Tucker-Regularized Tensor Bregman Co-clustering

Pedro A. Forero and Paul A. Baxley
Naval Information Warfare Center Pacific, San Diego, CA, USA
{pedro.a.forero; paul.baxley}@navy.mil

***Abstract*—Co-clustering of tensor data is an unsupervised learning task aiming to identify multidimensional structures hidden in a tensor. These structures are critical for understanding interdependencies across variables belonging to different tensor dimensions, often referred to as modes, which are frequently disregarded when tensor data are represented via one- or two-dimensional data structures. This work proposes a new tensor co-clustering algorithm that uses a class of Bregman divergences to measure the coherence of co-clusters on an individual mode basis, while ensuring that the interactions of their prototyping elements capture the tensor intra-modal structure. A co-clustering algorithm based on the alternating-direction method of multipliers is developed. The proposed algorithm decouples the co-clustering problem into an iterative two-step process whose steps are reminiscent of classical one-way clustering and Tucker decomposition problems. The performance of the proposed method is illustrated via numerical tests.**

***Index Terms*—Tucker decomposition, Tensor co-clustering, Bregman co-clustering**

I. Introduction

Clustering is a fundamental unsupervised-learning task that seeks to identify hidden grouping structure present in data. Classical clustering algorithms group data-objects according to a similarity measure that compares them using all features that define them. The increased availability of multiway array data, also known as *tensor* data, across a variety of fields has drawn increased interest for computationally-efficient techniques that can unravel their hidden structure [1]. It is natural to consider each of the tensor dimensions, i.e., modes, individually and apply classical clustering algorithms to partition the tensor data separately per mode, thereby identifying its intra-mode structure. This general approach fails to capture the multidimensional nature of the tensor and is unable to identify inter-mode structures that in many cases can provide additional insight into the hidden structure present in the tensor.

Co-clustering offers an alternative for unraveling tensor structure by jointly partitioning tensors across all their modes, thereby identifying sub-tensors as clusters. In the context of order-2 tensors, i.e., matrices, the co-clustering problem, often referred to as bi-clustering, seeks to jointly group rows and columns that have similar structure across a subset of their columns and rows, respectively [2]. Co-clusters can identify brain spatiotemporal activations related to physical stimuli or gene activations, joint spectra-temporal structure related to spatially-distributed acoustic sources, and advertisement-website pairing effects on consumer behaviors [1].

Several approaches have recently been proposed for co-clustering tensor data, often using classical clustering methods in combination with tensor factorization models. Spectral clustering techniques have been extended for co-clustering tensors that model hyper-graph structures [3], [4]. Clustering algorithms such as K-means have been used for partitioning a tensor by clustering the tensor fibers along each mode separately [5], and by clustering the rows of the factor matrix obtained from the canonical polyadic decomposition (CPD) of the tensor along the dimension of interest [6]. Tensor factorization methods based on the sparse CPD and the high-order singular value decomposition have also been used for tensor co-clustering [7], [8].

This work proposes a tensor co-clustering algorithm that partitions the tensor fibers along each tensor mode while jointly constructing a Tucker model that approximates the tensor. While clustering the tensor fibers, Bregman divergences are used to quantify similarity among fibers assigned to the same cluster via cluster-prototype elements [9]. These cluster-prototype elements are assumed to define the factor matrices associated with the Tucker model that approximates the tensor. Not only they capture the intra-mode structure individually, but when combined they also define the inter-mode structure that characterizes the hidden structure of the tensor. Although the Tucker decomposition is non-unique, it offers additional degrees of freedom when compared with the CPD while still being able to accommodate sparse and orthonormal structure in its components [1], [10]. Thus, it is considered more expressive and able to better characterize co-cluster structure. The generality of the proposed method stems from its ability to accommodate a class of Bregman divergences and Bregman-divergence approximations that can be individually adapted to the characteristics of the tensor fibers per mode. This work extends [11], which considered a related co-clustering method using the Euclidean distance, a type of Bregman divergence, as a cluster dissimilarity metric. A solver based on the alternating-directions method of multipliers (ADMM) is developed for solving the proposed co-clustering problem. Although iterative in nature, the proposed ADMM decomposes the co-clustering problem into updates that are reminiscent of classical clustering and Tucker-factorization problem formulations. Thus, techniques for solving such problems can be leveraged for solving the resulting ADMM updates. In our case, the ADMM updates are evaluated via a damped Gauss-Newton method and Nesterov's accelerated gradient-descent method.

The remainder of this paper is organized as follows. Section II introduces preliminary definitions. Section III presents the co-clustering problem. Section IV develops an ADMM-based solver for the proposed tensor co-clustering problem. Numerical tests illustrating the behavior of the propose algorithm are presented in Section V. The paper concludes in Section VI.

II. Preliminaries

A tensor $\underline{\mathbf{Y}} \in \mathbb{R}^{D_1 \times \cdots \times D_N}$ of order $N \in \mathbb{N}_{>0}$ is a multidimensional array with entries $[\underline{\mathbf{Y}}]_{d_1,\ldots,d_N} := y_{d_1,\ldots,d_N} \in \mathbb{R}$, where D_n denotes the dimensionality of the n-th mode of the tensor, and indices $d_n \in \{1,\ldots,D_N\}$, $\forall n$. A mode-n *fiber* is a one-dimensional array obtained from the $y_{d_1,\ldots,d_N}$'s by fixing all but the index d_n, and a *slice* is a two-dimensional array obtained from the $y_{d_1,\ldots,d_N}$'s by fixing all but two of their indices. *Matricization* is a tensor transformation that maps the entries of $\underline{\mathbf{Y}}$ into those of a matrix. In particular the mode-n matricization of $\underline{\mathbf{Y}}$ arranges all its mode-n fibers into the columns of $\mathbf{Y}_{(n)} \in \mathbb{R}^{D_n \times D_{-n}}$, where $D_{-n} := \prod_{m \neq n} D_m$. *Vectorization* of a matrix $\mathbf{Y} := [\mathbf{y}_1,\ldots,\mathbf{y}_M] \in \mathbb{R}^{D_n \times M}$ stacks all its columns into a vector $\text{vec}(\mathbf{Y}) := [\mathbf{y}_1',\ldots,\mathbf{y}_M']' \in \mathbb{R}^{D_n M}$, where $(\cdot)'$ denotes the *transpose* operation. The mode-n product

This work was funded by the Naval Innovative Science and Engineering program at the Naval Information Warfare Center Pacific.

between $\overline{\mathbf{Y}}$ and a matrix $\mathbf{U} \in \mathbb{R}^{Q \times D_n}$ yields a tensor $\overline{\mathbf{Y}} \times_n \mathbf{U} \in \mathbb{R}^{D_1 \times \ldots \times D_{n-1} \times Q \times D_{n+1} \times \ldots \times D_N}$ whose entries are given in terms of its mode-n matricization as $[\overline{\mathbf{Y}} \times_n \mathbf{U}]_{(n)} = \mathbf{U}\mathbf{Y}_{(n)} \in \mathbb{R}^{Q \times D_{-n}}$. For an N-order tensor $\overline{\mathbf{Y}}$, its high-order ℓ_p-norm is defined as $\|\overline{\mathbf{Y}}\|_p := (\sum_{d_1,\ldots,d_N} |y_{d_1,\ldots,d_N}|^p)^{1/p}$ for $p \in \{1, 2\}$.

