From K-Means to Higher-Way Co-Clustering: Multilinear Decomposition With Sparse Latent Factors

Evangelos E. Papalexakis, *Student Member, IEEE*, Nicholas D. Sidiropoulos, *Fellow, IEEE*, and Rasmus Bro

***Abstract*—Co-clustering is a generalization of unsupervised clustering that has recently drawn renewed attention, driven by emerging data mining applications in diverse areas. Whereas clustering groups entire columns of a data matrix, co-clustering groups columns over select rows only, i.e., it simultaneously groups rows and columns. The concept generalizes to data "boxes" and higher-way tensors, for simultaneous grouping along multiple modes. Various co-clustering formulations have been proposed, but no workhorse analogous to K-means has emerged. This paper starts from K-means and shows how co-clustering can be formulated as a constrained multilinear decomposition with sparse latent factors. For three- and higher-way data, uniqueness of the multilinear decomposition implies that, unlike matrix co-clustering, it is possible to unravel a large number of possibly overlapping co-clusters. A basic multi-way co-clustering algorithm is proposed that exploits multilinearity using Lasso-type coordinate updates. Various line search schemes are then introduced to speed up convergence, and suitable modifications are proposed to deal with missing values. The imposition of latent sparsity pays a collateral dividend: it turns out that sequentially extracting one co-cluster at a time is almost optimal, hence the approach scales well for large datasets. The resulting algorithms are benchmarked against the state-of-art in pertinent simulations, and applied to measured data, including the ENRON e-mail corpus.**

***Index Terms*—Co-clustering, compressed sensing, factor analysis, k-means, multi-way analysis, sparsity, tensor decomposition, triclustering, unsupervised clustering.**

Manuscript received November 18, 2011; revised July 16, 2012 and September 18, 2012; accepted September 19, 2012. Date of publication October 16, 2012; date of current version December 31, 2012. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Raviv Raich. This work was supported in part by ARL/ERO W911NF-10-1-0464, W911NF-11-1-0500, TU Crete, the European Union (European Social Fund—ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF)—Research Funding Program: Thales. This work was presented in part at the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Prague, Czech Republic, May22–27, 2011.

E. E. Papalexakis was with the Electrical and Computer Engineering Department at the Technical University of Crete, Chania, Greece. He is now with the Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA 15213 USA (e-mail: epapalex@cs.cmu.edu; website: http://www.cs.cmu.edu/~epapalex/).

N. D. Sidiropoulos was with the Electrical and Computer Engineering Department, Technical University of Crete, Chania, Greece. He is now with the Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: nikos@ece.umn.edu; website: http://www.ece.umn.edu/~nikos).

R. Bro is with the Department of Food Science, University of Copenhagen, 1958 Frederiksberg, Denmark (e-mail: rb@life.ku.dk; website: http://www.models.life.ku.dk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2012.2225052

I. Introduction

UNSUPERVISED clustering seeks to group the columns of a data matrix so that columns belonging to the same group are close to each other in some sense. The prominent example is K-means, which seeks to partition the columns in a way that columns falling in the same subset are close in terms of Euclidean distance. Unsupervised clustering is a core toolbox in pattern recognition and machine intelligence, where numerous extensions and variations of K-means have been developed over the years to account for overlapping groups, non-Euclidean distances and non-metric clustering, to name a few.

A limitation of clustering is that it groups *whole vectors*, i.e., if two vectors differ significantly even in one element, they cannot be clustered together. There are nowadays more and more applications wherein one is interested in detecting, e.g., groups of customers buying certain products, *even though their overall buying patterns are very different*, or gene co-expression under certain experimental conditions that are a priori unknown. These tasks cannot be accomplished with classical clustering methods.

Co-clustering is a generalization of unsupervised clustering that seeks to group columns over selected rows only (and vice-versa); that is, it simultaneously groups the rows and columns of a matrix to produce 'coherent' groups called co-clusters. The notion of coherence can range from constant to 'similar' co-cluster values, or proportional expression profiles in the case of gene expression data. This type of matrix co-clustering is also referred to as bi-clustering. Co-clustering goes back to the early '70's [14], but it has recently found applications in diverse areas, ranging from the analysis of gene co-expression to network traffic and social network analysis [1], [5], [7], [18], [23]. The concept readily generalizes to higher-way data sets (e.g., adding a temporal dimension). There are few papers dealing with three-way co-clustering (tri-clustering) [28], [36], [37] and no systematic study of three- and higher-way co-clustering, to the best of our knowledge. This is important because the algebraic properties of three- and higher-way data are very different from those of two-way (matrix) data; see, for example [19], [31].

K-means is NP-hard, but the (generalized) Lloyd-Max iteration usually yields acceptable solutions at affordable complexity. Hard co-clustering that partitions both rows and columns is a generalization of K-means, hence also NP-hard. Unlike K-means, there is unfortunately no algorithmic workhorse analogous to the Lloyd-Max iteration for hard co-clustering. Various hard and soft bi-clustering formulations

have been proposed in the literature [1], [5], [7], [18], [23], but numerical optimization remains challenging in most cases.

Starting from basic K-means and its extensions, we show how co-clustering can be formulated as a constrained multilinear decomposition with sparse latent factors. In the case of three- and higher-way data, this corresponds to a parallel factor (PARAFAC) decomposition with sparse latent factors. This has important implications for co-clustering, because PARAFAC is unique under relatively mild conditions. This allows to uniquely unravel a large number of possibly overlapping co-clusters that are hidden in the data – something impossible with matrix methods. We discuss modeling alternatives, paying particular attention to cases where one expects different co-cluster support along the different modes of the data (e.g., a typical co-cluster involves more rows than columns). We then propose a basic multi-way co-clustering algorithm that exploits multilinearity using Lasso-type coordinate updates. Each update is very simple, but the flip-side is that the number of iterations until convergence can be large. To speed up convergence, we propose line search schemes based on iterative majorization and polynomial fitting. We also show how one can modify our algorithms to deal with missing data – a situation that is common in many applications. The resulting algorithms are compared to the state-of-art in carefully designed simulations, and also applied to measured data – the ENRON e-mail corpus, Amazon co-purchase data, and chromatographic wine data – to illustrate the benefit of line search.

Interestingly, the imposition of latent sparsity pays a collateral dividend: as one increases the number of fitted co-clusters, new co-clusters are added without affecting those previously extracted. This is not normally true for PARAFAC without latent sparsity. An important corollary of this 'additivity' is that the co-clusters can be equivalently recovered one by one, in deflation mode. This is important because fitting a rank-one component is far easier computationally, implying that the approach remains operational even for large datasets.

A. Relevant Prior Art

References [20], [35] have considered bilinear matrix decompositions with sparse latent factors, [20] specifically for bi-clustering – this is the closest piece of work to ours. We will explain the differences with [20] in Section III-B1. We have previously considered a related doubly-sparse model in different contexts [25], [30]. None of the above has considered tri-clustering and higher-way co-clustering of tensors. The first attempt at tri-clustering was [36], followed by [28], [37]. These did not consider latent sparsity, which is at the heart of our approach for joint co-cluster support selection across all modes. An early version of part of our work appeared in [26], see also the introductory article [6] for an interesting application in food technology. The model is now different [(5) vs. (2) in what follows], and the algorithms have evolved significantly, to include enhancements such as line search and missing values.

Notation and Preliminaries

A scalar is denoted by an italic letter, e.g., a. A column vector is denoted by a bold lowercase letter, e.g., $\mathbf{a}$ whose i-th entry is $\mathbf{a}(i)$. A matrix is denoted by a bold uppercase letter, e.g., $\mathbf{A}$ with (i,j)-th entry $\mathbf{A}(i,j)$; $\mathbf{A}(:,j)$ ($\mathbf{A}(i,:)$) denotes the j-th column (resp. i-th row) of $\mathbf{A}$. A three-way array is denoted by an underlined bold uppercase letter, e.g., $\underline{\mathbf{A}}$, with (i,j,n)-th entry $\underline{\mathbf{A}}(i,j,n)$. Vector, matrix and three-way array size parameters (mode lengths) are denoted by uppercase letters, e.g., I. $I_1 : I_2$ denotes the range from I_1 to I_2 inclusive.

$\circ$ stands for the vector outer product; i.e., for two vectors $\mathbf{a}$ ($I \times 1$) and $\mathbf{b}$ ($J \times 1$), $\mathbf{a} \circ \mathbf{b}$ is an $I \times J$ rank-one matrix with (i,j)-th element $\mathbf{a}(i)\mathbf{b}(j)$; i.e., $\mathbf{a} \circ \mathbf{b} = \mathbf{a}\mathbf{b}^T$. For three vectors, $\mathbf{a}$ ($I \times 1$), $\mathbf{b}$ ($J \times 1$), $\mathbf{c}$ ($N \times 1$), $\mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$ is an $I \times J \times N$ rank -one three-way array with (i,j,n)-th element $\mathbf{a}(i)\mathbf{b}(j)\mathbf{c}(n)$. Notice that outer products are *simple* (rank-one) structures, in the sense that all columns are proportional to a single 'pattern', and the same holds for rows and 'fibers'. The rank of a matrix $\mathbf{A}$ can be defined as the smallest number of outer products needed to synthesize $\mathbf{A}$. The rank of a three-way array $\underline{\mathbf{A}}$ is likewise defined as the smallest number of outer products needed to synthesize $\underline{\mathbf{A}}$. It turns out that this is the proper way to generalize the concept of matrix rank to three- and higher-way arrays (e.g., column and row ranks generally do not coincide in the three-way case).

$\otimes$ stands for the Kronecker product; given $\mathbf{A}$ ($I \times K_1$) and $\mathbf{B}$ ($J \times K_2$), $\mathbf{A} \otimes \mathbf{B}$ is the $JI \times K_2K_1$ matrix

$$\mathbf{A} \otimes \mathbf{B} := \begin{bmatrix} \mathbf{B}\mathbf{A}(1,1) & \cdots & \mathbf{B}\mathbf{A}(1,K_1) \\ \vdots & \ddots & \vdots \\ \mathbf{B}\mathbf{A}(I,1) & \cdots & \mathbf{B}\mathbf{A}(I,K_1) \end{bmatrix}$$

$\odot$ stands for the Khatri-Rao (column-wise Kronecker) product; given $\mathbf{A}$ ($I \times K$) and $\mathbf{B}$ ($J \times K$) (notice same number of columns), $\mathbf{A} \odot \mathbf{B}$ is the $JI \times K$ matrix

$$\mathbf{A} \odot \mathbf{B} = [\, \mathbf{A}(:,1) \otimes \mathbf{B}(:,1) \cdots \mathbf{A}(:,K) \otimes \mathbf{B}(:,K) \,]$$

$\circledast$ stands for the Hadamard (element-wise) product.

Consider a three-way array $\underline{\mathbf{X}}$. A low-rank approximation of $\underline{\mathbf{X}}$ can be written as

$$\underline{\mathbf{X}} \cong \sum_{k=1}^{K} \mathbf{a}_k \circ \mathbf{b}_k \circ \mathbf{c}_k. \tag{1}$$

When the approximation is exact, the above is known as the PARAFAC decomposition [13]. The rank of $\underline{\mathbf{X}}$ is then $\leq K$. Any three-way array $\underline{\mathbf{X}}$ can be decomposed/synthesized as above, for sufficiently high K. Let $\mathbf{A} := [\mathbf{a}_1, \ldots, \mathbf{a}_K]$ ($I \times K$), $\mathbf{B} := [\mathbf{b}_1, \ldots, \mathbf{b}_K]$ ($J \times K$), and $\mathbf{C} := [\mathbf{c}_1, \ldots, \mathbf{c}_K]$ ($N \times K$). Then the $I \times J \times N$ three-way array $\underline{\mathbf{X}}$ above can be unfolded into matrix form in three useful ways: $\mathbf{X}_{(1)}$ of size $NJ \times I$, $\mathbf{X}_{(2)}$ of size $IN \times J$, and $\mathbf{X}_{(3)}$ of size $JI \times N$. These unfoldings can be specified as follows. Let $\underline{\mathbf{X}}(i,:,:)$ be the i-th $J \times N$ 'slab' of $\underline{\mathbf{X}}$ perpendicular to the I-mode, $\underline{\mathbf{X}}(:,j,:)$ the j-th $I \times N$ slab of $\underline{\mathbf{X}}$ perpendicular to the J-mode, and $\underline{\mathbf{X}}(:,:,n)$ the n-th $I \times J$ slab of $\underline{\mathbf{X}}$ perpendicular to the N-mode. Then

$$\mathbf{X}_{(1)} = \begin{bmatrix} \underline{\mathbf{X}}(:,1,:)^T \\ \vdots \\ \underline{\mathbf{X}}(:,J,:)^T \end{bmatrix}, \mathbf{X}_{(2)} = \begin{bmatrix} \underline{\mathbf{X}}(:,:,1) \\ \vdots \\ \underline{\mathbf{X}}(:,:,N) \end{bmatrix}, \mathbf{X}_{(3)} = \begin{bmatrix} \underline{\mathbf{X}}(1,:,:) \\ \vdots \\ \underline{\mathbf{X}}(I,:,:) \end{bmatrix},$$

and it can be shown that

$$\mathbf{X}_{(1)} = (\mathbf{B} \odot \mathbf{C})\, \mathbf{A}^T, \mathbf{X}_{(2)} = (\mathbf{C} \odot \mathbf{A})\, \mathbf{B}^T, \mathbf{X}_{(3)} = (\mathbf{A} \odot \mathbf{B})\, \mathbf{C}^T.$$

The operator $[\cdot]_{[\alpha,\beta]}$ projects its argument onto the interval $[\alpha,\beta]$; $\mathbb{I}_{\{\text{condition}\}}$ denotes an indicator function – equal to 1 if the condition is true, 0 otherwise.

