Journal of Machine Learning Research 21 (2020) 1-58 Submitted 3/18; Revised 2/20; Published 10/20

Provable Convex Co-clustering of Tensors

Eric C. Chi ERIC_CHI@NCSU.EDU
Department of Statistics
North Carolina State University
Raleigh, NC 27695, USA

Brian R. Gaines BRIAN.GAINES@SAS.COM
Advanced Analytics R&D
SAS Institute Inc.
Cary, NC 27513, USA

Will Wei Sun SUN244@PURDUE.EDU
Krannert School of Management
Purdue University
West Lafayette, IN 47907, USA

Hua Zhou HUAZHOU@UCLA.EDU
Department of Biostatistics
University of California
Los Angeles, CA 90095, USA

Jian Yang JIANYANG@OATH.COM
Advertising Sciences
Yahoo Research
Sunnyvale, CA 94089, USA

Editor: Francis Bach

Abstract

Cluster analysis is a fundamental tool for pattern discovery of complex heterogeneous data. Prevalent clustering methods mainly focus on vector or matrix-variate data and are not applicable to general-order tensors, which arise frequently in modern scientific and business applications. Moreover, there is a gap between statistical guarantees and computational efficiency for existing tensor clustering solutions due to the nature of their non-convex formulations. In this work, we bridge this gap by developing a provable convex formulation of tensor co-clustering. Our convex co-clustering (CoCo) estimator enjoys stability guarantees and its computational and storage costs are polynomial in the size of the data. We further establish a non-asymptotic error bound for the CoCo estimator, which reveals a surprising "blessing of dimensionality" phenomenon that does not exist in vector or matrix-variate cluster analysis. Our theoretical findings are supported by extensive simulated studies. Finally, we apply the CoCo estimator to the cluster analysis of advertisement click tensor data from a major online company. Our clustering results provide meaningful business insights to improve advertising effectiveness.

Keywords: Clustering, Fused lasso, High-dimensional Statistical Learning, Multiway Data, Non-asymptotic Error

1. Introduction

In this work, we study the problem of finding structure in multiway data, or tensors, via clustering. Tensors appear frequently in modern scientific and business applications involving complex heterogeneous data. For example, data in a neurogenomics study of brain development consists of a 3-way array of expression level measurements indexed by gene, space, and time (Liu et al., 2017). Other examples of 3-way data arrays consisting of matrices collected over time include email communications (sender, recipient, time) (Papalexakis et al., 2013), online chatroom communications (user, keyword, time) (Acar et al., 2006), bike rentals (source station, destination station, time) (Guigourès et al., 2015), and internet network traffic (source IP, destination IP, time) (Sun et al., 2006). The rise in tensor data has created new challenges in making predictions, such as in recommender systems for example (Zheng et al., 2016; Symeonidis, 2016; Symeonidis and Zioupos, 2016; Frolov and Oseledets, 2017; Bi et al., 2018) as well as inferring latent structure in multiway data (Acar and Yener, 2009; Anandkumar et al., 2014; Cichocki et al., 2015; Sidiropoulos et al., 2017).

As tensors become increasingly more common, the need for a reliable co-clustering method grows increasingly more urgent. Prevalent clustering methods, however, mainly focus on vector or matrix-variate data. The goal of vector clustering is to identify subgroups within the vector-variate observations (Ma and Zhong, 2008; Shen and Huang, 2010; Shen et al., 2012; Wang et al., 2013). Biclustering is the extension of clustering to two-way data where both the observations (rows) and the features (columns) of a data matrix are simultaneously grouped together (Hartigan, 1972; Madeira and Oliveira, 2004; Busygin et al., 2008). In spite of their prevalence, these approaches are not directly applicable to the cluster analysis of general-order (general-way) tensors. On the other hand, existing methods for co-clustering general D-way arrays, for $D \geq 3$, employ one of three strategies: (i) extensions of spectral clustering to tensors (Wu et al., 2016b), (ii) directly clustering the subarrays along each dimension, or way, of the tensor using either k-means or variants on it (Jegelka et al., 2009), and (iii) low rank tensor decompositions (Sun et al., 2009; Papalexakis et al., 2013; Zhao et al., 2016). While all these existing approaches may demonstrate good empirical performance, they have limitations. For instance, the spectral co-clustering method proposed by Wu et al. (2016b) is limited to nonnegative tensors and the CoTeC method proposed by Jegelka et al. (2009), like k-means, requires specifying the number of clusters along each dimension as a tuning parameter. Most importantly, none of the existing methods provide statistical guarantees for recovering an underlying co-clustering structure. There is a conspicuous gap between statistical guarantees and computational efficiency for existing tensor clustering solutions due to the nature of the non-convex formulations of the previously mentioned works.

In this paper, we propose a Convex Co-clustering (CoCo) procedure that solves a convex formulation of the problem of co-clustering a D-way array for $D \geq 3$. Our proposed CoCo estimator affords the following advantages over existing tensor co-clustering methods.

(i) Under modest assumptions on the data generating process, the CoCo estimator is guaranteed to recover an underlying co-clustering structure with high probability. In particular, we establish a non-asymptotic error bound for the CoCo estimator, which reveals a surprising "blessing of dimensionality" phenomenon: As the dimensions of the array increase, the CoCo estimator is *still* consistent even if the number of

underlying co-clusters grows as a function of the number of elements in the tensor sample. More importantly, an underlying co-clustering structure can be consistently recovered with even a single tensor sample, which is a typical case in real applications. This phenomenon does not exist in vector or matrix-variate cluster analysis.

(ii) The CoCo estimator possesses stability guarantees. In particular, the CoCo estimator is Lipschitz continuous in the data and jointly continuous in the data and its tuning parameter. We emphasize that Lipschitz continuity in the data guarantees that perturbations in the data lead to graceful and commensurate variations in the cluster assignments, and the continuity in the tuning parameter can be leveraged to expedite computation through warm starts.

(iii) The CoCo estimator can be iteratively computed with convergence guarantees via an accelerated first order method with storage and per-iteration cost that is linear in the size of the data.

In short, the CoCo estimator comes with (i) statistical guarantees, (ii) practically relevant stability guarantees at all sample sizes, and (iii) an algorithm with polynomial complexity. The theoretical properties of our CoCo estimator are supported by extensive simulation studies. To demonstrate its business impact, we apply the CoCo estimator to the cluster analysis of advertisement click tensor data from a major online company. Our clustering results provide meaningful business insights to help advertising planning.

Our work is related to, but also clearly distinct from, a number of recent developments in cluster analysis. The first related line of research tackles convex clustering (Hocking et al., 2011; Zhu et al., 2014; Chi and Lange, 2015; Chen et al., 2015; Tan and Witten, 2015; Wang et al., 2018; Radchenko and Mukherjee, 2017) and convex biclustering (Chi et al., 2017). These existing methods are not directly applicable to general-order tensors, however. Importantly, our CoCo estimator enjoys a unique "blessing of dimensionality" phenomenon that has not been established in the aforementioned approaches. Moreover, the CoCo estimator is similar in spirit to a recent series of work approximating a noisy observed array with an array that is smooth with respect to some latent organization associated with each dimension of the array (Gavish and Coifman, 2012; Ankenman, 2014; Mishne et al., 2016; Yair et al., 2017). Our proposed CoCo procedure seeks an approximating array that is smooth with respect to a latent clustering along each dimension of the array. While CoCo shares features with these array approximation techniques, namely the use of data-driven similarity graphs along tensor modes, a key distinction between our CoCo estimator and these methods is that CoCo produces an approximating array that explicitly recovers hard co-clustering assignments. As we will see shortly, focusing our attention in this work on the co-clustering model paves the way to the discovery and explicit characterization of new and interesting fundamental behavior in finding intrinsic organization within tensors.

The rest of the paper is organized as follows. In Section 2, we review standard facts and results about tensors that we will use. In Section 3, we introduce our convex formulation of the co-clustering problem. In Section 4, we establish the stability properties and prediction error bounds of the CoCo estimator. In Section 5, we describe the algorithm used to compute the CoCo estimator. In Section 6, we discuss how to specify weights used in our CoCo estimator, and in Section 7 we give guidance on how to set and select tuning

parameters used in the CoCo estimator in practice. In Section 8, we present simulation results. In Section 9, we discuss the results of applying the CoCo estimator to co-cluster a real data tensor from online advertising. In Section 10, we close with a discussion. The Appendix contains a brief review of the two main tensor decompositions that are discussed in this paper, all technical proofs, as well as additional experiments.

2. Preliminaries

2.1 Notation

We adopt the terminology and notation used by Kolda and Bader (2009). We call the number of ways or modes of a tensor its *order*. Vectors are tensors of order one and denoted by boldface lowercase letters, e.g. $\mathbf{a}$. Matrices are tensors of order two and denoted by boldface capital letters, e.g. $\mathbf{A}$. Tensors of higher-order, namely order three and greater, we denote by boldface Euler script letters, e.g. $\boldsymbol{\mathcal{A}}$. Thus, if $\boldsymbol{\mathcal{A}}$ represent a D-way data array of size $n_1 \times n_2 \times \cdots \times n_D$, we say $\boldsymbol{\mathcal{A}}$ is a tensor of order D. We denote scalars by lowercase letters, e.g. a. We denote the ith element of a vector $\mathbf{a}$ by a_i, the ijth element of a matrix $\mathbf{A}$ by a_{ij}, the ijkth element of a third-order tensor $\boldsymbol{\mathcal{A}}$ by a_{ijk}, and so on.

We can extract a subarray of a tensor by fixing a subset of its indices. For example, by fixing the first index of a matrix to be i, we extract the ith row of the matrix, and by fixing the second index of a matrix to be j, we extract a jth column of the matrix. We use a colon to indicate all elements of a mode. Consequently, we denote the ith row of a matrix $\mathbf{A}$ by $\mathbf{A}_{i:}$ and the jth column of a matrix $\mathbf{A}$ by $\mathbf{A}_{:j}$. *Fibers* are the subarrays of a tensor obtained by fixing all but one of its indices. In the case of a matrix, a mode-1 fiber is a matrix column and a mode-2 fiber is a matrix row. *Slices* are the two-dimensional subarrays of a tensor obtained by fixing all but two indices. For example, a third-order tensor $\mathbf{A}$ has three sets of slices denoted by $\boldsymbol{\mathcal{A}}_{i::}, \boldsymbol{\mathcal{A}}_{:j:}$, and $\boldsymbol{\mathcal{A}}_{::k}$.

2.2 Basic Tensor Operations

It is often convenient to reorder the elements of a D-way array into a matrix or a vector. Reordering a tensor's elements into a matrix is referred to as *matricization*, while reordering its elements into a vector is referred to as *vectorization*. There are many ways to reorder a tensor into a matrix or vector. In this paper, we use a canonical mode-d matricization, where the mode-d fibers of a D-way tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_D}$ become the columns of a matrix $\mathbf{A}_{(d)} \in \mathbb{R}^{n_d \times n_{-d}}$, where $n_{-d} = \prod_{j \neq d} n_j$. Recall that the column-major vectorization of a matrix maps a matrix $\mathbf{A} \in \mathbb{R}^{p \times q}$ to the vector $\mathbf{a} \in \mathbb{R}^{pq}$ by stacking the columns of $\mathbf{A}$ on top of each other, namely $\mathbf{a} = \begin{pmatrix} \mathbf{A}_{:1}^\mathsf{T} & \mathbf{A}_{:2}^\mathsf{T} & \cdots & \mathbf{A}_{:q}^\mathsf{T} \end{pmatrix}^\mathsf{T} \in \mathbb{R}^{pq}$. In this paper, we take the vectorization of a D-way tensor $\boldsymbol{\mathcal{A}}$, denoted $\mathrm{vec}(\boldsymbol{\mathcal{A}})$, to be the column-major vectorization of the mode-1 matriciziation of $\boldsymbol{\mathcal{A}}$, namely $\mathrm{vec}(\boldsymbol{\mathcal{A}}) = \mathrm{vec}(\mathbf{A}_{(1)}) \in \mathbb{R}^n$, where $n = \prod_d n_d$ the total number of elements in $\boldsymbol{\mathcal{A}}$. As a shorthand, when the context leaves no ambiguity, we denote this vectorization of a tensor $\boldsymbol{\mathcal{A}}$ by its boldface lowercase version $\mathbf{a}$.

The Frobenius norm of a D-way tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_D}$ is the natural generalization of the Frobenius norm of a matrix, namely it is the square root of the sum of the squares

of all its elements,

$$\|\mathcal{A}\|_{\mathrm{F}} \quad = \quad \sqrt{\sum_{i_1=1}^{n_1}\sum_{i_2=1}^{n_2}\cdots\sum_{i_D=1}^{n_D} a_{i_1 i_2 \cdots i_D}^2}.$$

The Frobenius norm of a tensor is equivalent to the ℓ_2-norm of the vectorization of the tensor, namely $\|\mathcal{A}\|_{\mathrm{F}} = \|\mathbf{a}\|_2$.

Let $\mathcal{A}$ be a tensor in $\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_D}$ and $\mathbf{B}$ be a matrix in $\mathbb{R}^{m \times n_d}$. The *d-mode (matrix) product* of the tensor $\mathcal{A}$ with the matrix $\mathbf{B}$, denoted by $\mathcal{A} \times_d \mathbf{B}$, is the tensor of size $n_1 \times \cdots \times n_{d-1} \times m \times n_{d+1} \times \cdots \times n_D$ whose $(i_1, i_2, \cdots, i_{d-1}, j, i_{d+1}, \cdots, i_D)$th element is given by

$$(\mathcal{A} \times_d \mathbf{B})_{i_1 \ldots i_{d-1} j i_{d+1} \cdots i_D} \quad = \quad \sum_{i_d=1}^{n_d} a_{i_1 i_2 \cdots i_D} b_{j i_d},$$

for $j \in \{1, \ldots, m\}$. The vectorization of the d-mode product $\mathcal{A} \times_d \mathbf{B}$ can be expressed as

$$\mathrm{vec}(\mathcal{A} \times_d \mathbf{B}) \quad = \quad (\mathbf{I}_{n_D} \otimes \cdots \otimes \mathbf{I}_{n_{d+1}} \otimes \mathbf{B} \otimes \mathbf{I}_{n_{d-1}} \otimes \cdots \otimes \mathbf{I}_{n_1})\mathbf{a}, \tag{1}$$

where $\mathbf{I}_p$ is the p-by-p identity matrix and $\otimes$ denotes the Kronecker product between two matrices. The identity given in (1) generalizes the well known formula for the column-major vectorization of a product of two matrices, namely $\mathrm{vec}(\mathbf{BA}) = (\mathbf{I} \otimes \mathbf{B})\mathbf{a}$.

3. A Convex Formulation of Co-clustering

We first consider a convex formulation of co-clustering problem when the data is a 3-way tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ before discussing the natural generalization to D-way tensors. Our basic assumption is that the observed data tensor is a noisy realization of an underlying tensor that exhibits a checkerbox structure modulo some unknown reordering along each of its modes. Specifically suppose that there are k_1, k_2, and k_3 clusters along modes 1, 2, and 3 respectively. If the (i_1, i_2, i_3)-th entry in $\mathcal{X}$ belongs to the cluster defined by the r_1th mode-1 group, r_2th mode-2 group, and r_3th mode-3 group, then we assume that the observed tensor element $x_{i_1 i_2 i_3}$ is given by

$$x_{i_1 i_2 i_3} \quad = \quad {c^*}_{r_1 r_2 r_3} + \epsilon_{i_1 i_2 i_3}, \tag{2}$$

where ${c^*}_{r_1 r_2 r_3}$ is the mean of the co-cluster defined by the r_1th mode-1 partition, r_2th mode-2 partition, and r_3th mode-3 partition, and $\epsilon_{i_1 i_2 i_3}$ are noise terms. We will specify a joint distribution on the noise terms later in Section 4.2 in order to derive prediction bounds. Thus, we model the observed tensor $\mathcal{X}$ as the sum of a mean tensor $\mathcal{U}^* \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, whose elements are expanded from the co-cluster means tensor $\mathcal{C}^* \in \mathbb{R}^{k_1 \times k_2 \times k_3}$, and a noise tensor $\mathcal{E} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$. We can write this expansion explicitly by introducing a membership matrix $\mathbf{M}_d \in \{0, 1\}^{n_d \times k_d}$ for the dth mode, where the ikth element of $\mathbf{M}_d$ is one if and only if the ith mode-d slice belongs to the kth mode-d cluster for $k \in \{1, \ldots, k_d\}$. We require that each row of the membership matrix sum to one, namely $\mathbf{M}_d \mathbf{1} = 1$, to ensure that each of the mode-d slices belongs to exactly one of the k_d mode-d clusters. Then,

$$\mathcal{U}^* \quad = \quad \mathcal{C}^* \times_1 \mathbf{M}_1 \times_2 \mathbf{M}_2 \times_3 \mathbf{M}_3.$$

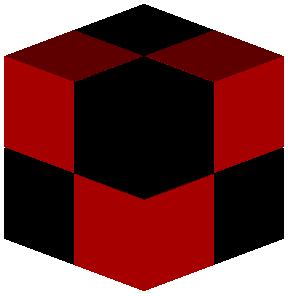


Figure 1: A 3-way tensor with a checkerbox structure

Figure 1 illustrates an underlying mean tensor $\mathcal{U}^*$ after permuting the slices along each of the modes to reveal a checkerbox structure.

The co-clustering model in (2) is the 3-way analogue of the checkerboard mean model often employed in biclustering data matrices (Madeira and Oliveira, 2004; Tan and Witten, 2014; Chi et al., 2017). Moreover, the tensor $\mathcal{C}^*$ of co-cluster means corresponds to the tensor of cluster "centers" in the tensor clustering work by Jegelka et al. (2009). The model is complete and exclusive in that each tensor element is assigned to exactly one co-cluster. This is in contrast to models that allow potentially overlapping co-clusters (Lazzeroni and Owen, 2002; Bergmann et al., 2003; Turner et al., 2005; Huang et al., 2008; Witten et al., 2009; Lee et al., 2010; Sill et al., 2011; Bhar et al., 2015).

Estimating the model in (2) consists of finding (i) the partitions along each mode and (ii) the mean values of each of the $k_1k_2k_3$ co-clusters. Estimating ${c^*}_{r_1r_2r_3}$, given the mode clustering assignments is trivial. Let $\mathcal{G}_1, \mathcal{G}_2$, and $\mathcal{G}_3$ denote the indices of the r_1th mode-1, r_2th mode-2, and r_3th mode-3 groups respectively. If the noise terms $\epsilon_{i_1i_2i_3}$ are iid $N(0, \sigma^2)$ for some positive σ^2, then the maximum likelihood estimate of ${c^*}_{r_1r_2r_3}$ is simply the sample mean of the entries of $\mathcal{X}$ over the indices defined by $\mathcal{G}_1, \mathcal{G}_2$, and $\mathcal{G}_3$, namely

$$\hat{c}^*_{r_1r_2r_3} \quad = \quad \frac{1}{|\mathcal{G}_1||\mathcal{G}_2||\mathcal{G}_3|} \sum_{i_1 \in \mathcal{G}_1} \sum_{i_2 \in \mathcal{G}_2} \sum_{i_3 \in \mathcal{G}_3} x_{i_1i_2i_3}.$$

Finding the partitions $\mathcal{G}_1, \mathcal{G}_2$, and $\mathcal{G}_3$, on the other hand, is a combinatorially hard problem. In recent years, however, many combinatorially hard problems, that initially appear computationally intractable, have been successfully attacked by solving a convex relaxation to the original combinatorial optimization problem. Perhaps the most celebrated convex relaxations is the lasso (Tibshirani, 1996), which simultaneously performs variable selection and parameter estimation for fitting sparse regression models by minimizing a non-smooth convex criterion.

In light of the lasso's success, we propose to simultaneously identify partitions along the modes of $\mathcal{X}$ and estimate the co-cluster means by minimizing the following convex objective function

$$F_\gamma(\mathcal{U}) \quad = \quad \frac{1}{2}\|\mathcal{X} - \mathcal{U}\|_{\mathrm{F}}^2 + \gamma \underbrace{\Big[R_1(\mathcal{U}) + R_2(\mathcal{U}) + R_3(\mathcal{U}) \Big]}_{R(\mathcal{U})}, \tag{3}$$

where

$$\begin{aligned} R_1(\mathcal{U}) &= \sum_{i<j} w_{1,ij} \|\mathcal{U}_{i::} - \mathcal{U}_{j::}\|_{\mathrm{F}} \\ R_2(\mathcal{U}) &= \sum_{i<j} w_{2,ij} \|\mathcal{U}_{:i:} - \mathcal{U}_{:j:}\|_{\mathrm{F}} \\ R_3(\mathcal{U}) &= \sum_{i<j} w_{3,ij} \|\mathcal{U}_{::i} - \mathcal{U}_{::j}\|_{\mathrm{F}}. \end{aligned}$$

By seeking the minimizer $\hat{\mathcal{U}}_\gamma \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ of (3), we have cast co-clustering as a signal approximation problem, modeled as a penalized regression, to estimate the true co-cluster means tensor $\mathcal{U}^*$. In the following discussion, we drop the dependence of γ in $\hat{\mathcal{U}}_\gamma$ and denote our estimator as $\hat{\mathcal{U}}$ when there is no confusion. The quadratic term in (3) quantifies how well $\mathcal{U}$ approximates $\mathcal{X}$, while the regularization term $R(\mathcal{U})$ in (3) penalizes deviations away from a checkerbox pattern. The nonnegative parameter γ tunes the relative emphasis on these two terms. The parameters $w_{d,ij}$ are nonnegative weights whose purpose will be discussed shortly.

To appreciate how the regularization term $R(\mathcal{U})$ steers the minimizer of (3) towards a checkerbox pattern, consider the effect of one of the terms $R_d(\mathcal{U})$ in isolation. Specifically, suppose that $R(\mathcal{U}) = R_1(\mathcal{U})$. When γ is zero, the minimum of (3) is attained when $\mathcal{U} = \mathcal{X}$. Or stated another way, $\mathcal{U}_{i::} = \mathcal{X}_{i::}$ for $i \in \{1, \ldots, n_1\}$. As γ increases, the mode-1 slices $\mathcal{U}_{i::}$ will shrink towards each other and in fact coalesce due to the non-differentiability of the Frobenius norm at zero. In other words, as γ gets larger, the pairwise differences of the mode-1 slices of $\hat{\mathcal{U}}$ will become increasingly sparser. Sparsity in these pairwise differences leads to a natural partitioning assignment. Two mode-1 slices $\mathcal{X}_{i::}$ and $\mathcal{X}_{j::}$ are assigned to the same mode-1 partition if $\mathcal{U}_{i::} = \mathcal{U}_{j::}$. Under mild regularity conditions, that we will spell out in Section 4, for sufficiently large γ, all mode-1 slices $\hat{\mathcal{U}}$ will be identical and therefore belong to a single cluster. Similar behavior holds if $R(\mathcal{U}) = R_2(\mathcal{U})$ or $R(\mathcal{U}) = R_3(\mathcal{U})$.

When $R(\mathcal{U})$ includes all three terms $R_d(\mathcal{U})$ for $d = 1, 2, 3$, pairs of mode-1, mode-2, and mode-3 slices are *simultaneously* shrunk towards each other and coalesce as the parameter γ increases. By coupling clustering along each of the modes simultaneously, our formulation explicitly seeks out a solution with a checkerbox mean structure. Moreover, we will show in Section 4 that the solution $\hat{\mathcal{U}}$ produces an entire solution path of checkerbox co-clustering estimates that varies continuously in γ. The solution path spans a range of models from the least smoothed model, where $\hat{\mathcal{U}}$ is $\mathcal{X}$ and each tensor element occupies its own co-cluster, to the most smoothed model, where all the elements of $\hat{\mathcal{U}}$ are identical and all tensor elements belong to a single co-cluster.

The nonnegative weights $w_{d,ij}$ fine tune the shrinkage of the slices along the dth mode. For example, if $w_{1,ij} > w_{1,i'j'}$, then there will be more pressure for $\mathcal{U}_{i::}$ and $\mathcal{U}_{j::}$ to fuse than for $\mathcal{U}_{i'::}$ and $\mathcal{U}_{j'::}$ to fuse as γ increases. Thus, the weight $w_{d,ij}$ quantifies the similarity between the ith and jth mode-d slices. A very large $w_{d,ij}$ indicates that the two slices are very similar, while a very small $w_{d,ij}$ indicates that they are very dissimilar. These pairwise similarities motivate a graphical view of clustering. For the dth mode, define the set $\mathcal{E}_d$ as the edge set of a similarity graph. Each slice is a node in the graph and the set $\mathcal{E}_d$ contains an edge (i, j) if and only if $w_{d,ij} > 0$. Figure 2 shows an example of a mode-1 similarity

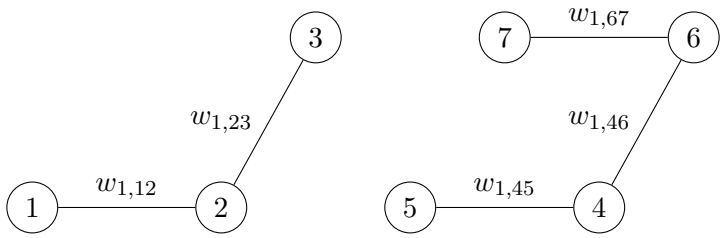


Figure 2: A graph that summarizes the similarities between pairs of the mode-1 subarrays. Only edges with positive weight are drawn.

graph, which corresponds to a tensor with seven mode-1 slices and positive weights that define the edge set

$$\mathcal{E}_1 \quad = \quad \{(1,2),(2,3),(4,5),(4,6),(6,7)\}.$$

Given the connectivity of the graph, as γ increases, the slices $\mathcal{U}_{1::}, \mathcal{U}_{2::}$, and $\mathcal{U}_{3::}$ will be shrunk towards each other while the slices $\mathcal{U}_{4::}, \mathcal{U}_{5::}, \mathcal{U}_{6::}$ and $\mathcal{U}_{7::}$ shrunk towards each other. Since $w_{d,ij} = 0$ for any $(i,j) \notin \mathcal{E}_d$, we can express the penalty terms for the dth mode as

$$R_d(\mathcal{U}) \quad = \quad \sum_{(i,j)\in\mathcal{E}_d} w_{d,ij} \|\mathcal{U}_{i::} - \mathcal{U}_{j::}\|_{\mathrm{F}}.$$

The graph in Figure 2 makes readily apparent that the convex objective in (3) separates over the connected components of the similarity graph for the mode-d slices. Consequently, one can solve for the optimal $\mathcal{U}$ component by component. Without loss of generality, we assume that the weights are such that all the similarity graphs are connected. Before leaving this preliminary description of the weights, however, we want to emphasize that in practice weights are set once in a data-adaptive manner and should be considered empirically chosen hyper-parameters rather than tuning parameters. Further discussion of the weights and practical recommendations for specifying them will be discussed in Section 6.

