# Tensors for Data Mining and Data Fusion: Models, Applications, and Scalable Algorithms

EVANGELOS E. PAPALEXAKIS, University of California Riverside
CHRISTOS FALOUTSOS, Carnegie Mellon University
NICHOLAS D. SIDIROPOULOS, University of Minnesota

Tensors and tensor decompositions are very powerful and versatile tools that can model a wide variety of heterogeneous, multiaspect data. As a result, tensor decompositions, which extract useful latent information out of multiaspect data tensors, have witnessed increasing popularity and adoption by the data mining community. In this survey, we present some of the most widely used tensor decompositions, providing the key insights behind them, and summarizing them from a practitioner's point of view. We then provide an overview of a very broad spectrum of applications where tensors have been instrumental in achieving state-of-the-art performance, ranging from social network analysis to brain data analysis, and from web mining to healthcare. Subsequently, we present recent algorithmic advances in scaling tensor decompositions up to today's big data, outlining the existing systems and summarizing the key ideas behind them. Finally, we conclude with a list of challenges and open problems that outline exciting future research directions.

CCS Concepts: • **General and reference** → **Document types**; • **Information systems** → **Information systems applications**; **Data mining**; • **Computing methodologies** → **Factorization methods**

Additional Key Words and Phrases: Tensors, tensor decomposition, tensor factorization, multi-aspect data, multi-way analysis

**ACM Reference Format:**
Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D. Sidiropoulos. 2016. Tensors for data mining and data fusion: Models, applications, and scalable algorithms. ACM Trans. Intell. Syst. Technol. 8, 2, Article 16 (October 2016), 44 pages.
DOI: http://dx.doi.org/10.1145/2915921

## 1. INTRODUCTION

Tensors are multidimensional extensions of matrices. Because of their ability to express multimodal or multiaspect data, they are very powerful tools in applications that inherently create such data. For instance, in online social networks, people tend to interact with each other in a variety of ways: they message each other, they post on each other's pages, and so on. All these different means of interaction are different aspects of the same social network of people, and can be modeled as a three-mode tensor, a "data cube," of (user, user, means of interaction). Given this tensor, there exists a rich variety of tools called tensor decompositions or factorizations that are able to extract meaningful, latent structure in the data.

In data mining, tensor decompositions have been very popular and successful in achieving state-of-the-art performance. The list of applications where tensors have

Authors' addresses: E. E. Papalexakis, 355 Winston Chung Hall, Computer Science & Engineering Department, University of California Riverside, 900 University Ave., Riverside, CA 92521, USA; C. Faloutsos, Dept. of Computer Science, GHC 8019, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213-3891; N. D. Sidiropoulos, 6-179 Keller Hall, Dept. of Electrical & Computer Engineering & Digital Technology Center, Dept. ECE, University of Minnesota, 200 Union St. SE, Minneapolis, MN 55455.

DOI: http://dx.doi.org/10.1145/2915921

been successful ranges from social network analysis to brain data analysis, and from web mining and information retrieval to healthcare analytics.

This survey is by no means the first attempt to summarize tensor decompositions. In fact, the work of Kolda and Bader [2009] is probably the most concise and most cited survey that contains a very detailed overview of different tensor decompositions. Subsequently, more survey papers have been published, some of them focusing on the applications [Acar and Yener 2009] and some on the algorithms [Cichocki et al. 2009; Lu et al. 2011; Grasedyck et al. 2013; Cichocki et al. 2015]. All the aforementioned surveys are great summaries of the work in a vast research area that spans the fields of chemometrics, psychometrics, signal processing, data mining, and machine learning.

However, we feel that the present survey differs from the existing ones in the following two ways: (1) we strongly emphasize the implications of the works we summarize from a practitioner's point of view, in terms of describing both the decompositions and the volume and breadth of the applications that this survey contains, and (2) the field is evolving very fast and many of the advances in applications and scalable algorithms have been published after the existing surveys.

In this article, we first give a few background definitions and notation in Section 2, and then present a selection of the most widely used decompositions in the aforementioned applications in Section 3. We give the essential definitions, provide state-of-the-art algorithms, and present the essence of each decomposition from a practitioner's perspective. In Section 4, we outline the application of tensor decompositions in a broad variety of real-world applications, outlining the particular ways that tensors have been used in each case. Subsequently, in Section 5, we provide a concise summary of scalable methods for tensor decompositions that have recently witnessed significant growth and have made tensor decompositions applicable to big data. Finally, in Section 6, we conclude by highlighting a few open challenges in tensor decompositions that mark interesting future research directions.

## 2. PRELIMINARY DEFINITIONS AND NOTATION

In this section, we provide a few necessary definitions and describe our notation. Table I summarizes our notation throughout this article.

*Definition* 2.1 (*Tensor*). A tensor is a multidimensional array, denoted by $\mathcal{X}$. We usually refer to the dimensions of $\mathcal{X}$ as modes. As a result, the order of a tensor is the number of its modes. For instance, we may refer to a third-order tensor as a three-mode tensor.

*Definition* 2.2 (*Outer Product*). Given two vectors $\mathbf{a} \in \mathbb{R}^I$ and $\mathbf{b} \in \mathbb{R}^I$, their outer product is an $I \times J$ matrix denoted by $\mathbf{a} \circ \mathbf{b}$. Its $(i, j)$ entry is $\mathbf{a}(i)\mathbf{b}(j)$. This definition can be extended to an arbitrary number of vectors.

*Definition* 2.3 (*Kronecker Product*). Given two matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{K \times L}$, their Kronecker product is an $IK \times JL$ matrix equal to

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} \mathbf{A}(1,1)\mathbf{B} & \cdots & \mathbf{A}(1,j)\mathbf{B} & \cdots & \mathbf{A}(1,J)\mathbf{B} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \mathbf{A}(i,1)\mathbf{B} & \cdots & \mathbf{A}(i,j)\mathbf{B} & \cdots & \mathbf{A}(i,J)\mathbf{B} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ \mathbf{A}(I,1)\mathbf{B} & \cdots & \mathbf{A}(I,j)\mathbf{B} & \cdots & \mathbf{A}(I,J)\mathbf{B} \end{bmatrix}.$$

*Definition* 2.4 (*Khatri-Rao Product*). Given two matrices $\mathbf{A}$ and $\mathbf{B}$, their Khatri-Rao product is the column-wise Kronecker product, that is,

$$\mathbf{A} \odot \mathbf{B} = \big[\mathbf{A}(:,1) \otimes \mathbf{B}(:,1) \ \cdots \ \mathbf{A}(:,j) \otimes \mathbf{B}(:,j) \ \cdots \ \mathbf{A}(:,J) \otimes \mathbf{B}(:,J)\big].$$