II. Motivation and Problem Formulation

A. Clustering as Constrained Outer Product Decomposition

Consider the familiar problem of clustering a set of vectors $\{\mathbf{x}_j \in \mathbb{R}^I\}_{j=1}^J$ in K clusters. The goal is to find $K \ll J$ cluster means $\{\boldsymbol{\mu}_k \in \mathbb{R}^I\}_{k=1}^K$ and an assignment of each $\mathbf{x}_j$ to a best-matching cluster $k^*(j)$ such that $\sum_j |\mathbf{x}_j - \boldsymbol{\mu}_{k^*(j)}|^2$ (or other suitable mismatch cost) is minimized. In matrix algebra terms, the problem can be posed as follows. Define $\mathbf{X} := [\mathbf{x}_1, \ldots, \mathbf{x}_J]$ $(I \times J)$, $\mathbf{M} := [\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K]$ $(I \times K)$, and an assignment matrix $\mathbf{B} := [\mathbf{b}_1, \ldots, \mathbf{b}_K]$ $(J \times K)$ having binary elements $\mathbf{B}(j,k) = \mathbf{b}_k(j) \in \{0,1\}$ and rows satisfying $\sum_{k=1}^K \mathbf{B}(j,k) = 1$, $\forall j$ (i.e., each row sums to 1). The most widely used version of the clustering problem, known as *K-means* clustering, can then be written as

$$\min_{\mathbf{M},\mathbf{B} \in \{0,1\}^{J \times K} \bigcap \mathcal{RS}} \|\mathbf{X} - \mathbf{M}\mathbf{B}^T\|_F^2,$$

where $\mathcal{RS}$ denotes the set of matrices with the property that each row sums to 1. K-means clustering is NP-hard; for this reason, iterative algorithms based on the Lloyd-Max iteration are typically used to compute suboptimal solutions, often with good results.

Note that K-means clustering is equivalent to finding a best-fitting approximation (in the least squares sense) of the matrix $\mathbf{X}$ as a sum of K outer products

$$\min \left|\left|\mathbf{X} - \left(\boldsymbol{\mu}_1 \mathbf{b}_1^T + \cdots + \boldsymbol{\mu}_K \mathbf{b}_K^T\right)\right|\right|_F^2,$$

but the loadings in one mode are constrained: $\mathbf{B} \in \{0,1\}^{J \times K} \bigcap \mathcal{RS}$. The binary constraint $\mathbf{B}(j,k) = \mathbf{b}_k(j) \in \{0,1\}, \forall j,k$ corresponds to the usual case of 'hard' clustering: every data vector either belongs to a certain cluster or not. The $\mathcal{RS}$ constraint $\sum_{k=1}^K \mathbf{B}(j,k) = 1$, $\forall j$ ensures that every data vector belongs to one and only one cluster – no data vector is left 'orphan' and the clusters are non-overlapping. Relaxing the binary 0–1 constraints to non-negativity while maintaining the $\mathcal{RS}$ constraint corresponds to 'soft' clustering (overlapping clusters); the magnitude of $\mathbf{B}(j,k)$ now indicates how well $\mathbf{x}_j$ fits in cluster k. Replacing the $\mathcal{RS}$ constraint by its 'lossy' counterpart $\sum_{k=1}^K \mathbf{B}(j,k) \leq 1$ (or even dropping it altogether) emphasizes the extraction of significant clusters at the expense of not modeling 'outlying' data points. This is often well-justified in the context of exploratory data analysis. From this point on, we mostly focus on soft lossy (co-) clustering. We also assume non-negative data and impose non-negativity on the latent factors. Non-negativity is valid in many but not all applications of co-clustering, so we also comment on how our approach can be modified for the real-valued case.

B. Co-Clustering as Constrained Outer Product Decomposition

K-means and related approaches cluster whole vectors – meaning that all elements of a given vector are considered when making clustering decisions, and vectors that are clustered together are ideally 'close' in each and every coordinate. A single cluster is modeled as a rank-one outer product plus noise:

$$\mathbf{C} = \boldsymbol{\mu}\mathbf{b}^T + \text{noise},$$

where $\boldsymbol{\mu}$ is unconstrained and $\mathbf{b}$ is binary; i.e., $\mathbf{b}(j) \in \{0,1\}$, with 1's in those elements corresponding to columns of the data matrix that belong to the given cluster. The vector $\mathbf{b}$ will typically be sparse, because most data columns *will not* belong to any given cluster – at least when $K > 2$ and the cluster populations are roughly balanced.

There are many applications where certain vectors are close only for a certain subset of their elements, and we need to spot this pattern. A good example is gene expression data, where the rows of the data matrix $\mathbf{X}$ correspond to specific genes being expressed, the columns to experimental conditions, and the objective is to detect patterns of joint gene expression and the conditions under which this happens. Note that we do not know *a priori* exactly which genes are expressed together, or under which conditions. Another example is marketing, where rows correspond to products, columns to customers, and the objective is *not* to cluster the products or the customers, but rather to detect (possibly overlapping) groups of customers that tend to buy certain subsets of products. This is the *co-clustering* (in this case *bi-clustering*) problem which has recently generated significant interest in numerous disciplines [1], [5], [7], [18], [23]. In social network analysis, co-clustering can be used to detect social groups (often called 'cliques') engaging in certain types of social behavior.

Whereas one-sided clustering involves selection (which columns belong to the given cluster) in one mode, co-clustering involves selection in both modes (rows and columns). This can be modeled as

$$\mathbf{G} = \mathbf{a}\mathbf{b}^T + \text{noise},$$

where $\mathbf{a}$ and $\mathbf{b}$ are both sparse. When only relative expression matters, we can relax the binary constraint on the elements of $\mathbf{a}$ and $\mathbf{b}$, possibly retaining non-negativity when appropriate. Assuming non-negative data $\mathbf{X}(i,j) \geq 0$, $\forall i,j$, and focusing on overlapping (soft) lossy co-clustering, the problem can then be formulated as

$$\min_{\mathbf{A} \geq \mathbf{0}, \mathbf{B} \geq \mathbf{0}} \|\mathbf{X} - \mathbf{A}\mathbf{B}^T\|_F^2,$$

where the inequalities should be interpreted element-wise, and the columns of $\mathbf{A}$ $(I \times K)$ and $\mathbf{B}$ $(J \times K)$ should typically contain many zeros.

One may envision using singular value decomposition (SVD) or non-negative matrix factorization (NMF) [21] for co-clustering; however, the columns of $\mathbf{A}$ and $\mathbf{B}$ will then be dense, destroying all support information which is crucial in co-clustering applications. SVD imposes orthogonality, which is artificial and limits analysis to non-overlapping co-clusters if non-negativity is also imposed.

Enforcing sparsity is ideally accomplished by penalizing the number of non-zero elements (the ℓ_0 norm), however this yields an intractable optimization problem. Recent research has shown that a practical alternative is to use an ℓ_1 penalty in lieu of the ℓ_0 norm [32]. This leads to the following formulation of bi-clustering:

$$\min_{\mathbf{A} \geq \mathbf{0}, \mathbf{B} \geq \mathbf{0}} \|\mathbf{X} - \mathbf{A}\mathbf{B}^T\|_F^2 + \lambda_a \sum_{i,k} |\mathbf{A}(i,k)| + \lambda_b \sum_{j,k} |\mathbf{B}(j,k)|, \tag{2}$$

where different 'prices' λ_a, λ_b have been introduced to account for the fact that co-clusters may involve more (or less) rows

than columns; e.g., 3 genes and 12 experimental conditions for gene co-expression data. This means that the level of latent sparsity (number or percentage of nonzero elements) differs across modes, which in turn implies that imbalanced sparsity penalties should be employed to reveal the underlying structure.

With $\mathbf{A} := [\mathbf{a}_1, \ldots, \mathbf{a}_K]$, $\mathbf{B} := [\mathbf{b}_1, \ldots, \mathbf{b}_K]$, (2) can be written as

$$\min_{\{\mathbf{a}_k \geq 0, \mathbf{b}_k \geq 0\}_{k=1}^K} \left\| \mathbf{X} - \sum_{k=1}^K \mathbf{a}_k \mathbf{b}_k^T \right\|_F^2 + \lambda_a \sum_k \|\mathbf{a}_k\|_1 + \lambda_b \sum_k \|\mathbf{b}_k\|_1. \quad (3)$$

The formulation in (2)/(3) has the following *(weighted) norm-balancing* property:

Claim 1: Let $\left\{\widehat{\mathbf{a}}_k, \widehat{\mathbf{b}}_k\right\}_{k=1}^K$ be a solution of (3) for $\lambda_a > 0$, $\lambda_b > 0$. Then

$$\lambda_a \|\widehat{\mathbf{a}}_k\|_1 = \lambda_b \|\widehat{\mathbf{b}}_k\|_1 = \sqrt{\lambda_a \lambda_b \|\mathbf{a}_k\|_1 \|\mathbf{b}_k\|_1}, \quad \forall k. \quad (4)$$

Proof: It is easy to see that, for $\lambda_a > 0$, $\lambda_b > 0$, either both $\widehat{\mathbf{a}}_k$, $\widehat{\mathbf{b}}_k$ are zero, in which case the equality holds trivially, or both are nonzero. In the latter case, note that the ℓ_2 part of the cost in (3) is invariant to simultaneous scaling of $\mathbf{a}_k$ by s and counter-scaling of $\mathbf{b}_k$ by $\frac{1}{s}$, whereas the ℓ_1 part $\lambda_a \|\mathbf{a}_k\|_1 + \lambda_b \|\mathbf{b}_k\|_1$ is sensitive with respect to such scaling. Consider $s\alpha + \frac{1}{s}\beta$, for some $\alpha > 0$, $\beta > 0$. Taking derivative with respect to s and setting it to zero yields $\widehat{s} = \pm\sqrt{\frac{\beta}{\alpha}}$. Substituting $\widehat{s} = \sqrt{\frac{\beta}{\alpha}}$ back yields[1] $\widehat{s}\alpha + \frac{1}{\widehat{s}}\beta = \sqrt{\alpha\beta} + \sqrt{\alpha\beta}$, i.e., the two terms are equalized. From the arithmetic mean – geometric mean inequality, we know that $2\sqrt{\alpha\beta} \leq \alpha + \beta$, with equality if and only if $\alpha = \beta$. Thus, if $\lambda_a \|\widehat{\mathbf{a}}_k\|_1 \neq \lambda_b \|\widehat{\mathbf{b}}_k\|_1$, scaling $\widehat{\mathbf{a}}_k$ by s and counter-scaling $\widehat{\mathbf{b}}_k$ by $\frac{1}{s}$ can be used to strictly reduce the cost, thereby contradicting optimality. ■

The norm-balancing property yields an important and perhaps unexpected corollary:

Claim 2: For $\lambda_a > 0$, $\lambda_b > 0$, Problem (3) is equivalent to

$$\min_{\{\mathbf{a}_k \geq 0, \mathbf{b}_k \geq 0\}_{k=1}^K} \left\| \mathbf{X} - \sum_{k=1}^K \mathbf{a}_k \mathbf{b}_k^T \right\|_F^2 + \sum_k 2\sqrt{\lambda_a \lambda_b \|\mathbf{a}_k\|_1 \|\mathbf{b}_k\|_1},$$

implying that it is impossible to individually control the row- and column-sparsity in (3) through appropriate choice of λ_a and λ_b, respectively; only the product $\lambda_a \lambda_b$ matters, and the effective penalty is always symmetric.