Having familiarized ourselves with the convex co-clustering of a 3-way array, we now present the natural extension of (3) for clustering the fibers of a general higher-order tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times \cdots \times n_D}$ along all its D modes. Let $\mathbf{\Delta}_{d,ij} = \mathbf{e}_i^{\mathsf{T}} - \mathbf{e}_j^{\mathsf{T}}$ where $\mathbf{e}_i$ is the ith standard basis vector in $\mathbb{R}^{n_d}$. The objective function of our convex co-clustering for a general higher-order tensor is as follows.

$$F_\gamma(\mathcal{U}) \quad = \quad \frac{1}{2}\|\mathcal{X} - \mathcal{U}\|_{\mathrm{F}}^2 + \gamma \sum_{d=1}^{D} \sum_{(i,j)\in\mathcal{E}_d} w_{d,ij} \|\mathcal{U} \times_d \mathbf{\Delta}_{d,ij}\|_{\mathrm{F}}. \tag{4}$$

The difference between the convex triclustering objective (3) and the general convex co-clustering objective (4) is in the penalty terms. Previously in (3) we penalized the difference between pairs slices whereas in (4) we penalize the differences between pairs of mode-d subarrays.

Note that the function $F_\gamma(\mathcal{U})$ defined in (4) has a unique global minimizer. This follows immediately from the fact that $F_\gamma(\mathcal{U})$ is strongly convex. The unique global minimizer of

$F_\gamma(\mathcal{U})$ is our proposed CoCo estimator, which is denoted by $\hat{\mathcal{U}}$ for the remainder of the paper.

At times it will be more convenient to work with vectors rather than tensors. By applying the identity in (1), we can rewrite the objective function in (4) in terms of the vectorizations of $\mathcal{U}$ and $\mathcal{X}$ as follows

$$F_\gamma(\mathbf{u}) \quad = \quad \frac{1}{2}\|\mathbf{x}-\mathbf{u}\|_2^2 + \gamma \sum_{d=1}^{D} \sum_{(i,j)\in\mathcal{E}_d} w_{d,ij}\|\mathbf{A}_{d,ij}\mathbf{u}\|_2. \tag{5}$$

where $\mathbf{A}_{d,ij}$ is the n_{-d}-by-n matrix

$$\mathbf{A}_{d,ij} \quad = \quad \mathbf{I}_{n_D} \otimes \cdots \otimes \mathbf{I}_{n_{d+1}} \otimes \mathbf{\Delta}_{d,ij} \otimes \mathbf{I}_{n_{d-1}} \otimes \cdots \otimes \mathbf{I}_{n_1} \tag{6}$$

where $\mathbf{I}_{n_d}$ is the n_d-by-n_d identity matrix. We will refer to the unique global minimizer of (5), $\hat{\mathbf{u}} = \arg\min_{\mathbf{u}} F_\gamma(\mathbf{u})$, as the vectorized version of our CoCo estimator.

Remark 1 *The fusion penalties $R_d(\mathcal{U})$ are a composition of the group lasso (Yuan and Lin, 2006) and the fused lasso (Tibshirani et al., 2005), a special case of the generalized lasso (Tibshirani and Taylor, 2011). When only a single mode is being clustered and only one of the terms $R_d(\mathcal{U})$ is employed, we recover the objective function in the convex clustering problem (Pelckmans et al., 2005; She, 2010; Lindsten et al., 2011; Hocking et al., 2011; Sharpnack et al., 2012; Zhu et al., 2014; Chi and Lange, 2015; Radchenko and Mukherjee, 2017). Most prior work on convex clustering employ an element-wise ℓ_1-norm penalty on pairwise differences, as in the original fused lasso, however, ℓ_2-norm and ℓ_∞-norm have also been considered (Hocking et al., 2011; Chi and Lange, 2015). In this paper, we restrict ourselves to the ℓ_2-norm for two reasons. First, the ℓ_2-norm is rotationally invariant. In general, we are reluctant to adopt a procedure whose co-clustering output may non-trivially change when the coordinate representation of the data along one of its modes is trivially changed. Second, the ℓ_2-norm promotes the group-wise shrinkage of pairwise differences of subarrays along each mode leading to more straightforward partitioning along each mode. Pairwise differences are either exactly zero or not. When the tensor is a matrix and the rows and columns are being simultaneously clustered, we recover the objective function in the convex biclustering problem (Chi et al., 2017). In general, the fusion penalties $R_d(\mathcal{U})$ shrink solutions to vector valued functions that are piece-wise constant over the mode-d similarity graph defined by the weights $w_{d,ij}$. Viewed this way, we can see our approach as simultaneously performing the network lasso (Hallac et al., 2015) on D similarity graphs.*

Remark 2 *The CoCo estimator is invariant to permutations in the data tensor $\mathcal{X}$ in the following sense. Suppose $\hat{\mathcal{U}}$ and $\hat{\mathcal{U}}'$ are the CoCo estimators when the data tensors are respectively $\mathcal{X}$ and $\mathcal{X}' = \mathcal{X}\times_1\mathbf{\Pi}_1\times_2\cdots\times_D\mathbf{\Pi}_D$ where $\mathbf{\Pi}_1 \in \{0,1\}^{n_1\times n_1}, \ldots, \mathbf{\Pi}_D \in \{0,1\}^{n_D\times n_D}$ are permutation matrices, namely $\mathbf{\Pi}_d^\mathsf{T}\mathbf{\Pi}_d = \mathbf{I}$. In words, $\mathcal{X}'$ can be obtained from $\mathcal{X}$ by permuting the subarrays of $\mathcal{X}$ along the dth mode according to $\mathbf{\Pi}_d$ for $d = 1, \ldots, D$, and $\mathcal{X}$ can be recovered from $\mathcal{X}'$ by permuting along the dth mode according to $\mathbf{\Pi}_d^\mathsf{T}$ for $d = 1, \ldots, D$. Since $\|\mathcal{U}\times_1\mathbf{\Pi}_1\times_2\cdots\times_D\mathbf{\Pi}_D\|_F = \|\mathcal{U}\|_F$, it follows that*

$$\hat{\mathcal{U}}' \quad = \quad \hat{\mathcal{U}}\times_1\mathbf{\Pi}_1\times_2\cdots\times_D\mathbf{\Pi}_D \quad \textit{and} \quad \hat{\mathcal{U}} \quad = \quad \hat{\mathcal{U}}'\times_1\mathbf{\Pi}_1^\mathsf{T}\times_2\cdots\times_D\mathbf{\Pi}_D^\mathsf{T}.$$

Permutation invariance is important because it means that the CoCo estimator is essentially unaltered by any reshuffling along the modes of the data tensor.

Remark 3 *Given the co-clustering structure assumed in* (2)*, one may wonder how much is added by explicitly seeking a co-clustering over clustering along each mode independently. In other words, why not solve D independent convex clustering problems with $R(\mathcal{U}) = R_d(\mathcal{U})$? To provide some intuition on why co-clustering should be preferred over independently clustering each mode, consider the following problem. Imagine trying to cluster row vectors $\mathbf{x}_i \in \mathbb{R}^{10,000}$ for $i = 1, \ldots, 100$ drawn from a two-component mixture of Gaussians, namely*

$$\mathbf{x}_i \overset{iid}{\sim} \frac{1}{2}N(\boldsymbol{\mu}, \sigma^2\mathbf{I}) + \frac{1}{2}N(\boldsymbol{\nu}, \sigma^2\mathbf{I}).$$

This is a challenging clustering problem due to the disproportionately small number of observations compared to the number of features. If, however, we were told that $\mu_j = \mu_1$ and $\nu_j = \nu_1$ for $j = 1, \ldots, 5,000$ and $\mu_j = \mu_2$ and $\nu_j = \nu_2$ for $i = 5,001, \ldots, 10,000$, in other words that the features were clustered into two groups, our fortunes have reversed and we now have an abundance of observations compared to the number of effective features. Even if we lack a clear-cut clustering structure in the features, this example suggests that leveraging similarity structure along the columns can expedite identifying similarity structure along the rows, and vice versa. Indeed, if there is an underlying checkerbox mean tensor we may expect that simultaneously clustering along each mode should make the task of clustering along any one given mode easier. Our prediction error result presented in Section 4.2 in fact supports this suspicion (See Remark 10).

4. Properties

We first discuss how the CoCo estimator $\hat{\mathcal{U}}$ behaves as a function of the data tensor $\mathcal{X}$, the tuning parameter γ, and the weights $w_{d,ij}$. We will then present its statistical properties under mild conditions on the data generating process. We highlight that these properties hold regardless of the algorithm used to minimize (4), as they are intrinsic to its convex formulation. All proofs are given in Appendix B and Appendix C.

4.1 Stability Properties

The CoCo estimator varies smoothly with respect to $\mathcal{X}$, γ, and $\{w_{d,ij}\}$. Let $\mathbf{W}_d = \{w_{d,ij}\}$ denote the weights matrix for mode d.

Proposition 4 *The minimizer $\hat{\mathcal{U}}$ of* (4) *is jointly continuous in $(\mathcal{X}, \gamma, \mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_D)$.*

As noted earlier, in practice we will typically fix the weights $w_{d,ij}$ and compute the CoCo estimator over a grid of the penalization parameters γ in order to select a final CoCo estimator from among the computed candidate estimators of varying levels of smoothness. Since (4) does not admit a closed form minimizer, we resort to iterative algorithms for computing the CoCo estimator. Continuity of $\hat{\mathcal{U}}$ in γ can be leveraged to expedite computation through warm starts, namely using the solution $\hat{\mathcal{U}}_\gamma$ as the initial guess for iteratively computing $\hat{\mathcal{U}}_{\gamma'}$ where γ' is slightly larger or smaller than γ. Due to the continuity of $\hat{\mathcal{U}}$ in γ, small changes

in γ will result in small changes in $\hat{\mathcal{U}}$. Empirically the use of warm starts can lead to a non-trivial reduction in computation time (Chi and Lange, 2015). From the continuity in γ, we also see that convex co-clustering performs continuous co-clustering just as the lasso (Tibshirani, 1996) performs continuous variable selection.

The penalization parameter γ tunes the complexity of the CoCo estimator. Clearly when $\gamma = 0$, the CoCo estimator coincides with the data tensor, namely $\hat{\mathcal{U}} = \mathcal{X}$. The key to understanding the CoCo estimator's behavior as γ increases is to recognize that the penalty functions $R_d(\mathcal{U})$ are semi-norms. Under suitable conditions on the weights given in Assumption 4.1 below, $R_d(\mathcal{U})$ vanishes if and only if the mode-d subarrays of $\mathcal{U}$ are identical.

Assumption 4.1 *For any pair of mode-d subarrays, indexed by i and j with $i < j$, there exists a sequence of indices $i \to k \to \cdots \to l \to j$ along which the weights, $w_{d,ik}, \ldots, w_{d,lj}$ are positive.*

Proposition 5 *Under Assumption 4.1, $R_d(\mathcal{U}) = 0$ if and only if $\mathbf{U}_{(d)} = \mathbf{1}\mathbf{c}^\mathsf{T}$ for some $\mathbf{c} \in \mathbb{R}^{n-d}$.*

To give some intuition for Proposition 5, note that the term $R_d(\mathcal{U})$ separates over the connected components of the mode-d similarity graph. Therefore, the term $R_d(U)$ penalizes variation in the mode-d subarrays over the connected components of the mode-d similarity graph. Assumption 4.1, states that the mode-d similarity graph is connected. Thus, the only way for $R_d(U)$ to attain its minimum value and vanish under Assumption 4.1, is if there is no variation in $\mathcal{U}$ along its mode-d subarrays.

Proposition 5 suggests that if Assumption 4.1 holds for all $d = 1, \ldots, D$ then as γ increases the CoCo estimator converges to the solution of the following constrained optimization problem:

$$\min_{\mathbf{u}} \frac{1}{2}\|\mathbf{x} - \mathbf{u}\|_{\mathrm{F}}^2 \quad \text{subject to } \mathbf{u} = c\mathbf{1} \text{ for some } c \in \mathbb{R},$$

the solution to which is just the global mean $\bar{\mathbf{x}}$, whose entries are all identically the average value of $\mathbf{x}$ over all its entries. The next result formalizes our intuition that as γ increases, the CoCo estimator will eventually coincide with $\bar{\mathbf{x}}$.

Proposition 6 *Suppose Assumption 4.1 holds for $d = 1, \ldots, D$, then $F_\gamma(\mathcal{U})$ is minimized by the grand mean $\bar{\mathcal{X}}$ for γ sufficiently large.*

Thus, as γ increases from 0, the CoCo estimator $\hat{\mathcal{U}}$ traces a continuous solution path that starts from n co-clusters, consisting of $u_{i_1 \cdots i_D} = x_{i_1 \cdots i_D}$, to a single co-cluster, where $u_{i_1 \cdots i_D} = \mathbf{x}^\mathsf{T}\mathbf{1}/n$ for all $i_1, \ldots, i_D$.

For a fixed γ, we can derive an explicit bound on sensitivity of the CoCo estimator to perturbations in the data.

Proposition 7 *The minimizer $\hat{\mathcal{U}}$ of (4) is a nonexpansive or 1-Lipschitz function of the data tensor $\mathcal{X}$, namely*

$$\|\hat{\mathcal{U}}(\mathcal{X}) - \hat{\mathcal{U}}(\tilde{\mathcal{X}})\|_{\mathrm{F}} \quad \leq \quad \|\mathcal{X} - \tilde{\mathcal{X}}\|_{\mathrm{F}}.$$

Nonexpansivity of $\hat{\mathcal{U}}$ in $\mathcal{X}$ provides an attractive stability result. Since $\hat{\mathcal{U}}$ varies smoothly with the data, small perturbations in the data are guaranteed to not lead to large variability of $\hat{\mathcal{U}}$, or consequently large variability in the cluster assignments. In a special case of our method, Chi et al. (2017) showed empirically that the co-clustering assignments made by the 2-way version of the CoCo estimator was noticeably less sensitive to perturbations in the data than those made by several existing biclustering algorithms.

4.2 Statistical Properties

We next provide a finite sample bound for the prediction error of the CoCo estimator. For simplicity, we consider the case where we take uniform weights within a mode in (5), namely $w_{d,ij} = w_{d,i'j'} = 1/n_d$ for all $i, j, i', j' \in \{1, \ldots, n_d\}$. Such uniform weight assumption has also been imposed in the analysis of the vector-version of convex clustering (Tan and Witten, 2015).

In order to derive the estimation error of $\hat{\mathbf{u}}$, we first define an important definition for the noise and introduce two regularity conditions.

Definition 8 (Vu and Wang (2015)) *We say a random vector $\mathbf{y} \in \mathbb{R}^n$ is M-concentrated if there are constants $C_1, C_2 > 0$ such that for any convex, 1-Lipschitz function $\phi : \mathbb{R}^n \to \mathbb{R}$ and any $t > 0$,*

$$\mathbb{P}\Big(\big|\phi(\mathbf{y}) - \mathbb{E}[\phi(\mathbf{y})]\big| \geq t\Big) \quad \leq \quad C_1 \exp\left(-\frac{C_2 t^2}{M^2}\right).$$

The M-concentrated random variable is more general than the Gaussian or sub-Gaussian random variables, and it allows dependence in its coordinates. Vu and Wang (2015) provided a few examples of M-concentrated random variables. For instance, if the coordinates of $\mathbf{y}$ are iid standard Gaussian, then $\mathbf{y}$ is 1-concentrated. If the coordinates of $\mathbf{y}$ are independent and M-bounded, then $\mathbf{y}$ is M-concentrated. If the coordinates of $\mathbf{y}$ come from a random walk with certain mixing properties, then $\mathbf{y}$ is M-concentrated for some M.

Assumption 4.2 (Model) *We assume the true cluster center $\mathcal{C}^* \in \mathbb{R}^{k_1 \times \cdots \times k_D}$ has a checkerbox structure such that the mode-d subarrays have k_d different values (number of clusters along the dth mode), and each entry of $\mathcal{C}^*$ is bounded above by a constant $C_0 > 0$. Define $\mathcal{U}^* \in \mathbb{R}^{n_1 \times \cdots \times n_D}$ as the true parameter expanded based on $\mathcal{C}^*$, namely*

$$\mathcal{U}^* \quad = \quad \mathcal{C}^* \times_1 \mathbf{M}_1 \times_2 \mathbf{M}_2 \times_3 \cdots \times_D \mathbf{M}_D,$$

where $\mathbf{M}_d \in \{0,1\}^{n_d \times k_d}$ are binary mode-d cluster membership matrices such that $\mathbf{M}_d \mathbf{1} = 1$. Denote $\mathbf{u}^* = \mathrm{vec}(\mathcal{U}^*) \in \mathbb{R}^n$ with $n = \prod_{d=1}^D n_d$. We assume the samples belonging to the $(r_1, \ldots, r_D)$-th cluster satisfy

$$x_{i_1,\ldots,i_D} \quad = \quad {c^*}_{r_1,\ldots,r_D} + \epsilon_{i_1,\ldots,i_D},$$

with $i_d \in \{1, \ldots, n_d\}$ and $r_d \in \{1, \ldots, k_d\}$. Furthermore, we assume $\boldsymbol{\epsilon} = \mathrm{vec}(\mathcal{E})$ is a M-concentrated random variable defined in (8) with mean zero.

The checkerbox means model in Assumption 4.2 provides the underlying cluster structure of the tensor data. As a special case, Assumption 4.2 with $D = 2$ reduces to the model assumption underlying convex biclustering (Chi et al., 2017). In contrast to the independent sub-Gaussian condition assumed in vector-version convex clustering (Tan and Witten, 2015), our error condition is much weaker since we allow for non-sub-Gaussian distributions as well as allow for dependence among its coordinates.

Assumption 4.3 (Tuning) *The tuning parameter γ satisfies*

$$\frac{2\log(n)\sqrt{n}}{D} \;\leq\; \gamma \;\leq\; \frac{2c_0\log(n)\sqrt{n}}{D},$$

for some constant $c_0 > 1$.

Theorem 9 *Suppose that Assumption* 4.2 *and Assumption* 4.3 *hold. The estimation error of $\hat{\mathbf{u}}$ in* (5) *with uniform weights satisfies,*

$$\frac{1}{n}\left\|\hat{\mathbf{u}} - \mathbf{u}^*\right\|_2^2 \;\leq\; \frac{1}{D}\sum_{d=1}^{D}\left(\frac{1}{n_d} + \frac{\log(n)}{\sqrt{nn_d}}\right) + \frac{C\log(n)}{D\sqrt{n}}\sum_{d=1}^{D} n_d\sqrt{\prod_{j\neq d} k_j}, \tag{7}$$

with a high probability, where $C = 12c_0C_0^2$ is a positive constant, and k_d is the true number of clusters in the dth mode.

Theorem 9 provides a finite sample error bound for the proposed CoCo tensor estimator. Our theoretical bound allows the number of clusters in each mode to diverge, which reflects a typical large-scale clustering scenario in big tensor data. A notable consequence of Theorem 9 is that, when $D \geq 3$, namely a higher-order tensor with at least 3 modes, the CoCo estimator can achieve estimation consistency along all the D modes even when we only have one tensor sample. Here the sample size refers to the number of available tensor samples. In our tensor clustering problem, we only have access to one tensor sample.

This property is uniquely enjoyed by co-clustering of tensor data with $D \geq 3$, and has not been previously established in the existing literature on vector clustering or biclustering. To see this, when n_d are of the same order as n_0, and k_d are of the same order as k_0, a sufficient condition for the consistency is that $n_0 \to \infty$ and $k_0 = o\big(n_0^{(D-2)/(D-1)}\big)$ up to a log term. When $D = 3$, the CoCo estimator is consistent so long as the number of clusters k_0 in each mode diverges slightly slower than $\sqrt{n_0}$. Remarkably, as we have more modes in the tensor data, this constraint on the rate of divergence of k_0 gets weaker. In short, we reap a unique and surprisingly welcome "blessing of dimensionality" phenomenon in the tensor co-clustering problem.

Remark 10 *Next we discuss the connections of our bound* (7) *with prior results in the literature. An intermediate step in the proof of Theorem 9 indicates that the estimation error in the dth mode is on the order of $1/n_d + \log(n)/\sqrt{nn_d} + \log(n)\sqrt{n_d\prod_{j\neq d} k_j/n_{-d}}$. In the clustering along the rows of a data matrix, our rate matches with that established for vector-version convex clustering (Tan and Witten, 2015), up to a log term $\sqrt{\log(n)}$. Such*

a log term is due to that fact that Tan and Witten (2015) considers the error to be iid sub-Gaussian while we consider a general M-concentrated error. In practice, the iid assumption on the noise $\boldsymbol{\epsilon} = \mathrm{vec}(\mathcal{E})$ could be restrictive. Consequently, our theoretical analysis is built upon a new concentration inequality of quadratic forms recently developed in Vu and Wang (2015). In addition, our rate reveals an interesting theoretical property of the convex bi-clustering method proposed by Chi et al. (2017). When $D = 2$, our rate indicates that the estimation error along the row and column of the data matrix is $\log(n_1 n_2)\sqrt{n_1 k_2/n_2}$ and $\log(n_1 n_2)\sqrt{n_2 k_1/n_1}$, respectively. Clearly, both errors can not converge to zero simultaneously. This indicates a disadvantage of matricizing a data tensor for co-clustering.

5. Estimation Algorithm

We next discuss a simple first order method for computing the solution to the convex co-clustering problem. The proposed algorithm generalizes the variable splitting approach introduced for convex clustering problem described in Chi and Lange (2015) to the CoCo problem. The key observation is that the Lagrangian dual of an equivalent formulation of the convex co-clustering problem is a constrained least squares problem that can be iteratively solved using the classic projected gradient algorithm.

5.1 A Lagrangian Dual of the CoCo Problem

Recall that we seek to minimize the objective function in (5)

$$F_\gamma(\mathbf{u}) \;=\; \frac{1}{2}\|\mathbf{x}-\mathbf{u}\|_2^2 + \gamma \sum_{d=1}^{D}\sum_{l\in\mathcal{E}_d} w_{d,l}\|\mathbf{A}_{d,l}\mathbf{u}\|_2.$$

Note that we have enumerated the edge indices in $\mathcal{E}_d$ to simplify the notation for the following derivation.

We perform variable splitting and introduce the dummy variables $\mathbf{v}_{d,l} = \mathbf{A}_{d,l}\mathbf{u}$. Let $\mathbf{V}_d$ denote the $n_{-d}\times|\mathcal{E}_d|$ matrix whose lth column is $\mathbf{v}_{d,l}$. Further denote the vectorization of $\mathbf{V}_d$ by $\mathbf{v}_d = \mathrm{vec}(\mathbf{V}_d)$ and let $\mathbf{v} = \begin{bmatrix}\mathbf{v}_1^\mathsf{T} & \mathbf{v}_2^\mathsf{T} & \cdots & \mathbf{v}_D^\mathsf{T}\end{bmatrix}^\mathsf{T}$ denote the vector obtained by stacking the vectors $\mathbf{v}_d$ on top of each other. We now solve the equivalent equality constrained minimization

$$\min_{\mathbf{v},\mathbf{u}} \frac{1}{2}\|\mathbf{x}-\mathbf{u}\|_2^2 + \gamma \sum_{d=1}^{D}\sum_{l\in\mathcal{E}_d} w_{d,l}\|\mathbf{v}_{d,l}\|_2 \qquad \text{subject to} \qquad \mathbf{v}_d = \mathbf{A}_d\mathbf{u},$$

where $\mathbf{A}_d = (\mathbf{I}_{n_D}\otimes\cdots\otimes\mathbf{I}_{n_{d+1}}\otimes\boldsymbol{\Phi}_d\otimes\mathbf{I}_{n_{d-1}}\otimes\cdots\otimes\mathbf{I}_{n_1})$ and $\boldsymbol{\Phi}_d$ is the oriented edge-vertex incidence matrix for the dth mode graph, namely

$$\Phi_{d,lv} = \begin{cases} 1 & \text{If node } v \text{ is the head of edge } l \\ -1 & \text{If node } v \text{ is the tail of edge } l \\ 0 & \text{otherwise.}\end{cases}$$

We introduce dual variables $\boldsymbol{\lambda}_d$ corresponding to the equality constraint $\mathbf{v}_d = \mathbf{A}_d\mathbf{u}$. Let $\boldsymbol{\Lambda}_d$ denote the $n_{-d}\times|\mathcal{E}_d|$ matrix whose lth column is $\boldsymbol{\lambda}_{d,l}$. Further denote the vectorization

of $\mathbf{\Lambda}_d$ by $\boldsymbol{\lambda}_d = \text{vec}(\mathbf{\Lambda}_d)$ and $\boldsymbol{\lambda} = \begin{bmatrix}\boldsymbol{\lambda}_1^\mathsf{T} & \boldsymbol{\lambda}_2^\mathsf{T} & \cdots & \boldsymbol{\lambda}_D^\mathsf{T}\end{bmatrix}^\mathsf{T}$. The Lagrangian dual objective is given by

$$G(\boldsymbol{\lambda}) \;=\; \frac{1}{2}\|\mathbf{x}\|_2^2 - \frac{1}{2}\|\mathbf{x} - \mathbf{A}^\mathsf{T}\boldsymbol{\lambda}\|_2^2 - \sum_{d=1}^{D}\sum_{l\in\mathcal{E}_d} \iota_{C_{d,l}}(\boldsymbol{\lambda}_{d,l}),$$

where $\mathbf{A} = \begin{bmatrix}\mathbf{A}_1^\mathsf{T} & \mathbf{A}_2^\mathsf{T} & \cdots & \mathbf{A}_D^\mathsf{T}\end{bmatrix}^\mathsf{T}$ and $\iota_{C_{d,l}}$ is the indicator function of the closed convex set $C_{d,l} = \{\mathbf{z} : \|\mathbf{z}\|_2 \leq \gamma w_{d,l}\}$, namely $\iota_{C_{d,l}}$ is the function that vanishes on the set of $C_{d,l}$ and is infinity on the complement of $C_{d,l}$. Details on the derivation of the dual objective $G(\boldsymbol{\lambda})$ are provided in Appendix D.