An order-N rank-one tensor $\mathbf{u}_1 \circ \ldots \circ \mathbf{u}_N \in \mathbb{R}^{D_1 \times \ldots \times D_N}$ is defined as the outer product of vectors $\{\mathbf{u}_n \in \mathbb{R}^{D_n}\}_{n=1}^N$, whose $(d_1, \ldots, d_N)$ entry is given by $\prod_{n=1}^N [\mathbf{u}_n]_{d_n}$. With $0 \leq K_n \leq D_n$ for all n, the Tucker tensor decomposition models $\overline{\mathbf{Y}}$ as

$$\overline{\mathbf{Y}} = \sum_{k_1=1}^{K_1} \cdots \sum_{k_N=1}^{K_N} g_{k_1,\ldots,k_N} \left(\mathbf{u}_{1,k_1} \circ \ldots \circ \mathbf{u}_{N,k_N}\right) \tag{1}$$

where $g_{k_1,\ldots,k_N} := [\overline{\mathbf{G}}]_{k_1,\ldots,k_N}$, $\overline{\mathbf{G}} \in \mathbb{R}^{K_1 \times \ldots \times K_N}$ denotes the so-called *core* tensor, $\mathcal{U} := \{\mathbf{U}_1, \ldots, \mathbf{U}_N\}$ the set of factor matrices where $\mathbf{U}_n := [\mathbf{u}_{n,1} \ldots \mathbf{u}_{n,K_n}] \in \mathbb{R}^{D_n \times K_n}$, for $n \in \{1, \ldots, N\}$. Equation (1) can be written in terms of the mode-n product notation as $\overline{\mathbf{Y}} = \overline{\mathbf{G}} \times \{\mathbf{U}\}$, where $\overline{\mathbf{G}} \times \{\mathbf{U}\} := \overline{\mathbf{G}} \times_1 \mathbf{U}_1 \times_2 \ldots \times_N \mathbf{U}_N$. A detailed discussion about tensors can be found in [10].

A Bregman divergence measures the distance between a convex function and its linear approximation with respect to $\mathbf{x}$ at a point $\mathbf{y}$, and is formally defined as follows [9].

Definition 2.1 (Bregman Divergence): Let $\phi : \mathcal{Q} \to \mathbb{R}$ be a strictly convex and differentiable function on a convex set $\mathcal{Q} \subseteq \mathbb{R}^{D_n}$. The Bregman divergence $d_\phi : \mathcal{Q} \times \mathcal{Q} \to [0, \infty)$ is defined as

$$d_\phi(\mathbf{y}, \mathbf{x}) = \phi(\mathbf{y}) - \phi(\mathbf{x}) - (\mathbf{y} - \mathbf{x})' \nabla \phi(\mathbf{x})$$

where $\nabla \phi(\mathbf{x})$ denotes the gradient of ϕ evaluated at $\mathbf{x}$.

III. Problem Statement

Given an order-N tensor $\overline{\mathbf{Y}} \in \mathbb{R}^{D_1 \times \ldots \times D_N}$, the tensor co-clustering problem seeks to jointly partition all the elements of $\overline{\mathbf{Y}}$ into K groups that reflect their intra-mode structure while capturing their joint structure across modes. Capturing the mode-n structure can be done by clustering the columns of $\mathbf{Y}_{(n)}$. Let $K_n \in \mathbb{N}_{>0}$ define the number of clusters into which the mode-n fibers of $\overline{\mathbf{Y}}$ are grouped, and $\mathcal{K}_n := \{1, \ldots, K_n\}$. Since each entry of $\overline{\mathbf{Y}}$ is assigned to only one column in each $\mathbf{Y}_{(n)}$, $n = 1, \ldots, N$, it is possible to define an N-tuple of memberships $(k_1, \ldots, k_N) \in \mathcal{K}_1 \times \ldots \times \mathcal{K}_N$ for each entry of $\overline{\mathbf{Y}}$. After setting $K = \prod_{n=1}^N K_n$, that is, to the cardinality of $\mathcal{K}_1 \times \ldots \times \mathcal{K}_N$, one can assign all entries of $\overline{\mathbf{Y}}$ having the same N-tuple membership to one of the K co-clusters. Although the mapping of entries of $\overline{\mathbf{Y}}$ to columns of $\mathbf{Y}_{(n)}$ is not necessarily unique, the specific column assignment is not critical as long as it remains consistent when operating on the matricizations. One such option maps the $(d_1, \ldots, d_N)$ entry of $\overline{\mathbf{Y}}$ into the $(d_n, j_{(n)}^{(d_1,\ldots,d_N)})$ entry of $\mathbf{Y}_{(n)}$, where $j_{(n)}^{(d_1,\ldots,d_N)} = 1 + \sum_{p \neq n} (d_p - 1) J_p$ and

$$J_p = \begin{cases} 1 & (p = 1) \vee (p = 2 \wedge n = 1) \\ \prod_{m \neq n}^{p-1} D_m & \text{otherwise} \end{cases}. \tag{2}$$

Tensor co-clustering methods that seek to compute each entry of the membership tuple $(k_1, \ldots, k_N)$ independently fail to capture the interdependencies across modes embedded in the structure of $\overline{\mathbf{Y}}$ [5], [6]. The underlying structure of $\overline{\mathbf{Y}}$ can be revealed by constructing a Tucker model approximation of the form $\overline{\mathbf{Y}} = \overline{\mathbf{G}} \times \{\mathbf{U}\} + \overline{\mathbf{E}}$, where $\overline{\mathbf{E}} \in \mathbb{R}^{D_1 \times \ldots \times D_N}$ is a tensor capturing the mismatch between the Tucker model and $\overline{\mathbf{Y}}$. Nonzero entries of $\overline{\mathbf{G}}$ allow rank-one tensors $\mathbf{u}_{1,k_1} \circ \ldots \circ \mathbf{u}_{N,k_N}$ formed by column vectors of their corresponding factor matrices in $\mathcal{U}$ to contribute towards approximating $\overline{\mathbf{Y}}$.