Proof:

$$\min_{\{\mathbf{a}_k \geq 0, \mathbf{b}_k \geq 0\}_{k=1}^K} \left\| \mathbf{X} - \sum_{k=1}^K \mathbf{a}_k \mathbf{b}_k^T \right\|_F^2 + \lambda_a \sum_k \|\mathbf{a}_k\|_1 + \lambda_b \sum_k \|\mathbf{b}_k\|_1$$

$$\Longleftrightarrow \min_{\{\mathbf{a}_k \geq 0, \mathbf{b}_k \geq 0, s_k > 0\}_{k=1}^K} \left\| \mathbf{X} - \sum_{k=1}^K \mathbf{a}_k s_k \frac{1}{s_k} \mathbf{b}_k^T \right\|_F^2 + \lambda_a \sum_k s_k \|\mathbf{a}_k\|_1 + \lambda_b \sum_k \frac{1}{s_k} \|\mathbf{b}_k\|_1$$

$$\Longleftrightarrow \min_{\{\mathbf{a}_k \geq 0, \mathbf{b}_k \geq 0\}_{k=1}^K} \left\| \mathbf{X} - \sum_{k=1}^K \mathbf{a}_k \mathbf{b}_k^T \right\|_F^2 + \sum_k 2\sqrt{\lambda_a \lambda_b \|\mathbf{a}_k\|_1 \|\mathbf{b}_k\|_1}$$

where for the last equivalence we used that we may restrict the search for minima over only those $(\mathbf{a}_k, \mathbf{b}_k)$ that are either both zero or both nonzero – having only one zero vector cannot be optimal, as explained earlier. ■

This 'forced symmetry' is the most important, but not the only drawback of the formulation in (2)/(3), which also exhibits *scaling bias*. Noting that $\|\mathbf{a}_k\|_1 \|\mathbf{b}_k\|_1 = |\mathbf{a}_k \mathbf{b}_k^T|$, the latter being the 'size' of co-cluster k as measured by the sum of absolute values of its elements, it follows that (2)/(3) penalizes co-cluster support times expression level according to $\sqrt{\text{support} \times \text{level}}$. Finally, from a computational standpoint, we have observed that alternating optimization – type algorithms aiming to solve (2)/(3) tend to get stuck in 'scaling swamps', during which there is very slow progress towards balancing the norms and virtually no progress in terms of the overall cost function in (2)/(3).

The above shortcomings are undesirable artifacts of using the ℓ_1 norm as a surrogate of the ℓ_0 norm. Whilst this substitution has proven merits in the context of variable selection in linear regression, we have argued that it is not as well-motivated in the case of bilinear (and higher-order) regression with latent sparsity. In addition to proven oracle properties, a fundamental reason that ℓ_1 is used in lieu of ℓ_0 for linear problems is that it yields a convex optimization problem after the substitution, which can be efficiently solved – scalar updates can be carried out in close-form, and coordinate descent can be used to find the global optimum. This advantage disappears for bilinear and higher-order models, because even after $\ell_0 \rightarrow \ell_1$ substitution the problem remains highly non-convex. As a result, alternative approximations of the ℓ_0 norm should be considered.

To circumvent these difficulties, we propose modifying the model and cost function as follows. Any $\mathbf{a}_k \geq 0$ can be written as $\sigma_k \widetilde{\mathbf{a}}_k$, with $0 \leq \widetilde{\mathbf{a}}_k(i) \leq 1$, and $\sigma_k := \max_i \mathbf{a}_k(i)$; and likewise for $\mathbf{b}_k$. If we are interested in inducing sparsity on $\mathbf{a}_k \geq 0$ while simultaneously retaining scaling freedom, it makes sense to penalize $\|\widetilde{\mathbf{a}}_k\|_1$ instead of $\|\mathbf{a}_k\|_1$. As illustrated in Fig. 1, this is a better approximation of the ℓ_0 norm. Dropping the tilde for brevity, we therefore obtain

$$\min_{\{0 \leq \rho_k \leq \bar{\rho}, 0 \leq \mathbf{a}_k \leq 1, 0 \leq \mathbf{b}_k \leq 1\}_{k=1}^K} \left\| \mathbf{X} - \sum_{k=1}^K \rho_k \mathbf{a}_k \mathbf{b}_k^T \right\|_F^2 + \lambda_a \sum_k \|\mathbf{a}_k\|_1 + \lambda_b \sum_k \|\mathbf{b}_k\|_1, \quad (5)$$

where $\bar{\rho} := \max_{i,j} \mathbf{X}(i,j)$. Notice that $\rho_k \leq \bar{\rho}$ can be enforced without loss of generality; an upper bound is needed to avoid arbitrarily scaling down $\{\mathbf{a}_k, \mathbf{b}_k\}$ while absorbing the overall scale in ρ_k. Also note that, when the elements of $\mathbf{a}_k$ are constrained in $[0, 1]$ as in (5), $\|\mathbf{a}_k\|_1 \leq \|\mathbf{a}_k\|_0$, with $\|\mathbf{a}_k\|_1 = \|\mathbf{a}_k\|_0$ (and likewise for $\mathbf{b}_k$) in the important special case of a constant co-cluster of level $\bar{\rho}$.

Note that one might be tempted to tighten the interval for the elements of $\mathbf{a}, \mathbf{b}$ to $[0, \epsilon]$ for some small positive $\epsilon < 1$, and compensate by changing the upper bound on ρ from $\bar{\rho}$ to $\frac{\bar{\rho}}{\epsilon}$, in order to employ an even better approximation of the zero norm, see

[1] Use of the positive square root preserves the sign of α, β. When $\alpha = \beta = 0$ any $s > 0$ is equally good and the result is 0.

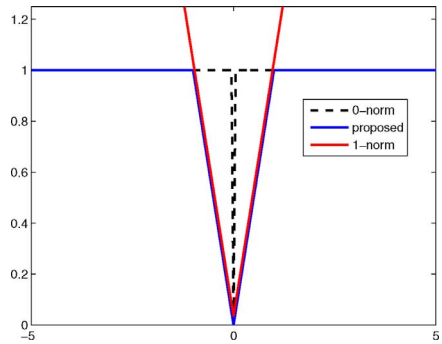


Fig. 1. Two approximations of the zero norm: note that the proposed one is much tighter for large $|x|$, due to saturation.

also Fig. 1. Although conditional updates of one variable given the rest will remain simple, the overall cost function will be harder to minimize (non-convexity will be more severe) in this case. Also, simultaneously working with very small and very large variables can give rise to numerical conditioning problems for small ϵ. Hence we advocate using $\epsilon = 1$ as a good trade-off.

In some applications of bi-clustering, $\mathbf{X}$ and the latent factors have real-valued elements; adjusting for this, the problem becomes

$$\min_{\{|\rho_k|\leq\check{\rho},-1\leq\mathbf{a}_k,\mathbf{b}_k\leq 1\}_{k=1}^{K}} \left\|\mathbf{X}-\sum_{k=1}^{K}\rho_k\mathbf{a}_k\mathbf{b}_k^T\right\|_F^2 + \lambda_a\sum_k\|\mathbf{a}_k\|_1+\lambda_b\sum_k\|\mathbf{b}_k\|_1, \quad (6)$$

where now $\check{\rho} := \max_{i,j}|\mathbf{X}(i,j)|$.

1) Related Work: We have previously considered a special case (with $\lambda_a = \lambda_b$) of the doubly sparse bilinear regression problem in (2) in different contexts [25], [30]. Lee *et al.* [20] proposed a bi-clustering formulation that is closely related to ours. In particular, they proposed using the following formulation to extract one co-cluster at a time:

$$\min_{\rho\geq 0,\mathbf{a},\mathbf{b}\ |\ ||\mathbf{a}||_2=||\mathbf{b}||_2=1} \|\mathbf{X}-\rho\mathbf{a}\mathbf{b}^T\|_F^2 + \lambda_a\rho\sum_i\mathbf{w}_1(i)|\mathbf{a}(i)| + \lambda_b\rho\sum_j\mathbf{w}_2(j)|\mathbf{b}(j)|, \quad (7)$$

where the $\mathbf{w}_1(i)$'s and $\mathbf{w}_2(j)$'s are data-dependent weights depending on the magnitude of the conditional least squares estimate of the respective vector element. In [20], parameters λ_a, λ_b are chosen according to the Bayesian Information Criterion, conditioned on interim estimates of $\mathbf{b}$, $\mathbf{a}$, respectively; parameter tuning is embedded in the overall iteration, i.e., λ_a is modified after each update of $\mathbf{b}$, and λ_b after each update of $\mathbf{a}$. Adapting the sparsity penalties using intermediate estimates of $\mathbf{a}$ and $\mathbf{b}$ may work well in many cases, but it also implies that subsequent updates do not reduce a well-defined cost function, opening the door to potential instability. In fact, it is not difficult to find cases where [20] oscillates between two states; an example is provided in the Appendix. For this and other reasons, we fix the sparsity penalties throughout the iterations. Our method of selecting parameters λ_a, λ_b will be discussed in Section IV.

Considering the plain version of (7) with all weights equal ($\mathbf{w}_1(i) = \mathbf{w}_2(j) = 1, \forall i,j$), i.e.,

$$\min_{\rho\geq 0,\mathbf{a},\mathbf{b}\ |\ ||\mathbf{a}||_2=||\mathbf{b}||_2=1} \|\mathbf{X}-\rho\mathbf{a}\mathbf{b}^T\|_F^2 + \lambda_a\rho||\mathbf{a}||_1+\lambda_b\rho||\mathbf{b}||_1, \quad (8)$$

the motivation for penalizing $\rho\|\mathbf{a}\|_1$ instead of $\|\mathbf{a}\|_1$ is not clear in [20] – note that ρ is the overall scaling of the outer product $\mathbf{a}\mathbf{b}^T$, not the individual scaling of $\mathbf{a}$; hence $\rho\|\mathbf{a}\|_1$ cannot be interpreted as the one-norm of the 'effective' $\mathbf{a}$, and likewise for $\mathbf{b}$. It seems that part of the reasoning behind [20] was to enable use of a simple block-coordinate descent strategy: notice that optimizing $(\rho,\mathbf{a})$ conditioned on $\mathbf{b}$ is a Lasso problem [32], which can be efficiently solved; and likewise for the conditional update of $(\rho,\mathbf{b})$ given $\mathbf{a}$. It is easy to see that in order to synthesize a constant co-cluster 'patch', the penalty in (7) is proportional to co-cluster level. As a side comment, note that in (5) we impose non-negativity constraints on the elements of $\mathbf{a}$ and $\mathbf{b}$, when appropriate.

Starting from a different perspective, Witten *et al.* [35] proposed bilinear decomposition with sparsity-inducing hard ℓ_1 constraints on both the left and right latent vectors, as a variation of sparse SVD and sparse canonical correlation analysis. Their model was not developed with co-clustering in mind, but it is similar to our formulation of bi-clustering – which uses soft ℓ_1 penalties instead of hard ℓ_1 constraints, and optionally non-negativity when appropriate. Hoyer [16] considered adding hard sparsity constraints to NMF with the aim of improving the interpretability of the decomposition in applications of NMF (co-clustering was not considered in [16]). Hoyer used a sparsity measure that combines the ℓ_1 and ℓ_2 norms.

References [16], [20], [35] did not consider extensions to the higher-way case, which is the topic of the next subsection.

C. Extension to Three- and Higher-Way Co-Clustering: PARAFAC With Sparse Latent Factors

In many cases, one works with data sets indexed by three or more variables, instead of two (as in matrices). A good example is several batches of gene expression data measured over several experimental conditions in two or more occasions or by different labs. Another is social network data, such as the ENRON e-mail corpus, where we have e-mail counts from sender to receiver as a function of time, stored in a three-way array $\underline{\mathbf{X}}$ whose (i,j,n)-th element $\underline{\mathbf{X}}(i,j,n)$ is the number of packets send by transmitting node i to receiving node j during time interval n. The natural generalization of the bi-clustering approach in (3) to tri-clustering is to consider a trilinear outer product decomposition

$$\underline{\mathbf{X}} \cong \sum_{k=1}^{K}\mathbf{a}_k\circ\mathbf{b}_k\circ\mathbf{c}_k,$$

with sparsity on all latent factors. Without sparsity, the above is the PARAFAC model, and K is the exact or 'essential' rank of $\underline{\mathbf{X}}$, depending on whether one seeks an exact decomposition or a low-rank approximation. Note that latent sparsity is key here, because the whole point of co-clustering is to *select* subsets along each mode. Even without sparsity, however, the PARAFAC decomposition is unique under relatively mild conditions – even in certain cases where $K \gg \min(I,J)$ (e.g.,

see [19], [31]). This means that our formulation of (overlapping and lossy) three-way co-clustering can *reveal the true latent patterns* in the data when used as an exploratory tool, even for a large number of co-clusters – possibly even exceeding some or all dimensions of the three-way array $\underline{\mathbf{X}}$. This is not the case for bi-clustering, which is either NP-hard (in the case of hard bi-clustering) or lacks uniqueness (in the case of soft bi-clustering).