Maximizing the dual objective $G(\boldsymbol{\lambda})$ is equivalent to solving the following constrained least squares problem:

$$\min_{\boldsymbol{\lambda}\in C} \frac{1}{2}\|\mathbf{x} - \mathbf{A}^\mathsf{T}\boldsymbol{\lambda}\|_2^2, \tag{8}$$

where $C = \{\boldsymbol{\lambda} : \boldsymbol{\lambda}_{d,l} \in C_{d,l}, l \in \mathcal{E}_d, d = 1, \ldots, D\}$. We can recover the primal solution via the relationship:

$$\hat{\mathbf{u}} \;=\; \mathbf{x} - \mathbf{A}^\mathsf{T}\hat{\boldsymbol{\lambda}},$$

where $\hat{\boldsymbol{\lambda}}$ is a solution to the dual problem (8). The dual problem (8) has at least one solution by the Weierstrass extreme value theorem, but the solution may not be unique since $\mathbf{A}^\mathsf{T}$ has a non-trivial kernel. Nonetheless, our CoCo estimator $\hat{\mathbf{u}}$ is still unique since $\mathbf{A}^\mathsf{T}\hat{\boldsymbol{\lambda}}_1 = \mathbf{A}^\mathsf{T}\hat{\boldsymbol{\lambda}}_2$ for any solutions $\hat{\boldsymbol{\lambda}}_1, \hat{\boldsymbol{\lambda}}_2$ to the problem (8).

We numerically solve the constrained least squares problem in (8) with the projected gradient algorithm, which alternates between taking a gradient step and projecting onto the set C. Algorithm 1 provides pseudocode of the projected gradient algorithm, which has several good features. The projected gradient algorithm is guaranteed to converge to a global minimizer of (8). Its per-iteration and storage costs using the weight choices, described in Section 6, are both $\mathcal{O}(Dn)$, namely linear in either the number of dimensions D or in the number of elements n. For a modest additional computational and storage cost, we can accelerate the projected gradient method, for example with FISTA (Beck and Teboulle, 2009) or SpaRSA (Wright et al., 2009). In our experiments, we use a version of the latter, namely FASTA (Goldstein et al., 2014, 2015). Additional details on the derivation of the algorithmic updates, convergence guarantees, computational and storage costs, as well as stopping rules can be found in Appendix E.

6. Specifying Non-Uniform Weights

In Section 4.2, we assumed uniform weights $w_{d,ij}$ in the penalty terms $R_d(\mathfrak{U})$ to establish a prediction error bound, which revealed a surprising and beneficial "blessing of dimensionality" phenomenon. Although this simplifying assumption gives clarity and insight into how the co-clustering problem gets easier as the number of modes increases, in practice choosing non-uniform weights can substantially improve the quality of the clustering results. In the context of convex clustering, Chen et al. (2015) and Chi and Lange (2015) provided

Algorithm 1 Convex Co-Clustering (CoCo) Estimation Algorithm

Initialize $\boldsymbol{\lambda}^{(0)}$; for $m = 0, 1, \ldots$

repeat

  $\mathbf{u}^{(m+1)} = \mathbf{x} - \mathbf{A}^\top \boldsymbol{\lambda}^{(m)}$ ▷ Gradient Step

  for $d = 1, \ldots, D$ **do**

    for $l \in \mathcal{E}_d$ **do**

      $\boldsymbol{\lambda}_{d,l}^{(m+1)} = \mathcal{P}_{C_{d,l}} \left(\boldsymbol{\lambda}_{d,l}^{(m)} + \eta \mathbf{A}_{d,l} \mathbf{u}^{(m+1)} \right)$ ▷ Projection Step

    end for

  end for

until convergence

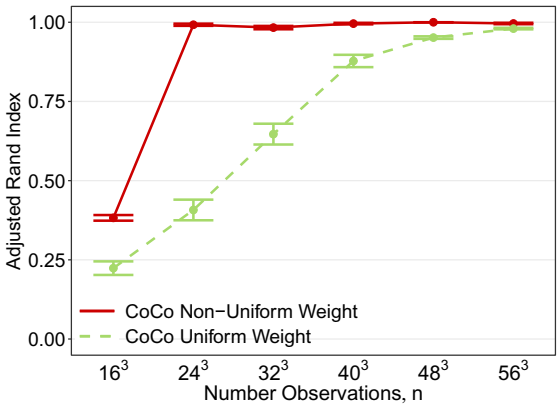


Figure 3: Uniform versus non-uniform weights: Average Adjusted Rand Index for an increasing size. Here $n = n_0^3$ refers to a tensor of size $n_0 \times n_0 \times n_0$.

empirical evidence that convex clustering with uniform weights struggled to produce exact sparsity in the pairwise differences of smooth estimates when there was not a strong separation between groups. Indeed, similar phenomena were observed in earlier work on the related clustered lasso (She, 2010). Several related works (She, 2010; Hocking et al., 2011; Chen et al., 2015; Chi and Lange, 2015) recommend a weight assignment strategy described below. In addition, the use of *sparse* weights can also lead to non-trivial improvements in both computational time and clustering performance (Chi and Lange, 2015; Chi et al., 2017).

To illustrate the practical value of non-uniform weights, we compare CoCo's ability to recover co-clusters, using both uniform and non-uniform weights, as the size of a 3-way tensor increases when there are two clusters per mode with balanced cluster sizes along each mode. We assess the quality of the recovered clustering performance using the Adjusted Rand Index (ARI). The ARI (Hubert and Arabie, 1985) varies between -1 and 1, where 1 indicates a perfect match between two clustering assignments whereas a value close to zero indicates the two clustering assignments match about as might be expected if they were both randomly generated. Negative values indicate that there is less agreement between clusterings than expected from random partitions.

Figure 3 shows a comparison between using non-uniform weights that are described in Section 6.2 and uniform weights. Each plotted point in Figure 3 is the average ARI over 100 replicates. For CoCo using non-uniform weights, the smoothing parameter γ is chosen with the data-driven extended BIC method that is detailed in Section 7.1. In contrast, for CoCo using uniform weights, γ is chosen as the value that produces the estimator that minimizes the true but unknown MSE.

We see that while using uniform weights in CoCo leads to recovering co-clusters exactly once a sufficient number of samples have been acquired, using non-uniform weights enables CoCo to recover the co-clusters exactly with notably fewer samples. The results of this experiment are especially remarkable because CoCo using non-uniform weights and a *data-adaptive* choice of γ outperformed CoCo using uniform weights and an ideally chosen *oracle* value of γ.

As in the case of convex clustering, using non-uniform weights can lead to significantly better performance over using uniform weights in practice. We give some explanation for why this is expected in Section 6.3 but leave it to future work to develop theory proving this performance improvement. Nonetheless based on this observation, we employ non-uniform weights in CoCo for the empirical studies presented later in the paper.

6.1 Basic Procedure for Specifying Weights

We first describe our basic two step procedure for constructing weights before elaborating on the final refinements used in our numerical experiments.

Step 1: We first calculate pre-weights $\tilde{w}_{d,ij}$ between the ith and jth mode-d subarrays as

$$\tilde{w}_{d,ij} \quad = \quad \iota^k_{\{i,j\}} \exp\left(-\tau_d \|\mathbf{X}_{(d),i:} - \mathbf{X}_{(d),j:}\|_{\mathrm{F}}^2\right). \tag{9}$$

The first factor on the right hand side of equation (9), $\iota^k_{\{i,j\}}$, is an indicator function that equals 1 if the jth slice is among the ith slice's k-nearest neighbors (or vice versa) and 0 otherwise. The purpose of this term is to control the sparsity of the weights. The corresponding tuning parameter k influences the connectivity of the mode-d similarity graph. One can explore different levels of granularity in the clustering by varying k (Chen et al., 2015). As a default, one can use the smallest k such that the similarity graph is still connected. Note it is not necessary to calculate the exact k-nearest neighbors, which scales quadratically in the number of fibers in the mode. A fast approximation to the k-nearest neighbors is sufficient for the sake of inducing sparsity into the weights. Chi and Lange (2015) provided two reasons for using k-nearest neighbor weights. First, we wish to prioritize fusions between pairs of subarrays that are most similar; the subarrays that are most dissimilar should be the last pair of subarrays to fuse as the smoothing parameter γ increases. Second, we wish to use a sparse similarity graph as the computational and storage complexity of the estimation algorithm is proportional to the number of non-zero edges in the similarity graphs (Appendix E). Using k-nearest-neighbors weights accomplishes both goals.

The second factor on the right hand side of equation (9) is the Gaussian kernel, which takes on larger values for pairs of mode-d subarrays that are more similar to each other. Chi and Steinerberger (2019) give a detailed theoretical justification for using weights like the Gaussian kernel weights in the context of convex clustering. For space considerations,

we refer readers interested in these technical details to their work and give a brief intuitive rationale for the employing the Gaussian kernel here. Intuitively, the weights should be inversely proportional to the distance between the ith and jth mode-d subarrays (Chen et al., 2015; Chi et al., 2017). The inverse of the nonnegative parameter τ_d is a measure of scale. In practice, we can set it to be the median Euclidean distance between the ith and jth mode-d subarrays that are k-nearest neighbors of each other. A value of $\tau_d = 0$ corresponds to uniform weights. Note that with minor modification, we can make the inverse scale parameter to be pair dependent as described in Zelnik-Manor and Perona (2005).

Step 2: To obtain the mode-d weights $w_{d,ij}$, we normalize the mode-d pre-weights $\tilde{w}_{d,ij}$ to sum to $\sqrt{n_d/n}$. The normalization step puts the penalty terms $R_d(\mathbf{U})$ on the same scale and ensures that clustering along any given single mode will not dominate the entire co-clustering as γ increases.

6.2 Improving Weights via the Tucker Decomposition

In our preliminary experiments, we found that substituting a low-rank approximation of $\mathbf{X}$, namely a Tucker decomposition $\tilde{\mathbf{X}}$, in place of $\mathbf{X}$ in (9) led to a marked improvement in co-clustering performance. To understand the boost in performance suppose that $\mathbf{X} = \mathbf{U}^* + \mathcal{E}$ with $\mathbf{U}^*$ having a checkerbox structure and the entries of $\mathcal{E}$ are iid $N(0, \sigma^2)$ for simplicity. Further suppose that the ith and jth mode-d subarrays of $\mathbf{U}^\star$ belong to the same partition and $\iota^k_{\{i,j\}} = 1$. Then

$$\tilde{w}_{d,ij} \quad = \quad \exp\left(-\tau_d \|\mathcal{E} \times_d \mathbf{\Delta}_{ij}\|_{\mathrm{F}}^2\right) \quad = \quad \exp\left(-2\tau_d \sigma^2 Z_{d,ij}\right),$$

where $Z = \frac{\|\mathcal{E} \times_d \mathbf{\Delta}_{ij}\|_{\mathrm{F}}^2}{2\sigma^2}$ is distributed as a χ^2 random variable with n_d degrees of freedom. If we were able to perfectly denoise the tensor $\mathbf{X}$ so that $\sigma = 0$, then the pre-weight $\tilde{w}_{d,ij}$ would be set to its maximal value of 1, the ideal value for $\tilde{w}_{d,ij}$ since we have assumed the ith and jth mode-d subarrays belong to the same partition. Thus, if we can reduce σ^2, namely denoise the observed tensor $\mathbf{X}$, we can approach the ideal value of pre-weights. Note that we are more focused with approaching the ideal pre-weight values for pairs of subarrays that belong to the same partition and not concerned with pairs of subarrays in different partitions as the Gaussian kernel weights decay very rapidly. The Tucker decomposition is effective at reducing σ^2 when $\mathbf{U}^*$ has a checkerbox pattern as the checkerbox pattern is a low-rank tensor that can be effectively approximated with the Tucker decomposition.

Employing the Tucker decomposition introduces another tuning parameter, namely the rank of the decomposition. In our simulation studies described in Section 8, we use two different methods for choosing the rank as a robustness check to ensure our CoCo estimator's performance does not crucially depend on the rank selection method. Details on these two methods can be found in Appendix F. While we found the Tucker decomposition to work well in practice, we suspect that other methods of denoising the tensor may work just as well or could possibly be more effective. We leave it to future work to explore alternatives to the Tucker decomposition.

6.3 Weights and Folded-Concave Penalties

We conclude our discussion on weights by highlighting how they provide a connection between convex clustering and other penalized regression-based clustering methods that use folded-concave penalties (Pan et al., 2013; Xiang et al., 2013; Zhu et al., 2013; Marchetti and Zhou, 2014; Wu et al., 2016a). Suppose we seek to minimize the objective

$$\tilde{f}_\gamma(\mathbf{u}) \quad = \quad \frac{1}{2}\|\mathbf{x}-\mathbf{u}\|_2^2 + \gamma \sum_{d=1}^{D} \sum_{(i,j)\in\mathcal{E}_d} \varphi_d\left(\|\mathbf{A}_{d,ij}\mathbf{u}\|_2\right), \tag{10}$$

where each $\varphi_d : [0,\infty) \mapsto [0,\infty)$ has the following properties: (i) φ_d is concave and differentiable on $(0,\infty)$, (ii) φ_d vanishes at the origin, and (iii) the directional derivative of φ_d exists and is positive at the origin. Such φ_d is collectively referred to as a folded-concave penalty; prominent examples of such function include the smoothly clipped absolute deviation (Fan and Li, 2001) or minimax concave penalty (Zhang, 2010).

Since φ_d is concave and differentiable, for all positive z and $\tilde{z}$

$$\varphi_d(z) \quad \leq \quad \varphi_d(\tilde{z}) + \varphi_d'(\tilde{z})(z-\tilde{z}). \tag{11}$$

The inequality (11) indicates that the first order Taylor expansion of a differentiable concave function φ_d provides a tight global upper bound at the expansion point $\tilde{z}$. Thus, we can construct a function that is a tight upper bound of the function $\tilde{f}_\gamma(\mathbf{u})$

$$g_\gamma(\mathbf{u} \mid \tilde{\mathbf{u}}) \quad = \quad \frac{1}{2}\|\mathbf{x}-\mathbf{u}\|_2^2 + \gamma \sum_{d=1}^{D} \sum_{(i,j)\in\mathcal{E}_d} w_{d,ij}\|\mathbf{A}_{d,ij}\mathbf{u}\|_2 + c, \tag{12}$$

where the constant c does not depend on $\mathbf{u}$ and $w_{d,ij}$ are weights that depend on $\tilde{\mathbf{u}}$, namely

$$w_{d,ij} \quad = \quad \varphi_d'\left(\|\mathbf{A}_{d,ij}\tilde{\mathbf{u}}\|_2\right). \tag{13}$$

Note that if we take $\tilde{\mathbf{u}}$ to be the vectorization of the Tucker approximation of the data, $\text{vec}(\tilde{\boldsymbol{\mathcal{X}}})$, and $\varphi_d(z)$ to be the following variation on the error function

$$\varphi_d(z) \quad = \quad \frac{1}{\sqrt{n_{-d}}\sum_{(i,j)\in\mathcal{E}_d} w_{d,ij}} \int_0^z e^{-\tau_d \omega^2} d\omega,$$

then the function given in (10) coincides with the CoCo objective using the prescribed Tucker derived Gaussian kernel weights.

The function $g_\gamma(\mathbf{u} \mid \tilde{\mathbf{u}})$ is said to majorize the function $\tilde{f}_\gamma(\mathbf{u})$ at the point $\tilde{\mathbf{u}}$ (Lange et al., 2000) and minimizing it corresponds to performing one-step of the local linear-approximation algorithm (Zou and Li, 2008; Schifano et al., 2010) which is a special case of the majorization-minimization (MM) algorithm (Lange et al., 2000). The corresponding MM algorithm would consist of repeating the following two steps: (i) using a previous CoCo estimate $\tilde{\boldsymbol{\mathcal{U}}}$ to compute weights $w_{d,ij}$ according to (13), and (ii) computing a new CoCo estimate using the new weights. In practice, we have found one-step to be adequate, however. Indeed, Zou and Li (2008) showed that the solution to the one-step algorithm was often sufficient in terms of its statistical estimation accuracy.

7. Other Practical Issues

In this section, we address other considerations for using the method in practice, namely how to choose the tuning parameter γ and how to recover the partitions along each mode from the CoCo estimator $\hat{\mathbf{U}}$.

7.1 Choosing γ

The first major practical consideration is how to choose γ to produce a final co-clustering result. Since co-clustering is an exploratory method, it may be suitable for a user to manually inspect a sequence of CoCo estimators $\hat{\mathbf{U}}_\gamma$ for a range of γ and use domain knowledge tied to a specific application to select γ to recover a co-clustering assignment of a desired complexity. Since this approach is time consuming and requires expert knowledge, an automated, data-driven procedure for selecting γ is desirable. Cross-validation (Stone, 1974; Geisser, 1975) and stability selection (Meinshausen and Bühlmann, 2010) are popular techniques for tuning parameter selection, but since both methods are based on resampling, they are unattractive in the tensor setting due to the computational burden. We turn to the extended Bayesian Information Criterion (eBIC) proposed by Chen and Chen (2008, 2012), as it does not rely on resampling and thus is not as computationally costly as cross-validation or stability selection.

$$\text{eBIC}(\gamma) \quad = \quad n \log\left(\frac{\text{RSS}_\gamma}{n}\right) + 2\text{df}_\gamma \log(n),$$

where RSS_γ is the residual sum of squares $\|\mathbf{X} - \hat{\mathbf{U}}_\gamma\|_{\text{F}}^2$ and df_γ is the degrees of freedom for a particular value of γ. We use the number of co-clusters in the CoCo estimator $\hat{\mathbf{U}}_\gamma$ as an estimate of df_γ, which is consistent with the spirit of degrees of freedom since each co-cluster mean is an estimated parameter. This criterion balances between model fitting and model complexity, and a similar version has been commonly employed in tuning parameter selection of tensor data analysis (Zhou et al., 2013; Sun et al., 2017).

The eBIC is calculated on a grid of values $\mathcal{S} = \{\gamma_1, \gamma_2, \ldots \gamma_s\}$, and we select the optimal γ, denoted $\gamma^\star$, which corresponds to the smallest value of the eBIC over $\mathcal{S}$, namely

$$\gamma^\star \quad = \quad \arg\min_{\gamma \in \mathcal{S}} \text{eBIC}(\gamma).$$

7.2 Recovering the Partitions along Each Mode

The second major practical consideration is how to extract the partitions from the CoCo estimator $\hat{\mathbf{U}}$. Recall that the ith and jth mode-d subtensors belong to the same partition if $\mathbf{v}_{d,ij} = \mathbf{U} \times_d \Delta_{ij} = \mathbf{0}$. Conversely, the ith and jth mode-d subtensors *do not* belong to the same partition if $\mathbf{v}_{d,ij} \neq \mathbf{0}$. Thus, a mode-$d$ partition consists of the maximal set of mode-d subarrays such that for any pair i and j in this collection $\mathbf{v}_{d,ij} = \mathbf{0}$. We can automatically identify these maximal sets by extending a simple procedure employed by Chi and Lange (2015) for extracting clusters in the convex clustering problem. Identifying partitions along the dth mode is equivalent to finding connected components of a graph, where each node corresponds to a subarray along the dth mode, and there is an edge between nodes i and j if and only if $\mathbf{v}_{d,ij} = \mathbf{0}$.

We would like to read off which centroids have fused as the amount of regularization increases, namely determine partition assignments as a function of γ. Such assignments can be performed in $\mathcal{O}(n_d)$ operations, using the differences variable $\mathbf{V}_d$. We simply apply breadth-first search to identify the connected components of the following graph induced by the $\mathbf{V}_d$. The graph identifies a node with every data point and places an edge between the lth pair of points if and only if $\mathbf{v}_l = \mathbf{0}$. Each connected component corresponds to a partition. Note that the graph constructed to determine partitions is *not* the same as the graph described in Section 3 with illustrative examples in Figure 2.

We emphasize that the recovered partition along each mode does *not* depend on the ordering of the input data $\mathcal{X}$, since it is based off of the pairwise differences along each mode, namely $\mathbf{V}_d$ for $d = 1, \ldots, D$. Finally, we note that due to finite precision limitations, the difference variables $\mathbf{v}_{d,ij}$ will likely not be exactly $\mathbf{0}$. In Appendix E.4, we detail a simple and principled procedure for ensuring sparsity in these difference variables.

8. Simulation Studies

To investigate the performance of the CoCo estimator in identifying co-clusters in tensor data, we first explore some simulated examples. We compare our CoCo estimator to a k-means based approach that is representative of various tensor generalizations of the spectral clustering method common in the tensor clustering literature (Kutty et al., 2011; Liu et al., 2013b; Zhang et al., 2013; Wu et al., 2016b). We refer to this method as CPD+k-means. The CPD+k-means method (Papalexakis et al., 2013; Sun and Li, 2019) first performs a rank-R CP decomposition on the D-way tensor $\mathcal{X}$ to reduce the dimensionality of the problem, and then independently applies k-means clustering to the rows of each of the D factor matrix from the resulting CP decomposition. The k-means algorithm has also been used to cluster the factor matrices resulting from a Tucker decomposition (Acar et al., 2006; Sun et al., 2006; Kolda and Sun, 2008; Sun et al., 2009; Kutty et al., 2011; Liu et al., 2013b; Zhang et al., 2013; Cao et al., 2015; Oh et al., 2017). We also considered this Tucker+k-means method in initial experiments, but its co-clustering performance was inferior to that of CPD+k-means so we only report co-clustering performance results for CPD+k-means in the comparison experiments that follow. Note, however, that we still use the Tucker decomposition to compute CoCo weights $w_{d,ij}$ as described Section 6. Both CoCo and CPD+kmeans account for the multiway structure of the data. To assess the importance of accounting for this structure, we also include comparisons with the CoTeC method (Jegelka et al., 2009), which applied k-means clustering along each mode and does not account for the multiway structure of the data.

All methods being compared have tuning parameters that need to be set. For the rank of the CP decomposition needed in CPD+k-means, we consider $R \in \{2, 3, 4, 5\}$ and use the tuning procedure in Sun et al. (2017) to automatically select the rank. A CP decomposition is then performed using the chosen rank, and those factor matrices are the input into the k-means algorithm. A well known drawback of k-means is that the number of clusters k needs to be specified *a priori*. Several methods for selecting k have been proposed in the literature, and we use the "gap statistic" developed by Tibshirani et al. (2001) to select an optimal k^* from the specified possible values. Since CoCo estimates an entire solution path of mode-clustering results, ranging from n_d clusters to a single cluster along mode d,

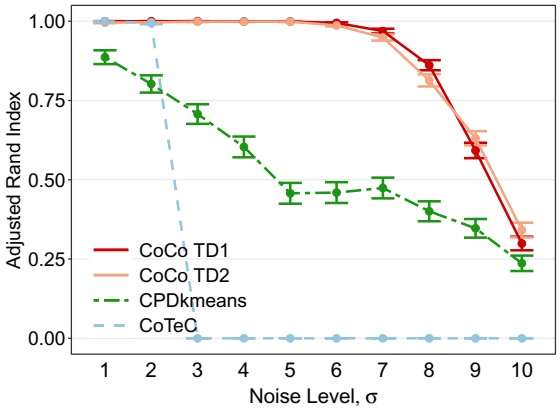


Figure 4: Checkerbox Simulation Results: Impact of Noise Level. Two balanced clusters per mode across different levels of homoskedastic noise for $n_1 = n_2 = n_3 = 60$. For each method, the confidence interval is calculated as the mean value plus/minus one standard error.

we consider a rather large set of possible k values to make the methods more comparable. Appendix G gives a more detailed description of the CPD+k-means procedure and the selection of its tuning parameters. CoTeC, which applies k-means clustering along each mode independently, also requires specifying the number of cluster along each mode. As in CPD+k-means, we also select this parameter along each mode using the "gap statistic."

As described in Section 6, we employ a Tucker approximation to the data tensor in constructing weights $w_{d,ij}$. In computing the Tucker decomposition we used one of two methods for selecting the rank. In the plots within this section, TD1 denotes the results where the Tucker rank was chosen using the SCORE algorithm (Yokota et al., 2017), while TD2 denotes results where the rank was chosen using a heuristic. Detailed discussion on these two methods are in Appendix F.

The results presented in this section report the average CoCo estimator performance quantified by the ARI across 200 simulated replicates. All simulations were performed in Matlab using the Tensor Toolbox (Bader et al., 2015). All the following plots, except the heatmaps in Figure 13, were made using the open source R package ggplot2 (Wickham, 2009).

8.1 Cubical Tensors, Checkerbox Pattern

For the first and main simulation setting, we study clustering data in a cubical tensor generated by a basic checkerbox mean model according to Assumption 4.2. Each entry in the observed data tensor is generated according to the underlying model (2) with independent errors $\epsilon_{i_1 i_2 i_3} \sim N(0, \sigma^2_{r_1 r_2 r_3})$. Unless specified otherwise, there are two true clusters along each mode for a total of eight underlying co-clusters.