Let $\mathbf{s}_{n,i} \in \{0, 1\}^{K_n}$ denote a membership assignment vector for the i-th column of $\mathbf{Y}_{(n)}$ satisfying that $\mathbf{s}'_{n,i} \mathbf{1}_{K_n} = 1$. The k-th entry of $\mathbf{s}_{n,i}$ is 1 if the i-th column of $\mathbf{Y}_{(n)}$ belongs to cluster $k \in \{1, \ldots, K_n\}$, and it is 0 otherwise. Thus, it defines a *hard* cluster-membership assignment. This work proposes finding estimates for the membership assignment matrices $\mathbf{S}_n := [\mathbf{s}_{n,1}, \ldots, \mathbf{s}_{n,D_{-n}}] \in \{0, 1\}^{K_n \times D_{-n}}$, for $n = 1, \ldots, N$ as the solution of

$$\begin{aligned} \min_{\substack{\overline{\mathbf{G}}, \{\mathbf{U}_n\}_{n=1}^N \\ \{\mathbf{S}_n \in \mathcal{S}_n\}_{n=1}^N}} & \sum_{n=1}^N \sum_{i=1}^{D_{-n}} \sum_{k=1}^{K_n} s_{n,i,k} d_{\phi_n}(\mathbf{u}_{n,k}, \mathbf{y}_{(n),i}) \\ & + \frac{\lambda}{2} \|\overline{\mathbf{Y}} - \overline{\mathbf{G}} \times \{\mathbf{U}\}\|_2^2 + \Omega(\overline{\mathbf{G}}, \mathcal{U}) \end{aligned} \tag{3}$$

where $\mathbf{y}_{(n),i}$ denotes the i-th column of $\mathbf{Y}_{(n)}$, $s_{n,i,k} := [\mathbf{s}_{n,i}]_k$, $\lambda > 0$ a tuning parameter, $\mathbf{U}_n \in \mathbb{R}^{D_n \times K_n}$ for $n = 1, \ldots, N$, $\overline{\mathbf{G}} \in \mathbb{R}^{K_1 \times \ldots \times K_N}$, $\mathcal{S}_n := \{\mathbf{S} \in \{0, 1\}^{K_n \times D_{-n}} : \mathbf{S}' \mathbf{1}_{K_n} = \mathbf{1}_{D_{-n}}\}$, d_{ϕ_n} a Bregman divergence defined by the convex function $\phi_n : \mathbb{R}^{D_n} \to \mathbb{R}$, and $\Omega(\overline{\mathbf{G}}, \mathcal{U}) := \mu \|\overline{\mathbf{G}}\|_1 + \sum_{n=1}^N \frac{\mu_n}{2} \|\mathbf{U}_n\|_2^2$ with $\{\mu, \{\mu_n\}_{n=1}^N > 0\}$ a set of tuning parameters. The first term in (3) aims to partition the columns of each $\mathbf{Y}_{(n)}$ into K_n non-overlapping clusters using the columns of $\mathbf{U}_n$ as cluster-center representatives. The desired partition minimizes the dissimilarity between the columns of a cluster center, say $\mathbf{u}_{n,k}$ and the columns of $\mathbf{Y}_{(n)}$ as defined by d_{ϕ_n}. Note that the unknown cluster centers appear in the first argument of d_{ϕ_n}. This judicious placement of the $\mathbf{u}_{n,k}$'s will enable the development of fast solvers for (3) in the ensuing section (cf. [9]). The second term in (3) seeks to obtain a Tucker model, namely $(\overline{\mathbf{G}}, \{\mathbf{U}_n\}_{n=1}^N)$, that serves as a good representative for $\overline{\mathbf{Y}}$. The regularizer Ω prevents (3) from yielding a trivial solution caused by the scaling ambiguity inherent to the Tucker model. Additionally, the ℓ_1-regularization on $\overline{\mathbf{G}}$ encourages the entries of $\overline{\mathbf{G}}$ to be zero as controlled by μ.

The solution of (3) yields the desired partition of tensor fibers and captures the tensor structure across modes by using the $\mathbf{U}_n$'s of the Tucker model as cluster centers. The dependency among clusters obtained per mode can be tuned via λ and the μ_n's. Solving (3) is challenging since it is a non-convex, non-differentiable and highly nonlinear optimization problem.

Remark 3.1 (On the Selection of d_{ϕ_n}): The choice of d_{ϕ_n} is driven by the type of data in $\overline{\mathbf{Y}}$. Problem (3) allows the use of different Bregman divergences for clustering tensor fibers across different modes. Although this work focuses on d_{ϕ_n} where the domain of ϕ_n, denoted $\mathcal{Q}_n$, is $\mathbb{R}^{D_n}$ to develop fast solvers for (3), it is possible to adapt the approach developed in the ensuing sections to accommodate a broader set of Bregman divergences with minimal modifications. For instance, one can approximate a ϕ_n as a piecewise linear function to model arbitrary Bregman divergences [12].

IV. Bregman Co-Clustering via ADMM

In order to develop a computationally tractable solver for (3), one can consider solving the following related problem

$$\begin{aligned} \min_{\substack{\overline{\mathbf{G}}, \{\mathbf{U}_n, \mathbf{Z}_n\}_{n=1}^N \\ \{\mathbf{S}_n \in \mathcal{S}_n\}_{n=1}^N}} & \sum_{n=1}^N \sum_{i=1}^{D_{-n}} \sum_{k=1}^{K_n} s_{n,i,k} d_{\phi_n}(\mathbf{z}_{n,k}, \mathbf{y}_{(n),i}) \\ & + \frac{\lambda}{2} \|\overline{\mathbf{Y}} - \overline{\mathbf{G}} \times \{\mathbf{U}\}\|_2^2 + \Omega(\overline{\mathbf{G}}, \mathcal{U}) \\ \text{Subj. to} \quad & \mathbf{z}_{n,k} = \mathbf{u}_{n,k} \quad \forall n, k \end{aligned} \tag{4}$$

where the constraints $\{\mathbf{z}_{n,k} = \mathbf{u}_{n,k}, \ \forall n, k\}$ guarantee that any feasible solution for (4) is also a feasible solution for (3). A computationally tractable solver for (4) based on ADMM is developed next. First, consider the augment Lagrangian for (4) given by

$$\begin{aligned}\mathcal{L}(\{\mathbf{Z}_n,\mathbf{U}_n,\mathbf{S}_n,\boldsymbol{\Lambda}_n\}_{n=1}^N,\overline{\mathbf{G}}) &= \sum_{n=1}^N\sum_{i=1}^{D_{-n}}\sum_{k=1}^{K_n} s_{n,i,k} d_{\phi_n}(\mathbf{z}_{n,k},\mathbf{y}_{(n),i})\\ &+ \frac{\lambda}{2}\|\overline{\mathbf{Y}}-\overline{\mathbf{G}}\times\{\mathbf{U}\}\|_2^2 + \Omega(\overline{\mathbf{G}},\mathcal{U})\\ &+ \sum_{n=1}^N\sum_{k=1}^{K_n}\boldsymbol{\lambda}'_{n,k}(\mathbf{z}_{n,k}-\mathbf{u}_{n,k}) + \sum_{n=1}^N \frac{\rho_n}{2}\|\mathbf{Z}_n-\mathbf{U}_n\|_2^2 \qquad (5)\end{aligned}$$

where $\mathbf{Z}_n := [\mathbf{z}_{n,1}\ldots\mathbf{z}_{n,K_n}] \in \mathbb{R}^{D_n\times K_n}$, $\boldsymbol{\Lambda}_n := [\boldsymbol{\lambda}_{n,1}\ldots\boldsymbol{\lambda}_{n,K_n}] \in \mathbb{R}^{D_n\times K_n}$, $\boldsymbol{\lambda}_{n,k}$ the Lagrange multiplier vector associated with the constraints $\mathbf{z}_{n,k}=\mathbf{u}_{n,k}$, and $\{\rho_n>0\}_{n=1}^N$ tuning parameters.