Table I. Table of Symbols

| Symbol | Definition |
|---|---|
| $\mathcal{X}, \mathbf{X}, \mathbf{x}, x$ | Tensor, matrix, column vector, scalar |
| $\mathbb{R}$ | The set of real numbers |
| $\circ$ | Outer product |
| vec( ) | Vectorization operator |
| diag($\mathbf{A}$) | Extract diagonal of matrix $\mathbf{A}$ |
| diag($\mathbf{a}$) | Diagonal matrix with $\mathbf{a}$ on the diagonal |
| $\otimes$ | Kronecker product |
| $\odot$ | Khatri-Rao product |
| $* \oslash$ | Element-wise multiplication and division |
| $\times_n$ | $n$-mode product |
| $\mathbf{X}_{(n)}$ | $n$-mode matricization of tensor $\mathcal{X}$ |
| $\mathbf{A}^{-1}$ | Inverse of $\mathbf{A}$ |
| $\mathbf{A}^{\dagger}$ | Moore-Penrose Pseudoinverse of $\mathbf{A}$ |
| $\mathrm{D}_{\mathrm{KL}}(a\|b)$ | KL-Divergence |
| $\|\mathbf{A}\|_F$ | Frobenius norm |
| $\mathbf{x}(i)$ | $i$th entry of $\mathbf{x}$ (same for matrices and tensors) |
| $\mathbf{x}(\mathcal{I})$ | Spans the elements of $\mathbf{x}$ corresponding to indices in set $\mathcal{I}$ |
| $\mathbf{X}(:, i)$ | Spans the entire $i$th column of $\mathbf{X}$ (same for tensors) |
| $\mathbf{X}(i, :)$ | Spans the entire $i$th row of $\mathbf{X}$ (same for tensors) |
| reshape( ) | Rearrange the entries of a given matrix or tensor to a given set of dimensions |
| numel( ) | For an $I_1 \times I_2 \cdots \times I_N$ tensor, returns $\prod_{n=1}^{N} I_n$ |
| MTTKRP | Matricized Tensor Times Khatri-Rao Product |

*Definition* 2.5 (*N-mode Product*). Given an $N$-mode tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $\mathbf{A} \in \mathbb{R}^{I_n \times R}$, the $n$-mode product of $\mathcal{X}$ and $\mathbf{A}$ is denoted as $\mathcal{Y} = \mathcal{X} \times_n \mathbf{A}$, where $\mathcal{Y} \in \mathbb{R}^{I_1 \times \cdots I_{n-1} \times R \times I_{n+1} \times \cdots \times I_N}$, and

$$\mathcal{Y}(i_1, \ldots, i_{n-1}, r, i_{n+1}, \ldots, i_n) = \sum_{j=1}^{I_n} \mathcal{X}(i_1, \ldots, i_{n-1}, j, i_{n+1}, \ldots i_N)\mathbf{A}(j, r).$$

*Definition* 2.6 (*Frobenius Norm of a Tensor*). The Frobenius norm of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is defined as

$$\|\mathcal{X}\|_F = \sqrt{\sum_{i_1}^{I_1} \sum_{i_2}^{I_2} \cdots \sum_{i_N}^{I_N} \mathcal{X}(i_1, i_2, \ldots, i_n)^2}.$$

*Definition* 2.7 (*N-mode Matricization*). An $N$-mode tensor can be unfolded or matricized into a matrix in $N$ ways, one for each mode. The $n$-mode matricization of $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted as $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_N}$ and is taken by keeping the $n$th mode intact and concatenating the slices of the rest of the modes into a long matrix. Figure 1 shows an example of a three-mode tensor and its one-mode matricization. Note that, as mentioned in Kolda and Bader [2009], there exist various definitions of the order in which the slices are concatenated during the matricization; however, the end result is the same, as long as the order is consistent across different matricizations.

The following property will be used in Section 3.1.2, where we derive the Alternating Least Squares Algorithm for the CP decomposition.

Property 1 (Vectorization and Khatri-Rao Product [Brewer 1978]). *Given matrices* $\mathbf{A}, \mathbf{B}$ *with the same number of columns and a diagonal matrix* $\mathbf{D} = \mathrm{diag}(\mathbf{d})$*, we have*

$$\mathrm{vec}(\mathbf{A}\mathbf{D}\mathbf{B}^T) = (\mathbf{B} \odot \mathbf{A})\mathbf{d}.$$

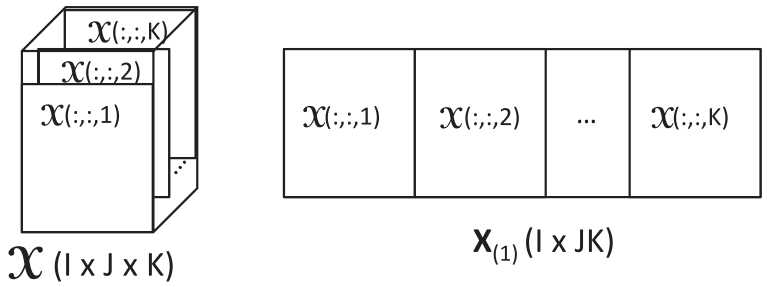


Fig. 1. One-mode matricization of $\mathcal{X}$ to matrix $\mathbf{X}_{(1)}$.

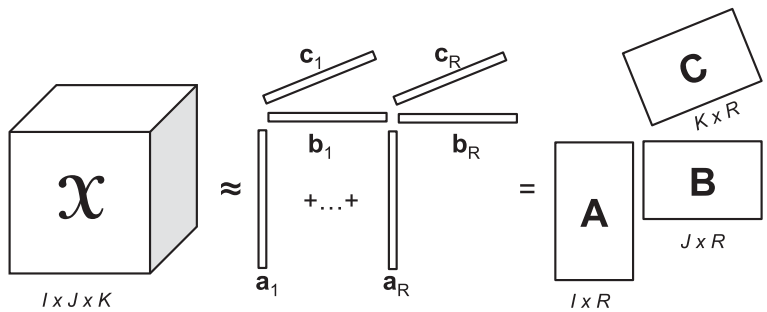


Fig. 2. The PARAFAC decomposition in its two equivalent representations, the sum of rank-one components and the factor matrices.

## 3. TENSOR DECOMPOSITIONS

There is a rich variety of tensor decompositions in the literature. In this section, we provide a comprehensive overview of the most widely used decompositions in data mining, from a practitioner's point of view.