8.1.1 Balanced Cluster Sizes and Homoskedastic Noise

To get an initial feel for how the different co-clustering methods perform at recovering the true underlying checkerbox structure, we first consider a situation where the clusters corresponding to the two classes along each mode are all equally-sized, or balanced, and

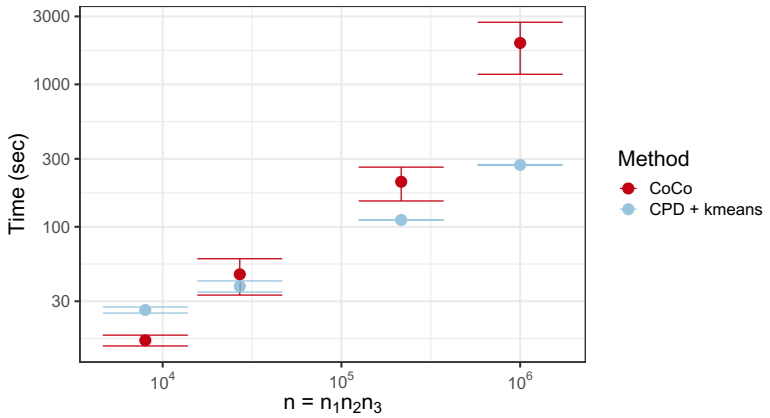


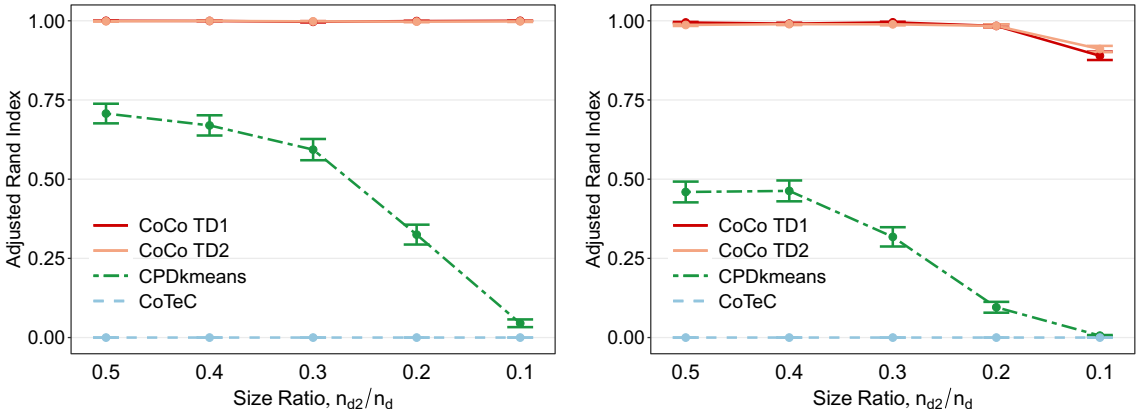
Figure 5: Timing Results: Balanced Cluster Size and Homoskedastic Noise. Two balanced clusters per mode with a fixed level of homoskedastic noise for $n_1 = n_2 = n_3 = 20, 30, 60$, and 100. Vertical and horizontal axes are on a log scale.

share the same error variance, namely $\sigma_{r_1 r_2 r_3} = \sigma$ for all r_1, r_2, and r_3. The average co-clustering performance for this setting in a tensor with dimensions $n_1 = n_2 = n_3 = 60$ are given in Figure 4 for different noise levels. Figure 4 shows that all three methods perform well when the noise level is low ($\sigma = 1$). As the noise level increases, however, CPD+k-means experiences an immediate and noticeable drop off in performance. CoTeC's performance decays even more rapidly highlighting the importance of accounting for multiway structure. The CoCo estimator, on the other hand, is able to maintain near-perfect performance until the noise level becomes rather high ($\sigma = 8$).

Figure 5 shows how the run times of CoCo and CPD+k-means vary as the size of a cubic tensor, $n = n_1 n_2 n_3$ with $n_1 = n_2 = n_3$ takes on the values $20^3, 30^3, 60^3$, and 100^3. These run times include all computations needed to fit and select a final model. For CoCo, a sequence of models were fit over a grid of γ parameters, and a final γ parameter was chosen using the eBIC. For CPD+k-means, a sequence of models were fit over a grid of possible (k_1, k_2, k_3) parameters corresponding to the 3 factor matrices, and a final triple of (k_1, k_2, k_3) parameters were chosen using the "gap statistic." Timing comparisons were performed on a 3.2 GHz quad-core Intel Core i5 processor and 8 GB of RAM. The run time for CoCo scales linearly in the size of the data tensor as expected, namely proportionately with n_1^3. Nonetheless, as also might be expected, the clustering performance enjoyed by CoCo does not come for free, and the simpler but less reliable CPD+k-means algorithm enjoys a better scaling as the tensor size grows. Timing results were similar for the following experiments and are omitted for space considerations.

8.1.2 Imbalanced Cluster Sizes

When comparing clustering methods, one factor of interest is the extent to which the relative sizes of the clusters impact clustering performance. To investigate this, we again use a cubical tensor of size $n_1 = n_2 = n_3 = 60$ but introduce different levels of cluster size imbalance along each mode, which we quantify via the ratio of the number of samples in cluster 2 of mode d and the total number of samples along mode d, for $d = 1, 2, 3$. Figure 6a



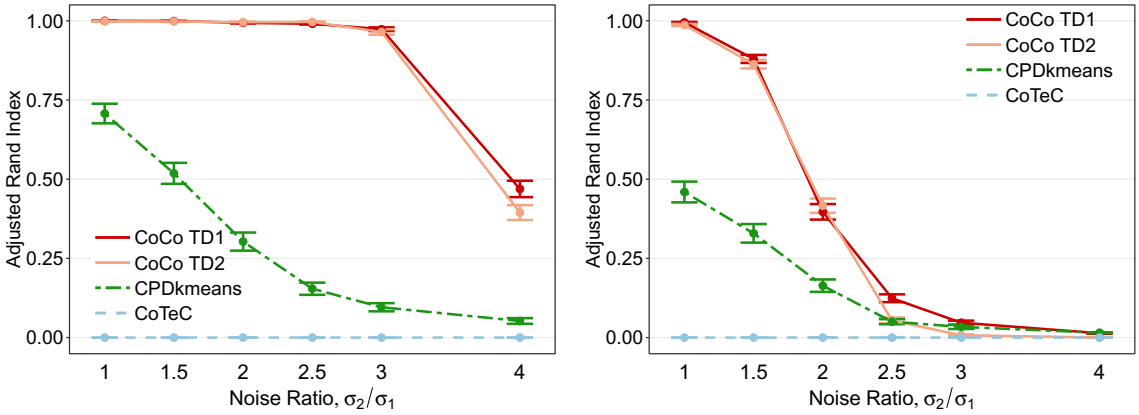
(a) Low Noise (b) High Noise

Figure 6: Checkerbox Simulation Results: Impact of Cluster Size Imbalance. Two imbalanced clusters per mode with either low or high homoskedastic noise for $n_1 = n_2 = n_3 = 60$. Low noise corresponds to $\sigma = 3$ while high noise refers to $\sigma = 6$.

shows that when the noise level is low, CPD+k-means is unaffected by the imbalance until the size of cluster 2 is less than 30% of the mode's length. At this point, the performance of CPD+k-means drops off significantly and it performs as well as a random clustering assignment when the sizes are highly skewed ($n_{d2}/n_d = 0.1$). The CoCo estimator is more or less invariant to the imbalance, and its performance is almost perfect across all levels of cluster size imbalance. Figure 6b shows that the CoCo estimator exhibits a slight deterioration in performance only when the cluster size ratio is 0.1 in the high noise case. In both low and high noise scenarios, CoTeC performs poorly.

8.1.3 Heteroskedastic Noise

Another factor of interest is how the clustering methods perform when there is heteroskedasticity in the variability of the two classes. Figure 7 displays the co-clustering performance for different degrees of heteroskedasticity, as measured by the standard deviation for class 2 relative to class 1's standard deviation, σ_2/σ_1. In the low noise setting, the CoCo estimator is immune to the heteroskedasticity until the noise levels differ by a factor of 4. CPD+k-means in contrast is very sensitive to a deviation from homoskedasticty, experiencing a decline even when the noise ratio increases from 1 to only 1.5. The CoCo estimator fares worse in the high noise setting and also has a drop in performance with a small deviation from homoskedasticty. Once class 2's standard deviation is more than double the standard deviation for class 1, all three methods are essentially the same as random clustering. This result is not terribly surprising since, in the high noise setting, this would result in one class having a very high standard deviation of $\sigma_2 = 12$. In both low and high noise scenarios, CoTeC performs poorly.



(a) Low Noise (b) High Noise

Figure 7: Checkerbox Simulation Results: Impact of Heteroskedasticity. Two balanced clusters per mode with either low or high heteroskedastic noise for $n_1 = n_2 = n_3 = 60$. Low noise corresponds to $\sigma_1 = 3$ while high noise refers to $\sigma_1 = 6$.

8.1.4 Different Clustering Structures

So far, we have considered only a simple situation where there are exactly two true clusters along each mode, for a total of eight triclusters. Another factor of practical importance is how the clustering methods perform when there are more than two clusters per mode, and also when the number of clusters along each mode differs. We investigate both of these settings in this section. As before, the tensor is a perfect cube with $n_1 = n_2 = n_3 = 60$ observations along each mode and an underlying checkerbox pattern. To gauge the performance, we again focus the attention on how the methods perform in the presence of both low and high noise.

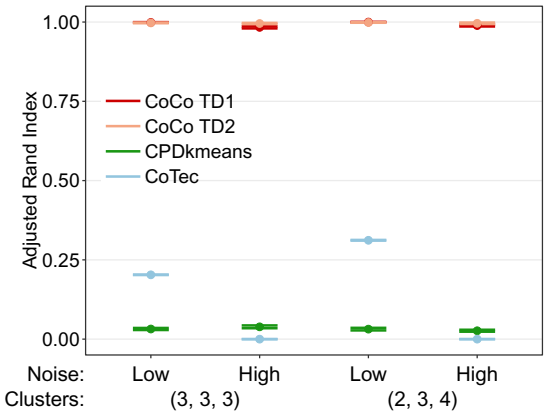


Figure 8: Checkerbox Simulation Results: Impact of Clustering Structure. Different balanced clusters per mode with either low or high homoskedastic noise for $n_1 = n_2 = n_3 = 60$. Low noise corresponds to $\sigma = 3$ while high noise refers to $\sigma = 6$.

The first situation studied is one in which there are three true clusters along each mode, resulting in a total of 27 triclusters. The left hand side of the graphs in Figure 8 show the results from this simulation setting. The graphs show that CoCo estimator consistently outperforms CPD+k-means and CoTeC in this setting across both noise levels. The CoCo estimator is able to recover the true co-clusters almost perfectly, while CPD+k-means struggles to handle the increased number of clusters per mode.

We also investigated the clustering performance when the number of clusters per mode varies. In this setting, there are two, three, and four clusters along modes one, two, and three, respectively. From the right hand side of the graphs in Figure 8, we can see that the results are similar to the situation with three clusters per mode. CPD+k-means again performs very poorly across both noise levels, while convex co-clustering is again able to essentially recover the true co-clustering structure. Compared to the setting with three clusters per mode, CPD+k-means performs slightly worse in the face of a more complex clustering structure, while convex co-clustering is able to handle it in stride. These results bode well for convex co-clustering as the basic clustering structure of only two clusters per mode is unlikely to be observed in practice.

8.2 Rectangular Tensors

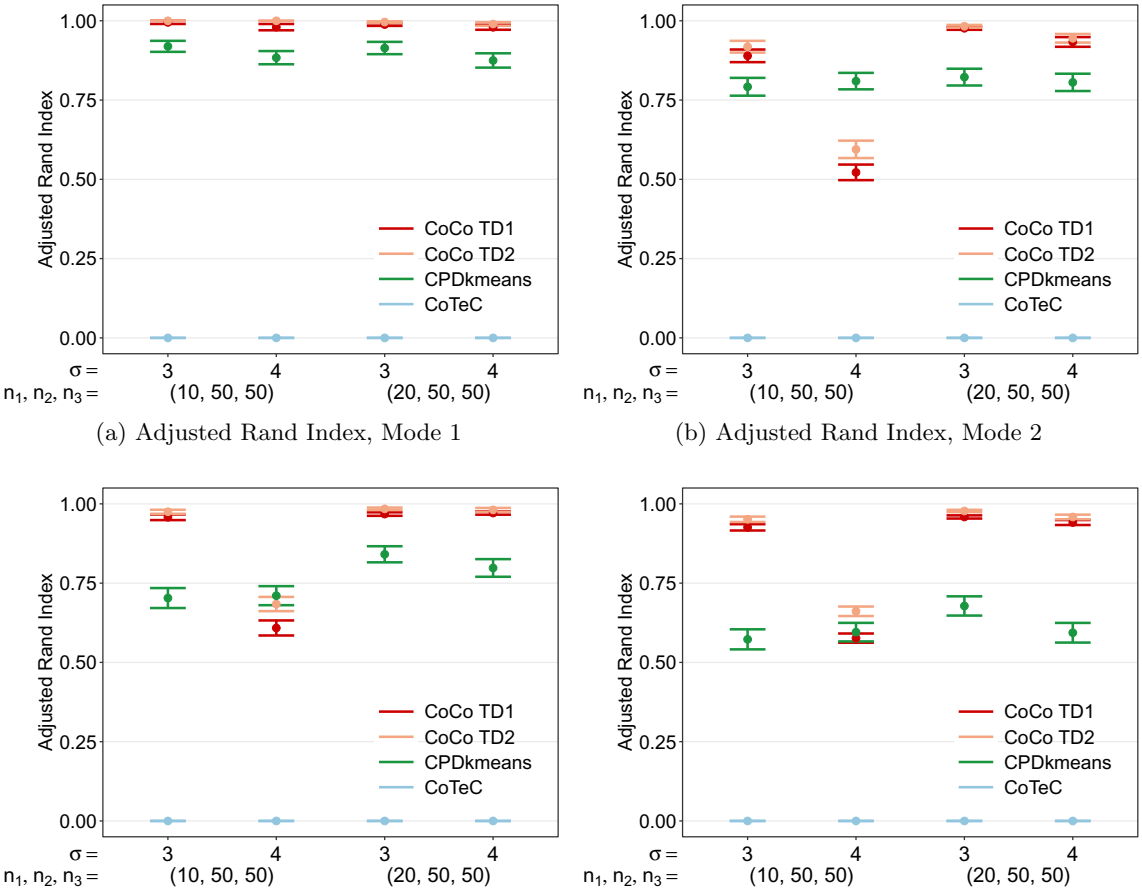
Up to this point, to get an initial feel for CoCo's performance, we restricted our attention to cubical tensors with the same number of observations per mode so as to avoid changing too many factors at once. It is unlikely that the data tensor at hand will be a perfect cube, however, so it is important to understand the clustering performance when the methods are applied to rectangular tensors.

Now we turn to cluster a rectangular tensor with one short mode and two longer modes. Two additional simulations involving rectangular tensors can be found in Appendix H. Figure 9 shows that CoCo performs very well and better than CPD+k-means and CoTeC at the lower noise level ($\sigma = 3$) but has a sharp decrease in ARI at the higher noise level ($\sigma = 4$). The decline is more pronounced for the longer modes (Figure 9b and Figure 9a) as the short mode (Figure 9a) is still able to maintain perfect performance despite the increase in noise. This is not surprising, since the shorter mode has effectively more samples. Moreover, we see the "blessing of dimensionality" at work when the number of samples along the short mode are doubled ($n_1 = 20$, $n_2 = n_3 = 50$), the performance along the two longer modes improves drastically in the high noise setting.

We finally note that, along the shorter mode, the use of the heuristic in determining the rank of the Tucker decomposition for calculating the weights performs better than the SCORE algorithm method along modes 1 and 2, though ultimately the co-clustering performance is comparable. This may indicate that the SCORE algorithm struggles to correctly identify the optimal Tucker rank for short modes in the presence of relatively higher noise, while the heuristic is more immune to the noise level as it is based simply on the dimensions of the tensor.

8.3 CANDECOMP/PARAFAC Model

In Section 8.1, we saw that the CoCo estimator performs well and typically better than CPD+k-means when clustering tensors whose co-clusters have an underlying checkerbox



(a) Adjusted Rand Index, Mode 1

(b) Adjusted Rand Index, Mode 2

(c) Adjusted Rand Index, Mode 3

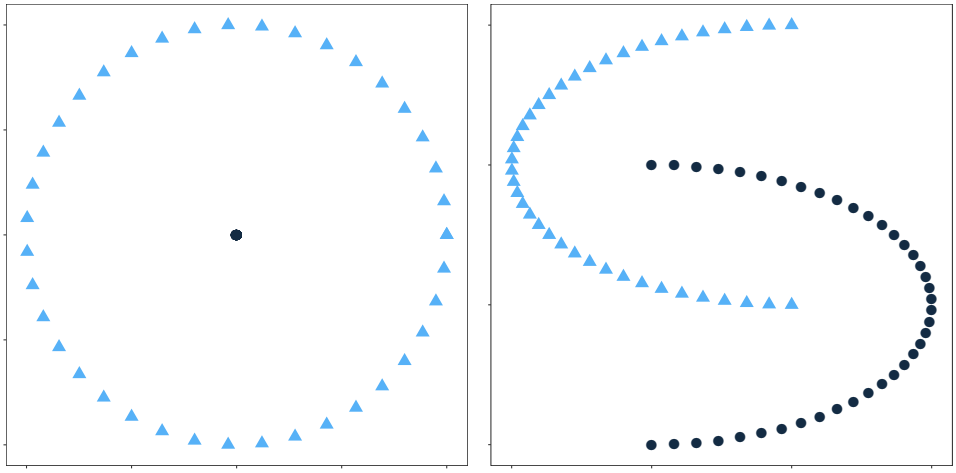
(d) Adjusted Rand Index, Triclusters

Figure 9: Checkerbox Simulation Results: Impact of Tensor Shape. Two balanced clusters per mode with two levels of homoskedastic noise for a tensor with one short mode and two longer modes. Average adjusted rand index plus/minus one standard error for different noise levels and mode lengths.

pattern. To evaluate the performance of our CoCo estimator under model misspecification, we consider the generative model as the following CP decomposition model. We first construct the factor matrix $\mathbf{A} \in \mathbb{R}^{80\times 2}$ and construct the following rank-2 CP means tensor

$$\mathcal{U}^* = \sum_{i=1}^{2} \mathbf{a}_i \circ \mathbf{a}_i \circ \mathbf{a}_i,$$

where $\circ$ denotes the outer product. We then added varying levels of Gaussian noise to the $\mathcal{U}^*$ to generate the observed data tensor. We consider two different types of factor matrices. As shown in Figure 10, one shape consists of two half-moon clusters (Hocking et al., 2011; Chi and Lange, 2015; Tan and Witten, 2015) while the other shape contains a bullseye, similar to the two-circles shape studied by Ng et al. (2002) and Tan and Witten (2015). In either case, the triangles in Figure 10 correspond to the first 40 rows of $\mathbf{A}$, whereas the



(a) Bullseye (b) Half Moons

Figure 10: Factor Matrices for the CP Models.

circles correspond to the second 40 rows of $\mathbf{A}$. Note that this data generating mechanism should favor the CPD+k-means method.

Figure 11 shows the simulation results for using the CP model with these two non-convex shapes generating the data. The discrepancy in performance between the CoCo estimator and the other two methods is quite large. The CoCo estimator almost perfectly identifies the true co-clusters. In contrast, both CPD+k-means and CoTeC perform very poorly, even when the noise variance is small. The poor performance of CPD+k-means and CoTeC are not completely surprising as other have noted the difficulty that k-means methods have in recovering non-convex clusters (Ng et al., 2002; Hocking et al., 2011; Tan and Witten, 2015). These results give us some assurances that the CoCo estimator is able to still perform well even under some model misspecification since the true co-clusters do not have a checkerbox pattern.

8.4 Comparison with Convex Biclustering

It is natural to ask how much additional gain there is in using CoCo over convex biclustering (Chi et al., 2017) on the matricizations of a data tensor. To answer this question, we compare CoCo to the following strategy for applying convex biclustering to estimate co-clusters. We explain the strategy for a 3-way tensor; the generalization to D-way tensors is straightforward. We first matricize the tensor $\mathcal{X}$ along mode-1 to obtain the matrix $\mathbf{X}_{(1)}$, apply convex biclustering on $\mathbf{X}_{(1)}$, and retain the mode-1 clustering results. Note that the mode-2 and mode-3 fibers have been mixed together through the matricization process. We then repeat the two-step procedure for mode-2 and mode-3. The final co-cluster estimates are obtained by taking the cross-products of the mode-1, mode-2, and mode-3 cluster assignments.

We consider two illustrative scenarios to understand the value of preserving the full multiway structure with CoCo: a balanced case and imbalanced case. In the balanced

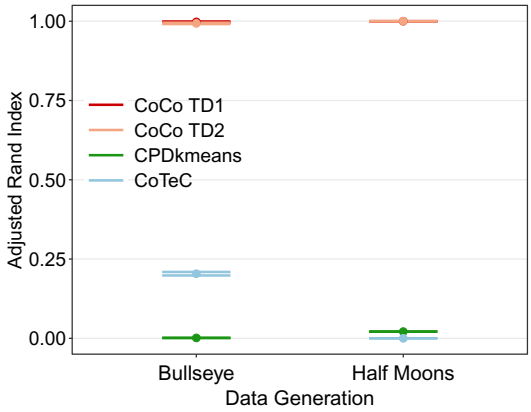
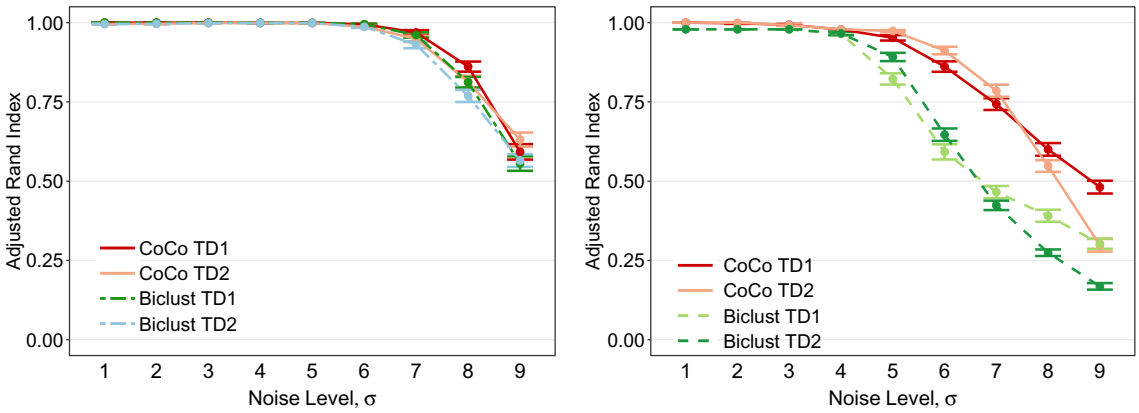


Figure 11: CP Model Simulation Results. Two balanced clusters per mode with low homoskedastic noise for $n_1 = n_2 = n_3 = 40$. "Bullseye" and "Half Moons" refer to the shape embedded in the factor matrices used to generate the true tensor.



(a) Balanced (b) Imbalanced

Figure 12: A Comparison between CoCo and Convex Biclustering Average Adjusted Rand Index plus/minus one standard error for different noise levels.

case, we have a 3-way data tensor $\mathcal{X} \in \mathbb{R}^{60\times60\times60}$ with two clusters along each mode, where clusters are of equal size and homoskedastic iid Gaussian noise has been added to all elements of the tensor. This scenario is similar to the one shown in Figure 4. In the imbalanced case, we have a 3-way data tensor $\mathcal{X} \in \mathbb{R}^{30\times40\times80}$. There are two clusters along mode-1 of sizes 10 and 20, three clusters along mode-2 of sizes 8, 12, and 20, and four clusters along mode-3 of sizes 5, 10, 20, and 45. Homoskedastic iid Gaussian noise has been added to all elements of the tensor. Finally, we note that the empirical performance of convex biclustering, like that of CoCo's, depends on choosing good weights for the rows and columns of the input data matrix (Chi et al., 2017). To create a fair comparison, we construct convex biclustering weights based off of the same TD1 and TD2 denoising procedure used for CoCo, putting the preprocessing for both methods on equal footing.

Figure 12a and Figure 12b show the co-clustering performance of CoCo and the convex biclustering method in the balanced and imbalanced cases respectively. We see that in the balanced case, CoCo's performance is marginally better than that of the convex biclustering method. On the other hand, we see that in the imbalanced case, CoCo's performance degrades more gracefully than that of the convex biclustering method as the noise level increases. The example illustrates that CoCo has better co-cluster recovery when there is more imbalance in the data tensor - the aspect ratios of the tensor dimensions are more skewed and the number of clusters and the cluster sizes are more heterogenous.

The key formulation difference between CoCo and the convex biclustering method that provides some insight into these two results is that CoCo imposes a finer level of smoothness that respects the multi-way structure in the data tensor. Imposing such finer level of smoothness imparts greater robustness in the presence of increasing noise to recovering the smaller co-clusters in the imbalanced scenario. An added incentive for using CoCo and preserving the multiway structure in the data is that the gains in co-cluster recovery over the convex biclustering method do not come at a greater computational cost. Note that the computational complexity of convex biclustering is $\mathcal{O}(n)$, using sparse weights for the row and column similarity graphs. For a D-way tensor, the computational complexity then becomes $\mathcal{O}(Dn)$, which is the same as the computational complexity of CoCo applied directly on the D-way tensor.

To summarize, in comparison to the convex biclustering method, CoCo (i) does not come at additional computational costs, (ii) can recover underlying co-clustering structure in imbalanced scenarios which are more likely to be encountered in practice, and (iii) has the ability to consistently recover an underlying co-clustering structure according to Theorem 9, with even a single tensor sample, which is a typical case in real applications. Since this phenomenon does not exist in vector or matrix variate cluster analysis, the convex biclustering method lacks this theoretical guarantee.