With $\tau\in\mathbb{N}_{>0}$ denoting an iteration index, the ADMM updates for solving (4) are given by

$$\begin{aligned}&(\overline{\mathbf{G}}^{[\tau]},\{\mathbf{U}_n^{[\tau]},\mathbf{S}_n^{[\tau]}\}_{n=1}^N) &(6a)\\ &\quad=\underset{\overline{\mathbf{G}},\{\mathbf{U}_n,\mathbf{S}_n\in\mathcal{S}_n\}_{n=1}^N}{\arg\min}\ \mathcal{L}(\{\mathbf{Z}_n^{[\tau-1]},\mathbf{U}_n,\mathbf{S}_n,\boldsymbol{\Lambda}_n^{[\tau-1]}\}_{n=1}^N,\overline{\mathbf{G}})\\ &(\{\mathbf{Z}_n^{[\tau]}\}_{n=1}^N) &(6b)\\ &\quad=\underset{\{\mathbf{Z}_n\}_{n=1}^N}{\arg\min}\ \mathcal{L}(\{\mathbf{Z}_n,\mathbf{U}_n^{[\tau]},\mathbf{S}_n^{[\tau]},\boldsymbol{\Lambda}_n^{[\tau-1]}\}_{n=1}^N,\overline{\mathbf{G}}^{[\tau]})\\ &\boldsymbol{\lambda}_{n,k}^{[\tau]}=\boldsymbol{\lambda}_{n,k}^{[\tau-1]}+\rho_n(\mathbf{z}_{n,k}^{[\tau]}-\mathbf{u}_{n,k}^{[\tau]}),\quad \forall n,k &(6c)\end{aligned}$$

where (6a) updates $\overline{\mathbf{G}}$, $\mathbf{U}_n$'s and $\mathbf{S}_n$'s with all other variables fixed, (6b) updates the $\mathbf{Z}_n$'s with all other variables fixed, and (6c) updates the $\boldsymbol{\lambda}_{n,k}$'s with all other variables fixed.

Solving (6a) decomposes into updating separately $(\overline{\mathbf{G}}^{[\tau]},\{\mathbf{U}_n^{[\tau]}\}_{n=1}^N)$ and $\{\mathbf{S}_n^{[\tau]}\}_{n=1}^N$. The updates $\{\mathbf{U}_n^{[\tau]}\}_{n=1}^N$ and $\overline{\mathbf{G}}^{[\tau]}$ are given by the solution of

$$\begin{aligned}\min_{\overline{\mathbf{G}},\{\mathbf{U}_n\}_{n=1}^N}\ &\frac{1}{2}\|\overline{\mathbf{Y}}-\overline{\mathbf{G}}\times\{\mathbf{U}\}\|_2^2+\frac{1}{\lambda}\Omega(\overline{\mathbf{G}},\mathcal{U}) \qquad (7)\\ &-\frac{1}{\lambda}\sum_{n=1}^N\sum_{k=1}^{K_n}(\boldsymbol{\lambda}_{n,k}^{[\tau-1]})'\mathbf{u}_{n,k}+\sum_{n=1}^N\frac{\rho_n}{2\lambda}\|\mathbf{Z}_n^{[\tau-1]}-\mathbf{U}_n\|_2^2.\end{aligned}$$

Problem (7) seeks estimates for $\overline{\mathbf{G}}$ and $\{\mathbf{U}_n\}_{n=1}^N$ where the $\mathbf{U}_n^{[\tau]}$'s must be *close* to the corresponding $\mathbf{Z}_n^{[\tau-1]}$'s. Note that (7) is a nonconvex and highly nonlinear optimization problem whose solution must be computed numerically.

Computing $\{\mathbf{S}_n^{[\tau]}\}_{n=1}^N$ decomposes into updating each $\mathbf{S}_n^{[\tau]}$ separately via the solution of $\min_{\mathbf{S}_n\in\mathcal{S}_n}\sum_{i=1}^{D_{-n}}\sum_{k=1}^{K_n}s_{n,i,k}d_{\phi_n}(\mathbf{z}_{n,k}^{[\tau-1]},\mathbf{y}_{(n),i})$. Since d_{ϕ_n} is nonnegative, $\mathbf{S}_n^{[\tau]}$ can be computed entry-wise as

$$s_{n,i,k}^{[\tau]}=\begin{cases}1 & d_{\phi_n}(\mathbf{z}_{n,k}^{[\tau-1]},\mathbf{y}_{(n),i})<d_{\phi_n}(\mathbf{z}_{n,\tilde{k}}^{[\tau-1]},\mathbf{y}_{(n),i}),\forall\tilde{k}\neq k\\ 0 & \text{otherwise}\end{cases} \qquad (8)$$

where ties are broken randomly and uniformly.

Solving (6b) decomposes into updating each $\mathbf{z}_{n,k}^{[\tau]}$ separately via

$$\begin{aligned}\min_{\mathbf{z}_{n,k}}\ &\sum_{i=1}^{D_{-n}}s_{n,i,k}^{[\tau]}d_{\phi_n}(\mathbf{z}_{n,k},\mathbf{y}_{(n),i}) \qquad (9)\\ &+\left(\boldsymbol{\lambda}_{n,k}^{[\tau-1]}\right)'\mathbf{z}_{n,k}+\frac{\rho_n}{2}\|\mathbf{z}_{n,k}-\mathbf{u}_{n,k}^{[\tau]}\|_2^2.\end{aligned}$$

Problem (9) is convex and differentiable with respect to $\mathbf{z}_{n,k}$. Note that when $\phi_n(\mathbf{y})=\|\mathbf{y}\|_2^2$, (9) can be solved in closed form [11].

The ADMM Bregman co-clustering algorithm, termed BreCo, for solving (3) is summarized as Algorithm 1. Although BreCo requires iterative solvers, the computation of these updates can be easily parallelized as it is the case for those updates in (8) and (9). Convergence analysis of ADMM for non-convex optimization problems is an active area of research that lies beyond the scope of this paper, see [13] and references therein for related work.