### 3.1. CP Decomposition

*3.1.1. Definition.* The canonical polyadic (CP) decomposition (also known as PARAFAC or CANDECOMP) was independently proposed by Hitchcock [1927], Carroll and Chang [1970], and Harshman [1970]. The CP decomposition of a three-mode tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ is the sum of three-way outer products, that is,

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{1}$$

where $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$, $\mathbf{c}_r \in \mathbb{R}^K$, and their three-way outer product is given by

$$(\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r)(i, j, k) = \mathbf{a}_r(i)\mathbf{b}_r(j)\mathbf{c}_r(k) \quad \text{for all} \quad i, j, k.$$

We say $R$ is the number of components. The minimal $R$, the yield equality in Equation (1), is called the *rank* of the tensor. The *factor matrices* are defined as

$$\mathbf{A} = [\mathbf{a}_1 \ \mathbf{a}_2 \ \cdots \ \mathbf{a}_R] \in \mathbb{R}^{I \times R}$$
$$\mathbf{B} = [\mathbf{b}_1 \ \mathbf{b}_2 \ \cdots \ \mathbf{b}_R] \in \mathbb{R}^{J \times R}$$
$$\mathbf{C} = [\mathbf{c}_1 \ \mathbf{c}_2 \ \cdots \ \mathbf{c}_R] \in \mathbb{R}^{K \times R}.$$

Figure 2 shows a pictorial representation of the decomposition. Often, we may assume that the columns of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are normalized, and in this case, each latent component is accompanied by a scalar $\lambda_r$ that absorbs the scaling for $\mathbf{a}_r$, $\mathbf{b}_r$, and $\mathbf{c}_r$.

In the case of an $N$-mode tensor, we can readily extend the three-way CP decomposition by adding a new factor matrix for each additional mode. When $N > 3$, for notational simplicity we denote the $n$th factor matrix as $\mathbf{A}^{(n)}$, for $n = 1, \ldots, N$, and we write

$$\mathcal{X} \approx \sum_{r=1}^{R} \mathbf{a}_r^{(1)} \circ \mathbf{a}_r^{(2)} \circ \cdots \circ \mathbf{a}_r^{(N)}.$$

CP is one of the most popular tensor decompositions, in part due to its *ease of interpretation*. Each rank-one component of the decomposition serves as a latent "concept" or cluster in the data. The factor vectors for component $r$ can be interpreted as soft membership to the $r$th latent cluster. As Harshman [1970] stated when he introduced CP, this is an *explanatory* model. As such, it is suitable for exploratory data mining, as we will also see when we discuss applications (Section 4).

In addition to admitting a very intuitive interpretation, CP enjoys important theoretical properties. In particular, it is shown that CP is unique, under very mild conditions, up to scaling and permutation of the $R$ components [Kruskal 1977; Sidiropoulos and Bro 2000; ten Berge and Sidiropoulos 2002; Jiang and Sidiropoulos 2004; Stegeman et al. 2006; Chiantini and Ottaviani 2012]. It is beyond the scope of this survey to delve deeper into the theoretical underpinnings of CP uniqueness; however, we briefly mention relevant work: Kruskal [1977] gave the first three-way uniqueness result, and Sidiropoulos and Bro [2000] extended that to higher orders. It was known that the condition of Kruskal [1977] is sufficient but not necessary; in fact, Jiang and Sidiropoulos [2004] showed that it is not necessary when one factor matrix is full column rank. Stegeman et al. [2006] used the results of Jiang and Sidiropoulos [2004] to derive an almost-sure uniqueness result. Recently, Chiantini and Ottaviani [2012] provided the most relaxed conditions to date, using algebraic geometry.

Intuitively and practically, uniqueness means that the CP decomposition uncovers the actual latent factors rather than an arbitrarily rotated version. This is in sharp contrast to matrix factorization , where we generally have

$$\mathbf{X} \approx \sum_{r=1}^{R} \mathbf{a}_r \mathbf{b}_r^T = \mathbf{A}\mathbf{B}^T = \mathbf{A}\mathbf{Q}\mathbf{Q}^{-1}\mathbf{B}^T = \tilde{\mathbf{A}}\tilde{\mathbf{B}}^T$$

for any invertible $\mathbf{Q}$; that is, there exist multiple such decompositions of $\mathbf{X}$ that yield the same approximation error. The matrix singular value decomposition (SVD) is only unique because it adds an orthogonality constraint and only then if all the singular values are distinct. In practice, this means that in applications where we care not only about approximating the original data well but also about *interpreting* the latent factors, CP has a great advantage compared to matrix decomposition because it is generally the case that there is no equivalent rotated solution that yields the same fit.

*3.1.2. Algorithms.* In this section, we will elaborate on the Alternating Least Squares (ALS) algorithm for CP, which is probably the most widely used algorithm and dates back to the original papers by Carroll and Chang [1970] and Harshman [1970]. We also mention in passing all-at-once optimization approaches. For a detailed comparison of algorithms for CP, Tomasi and Bro [2006] and Kolda and Bader [2009] are excellent sources.

In order to solve for CP, we minimize the sum of squares of the difference between the tensor $\mathcal{X}$ and the model:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \left\| \mathcal{X} - \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \right\|_F^2. \tag{2}$$

The previous function is nonconvex; however, if we fix two of the factor matrices, the problem reduces to a linear least squares problem for the third one.

We show a particular way to derive the ALS algorithm in the three-way case. We can express CP in Equation (1) in "slab" format, relating each 2D slice of the tensor to the factor matrices, that is,

$$\mathcal{X}(:,:,k) \approx \mathbf{A}\mathbf{D}_k\mathbf{B}^T \quad \text{where} \quad \mathbf{D}_k = \text{diag}(\mathbf{C}(k,:)).$$

Using Property 1, after vectorizing the $k$th slice of the tensor into a long $IJ$ vector, we have

$$\text{vec}\,(\mathcal{X}(:,:,k)) \approx (\mathbf{B} \odot \mathbf{A})\,\mathbf{C}(k,:).$$

Then, gathering all the slices together yields

$$[\text{vec}\,(\mathcal{X}(:,:,1))\ \ \text{vec}\,(\mathcal{X}(:,:,2))\ \ \cdots\ \ \text{vec}\,(\mathcal{X}(:,:,K))] \approx (\mathbf{B} \odot \mathbf{A})[\mathbf{C}(1,:)\ \ \mathbf{C}(2,:)\ \ \cdots\ \ \mathbf{C}(K,:)].$$

Thus, we have

$$\mathbf{X}_{(3)}^T \approx (\mathbf{B} \odot \mathbf{A})\,\mathbf{C}^T \quad \Rightarrow \quad \mathbf{X}_{(3)} \approx \mathbf{C}\,(\mathbf{B} \odot \mathbf{A})^T\,,$$

where $\mathbf{X}_{(3)}$ is as defined in Definition 2.7. Similarly, we can write the first- and second-mode slabs of the tensor in similar ways:

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

The previous observation is very crucial because it offers three linear equations that relate a matricized version of the tensor (which is known) to one of the factor matrices. Thus, assuming that $(\mathbf{B} \odot \mathbf{C})$ is fixed, we can solve for $\mathbf{A}$ as

$$\mathbf{A} = \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})^{\dagger}.$$

Using a property of the pseudoinverse of the Khatri-Rao product (Eq. 2.2 in Kolda and Bader [2009]), this can be simplified to

$$\mathbf{A} = \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{C}^T\mathbf{C} * \mathbf{B}^T\mathbf{B})^{\dagger}.$$