9. Real Data Application

Having studied the performance of the CoCo estimator in a variety of simulated settings, we now turn to using the CoCo estimator on a real data set. The proprietary data set comes from a major online company and contains the click-through rates for advertisements displayed on the company's webpages from May 19, 2016 through June 15, 2016. The click-through rate is the number of times a user clicks on a specific advertisement divided by the number of times the advertisement was displayed. The data set contains information on 1000 users, 189 advertisements, 19 publishers, and 2 different devices, aggregated across time. Thus, the data forms a fourth-order tensor where each entry in the tensor corresponds to the click-through rate for the given combination of user, advertisement, publisher, and device. Here a publisher refers to a different webpage within the online company's website, such as the main home page versus a page devoted to either breaking news or sports scores. The two device types correspond to how the user accessed the page, using either a personal computer or a mobile device such as a cell phone or tablet computer. The goal in this real application is to simultaneously cluster users, advertisements, and publishers to improve user behavior targeting and advertising planning.

In the click-through rate tensor data, over 99% of the values are missing since one user likely has seen only a handful of the possible advertisements. If a specific advertisement is never seen by a user, it is considered as a missing value. Since the proposed CoCo estimator can only handle complete data, we first preprocess the data by imputing the missing values before any clustering can be done. To impute the missing entries, we use the CP-based tensor completion method Jain and Oh (2014) and tune its rank via the information criterion proposed by Sun et al. (2017). This tuning method chooses the optimal rank as $R = 20$ from the rank list $\{1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 16, 18, 20, 22\}$. Finally, the imputed values are truncated to ensure all the values of the tensor are within 0 and 1 since click-through rates are proportions.

One mode of the fourth-order tensor has only two observations and those observations already have a natural grouping (device type). Therefore, for the sake of clustering we analyze the devices separately. We compare our method with CPD+k-means. Furthermore, the tuning parameter for convex co-clustering is automatically selected using the eBIC (Section 7.1) while the number of clusters in CPD+k-means is chosen via the gap statistic (Tibshirani et al., 2001). We do not include comparisons with CoTeC given its poor performance in the simulation experiments.

We first look at the clustering results from clustering the click-through rates for users accessing the advertisements through a personal computer (PC). Table 1 contains the number of clusters identified as well as the sizes of the clusters, while Figure 13a visualizes the advertisement-by-publisher biclusters for a randomly selected user. As to be expected, the advertisement-by-publisher slices display a checkerbox pattern, which turns into a checkerbox pattern when the slices are meshed together. The clustering results for the users are omitted in this paper to ensure user privacy. However, co-clustering the tensor does not result in the loss of information that would occur if the tensor was converted into a matrix by averaging across users or flattening along one of the modes. Table 1 and Figure 13a show that the CoCo estimator identifies four advertisement clusters, with one cluster being much bigger than the others. The advertisements in this large cluster have click-through rates that are close to the grand average in the data set. One of the small clusters has very low click-through rates, while the other two clusters tend to have much higher click-through rates than the rest of the advertisements. On the other hand, CPD+k-means clusters the advertisements into 57 groups, which is less-useful from a practical standpoint. Many of the clusters are similarly-sized and contain only a few advertisements, likely due to the inability of CPD+k-means to handle imbalanced cluster sizes as was observed in the simulation experiments (Section 8.1.2). In terms of the publishers, the CoCo estimator identifies 3 clusters while CPD+k-means does not find any underlying grouping and simply identifies one big cluster, which again is not terribly useful (Table 1). We next provide some interpretations of the obtained clustering results of the publishers. One way online advertisers can reach more users is by entering agreements with other companies to route traffic to the advertiser's website. For example, Google and Apple have a revenue-sharing agreement in which Google pays Apple a percentage of the revenue generated by searches on iPhones (McGarry, 2016). Similarly, the online company being studied partners with several internet service providers (ISPs) to host the defaut home pages for the ISP's customers. It would make sense that these slightly different variants of the online company's main home

Device	CoCo Estimator				CPD+kmeans	
	Advertisements		Publisher		Advertisements	Publisher
	# of clusters	Cluster Sizes	# of clusters	Cluster Sizes	# of clusters	# of clusters
PC	4	(156, 22, 8, 3)	3	(4, 3, 12)	57	1
Mobile	3	(145, 22, 22)	2	(7, 12)	49	13

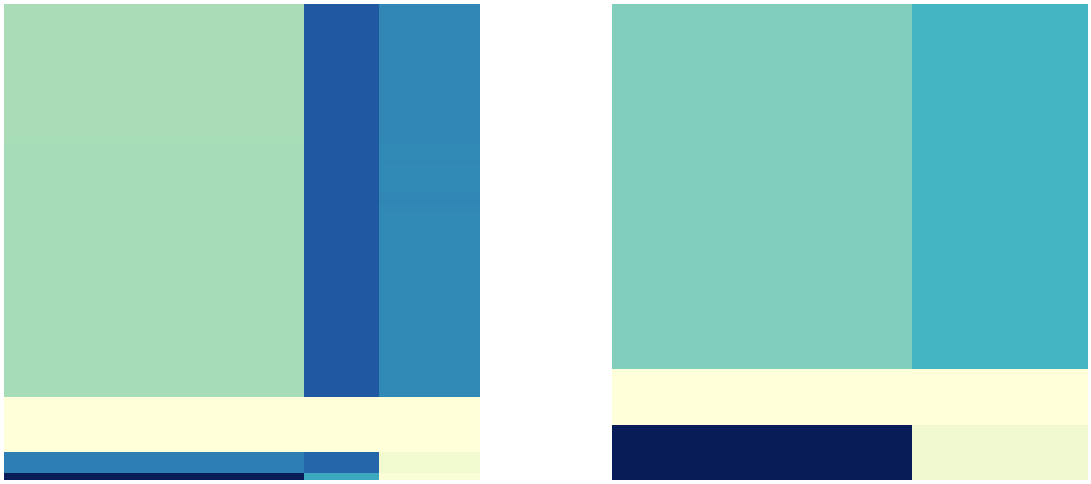
Table 1: Advertising Data Clustering Results

page would have similar click-through rates, and the CoCo estimator in fact assigned these variants into the same cluster.

For users accessing the advertisements through a mobile device, such as a mobile phone or tablet computer, the CoCo estimator results for the advertisements are largely similar to the results for PCs (Table 1 and Figure 13b). There is one large cluster that contains click-through rates similar to the overall average, while the two other equally-sized clusters have relatively very low or very high click-through rates, respectively. The underlying click-through rates for the PC data have more variability than the mobile data, which is consistent with the identification of an additional cluster for the PC data. As before, CPD+k-means finds a large number of advertisement clusters, most of which are roughly the same size, again likely impacted by the imbalance in the cluster sizes. When compared to the personal computer device, one difference is that the cluster with the higher click-through rates for mobile devices is larger and has a higher average click-through rate than the similar clusters for the personal computer device. This finding is consistent with research by the Pew Research Center that found that click-through rates for mobile devices are higher than for advertisements viewed on a personal computer or laptop (Mitchell et al., 2012).

It is also enlightening to take a closer look at the underlying advertisements clustered across the two devices. All of the advertisements clustered in the high click-through rate cluster for the mobile devices are in the average click-through rate cluster for personal computers. In taking a closer look at the ads in these clusters, there are several ads related to online shopping for personal goods, such as jeans, workout clothes, or neck ties. It makes sense to shop for these types of goods using a mobile device, such as while at work when it is not appropriate to do so on a work computer. Conversely, all of the advertisements in either of the two higher PC click-through rate clusters are in the large, average click-through rate cluster for the mobile devices. There are several financial-related ads in these two PC clusters, such as for mortgages or general investment advice. On the other hand, there are not many online shopping ads in those clusters, with the exception of more expensive technology-related goods that one may want to invest more time in researching before making a purchase.

In terms of the publisher clusters on mobile device, Table 1 shows that the CoCo estimator identifies two clusters of publishers while CPD+k-means identifies 13 small clusters. Contrary to the advertisement clusters, the publisher clusters across both devices are very similar. In fact, the only difference is that the smaller cluster for the mobile device, which contains seven publishers, is split into two clusters for personal computers. This can be



(a) Personal Computers (b) Mobile

Figure 13: Advertisement and Publisher Click-Through Rate Biclusters for a Randomly Selected User. The rows correspond to different advertisements and the columns correspond to different publishers. Darker blue corresponds to higher click-through rates for a given device.

seen in the click-through rate heatmaps given in Figure 13 in looking at the right part of each heatmap. The publishers in these smaller clusters have higher click-through rates on average than those in the larger cluster. Additionally, five of the seven (71%) publishers in the high click-through rate clusters have stand-alone apps that display ads, while only three of the twelve (25%) publishers in the larger cluster do. For mobile devices, it has been observed that in-app advertisements have higher click-through rates and browser-based ads (Hof, 2014). We conjecture that this is also true for personal computer apps, which is consistent with the clustering results. Thus it again appears that the clusters identified by CoCo also make sense practically.

10. Discussion

In this paper, we formulated and studied the problem of co-clustering of tensors as a convex optimization problem. The resulting CoCo estimator enjoys features in theory and practice that are arguably lacking in existing alternatives, namely statistical consistency, stability guarantees, and an algorithm with polynomial computational complexity. Through a battery of simulations, we observed that the CoCo estimator can identify co-clustering structures under realistic scenarios such as imbalanced co-cluster sizes, imbalanced number of clusters along each mode, heteroskedasticity in the noise distribution associated with each co-cluster, and even some violation of the checkerbox mean tensor assumption.

We have leveraged the power of the convex relaxation to engineer a computationally tractable co-clustering method that comes with statistical guarantees. These benefits, however, do not come for free. The CoCo estimator incurs similar costs that using the lasso incurs as a surrogate for a cardinality constraint or penalty. It is well known that the lasso leads to parameter estimates that are shrunk towards zero. This shrinkage toward zero

is the price for simultaneously estimating the support, or locations of the nonzero entries, in a sparse vector as well as the values of the nonzero entries. In the context of convex co-clustering, the CoCo estimator $\hat{\mathbf{U}}$ is shrunk towards the tensor $\bar{\mathbf{X}}$, namely the tensor whose entries are all equal to the average over all entries of $\mathbf{X}$. The weights, however, play a critical role in reducing this bias. In fact, the weights can be seen as serving the same role as weights used in the adaptive lasso (Zou, 2006).

There are several possible extensions and open problems that have been left for future work. First, we note that there is a gap between what our theory predicts and what seems possible from our experiments. Specifically, Theorem 9 assumes uniform weights for each mode, yet simulation experiments indicate that the CoCo estimator using Tucker derived Gaussian kernel weights (9) can significantly outperform the CoCo estimator using uniform weights. One open problem is to derive prediction error bounds that relax the uniform weights assumption.

Second, although we have developed automatic methods for constructing the weights that work well empirically, other approaches to constructing the weights is a direction of future search. For example, other tensor approximation methods, such as the use of the ℓ_1-norm to make the decomposition most robust to heavy tail noise as done by Cao et al. (2015), could possibly improve the quality of the weights.

Third, in this paper we have focused on additive noise that is a zero-mean M-concentrated random variable. Real data, however, may not follow such a distribution motivating co-clustering procedures that can handle outliers. To address potential robustness issues, the CoCo framework could be extended to handle outliers by swapping the sum of squared residuals term in (5) with an analogous Huber loss or Tukey's Biweight function.

Finally, while our first order algorithm for co-clustering tensors scales linearly in the size of the data, data tensors inevitably will only increase in size motivating the need for more scalable algorithms for computing the CoCo estimator. A natural approach would be to adopt an existing distributed version of the proximal methods, such as one the methods proposed by Combettes and Pesquet (2011), Chen and Ozdaglar (2012), Li et al. (2013), or Eckstein (2017). Another natural approach would be to investigate if stochastic versions of the recently proposed generalized dual gradient ascent (Ho et al., 2019) could be adapted to compute the CoCo estimator. Additionally, in practice many data tensors that we would like to co-cluster may be very sparse. The first order algorithm presented here assumes the data tensor is dense. Consequently, an important direction of future work is to investigate alternative optimization algorithms that could leverage the sparsity structure within a data tensor.

Acknowledgments

The authors thank Xu Han for his help with the simulation experiments during the revision of this work. The authors also thank the action editor and three reviewers for their helpful comments and suggestions which led to a much improved presentation. Eric Chi acknowledges support from the National Science Foundation (DMS-1752692) and National Institutes of Health (R01GM135928). Will Wei Sun acknowledges support from the Office of Naval Research (ONR N00014-18-1-2759). Hua Zhou acknowledges support from

the National Institutes of Health (R01GM053275 and R01HG006139). Finally, this research collaboration was partially funded by the National Science Foundation under grant DMS-1127914 (the Statistical and Applied Mathematical Sciences Institute). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation, the National Institutes of Health, or the Office of Naval Research.

Appendix A. Tensor Decompositions

We review two basic tensor decompositions that generalize the singular value decomposition (SVD) of a matrix: (i) the CANDECOMP/PARAFAC (CP) decomposition (Carroll and Chang, 1970; Harshman, 1970) and (ii) the Tucker decomposition (Tucker, 1966). Just as the SVD can be used to construct a lower-dimensional approximation to a data matrix, these two decompositions can be used to construct a lower dimensional approximation to a D-way tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_D}$

The CP decomposition aims to approximate $\mathcal{X}$ by a sum of rank-one tensors, namely

$$\mathcal{X} \approx \sum_{i=1}^{R} \mathbf{a}_i^{(1)} \circ \mathbf{a}_i^{(2)} \circ \cdots \circ \mathbf{a}_i^{(D)},$$

where $\circ$ represents the outer product and $\mathbf{a}_i^{(d)}$ is the ith column of the dth factor matrix $\mathbf{A}^{(d)} \in \mathbb{R}^{n_d \times R}$. The positive integer R denotes the rank of the approximation. For sufficiently large R, we can exactly represent $\mathcal{X}$ with a CP decomposition.

The Tucker decomposition aims to approximate $\mathcal{X}$ by a core tensor $\mathcal{H} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_D}$ multiplied by factor matrices along each of its modes, namely

$$\mathcal{X} \approx \mathcal{H} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \times_3 \cdots \times_D \mathbf{A}^{(D)} = \sum_{i_1=1}^{R_1} \sum_{i_2=1}^{R_2} \cdots \sum_{i_D=1}^{R_D} h_{i_1 i_2 \cdots i_D} \mathbf{a}_{i_1}^{(1)} \circ \mathbf{a}_{i_2}^{(2)} \circ \cdots \circ \mathbf{a}_{i_D}^{(D)},$$

where $\mathbf{a}_{i_d}^{(d)}$ is the i_dth column of the dth factor matrix $\mathbf{A}^{(d)} \in \mathbb{R}^{n_d \times R_d}$. Typically the columns of $\mathbf{A}^{(d)}$ are computed to be orthonomal and can be interpreted as principal components or basis vectors for the dth mode. For sufficiently large $R_1, \ldots, R_D$, we can exactly represent $\mathcal{X}$ with a Tucker decomposition.

Appendix B. Proofs of Smoothness Properties

B.1 Proof of Proposition 4

Without loss of generality, we can absorb γ into the weights matrices. Thus, we seek to show the continuity of $\hat{\mathcal{U}}$ with respect to $(\mathcal{X}, \mathbf{W}_1, \ldots, \mathbf{W}_D)$. We use the following compact representation of the weights

$$\mathbf{w} = (\mathrm{vec}(\mathbf{W}_1)^\mathsf{T}, \mathrm{vec}(\mathbf{W}_2)^\mathsf{T}, \ldots, \mathrm{vec}(\mathbf{W}_D))^\mathsf{T} \in \mathbb{R}^{\sum_{d=1}^{D} \binom{n_d}{2}}.$$

We check to see if the solution $\hat{\mathcal{U}}$ is continuous in the variable $\boldsymbol{\zeta} = (\mathbf{x}^\mathsf{T}, \mathbf{w}^\mathsf{T})^\mathsf{T}$. It is easy to verify that the following function is jointly continuous in $\mathcal{U}$ and $\boldsymbol{\zeta}$

$$f(\mathcal{U}, \boldsymbol{\zeta}) = \frac{1}{2}\|\mathcal{X} - \mathcal{U}\|_{\mathrm{F}}^2 + R(\mathcal{U}, \mathbf{w}),$$

where

$$R(\mathcal{U}, \mathbf{w}) \quad = \quad \sum_{d=1}^{D} \sum_{i<j} w_{d,ij} \|\mathcal{U} \times_d \mathbf{\Delta}_{d,ij}\|_{\mathrm{F}}$$

is a convex function of $\mathcal{U}$ that is continuous in $(\mathcal{U}, \mathbf{w})$. Let

$$\mathcal{U}^\star(\boldsymbol{\zeta}) \quad = \quad \arg\min_{\mathcal{U}} f(\mathcal{U}, \boldsymbol{\zeta}).$$

Since $f(\mathcal{U}, \boldsymbol{\zeta})$ is strongly convex in $\mathcal{U}$, the minimizer $\mathcal{U}^\star(\boldsymbol{\zeta})$ exists and is unique.

We proceed with a proof by contradiction. Suppose $\mathcal{U}^\star(\boldsymbol{\zeta})$ is not continuous at a point $\boldsymbol{\zeta}$. Then there exists an $\epsilon > 0$ and a sequence $\{\boldsymbol{\zeta}^{(m)}\}$ converging to a limit $\boldsymbol{\zeta}$ such that $\|\mathcal{U}^{(m)} - \mathcal{U}^\star(\boldsymbol{\zeta})\|_{\mathrm{F}} \geq \epsilon$ for all m where

$$\mathcal{U}^{(m)} \quad = \quad \arg\min_{\mathcal{U}} f(\mathcal{U}, \boldsymbol{\zeta}^{(m)}).$$

Since $f(\mathcal{U}, \boldsymbol{\zeta})$ is strongly convex in $\mathcal{U}$, the minimizer $\mathcal{U}^{(m)}$ exists and is unique. Without loss of generality, we can assume $\|\boldsymbol{\zeta}^{(m)} - \boldsymbol{\zeta}\|_{\mathrm{F}} \leq 1$. This fact will be used later in proving the boundedness of the sequence $\mathcal{U}^{(m)}$.

If $\mathcal{U}^{(m)}$ is a bounded sequence, then we can pass to a convergent subsequence with limit $\bar{\mathcal{U}}$. Fix an arbitrary point $\tilde{\mathcal{U}}$. Note that $f(\mathcal{U}^{(m)}, \boldsymbol{\zeta}^{(m)}) \leq f(\tilde{\mathcal{U}}, \boldsymbol{\zeta}^{(m)})$ for all m. Since f is continuous in $(\mathcal{U}, \boldsymbol{\zeta})$, taking limits gives us the inequality

$$f(\bar{\mathcal{U}}, \boldsymbol{\zeta}) \quad \leq \quad f(\tilde{\mathcal{U}}, \boldsymbol{\zeta}).$$

Since $\tilde{\mathcal{U}}$ was selected arbitrarily, it follows that $\bar{\mathcal{U}} = \mathcal{U}^\star(\boldsymbol{\zeta})$, which is a contradiction. It only remains for us to show that the sequence $\mathcal{U}^{(m)}$ is bounded.

Consider the function

$$g(\mathcal{U}) \quad = \quad \sup_{\tilde{\boldsymbol{\zeta}}: \|\tilde{\boldsymbol{\zeta}} - \boldsymbol{\zeta}\|_{\mathrm{F}} \leq 1} \frac{1}{2} \|\tilde{\mathcal{X}} - \mathcal{U}\|_{\mathrm{F}}^2 + R_{\tilde{\mathbf{w}}}(\mathcal{U}).$$

Note that g is convex, since it is the point-wise supremum of a collection of convex functions. Since $f(\mathcal{U}, \boldsymbol{\zeta}^{(m)}) \leq g(\mathcal{U})$ and f is strongly convex in $\mathcal{U}$, it follows that $g(\mathcal{U})$ is also strongly convex and therefore has a unique global minimizer $\mathcal{U}^*$ such that $g(\mathcal{U}^*) < \infty$. It also follows that

$$f(\mathcal{U}^{(m)}, \boldsymbol{\zeta}^{(m)}) \quad \leq \quad f(\mathcal{U}^*, \boldsymbol{\zeta}^{(m)}) \quad \leq \quad g(\mathcal{U}^*) \tag{14}$$

for all m. By the reverse triangle inequality it follows that

$$\frac{1}{2} \left(\|\mathcal{U}^{(m)}\|_{\mathrm{F}} - \|\mathcal{X}^{(m)}\|_{\mathrm{F}} \right)^2 \quad \leq \quad \frac{1}{2} \|\mathcal{U}^{(m)} - \mathcal{X}^{(m)}\|_{\mathrm{F}}^2 \quad \leq \quad f(\mathcal{U}^{(m)}, \boldsymbol{\zeta}^{(m)}). \tag{15}$$

Combining the inequalities in (14) and (15), we arrive at the conclusion that

$$\frac{1}{2} \left(\|\mathcal{U}^{(m)}\|_{\mathrm{F}} - \|\mathcal{X}^{(m)}\|_{\mathrm{F}} \right)^2 \quad \leq \quad g(\mathcal{U}^*),$$

for all m. Suppose the sequence $\mathcal{U}^{(m)}$ is unbounded, namely $\|\mathcal{U}^{(m)}\|_{\mathrm{F}} \to \infty$. But since $\mathcal{X}^{(m)}$ converges to $\mathcal{X}$, the left hand side must diverge. Thus, we arrive at a contradiction if $\mathcal{U}^{(m)}$ is unbounded. ■

B.2 Proof of Proposition 5

First suppose that $\mathbf{U}_{(d)} = \mathbf{1}\mathbf{c}^\mathsf{T}$, namely all the mode-$d$ subarrays of $\mathcal{U}$ are identical. Recall that $\mathcal{Z} = \mathcal{U} \times_d \mathbf{A}$ if and only if $\mathbf{Z}_{(d)} = \mathbf{A}\mathbf{U}_{(d)}$. Therefore, $R_d(\mathcal{U}) = 0$ since $\mathbf{\Delta}_{d,ij}\mathbf{1}\mathbf{c}^\mathsf{T} = 0$ for all $(i,j) \in \mathcal{E}_d$.

Now suppose that $R_d(\mathcal{U})$ is zero. Take an arbitrary pair (i,j) with $i < j$. By Assumption 4.1, there exists a path $i \to k \to \cdots \to l \to j$ along which the weights are positive. Let w denote the smallest weight along this path, namely $w = \min\{w_{d,ik}, \ldots, w_{d,lj}\}$. By the triangle inequality

$$\|\mathcal{U} \times_d \mathbf{\Delta}_{d,ij}\|_\mathrm{F} \quad \leq \quad \|\mathcal{U} \times_d \mathbf{\Delta}_{d,ik}\|_\mathrm{F} + \cdots + \|\mathcal{U} \times_d \mathbf{\Delta}_{d,lj}\|_\mathrm{F}.$$

We can then conclude that

$$w\|\mathcal{U} \times_d \mathbf{\Delta}_{d,ij}\|_\mathrm{F} \quad \leq \quad R_d(\mathcal{U}) \quad = \quad 0.$$

It follows that $\mathbf{e}_i^\mathsf{T}\mathbf{U}_{(d)} = \mathbf{e}_j^\mathsf{T}\mathbf{U}_{(d)}$, since w is positive. Since the pair (i,j) is arbitrary, it follows that all the rows of $\mathbf{U}_{(d)}$ are identical or in other words, $\mathbf{U}_{(d)} = \mathbf{1}\mathbf{c}^\mathsf{T}$ for some $\mathbf{c} \in \mathbb{R}^{n-d}$. ■

B.3 Proof of Proposition 6

We will show that there is a $\gamma_{\max}$ such that for all $\gamma \geq \gamma_{\max}$, the grand mean tensor $\bar{\mathcal{X}}$ is the unique global minimizer to the primal objective (4). We will certify that $\bar{\mathcal{X}}$ is the solution to the primal problem by showing that the optimal value of a dual problem, which lower bounds the primal, equals $F_\gamma(\bar{\mathcal{X}})$.

Note that the Lagrangian dual given in (28) is a tight lower bound on $F_\gamma(\mathcal{U})$.

$$\max_{\boldsymbol{\lambda} \in C_\gamma} \; -\frac{1}{2}\|\mathbf{A}^\mathsf{T}\boldsymbol{\lambda}\|_2^2 + \langle \boldsymbol{\lambda}, \mathbf{A}\mathbf{x} \rangle.$$

For sufficiently large γ, the solution to the dual maximization problem coincides with the solution to the unconstrained maximization problem

$$\max_{\boldsymbol{\lambda}} \; -\frac{1}{2}\|\mathbf{A}^\mathsf{T}\boldsymbol{\lambda}\|_2^2 + \langle \boldsymbol{\lambda}, \mathbf{A}\mathbf{x} \rangle,$$

whose solution is $\boldsymbol{\lambda}^\star = \left(\mathbf{A}\mathbf{A}^\mathsf{T}\right)^\dagger \mathbf{A}\mathbf{x}$. Plugging $\boldsymbol{\lambda}^\star$ into the dual objective gives an optimal value of

$$\frac{1}{2}\|\mathbf{A}^\mathsf{T}\left(\mathbf{A}\mathbf{A}^\mathsf{T}\right)^\dagger \mathbf{A}\mathbf{x}\|_2^2 \quad = \quad \frac{1}{2}\|\mathbf{x} - \left[\mathbf{I} - \mathbf{A}^\mathsf{T}\left(\mathbf{A}\mathbf{A}^\mathsf{T}\right)^\dagger \mathbf{A}\right]\mathbf{x}\|_2^2.$$

Note that $\left[\mathbf{I} - \mathbf{A}^\mathsf{T}\left(\mathbf{A}\mathbf{A}^\mathsf{T}\right)^\dagger \mathbf{A}\right]$ is the projection onto the orthogonal complement of the column space of $\mathbf{A}^\mathsf{T}$, which is equivalent to the null space or kernel of $\mathbf{A}$, denoted $\mathrm{Ker}(\mathbf{A})$. We will show below that $\mathrm{Ker}(\mathbf{A})$ is the span of the all ones vector. Consequently,

$$\left[\mathbf{I} - \mathbf{A}^\mathsf{T}\left(\mathbf{A}\mathbf{A}^\mathsf{T}\right)^\dagger \mathbf{A}\right]\mathbf{x} \quad = \quad \frac{1}{n}\langle \mathbf{x}, \mathbf{1} \rangle \mathbf{1}.$$

Note that the smallest γ such that $\boldsymbol{\lambda}^\star \in C_\gamma$ is an upper bound on $\gamma_{\max}$.