1) Related Work: There are very few papers on tri-clustering in the literature [28], [36], [37] (note that tri-clustering is very different from K-means clustering of three-way data, as considered, e.g., in [17]). Off-the-shelf non-negative PARAFAC has been used for tri-clustering of web data in [37], albeit without motivation as to why it is an appropriate tool for co-clustering. A hybrid PARAFAC-Tucker model is proposed in [28], again without clear motivation regarding its application to co-clustering. Still, these are the closest pieces of work, and so we will use non-negative PARAFAC as a baseline for comparison in our simulations. We underscore, however, that latent sparsity is key in our present context, because the whole point of co-clustering is to *select* subsets along each mode. Latent factor sparsity has not been considered in the aforementioned references, which did not start from a 'first principles' formulation, as we did.

One may wonder if there is a need to impose sparsity in our present context, in light of uniqueness of unconstrained (or non-negative) PARAFAC. The answer is two-fold. First, in practice we compute truncated PARAFAC approximations, instead of a full decomposition; noise and unmodeled dynamics will thereby render the extracted factors non-zero everywhere, with probability one. This destroys the support information that is crucial for co-clustering. Enforcing latent sparsity suppresses noise and automatically selects the desired support in all modes, simultaneously. Second, the imposition of sparsity (and non-negativity) may allow stable extraction of more co-clusters than would otherwise be possible with plain PARAFAC. For these two reasons, sparsity constraints are very important here.

Motivated by the aforementioned considerations, we may consider the following formulation of tri-clustering

$$\min_{\{\mathbf{a}_k\geq 0,\mathbf{b}_k\geq 0,\mathbf{c}_k\geq 0,\}_{k=1}^K} \left\|\underline{\mathbf{X}}-\sum_{k=1}^K \mathbf{a}_k\circ\mathbf{b}_k\circ\mathbf{c}_k\right\|_F^2 + \lambda_a\sum_k\|\mathbf{a}_k\|_1+\lambda_b\sum_k\|\mathbf{b}_k\|_1+\lambda_c\sum_k\|\mathbf{c}_k\|_1. \quad (9)$$

However, it is easy to prove by contradiction (similar to Claim 1) that norm balancing now extends across all three modes (and in fact slows down convergence of alternating optimization schemes even further):

Claim 3: Let $\left\{\widehat{\mathbf{a}}_k,\widehat{\mathbf{b}}_k,\widehat{\mathbf{c}}_k\right\}_{k=1}^K$ be a solution of (9), for $\lambda_a,\lambda_b,\lambda_c$ all >0. Then

$$\lambda_a\|\widehat{\mathbf{a}}_k\|_1=\lambda_b\|\widehat{\mathbf{b}}_k\|_1=\lambda_c\|\widehat{\mathbf{c}}_k\|_1,\quad \forall k. \quad (10)$$

This leads us to propose the following formulation of tri-clustering:

$$\min_{\{0\leq\rho_k\leq\bar{\rho},0\leq\mathbf{a}_k,\mathbf{b}_k,\mathbf{c}_k\leq 1,\}_{k=1}^K} \left\|\underline{\mathbf{X}}-\sum_{k=1}^K \rho_k\mathbf{a}_k\circ\mathbf{b}_k\circ\mathbf{c}_k\right\|_F^2 + \lambda_a\sum_k\|\mathbf{a}_k\|_1+\lambda_b\sum_k\|\mathbf{b}_k\|_1+\lambda_c\sum_k\|\mathbf{c}_k\|_1, \quad (11)$$

where $\bar{\rho}:=\max_{i,j,n}\mathbf{X}(i,j,n)$.

If $\mathbf{X}$ and the latent factors have real-valued elements, then the problem becomes

$$\min_{\{|\rho_k|\leq\check{\rho},-1\leq\mathbf{a}_k,\mathbf{b}_k,\mathbf{c}_k\leq 1,\}_{k=1}^K} \left\|\underline{\mathbf{X}}-\sum_{k=1}^K \rho_k\mathbf{a}_k\circ\mathbf{b}_k\circ\mathbf{c}_k\right\|_F^2 + \lambda_a\sum_k\|\mathbf{a}_k\|_1+\lambda_b\sum_k\|\mathbf{b}_k\|_1+\lambda_c\sum_k\|\mathbf{c}_k\|_1, \quad (12)$$

where $\check{\rho}:=\max_{i,j,n}|\mathbf{X}(i,j,n)|$.

III. Algorithms

The problem in (5) is highly non-convex, so global optimal solution cannot be guaranteed. On the other hand, the formulation in (5) admits element-wise or block coordinate descent updates that reduce the cost and yield a monotonically improving sequence of admissible solutions, at low per-iteration complexity. Consider, for example, the update of a generic element of $\mathbf{A}$, denoted as α, conditioned on the remaining model parameters. It can be shown that the problem then boils down to

$$\min_{0\leq\alpha\leq 1}\|\mathbf{y}-\mathbf{d}\alpha\|_2^2+\lambda|\alpha|,$$

for given $\mathbf{y}$, and $\mathbf{d}$. Define

$$f(\alpha)=\|\mathbf{y}-\mathbf{d}\alpha\|_2^2+\lambda|\alpha|,\qquad g(\alpha)=\|\mathbf{y}-\mathbf{d}\alpha\|_2^2+\lambda\alpha,$$

and let α_f^* and α_g^* denote the minimizers of $f(\alpha)$ and $g(\alpha)$, respectively. Note that $f(\cdot)$ and $g(\cdot)$ are convex functions, $g(\cdot)\leq f(\cdot)$, and $g(\alpha)=f(\alpha)$ for $\alpha\geq 0$. The first derivative of $g(\alpha)$ is

$$\frac{dg(\alpha)}{d\alpha}=-2\mathbf{y}^T\mathbf{d}+2\alpha\mathbf{d}^T\mathbf{d}+\lambda,$$

and equating to zero yields

$$\alpha_g^*=\frac{\mathbf{y}^T\mathbf{d}-\frac{\lambda}{2}}{\mathbf{d}^T\mathbf{d}}.$$

There are three possibilities, as illustrated in Fig. 2:

1) $\alpha_g^*\in[0,1]$, in which case $\alpha_f^*=\alpha_g^*$.
2) $\alpha_g^*<0$, in which case $\alpha_f^*=0$.
3) $\alpha_g^*>1$, in which case $\alpha_f^*=1$.

Therefore, the optimal $[0,1]$-Lasso scalar update is given by

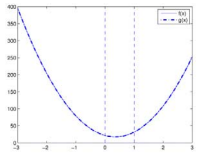
$$\alpha=\left[\frac{\mathbf{y}^T\mathbf{d}-\frac{\lambda}{2}}{\mathbf{d}^T\mathbf{d}}\right]_{[0,1]}.$$

The same type of problem arises when updating a single element of $\mathbf{B}$ – the model in (5) is symmetric in terms of the roles of $\mathbf{A}$ and $\mathbf{B}$. The conditional update of ρ given $\mathbf{A}$ and $\mathbf{B}$ is a simple scalar least squares problem: it can be put in the form

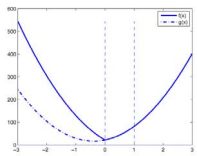
$$\min_{0\leq\rho\leq\bar{\rho}}\|\mathbf{x}-\mathbf{z}\rho\|_2^2$$

for given $\mathbf{x}$, $\mathbf{z}$; the solution is

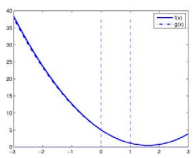
$$\rho=\left[\frac{\mathbf{z}^T\mathbf{x}}{\mathbf{z}^T\mathbf{z}}\right]_{[0,\bar{\rho}]}.$$



(a) case 1



(b) case 2



(c) case 3

Fig. 2. Three cases for the $[0, 1]$ constrained Lasso. (a) Case 1. (b) Case 2. (c) Case 3.

In the real-valued case, the scaling factor ρ is still restricted to be non-negative, but the latent factor elements take values in $[-1, 1]$. The scalar update problem becomes

$$\min_{-1\leq\alpha\leq 1} \|\mathbf{y} - \mathbf{d}\alpha\|_F^2 + \lambda|\alpha|$$

whose solution is

$$\alpha = \begin{cases} \left[\frac{\mathbf{y}^T\mathbf{d} - \frac{\lambda}{2}}{\mathbf{d}^T\mathbf{d}}\right]_{[0,1]} & \text{if } \mathbf{y}^T\mathbf{d} - \frac{\lambda}{2} > 0 \\ \left[\frac{\mathbf{y}^T\mathbf{d} + \frac{\lambda}{2}}{\mathbf{d}^T\mathbf{d}}\right]_{[-1,0]} & \text{if } \mathbf{y}^T\mathbf{d} + \frac{\lambda}{2} < 0 \\ 0 & \text{otherwise} \end{cases}.$$

The same type of element-wise updates can be used for both bi-clustering and tri- (or higher-way) co-clustering: the key point is that *multilinearity* renders the conditional updates in linear regression form. The overall algorithm cycles over the elements of $\mathbf{A}$, $\mathbf{B}$ ($\mathbf{C}, \ldots$), interleaved with updates of ρ, until convergence of the cost function. Initialization does matter: for bi-clustering, we initialize the iterations using a truncated SVD, or Non-negative Matrix Factorization (NMF). For tri-clustering, we use non-negative PARAFAC – in particular, Non-Negative Alternating Least Squares (NN-ALS), as implemented in the N-way Toolbox for Matlab [2]. Algorithm 1 is pseudo-code for the proposed tri-clustering algorithm for a single co-cluster ($K = 1$).

Remark 1: As we will soon see, a fortuitous side-benefit of our formulation enables far simpler computation of the dominant co-clusters in an incremental fashion; hence an algorithm for $K = 1$ is all that is needed in practice. After extracting a co-cluster $\rho\mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$ using Algorithm 1, we can remove it from the data (i.e., replace $\underline{\mathbf{X}}$ by $\underline{\mathbf{X}} - \rho\mathbf{a} \circ \mathbf{b} \circ \mathbf{c}$), and continue to extract another co-cluster, using Algorithm 1. Such a deflation procedure is generally inferior to jointly fitting all co-clusters at the same time; however here sparsity helps make it almost as good, as we will see.

Algorithm 1: Extracting a Single Tri-Cluster: Rank-One PARAFAC With Non-Negative Sparse Latent Factors

Input: $\underline{\mathbf{X}}$ of size $I \times J \times N$, λ_a, λ_b, λ_c

Output: $\mathbf{a}$ of size $I \times 1$, $\mathbf{b}$ of size $J \times 1$, $\mathbf{c}$ of size $N \times 1$, ρ

Unfold $\underline{\mathbf{X}}$ into $\mathbf{X}_{(1)}, \mathbf{X}_{(2)}, \mathbf{X}_{(3)}$ (see *notation and preliminaries* section)

Initialize $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ using NN ALS

Initialize $\rho = \max(\mathbf{a})\max(\mathbf{b})\max(\mathbf{c})$ and normalize $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ to $[0, 1]$.