For the general $N$-mode case, where we have factors $\mathbf{A}_1 \cdots \mathbf{A}_N$, the ALS solution for mode $n$ is

$$\begin{aligned}\mathbf{A}_{(n)} \ &= \mathbf{X}_{(n)}\,(\mathbf{A}_N \odot \cdots \odot \mathbf{A}_{n+1} \odot \mathbf{A}_{n-1} \odot \cdots \odot \mathbf{A}_1) \\ &\quad \left(\mathbf{A}_N^T\mathbf{A}_N * \cdots * \mathbf{A}_{n+1}^T\mathbf{A}_{n+1} * \mathbf{A}_{n-1}^T\mathbf{A}_{n-1} * \cdots * \mathbf{A}^{(1)T}\mathbf{A}^{(1)}\right)^{\dagger}.\end{aligned}$$

This gives rise to the ALS algorithm for the CP decomposition, which assumes that two out of the three matrices are fixed and solves for the third, iterating until a convergence criterion is met, for example, when the approximation error stops decreasing or when a certain number of iterations is reached. The ALS algorithm is a type of block coordinate descent algorithm and is guaranteed to decrease the approximation error monotonically. A listing of ALS is in Algorithm 1. In addition to solving for each factor matrix, in Algorithm 1 we normalize the columns of the factor matrices, which proves to be very important from an implementation point of view, offering stability to the algorithm. We absorb the norm of each component in the coefficients of an $R \times 1$ vector $\boldsymbol{\lambda}$.

It is important to note that the kernel

$$\mathbf{Y} = \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B}) \tag{3}$$

is also called *Matricized Tensor Times Khatri-Rao Product* or *MTTKRP* for short [Bader and Kolda 2007; Kolda and Bader 2009]. It can be similarly defined for $\mathbf{X}_{(2)}$ and $\mathbf{X}_{(3)}$ matricizations, as per the Alternating Least Squares Algorithm. This operation is key for scalability, as we will discuss in Section 5.1.

In addition to the ALS algorithm, we can also use all-at-once optimization. Paatero [1997] and Tomasi and Bro [2006] propose formulating the problem as nonlinear least squares and solving with a damped Gauss-Newton method. Acar et al. [2011] apply

**ALGORITHM 1:** Alternating Least Squares for CP

**Input:** Tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ and rank $R$.
**Output:** CP decomposition $\boldsymbol{\lambda}, \mathbf{A} \in \mathbb{R}^{I \times R}, \mathbf{B} \in \mathbb{R}^{J \times R}, \mathbf{C} \in \mathbb{R}^{K \times R}$

Initialize $\mathbf{A}, \mathbf{B}, \mathbf{C}$ (e.g., at random)
**while** convergence criterion is not met **do**
  $\mathbf{A} \leftarrow \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{C}^T\mathbf{C} * \mathbf{B}^T\mathbf{B})^{\dagger}$
  Normalize columns of $\mathbf{A}$
  $\mathbf{B} \leftarrow \mathbf{X}_{(2)}(\mathbf{C} \odot \mathbf{A})(\mathbf{C}^T\mathbf{C} * \mathbf{A}^T\mathbf{A})^{\dagger}$
  Normalize columns of $\mathbf{B}$
  $\mathbf{C} \leftarrow \mathbf{X}_{(3)}(\mathbf{B} \odot \mathbf{A})(\mathbf{B}^T\mathbf{B} * \mathbf{A}^T\mathbf{A})^{\dagger}$
  Normalize columns of $\mathbf{C}$ and store norm in $\boldsymbol{\lambda}$
**end while**

first-order gradient optimization methods such as limited-memory BFGS. Phan et al. [2013] exploit the structure of the Hessian for an efficient second-order optimization using Newton's method.

*3.1.3. Handling Missing Values.* When dealing with real data, there are many reasons we can expect some elements to be missing. Whether because of corruption, faulty measurements, or incomplete information (e.g., in recommender systems), there is a need to equip our algorithms with the ability to handle missing data. The traditional way for doing so is an expectation maximization approach that alternates between estimating the missing values with the current model and updating the model [Kiers 1997; Tomasi and Bro 2005]. This can be expensive since it requires repeatedly constructing the model; additionally, it is only feasible when the amount of missing data is fairly small. Another approach is to operate only on the observed data, ignoring every entry that is missing. Mathematically, this translates to

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \left\| \mathcal{W} * \left( \mathcal{X} - \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \right) \right\|_F^2,$$

where

$$\mathcal{W}(i,j,k) = \begin{cases} 1 & \text{if } (i,j,k) \text{ element is present,} \\ 0 & \text{if } (i,j,k) \text{ element is missing.} \end{cases}$$

The element-wise multiplication of the difference between the data and the model with this mask tensor $\mathcal{W}$ disregards entries that are missing from the optimization process. The works of Tomasi and Bro [2005] and Acar et al. [2010, 2011] show how this can be done algorithmically.

*3.1.4. Extensions.* Due to its popularity, there exist a variety of extensions to the CP decomposition, inspired by real-world constraints.

*Nonnegative Tensor Factorization*. In their seminal paper, Lee and Seung [1999] demonstrate that enforcing nonnegativity constraints on the factors of a matrix factorization can lead to more interpretable and intuitive results. Conceptually, when we are dealing with data that can be expressed as a "sum of parts," then incorporating nonnegativity constraints successfully expresses this property. In tensors, there exist algorithms that enforce nonnegativity constraints on the values of $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and have been shown to be more effective in terms of interpretability [Shashua and Hazan 2005; Cichocki et al. 2009]. We invite the interested reader to also explore the overview paper of Kolda and Bader [2009], which contains a more in-depth introduction to this subject.

*Sparsity on the Latent Factors*. Another constraint to enhance interpretability is sparsity. When the latent factors have only a few nonzero entries, an analyst can focus

on those, ignoring the zero entries, to understand the result of the decomposition. In addition to interpretability, Papalexakis et al. [2013] show that sparsity on the latent factors of CP actually leads to higher-order coclustering, which simultaneously groups rows, columns, and mode-three fibers (in case of a three-mode tensor) into so-called coclusters. In order to achieve sparsity on the latent factors, we use the $\ell_1$ norm, leading to

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \left\| \mathcal{X} - \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \right\|_F^2 + \lambda_a \sum_{i,r} |\mathbf{a}_r(i)| + \lambda_b \sum_{j,r} |\mathbf{b}_r(j)| + \lambda_c \sum_{k,r} |\mathbf{c}_r(k)|.$$

Furthermore, Papalexakis et al. [2013] hypothesize that imposing sparsity on the latent factors may produce nearly orthogonal factors and so allow for a *deflation* process where we compute only one rank-one component at a time.

*Boolean CP Decomposition*. Oftentimes, tensors contain binary relations (e.g., when we are recording whether two people spoke on the phone or not on a particular day). Miettinen [2011] proposes a set of Boolean tensor decompositions (CP and Tucker), where binary tensors are decomposed using models that operate in the Boolean algebra (e.g., $1 + 1 = 1$). The advantage of this approach is that for tensors that contain binary relations, such representation is more appropriate and respects that nature of the data.