We now argue that $\mathrm{Ker}(\mathbf{A})$ is the span of $\mathbf{1} \in \mathbb{R}^n$. We rely on the following fact: If $\boldsymbol{\Phi}_d$ is an incidence matrix of a connected graph with n_d vertices, then the rank of $\boldsymbol{\Phi}_d$ is $n_d - 1$ (Deo, 1974, Theorem 7.2). According to Assumption 4.1, the mode-d graphs are connected; it follows that $\boldsymbol{\Phi}_d \in \{-1, 0, 1\}^{|\mathcal{E}_d| \times n_d}$ has rank $n_d - 1$. It follows then that $\mathrm{Ker}(\boldsymbol{\Phi}_d)$ has dimension one. Furthermore, since each row of $\boldsymbol{\Phi}_d$ has one 1 and one -1, it follows that $\mathbf{1} \in \mathrm{Ker}(\boldsymbol{\Phi}_d) \subset \mathbb{R}^{n_d}$. A vector $\mathbf{z} \in \mathrm{Ker}(\mathbf{A})$ if and only if $\mathbf{z} \in \mathrm{Ker}(\mathbf{A}_d)$ for all d.

Recall that the rank of the Kronecker product $\mathbf{A} \otimes \mathbf{B}$ is the product of the ranks of the matrices $\mathbf{A}$ and $\mathbf{B}$. This rank property of Kronecker products of matrices implies that the dimension of $\mathrm{Ker}(\mathbf{A}_d)$ equals n_{-d}. Let $\mathbf{b}_i = \mathbf{1}_{n_D} \otimes \cdots \otimes \mathbf{1}_{n_{d+1}} \otimes \mathbf{e}_i \otimes \mathbf{1}_{n_{d-1}} \otimes \cdots \otimes \mathbf{1}_{n_1}$ where $\mathbf{1}_p \in \mathbb{R}^p$ is the vector of all ones and $\mathbf{e}_i \in \mathbb{R}^{n_d}$ is the ith standard basis vector. Then that the set of vectors $\mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{n_d}\}$ forms a basis for $\mathrm{Ker}(\mathbf{A}_d)$.

Take an arbitrary element from $\mathrm{Ker}(\mathbf{A}_d)$, namely a vector of the form $\mathbf{1}_{n'} \otimes \mathbf{a} \otimes \mathbf{1}_{n''}$, where $n' = \prod_{j=d+1}^{D} n_j$ and $n'' = \prod_{j=1}^{d-1}$. We will show that in order for $\mathbf{1}_{n'} \otimes \mathbf{a} \otimes \mathbf{1}_{d''} \in \mathrm{Ker}(\mathbf{I} \otimes \boldsymbol{\Phi}_d)$, $\mathbf{a}$ must be a multiple of $\mathbf{1}_{n_d}$. Consider the relevant matrix-vector product

$$\mathbf{A}_d\Big(\mathbf{1}_{n_D} \otimes \mathbf{a} \otimes \mathbf{1}_{n_1}\Big) \quad = \quad (\mathbf{1}_{n_D} \otimes \cdots \otimes \mathbf{1}_{n_{d+1}} \otimes \boldsymbol{\Phi}_d \mathbf{a} \otimes \mathbf{1}_{n_{d-1}} \otimes \cdots \otimes \mathbf{1}_{n_1}).$$

Therefore, $\mathbf{A}_d\Big(\mathbf{1}_{n'} \otimes \mathbf{a} \otimes \mathbf{1}_{n''}\Big) = \mathbf{0}$ if and only if $\boldsymbol{\Phi}_d \mathbf{a} = \mathbf{0}$. But the only way for $\boldsymbol{\Phi}_d \mathbf{a}$ to be zero is for $\mathbf{a} = c\mathbf{1}_{n_d}$ for some $c \in \mathbb{R}$. Thus, $\mathrm{Ker}(\mathbf{A})$ is the span of $\mathbf{1}_n$. ■

B.4 Proof of Proposition 7

Note that $\hat{\boldsymbol{\mathcal{U}}}$ is the proximal mapping of the closed, convex function

$$\sum_{d=1}^{D} R_d(\boldsymbol{\mathcal{U}})$$

Then $\hat{\boldsymbol{\mathcal{U}}}$ is firmly nonexpansive in $\boldsymbol{\mathcal{X}}$ (Combettes and Wajs, 2005, Lemma 2.4). Finally, firmly nonexpansive mappings are nonexpansive, which completes the proof. ■

Appendix C. Proof of Theorem 9

We first prove some auxiliary lemmas before proving our prediction error result.

C.1 Auxiliary Lemmas

The following lemma considers the concentration of a random quadratic form $\mathbf{y}^\mathsf{T}\mathbf{B}\mathbf{y}$ for a M-concentrated random vector $\mathbf{y}$ and a deterministic matrix $\mathbf{B}$ (Vu and Wang, 2015). It can be viewed as a generalization of the standard Hanson and Wright inequality for the quadratic forms of independent sub-Gaussian random variables (Hanson and Wright, 1971).

Lemma 11 *Let $\mathbf{y} \in \mathbb{R}^n$ be a M-concentrated random vector, see Definition 8. Then there are constants $C, C' > 0$ such that for any matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$,*

$$\mathbb{P}\Big(|\mathbf{y}^\mathsf{T}\mathbf{B}\mathbf{y} - \mathrm{tr}(\mathbf{B})| \geq t\Big) \quad \leq \quad C \log(n) \exp\left\{-C' M^{-2} \min\left[\frac{t^2}{\|\mathbf{B}\|_F^2 \log(n)}, \frac{t}{\|\mathbf{B}\|_2}\right]\right\}.$$

The next lemma studies the properties of the matrix $\mathbf{A}_{d,ij}$, defined in (6), in the penalty function. Denote $\mathbf{S}_d$ as the matrix constructed by concatenating $\mathbf{A}_{d,ij}, i < j$ vertically. That is,

$$\mathbf{S}_d \quad = \quad \begin{pmatrix} \mathbf{A}_{d,12}^\mathsf{T} & \mathbf{A}_{d,13}^\mathsf{T} & \dots & \mathbf{A}_{d,n_d-1n_d}^\mathsf{T} \end{pmatrix}^\mathsf{T} \in \mathbb{R}^{[\binom{n_d}{2} n_{-d}] \times n}. \tag{16}$$

Lemma 12 *For each $d = 1, \dots, D$, the rank of the matrix $\mathbf{S}_d$ is $(n_d - 1)n_{-d}$. Denote $\sigma_{\min}(\mathbf{S}_d)$ and $\sigma_{\max}(\mathbf{S}_d)$ as the minimum non-zero singular value and maximum singular value of $\mathbf{S}_d$, respectively. We have $\sigma_{\min}(\mathbf{S}_d) = \sigma_{\max}(\mathbf{S}_d) = \sqrt{n_d}$.*

The proof of Lemma 12 follows from Lemma 1 in Tan and Witten (2015) and is omitted. According to Lemma 12, we can construct a singular value decomposition of $\mathbf{S}_d = \mathbf{U}_d \mathbf{\Lambda}_d \mathbf{V}_d^\mathsf{T}$, where $\mathbf{U}_d \in \mathbb{R}^{[\binom{n_d}{2} n_{-d}] \times (n_d - 1)n_{-d}}$, $\mathbf{\Lambda}_d \in \mathbb{R}^{(n_d-1)n_{-d} \times (n_d-1)n_{-d}}$, and $\mathbf{V}_d \in \mathbb{R}^{n \times (n_d-1)n_{-d}}$. Denote

$$\mathbf{G}_d \quad = \quad \mathbf{U}_d \mathbf{\Lambda}_d \in \mathbb{R}^{[\binom{n_d}{2} n_{-d}] \times (n_d-1)n_{-d}}, \tag{17}$$

and its pseudo-inverse as $\mathbf{G}_d^\dagger \in \mathbb{R}^{(n_d-1)n_{-d} \times [\binom{n_d}{2} n_{-d}]}$. The following lemma studies the properties of $\mathbf{G}_d$ and $\mathbf{G}_d^\dagger$, for each $d = 1, \dots, D$.

Lemma 13 *For each $d = 1, \dots, D$, the rank of the matrix $\mathbf{G}_d$ is $(n_d - 1)n_{-d}$. The minimal non-zero singular value and maximal singular value of $\mathbf{G}_d$ are $\sigma_{\min}(\mathbf{G}_d) = \sigma_{\max}(\mathbf{G}_d) = \sqrt{n_d}$. Moreover, $\sigma_{\min}(\mathbf{G}_d^\dagger) = \sigma_{\max}(\mathbf{G}_d^\dagger) = 1/\sqrt{n_d}$.*

Lemma 13 follows directly from the conclusions in Lemma 12.

C.2 Proof of Main Theorem

We first reformulate our optimization problem via a decomposition approach to simplify the theoretical analysis. Such strategy was developed in Liu et al. (2013a) and has been successfully applied in Tan and Witten (2015); Wang et al. (2018).

Denote $\gamma_d = \gamma / n_d$. Our convex tensor co-clustering method is equivalent to solving

$$\hat{\mathbf{u}} \quad = \quad \arg\min_{\mathbf{u}} \left\{ \frac{1}{2} \|\mathbf{x} - \mathbf{u}\|_2^2 + \sum_{d=1}^{D} \gamma_d \sum_{(i,j) \in \mathcal{E}_d} \|\mathbf{A}_{d,ij} \mathbf{u}\|_2 \right\}. \tag{18}$$

According to the definition of $\mathbf{S}_d$ in (16), we define the penalty function $R(\cdot)$ such that

$$R(\mathbf{S}_d \mathbf{u}) \quad = \quad \sum_{(i,j) \in \mathcal{E}_d} \|\mathbf{A}_{d,ij} \mathbf{u}\|_2.$$

According to the singular value decomposition of $\mathbf{S}_d = \mathbf{U}_d \mathbf{\Lambda}_d \mathbf{V}_d^\mathsf{T}$, there exists a matrix $\mathbf{W}_d \in \mathbb{R}^{n \times n_{-d}}$ such that $\tilde{\mathbf{V}}_d = [\mathbf{W}_d, \mathbf{V}_d] \in \mathbb{R}^{n \times n}$ is an orthogonal matrix and $\mathbf{W}_d^\mathsf{T} \mathbf{V}_d = \mathbf{0}$. Let $\boldsymbol{\alpha}_d = \mathbf{W}_d^\mathsf{T} \mathbf{u} \in \mathbb{R}^{n_{-d}}$ and $\boldsymbol{\beta}_d = \mathbf{V}_d^\mathsf{T} \mathbf{u} \in \mathbb{R}^n$. Clearly, we have

$$\mathbf{W}_d \boldsymbol{\alpha}_d + \mathbf{V}_d \boldsymbol{\beta}_d \quad = \quad \mathbf{W}_d \mathbf{W}_d^\mathsf{T} \mathbf{u} + \mathbf{V}_d \mathbf{V}_d^\mathsf{T} \mathbf{u} \quad = \quad \tilde{\mathbf{V}}_d \tilde{\mathbf{V}}_d^\mathsf{T} \mathbf{u} \quad = \quad \mathbf{u}, \tag{19}$$

for any $d = 1, \ldots, D$. This fact together with the definition of $\mathbf{G}_d = \mathbf{U}_d \mathbf{\Lambda}_d$ in (17) imply that solving our convex tensor clustering in (18) is equivalent to solving

$$\min_{\boldsymbol{\alpha}_d, \boldsymbol{\beta}_d, d=1,\ldots,D} \sum_{d=1}^{D} \left\{ \frac{1}{2D} \|\mathbf{x} - \mathbf{W}_d \boldsymbol{\alpha}_d + \mathbf{V}_d \boldsymbol{\beta}_d\|_2^2 + \gamma_d R(\mathbf{G}_d \boldsymbol{\beta}_d) \right\} \tag{20}$$

Denote the solution of (20) as $\hat{\boldsymbol{\alpha}}_d, \hat{\boldsymbol{\beta}}_d$, $d = 1, \ldots, D$, which corresponds to the estimator $\hat{\mathbf{u}}$ in (18) according to (19). Similarly, we denote the true parameters as $\boldsymbol{\alpha}_d^*, \boldsymbol{\beta}_d^*$ that corresponds to $\mathbf{u}^*$ defined in Assumption 4.2. Our goal is to derive the upper bound of $\|\hat{\mathbf{u}} - \mathbf{u}^*\|_2^2$ by above reparametrization. Since $\hat{\boldsymbol{\alpha}}_d, \hat{\boldsymbol{\beta}}_d$, $d = 1, \ldots, D$ minimizes the objective function in (20), we have

$$\begin{aligned} &\sum_{d=1}^{D} \left\{ \frac{1}{2D} \|\mathbf{x} - \mathbf{W}_d \hat{\boldsymbol{\alpha}}_d + \mathbf{V}_d \hat{\boldsymbol{\beta}}_d\|_2^2 + \gamma_d R(\mathbf{G}_d \hat{\boldsymbol{\beta}}_d) \right\} \\ \leq \quad &\sum_{d=1}^{D} \left\{ \frac{1}{2D} \|\mathbf{x} - \mathbf{W}_d \boldsymbol{\alpha}_d^* + \mathbf{V}_d \boldsymbol{\beta}_d^*\|_2^2 + \gamma_d R(\mathbf{G}_d \boldsymbol{\beta}_d^*) \right\}. \end{aligned}$$

Note that $\|\mathbf{x} - \hat{\mathbf{u}}\|_2^2 - \|\mathbf{x} - \mathbf{u}^*\|_2^2 = \|\hat{\mathbf{u}}\|_2^2 - \|\mathbf{u}^*\|_2^2 - 2\mathbf{x}^\mathsf{T}(\hat{\mathbf{u}} - \mathbf{u}^*) = \|\hat{\mathbf{u}} - \mathbf{u}^*\|_2^2 + 2\boldsymbol{\epsilon}^\mathsf{T}(\hat{\mathbf{u}} - \mathbf{u}^*)$, where the last equality is due to the model assumption $\mathbf{x} = \mathbf{u}^* + \boldsymbol{\epsilon}$. Therefore, we have

$$\begin{aligned} \frac{1}{2}\|\hat{\mathbf{u}} - \mathbf{u}^*\|_2^2 + \sum_{d=1}^{D} \gamma_d R(\mathbf{G}_d \hat{\boldsymbol{\beta}}_d) \quad &\leq \quad \frac{1}{2D} \sum_{d=1}^{D} \boldsymbol{\epsilon}^\mathsf{T}(\mathbf{u}^* - \hat{\mathbf{u}}) + \sum_{d=1}^{D} \gamma_d R(\mathbf{G}_d \boldsymbol{\beta}_d^*) \\ \leq \quad \frac{1}{2D} \sum_{d=1}^{D} \underbrace{\left| \boldsymbol{\epsilon}^\mathsf{T} [\mathbf{W}_d(\boldsymbol{\alpha}_d^* - \hat{\boldsymbol{\alpha}}_d) + \mathbf{V}_d(\boldsymbol{\beta}_d^* - \hat{\boldsymbol{\beta}}_d)] \right|}_{f(\hat{\boldsymbol{\alpha}}_d, \hat{\boldsymbol{\beta}}_d)} &+ \sum_{d=1}^{D} \gamma_d R(\mathbf{G}_d \boldsymbol{\beta}_d^*). \end{aligned} \tag{21}$$

Next we derive the bound for $f(\hat{\boldsymbol{\alpha}}_d, \hat{\boldsymbol{\beta}}_d)$. Note that the optimization over $\boldsymbol{\alpha}_d$ in (20) has a closed-form since the penalty term is independent of $\boldsymbol{\alpha}_d$. In particular, by setting the derivative of $\|\mathbf{x} - \mathbf{W}_d \boldsymbol{\alpha}_d + \mathbf{V}_d \boldsymbol{\beta}_d\|_2^2$ with respect to $\boldsymbol{\alpha}_d$ to be zero, we obtain that $\boldsymbol{\alpha}_d = \mathbf{W}_d^\mathsf{T}(\mathbf{x} - \mathbf{V}_d \boldsymbol{\beta}_d)$. This implies that

$$\begin{aligned} \hat{\boldsymbol{\alpha}}_d &= \mathbf{W}_d^\mathsf{T}(\mathbf{x} - \mathbf{V}_d \hat{\boldsymbol{\beta}}_d) \\ &= \mathbf{W}_d^\mathsf{T}(\mathbf{W}_d \boldsymbol{\alpha}_d^* + \mathbf{V}_d \boldsymbol{\beta}_d^* + \boldsymbol{\epsilon} - \mathbf{V}_d \hat{\boldsymbol{\beta}}_d) \\ &= \boldsymbol{\alpha}_d^* + \mathbf{W}_d^\mathsf{T} \boldsymbol{\epsilon}, \end{aligned} \tag{22}$$

where the second equality is due to $\mathbf{x} = \mathbf{u}^* + \boldsymbol{\epsilon}$ and the last equality is due to the fact that $\mathbf{W}_d^\mathsf{T} \mathbf{V}_d = 0$ and $\mathbf{W}_d^\mathsf{T} \mathbf{W}_d = \mathbf{I}$. According to (22), we have

$$\begin{aligned} f(\hat{\boldsymbol{\alpha}}_d, \hat{\boldsymbol{\beta}}_d) &= \left| \boldsymbol{\epsilon}^\mathsf{T} \mathbf{W}_d \mathbf{W}_d^\mathsf{T} \boldsymbol{\epsilon} + \boldsymbol{\epsilon}^\mathsf{T} \mathbf{V}_d (\boldsymbol{\beta}_d^* - \hat{\boldsymbol{\beta}}_d)] \right| \\ &\leq \underbrace{\left| \boldsymbol{\epsilon}^\mathsf{T} \mathbf{W}_d \mathbf{W}_d^\mathsf{T} \boldsymbol{\epsilon} \right|}_{(I)} + \underbrace{\left| \boldsymbol{\epsilon}^\mathsf{T} \mathbf{V}_d (\boldsymbol{\beta}_d^* - \hat{\boldsymbol{\beta}}_d)] \right|}_{(II)}. \end{aligned} \tag{23}$$

Bound (I)**:** We apply the concentration inequality in Lemma 11 to bound (I). It remains to compute $\|\mathbf{W}_d\mathbf{W}_d^\mathsf{T}\|_\mathrm{F}^2$ and $\|\mathbf{W}_d\mathbf{W}_d^\mathsf{T}\|_2$. By construction, $\mathbf{W}_d\mathbf{W}_d^\mathsf{T} \in \mathbb{R}^{n\times n}$ is a projection matrix since $\tilde{\mathbf{V}}_d\tilde{\mathbf{V}}_d^\mathsf{T} = \mathbf{W}_d\mathbf{W}_d^\mathsf{T} + \mathbf{V}_d\mathbf{V}_d^\mathsf{T} = \mathbf{I}$. Therefore, the rank of $\mathbf{W}_d\mathbf{W}_d^\mathsf{T}$ is $\prod_{j\neq d} n_j$, $\|\mathbf{W}_d\mathbf{W}_d^\mathsf{T}\|_\mathrm{F}^2 = \prod_{j\neq d} n_j, \|\mathbf{W}_d\mathbf{W}_d^\mathsf{T}\|_2 = 1$, and $\mathrm{tr}(\mathbf{W}_d\mathbf{W}_d^\mathsf{T}) = \prod_{j\neq d} n_j$.

Denote $n = \prod_{d=1}^D n_d$. By Lemma 11 and Assumption 4.2, we have

$$\mathbb{P}\Big(\boldsymbol{\epsilon}^\mathsf{T}\mathbf{W}_d\mathbf{W}_d^\mathsf{T}\boldsymbol{\epsilon} \geq t + n_{-d}\Big) \;\leq\; C\log(n)\exp\left\{-C'M^{-2}\min\left[\frac{t^2}{\log(n)n_{-d}}, t\right]\right\}.$$

Setting $t = \sqrt{n_{-d}\log(n)^2}$, we have

$$\mathbb{P}\left(\boldsymbol{\epsilon}^\mathsf{T}\mathbf{W}_d\mathbf{W}_d^\mathsf{T}\boldsymbol{\epsilon} \geq \log(n)\sqrt{n_{-d}} + n_{-d}\right) \;\leq\; C\exp\Big\{\log\log(n) - C'M^{-2}\log(n)\Big\}, \quad (24)$$

where the right hand side converges to zero as the dimension $n = \prod_{d=1}^D n_d \to \infty$. Note that our error $\boldsymbol{\epsilon}$ in Assumption 4.2 is assumed to be a M-concentrated random variable. If we assume a stronger condition such that $\boldsymbol{\epsilon}$ is a vector with iid sub-Gaussian, we can obtain a upper bound $\sqrt{\log(n)n_{-d}} + n_{-d}$ according to the Hanson and Wright inequality (Hanson and Wright, 1971). Therefore, in spite of the relaxation in the error assumption, our bound in (24) is only up to a log-term larger.

Bound (II)**:** By definitions of $\mathbf{G}_d$ in (17) and $\mathbf{G}_d^\dagger$, we have $\mathbf{G}_d^\dagger\mathbf{G}_d = \mathbf{I}$. Furthermore, let $\mathbf{G}_{d,ij}^\dagger$ refer to the column of $\mathbf{G}_d^\dagger$ that corresponds to the index (i,j), and let $\mathbf{G}_{d,ij}$ refer to the row of $\mathbf{G}_d$ that corresponds to the index (i,j). We have

$$\begin{aligned}(II) \;&=\; \Big|\boldsymbol{\epsilon}^\mathsf{T}\mathbf{V}_d(\boldsymbol{\beta}_d^* - \hat{\boldsymbol{\beta}}_d)\Big| \;=\; \Big|\boldsymbol{\epsilon}^\mathsf{T}\mathbf{V}_d\mathbf{G}_d^\dagger\mathbf{G}_d(\boldsymbol{\beta}_d^* - \hat{\boldsymbol{\beta}}_d)\Big| \;=\; \left|\sum_{i<j}\boldsymbol{\epsilon}^\mathsf{T}\mathbf{V}_d\mathbf{G}_{d,ij}^\dagger\mathbf{G}_{d,ij}(\boldsymbol{\beta}_d^* - \hat{\boldsymbol{\beta}}_d)\right| \\ &\leq\; \sum_{i<j}\|\boldsymbol{\epsilon}^\mathsf{T}\mathbf{V}_d\mathbf{G}_{d,ij}^\dagger\|_2\|\mathbf{G}_{d,ij}(\boldsymbol{\beta}_d^* - \hat{\boldsymbol{\beta}}_d)\|_2 \;\leq\; \underbrace{\max_{i<j}\|\boldsymbol{\epsilon}^\mathsf{T}\mathbf{V}_d\mathbf{G}_{d,ij}^\dagger\|_2}_{II_1}\cdot\sum_{i<j}\|\mathbf{G}_{d,ij}(\boldsymbol{\beta}_d^* - \hat{\boldsymbol{\beta}}_d)\|_2\end{aligned}$$

Bound II_1**:** By construction, $\boldsymbol{\epsilon}^\mathsf{T}\mathbf{V}_d\mathbf{G}_{d,ij}^\dagger \in \mathbb{R}^{n_{-d}}$. We have

$$\|\boldsymbol{\epsilon}^\mathsf{T}\mathbf{V}_d\mathbf{G}_{d,ij}^\dagger\|_2 \;\leq\; \sqrt{n_{-d}}\,\|\boldsymbol{\epsilon}^\mathsf{T}\mathbf{V}_d\mathbf{G}_{d,ij}^\dagger\|_\infty,$$

and hence

$$\max_{i<j}\|\boldsymbol{\epsilon}^\mathsf{T}\mathbf{V}_d\mathbf{G}_{d,ij}^\dagger\|_2 \;\leq\; \sqrt{n_{-d}}\max_{i<j}\|\boldsymbol{\epsilon}^\mathsf{T}\mathbf{V}_d\mathbf{G}_{d,ij}^\dagger\|_\infty \;=\; \sqrt{n_{-d}}\,\|\boldsymbol{\epsilon}^\mathsf{T}\mathbf{V}_d\mathbf{G}_d^\dagger\|_\infty$$

Let $\eta_j = \mathbf{e}_j^\mathsf{T}\mathbf{G}_d^{\dagger\top}\mathbf{V}_d^\mathsf{T}\boldsymbol{\epsilon} \in \mathbb{R}$, where $\mathbf{e}_j \in \mathbb{R}^{\binom{n_d}{2}n_{-d}}$ is the basis vector with the jth entry one and the rest zeros. According to Lemma 13 and the property of $\mathbf{V}_d$ which consists of singular vectors, we have $\sigma_{\max}(\mathbf{V}_d) = 1$ and $\sigma_{\max}(\mathbf{G}_d^\dagger) = 1/\sqrt{n_d}$. Therefore, we have η_j is a $M/\sqrt{n_d}$-concentrated random variable with mean zero. According to the definition of concentrated random variable in Definition 8, we have

$$\mathbb{P}\left(|\eta_j| \geq t_1\right) \;\leq\; C_1\exp\left(-\frac{C_2 n_d t_1^2}{M^2}\right).$$

Therefore, by union bound, we have

$$\mathbb{P}\left(\max_j |\eta_j| \geq t_1\right) \leq C_1 \binom{n_d}{2} (n_{-d}) \exp\left(-\frac{C_2 n_d t_1^2}{M^2}\right).$$