Set upper bound $\rho_{\max} = \max(\max(\max(\underline{\mathbf{X}})))$ w/out loss of generality

while change in cost in (11) $> \epsilon$ **do**

$\mathbf{a} = \arg\min_{\mathbf{0}\leq\mathbf{a}\leq\mathbf{1}} \|\mathbf{X}_{(1)} - \rho(\mathbf{b} \odot \mathbf{c})\mathbf{a}^T\|_F^2 + \lambda_a \sum_i |\mathbf{a}(i)|$

$\rho = \arg\min_{0\leq\rho\leq\rho_{\max}} \|\mathbf{X}_{(1)} - \rho(\mathbf{b} \odot \mathbf{c})\mathbf{a}^T\|_F^2$

$\mathbf{b} = \arg\min_{\mathbf{0}\leq\mathbf{b}\leq\mathbf{1}} \|\mathbf{X}_{(2)} - \rho(\mathbf{c} \odot \mathbf{a})\mathbf{b}^T\|_F^2 + \lambda_b \sum_j |\mathbf{b}(j)|$

$\rho = \arg\min_{0\leq\rho\leq\rho_{\max}} \|\mathbf{X}_{(2)} - \rho(\mathbf{c} \odot \mathbf{a})\mathbf{b}^T\|_F^2$

$\mathbf{c} = \arg\min_{\mathbf{0}\leq\mathbf{c}\leq\mathbf{1}} \|\mathbf{X}_{(3)} - \rho(\mathbf{a} \odot \mathbf{b})\mathbf{c}^T\|_F^2 + \lambda_c \sum_n |\mathbf{c}(n)|$

$\rho = \arg\min_{0\leq\rho\leq\rho_{\max}} \|\mathbf{X}_{(3)} - \rho(\mathbf{a} \odot \mathbf{b})\mathbf{c}^T\|_F^2$

end while

A. Convergence Speed-Up Using Line Search

Despite its conceptual and computational simplicity and versatility, coordinate descent – type optimization can suffer from slow convergence. This is a well-known issue in the context of PARAFAC, where Alternating Least Squares (ALS – a block coordinate descent procedure) is commonly used for model fitting. In related contexts, it has been shown that *line search* is an effective way to speed-up convergence and avoid so-called *swamps* [29], [33]. In a nutshell, line search seeks to accelerate convergence by improving the present model estimate via extrapolation along a line drawn from the previous to the present estimate. Existing line search methods for PARAFAC are specifically designed for the least squares cost function, and modifications to account for the ℓ_1 parts of the cost are not obvious.

Consider a rank-one PARAFAC model with non-negative (NN) sparse latent factors (SLF). The model has four parameters: $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ and ρ. Define the direction vector

$$\mathbf{g}_a^{(it)} = \mathbf{a}^{(it)} - \mathbf{a}^{(it-1)},$$

with it denoting an iteration index, henceforth dropped for brevity. We similarly define direction vectors $\mathbf{g}_b$, $\mathbf{g}_c$, and the scalar g_ρ for $\mathbf{b}, \mathbf{c}$ and ρ, respectively. Then the optimal line search step s may be obtained by minimizing the function $\phi(s)$. See the equation at the bottom of the next page. Were it not for the absolute value terms, $\phi(s)$ would have been a polynomial of degree 8 in s, in which case the optimal step-size could have been found by rooting the derivative of $\phi(s)$. Due to the presence of the absolute value terms, however, $\phi(s)$ is

no longer polynomial. This is the main difficulty in converting existing methods for optimal step-size selection to work with our mixed $\ell_2 - \ell_1$ formulation. We have explored the following possibilities:

- Discrete line search (computing the minimum of $\phi(s)$ over a uniform sampling grid) is an obvious (albeit potentially computationally expensive) possibility, which could be refined using 'zoom' search or sub-gradient descent. Backtracking line search could also be used, as a computationally cheaper alternative. The important thing to keep in mind here is that line search need not be optimal to significantly cut down the required number of iterations.
- We know that $\phi(s)$ is not polynomial, but, keeping in mind that the line search step need only be good but not necessarily optimal, we may *approximate* $\phi(s)$ using a polynomial, then determine a good step-size by rooting the derivative of the approximating polynomial. A polynomial of order d can be determined by $d+1$ samples (values at particular locations)[2]. We have tried approximating $\phi(s)$ by a polynomial of degree $d = 8$, with very encouraging results, as we will see.
- We can aim for an iterative majorization-type algorithm. The main idea of iterative majorization is to upper bound the function you wish to minimize by another one that is always above it for all s, and equal to it at the current s. The upper bound should ideally be tight *and* easy to minimize, e.g., a quadratic, or a polynomial which we can easily root. It is easy to show that by minimizing such an upper bound we are guaranteed to reduce the desired function.

Towards obtaining a suitable majorizing function, note that the ℓ_1 part of $\phi(s)$ is separable, and the remainder is already a polynomial of degree 8 in s. It follows that it suffices to find a majorizing polynomial for a generic scalar term of the form $p(s) = |(a + sg_a)|$. The majorizing function $p_m(s|s_c)$ should have two properties: i) $p_m(s|s_c) \geq p(s)$ for all s, and ii) $p_m(s_c|s_c) = p(s_c)$. A suitable choice for our purposes can be drawn from [15]: $p_m(s|s_c) = \frac{1}{2}p(s_c) + \frac{1}{2}\frac{(a+sg_a)^2}{p(s_c)}$. Property i) holds due to the fact that $(|(a + sg_a)| - |(a + s_c g_a)|)^2 \geq 0$; property ii) can be verified by direct substitution of s_c. As the initial support point s_c, we propose using the solution obtained by approximating the cost function as an 8-th degree polynomial. Using $p_m(s|s_c)$ as a majorizing function for each absolute value term, and adding the squared norm part of $\phi(s)$, yields a majorizing polynomial of degree 8, whose derivative we root to obtain a new support point (testing all roots to find the one that yields the smallest value of the majorizing polynomial). This majorization – minimization (rooting) process is repeated until convergence of the cost to within a pre-specified tolerance. Monotone convergence to at least a local minimum is assured, see, e.g., [15].

[2]Let $\mathbf{v}(s)$ be a Vandermonde vector with generator s, and $\boldsymbol{\sigma}$ denote the polynomial's coefficient vector. Then the value of the polynomial at point s is $\boldsymbol{\sigma}^T\mathbf{v}(s)$.

B. Handling Missing Values

In exploratory data analysis, there are cases where a (possibly large) proportion of the sought data is not available. Still, co-clustering on the basis of whatever data is available can be relevant and is often desirable. A good way to handle missing values (in the absence of prior/side information) is to ignore them in the model fitting process; e.g., see [34] for plain PARAFAC with missing values. Define a weight array $\underline{\mathbf{W}}$ such that:

$$\underline{\mathbf{W}}(i,j,n) = \begin{cases} 1 & \text{if } \underline{\mathbf{X}}(i,j,n) \text{ is available} \\ 0 & \text{otherwise} \end{cases}.$$

We build upon the imbalanced sparsity, rank one algorithm. The problem of interest can then be written as

$$\min_{\mathbf{0}\leq\mathbf{a},\mathbf{b},\mathbf{c}\leq\mathbf{1},0\leq\rho\leq M(\underline{\mathbf{X}})} \|\underline{\mathbf{W}} \circledast [\underline{\mathbf{X}} - \rho(\mathbf{a}\circ\mathbf{b}\circ\mathbf{c})]\|_F^2 + \lambda_a\sum_i|\mathbf{a}(i)| + \lambda_b\sum_j|\mathbf{b}(j)| + \lambda_c\sum_n|\mathbf{c}(n)|.$$

The key observation is this: in updating an element of $\mathbf{c}$, say $\mathbf{c}(n)$, conditioned on interim values for the remaining parameters, we may absorb the weights in the data and the remaining parameters. In particular, let $\underline{\mathbf{X}}(:,:,n)$ denote the $I \times J$ 'slice' of $\underline{\mathbf{X}}$ obtained by fixing the third index to value n, and likewise $\underline{\mathbf{W}}(:,:,n)$ denote the $I \times J$ slice of $\underline{\mathbf{W}}$. Then the conditional update of $\mathbf{c}(n)$ given all other parameters can be written as the minimization over $0 \leq \mathbf{c}(n) \leq 1$ of

$$\left\|\underline{\mathbf{W}}(:,:,n) \circledast \underline{\mathbf{X}}(:,:,n) - \left(\underline{\mathbf{W}}(:,:,n) \circledast \left(\rho\mathbf{a}\mathbf{b}^T\right)\right)\mathbf{c}(n)\right\|_F^2 + \lambda_c|\mathbf{c}(n)|.$$

Letting $\mathbf{y} := vec\left(\underline{\mathbf{W}}(:,:,n) \circledast \underline{\mathbf{X}}(:,:,n)\right)$ and $\mathbf{r} := vec\left(\underline{\mathbf{W}}(:,:,n) \circledast \left(\rho\mathbf{a}\mathbf{b}^T\right)\right)$, the update of $\mathbf{c}(n)$ can be written as

$$\min_{0\leq\mathbf{c}(n)\leq 1} \|\mathbf{y} - \mathbf{r}\mathbf{c}(n)\|_2^2 + \lambda_c|\mathbf{c}(n)|.$$

This way, our previous algorithms can be reused to account for missing data, without further modification.

IV. Guidelines on Choosing λ_a, λ_b, and λ_c

The original Lasso objective function is

$$\min \frac{1}{2}\|\mathbf{y} - \mathbf{M}\mathbf{z}\|_2^2 + \lambda\sum_i|\mathbf{z}(i)|.$$

$$\phi(s) = \sum_{i,j,n}\left(\underline{\mathbf{X}}(i,j,n) - (\rho + sg_\rho)\left(\mathbf{a}(i) + s\mathbf{g}_a(i)\right)\left(\mathbf{b}(j) + s\mathbf{g}_b(j)\right)\left(\mathbf{c}(n) + s\mathbf{g}_c(n)\right)\right)^2 + \lambda_a\sum_i|\left(\mathbf{a}(i) + s\mathbf{g}_a(i)\right)| + \lambda_b\sum_j|\left(\mathbf{b}(j) + s\mathbf{g}_b(j)\right)| + \lambda_c\sum_n|\left(\mathbf{c}(n) + s\mathbf{g}_c(n)\right)|.$$

In [24], it is shown that $\mathbf{z}_{\text{opt}} = \mathbf{0}$ for any $\lambda \geq \lambda^*$, with

$$\lambda^* = \|\mathbf{M}^T\mathbf{y}\|_\infty \tag{13}$$

where $\|\mathbf{v}\|_\infty$ denotes the infinity norm (the maximum absolute value of the elements of $\mathbf{v}$).

We focus on the rank-one imbalanced sparsity model and the derivation of a suitable bound on λ_c. Formulas for λ_b, λ_c can be analogously derived. Consider the conditional update of $\mathbf{c}$ given the remaining parameters. The pertinent part of the cost can be written as

$$\min \|\mathbf{X}_{(3)} - \rho(\mathbf{a} \odot \mathbf{b})\mathbf{c}^T\|_F^2 + \lambda_c \sum_n |\mathbf{c}(n)|,$$

where $\mathbf{X}_{(3)}$ is the $JI \times N$ matrix unfolding of the three-way array $\underline{\mathbf{X}}$. Let us further zoom down to the scalar case – the conditional update of $\mathbf{c}(n)$ given all remaining parameters (including other elements of the vector $\mathbf{c}$). The function to minimize can then be written as

$$\min \|\mathbf{X}_{(3)}(:,n) - \rho(\mathbf{a} \odot \mathbf{b})\mathbf{c}(n)\|_2^2 + \lambda_c|\mathbf{c}(n)|,$$

where $\mathbf{X}_{(3)}(:,n)$ is the n-th column of matrix $\mathbf{X}_{(3)}$. The bound for Lasso in (13) implies that the optimal $\mathbf{c}(n)$ will be zero when λ_c exceeds

$$\begin{aligned}\lambda^*_{c,n} &= 2\|\rho(\mathbf{a} \odot \mathbf{b})^T\mathbf{X}_{(3)}(:,n)\|_\infty = 2\|\rho(\mathbf{a} \odot \mathbf{b})^T\mathbf{X}_{(3)}(:,n)\|_2 \\ &\leq 2\rho\|(\mathbf{a} \odot \mathbf{b})^T\|_2\|\mathbf{X}_{(3)}(:,n)\|_2 \leq 2\rho_{\max}IJ\|\mathbf{X}_{(3)}(:,n)\|_2,\end{aligned}$$

where the equality holds because the expression within the infinity norm is scalar, the first inequality is the Cauchy-Schwartz, and the second inequality simply uses that any element of $\mathbf{a}$ and $\mathbf{b}$ is upper bounded by 1. This immediately yields an upper bound on λ_c

$$\lambda^*_c = 2\rho_{\max}IJ \max_n \|\mathbf{X}_{(3)}(:,n)\|_2$$

which depends solely on the input data array $\underline{\mathbf{X}}$. Had we known *a-priori* an estimate of $\|\mathbf{a}\|_0$ and $\|\mathbf{b}\|_0$, i.e., the number of non zero elements of the 'true' $\mathbf{a}$ and $\mathbf{b}$, we could substitute those estimates in lieu of I and J, thus obtaining a tighter bound for λ_c.