*Modeling Sparse Count Data*. Chi and Kolda [2012] consider the problem that, in a wide variety of data mining applications, we are dealing with sparse "count" data. For instance, a tensor entry $(i, j.k)$ might indicate how many times person $i$ sent a message to person $j$ on a social network during week $k$. Traditionally, tensor algorithms seek to minimize the Frobenius norm of the difference between the data and the model Equation (2); however, using the Frobenius norm assumes that the data are normally distributed. In the case of sparse count data, this assumption is not well suited and may lead to results that misrepresent the data. On the other hand, postulating a Poisson distribution for the data turns out to be a more realistic assumption, which implies the use of the KL-Divergence as an objective function. Chi and Kolda [2012] demonstrate the effectiveness of this assumption, and Hansen et al. [2015] develop efficient algorithms.

*Incremental Decomposition*. Finally, in many real-world scenarios, the tensor is not static but generated dynamically and updated over time. Suppose, for instance, that the third mode of the tensor is being updated by new slices of data, for example, new interactions on a time-evolving social network. Instead of recomputing the CP decomposition of the updated data every time a new slice arrives, Nion and Sidiropoulos [2009] propose a method that tracks the decomposition, updating the factors given the new data and avoiding the overhead of rerunning the full-algorithm for every update.

**Practitioner's Guide for CP**
**Strengths**: Explanatory model, essentially unique under mild conditions, and easy to interpret.
**Weaknesses**: Hard to determine the appropriate rank and the global minimum.
**Utility**: Extracting latent factors for interpretation, explanatory clustering, etc.

### 3.2. Tucker Decomposition

*3.2.1. Definition.* Another extremely popular decomposition is the Tucker decomposition, originally proposed by Tucker [1966]. In fact, Tucker [1966] proposed three different models, but we are going to focus on the third, also known as Tucker-3 (henceforth referred to as Tucker). Kolda and Bader [2009] provide an excellent overview of Tucker-1 and Tucker-2. The Tucker decomposition was further popularized by De Lathauwer

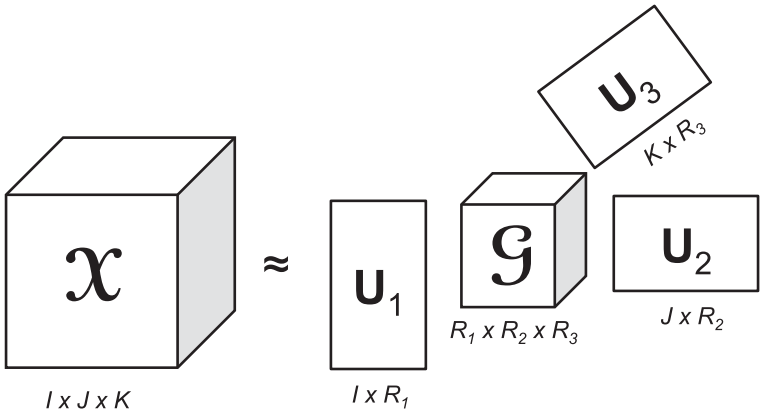


Fig. 3. The Tucker decomposition.

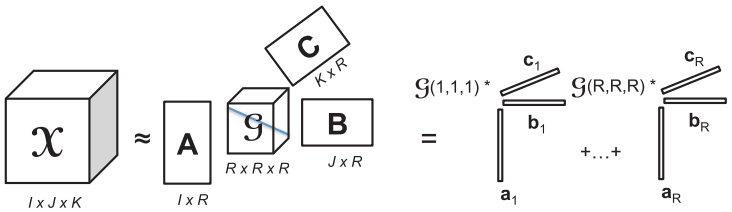


Fig. 4. CP can be seen as restricted Tucker, when the core is super-diagonal.

et al. [2000], wherein they coin the phrase Higher-Order Singular Value Decomposition (HOSVD) for a particular method for computing the Tucker decomposition.

In contrast to CP, Tucker decomposes a three-mode tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ into three factor matrices $\mathbf{U}_1 \in \mathbb{R}^{I \times R_1}$, $\mathbf{U}_2 \in \mathbb{R}^{J \times R_2}$, $\mathbf{U}_3 \in \mathbb{R}^{K \times R_3}$, which are also multiplied by a core tensor $\mathcal{G}$:

$$\mathcal{X} \approx \mathcal{G} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_3 \times_3 \mathbf{U}_3 \tag{4}$$

Alternatively, the decomposition can be written element-wise as

$$\mathcal{X}(i, j, k) \approx \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} \mathcal{G}(r_1, r_2, r_3) \mathbf{U}_1(i, r_2) \mathbf{U}_2(j, r_2) \mathbf{U}_3(k, r_3).$$

A pictorial example of Tucker is shown in Figure 3. The core tensor captures interactions between the columns of $\mathbf{U}_i$. Furthermore, we can assume that $\mathbf{U}_i$ are orthogonal and matrices, and as stated in Acar et al. [2007], they reflect the main subspace variation in each mode assuming a multilinear structure.

Tucker is nonunique, in contrast to CP. As in the matrix factorization case, we can rotate the factors without affecting the reconstruction error. However, Tucker yields a good low-rank approximation of a tensor (much like the SVD for matrices), in the squared error sense. This approximation can also be seen as compression, since the core tensor $\mathcal{G}$ is the best compression of the original tensor with respect to squared error, a property that we will revisit in Section 5.1 when discussing ways of speeding up the CP decomposition.

An interesting observation is that CP can be written as a Tucker model where $R_1 = R_2 = R_3 = R$, the factor matrices are *not* orthogonal, and the core only has entries along the superdiagonal. This is shown pictorially in Figure 4. This observation is used in estimating the model order of CP, as we will see in Section 3.8.