Algorithm 1: Bregman Tensor Co-Clustering (BreCo)

```
Data: $\overline{\mathbf{Y}}$, $\{\mathbf{Z}_n^{[0]},\boldsymbol{\Lambda}_n^{[0]}\}_{n=1}^N$, and the tuple $(\lambda,\mu,\{\mu_n,\rho_n\}_{n=1}^N)$.
Result: Membership-tuple for all entries of $\overline{\mathbf{Y}}$ via $\{\mathbf{S}_n\}_{n=1}^N$.
1 begin
2     Construct $\{\mathbf{Y}_{(n)}\}_{n=1}^N$ from $\overline{\mathbf{Y}}$.
3     for $\tau=1,\ldots,\tau_{\max}$ do
4         Compute $(\overline{\mathbf{G}}^{[\tau]},\{\mathbf{U}_n^{[\tau]}\}_{n=1}^N)$ via (7).
5         for $n=1,\ldots,N$ do
6             Compute the entries of $\mathbf{S}_n^{[\tau]}$ via (8).
7             Compute each column of $\mathbf{Z}_n^{[\tau]}$ via (9).
8             Update each column of $\boldsymbol{\Lambda}_n^{[\tau]}$ via (6c).
```

A. Tucker Decomposition via the Levenberg-Marquardt Method

In this section a solver for (7), based on the Levenberg-Marquardt algorithm [14, Ch. 1], is developed. Let $\boldsymbol{v}^{[\tau]} := [\mathrm{vec}(\mathbf{U}_1^{[\tau]})'\ldots\mathrm{vec}(\mathbf{U}_N^{[\tau]})'\ \mathrm{vec}(\mathbf{G}_{(1)}^{[\tau]})']' \in \mathbb{R}^{\Theta}$, where $\Theta := K+\sum_{n=1}^N K_nD_n$, $\mathbf{U}_n^{[\tau]}$ the estimate for $\mathbf{U}_n$ at iteration τ, and $\mathbf{G}_{(1)}^{[\tau]}$ denotes the mode-1 matricization of $\overline{\mathbf{G}}$ at iteration τ. Then, the proposed Levenberg-Marquardt algorithm iteratively updates $\boldsymbol{v}$ via

$$\boldsymbol{v}^{[\tau,j]}=\boldsymbol{v}^{[\tau,j-1]}+\left[\hat{\mathbf{H}}(\boldsymbol{v}^{[\tau,j-1]})+\xi_j\mathbf{I}_{\Theta}\right]^{-1}\boldsymbol{\delta}(\boldsymbol{v}^{[\tau,j-1]}) \qquad (10)$$

where $j\in\mathbb{N}_{>0}$ denotes the iteration index, $\hat{\mathbf{H}}(\boldsymbol{v}^{[\tau,j-1]})\in\mathbb{R}^{\Theta\times\Theta}$ an approximation to the Hessian of (7) evaluated at $\boldsymbol{v}^{[\tau,j-1]}$, $\boldsymbol{\delta}(\boldsymbol{v}^{[\tau,j-1]})\in\mathbb{R}^{\Theta}$ the gradient of (7) evaluated at $\boldsymbol{v}^{[\tau,j-1]}$, $\mathbf{I}_{\Theta}\in\mathbb{R}^{\Theta\times\Theta}$ an identity matrix, and $\xi_j>0$ is a tuning parameter that guarantees the matrix inversion operation to be well defined. The value of ξ_j affects both the direction and step size of the update in (10). Large values of ξ_j make (10) closer to a steepest-decent update, whereas small values of ξ_j make (10) closer to a Newton update [15].

With $D:=\prod_{n=1}^N D_n$, $\boldsymbol{\delta}(\boldsymbol{v})$ and $\hat{\mathbf{H}}(\boldsymbol{v})$ in (10) are given by

$$\begin{aligned}\boldsymbol{\delta}(\boldsymbol{v})&=\mathbf{J}(\hat{\mathbf{y}}_1-\mathbf{y}_1)+\frac{\partial}{\partial\boldsymbol{v}}f(\boldsymbol{v}) &(11a)\\ \hat{\mathbf{H}}(\boldsymbol{v})&=\mathbf{J}(\boldsymbol{v})'\mathbf{J}(\boldsymbol{v})+\hat{\boldsymbol{\Sigma}}(\boldsymbol{v}) &(11b)\end{aligned}$$

where $\mathbf{J}(\boldsymbol{v})\in\mathbb{R}^{D\times\Theta}$ denotes the Jacobian of the residuals $\overline{\mathbf{Y}}-\overline{\mathbf{G}}\times\{\mathbf{U}\}$ evaluated at $\boldsymbol{v}$, $\mathbf{y}_1:=\mathrm{vec}(\mathbf{Y}_{(1)})\in\mathbb{R}^D$, $f(\boldsymbol{v}):=\frac{1}{\lambda}\Omega(\overline{\mathbf{G}},\mathcal{U})-\frac{1}{\lambda}\sum_{n=1}^N\sum_{k=1}^{K_n}(\boldsymbol{\lambda}_{n,k}^{[\tau-1]})'\mathbf{u}_{n,k}+\sum_{n=1}^N\frac{\rho_n}{2\lambda}\|\mathbf{Z}_n^{[\tau-1]}-\mathbf{U}_n\|_2^2$, and $\hat{\boldsymbol{\Sigma}}(\boldsymbol{v})$ an approximation to the Hessian of f evaluated at $\boldsymbol{v}$. Let $\otimes$ denote the Kronecker product and define $\boldsymbol{\Upsilon}_n^m := \mathbf{U}_m\otimes\ldots\otimes\mathbf{U}_n$ with $m>n$. Using the following identity for the case $n=1$ [10]

$$\|\overline{\mathbf{Y}}-\overline{\mathbf{G}}\times\{\mathbf{U}\}\|_2^2=\|\mathbf{Y}_{(n)}-\mathbf{U}_n\mathbf{G}_{(n)}(\boldsymbol{\Upsilon}_{n+1}^N\otimes\boldsymbol{\Upsilon}_1^{n-1})'\|_2^2 \qquad (12)$$

it readily follows that $\hat{\mathbf{y}}_1:=\mathrm{vec}(\mathbf{U}_1\mathbf{G}_1(\boldsymbol{\Upsilon}_2^N)')\in\mathbb{R}^D$.

