mathcal{V}}, \overrightarrow{\mathcal{E}}; \overrightarrow{\mathcal{Z}}_t)$.
4: Compute the coefficients $\overrightarrow{\mathcal{U}}_t$ by minimizing $\widetilde{\ell}_2(\overrightarrow{\mathcal{Y}}_t, \mathcal{D}_{t-1}, \mathcal{M}_{t-1}, \overrightarrow{\mathcal{U}})$.
5: Update the accumulation tensors:

$$\mathcal{M}_t = \mathcal{M}_{t-1} + \overrightarrow{\mathcal{Y}}_t * \overrightarrow{\mathcal{U}}_t^T,$$
$$\mathcal{A}_t = \mathcal{A}_{t-1} + \overrightarrow{\mathcal{V}}_t * \overrightarrow{\mathcal{V}}_t^T,$$
$$\mathcal{B}_t = \mathcal{B}_{t-1} + (\overrightarrow{\mathcal{Z}}_t - \overrightarrow{\mathcal{E}}_t) * \overrightarrow{\mathcal{V}}_t^T.$$

6: Update the basis $\mathcal{D}_t$ (See Algorithm 3):

$$\widehat{\mathbf{D}}_t = \arg\min_{\widehat{\mathbf{D}}} \operatorname{tr}\big(\widehat{\mathbf{D}}^T \widehat{\mathbf{D}}(\frac{\lambda_1}{2}\widehat{\mathbf{A}}_t + \frac{\lambda_3}{2}\mathbf{I})\big) - \operatorname{tr}\big(\widehat{\mathbf{D}}^T(\lambda_1 \widehat{\mathbf{B}}_t + \lambda_3 \widehat{\mathbf{M}}_t)\big).$$

7: **end for**
Output: Optimal basis $\mathcal{D}_N$.

t-th lateral slice of tensors $\mathcal{Z}$, $\mathcal{Y}$, $\mathcal{E}$, $\mathcal{U}^T$ and $\mathcal{V}^T$, respectively. We define

$$\widetilde{\ell}(\mathcal{D}, \overrightarrow{\mathcal{V}}, \overrightarrow{\mathcal{E}}; \overrightarrow{\mathcal{Z}}) \stackrel{\text{def}}{=} \frac{\lambda_1}{2}\|\overrightarrow{\mathcal{Z}} - \mathcal{D} * \overrightarrow{\mathcal{V}} - \overrightarrow{\mathcal{E}}\|_F^2 + \frac{n_3}{2}\|\overrightarrow{\mathcal{V}}\|_F^2 + \lambda_2\|\overrightarrow{\mathcal{E}}\|_1,$$
$$\ell(\mathcal{D}; \overrightarrow{\mathcal{Z}}) \stackrel{\text{def}}{=} \min_{\overrightarrow{\mathcal{V}}, \overrightarrow{\mathcal{E}}} \widetilde{\ell}(\mathcal{D}, \overrightarrow{\mathcal{V}}, \overrightarrow{\mathcal{E}}; \overrightarrow{\mathcal{Z}}),$$
$$\widetilde{h}(\mathcal{D}, \mathcal{U}; \mathcal{Y}) \stackrel{\text{def}}{=} \sum_{t=1}^{N} \frac{n_3}{2}\|\overrightarrow{\mathcal{U}}_t\|_F^2 + \frac{\lambda_3}{2}\|\mathcal{D} - \sum_{t=1}^{N} \overrightarrow{\mathcal{Y}}_t * \overrightarrow{\mathcal{U}}_t^T\|_F^2,$$
$$h(\mathcal{D}; \mathcal{Y}) \stackrel{\text{def}}{=} \min_{\mathcal{U}} \widetilde{h}(\mathcal{D}, \mathcal{U}; \mathcal{Y}).$$