*3.2.2. Algorithms.* There exist two very popular algorithms for computing the Tucker decomposition. The first one is due to Tucker [1966] and popularized under the name Higher-Order Singular Value Decomposition [De Lathauwer et al. 2000], and the main idea behind it is that for each $\mathbf{U}_i$, it computes the $R_i$ leading singular vectors of the $i$th matricization of tensor $\mathbf{X}$. Having computed all $\mathbf{U}_i$ this way, it then computes the core tensor $\mathcal{G}$ conditioned on those matrices. In order to find the solution for $\mathcal{G}$, we can rewrite the Tucker model as

$$\text{vec}(\mathcal{X}) \approx (\mathbf{U}_3 \otimes \mathbf{U}_2 \otimes \mathbf{U}_1)\,\text{vec}(\mathcal{G}).$$

The least squares solution for $vec(\mathcal{G})$ is

$$\text{vec}(\hat{\mathcal{G}}) = (\mathbf{U}_3 \otimes \mathbf{U}_2 \otimes \mathbf{U}_1)^{\dagger}\,\text{vec}(\mathcal{X}),$$

which due to a property of the Kronecker product becomes

$$\text{vec}(\hat{\mathcal{G}}) = \left(\mathbf{U}_3^{\dagger} \otimes \mathbf{U}_2^{\dagger} \otimes \mathbf{U}_1^{\dagger}\right)\text{vec}(\mathcal{X}).$$

By orthogonality of the columns of $\mathbf{U}_n$, it holds that $\mathbf{U}_n^{\dagger} = \mathbf{U}_n^T$, and therefore,

$$\text{vec}(\hat{\mathcal{G}}) = \left(\mathbf{U}_3^T \otimes \mathbf{U}_2^T \otimes \mathbf{U}_1^T\right)\text{vec}(\mathcal{X}),$$

and by folding back the vectorized tensor to its original form, we have

$$\hat{\mathcal{G}} = \mathcal{X} \times_3 \mathbf{U}_3^T \times_2 \mathbf{U}_2^T \times_1 \mathbf{U}_1^T.$$

With the previous equation, we conclude the computation of Tucker using the HOSVD algorithm. A listing of HOSVD is shown in Algorithm 2. Note that HOSVD is not computing the optimal solution to the decomposition and does not compute joint subspaces of the tensor. However, it has been widely used due to its simplicity.

**ALGORITHM 2:** Higher-Order Singular Value Decomposition (HOSVD)

**Input:** $N-$mode tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and ranks $R_1, \ldots, R_N$.
**Output:** Tucker factors $\mathbf{U}_1 \in \mathbb{R}^{I_1 \times R_1}, \cdots \mathbf{U}_N \in \mathbb{R}^{I_N \times R_N}$ and core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$

1: **for** $n = 1 \cdots N$ **do**
2: $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] \leftarrow \text{SVD}(\mathbf{X}_{(n)})$
3: $\mathbf{U}_n \leftarrow \mathbf{U}(:, 1 : R_n)$, i.e., set $\mathbf{U}_n$ equal to the $R_n$ left singular vectors of $\mathbf{X}_{(n)}$
4: **end for**
5: $\mathcal{G} \leftarrow \mathcal{X} \times_N \mathbf{U}_N^T \times_{N-1} \mathbf{U}_{N-1}^T \cdots \times_1 \mathbf{U}_1^T$

The solution of HOSVD can also be used as a starting input for the second algorithm for Tucker, an ALS method called Higher-Order Orthogonal Iteration (HOOI). In order to derive HOOI, we need to compute the conditional update of $\mathbf{U}_i$ given the rest of the factor matrices. After manipulating the objective function (a detailed derivation can be found in Kolda and Bader [2009]), it turns out that in order to solve for $\mathbf{U}_1$, we need to maximize

$$\max_{\mathbf{U}_1} \left\| \mathbf{U}_1^T \underbrace{\left(\mathbf{X}_{(1)}(\mathbf{U}_3 \otimes \mathbf{U}_2)\right)}_{\mathbf{W}} \right\|_F^2.$$

The optimal solution to this maximization is the top $R_1$ singular vectors of $\mathbf{W} = \mathbf{X}_{(1)}(\mathbf{U}_3 \otimes \mathbf{U}_2)$. By symmetry of the model, in order to solve for $\mathbf{U}_2$, we take the top $R_2$ singular vectors of $\mathbf{X}_{(2)}(\mathbf{U}_3 \otimes \mathbf{U}_1)$, and for $\mathbf{U}_3$, we take the top $R_3$ singular vectors of $\mathbf{X}_{(3)}(\mathbf{U}_2 \otimes \mathbf{U}_1)$. For the $N$-mode case, the solution for $\mathbf{U}_n$ is by taking the top $R_n$ singular vectors of

$$\mathbf{X}_{(n)}\left(\mathbf{U}_N \otimes \cdots \mathbf{U}_{n+1} \otimes \mathbf{U}_{n-1} \otimes \cdots \otimes \mathbf{U}_1\right).$$

Combining these updates with the update for $\mathcal{G}$ that we showed earlier for HOSVD, we have HOOI, outlined in Algorithm 3.

**ALGORITHM 3:** Higher-Order Orthogonal Iteration (HOOI) for Tucker

**Input:** $N-$mode tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ and ranks $R_1, \ldots, R_N$.
**Output:** Tucker factors $\mathbf{U}_1 \in \mathbb{R}^{I_1 \times R_1}, \ldots \mathbf{U}_N \in \mathbb{R}^{I_N \times R_N}$ and core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times \cdots \times R_N}$

Initialize $\mathbf{U}_1, \ldots \mathbf{U}_n$ (e.g., at random or using Algorithm 2).
**while** convergence criterion is not met **do**
  **for** $n = 1 \cdots N$ **do**
    $\mathcal{W} \leftarrow \mathcal{X} \times_N \mathbf{U}_N^T \cdots \times_{n+1} \mathbf{U}_{n+1}^T \times_{n-1} \mathbf{U}_{n+1}^T \cdots \times_1 \mathbf{U}_1^T$
    $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] \leftarrow \text{SVD}(\mathbf{W}_{(n)})$
    $\mathbf{U}_n \leftarrow \mathbf{U}(:, 1 : R_n)$
  **end for**
**end while**
$\mathcal{G} = \mathcal{X} \times_N \mathbf{U}_N^T \times_{N-1} \mathbf{U}_{N-1}^T \cdots \times_1 \mathbf{U}_1^T$

*3.2.3. Handling Missing Values.* Tucker can handle missing values the same way as CP, by introducing the weight tensor $\mathcal{W}$ that masks the missing entries from the optimization. In addition to handling missing values, Tucker has been shown to be generally more effective at estimating missing values, compared to CP [Karatzoglou et al. 2010]. The reason for this behavior is the fact that Tucker, by virtue of its core that models interactions between components, can capture variation in the data that is not strictly trilinear.

*3.2.4. Extensions.*

*Boolean Tucker Decomposition*. As we saw in CP, Miettinen [2011] introduces Boolean tensor decomposition using the Tucker model as well.

*Incremental Decomposition*. In cases where the tensor is dynamically created or viewed as a stream of incoming slices, Sun et al. [2006] introduce an algorithm that can track the decomposition without recomputing it for every incoming slice.

*Handling Time*. Finally, the work of Sun et al. [2008] shows how to effectively handle time as one of the modes of the tensor. Due to the fact that time is treated as any other categorical attribute in a tensor (and the decompositions work as well if we permute the time positions), oftentimes time has to be treated carefully, especially when such permutation freedom violates the assumptions of the application. Sun et al. [2008] show how to model time using a wavelet decomposition and incorporate that into a Tucker decomposition.