Several authors have considered the problem of tuning the regularization parameter λ for the usual linear regression version of Lasso [27], [32], [38], assuming availability of training data for cross-validation, or that the data come from a known distribution. Here, however, we deal with what can be viewed as a multi-linear extension of Lasso, applied in an exploratory data analysis mode, where labeled data or detailed statistical information is hard to come by. Instead, it is more plausible to assume that the practitioner has (perhaps coarse) knowledge of the expected number of non-zero elements in $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, which reflect the expected co-cluster membership along the different modes. If this is the case, we may use such estimates of mode support to obtain 'clairvoyant' bounds λ^*_a, λ^*_b, λ^*_c. For example, if we know that the expected number of nonzero elements in mode 1 is $\bar{s}_1$ and in mode 2 it is $\bar{s}_2$, then we can set $\lambda^*_c = 2\rho_{\max}\bar{s}_1\bar{s}_2 \max_n \|\mathbf{X}_{(3)}(:,n)\|_2$. We have experimented with this idea, and found that choosing the λ's to be a small percentage (typically, 0.1%) of the respective clairvoyant bound works very well on both simulated and real data. For simulated data, this choice approximately recovers the desired number of nonzero elements per latent factor; for real data, it produces interpretable results.

V. Experimental Evaluation

In this section, we first compare our approach with established bi-clustering and recent tri-clustering methods, using synthetic data. In the process, we also illustrate the power of tri-clustering vis-a-vis bi-clustering of three-way data averaged out across one mode. We next turn to a well-known three-way dataset: the ENRON e-mail corpus, containing e-mail counts from sender to receiver as a function of time. We then assess the impact of the choice of λ's and the use of line search on convergence speed, using synthetic data as well as three measured datasets: the ENRON e-mail corpus, Amazon co-purchase data, and wine data. Finally, we illustrate performance with various proportions of missing entries for the ENRON data. All experiments were conducted on a 1.7 GHz Intel Core i5, with 4 GB of memory, using Matlab implementations of all algorithms considered.

A. Synthetic Data

Starting from an all-zero array $\underline{\mathbf{X}}$ of size $80 \times 80 \times 8$, we implant three co-clusters, two of which overlap in 'space' (first two modes) and 'time' (the third mode), whereas the third is isolated in space but overlaps with one of the other two in time:

$$\begin{aligned}\underline{\mathbf{X}}(20:24, 20:24, 1:3) &= 4\mathbf{I}_{5,5,3} \\ \underline{\mathbf{X}}(40:44, 70:74, 2:5) &= 2\mathbf{I}_{5,5,4} \\ \underline{\mathbf{X}}(37:41, 73:77, 4:8) &= 4\mathbf{I}_{5,5,5}\end{aligned}$$

where $\mathbf{I}_{I,J,K}$ denotes an $I \times J \times K$ array of unit elements. We then add i.i.d. sparse (Bernoulli-modulated) Gaussian noise: the probability that a sample is contaminated is set to 0.1, and if contaminated the noise is Gaussian, of zero mean and unit variance. In order to illustrate the performance of various bi-clustering algorithms (also *vis-a-vis* tri-clustering), we use the sum of frontal slices of $\underline{\mathbf{X}}$, which yields an 80×80 matrix $\mathbf{X}$. We further take the absolute value of $\mathbf{X}$ as the input to all bi-clustering algorithms, because, e.g., [10] assumes non-negative input data. The number of co-clusters to be extracted is set to $K = 3$, matching the actual number of co-clusters in the data. We illustrate the performance of the proposed methods against well-known bi-clustering [1], [5], [8]–[10], as well as tri-clustering [36], [37] methods.

1) Bi-Clustering: Fig. 3(a) – (e) demonstrate the performance of bi-clustering methods. We make a distinction between soft co-clustering (Fig. 3(a)) and hard co-clustering (Fig. 3(b) – (e)). For the latter, the number of extracted co-clusters should be at least $K = 3 + 1 = 4$, since the underlying noise in $\mathbf{X}$ must be assigned *somewhere*. We use $K = 4$ to capture the noise plus any systematic residual. Even though there is no guarantee that most of the noise will end up in the 'residual' co-cluster, this is what happens in this example. With soft co-clustering one does not need to provision extra co-cluster(s), since rows and columns can be left out altogether.

Fig. 3(a) shows the co-clusters extracted using the two-way analogue of the proposed algorithm, i.e., solving (5) with $\lambda_a = \lambda_b = 80$, using the three most significant left and right singular vectors to initialize $\mathbf{A}$ and $\mathbf{B}$. Observe that (5) manages to extract all three co-clusters, with minimal leakage. Fig. 3(b) shows

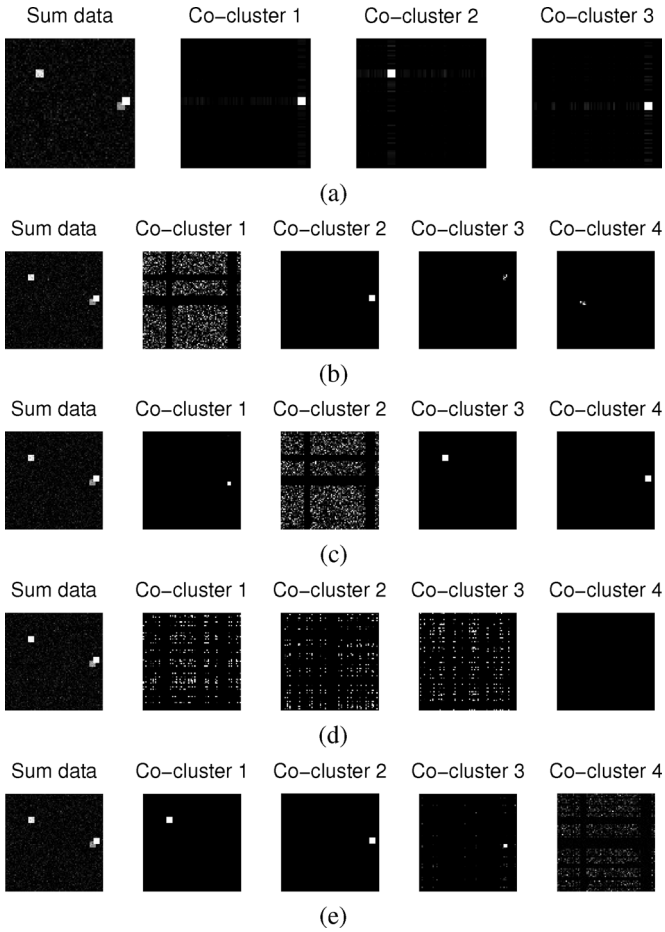


Fig. 3. Two-way co-clustering methods on synthetic data. (a) Sparse Matrix Regression on the sum matrix ($\lambda_a = \lambda_b = 80$). (b) K-means co-clustering on the sum matrix [1]. (c) Bi-clustering using spectral graph partitioning [10]. (d) Information-theoretic bi-clustering [9]. (e) Bregman bi-clustering using the Euclidean distance [5], [8].

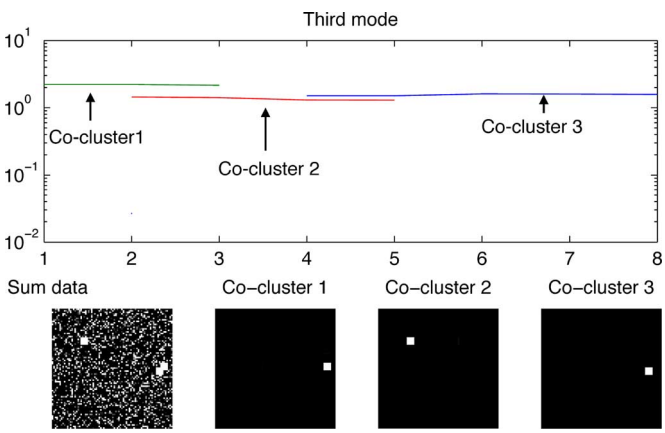


Fig. 4. Overlapping co-clusters: PARAFAC w/ NN SLF ($\lambda = 12$).

the output co-clusters of a simple K-means based approximation algorithm, as introduced in [1]. Observe that the co-clusters have been extracted with major losses, due to the overlap. Another approach that utilizes K-means for spectral graph partitioning is introduced in [10]; Fig. 3(c) shows the output of this algorithm, which has similar behavior to [1] but with slightly better definition – although the overlapping co-clusters have not been successfully resolved.

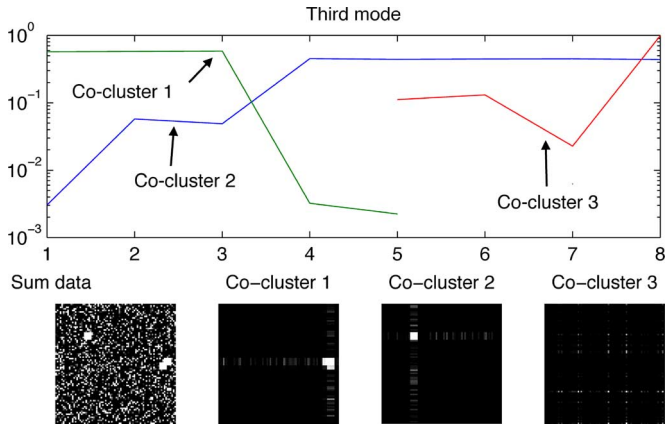


Fig. 5. Overlapping co-clusters: PARAFAC w/ NN ($\lambda = 0$) [28], [37].

TABLE I
COMPARISON OF CORRECT CLASSIFICATION RATES AND EXECUTION TIMES (SYNTHETIC DATA) PLUS COMPLEXITY ORDERS FOR ALL METHODS CONSIDERED. MULTIPLE PERCENTAGES ARE REPORTED FOR SMR AND [9], [10], DUE TO LOCAL MINIMA. PARAMETERS K_r AND K_c FOR THE LAST TWO ALGORITHMS INDICATE THE NUMBER OF ROW AND COLUMN CLUSTERS, RESPECTIVELY

Method	Correct %	Time	Complexity
PARAFAC NN SLF	97.5%	4.9 sec	$O(IJNK)$ / iter
[28], [37]	49%	1.3 sec	$O(IJNK)$ / iter
SMR	80.3-86.3%	2.95 sec	$O(IJK)$ / iter
[1]	71%	0.76 sec	$O(IJK)$ / iter
[10]	5.8-39.7%	0.05 sec	$O(IJK)$
[9]	14-71%	0.25 sec	$O(\|\|\mathbf{X}\|\|_0(K_r + K_c))$ / iter
[5] (Euclidean), cf. [8]	70.3%	0.25 sec	$O((K_r + K_c)J)$ / iter

The co-clustering framework introduced in [5] subsumes [9] (using the I-Divergence as loss function) and [8] (using the Euclidean distance as loss function). Fig. 3(d) shows the results of [9]. Finally, Fig. 3(e) shows the results of [5] (using the Euclidean distance as loss function), which manages to uncover all three co-clusters, albeit with noticeable loss in the overlapping region. We should note that [5], [8], [9] seek to find a checkerboard structure of co-clusters, i.e., *disjoint* sets of row and column indices of the data matrix, so it is no surprise that overlapping co-clusters cannot be fully recovered. Yet overlapping co-clusters often occur in applications, and resolving them is a major challenge in co-clustering.

2) Tri-Clustering: Fig. 4 shows the results of the proposed tri-clustering approach [cf. (11) with $\lambda_a = \lambda_b = \lambda_c = 12$] applied to the three-way synthetic dataset $\underline{\mathbf{X}}$. As a line of comparison, Fig. 5 shows the results of tri-clustering using PARAFAC with non-negativity but without latent sparsity, as suggested in [37]. In Fig. 5, observe how the two overlapping clusters have been merged into one; a 'phantom' co-cluster has emerged; and there is loss of localization (leakage) due to noise. It is important to note here that thresholding the results of Fig. 5 in a post-processing step may reduce leakage, but it will not recover the correct support information, and the phantom co-cluster will of course remain.

The following points may be distilled from our experiments with synthetic data:

- Tensor to matrix data reduction (going from $\underline{\mathbf{X}}$ to $\mathbf{X}$) leaves out useful temporal information, which can also be crucial for proper spatial resolution of the co-clusters, especially overlapping ones.