These definitions allow us to rewrite (4) as

$$\min_{\mathcal{D}} \min_{\mathcal{U}, \mathcal{V}, \mathcal{E}} \sum_{t=1}^{N} \widetilde{\ell}(\mathcal{D}, \overrightarrow{\mathcal{V}}_t, \overrightarrow{\mathcal{E}}_t; \overrightarrow{\mathcal{Z}}_t) + \widetilde{h}(\mathcal{D}, \mathcal{U}; \mathcal{Y}), \tag{5}$$

which amounts to minimizing the empirical loss function:

$$f_N(\mathcal{D}) \stackrel{\text{def}}{=} \frac{1}{N}\sum_{t=1}^{N} \ell(\mathcal{D}; \overrightarrow{\mathcal{Z}}_t) + \frac{1}{N} h(\mathcal{D}; \mathcal{Y}). \tag{6}$$

We summarize our online low-rank tensor subspace clustering approach in Algorithm 1. The key idea of our algorithm is that at each iteration t, we minimize the loss function with respect to $\overrightarrow{\mathcal{V}}_t$, $\overrightarrow{\mathcal{E}}_t$ and $\overrightarrow{\mathcal{U}}_t$ given the previous $\mathcal{D}_{t-1}$, and then update the dictionary $\mathcal{D}_t$ by minimizing the cumulative loss.

Optimize $\overrightarrow{\mathcal{V}}$ and $\overrightarrow{\mathcal{E}}$: Given $\mathcal{D}_{t-1}$ in the previous iteration, we obtain the optimal solution $\{\overrightarrow{\mathcal{V}}_t, \overrightarrow{\mathcal{E}}_t\}$ by solving

$$\{\overrightarrow{\mathcal{V}}_t, \overrightarrow{\mathcal{E}}_t\} = \arg\min_{\overrightarrow{\mathcal{V}}, \overrightarrow{\mathcal{E}}} \frac{\lambda_1}{2}\|\overrightarrow{\mathcal{Z}}_t - \mathcal{D}_{t-1} * \overrightarrow{\mathcal{V}} - \overrightarrow{\mathcal{E}}\|_F^2 + \frac{n_3}{2}\|\overrightarrow{\mathcal{V}}\|_F^2 + \lambda_2\|\overrightarrow{\mathcal{E}}\|_1. \tag{7}$$

Both $\overrightarrow{\mathcal{V}}_t$ and $\overrightarrow{\mathcal{E}}_t$ can be optimized using the coordinate descent algorithm [19] as described in Algorithm 2. In step 8 of Algorithm 2, $S_\epsilon[\cdot]$ is the soft-thresholding operator [20]:

$$S_\epsilon[x] = \begin{cases} x - \epsilon & \text{if } x > \epsilon, \\ x + \epsilon & \text{if } x < -\epsilon, \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 2 Updating $\mathcal{V}$ and $\mathcal{E}$

Input: $\mathcal{D} \in \mathbb{R}^{n_1 \times d \times n_3}$, $\overrightarrow{\mathcal{Z}} \in \mathbb{R}^{n_1 \times 1 \times n_3}$, parameters λ_1 and λ_2.
Initialize: $\overrightarrow{\mathcal{E}} = 0$.

1: $\widehat{\mathcal{D}} = \text{fft}(\mathcal{D}, [\,], 3)$, $\widehat{\overrightarrow{\mathcal{Z}}} = \text{fft}(\overrightarrow{\mathcal{Z}}, [\,], 3)$.
2: **while** not converged **do**
3: $\widehat{\overrightarrow{\mathcal{E}}} = \text{fft}(\overrightarrow{\mathcal{E}}, [\,], 3)$.
4: **for** $k = 1, 2, \ldots, n_3$ **do**
5: $\widehat{\overrightarrow{\mathbf{V}}}^{(k)} = \big((\widehat{\mathbf{D}}^{(k)})^T \widehat{\mathbf{D}}^{(k)} + \frac{n_3}{\lambda_1}\mathbf{I}\big)^{-1} (\widehat{\mathbf{D}}^{(k)})^T (\widehat{\overrightarrow{\mathbf{Z}}}^{(k)} - \widehat{\overrightarrow{\mathbf{E}}}^{(k)})$.
6: **end for**
7: $\overrightarrow{\mathcal{V}} = \text{ifft}(\widehat{\overrightarrow{\mathcal{V}}}, [\,], 3)$.
8: $\overrightarrow{\mathcal{E}} = S_{\lambda_2/\lambda_1}[\overrightarrow{\mathcal{Z}} - \mathcal{D} * \overrightarrow{\mathcal{V}}]$.
9: **end while**
Output: $\overrightarrow{\mathcal{V}}$ and $\overrightarrow{\mathcal{E}}$.