**Practitioner's Guide for Tucker**
**Strengths**: Captures nontrilinear variation and compresses a tensor optimally.
**Weaknesses**: Nonunique and hard to interpret, especially if the core tensor is dense.
**Utility** Tensor compression (see Section 5.1), analyzing data where relations between latent components is expected, and estimating missing values (since it can capture variation that CP cannot). Note here that if the data have low-rank multilinear structure, CP may be better than Tucker at compressing that data; however, Tucker can compress well datasets that do not necessarily have low tensor rank.

### 3.3. DEDICOM and Related Models

*3.3.1. Definition.* In many real data applications, we have relations between entities (such as people in a social network) that are inherently asymmetric. For instance, when we are recording the number of emails that someone sent to a person, this

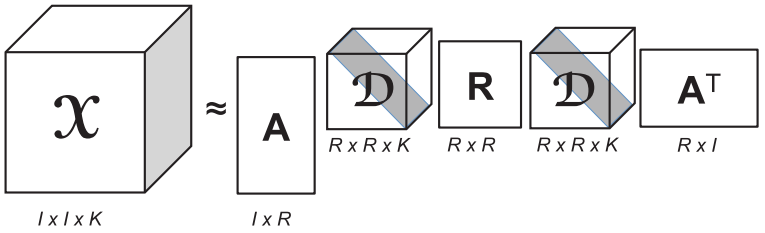


Fig. 5. The three-mode DEDICOM decomposition.

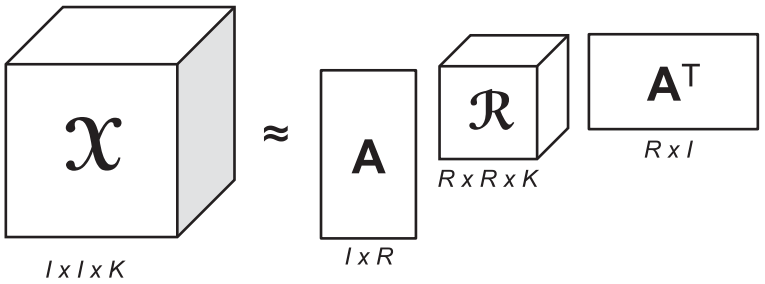


Fig. 6. The RESCAL decomposition.

(sender, receiver) relation is asymmetric. In order to analyze such data, there exists a tensor decomposition called DEDICOM that is able to capture such asymmetries. The decomposition was first proposed in Harshman [1978] and was later on used in Bader et al. [2007]. In DEDICOM, we write an $I \times I \times K$ tensor (note that the first two modes have the same size) as

$$\mathcal{X} \approx \mathbf{A}\mathcal{D}\mathbf{R}\mathcal{D}\mathbf{A}^T.$$

$\mathbf{A}$ is the loading or embedding matrix for the rows and columns of $\mathcal{X}$, which correspond to the same entities. The columns of $\mathbf{A}$ correspond to latent components, and as Bader et al. [2007] state, they can be thought of as *roles* that an entity in the rows of $\mathcal{X}$ participates in. Each slice of $\mathcal{D}$ is an $I \times I$ diagonal matrix, giving weights to the columns of $\mathbf{A}$, and $\mathbf{R}$ captures the asymmetric relations, and according to Bader et al. [2007], it captures the aggregate trends over the third mode (which in that particular paper is time but can be anything else). The model is shown pictorially in Figure 5.

A more recent variation of DEDICOM is called RESCAL and can be found in Nickel et al. [2011]. RESCAL was especially proposed for modeling multirelational data. The RESCAL decomposition is

$$\mathcal{X} \approx \mathbf{A}\mathcal{R}\mathbf{A}^T,$$

and the resemblance to DEDICOM is apparent; however, $\mathcal{R}$ is not restricted to a particular form and can capture more complex relations. An interesting observation here is that RESCAL is a restricted and symmetric Tucker model, where $\mathbf{W}$ is the identity matrix (leaving the corresponding mode uncompressed), and $\mathbf{U} = \mathbf{V} = \mathbf{A}$; Tucker models where one of the modes is uncompressed are also known as Tucker-2 [Tucker 1966; Kolda and Bader 2009]. The decomposition is also shown in Figure 6.

According to Nickel et al. [2012], its advantage over other tensor decompositions is that it can exploit a *collective learning* effect when applied to relational data. Collective learning refers to the automatic exploitation of attribute and relationship correlations across multiple interconnections of entities and relations.

**ALGORITHM 4:** ASALSAN Algorithm for DEDICOM

**Input:** Tensor $\mathcal{X} \in \mathbb{R}^{I \times I \times K}$ and latent dimension $R$.
**Output:** DEDICOM model $\mathbf{A}, \mathbf{R}, \mathcal{D}^{I \times I \times K}$

1: Initialize $\mathbf{A}, \mathbf{R}, \mathcal{D}$ at random (or per Bader et al. [2007]).
2: **while** convergence criterion is not met **do**
3: $$\mathbf{A} \leftarrow \left( \sum_{k=1}^{K} \left( \mathcal{X}(:,:,k)\mathbf{A}\mathcal{D}(:,:,k)\mathbf{R}^T\mathcal{D}(:,:,k) + \mathcal{X}(:,:k)^T\mathbf{A}\mathcal{D}(:,:,k)\mathbf{R}\mathcal{D}(:,:,k) \right) \right) \left( \sum_{k=1}^{K} \left( \mathbf{B}_k + \mathbf{C}_k \right) \right)^{-1}, \text{where}$$
$$\mathbf{B}_k = \mathcal{D}(:,:,k)\mathbf{R}\mathcal{D}(:,:k)\left(\mathbf{A}^T\mathbf{A}\right)\mathcal{D}(:,:,k)\mathbf{R}^T\mathcal{D}(:,:,k)$$
$$\mathbf{C}_k = \mathcal{D}(:,:,k)\mathbf{R}^T\mathcal{D}(:,:,k)\left(\mathbf{A}^T\mathbf{A}\right)\mathcal{D}(:,:,k)\mathbf{R}\mathcal{D}(:,:,k)$$
4: $$\text{vec}\left(\mathbf{R}\right) \leftarrow \left( \sum_{k=1}^{K} (\mathcal{D}(:,:,k)\mathbf{A}^T\mathbf{A}\mathcal{D}(:,:,k)) \otimes (\mathcal{D}(:,:,k)\mathbf{A}^T\mathbf{A}\mathcal{D}(:,:,k)) \right)^{-1} \sum_{k=1}^{K} \text{vec}\left(\mathcal{D}(:,:,k)\mathbf{A}^T\mathcal{X}(:,:,k)\mathbf{A}\mathcal{D}(:,:,k)\right)$$
5: Solve for $\mathcal{D}$ using Newton's method:
$$\min_{\mathcal{D}(:,:,k)} \|\mathcal{X}(:,:,k) - \mathbf{A}\mathcal{D}(:,:,k)\mathbf{R}\mathcal{D}(:,:,k)\mathbf{A}^T\|_F^2$$
6: **end while**

*3.3.2. Algorithms.* In this section, we will provide the latest ALS algorithm for DEDICOM, which is an improvement upon the original ALS algorithm introduced in Kiers [1993]. Algorithm 4 is called ASALSAN and was proposed in Bader et al. [2007]. As with all ALS algorithms, ASALSAN consists of conditional updates of $\mathbf{A}, \mathbf{R}$, and $\mathcal{D}$, until convergence. We omit the derivation of the update rules, which can be found in Bader et al. [2007].