Since f is not differentiable due to the term $\|\overline{\mathbf{G}}\|_1$, $\partial f(\boldsymbol{v})/\partial\boldsymbol{v}$ is computed using a sub-gradient for its non-differentiable part yielding

$$\frac{\partial f(\boldsymbol{v})}{\partial\boldsymbol{v}}=\frac{1}{\lambda}\left[\mathrm{vec}(\mathbf{F}_1)'\ldots\mathrm{vec}(\mathbf{F}_N)'\ \mu\cdot\mathrm{sign}(\mathrm{vec}(\mathbf{G}_{(1)}))'\right]' \qquad (13)$$

Algorithm 2: Levenberg-Marquardt solver for (7)

```
Data: $\overline{\mathbf{Y}}$, $\overline{\mathbf{G}}^{[\tau,0]}$, $\{\mathbf{U}_n^{[\tau,0]}, \mathbf{\Lambda}_n^{[\tau]}, \mathbf{Z}_n^{[\tau]}, \mu_n, \rho_n\}_{n=1}^N$, $\lambda$, and $\mu$.
Result: Updated $\boldsymbol{\upsilon}^{[\tau]} = \boldsymbol{\upsilon}^{[\tau,j_{\max}]}$.
1 begin
2     Construct $\mathbf{y}_1 = \text{vec}(\mathbf{Y}_{(1)})$ and matrices $\{\mathbf{P}_n\}_{n=1}^N$.
3     Set $\boldsymbol{\upsilon}^{[\tau,0]} := \boldsymbol{\upsilon}^{[\tau]}$.
4     for $j = 0, \ldots, j_{\max}$ do
5         Construct $\hat{\mathbf{y}} := \text{vec}(\mathbf{U}_1\mathbf{G}_1(\mathbf{U}_N \otimes \ldots \otimes \mathbf{U}_2)')$.
6         Construct $\mathbf{J}$ using (15).
7         Compute $\boldsymbol{\delta}$ via (11a) and (13).
8         Compute $\hat{\mathbf{H}}$ via (11b) and (14).
9         Update $\boldsymbol{\upsilon}^{[\tau,j]}$ via (10).
```

Algorithm 3: Accelerated gradient-descent solver for (9)

```
Data: $\overline{\mathbf{Y}}$, $\{s_{n,i,k}^{[\tau]}\}_{i=1}^{D_{-n}}$ $\mathbf{u}_{n,k}^{[\tau]}, \boldsymbol{\lambda}_{n,k}^{[\tau-1]}, \mathbf{z}_{n,k}^{[\tau-1]}$, and $\alpha, \rho_n > 0$.
Result: Updated $\mathbf{z}_{n,k}^{[\tau]} = \mathbf{z}_{n,k}^{[\tau,\eta_{\max}]}$.
1 begin
2     Build $\mathbf{Y}_{(n)}$ from $\overline{\mathbf{Y}}$, and set $\zeta^{[0]} = 0$ and $\mathbf{z}_{n,k}^{[\tau,0]} := \mathbf{z}_{n,k}^{[\tau-1]}$.
3     Compute $\zeta^{[1]}$ via (17).
4     for $\eta = 1, \ldots, \eta_{\max}$ do
5         Compute $\nabla h(\mathbf{z}_{n,k}^{[\tau,\eta]})$ via (18).
6         Compute $\boldsymbol{\varphi}^{[\eta]}$ via (16a).
7         Compute $\zeta^{[\eta+1]}$ and $\beta^{[\eta]}$ via (17).
8         Update $\mathbf{z}^{[\tau,\eta]}$ via (16b) .
```

where $\mathbf{F}_n := \gamma_n \mathbf{U}_n - \mathbf{\Lambda}_n^{[\tau-1]} - \rho_n \mathbf{Z}_n^{[\tau-1]}$, $\gamma_n := \mu_n + \rho_n$, and $\text{sign}(\text{vec}(\mathbf{G}_{(1)})) \in \{-1, 0, 1\}^K$ a vector comprising the signs of the entries of $\overline{\mathbf{G}}$. Matrix $\hat{\mathbf{\Sigma}}(\boldsymbol{\upsilon})$ is defined as

$$\hat{\mathbf{\Sigma}}(\boldsymbol{\upsilon}) = \frac{1}{\lambda}\text{diag}([\gamma_1 \mathbf{1}'_{D_1K_1} \ldots \gamma_N \mathbf{1}'_{D_N k_N}\ \mathbf{0}'_K]') \tag{14}$$

where $\mathbf{1}_D \in \mathbb{R}^D$ a vector of ones, $\mathbf{0}_K \in \mathbb{R}^K$ a vector of zeros, and $\text{diag}(\mathbf{y})$ a diagonal matrix whose main diagonal is given by $\mathbf{y}$.

Using (12), it follows that the Jacobian $\mathbf{J} := [\mathbf{J}_1, \ldots, \mathbf{J}_N, \mathbf{J}_{\overline{\mathbf{G}}}]$ can be written block-wise as

$$\mathbf{J}_n = \mathbf{P}_n[(\mathbf{\Upsilon}_{n+1}^N \otimes \mathbf{\Upsilon}_1^{n-1})\mathbf{G}'_{(n)} \otimes \mathbf{I}_{D_n}] \in \mathbb{R}^{D \times D_n K_n} \tag{15a}$$

$$\mathbf{J}_{\overline{\mathbf{G}}} = \mathbf{\Upsilon}_1^N \in \mathbb{R}^{D \times K} \tag{15b}$$

where $\mathbf{P}_n \in \{0,1\}^{D \times D}$ is a permutation matrix that guarantees $\mathbf{y}_1 = \mathbf{P}_n \text{vec}(\mathbf{Y}_{(n)})$, $\mathbf{P}_1 = \mathbf{I}_D$, and $\mathbf{J}_{\overline{\mathbf{G}}}$ follows from $\|\overline{\mathbf{Y}} - \overline{\mathbf{G}} \times \{\mathbf{U}\}\|_2^2 = \|\mathbf{y}_1 - \mathbf{\Upsilon}_1^N \text{vec}(\mathbf{G}_{(1)})\|_2^2$ (cf. (12)). The Levenberg-Marquardt algorithm for solving (7) is summarized as Algorithm 2. Since the update directions (10) are gradient related, it is possible to show that $\boldsymbol{\upsilon}^{[\tau,j]}$ converges to a point whose entries characterize a stationary point of the cost in (7) using the results in [14].

B. Gradient-Descent Method for Computing Cluster Centroids

In this section an accelerated gradient-descent algorithm for solving (9) is developed. Let $h(\mathbf{z})$ denote the cost of (9). With $\eta \in \mathbb{N}_{>0}$ denoting the iteration index, Nesterov's accelerated gradient-descent iterations for (9) are given by [16, Ch. 3]

$$\boldsymbol{\varphi}^{[\eta]} = \mathbf{z}_{n,k}^{[\tau,\eta-1]} - \alpha \nabla h(\mathbf{z}_{n,k}^{[\tau,\eta-1]}) \tag{16a}$$

$$\mathbf{z}_{n,k}^{[\tau,\eta]} = \boldsymbol{\varphi}^{[\eta]} + \beta^{[\eta]}\left(\boldsymbol{\varphi}^{[\eta-1]} - \boldsymbol{\varphi}^{[\eta]}\right) \tag{16b}$$

where $\zeta^{[0]} = 0$, $\alpha > 0$ is a constant step size, $\text{Proj}_{\mathcal{Q}_n}(\mathbf{y})$ the projection of $\mathbf{y}$ into the set $\mathcal{Q}_n$,

$$\zeta^{[\eta]} = \frac{1}{2}\left[1 + \sqrt{1 + 4(\zeta^{[\eta-1]})^2}\right] \text{ and } \beta^{[\eta]} = (1 - \zeta^{[\eta]})/\zeta^{[\eta+1]}. \tag{17}$$