Optimize $\overrightarrow{\mathcal{U}}$: We now move onto the step of minimizing $\widetilde{h}(\mathcal{D}, \mathcal{U}; \mathcal{Y})$ with respect to $\mathcal{U}$. Unlike the scheme of optimizing $\overrightarrow{\mathcal{V}}_t$ and $\overrightarrow{\mathcal{E}}_t$, whose solutions only depend on the new sample, however, we need to load the entire dictionary $\mathcal{Y}$ in order to obtain the optimal $\overrightarrow{\mathcal{U}}_t$. In order to circumvent this problem, we define the accumulation tensor $\mathcal{M}_{t-1} = \sum_{i=1}^{t-1} \overrightarrow{\mathcal{Y}}_i * \overrightarrow{\mathcal{U}}_i^T$ and solve $\overrightarrow{\mathcal{U}}_t$ by minimizing the following proxy function

$$\widetilde{\ell}_2(\overrightarrow{\mathcal{Y}}_t, \mathcal{D}_{t-1}, \mathcal{M}_{t-1}, \overrightarrow{\mathcal{U}}_t) \stackrel{\text{def}}{=} \frac{n_3}{2}\|\overrightarrow{\mathcal{U}}_t\|_F^2 + \frac{\lambda_3}{2}\|\mathcal{D}_{t-1} - \mathcal{M}_{t-1} - \overrightarrow{\mathcal{Y}}_t * \overrightarrow{\mathcal{U}}_t^T\|_F^2. \tag{8}$$

Using the fact that $\|\overrightarrow{\mathcal{U}}_t\|_F^2 = \|\overrightarrow{\mathcal{U}}_t^T\|_F^2$, we define $\overrightarrow{\mathcal{W}} = \overrightarrow{\mathcal{U}}_t^T$ and transform (8) into the Fourier domain to obtain

$$\min_{\widehat{\overrightarrow{\mathcal{W}}}} \frac{n_3}{2}\|\widehat{\overrightarrow{\mathcal{W}}}\|_F^2 + \frac{\lambda_3}{2}\|\widehat{\mathcal{D}}_{t-1} - \widehat{\mathcal{M}}_{t-1} - \widehat{\overrightarrow{\mathcal{Y}}}_t \otimes \widehat{\overrightarrow{\mathcal{W}}}\|_F^2, \tag{9}$$

where $\otimes$ denotes slicewise multiplication. We break (9) into n_3 problems and each frontal slice of $\widehat{\overrightarrow{\mathcal{W}}}$ has a closed-form solution

$$\widehat{\overrightarrow{\mathbf{W}}}^{(k)} = \big(\|\widehat{\overrightarrow{\mathbf{Y}}}_t^{(k)}\|_2^2 + n_3/\lambda_3\big)^{-1} (\widehat{\mathbf{D}}_{t-1}^{(k)} - \widehat{\mathbf{M}}_{t-1}^{(k)})^T \widehat{\overrightarrow{\mathbf{Y}}}_t^{(k)},$$

where $\widehat{\overrightarrow{\mathbf{W}}}^{(k)}$ and $\widehat{\overrightarrow{\mathbf{Y}}}_t^{(k)}$ denote the k-th frontal slices of $\widehat{\overrightarrow{\mathcal{W}}}$ and $\widehat{\overrightarrow{\mathcal{Y}}}_t$, respectively. After solving each frontal slice of $\widehat{\overrightarrow{\mathcal{W}}}$, we can get the solution of $\overrightarrow{\mathcal{W}}$ as $\overrightarrow{\mathcal{W}} = \text{ifft}(\widehat{\overrightarrow{\mathcal{W}}}, [\,], 3)$ and then set $\overrightarrow{\mathcal{U}}_t = \overrightarrow{\mathcal{W}}^T$. We

Table 1. Clustering accuracy and running time (in seconds) on COIL-20 and Weizmann datasets.