TABLE II
EXTRACTED CO-CLUSTERS FOR ENRON ($\lambda = 30$)

$K = 1$	Legal	-	-	-	-
$K = 2$	Legal	Executive, Govt. affairs	-	-	-
$K = 3$	Legal	Executive, Govt. affairs	Trading	-	-
$K = 4$	Legal	Executive, Govt. affairs	Trading	Pipeline	-
$K = 5$	Legal	Executive	Executive, Govt. affairs	Trading	Pipeline

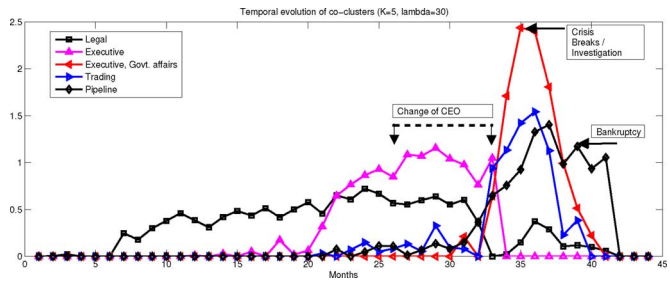


Fig. 6. Temporal co-cluster profiles for ENRON, $K = 5$ and $\lambda = 30$.

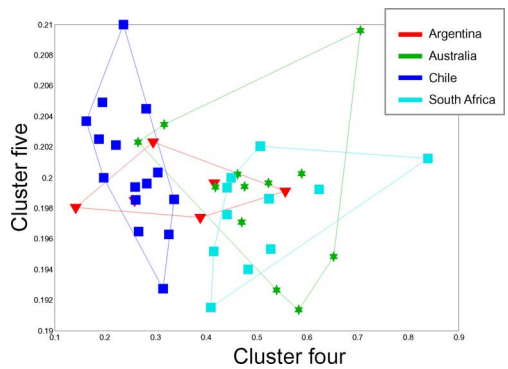


Fig. 7. Score plot based on tri-clustering analysis of wine data. Wine i is represented by a point whose coordinates are $\mathbf{A}(i,4)$ and $\mathbf{A}(i,5)$, indicating degree of membership in co-cluster 4 and 5, respectively.

- Latent sparsity is essential for recovering the correct support information in case of overlapping co-clusters; non-negativity alone does not work.

Table I compares all methods considered in terms of correct classification rate (the probability that a tensor/matrix element is correctly assigned to the co-cluster(s) that it belongs to) and execution time for the synthetic data. The complexity order of each method is also included. The proposed methods have the best co-clustering performance on the synthetic data, at the cost of moderately higher execution time.

B. Enron E-Mail Corpus

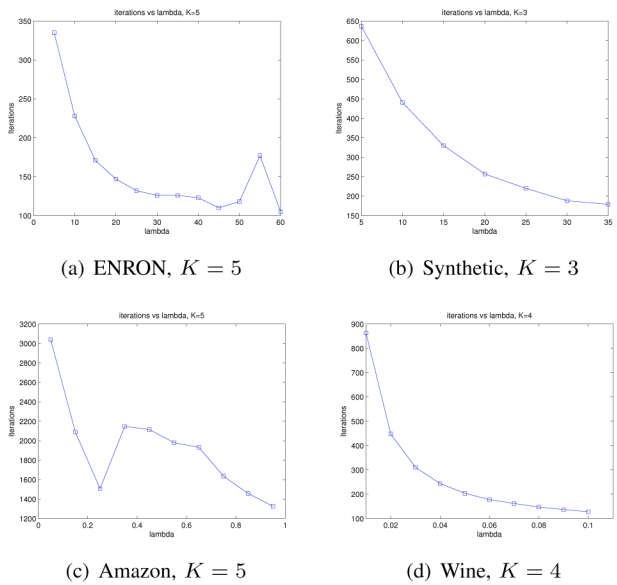
We used a summary version of the ENRON dataset stored in a three-way array $\underline{\mathbf{X}}$ of size $168 \times 168 \times 44$, containing the number of e-mails exchanged between 168 employees over 44 months (spanning from 1998 to 2002), indexed by sender, receiver, and month. Similar to [3], we first compress the dynamic range of the raw data using a logarithmic transformation: each non-zero entry of $\underline{\mathbf{X}}$ is mapped to $x' = \log_2(x) + 1$. We then fit a non-negative PARAFAC model with sparse latent factors to extract the dominant co-clusters ($\lambda_a = \lambda_b = \lambda_c = 12$). In our present context, each co-cluster captures a 'clique' and its temporal evolution. Total running time was 129.36 seconds. Table II summarizes the extracted cliques, which turn out to match the structure of the organization remarkably well – e.g., the label 'legal' in Table II means that the corresponding co-cluster contains the employees in the legal department (plus/minus one employee in all cases reported in Table II). Table II suggests that increasing K yields a nested sequence of co-clusters. This is indeed the case. The temporal profiles for $K = 5$ are plotted in Fig. 6. Two distinct peaks can be identified in the temporal communication patterns among the various cliques. Namely, the first peak can be found between the end of 2000 and the middle of 2001 (points 26–33 in Fig. 6), when a change of CEO occurred. The second peak corresponds to bankruptcy, between September and November 2001 (points 36–38 in Fig. 6). In [3], four class labels are identified: *Legal, Executive/Govt. Affairs, Executive, Pipeline*. The same class labels are also identified in [28], where non-negative PARAFAC is used, among other methods. Our results (cf. Table II) are qualitatively consistent with [3], [11], [28], but our cliques are cleaner, containing fewer outliers due to the imposition of sparsity.

C. Impact of Choice of λ and Line Search on Convergence Speed

Next, we illustrate the impact of the choice of λ ($= \lambda_a = \lambda_b = \lambda_c$ here) and the incorporation of line search on the required number of iterations. For this purpose we use the previously discussed data, plus Amazon co-purchase data downloaded from http://snap.stanford.edu/data/#amazon (cf. [22]), and wine data from [4]. The wine data is challenging in terms of speed of convergence, thus offering a rigorous test for line search. It is also challenging to interpret the results of co-clustering analysis of the wine data, since the differences between the various wines are subtle, so there are no clean and 'obvious' co-clusters. Fig. 7 shows a score plot based on tri-clustering analysis of the wine data for $\lambda_a = \lambda_b = \lambda_c = 1$. Wine i is represented by a point whose coordinates are $\mathbf{A}(i,4)$ and $\mathbf{A}(i,5)$, indicating degree of membership in co-cluster 4 and 5, respectively. These reflect subtle differences in relative concentrations of various aromatic compounds in the wine, which are hard to reveal without imposing latent sparsity, see [4].

Fig. 8 shows the number of iterations versus λ for all the datasets. With the exception of a few occasional problems with local minima, the general message is clear: the number of iterations until convergence (to within pre-specified tolerance of 10^{-8}) is a decreasing function of λ. The sparser the solution sought, the faster the convergence. This is reasonable, considering that higher λ's zero-out more elements (cf. the thresholding interpretation of element-wise updates).

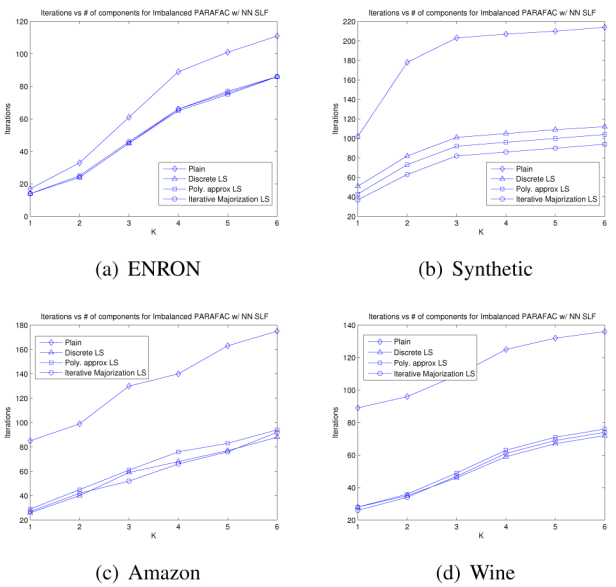
Fig. 9 shows the convergence speedup obtained by line search. We plot the total number of iterations required for convergence (with tolerance equal to 10^{-10}) versus the number of components. We present the number of iterations for 4 different versions of the PARAFAC w/ NN SLF algorithm: the plain one, and three variations obtained by incorporating



(a) ENRON, $K = 5$ (b) Synthetic, $K = 3$

(c) Amazon, $K = 5$ (d) Wine, $K = 4$

Fig. 8. Number of iterations vs λ, for fixed K. (



(a) ENRON (b) Synthetic

(c) Amazon (d) Wine

Fig. 9. Convergence speedup using line search, for fixed λ.

the three proposed line search schemes: grid, polynomial approximation, and polynomial majorization using polynomial approximation for initialization. From Fig. 9, it is clear that line search speeds up the convergence rate of the basic algorithm quite significantly. In most cases, polynomial approximation and polynomial majorization line search are slightly better than grid search in terms of number of iterations (a fine-resolution grid search was used in these experiments). Polynomial-based line search schemes are preferable in difficult scenarios, but our experience is that any reasonable line search scheme will usually reap most of the potential improvement.

D. Missing Values

Data analysis practitioners often have to deal with incomplete data – ranging from few randomly or systematically missing values all the way to very sparse data sets, where one has access to only a small percentage of the full data to be modeled

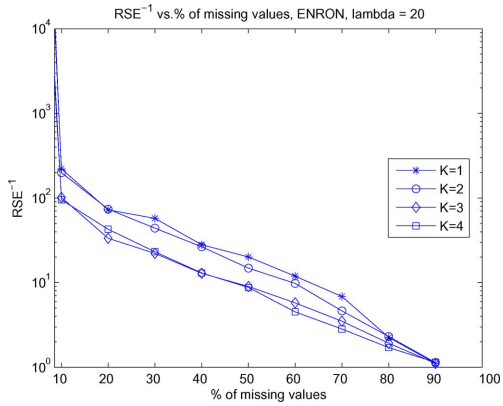


Fig. 10. RSE^{-1} (a signal-to-noise – like measure, see text) vs. % missing values.

or analyzed. We have already explained how data and regressor weighting can be tailored to ignore the missing values in the fitting process, relying on the postulated low-rank latent structure to *implicitly* interpolate what is missing. Here, we illustrate the effectiveness of this approach for randomly missing values, using the ENRON data as an example.

When fitting a model without full access to the pertinent data, the best one can hope for is to come close to the model fitted from the full data. Accordingly, a good figure of merit is the Relative Squared Error (RSE) between the models fitted from full and partial data. In our present context, one could compare at the level of individual co-clusters; but in order to plot a single meaningful metric, we compare the interpolated arrays $(\mathbf{A}_p \odot \mathbf{B}_p)\,\mathbf{C}_p^T$ vs. $(\mathbf{A}_f \odot \mathbf{B}_f)\,\mathbf{C}_f^T$, where subscripts $_p$ and $_f$ denote factor matrices fitted from partial and full data, respectively. Then RSE is defined as

$$\mathrm{RSE} := \frac{\|\,(\mathbf{A}_f \odot \mathbf{B}_f)\,\mathbf{C}_f^T - (\mathbf{A}_p \odot \mathbf{B}_p)\,\mathbf{C}_p^T\|_F^2}{\|\,(\mathbf{A}_f \odot \mathbf{B}_f)\,\mathbf{C}_f^T\|_F^2}$$

Fig. 10 shows RSE^{-1} (a signal-to-noise – like measure) versus the percentage of missing values for $\lambda = 20$ in all three modes, K ranging from 1 to 4 co-clusters. up to $K = 4$, and $\lambda = 20$. A simple i.i.d. Bernoulli model with miss probability q was used for selecting the missing values, and for each q the results are averaged over 10 realizations of the Bernoulli process. There are two points worth noticing. First, in this example RSE^{-1} stays roughly above 10 dB for up to 50% missing values, suggesting that we can tolerate a significant portion of missing values. Second, lower K yield higher RSE^{-1} for the same percentage of missing values. This is intuitive, because for lower K we have fewer unknowns to estimate from the same data.

VI. Discussion and Conclusions

Starting from first principles, we have formulated a new approach to multi-way co-clustering, as a constrained multilinear decomposition with sparse latent factors. In the case of three- and higher-way data, this corresponds to a PARAFAC decomposition with sparse latent factors. The inherent uniqueness of PARAFAC is further enhanced by sparsity, thereby allowing stable identification of a large number of possibly overlapping co-clusters. We have proposed an associated co-clustering algorithm that is based on simple coordinate updates coupled with

efficient line search, and useful guidelines on the choice of sparsity penalties. We have also explained how to handle missing values.