Since the cost in (9) is differentiable with respect to $\mathbf{z}_{n,k}$, $\nabla h(\mathbf{z}_{n,k}^{[\tau,\eta-1]})$ is well-defined for all η and given by

$$\nabla h(\mathbf{z}_{n,k}) = \sum_{i=1}^{D_{-n}} s_{n,i,k}^{[\tau]}\left[\nabla \phi_n(\mathbf{z}_{n,k}) - \nabla \phi_n(\mathbf{y}_{(n),i})\right] + \boldsymbol{\lambda}_{n,k}^{[\tau-1]} + \rho_n\left(\mathbf{z}_{n,k} - \mathbf{u}_{n,k}^{[\tau]}\right). \tag{18}$$

Nesterov's accelerated gradient-descent algorithm is summarized as Algorithm (3). Note that h is convex and its global minimum is denoted as $\mathbf{z}_{n,k}^*$. It is possible to show that when ϕ_n is an L_ϕ-smooth function in $\mathbb{R}^{D_n}$, the iterations in (16) achieve the optimal rate of convergence for first order methods of $O(1/\eta^2)$ as summarized by the following proposition.

Proposition 4.1: Let $\phi_n : \mathbb{R}^{D_n} \to \mathbb{R}$ be a strictly convex and differentiable function that satisfies $\|\nabla\phi_n(\mathbf{z}) - \nabla\phi_n(\mathbf{y})\| \leq L_{\phi_n}\|\mathbf{z} - \mathbf{y}\|$, $\forall \mathbf{z}, \mathbf{y} \in \text{dom}(\phi_n)$. Then, the iterates in (16) satisfy $h(\mathbf{z}_{n,k}^{[1]}) - h(\mathbf{z}_{n,k}^*) \leq 2\alpha\|\mathbf{z}_{n,k}^{[1]} - \mathbf{z}_{n,k}^*\|^2/\eta^2$, where $\alpha = L_{\phi_n}C + \rho_n$ and $C := \sum_{i=1}^{D-n} s_{n,i,k}^{[\tau]}$.

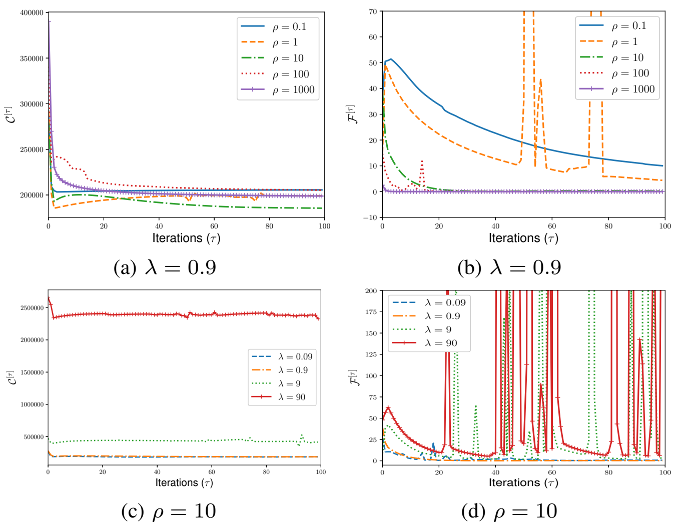
Remark 4.1 (Arbitrary Bregman Divergences): Although beyond the scope of this work, it is worth mentioning that BreCo can be generalized to capture arbitrary Bregman divergences after modifying the gradient descent method for solving (9). Specifically, for an arbitrary Bregman divergence defined by ϕ_n with domain $\mathcal{Q}_n \subseteq \mathbb{R}^{D_n}$, one can replace (16) with a projected gradient descent update of the form $\mathbf{z}^{[\eta]} = \text{Proj}_{\mathcal{Q}_n}[\mathbf{z}_{n,k}^{[\tau,\eta-1]} - \alpha\nabla h(\mathbf{z}_{n,k}^{[\tau,\eta-1]})]$, where $\text{Proj}_{\mathcal{Q}_n}[\cdot]$ is the projection operator onto the interior of $\mathcal{Q}_n$.

V. Numerical Tests

In this section, the performance of BreCo is illustrated on a synthetic tensor $\overline{\mathbf{Y}} \in \mathbb{R}^{50 \times 60 \times 30}$ where $N = 3$. The tensor was constructed by adding four non-overlapping co-clusters with entries drawn uniformly at random from intervals [3,5], [-3,-2], [-4,-3] and [5,7] to a tensor whose entries were drawn from a univariate, zero-mean Gaussian distribution with variance 0.5. The Bregman divergence defined by $\phi_n(\mathbf{y}) = \|\mathbf{y}\|_2^2$, which corresponds to the Euclidean distance, is used herein as an illustrative example.

BreCo's tuning parameters were empirically set to $\mu_n = 0.01\ \forall n$, $\mu = 2$, $\lambda = 0.9$, and $\rho_n = \rho\ \forall n$. Research on a systematic approach for selecting these parameters in ongoing. The number of clusters per mode was set to $K_1 = K_2 = K_3 = 5$ to attempt capturing the 4 artificial co-clusters and the additional cluster with Gaussian-distributed entries per tensor mode. The Levenberg-Marquardt algorithm was initialized using values for $\overline{\mathbf{G}}^{[1,0]}$ and $\{\mathbf{U}_n^{[1,0]}\}_{n=1}^n$ obtained via the high-order singular-value decomposition (HOSVD) for $\overline{\mathbf{Y}}$ [1]. The initial value for ξ_j was set to a fraction $\theta = 0.0001$ of the largest entry on the diagonal of $\mathbf{J}(\boldsymbol{\upsilon}^{[\tau,0]})'\mathbf{J}(\boldsymbol{\upsilon}^{[\tau,0]})$. The schedule for updating ξ_j used herein is chosen as in [15, Ch. 3]. The $\{\mathbf{Z}_n^{[0]}\}_{n=1}^N$ were set to randomly chosen columns of the corresponding $\{\mathbf{Y}_{(n)}\}_{n=1}^N$, and $\{\mathbf{\Lambda}_n^{[0]}\}_{n=1}^N$ were chosen entry-wise at random from a standard normal distribution. The learning rate in Algorithm 3 was set to $\alpha = 10^{-4}$.