Data	Metric	OLRTSC	OLRTSC-F	TORPCA	TLRR	SSmC	OLRSC	OLRSC-F	ORPCA	LRR	SSC
COIL-20	ACC	0.7118	**0.8083**	0.2181	0.6937	0.6722	0.6736	0.7299	0.6819	0.6799	0.7764
	Time	78.88	78.88	64.67	64.02	457.5	11.90	11.90	**11.71**	85.81	15.41
Weizmann	ACC	0.5833	**0.8633**	0.3400	0.7867	0.7733	0.3633	0.3933	0.7167	0.5967	0.5567
	Time	26.44	26.44	22.96	47.11	16.68	3.62	3.62	3.60	10.12	**1.50**

update only $\overrightarrow{\mathcal{U}}_t$ at a time while keeping the other $\overrightarrow{\mathcal{U}}_i$'s ($i < t$) fixed and each $\overrightarrow{\mathcal{U}}_t$ is sequentially updated only when the t-th atom $\overrightarrow{\mathcal{Y}}_t$ is observed. Hence, our minimization strategy can be seen as a one-pass block coordinate descent algorithm for the function $\widetilde{h}(\mathcal{D}, \mathcal{U}; \mathcal{Y})$.

Optimize $\mathcal{D}$: When $\{\overrightarrow{\mathcal{V}}_i, \overrightarrow{\mathcal{E}}_i, \overrightarrow{\mathcal{U}}_i\}_{i=1}^t$ are readily available to us, the update of $\mathcal{D}_t$ is equivalent to solving

$$\begin{aligned}\mathcal{D}_t &= \arg\min_{\mathcal{D}} \sum_{i=1}^{t} \frac{\lambda_1}{2}\|\overrightarrow{\mathcal{Z}}_i - \mathcal{D} * \overrightarrow{\mathcal{V}}_i - \overrightarrow{\mathcal{E}}_i\|_F^2 + \frac{\lambda_3}{2}\|\mathcal{D} - \mathcal{M}_t\|_F^2 \\ &= \arg\min_{\mathcal{D}} \frac{\lambda_1}{2}\|\mathcal{Z}_t - \mathcal{D} * \mathcal{V}_t - \mathcal{E}_t\|_F^2 + \frac{\lambda_3}{2}\|\mathcal{D} - \mathcal{M}_t\|_F^2,\end{aligned} \tag{10}$$

where $\mathcal{Z}_t = [\overrightarrow{\mathcal{Z}}_1, \ldots, \overrightarrow{\mathcal{Z}}_t] \in \mathbb{R}^{n_1 \times t \times n_3}$, $\mathcal{V}_t = [\overrightarrow{\mathcal{V}}_1, \ldots, \overrightarrow{\mathcal{V}}_t] \in \mathbb{R}^{d \times t \times n_3}$ and $\mathcal{E}_t = [\overrightarrow{\mathcal{E}}_1, \ldots, \overrightarrow{\mathcal{E}}_t] \in \mathbb{R}^{n_1 \times t \times n_3}$. Let $\widehat{\mathbf{D}}_t = \text{bdiag}(\widehat{\mathcal{D}}_t)$, then (10) can again be transformed into the Fourier domain as