By setting $t_1 = \sqrt{\log(n)\log\left[\binom{n_d}{2} n_{-d}\right]/n_d}$, we have

$$\mathbb{P}\left(\|\boldsymbol{\epsilon}^\mathsf{T}\mathbf{V}_d\mathbf{G}_d^\dagger\|_\infty \geq \sqrt{\log(n)\log\left[\binom{n_d}{2} n_{-d}\right]/n_d}\right) \quad \leq \quad \frac{C_3}{n},$$

for some constant $C_3 > 0$. Hence with probability at least $1 - C_3/n$, we have

$$II_1 \quad \leq \quad \sqrt{n_{-d}\log(n)\log\left[\binom{n_d}{2} n_{-d}\right]/n_d}. \tag{25}$$

Plugging the results in (24) and (25) into (23), we obtain that, for each $d = 1, \ldots, D$

$$f(\hat{\boldsymbol{\alpha}}_d, \hat{\boldsymbol{\beta}}_d) \quad \leq \quad \log(n)\sqrt{n_{-d}} + n_{-d} + \sqrt{n_{-d}\log(n)\log\left[\binom{n_d}{2} n_{-d}\right]/n_d} \sum_{i<j} \|\mathbf{G}_{d,ij}(\boldsymbol{\beta}_d^* - \hat{\boldsymbol{\beta}}_d)\|_2.$$

Therefore, Assumption 4.3 on the tuning parameter γ_d implies that

$$f(\hat{\boldsymbol{\alpha}}_d, \hat{\boldsymbol{\beta}}_d) \quad \leq \quad \log(n)\sqrt{n_{-d}} + n_{-d} + D\gamma_d \sum_{d=1}^{D} \gamma_d \sum_{i<j} \|\mathbf{G}_{d,ij}(\boldsymbol{\beta}_d^* - \hat{\boldsymbol{\beta}}_d)\|_2,$$

by noting that $\log(\binom{n_d}{2} n_{-d}) \leq \log(n_d^2 n_{-d}) \leq 2\log(n)$. This combines with the inequality in (21) lead to

$$\frac{1}{2}\|\hat{\mathbf{u}} - \mathbf{u}^*\|_2^2 \quad \leq \quad \frac{1}{2D}\sum_{d=1}^{D}\left[\log(n)\sqrt{n_{-d}} + n_{-d}\right] + \frac{3}{2}\sum_{d=1}^{D}\gamma_d R(\mathbf{S}_d\mathbf{u}^*). \tag{26}$$

According to the cluster structure assumption in Assumption 4.2, there are k_d clusters along the dth mode of the tensor. Therefore, along each mode the true parameter $\boldsymbol{\mathcal{U}}^*$ only has a few different slices. Denote $\boldsymbol{\mathcal{U}}^*_{\ldots i \ldots}$ as the i-th mode-d subarray. Formally, we have

$$\begin{aligned} R(\mathbf{S}_d\mathbf{u}^*) \quad &= \quad \sum_{(i,j), i<j, i,j=1,\ldots,n_d} \|\mathbf{A}_{d,ij}\mathbf{u}\|_2 \\ &= \quad \sum_{(i,j), i<j, i,j=1,\ldots,n_d} \|\boldsymbol{\mathcal{U}}^*_{\ldots i \ldots} - \boldsymbol{\mathcal{U}}^*_{\ldots j \ldots}\|_{\mathrm{F}} \quad \leq \quad 4C_0^2\binom{n_d}{2}\sqrt{\prod_{j\neq d} k_j}, \end{aligned} \tag{27}$$

where C_0 is a constant upper bound for the entries of $\boldsymbol{\mathcal{U}}^*$. Combining the inequalities in (26) and (27) with the condition on γ_d given in Assumption 4.3 implies that

$$\frac{1}{2}\|\hat{\mathbf{u}} - \mathbf{u}^*\|_2^2 \quad \leq \quad \frac{1}{2D}\sum_{d=1}^{D}\left(\log(n)\sqrt{n_{-d}} + n_{-d}\right) + \frac{3}{2}\sum_{d=1}^{D}\frac{2c_0\log(n)\sqrt{n}}{Dn_d} 4C_0^2\binom{n_d}{2}\sqrt{\prod_{j\neq d} k_j}.$$

Dividing both sides by n gives to the prediction error bound in (7). This ends the proof of Theorem 9. ■

Appendix D. Derivation of Lagrangian Dual

Let $\mathcal{U} \times_d \mathbf{A}$ denote the multiplication of $\mathcal{U}$ along mode d by the matrix $\mathbf{A}$. Recall that for a tensor $\mathcal{U} \in \mathbb{R}^{n_1 \times \cdots \times n_d}$ and a matrix $\mathbf{A} \in \mathbb{R}^{L \times n_d}$

$$\mathrm{vec}(\mathcal{U} \times_d \mathbf{A}) \;=\; (\mathbf{I}_{n_D} \otimes \cdots \otimes \mathbf{I}_{n_{d+1}} \otimes \mathbf{A} \otimes \mathbf{I}_{n_{d-1}} \otimes \cdots \otimes \mathbf{I}_{n_1})\mathbf{u},$$

where $\mathbf{u} = \mathrm{vec}(\mathcal{U}) = \mathrm{vec}(\mathbf{U}_{(1)})$, namely the column-major vectorization of the mode-1 matricization of the tensor $\mathcal{U}$. So, Note that $\mathcal{Y} = \mathcal{U} \times_d \mathbf{A}$ is equivalent to $\mathbf{Y}_{(d)} = \mathbf{A}\mathbf{U}_{(d)}$. We rewrite the penalty function R_d as follows.

$$R_d(\mathcal{U}) \;=\; \sum_{l \in \mathcal{E}_d} w_{d,l} \|\mathcal{U} \times_d \mathbf{\Delta}_{d,l}\|_{\mathrm{F}} \;=\; \sum_{l \in \mathcal{E}_d} w_{d,l} \|\mathrm{vec}(\mathcal{U} \times_d \mathbf{\Delta}_{d,l})\|_2 \;=\; \sum_{l \in \mathcal{E}_d} w_{d,l} \|\mathbf{A}_{d,l}\mathbf{u}\|_2,$$

where $\mathbf{A}_{d,l} = (\mathbf{I}_{n_D} \otimes \cdots \otimes \mathbf{I}_{n_{d+1}} \otimes \mathbf{\Delta}_{d,l} \otimes \mathbf{I}_{n_{d-1}} \otimes \cdots \otimes \mathbf{I}_{n_1})$.

We now write down the Lagrangian:

$$\begin{aligned}
\mathcal{L}(\mathbf{u}, \mathbf{v}, \boldsymbol{\lambda}) &= \frac{1}{2}\|\mathbf{x} - \mathbf{u}\|_2^2 + \sum_{d=1}^{D} \sum_{l \in \mathcal{E}_d} \Big\{ \gamma w_{d,l} \|\mathbf{v}_{d,l}\|_2 + \langle \boldsymbol{\lambda}_{d,l}, \mathbf{A}_{d,l}\mathbf{u} - \mathbf{v}_{d,l} \rangle \Big\} \\
&= \Big\{ \frac{1}{2}\|\mathbf{x} - \mathbf{u}\|_2^2 + \sum_{d=1}^{D} \langle \mathbf{A}_d^{\mathsf{T}} \boldsymbol{\lambda}_d, \mathbf{u} \rangle \Big\} - \sum_{d=1}^{D} \sum_{l \in \mathcal{E}_d} \Big\{ \langle \boldsymbol{\lambda}_{d,l}, \mathbf{v}_{d,l} \rangle - \gamma w_{d,l} \|\mathbf{v}_{d,l}\|_2 \Big\} \\
&= \Big\{ \frac{1}{2}\|\mathbf{x} - \mathbf{u}\|_2^2 + \langle \mathbf{A}^{\mathsf{T}} \boldsymbol{\lambda}, \mathbf{u} \rangle \Big\} - \sum_{d=1}^{D} \sum_{l \in \mathcal{E}_d} \Big\{ \langle \boldsymbol{\lambda}_{d,l}, \mathbf{v}_{d,l} \rangle - \gamma w_{d,l} \|\mathbf{v}_{d,l}\|_2 \Big\}.
\end{aligned}$$

The Lagrangian dual objective is given by $G(\boldsymbol{\lambda})$ by minimizing the Lagrangian $\mathcal{L}(\mathbf{u}, \mathbf{v}, \boldsymbol{\lambda})$ over the primal variables $\mathbf{u}$ and $\mathbf{v}$, namely

$$\begin{aligned}
G(\boldsymbol{\lambda}) &= \min_{\mathbf{u},\mathbf{v}} \mathcal{L}(\mathbf{u}, \mathbf{v}, \boldsymbol{\lambda}) \\
&= \min_{\mathbf{u}} \Big\{ \frac{1}{2}\|\mathbf{x} - \mathbf{u}\|_2^2 + \langle \mathbf{A}^{\mathsf{T}} \boldsymbol{\lambda}, \mathbf{u} \rangle \Big\} - \sum_{d=1}^{D} \sum_{l \in \mathcal{E}_d} \max_{\mathbf{v}_{d,l}} \Big\{ \langle \boldsymbol{\lambda}_{d,l}, \mathbf{v}_{d,l} \rangle - \gamma w_{d,l} \|\mathbf{v}_{d,l}\|_2 \Big\} \\
&= \frac{1}{2}\|\mathbf{x}\|_2^2 - \frac{1}{2}\|\mathbf{x} - \mathbf{A}^{\mathsf{T}} \boldsymbol{\lambda}\|_2^2 - \sum_{d=1}^{D} \sum_{l \in \mathcal{E}_d} \iota_{C_{d,l}}(\boldsymbol{\lambda}_{d,l}), \qquad (28)
\end{aligned}$$

where $\iota_{C_{d,l}}$ is the indicator function of the closed convex set $C_{d,l} = \{\mathbf{z} : \|\mathbf{z}\|_2 \leq \gamma w_{d,l}\}$.

The last equality in (28) follows from the fact that the Fenchel conjugate of a norm is the indicator function of the unit dual norm ball. Recall that the Fenchel conjugate $f^\star$ of a function f is given by

$$f^\star(\boldsymbol{\lambda}) \;=\; \sup_{\mathbf{v}} \Big\{ \langle \boldsymbol{\lambda}, \mathbf{v} \rangle - f(\mathbf{v}) \Big\}.$$

Let $B = \{\boldsymbol{\lambda} : \|\boldsymbol{\lambda}\|_2 \leq 1\}$ denote the unit ℓ_2-norm ball. Since the ℓ_2-norm is self dual, we arrive at the identity

$$\iota_B(\boldsymbol{\lambda}) \;=\; \sup_{\mathbf{v}} \Big\{ \langle \boldsymbol{\lambda}, \mathbf{v} \rangle - \|\mathbf{v}\|_2 \Big\}.$$

Appendix E. Projected Gradient Applied to the Lagrangian Dual

Note that the dual problem (8) has the form

$$\begin{array}{c} \text{minimize } g(\boldsymbol{\lambda}) \\ \text{subject to } \boldsymbol{\lambda} \in C, \end{array} \tag{29}$$

where $g(\boldsymbol{\lambda})$ is a convex and Lipschitz-differentiable function and the constraint set C is a closed convex set, which implies that every point $\boldsymbol{\lambda}$ possesses a unique orthogonal projection, $\mathcal{P}_C(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{\theta} \in C} \|\boldsymbol{\theta} - \boldsymbol{\lambda}\|_2$, onto C. When $\mathcal{P}_C(\boldsymbol{\lambda})$ can be computed analytically, a simple and effective iterative algorithm for solving problems like (29) is the projected gradient descent algorithm, a special case of proximal gradient descent algorithm (Combettes and Wajs, 2005; Combettes and Pesquet, 2011). Recall that projected gradient descent alternates between taking a gradient step and projecting onto the set C. Thus, at the mth iteration, we perform the following update

$$\boldsymbol{\lambda}^{(m)} \;=\; \mathcal{P}_C\left(\boldsymbol{\lambda}^{(m-1)} - \eta \nabla g(\boldsymbol{\lambda})\right), \tag{30}$$

where η is a step-length parameter.

Applying the update rule in (30) to the dual problem (8), we obtain the following rule for computing the mth iteration

$$\begin{aligned} \mathbf{u}^{(m)} &= \mathbf{x} - \mathbf{A}^\mathsf{T} \boldsymbol{\lambda}^{(m-1)} \\ \boldsymbol{\lambda}^{(m)} &= \mathcal{P}_C\left(\boldsymbol{\lambda}^{(m-1)} + \eta \mathbf{A}\mathbf{u}^{(m)}\right). \end{aligned}$$

Note that, at the mth iteration, the gradient of the least squares objective in (8) is given by $-\mathbf{A}\mathbf{u}^{(m)}$. Thus, we automatically update our CoCo estimator $\mathbf{u}^{(m)}$ as part of our gradient calculation. Finally, we note that the projection onto the set C consists of independent projections onto the sets $C_{d,l}$ that can be carried out in parallel.

E.1 Per-Iteration and Storage Costs

The gradient update is dominated by the matrix-vector multiplications $\mathbf{A}^\mathsf{T}\boldsymbol{\lambda}$ and $\mathbf{A}\mathbf{u}$. Although $\mathbf{A}$ is a $\sum_{d=1}^D |\mathcal{E}_d| n_{-d}$-by-$n$ matrix it has only $2\sum_{d=1}^D |\mathcal{E}_d| n_{-d}$ non-zero elements. Thus, computing the gradient step requires $\mathcal{O}(\sum_{d=1}^D |\mathcal{E}_d| n_{-d})$ flops. Projecting onto the set C also requires $\mathcal{O}(\sum_{d=1}^D |\mathcal{E}_d| n_{-d})$ flops since projecting onto the set $C_{d,l}$ requires $\mathcal{O}(n_{-d})$ flops. Thus, the per-iteration cost is $\mathcal{O}(\sum_{d=1}^D |\mathcal{E}_d| n_{-d})$ flops. The storage cost is dominated by storing the dual variable $\boldsymbol{\lambda}$, which has $\sum_{d=1}^D |\mathcal{E}_d| n_{-d}$ elements. At first glance these storage and per-iteration costs may seem prohibitive, as $|\mathcal{E}_d|$ can be as large as $\mathcal{O}(n_d^2)$ for a fully connected mode-d graph. Shrinking together all combinations of pairs of mode-d subarrays, however, typically produces poor clustering results in comparison to shrinking together mode-d subarrays that are nearest-neighbors as observed in prior work in convex clustering (Chen et al., 2015; Chi and Lange, 2015) and convex biclustering (Chi et al., 2017). Consequently, we employ sparse weights. Specifically, we keep positive weights between approximately nearest-neighbor mode-d subarrays so that $|\mathcal{E}_d|$ is $\mathcal{O}(n_d)$. By using these sparse weights, the per-iteration and storage costs scale more reasonably as $\mathcal{O}(Dn)$, namely linearly in either the number of dimensions D or in the number of elements n. Details on our weights choices are elaborated in Section 6.

E.2 Convergence

The sequence of dual iterates $\boldsymbol{\lambda}^{(m)}$ is guaranteed to converge to a solution $\hat{\boldsymbol{\lambda}}$ of (8) provided that the step-size parameter η is less than twice the reciprocal of the spectral radius of the matrix $\mathbf{A}^\mathsf{T}\mathbf{A}$ (Combettes and Wajs, 2005, Theorem 3.4). Consequently, the sequence of primal iterates $\mathbf{u}^{(m)}$ is guaranteed to converge to the CoCo estimator $\hat{\mathbf{u}}$. We note that under the same step-size conditions, convergence of the sequence $\mathbf{u}^{(m)}$ can also be guaranteed by observing that the projected gradient algorithm applied to the dual problem (8) is an example of the alternating minimization algorithm (Tseng, 1991, Proposition 2).

E.3 Monitoring Convergence via the Duality Gap

Recall that we can bound the suboptimality of the mth iterate, $F_\gamma(\mathbf{u}^{(m)}) - F_\gamma(\hat{\mathbf{u}})$, by the duality gap $F_\gamma(\mathbf{u}^{(m)}) - G(\boldsymbol{\lambda}^{(m)})$, which can be expressed solely in terms of the mth iterate of the primal variable $\mathbf{u}^{(m)}$, namely

$$F_\gamma(\mathbf{u}^{(m)}) - G(\boldsymbol{\lambda}^{(m)}) \quad = \quad \|\mathbf{u}^{(m)}\|_2^2 - \langle \mathbf{x}, \mathbf{u}^{(m)} \rangle + \gamma \sum_{d=1}^{D} \sum_{l \in \mathcal{E}_d} w_{d,l} \|\mathbf{A}_{d,l}\mathbf{u}^{(m)}\|_2.$$

For any optimal dual solution $\hat{\boldsymbol{\lambda}}$, the gap vanishes, namely $F_\gamma(\hat{\mathbf{u}}) = G(\hat{\boldsymbol{\lambda}})$. Note that computing the duality gap incurs minimal additional cost as $\mathbf{u}^{(m)}$ and $\mathbf{A}_{d,l}\mathbf{u}^{(m)}$ are already computed as part of the gradient step. In short, including a duality gap computation will not change the $\mathcal{O}(Dn)$ per-iteration cost of the projected gradient algorithm. In practice, we can terminate the algorithm once the duality gap falls below some small tolerance.

E.4 Computing Mode-d Difference Variables

In Section 7.2, we explained how clustering assignments along the dth mode are made using the mode-d difference variables $\mathbf{v}_{d,l} = \mathfrak{U} \times_d \boldsymbol{\Delta}_{d,l}$. In practice we must deal with the fact that the $\hat{\mathbf{u}}$ recovered by computing $\mathbf{x} - \mathbf{A}^\mathsf{T}\hat{\boldsymbol{\lambda}}$ may exhibit a nearly but not exactly checkerbox structure due to limitations in numerical precision. This creates a practical issue as a small but non-zero difference variable will lead to an incorrect clustering assignment. Addressing this issue, however, is simple. The projected gradient algorithm used to compute CoCo is a natural generalization of the projected gradient algorithm used in Chi and Lange (2015) for convex clustering. Consequently, we can use the obvious adaptation of the procedure for computing the differences variables in convex clustering. The following brief technical discussion is expanded in more detail in Chi and Lange (2015).

The key fact that we use is that the projected gradient algorithm is equivalent to the alternating minimization algorithm (AMA) applied to the following augmented Lagrangian function

$$\mathcal{L}_\eta(\mathbf{u}, \mathbf{v}, \boldsymbol{\lambda}) = \frac{1}{2}\|\mathbf{x} - \mathbf{u}\|_2^2 + \sum_{d=1}^{D} \sum_{l \in \mathcal{E}_d} \left[\gamma w_{d,l} \|\mathbf{v}_{d,l}\|_2 + \langle \boldsymbol{\lambda}_{d,l}, \mathbf{v}_{d,l} - \mathbf{A}_{d,l}\mathbf{u} \rangle + \frac{\eta}{2} \|\mathbf{v}_{d,l} - \mathbf{A}_{d,l}\mathbf{u}\|_2^2 \right].$$

The mode-d difference vector $\mathbf{v}_{d,l}$ is determined by the proximal map

$$\begin{aligned} \mathbf{v}_{d,l} &= \underset{\mathbf{v}_{d,l}}{\arg\min}\, \frac{1}{2}\left[\|\mathbf{v}_{d,l} - \mathbf{A}_{d,l}\mathbf{u} - \eta^{-1}\boldsymbol{\lambda}_{d,l}\|_2^2 + \frac{\gamma w_{d,l}}{\eta}\|\mathbf{v}_{d,l}\|_2\right] \\ &= \operatorname{prox}_{\sigma_{d,l}\|\cdot\|_2}\left(\mathbf{A}_{d,l}\mathbf{u} - \eta^{-1}\boldsymbol{\lambda}_{d,l}\right), \end{aligned} \tag{31}$$

where $\sigma_{d,l} = \gamma w_{d,l}/\eta$. Because the proximal mapping can produce mode-d difference variables that are *exactly* zero, the procedure for computing $\mathbf{v}_{d,l}$ in (31) is immune to the numerical precision issues that hinder the direct computation $\hat{\mathbf{u}} \times_d \boldsymbol{\Delta}_{d,l}$.

Appendix F. Details on Denoising with the Tucker Decomposition for Setting Weights

Employing the Tucker decomposition introduces another tuning parameter, namely the rank of the decomposition. When applicable, a user can leverage problem-specific knowledge to select the rank for the decomposition. Nonetheless, the availability of an automatic approach is desirable to handle cases when such knowledge is unavailable. Selecting the rank in a tensor decomposition, however, is an open question (Kolda and Bader, 2009; Yokota et al., 2017). During initial experiments, a few different methods for selecting the Tucker decomposition rank from the literature were compared: an L-curve approach that attempts to strike a balance between the decomposition's relative error and compression ratio, as implemented by the `mlrankest` function in the `Tensorlab` MATLAB toolbox (Vervliet et al., 2016), minimum description length (Rissanen, 1978; Yokota et al., 2017), and the recently-proposed SCORE algorithm (Yokota et al., 2017). Out of these, the SCORE algorithm produced the best average CoCo estimator performance. The SCORE algorithm itself includes a tuning parameter, $\hat{\rho}$, and Yokota et al. (2017) suggest setting $\hat{\rho} \in [10^{-4}, 10^{-2}]$. We considered $\hat{\rho} \in \{10^{-4}, 10^{-3}, 10^{-2}\}$ and found 10^{-3} to perform the best, which also matches the value used in the experiments by Yokota et al. (2017).

We also developed a simple yet effective heuristic for choosing the rank where we set the Tucker rank for the dth mode to be the floor of $\sqrt{n_d}/2$. Two principles motivating the heuristic are that the rank of the decomposition should be both small relative to and also in proportion to the length of the modes. Both the SCORE algorithm and our heuristic were employed in our simulations described in Section 8 as a robustness check to ensure our CoCo estimator's performance does not crucially depend on the choice of the rank.

The basic Tucker decomposition computation is accomplished by the higher order SVD (HOSVD) method (De Lathauwer et al., 2000) which computes for each mdoe k the r_k leading left singular values of the mode-k matricization and stores them as a factor matrix $\mathbf{U}_k$. The HOSVD then computes the core tensor by contracting the data tensor $\boldsymbol{\mathcal{X}} \times_k \mathbf{U}_k$. Thus, the main cost is computing D SVDs. This is an illustrative calculation, however, and more efficient alternatives exist (Vannieuwenhoven et al., 2012; Minster et al., 2020).

Appendix G. CPD+k-means

We describe in greater detail the CPD+k-means method for co-clustering a D-way tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times \cdots \times n_D}$. The method consists of two steps

Step 1. Compute a rank-R CP decomposition

$$\mathcal{X} \approx \sum_{i=1}^{R} \mathbf{a}_i^{(1)} \circ \mathbf{a}_i^{(2)} \circ \cdots \circ \mathbf{a}_i^{(D)},$$

where $\circ$ represents the outer product and $\mathbf{a}_i^{(d)}$ is the ith column of the dth factor matrix $\mathbf{A}^{(d)} \in \mathbb{R}^{n_d \times R}$.

Step 2. For each factor matrices $\mathbf{A}^{(d)}$, apply k-means clustering on the n_d rows of $\mathbf{A}^{(d)}$. Note that the D applications of k-means are done independently for each mode-d factor matrix $\mathbf{A}^{(d)}$.

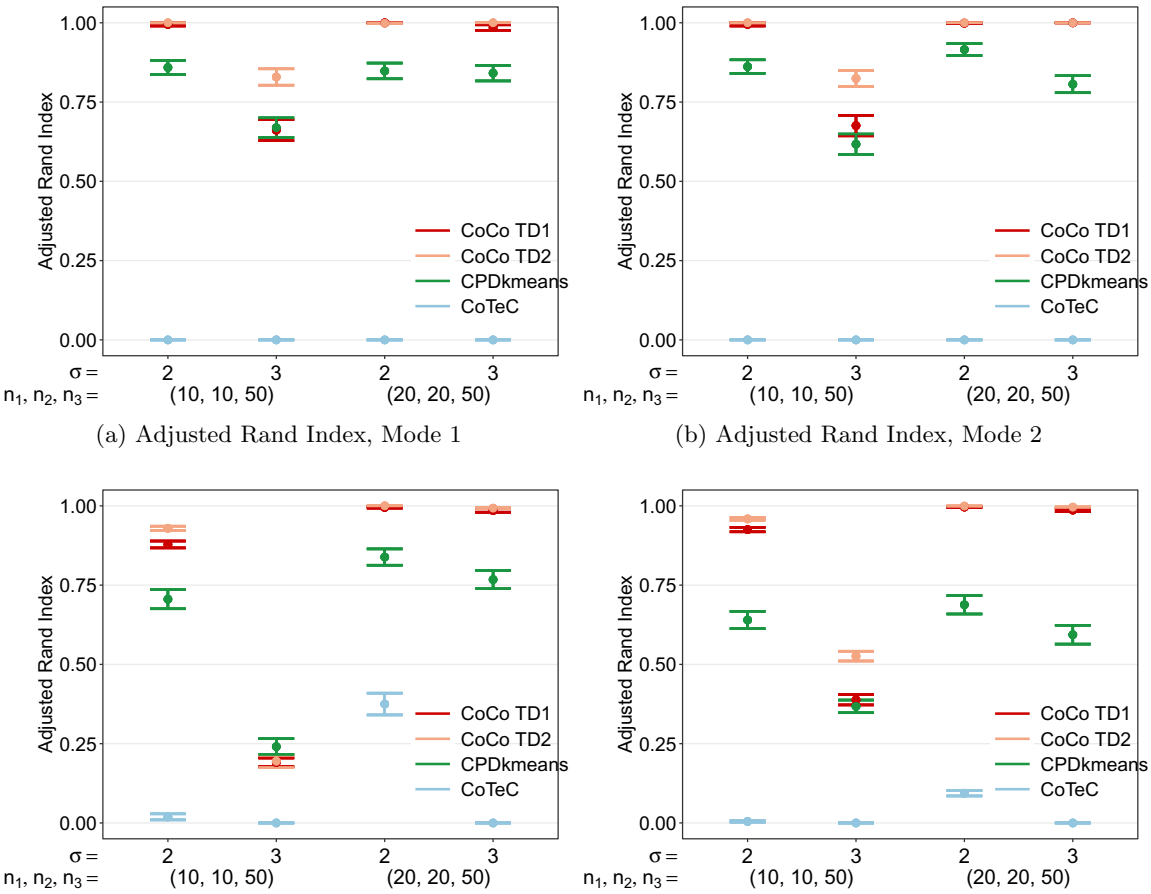
Tuning parameters: There are two sets of tuning parameters: (i) the rank parameter R, used in Step 1, and (ii) the D cluster number parameters for each factor matrix, used in Step 2. To choose the rank parameter R, we create a candidate set of ranks $\mathcal{R}_{\text{candidate}} \subset \{1, 2, 3, \ldots\}$ and select $R^\star \in \mathcal{R}_{\text{candidate}}$ using the tuning procedure in Sun et al. (2017). We then compute a CP decomposition using the selected rank $R^\star$ and obtain the factor matrices $\mathbf{A}^{(d)}$ for $d = 1, \ldots, D$. To choose the D cluster number parameters, we create D candidate sets of cluster numbers $\mathcal{K}^d_{\text{candidate}} \subset \{1, 2, 3, \ldots, n_d\}$ and select $k_d^\star \in \mathcal{K}^d_{\text{candidate}}$ for $d = 1, \ldots, D$ using the gap statistic procedure in Tibshirani et al. (2001). We use the D clustering results from running k-means on the rows of each of the $\mathbf{A}^{(d)}$ using $k_d^\star$.