Intrigued by our experimental results, we further investigated the apparent *nesting property* that we first reported in [26]. Namely, we observed that as K increases, our formulation produces a nested sequence of co-clusters; the sequence of fitted models is nested in terms of the model order. This behavior is very different from that exhibited by the classical PARAFAC decomposition (with or without non-negativity), which often yields very different sets of rank-one components for successive values of K. Through extensive follow-up experiments (not detailed here due to space limitations), we have observed that the nesting property holds for strictly positive λ; it is only approximate for relatively low λ, and becomes more accurate with increasing λ. The property holds best for three-way data with latent non-negativity, but also holds well for three-way data without non-negativity. In the matrix case, it holds well when the data is non-negative (whether we enforce latent non-negativity or not), but it does not hold with much accuracy when the data matrix elements are of mixed signs.

We believe that nesting property is a fortuitous side-benefit of latent sparsity, aided by non-negativity to a certain extent. Despite trying, we have not been able to provide analytical insights as to why the nesting property holds. Analysis is complicated because nesting is not exact but only approximate. When latent sparsity is imposed, the vectors $\{\mathbf{a}_k \otimes \mathbf{b}_k \otimes \mathbf{c}_k\}_{k=1}^{K}$ often turn out being quasi-orthogonal, which goes some way towards explaining the nesting property in a qualitative way. Nesting has important practical implications, as far simpler and scalable 'deflation' algorithms can be used to extract one co-cluster at a time, without significant loss of optimality.

APPENDIX

Example of SSVD Limit Cycle: It is not difficult to find cases where SSVD [20] oscillates between two states; one such example is

$$\mathbf{X} = \begin{bmatrix} 0.06130 & -0.0743 & 0.00580 & 0.00580 & -0.0467 \\ -0.1872 & 0.22670 & -0.0176 & -0.0176 & 0.14250 \\ 0.13930 & -0.1686 & 0.01310 & 0.01310 & -0.1060 \\ -0.0688 & 0.08330 & -0.0065 & -0.0065 & 0.05240 \\ -0.0075 & 0.00900 & -0.0007 & -0.0007 & 0.00570 \end{bmatrix}, \tag{14}$$

and $K = 1$. We used the implementation provided by the authors of [20], downloaded from http://www.unc.edu/~haipeng/publication/ssvd-code.rar. In this example, SSVD oscillates between $\{\mathbf{u}_1, \mathbf{v}_1\}$, and $\{\mathbf{u}_2, \mathbf{v}_2\}$. The first dyad of vectors is $\mathbf{u}_1 = [0 \quad 0.8594 \quad -0.5113 \quad 0 \quad 0]^T$ and $\mathbf{v}_1 = [0 \quad 1 \quad 0 \quad 0 \quad 0]^T$; the second is $\mathbf{u}_2 = [-0.2444 \quad 0.7459 \quad -0.5548 \quad 0.2741 \quad 0.0297]^T$, and $\mathbf{v}_2 = [-0.5715 \quad 0.7044 \quad 0 \quad 0 \quad 0.4210]^T$. The same type of limit cycle behavior has been observed for non-negative data; for instance, when the absolute value of $\mathbf{X}$ in (14) is taken as input to SSVD.

REFERENCES

[1] A. Anagnostopoulos, A. Dasgupta, and R. Kumar, "Approximation algorithms for co-clustering," presented at the PODS, Vancouver, BC, Canada, Jun. 9–12, 2008.

[2] C. A. Andersson and R. Bro, "The N-way toolbox for MATLAB," *Chemometrics Intell. Lab. Syst.*, 2000 [Online]. Available: http://www.models.kvl.dk/nwaytoolbox

[3] B. W. Bader, R. A. Harshman, and T. G. Kolda, "Temporal analysis of social networks using three-way DEDICOM," Sandia National Labs., TR SAND2006-2161, 2006.

[4] D. Ballabio, T. Skov, R. Leardi, and R. Bro, "Classification of GC-MS measurements of wines by combining data dimension reduction and variable selection techniques," *J. Chemometrics*, vol. 22, no. 8, pp. 457–463, 2008.

[5] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha, "A generalized maximum entropy approach to Bregman co-clustering and matrix approximation," *J. Mach. Learn. Res.*, vol. 8, pp. 1919–1986, Aug. 2007.

[6] R. Bro, E. E. Papalexakis, E. Acar, and N. D. Sidiropoulos, "Co-clustering – A useful tool for Chemometrics," *J. Chemometrics*, vol. 26, no. 6, pp. 256–263, Jun. 2012.

[7] Y. Cheng and G. M. Church, "Biclustering of expression data," in *Proc. 8th Int. Conf. Intell. Syst. Molecular Biol.*, 2000, pp. 93–103.

[8] H. Cho, I. S. Dhillon, Y. Guan, and S. Sra, "Minimum sum-squared residue co-clustering of gene expression data," in *Proc. 4th SIAM Int. Conf. Data Min.*, 2004, pp. 114–125.

[9] I. S. Dhillon, S. Mallela, and D. S. Modha, "Information-theoretic co-clustering," in *Proc. 9th ACM SIGKDD*, 2003, pp. 89–98.

[10] I. S. Dhillon, "Co-clustering documents and words using bipartite spectral graph partitioning," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Min.*, 2001, pp. 269–274.

[11] J. Diesner, T. L. Frantz, and K. M. Carley, "Communication networks from the Enron email corpus: It's always about the people. Enron is no different," *Comput. Math. Organizat. Theory*, vol. 3, no. 3, pp. 201–228, 2005.

[12] W. J. Fu, "Penalized regressions: The bridge versus the Lasso," *J. Computat. Graph. Statist.*, vol. 7, no. 3, pp. 397–416, 1998.

[13] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis," *UCLA Working Papers in Phonet.*, vol. 16, pp. 1–84, 1970.

[14] J. A. Hartigan, "Direct clustering of a data matrix," *J. Amer. Statist. Assoc.*, vol. 67, pp. 123–129, 1972.

[15] W. J. Heiser, "Convergent computation by iterative majorization: Theory and applications in multidimensional data analysis," in *Recent Advances in Descriptive Multivariate Analysis*. New York: Oxford Univ. Press, 1994, pp. 157–189 [Online]. Available: http://www.oup.com/us/catalog/general/subject/Mathematics/ComputationalMathematics/?view=usa&ci=9780198522850

[16] P. O. Hoyer, "Non-negative Matrix factorization with sparseness constraints," *J. Mach. Learn. Res.*, vol. 5, pp. 1457–1469, 2004.

[17] H. Huang, C. Ding, D. Luo, and T. Li, "Simultaneous tensor subspace selection and clustering: The equivalence of high order SVD and k-means clustering," in *Proc. 14th ACM SIGKDD*, 2008, pp. 327–335.

[18] Y. Kluger, R. Basri, J. T. Chang, and M. Gerstein, "Spectral biclustering of microarray data: Coclustering genes and conditions," *Genome Res.*, vol. 13, pp. 703–716, 2003.

[19] J. B. Kruskal, "Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics," *Linear Algebra Appl.*, vol. 18, no. 2, pp. 95–138, 1977.

[20] M. Lee, H. Shen, J. Z. Huang, and J. S. Marron, "Biclustering via sparse singular value decomposition," *Biometrics*, vol. 66, no. 4, pp. 1087–1095, Dec. 2010.

[21] D. Lee and H. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[22] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The dynamics of viral marketing," *ACM Trans. Web (TWEB)*, vol. 1, no. 1, 2007, Article 5.

[23] S. C. Madeira and A. L. Oliveira, "Biclustering algorithms for biological data analysis: A survey," *IEEE/ACM Trans. Computat. Biol. Bioinf.*, vol. 1, no. 1, pp. 24–45, Jan.-Mar. 2004.

[24] M. R. Osborne, B. Presnell, and B. A. Turlach, "On the Lasso and its dual," *J. Computat. Graphic. Statist.*, vol. 9, no. 2, pp. 319–337, 2000.

[25] E. E. Papalexakis, N. D. Sidiropoulos, and M. N. Garofalakis, "Reviewer profiling using sparse matrix regression," in *Proc. IEEE Int. Conf. Data Min. Workshops*, 2010, pp. 1214–1219.

[26] E. E. Papalexakis and N. D. Sidiropoulos, "Co-clustering as multilinear decomposition with sparse latent factors," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, Prague, Czech Republic, May 22–27, 2011, pp. 2064–2067.

[27] T. Park and G. Casella, "The Bayesian Lasso," *J. Amer. Statist. Assoc.*, vol. 103, no. 482, pp. 681–686, 2008.

[28] W. Peng and T. Li, "Temporal relation co-clustering on directional social network and author-topic evolution," *Knowl. Inf. Syst.*, pp. 1–20, Mar. 2010.
[29] M. Rajih, P. Comon, and R. A. Harshman, "Enhanced line search: A novel method to accelerate PARAFAC," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1148–1171, 2008.
[30] I. Schizas, G. B. Giannakis, and N. D. Sidiropoulos, "Exploiting covariance-domain sparsity for dimensionality reduction," in *Proc. IEEE CAMSAP*, Aruba, Dutch Antilles, Dec. 13–16, 2009.
[31] T. Jiang and N. D. Sidiropoulos, "Kruskal's permutation lemma and the identification of CANDECOMP/PARAFAC and bilinear models with constant modulus constraints," *IEEE Trans. Signal Process.*, vol. 52, no. 9, pp. 2625–2636, Sep. 2004.
[32] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *J. Roy. Statist. Soc.. Ser. B (Methodol.)*, vol. 58, no. 1, pp. 267–288, 1996.
[33] G. Tomasi and R. Bro, "A comparison of algorithms for fitting the PARAFAC model," *Comput. Statist. Data Anal.*, vol. 50, no. 7, pp. 1700–1734, 2006.
[34] G. Tomasi and R. Bro, "PARAFAC and missing values," *Chemometrics Intell. Lab. Syst.*, vol. 75, no. 2, pp. 163–180, 2005.
[35] D. M. Witten, R. Tibshirani, and T. Hastie, "A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis," *Biostatist.*, vol. 10, no. 3, pp. 515–534, 2009.
[36] L. Zhao and M. J. Zaki, "Tricluster: An effective algorithm for mining coherent clusters in 3D microarray data," in *Proc. ACM SIGMOD 2005*, p. 705.
[37] Q. Zhou, G. Xu, and Y. Zong, "Web co-clustering of usage network using tensor decomposition," in *Proc. IEEE/WIC/ACM Int. Joint Conf. Web Intell. Intell. Agent Technol.*, 2009, vol. 3, pp. 311–314.
[38] H. Zou, T. Hastie, and R. Tibshirani, "On the degrees of freedom of the Lasso," *Ann. Statist.*, vol. 35, no. 5, pp. 2173–2192, 2007.

Evangelos E. Papalexakis (S'11) received the Diploma and M.Sc. degrees in electronic and computer engineering from the Technical University of Crete, Chania, Greece, in 2010 and 2011, respectively.

He is currently working toward the Ph.D. degree in the Computer Science Department of Carnegie Mellon University, Pittsburgh, PA. His research interests include data mining, tensor analysis, time evolving graph mining, and anomaly detection.

Nicholas D. Sidiropoulos (F'09) received the Diploma degree in electrical engineering from the Aristotelian University of Thessaloniki, Thessaloniki, Greece, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland—College Park, in 1988, 1990 and 1992, respectively.

He served as Assistant Professor at the University of Virginia from 1997 to 1999; Associate Professor at the University of Minnesota, Minneapolis, from 2000 to 2002; Professor at the Technical University of Crete, Greece, from 2002 to 2011; and Professor at the University of Minnesota from 2011 to present. His current research focuses primarily on signal and tensor analytics, with applications in cognitive radio, big data, and preference measurement.

Dr. Sidiropoulos received the NSF/CAREER award in 1998, the IEEE Signal Processing Society (SPS) Best Paper Award in 2001, 2007, and 2011, and the IEEE SPS Meritorious Service Award in 2010. He has served as IEEE SPS Distinguished Lecturer from 2008 to 2009, and Chair of the IEEE Signal Processing for Communications and Networking Technical Committee from 2007 to 2008.

Rasmus Bro studied mathematics and analytical chemistry and received the M.Sc. degree from the Technical University of Denmark in 1994 and the Ph.D. (*cum laude*) degree in multiway analysis from the University of Amsterdam, The Netherlands, in 1998.

Since 1994, he has been employed at the Department of Food Science at the University of Copenhagen (formerly Royal Veterinary & Agricultural University), and in 2002, he was appointed Full Professor of chemometrics. He has had several stays abroad at research institutions in The Netherlands, Norway, France, and the United States. His current research interests include chemometrics, multivariate calibration, multiway analysis, exploratory analysis, experimental design, numerical analysis, blind source separation, curve resolution, and constrained regression. He has authored more than 140 peer-reviewed scientific papers and three books on chemometrics.