$$\begin{aligned}\widehat{\mathbf{D}}_t &= \arg\min_{\widehat{\mathbf{D}}} \frac{\lambda_1}{2}\|\widehat{\mathbf{Z}}_t - \widehat{\mathbf{D}}\widehat{\mathbf{V}}_t - \widehat{\mathbf{E}}_t\|_F^2 + \frac{\lambda_3}{2}\|\widehat{\mathbf{D}} - \widehat{\mathbf{M}}_t\|_F^2 \\ &= \arg\min_{\widehat{\mathbf{D}}} \frac{\lambda_1}{2}\text{tr}\big((\widehat{\mathbf{Z}}_t - \widehat{\mathbf{D}}\widehat{\mathbf{V}}_t - \widehat{\mathbf{E}}_t)^T(\widehat{\mathbf{Z}}_t - \widehat{\mathbf{D}}\widehat{\mathbf{V}}_t - \widehat{\mathbf{E}}_t)\big) \\ &\qquad + \frac{\lambda_3}{2}\text{tr}\big((\widehat{\mathbf{D}} - \widehat{\mathbf{M}}_t)^T(\widehat{\mathbf{D}} - \widehat{\mathbf{M}}_t)\big) \\ &= \arg\min_{\widehat{\mathbf{D}}} \text{tr}\big(\widehat{\mathbf{D}}^T\widehat{\mathbf{D}}(\frac{\lambda_1}{2}\widehat{\mathbf{V}}_t\widehat{\mathbf{V}}_t^T + \frac{\lambda_3}{2}\mathbf{I})\big) \\ &\qquad - \text{tr}\big(\widehat{\mathbf{D}}^T(\lambda_1(\widehat{\mathbf{Z}}_t - \widehat{\mathbf{E}}_t)\widehat{\mathbf{V}}_t^T + \lambda_3\widehat{\mathbf{M}}_t)\big) \\ &= \arg\min_{\widehat{\mathbf{D}}} \text{tr}\big(\widehat{\mathbf{D}}^T\widehat{\mathbf{D}}(\frac{\lambda_1}{2}\widehat{\mathbf{A}}_t + \frac{\lambda_3}{2}\mathbf{I})\big) \\ &\qquad - \text{tr}\big(\widehat{\mathbf{D}}^T(\lambda_1\widehat{\mathbf{B}}_t + \lambda_3\widehat{\mathbf{M}}_t)\big) \\ &= (\lambda_1\widehat{\mathbf{B}}_t + \lambda_3\widehat{\mathbf{M}}_t)(\lambda_1\widehat{\mathbf{A}}_t + \lambda_3\mathbf{I})^{-1},\end{aligned} \tag{11}$$

where $\mathcal{A}_t = \sum_{i=1}^t \overrightarrow{\mathcal{V}}_i * \overrightarrow{\mathcal{V}}_i^T$, $\mathcal{B}_t = \sum_{i=1}^t (\overrightarrow{\mathcal{Z}}_i - \overrightarrow{\mathcal{E}}_i) * \overrightarrow{\mathcal{V}}_i^T$ and $\text{tr}(\cdot)$ denotes the trace operation. We use block-coordinate descent [19] in the Fourier domain to update the dictionary due to its ease of computation. See more details in Algorithm 3.

3.3. Complexity Analysis

Memory Cost. First, notice that we need to load $\mathcal{D}_{t-1}, \mathcal{M}_{t-1} \in \mathbb{R}^{n_1 \times d \times n_3}$ and one sample $\overrightarrow{\mathcal{Z}}_t$ into the memory to obtain $\{\overrightarrow{\mathcal{V}}_t, \overrightarrow{\mathcal{E}}_t, \overrightarrow{\mathcal{U}}_t\}$, which costs $\mathcal{O}(n_1 d n_3)$. Second, all the past information required to update $\mathcal{D}_t$ has been stored in the accumulation tensors $\mathcal{A}_t$, $\mathcal{B}_t$ and $\mathcal{M}_t$, which only costs at most $\mathcal{O}(n_1 d n_3)$. Hence, the memory cost of OLRTSC is $\mathcal{O}(n_1 d n_3)$ per iteration.

Computational Complexity. In terms of computational complexity of the algorithm in each iteration, we first notice that it is computationally efficient to compute $\{\overrightarrow{\mathcal{V}}_t, \overrightarrow{\mathcal{E}}_t\}$ as one can use block coordinate descent method in [21] which enjoys linear convergence. Next, $\overrightarrow{\mathcal{U}}_t$ requires $\mathcal{O}(n_1 d n_3 + n_1 n_3 \log n_3)$ operations to conduct simple matrix-vector multiplication and the Fast Fourier Transform (FFT). Moreover, it is easy to verify that the complexity of Step 5 of Algorithm 1 is $\mathcal{O}(n_1 d n_3 + n_1 n_3 \log n_3)$. Finally, computing $\mathcal{D}_t$ costs $\mathcal{O}(n_1 n_3 d^2)$ as we only need to update $\widehat{\mathbf{D}}_t$ in practice.