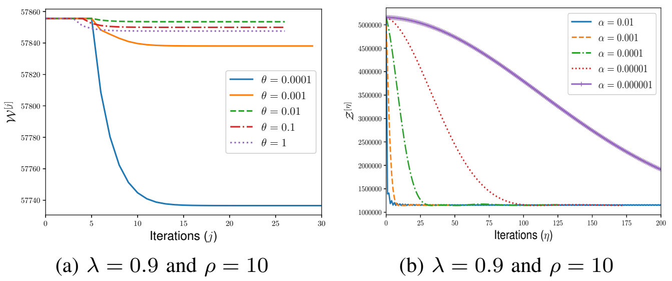
The figures-of-merit used to assess the numerical behavior of BreCo were the evolution of the cost in (4), which is denoted by $\mathcal{C}^{[\tau]}$, and the aggregate equality-constraint violation as quantified by $\mathcal{F}^{[\tau]} := 0.5\rho\sum_{n=1}^N \|\mathbf{Z}_n^{[\tau]} - \mathbf{U}_n^{[\tau]}\|_2^2$. Fig. 1 shows the evolution of the proposed figures-of-merit for different values of ρ and λ. In



(a) $\lambda = 0.9$ (b) $\lambda = 0.9$

(c) $\rho = 10$ (d) $\rho = 10$

Fig. 1. BreCo's numerical evolution summary.



(a) $\lambda = 0.9$ and $\rho = 10$ (b) $\lambda = 0.9$ and $\rho = 10$

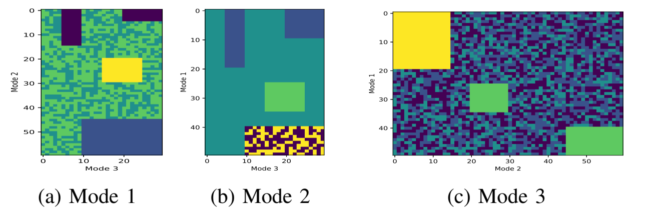
Fig. 2. Sample trajectories for $\mathcal{W}^{[j]}$ and $\mathcal{Z}^{[\eta]}$.

general, BreCo was able to identify "stable" co-clusters in less than 100 iterations. Although spikes in $\mathcal{F}^{[\tau]}$ reveal sudden increases in the violation of the constraint in (4) possibly triggered by significant $\{\mathbf{S}_n\}_{n=1}^{N}$ realignments caused by the reordering of the columns of $\{\mathbf{U}\}$ when solving (7), BreCo was able to recover and eventually identify stable per-mode clusters. Large values for λ translated to more frequent, sudden increments in constraint violations. The tuning parameter λ was adjusted so that the Tucker model approximation to $\overline{\mathbf{Y}}$ and the aggregate Bregman divergence terms are balanced. In the case of the Euclidean distance considered here, a λ value close to 1 is appropriate since both approximation terms are of the same order of magnitude. Fig. 2 shows sample trajectories of the cost of (7) and (9), denoted by $\mathcal{W}^{[j]}$ and $\mathcal{Z}^{[\eta]}$, respectively.

Fig. 3 illustrates the clustering results of the fibers of $\overline{\mathbf{Y}}$ as summarized by $\{\mathbf{S}_n\}_{n=1}^{N}$. After mapping these per-mode cluster assignments to unique co-cluster labels, one finds $K = 125$ different co-cluster labels for the entries of $\overline{\mathbf{Y}}$. Thus, defining how to compare the co-clustering results with the true co-cluster labels is not obvious. Here, we compared the co-clustering assignments for the sub-tensors where the artificial co-clusters were added only. In this case the Fowlkes-Mallows score, which computes the geometric mean of the pairwise precision and recall scores, was 0.735 and the normalized mutual information score was 0.791. In this example, the relatively low scores were due to the fact that two different co-clusters, the co-clusters with values drawn from the intervals [3,5] and [5,7], were assigned the same co-cluster label.

VI. Summary

This paper proposed a tensor co-clustering algorithm termed BreCo that used Bregman divergences as a measure of co-cluster dissimilarity. BreCo, via ADMM, decomposed the tensor co-clustering problem into two updates that are reminiscent of a regularized Bregman-clustering and a regularized Tucker-decomposition update, respectively. Numerical solvers based on the Levenberg-Marquardt algorithm and accelerated gradient descent were developed. BreCo's performance was illustrated via numerical tests on synthetic data.



(a) Mode 1 (b) Mode 2 (c) Mode 3

Fig. 3. Clustering results per mode of $\overline{\mathbf{Y}}$. The colors on each mask correspond to different cluster labels per mode.

References

[1] A. Cichocki, D. Mandic, L. D. Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, March 2015.

[2] J. A. Hartigan, "Direct clustering of a data matrix," *Journal of the American Statistical Association*, vol. 67, no. 337, pp. 123–129, 1972.

[3] D. Hatano, T. Fukunaga, T. Maehara, and K. Kawarabayashi, "Scalable algorithm for higher-order co-clustering via random sampling," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, CA, USA.*, 2017, pp. 1992–1999.

[4] T. Wu, A. R. Benson, and D. F. Gleich, "General tensor spectral co-clustering for higher-order data," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. USA: Curran Associates Inc., 2016, pp. 2567–2575.

[5] S. Jegelka, S. Sra, and A. Banerjee, "Approximation algorithms for tensor clustering," in *Proceedings of the 20th International Conference on Algorithmic Learning Theory*, ser. ALT'09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 368–383.

[6] W. W. Sun and L. Li, "Dynamic tensor clustering," *J. Am. Stat. Assoc.*, vol. 114, no. 528, pp. 1894–1907, Mar. 2019.

[7] E. E. Papalexakis, N. D. Sidiropoulos, and R. Bro, "From K-means to higher-way co-clustering: Multilinear decomposition with sparse latent factors," *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 493–506, 2013.

[8] H. Zhao, D. D. Wang, L. Chen, X. Liu, and H. Yan, "Identifying multi-dimensional co-clusters in tensors based on hyperplane detection in singular vector spaces," *PloS One*, vol. 11, no. 9, Sep 2016.

[9] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh, "Clustering with Bregman divergences," *JMLR*, vol. 6, pp. 1705–1749, 2005.

[10] N. D. Sidiropoulos, L. D. Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, July 2017.

[11] P. A. Forero, P. A. Baxley, and M. Capella, "Co-clustering of high-order data via regularized Tucker decompositions," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, May 12-17, Brighton, UK, 2019, pp. 3442–3446.

[12] A. Siahkamari, V. Saligrama, D. Castanon, and B. Kulis, "Learning Bregman Divergences," *arXiv e-prints*, p. arXiv:1905.11545, May 2019.

[13] S. Magnússon, P. C. Weeraddana, M. G. Rabbat, and C. Fischione, "On the convergence of alternating direction Lagrangian methods for nonconvex structured optimization problems," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 3, pp. 296–309, Sept 2016.

[14] D. P. Bertsekas, *Nonlinear Programming*, 3rd ed. Athena Scientific, 2016.

[15] K. Madsen, H. B. Nielsen, and O. Tingleff, "Methods for non-linear least squares problems," 2nd. Ed., Danmarks Tekniske Universitet, Tech. Rep., Apr. 2004.

[16] S. Bubeck, "Convex optimization: Algorithms and complexity," *Found. Trends Mach. Learn.*, vol. 8, no. 3-4, pp. 231–357, Nov. 2015.