Appendix H. Additional Simulations on Rectangular Tensors

The first rectangular tensor is one in which there are two short modes ($n_1 = n_2 = 10$) and one relatively longer mode ($n_3 = 50$). Figure 14 presents the clustering results for this tensor shape.

At a lower noise level ($\sigma = 2$), CoCo performs very well and outperforms CPD+k-means and CoTeC in terms of both single-mode clustering and co-clustering. When the noise level is bumped up ($\sigma = 3$), both methods experience a noticeable drop off in their performance and now perform more similarly. Interestingly, CoCo's single-mode clustering results are better along the two shorter modes (modes 1 and 2), which is not what we expected. This provides some evidence that the performance along a mode depends on both the length of that mode as well as the lengths of the other modes. When the length of the shorter modes are increased slightly (from $n_d = 10$ to $n_d = 20$ for $d = 1, 2$), CoCo has near-perfect performance while CPD+k-means performs roughly the same as before. Thus, CoCo struggles with this tensor shape only when the short modes are really short (only 10 observations).

To further investigate the mode-by-mode performance with rectangular tensors, we also apply the clustering methods to a "Goldilocks" tensor with mode lengths that are short, medium, and long. This setting was again motivated by the results from the previous two tensor shapes to see how the performance is impacted when the size of a longer mode is increased. The ARI results for this tensor shape are given in Figure 15d, and they are consistent with what was observed previously. When the short mode has only 10 observations, CoCo initially performs very well until the noise reaches a certain level. At this point, its performance for the longer modes declines sharply and actually performs worse



(a) Adjusted Rand Index, Mode 1

(b) Adjusted Rand Index, Mode 2

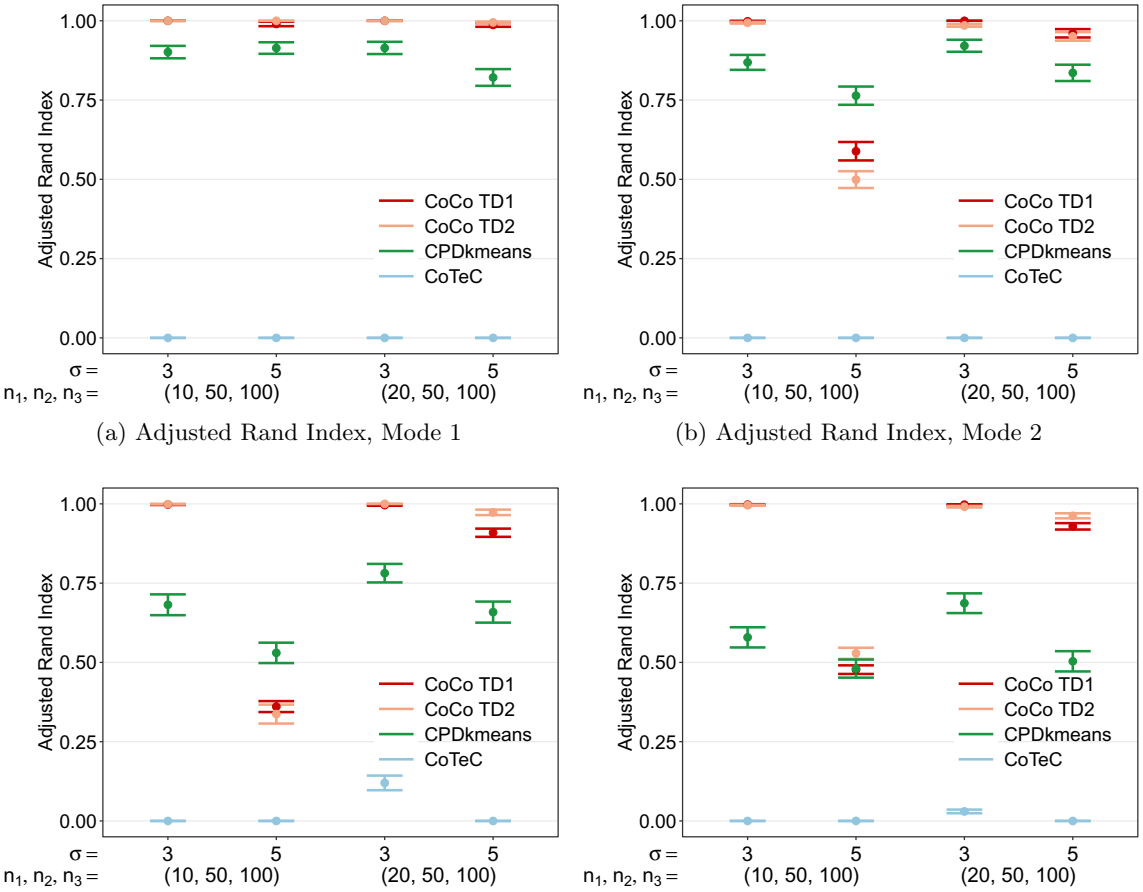
(c) Adjusted Rand Index, Mode 3

(d) Adjusted Rand Index, Co-clusters

Figure 14: Checkerbox Simulation Results: Impact of Tensor Shape. Two balanced clusters per mode with two levels of homoskedastic noise for a tensor with two short modes and one longer mode. Average adjusted rand index plus/minus one standard error for different noise levels and mode lengths.

than CPD+k-means, and this pattern is more pronounced for the longest mode ($n_3 = 100$). The overall co-clustering performance for both methods remains similar, however. As before, CoCo does not experience as much of a decrease when the shortest mode is made slightly longer ($n_1 = 20$), and does noticeably better than CPD+k-means for the most part.

Overall, from clustering these different tensor shapes we see that CoCo still generally performs very well and better than CPD+k-means. The main issue it encounters is when at least one mode is very short ($n_d = 10$). CoCo performs very well a lower noise levels but has a sharp decline in performance once the noise reaches a certain level. Unexpectedly, the decline in single-mode performance is worse for the longer modes. However, even when this happens, CoCo's overall co-clustering performance is still comparable to CPD+k-means. Additionally, this pattern is much less striking when the length of the shortest mode is increased slightly.



(a) Adjusted Rand Index, Mode 1

(b) Adjusted Rand Index, Mode 2

(c) Adjusted Rand Index, Mode 3

(d) Adjusted Rand Index, Co-clusters

Figure 15: Checkerbox Simulation Results: Impact of Tensor Shape. Two balanced clusters per mode with two levels of homoskedastic noise for a tensor with short, medium, and long mode lengths. Average adjusted rand index plus/minus one standard error for different noise levels and mode lengths.

References

Evrim Acar and Bülent Yener. Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering*, 21(1):6–20, 2009.

Evrim Acar, Seyit A. Çamtepe, and Bülent Yener. Collective sampling and analysis of high order tensors for chatroom communications. In *International Conference on Intelligence and Security Informatics*, pages 213–224. Springer, 2006.

Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M. Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832, 2014.

Jerrod I. Ankenman. Geometry and analysis of dual networks on questionnaires, 2014.

Brett W. Bader, Tamara G. Kolda, et al. Matlab tensor toolbox version 2.6. Available online, February 2015. URL `http://www.sandia.gov/~tgkolda/TensorToolbox/`.

Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Sven Bergmann, Jan Ihmels, and Naama Barkai. Iterative signature algorithm for the analysis of large-scale gene expression data. *Physical Review E*, 67(3):031902, 2003.

Anirban Bhar, Martin Haubrock, Anirban Mukhopadhyay, and Edgar Wingender. Multiobjective triclustering of time-series transcriptome data reveals key genes of biological processes. *BMC Bioinformatics*, 16(1):200, 2015.

Xuan Bi, Annie Qu, and Xiaotong Shen. Multilayer tensor factorization with applications to recommender systems. *The Annals of Statistics*, 46(6B):3308–3333, 2018.

Stanislav Busygin, Oleg Prokopyev, and Panos M. Pardalos. Biclustering in data mining. *Computers and Operations Research*, 35(9):2964–2987, 2008.

Xiaochun Cao, Xingxing Wei, Yahong Han, and Dongdai Lin. Robust face clustering via tensor decomposition. *IEEE Transactions on Cybernetics*, 45(11):2546–2557, 2015.

J. Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

Annie I. Chen and Asuman Ozdaglar. A fast distributed proximal-gradient method. In *Communication, Control, and Computing (Allerton), 2012 50th Annual Allerton Conference on*, pages 601–608. IEEE, 2012.

Gary K. Chen, Eric C. Chi, John Michael O. Ranola, and Kenneth Lange. Convex clustering: An attractive alternative to hierarchical clustering. *PLOS Computational Biology*, 11(5): e1004228, 2015.

Jiahua Chen and Zehua Chen. Extended Bayesian information criteria for model selection with large model spaces. *Biometrika*, 95(3):759–771, 2008.

Jiahua Chen and Zehua Chen. Extended BIC for small-n-large-P sparse GLM. *Statistica Sinica*, pages 555–574, 2012.

Eric C. Chi and Kenneth Lange. Splitting methods for convex clustering. *Journal of Computational and Graphical Statistics*, 24(4):994–1013, 2015.

Eric C. Chi and Stefan Steinerberger. Recovering trees with convex clustering. *SIAM Journal on Mathematics of Data Science*, 1(3):383–407, 2019.

Eric C. Chi, Genevera I. Allen, and Richard G. Baraniuk. Convex biclustering. *Biometrics*, 73(1):10–19, 2017.

Andrzej Cichocki, Danilo Mandic, Lieven De Lathauwer, Guoxu Zhou, Qibin Zhao, Cesar Caiafa, and Huy Anh Phan. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Processing Magazine*, 32 (2):145–163, 2015.

Patrick L. Combettes and Jean-Christophe Pesquet. Proximal splitting methods in signal processing. In *Fixed-point Algorithms for Inverse Problems in Science and Engineering*, pages 185–212. Springer, 2011.

Patrick L. Combettes and Valérie R. Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

Narsingh Deo. *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1974.

Jonathan Eckstein. A simplified form of block-iterative operator splitting and an asynchronous algorithm resembling the multi-block alternating direction method of multipliers. *Journal of Optimization Theory and Applications*, 173(1):155–182, Apr 2017.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360, 2001.

Evgeny Frolov and Ivan Oseledets. Tensor methods and recommender systems. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(3):e1201, 2017.

Matan Gavish and Ronald R. Coifman. Sampling, denoising and compression of matrices by coherent matrix organization. *Applied and Computational Harmonic Analysis*, 33(3): 354 – 369, 2012.

Seymour Geisser. The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70(350):320–328, 1975.

Tom Goldstein, Christoph Studer, and Richard Baraniuk. A field guide to forward-backward splitting with a FASTA implementation. *arXiv eprint*, abs/1411.3406, 2014. URL `http://arxiv.org/abs/1411.3406`.

Tom Goldstein, Christoph Studer, and Richard Baraniuk. FASTA: A generalized implementation of forward-backward splitting, January 2015. http://arxiv.org/abs/1501.04979.

Romain Guigourès, Marc Boullé, and Fabrice Rossi. Discovering patterns in time-varying graphs: A triclustering approach. *Advances in Data Analysis and Classification*, pages 1–28, 2015.

David Hallac, Jure Leskovec, and Stephen Boyd. Network lasso: Clustering and optimization in large graphs. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 387–396, New York, NY, USA, 2015. ACM.

D. L. Hanson and F. T. Wright. A bound on tail probabilities for quadratic forms in independent random variables. *The Annals of Mathematical Statistics*, 42:1079–1083, 1971.

Richard A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970.

John A. Hartigan. Direct clustering of a data matrix. *Journal of the American Statistical Association*, 67(337):123–129, 1972.

Nhat Ho, Tianyi Lin, and Michael I. Jordan. On structured filtering-clustering: Global error bound and optimal first-order algorithms. arXiv:1904.07462 [stat.ML], 2019. URL `https://arxiv.org/abs/1904.07462`.

Toby D. Hocking, Armand Joulin, Francis Bach, and Jean-Philippe Vert. Clusterpath an algorithm for clustering using convex fusion penalties. In Lise Getoor and Tobias Scheffer, editors, *28th International Conference on Machine Learning*, page 1. ACM, 2011.

Robert Hof. Study: Mobile Ads Actually Do Work - Especially In Apps. *Forbes, August 27, 2014*, August 2014. Last Accessed July 9, 2017 from `https://www.forbes.com/sites/roberthof/2014/08/27/study-mobile-ads-actually-do-work-especially-in-apps/#27ce654057aa`.

Heng Huang, Chris Ding, Dijun Luo, and Tao Li. Simultaneous tensor subspace selection and clustering: The equivalence of high order SVD and k-means clustering. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 327–335. ACM, 2008.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2 (1):193–218, 1985.

Prateek Jain and Sewoong Oh. Provable tensor factorization with missing data. In *Advances in Neural Information Processing Systems*, pages 1431–1439, 2014.

Stefanie Jegelka, Suvrit Sra, and Arindam Banerjee. Approximation algorithms for tensor clustering. In *International Conference on Algorithmic Learning Theory*, pages 368–383. Springer, 2009.

Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

Tamara G. Kolda and Jimeng Sun. Scalable tensor decompositions for multi-aspect data mining. In *2008 Eighth IEEE International Conference on Data Mining*, pages 363–372, 2008.

Sangeetha Kutty, Richi Nayak, and Yuefeng Li. XML documents clustering using a tensor space model. *Advances in Knowledge Discovery and Data Mining*, pages 488–499, 2011.

Kenneth Lange, David R. Hunter, and Ilsoon Yang. Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, 9(1):1–20, 2000.

Laura Lazzeroni and Art Owen. Plaid models for gene expression data. *Statistica Sinica*, 12:61–86, 2002.

Mihee Lee, Haipeng Shen, Jianhua Z. Huang, and J. S. Marron. Biclustering via sparse singular value decomposition. *Biometrics*, 66(4):1087–1095, 2010.

Mu Li, David G. Andersen, and Alexander Smola. Distributed delayed proximal gradient methods. In *NIPS Workshop on Optimization for Machine Learning*, volume 3, page 3, 2013.

Fredrik Lindsten, Henrik Ohlsson, and Lennart Ljung. Just relax and come clustering! A convexification of k-means clustering. Technical report, Linköpings Universitet, 2011. URL `http://www.control.isy.liu.se/research/reports/2011/2992.pdf`.

Ji Liu, Lei Yuan, and Jieping Ye. Guaranteed sparse recovery under linear transformation. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 91–99. PMLR, 2013a.

Tianqi Liu, Ming Yuan, and Hongyu Zhao. Characterizing spatiotemporal transcriptome of human brain via low rank tensor decomposition. arXiv:1702.07449 [stat.ME], 2017. URL `https://arxiv.org/abs/1702.07449`.

Xinhai Liu, Shuiwang Ji, Wolfgang Glänzel, and Bart De Moor. Multiview partitioning via tensor methods. *IEEE Transactions on Knowledge and Data Engineering*, 25(5): 1056–1069, 2013b.

Ping Ma and Wenxuan Zhong. Penalized clustering of large scale functional data with multiple covariates. *Journal of the American Statistical Association*, 103:625–636, 2008.

Sara C. Madeira and Arlindo L. Oliveira. Biclustering algorithms for biological data analysis: A survey. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 1(1): 24–45, 2004.

Yuliya Marchetti and Qing Zhou. Solution path clustering with adaptive concave penalty. *Electronic Journal of Statistics*, 8(1):1569–1603, 2014.

Caitlin McGarry. Report: Google is the default iPhone search engine because it paid Apple $1 billion. *Macworld, January 22, 2016*, January 2016. Last Accessed July 9, 2017 from `http://www.macworld.com/article/3025783/iphone-ipad/report-google-is-the-default-iphone-search-engine-because-it-paid-apple-1-billion.html`.

Nicolai Meinshausen and Peter Bühlmann. Stability Selection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 72(4):417–473, 2010.

Rachel Minster, Arvind K. Saibaba, and Misha E. Kilmer. Randomized algorithms for low-rank tensor decompositions in the Tucker format. *SIAM Journal on Mathematics of Data Science*, 2(1):189–215, 2020.

Gal Mishne, Ronen Talmon, Ron Meir, Jackie Schiller, Maria Lavzin, Uri Dubin, and Ronald R. Coifman. Hierarchical coupled-geometry analysis for neuronal structure and activity pattern discovery. *IEEE Journal of Selected Topics in Signal Processing*, 10(7): 1238–1253, 2016.

Amy Mitchell, Tom Rosenstiel, Laura Houston Santhanam, and Leah Christian. Future of mobile news. *Project for Excellence in Journalism (PEJ)— Understanding News in the Information Age*, 2012.

Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856, 2002.

Jinoh Oh, Kijung Shin, Evangelos E. Papalexakis, Christos Faloutsos, and Hwanjo Yu. S-HOT: Scalable High-Order Tucker Decomposition. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 761–770. ACM, 2017.

Wei Pan, Xiaotong Shen, and Binghui Liu. Cluster analysis: Unsupervised learning via supervised learning with a non-convex penalty. *Journal of Machine Learning Research*, 14:1865–1889, 2013.

Evangelos E. Papalexakis, Nicholas D. Sidiropoulos, and Rasmus Bro. From K-Means to Higher-Way Co-Clustering: Multilinear Decomposition With Sparse Latent Factors. *IEEE Transactions on Signal Processing*, 61(2):493–506, 2013.

Kristiaan Pelckmans, Jos De Brabanter, Johan A.K. Suykens, and Bart L.R. De Moor. Convex clustering shrinkage. In *PASCAL Workshop on Statistics and Optimization of Clustering Workshop*, 2005.

Peter Radchenko and Gourab Mukherjee. Convex clustering via l1 fusion penalization. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(5):1527–1546, 2017.

Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

Elizabeth D. Schifano, Robert L. Strawderman, and Martin T. Wells. Majorization-minimization algorithms for nonsmoothly penalized objective functions. *Electronic Journal of Statistics*, 4:1258–1299, 2010.

James Sharpnack, Aarti Singh, and Alessandro Rinaldo. Sparsistency of the edge lasso over graphs. In Neil D. Lawrence and Mark Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 1028–1036, 2012.

Yiyuan She. Sparse regression with exact clustering. *Electronic Journal of Statistics*, 4: 1055–1096, 2010.

Xiaotong Shen and Hsin-Cheng Huang. Grouping pursuit through a regularization solution surface. *Journal of the American Statistical Association*, 105(490):727–739, 2010.

Xiaotong Shen, Hsin-Cheng Huang, and Wei Pan. Simultaneous supervised clustering and feature selection over a graph. *Biometrika*, 99:899–914, 2012.

Nicholas D. Sidiropoulos, Lieven De Lathauwer, Xiao Fu, Kejun Huang, Evangelos E. Papalexakis, and Christos Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017.

Martin Sill, Sebastian Kaiser, Axel Benner, and Annette Kopp-Schneider. Robust biclustering by sparse singular value decomposition incorporating stability selection. *Bioinformatics*, 27(15):2089–2097, 2011.

Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, pages 111–147, 1974.

Jimeng Sun, Dacheng Tao, and Christos Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 374–383. ACM, 2006.

Jimeng Sun, Spiros Papadimitriou, Ching-Yung Lin, Nan Cao, Shixia Liu, and Weihong Qian. Multivis: Content-based social network exploration through multi-way visual analysis. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 1064–1075. SIAM, 2009.

Will Wei Sun and Lexin Li. Dynamic tensor clustering. *Journal of the American Statistical Association*, 114(528):1894–1907, 2019.

Will Wei Sun, Junwei Lu, Han Liu, and Guang Cheng. Provable sparse tensor decomposition. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3): 899–916, 2017.

Panagiotis Symeonidis. Matrix and tensor decomposition in recommender systems. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 429–430, New York, NY, USA, 2016. ACM.

Panagiotis Symeonidis and Andreas Zioupos. *Matrix and Tensor Factorization Techniques for Recommender Systems*. Springer International Publishing, 1 edition, 2016.

Kean Ming Tan and Daniela Witten. Statistical properties of convex clustering. *Electronic Journal of Statistics*, 9:2324–2347, 2015.

Kean Ming Tan and Daniela M. Witten. Sparse biclustering of transposable data. *Journal of Computational and Graphical Statistics*, 23(4):985–1008, 2014.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 58(1):267–288, 1996.

Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.

Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

Ryan J. Tibshirani and Jonathan Taylor. The solution path of the generalized lasso. *The Annals of Statistics*, 39(3):1335–1371, 2011.

Paul Tseng. Applications of a Splitting Algorithm to Decomposition in Convex Programming and Variational Inequalities. *SIAM Journal on Control and Optimization*, 29(1): 119–138, 1991.

Ledyard R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

Heather Turner, Trevor Bailey, and Wojtek Krzanowski. Improved biclustering of microarray data demonstrated through systematic performance tests. *Computational Statistics and Data Analysis*, 48(2):235–254, 2005.

Nick. Vannieuwenhoven, Raf. Vandebril, and Karl. Meerbergen. A new truncation strategy for the higher-order singular value decomposition. *SIAM Journal on Scientific Computing*, 34(2):A1027–A1052, 2012.

Nico Vervliet, Otto Debals, Laurent Sorber, Marc Van Barel, and Lieven De Lathauwer. Tensorlab 3.0, Mar. 2016. URL `http://www.tensorlab.net`. Available online.

Van Vu and Ke Wang. Random weighted projections, random quadratic forms and random eigenvectors. *Random Structures and Algorithms Archive*, 47(4):792–821, 2015.

Binhuan Wang, Yilong Zhang, Will Wei Sun, and Yixin Fang. Sparse Convex Clustering. *Journal of Computational and Graphical Statistics*, 27(2):393–403, 2018.

Yuxiang Wang, Huan Xu, and Chenlei Leng. Provable subspace clustering: When LRR meets SSC. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 64–72. Curran Associates, Inc., 2013.

Hadley Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009. ISBN 978-0-387-98140-6. URL `http://ggplot2.org`.

Daniela M. Witten, Robert Tibshirani, and Trevor Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, 2009.

Stephen J. Wright, Robert D. Nowak, and Mário A.T. Figueiredo. Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57(7):2479–2493, July 2009.

Chong Wu, Sunghoon Kwon, Xiaotong Shen, and Wei Pan. A new algorithm and theory for penalized regression-based clustering. *Journal of Machine Learning Research*, 17(188): 1–25, 2016a.

Tao Wu, Austin R. Benson, and David F. Gleich. General tensor spectral co-clustering for higher-order data. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2559–2567. Curran Associates, Inc., 2016b.

Shuo Xiang, Xiaoshen Tong, and Jieping Ye. Efficient sparse group feature selection via nonconvex optimization. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 284–292, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

Or Yair, Ronen Talmon, Ronald R. Coifman, and Ioannis G. Kevrekidis. Reconstruction of normal forms by learning informed observation geometries from data. *Proceedings of the National Academy of Sciences*, 114(38):E7865–E7874, 2017.

Tatsuya Yokota, Namgil Lee, and Andrzej Cichocki. Robust multilinear tensor rank estimation using higher order singular value decomposition and information criteria. *IEEE Transactions on Signal Processing*, 65(5):1196–1206, 2017.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Lihi Zelnik-Manor and Pietro Perona. Self-tuning spectral clustering. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1601–1608. MIT Press, 2005.

Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942, 2010.

Zhong-Yuan Zhang, Tao Li, and Chris Ding. Non-negative tri-factor tensor decomposition with applications. *Knowledge and Information Systems*, 34(2):243–265, 2013.

Hongya Zhao, Debby D. Wang, Long Chen, Xinyu Liu, and Hong Yan. Identifying multi-dimensional co-clusters in tensors based on hyperplane detection in singular vector spaces. *PLOS ONE*, 11(9):1–27, 09 2016.

Xiaolin Zheng, Weifeng Ding, Zhen Lin, and Chaochao Chen. Topic tensor factorization for recommender system. *Information Sciences*, 372(Supplement C):276 – 293, 2016.

Hua Zhou, Lexin Li, and Hongtu Zhu. Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association*, 108:540–552, 2013.

Changbo Zhu, Huan Xu, Chenlei Leng, and Shuicheng Yan. Convex optimization procedure for clustering: Theoretical revisit. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1619–1627. Curran Associates, Inc., 2014.

Yunzhang Zhu, Xiaotong Shen, and Wei Pan. Simultaneous grouping pursuit and feature selection over an undirected graph. *Journal of the American Statistical Association*, 108 (502):713–725, 2013.

Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.

Hui Zou and Runze Li. One-step sparse estimates in nonconcave penalized likelihood models. *The Annals of Statistics*, 36(4):1509–1533, 2008.