Algorithm 3 Updating $\mathcal{D}$

Input: $\mathcal{D} \in \mathbb{R}^{n_1 \times d \times n_3}$ in the previous iteration, accumulation tensors $\mathcal{M}$, $\mathcal{A}$ and $\mathcal{B}$, and parameters λ_1 and λ_3.

1: $\widehat{\mathcal{D}} = \text{fft}(\mathcal{D}, [\,], 3)$.
2: $\widehat{\mathcal{M}} = \text{fft}(\mathcal{M}, [\,], 3)$, $\widehat{\mathcal{A}} = \text{fft}(\mathcal{A}, [\,], 3)$, and $\widehat{\mathcal{B}} = \text{fft}(\mathcal{B}, [\,], 3)$.
3: **for** $k = 1, 2, \ldots, n_3$ **do**
4: $\quad\widetilde{\mathbf{A}} = \lambda_1\widehat{\mathbf{A}}^{(k)} + \lambda_3\mathbf{I}$ and $\widetilde{\mathbf{B}} = \lambda_1\widehat{\mathbf{B}}^{(k)} + \lambda_3\widehat{\mathbf{M}}^{(k)}$.
5: $\quad$**for** $j = 1, 2, \ldots, d$ **do**
6: $\quad\quad\widehat{\mathcal{D}}(:, j, k) = \widehat{\mathcal{D}}(:, j, k) - \frac{1}{\widetilde{\mathbf{A}}_{j,j}}(\widehat{\mathbf{D}}^{(k)}\widetilde{\mathbf{a}}_j - \widetilde{\mathbf{b}}_j)$.
7: $\quad$**end for**
8: **end for**

Output: $\mathcal{D} = \text{ifft}(\widehat{\mathcal{D}}, [\,], 3)$.

3.4. A Fully Online Scheme

So far, we have discussed an algorithm to optimize the TLRR problem (1) in an online fashion. Similar to [8], we use the dataset itself as the dictionary $\mathcal{Y}$ in practice. For clustering purpose, we first collect all the $\overrightarrow{\mathcal{U}}_t$'s and $\overrightarrow{\mathcal{V}}_t$'s and compute the representation tensor $\mathcal{X} = \mathcal{U} * \mathcal{V}^T$, then apply spectral clustering [7] on an affinity matrix $\mathbf{\Theta}$ whose (i, j)-th entry $\mathbf{\Theta}_{i,j} = \|\mathcal{X}(i, j, :)\|_1 + \|\mathcal{X}(j, i, :)\|_1$ to obtain final clustering. The memory cost in this case is $\mathcal{O}(N^2 n_3)$. Alternatively, as suggested in [8], we can also define $\mathbf{v}_t = \sum_{k=1}^{n_3} \overrightarrow{\mathbf{V}}_t^{(k)}$ for $t = 1, \ldots, N$ and utilize the well-known k-means clustering on $\mathbf{v}_t$'s. Thus a fully online implementation can be achieved with only $\mathcal{O}(Ld)$ memory usage.

4. EXPERIMENTAL RESULTS

In this section, we conduct experiments to evaluate the performance of our proposed approach.

4.1. Datasets

Three widely used image clustering benchmark datasets are selected in our experiments.

The *COIL-20 dataset*[1] consists of 1440 grayscale images of 20 objects. Each object has 72 images, which are taken with poses varying at an interval of 5 degrees. We downsample the images to 32×32 pixels.

[1] https://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php

Table 2. Clustering accuracy and running time (in seconds) on USPS dataset.

N	Metric	OLRTSC	OLRTSC-F	TORPCA	TLRR	SSmC	OLRSC	OLRSC-F	ORPCA	LRR	SSC
1000	ACC	**0.7033**	0.4262	0.5968	0.3899	0.5595	0.4157	0.5034	0.5484	0.4462	0.4360
	Time	15.10	15.10	14.31	9.84	144	2.25	2.25	**2.20**	7.40	3.45
2000	ACC	**0.6781**	0.3961	0.6367	0.3802	0.5451	0.4100	0.4902	0.5640	0.3909	0.3953
	Time	27.65	27.65	25.26	19.42	702.7	**4.42**	**4.42**	4.68	10.35	23.34
3000	ACC	**0.6538**	0.4015	0.6246	0.3756	0.5448	0.4169	0.5071	0.5756	0.3773	0.3802
	Time	39.49	39.49	34.65	32.61	1667.9	**6.79**	**6.79**	6.98	12.84	59.92
4000	ACC	0.6467	0.4033	**0.6911**	0.3678	0.5474	0.4148	0.5083	0.5889	0.3680	0.3951
	Time	53.90	53.90	52.25	42.34	3538.7	**8.97**	**8.97**	9.17	15.57	106.7

The *Weizmann face database*[2] contains 512×352 grayscale images of 28 individuals, where each subject has 30 frontal images with variations of pose, illumination and expression. We select a subset of this dataset composed of images from subjects 11–20, resulting in $N = 300$. We downsample the images by a factor of 8 along each axis and crop them into 60×40 pixels.

The *USPS dataset* [22] consists of 9298 images of 10 handwritten digits, and each image has 16×16 pixels. In each experiment, $N_\ell \in \{100, 200, 300, 400\}$ images are randomly selected from the training set for each of the 10 digits; hence $N \in \{1000, 2000, 3000, 4000\}$. For each image, we randomly shift the digit horizontally with respect to the center by 3 pixels either side (left or right). We run experiments 20 times for each N, and average results are reported.

4.2. Compared Methods

We compare the performance of OLRTSC with several state-of-the-art online learning methods and batch algorithms, including tensor online robust PCA (TORPCA) [18], online low-rank subspace clustering (OLRSC) [8], and online robust PCA (ORPCA) [23], TLRR [15], SSmC [13], LRR [5], and SSC [6]. The OLRTSC (resp., OLRSC) followed by k-means clustering is denoted by OLRTSC-F (resp., OLRSC-F). For TORPCA, we perform spectral clustering on $\mathbf{\Theta} = \sum_{k=1}^{n_3} \mathbf{A}^{(k)}$ with $\mathcal{A} = \mathcal{R}_0 * \mathcal{R}_0^T$, where $\mathcal{R}_0$ is from the t-SVD of $\mathcal{Z}_0 = \mathcal{L}_0 * \mathcal{S}_0 * \mathcal{R}_0^T$ and $\mathcal{Z}_0$ is the clean tensor recovered by TORPCA. For ORPCA, we perform spectral clustering on $\mathbf{R}_0\mathbf{R}_0^T$, where $\mathbf{R}_0$ is the row space of $\mathbf{Z}_0 = \mathbf{L}_0\mathbf{\Sigma}_0\mathbf{R}_0^T$ and $\mathbf{Z}_0$ is the clean matrix recovered by ORPCA.

4.3. Parameter Settings

We set $\lambda_1 = 1$, $\lambda_2 = 1/\sqrt{n_1 n_3}$ and $\lambda_3 = \sqrt{t/n_1 n_3}$, where t is the iteration counter. Here, we use $d = \min(3L, n_1)$ as an upper bound of the tensor tubal rank of $\mathcal{X}$, where L is the number of classes. We follow the default parameter settings for the baselines.

4.4. Comparative Results

The results are reported in Tables 1 and 2. We run our algorithm and TORPCA with only 1 epoch. For fair comparison, the running time of spectral clustering or k-means is excluded so we can focus on comparing the efficiency of the algorithms themselves. We can make the following observations: 1) OLRTSC (or OLRTSC-F) achieves the best performance in most cases on all three datasets in terms of clustering accuracy. 2) Our method is able to identify the samples at a superior rate compared to TLRR. 3) Compared to TORPCA, OLRTSC takes comparable running time but shows noticeable improvement in accuracy. 4) The k-means alternative (OLRTSC-F) outperforms the spectral clustering counterpart on COIL-20 and Weizmann datasets, which makes OLRTSC more robust and scalable.

[2] http://www.wisdom.weizmann.ac.il/ /vision/FaceBase/

5. CONCLUSION

In this paper, we proposed a novel extension of the tensor low-rank representation, termed online low-rank tensor subspace clustering (OLRTSC), for streaming tensor data clustering. Experimental results on three real-world datasets showed the usefulness of our algorithm and its superiority over the state-of-the-art clustering algorithms. Our future work includes theoretical analysis of the performance and convergence behavior of the algorithm.

6. REFERENCES

[1] R. Vidal, "Subspace clustering," *IEEE Signal Process. Mag.*, vol. 28, no. 2, pp. 52–68, Mar. 2011.

[2] J. Ho, M.-H. Yang, J. Lim, K.-C. Lee, and D. Kriegman, "Clustering appearances of objects under varying illumination conditions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2003, pp. 11–18.

[3] R. Vidal, Y. Ma, and S. Sastry, "Generalized principal component analysis (GPCA)," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 12, pp. 1945–1959, Dec. 2005.

[4] G. Chen and G. Lerman, "Spectral curvature clustering (SCC)," *Int. J. Comput. Vis.*, vol. 81, pp. 317–330, 2009.

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

[6] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2765–2781, Nov. 2013.

[7] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[8] J. Shen, P. Li, and H. Xu, "Online low-rank subspace clustering by basis dictionary pursuit," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 622–631.

[9] M. E. Kilmer, K. Braman, N. Hao, and R. C. Hoover, "Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging," *SIAM J. Matrix Anal. Appl.*, vol. 34, no. 1, pp. 148–172, Feb. 2013.

[10] Z. Zhang, G. Ely, S. Aeron, N. Hao, and M. Kilmer, "Novel methods for multilinear data completion and de-noising based on tensor-SVD," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 3842–3849.

[11] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, "Tensor robust principal component analysis with a new tensor nuclear norm," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 925–938, Apr. 2020.

[12] M. E. Kilmer and C. D. Martin, "Factorization strategies for third-order tensors," *Linear Algebra Appl.*, vol. 435, no. 3, pp. 641–658, Aug. 2011.

[13] E. Kernfeld, S. Aeron, and M. Kilmer, "Clustering multi-way data: A novel algebraic approach," *arXiv preprint*, 2014.

[14] T. Wu and W. U. Bajwa, "A low tensor-rank representation approach for clustering of imaging data," *IEEE Signal Process. Lett.*, vol. 25, no. 8, pp. 1196–1200, Aug. 2018.

[15] P. Zhou, C. Lu, J. Feng, Z. Lin, and S. Yan, "Tensor low-rank representation for data recovery and clustering," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019, doi: 10.1109/TPAMI.2019.2954874.

[16] T. Wu, "Clustering-aware structure-constrained low-rank submodule clustering," in *Proc. 53rd Asilomar Conf. Signals, Syst., Computers*, 2019, pp. 1852–1856.

[17] T. Wu, "Graph regularized low-rank representation for submodule clustering," *Pattern Recognit.*, vol. 100, pp. 107145, Apr. 2020.

[18] Z. Zhang, D. Liu, S. Aeron, and A. Vetro, "An online tensor robust PCA algorithm for sequential 2D data," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2016, pp. 2434–2438.

[19] D. P. Bertsekas, *Nonlinear programming*, Athena Scientific, 1999.

[20] D. L. Donoho, "De-noising by soft-thresholding," *IEEE Trans. Inf. Theory*, vol. 41, no. 3, pp. 613–627, 1995.

[21] P. Richtárik and M. Takác, "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function," *Math. Program.*, vol. 144, no. 1-2, pp. 1–38, 2014.

[22] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 5, no. 16, pp. 550–554, May 1994.

[23] J. Feng, H. Xu, and S. Yan, "Online robust PCA via stochastic optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 404–412.