**Practitioner's Guide for DEDICOM**
**Strengths**: Can model multirelational data that also exhibit asymmetries. The DEDICOM model is also unique.
**Weaknesses**: Restricted to multirelational data
**Utility** When modeling social network data that exhibit asymmetric relations, Knowledge Bases relations or Linked Data.

### 3.4. Hierarchical Tucker Decomposition (H-Tucker)

*3.4.1. Definition.* So far, we have mostly used three-mode tensors in our examples, mostly for ease of exhibition. However, in many real-world scenarios (as we will also see in Section 4), we have tensors of higher order. When the tensor order (i.e., the number of modes) increases, supposing that we would like to compute a Tucker model, the number of variables we need to estimate increases exponentially to the number of modes. For example, assuming an $I \times I \times I \times I$ four-mode tensor, its $(R, R, R, R)$ Tucker decomposition requires the computation of $R^4$ values for the core tensor $\mathcal{G}$.

This curse of dimensionality, however, can be avoided in a number of ways. The first way is the so-called Hierarchical Tucker Decomposition (H-Tucker) [Hackbusch and Kühn 2009; Grasedyck 2010; Kressner and Tobler 2012; Ballani et al. 2013].

The basic idea behind H-Tucker is the following: Suppose we have a binary tree of hierarchies of the modes of the tensor that can potentially be given to us by the application (e.g., see overview of Perros et al. [2015] in Section 4.7). Given this binary tree

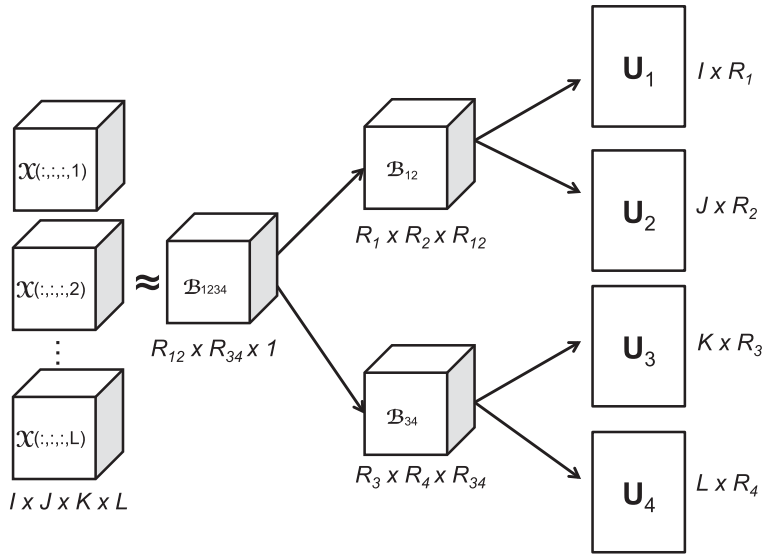


Fig. 7. The H-Tucker decomposition.

hierarchy, H-Tucker creates a set of generalized matricizations of the tensor according to each internal node of the tree. These matricizations are defined over a set of indices indicated by the particular node of the tree: for instance, given an $I \times J \times K \times L$ tensor, if node $t$ splits the modes into two disjoint sets $\{I, J\}$ and $\{K, L\}$, then the generalized matricization $\mathbf{X}_{(t)}$ will create an $IJ \times KL$ matrix where slices of the modes that are compacted into a single mode are stacked in a systematic fashion. Details on how this matricization is done can be found in Kressner and Tobler [2012].

For each internal node of the tree, it computes a "transfer" core tensor, which requires the estimation of much fewer values than in the Tucker case. The core tensor $\mathbf{B}_t$ is computed via

$$\mathbf{U}_t = \left(\mathbf{U}_{t_l} \otimes \mathbf{U}_{t_r}\right) \mathbf{B}_t,$$

where $\mathbf{B}_t$ is an $r_{t_l} r_{t_r} \times r_t$ matrix and $\mathbf{U}_t$ contains the $r_t$ left singular vectors of the $\mathcal{X}_{(t)}$ matricization. Finally, the leaf nodes contain the factor matrices that are similar to the ones that Tucker would give.

A pictorial example of a binary tree hierarchy and its corresponding H-Tucker decomposition is shown in Figure 7.

*3.4.2. Algorithms.* The algorithm for computing H-Tucker is described in Kressner and Tobler [2012], and here we provide an outline, in Algorithm 5.

**Practitioner's Guide for H-Tucker**
**Strengths**: Approximates well high-order tensors (with number of modes much larger than three) without suffering from the curse of dimensionality.
**Weaknesses**: Requires a priori knowledge of a binary tree of matricizations of the tensor.
**Utility** For very high-order tensors, especially when the application at hand provides an intuitive and natural hierarchy over the modes.

### 3.5. Tensor-Train Decomposition (TT)

*3.5.1. Definition.* Along the same lines as H-Tucker, there is the Tensor-Train decomposition proposed in Oseledets [2011], which tackles the curse of dimensionality in

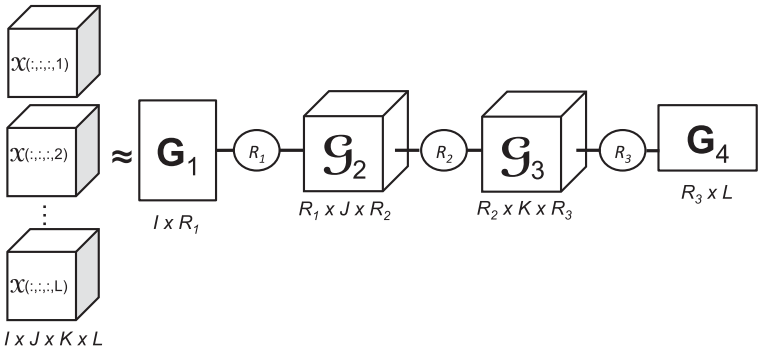


Fig. 8. Tensor-Train decomposition of a four-mode tensor. The circled variables indicate the reduced dimension that connects the core tensors.

**ALGORITHM 5:** H-Tucker Algorithm

**Input:** $N-mode$ tensor $\mathcal{X}$, ranks $R_1, \ldots, R_N$, and a binary tree $\mathcal{T}$ of the matricizations of $\mathcal{X}$.
**Output:** H-Tucker Decomposition $\mathbf{U}_1, \ldots \mathbf{U}_N$, $\mathcal{B}_t$, where $t \in N(\mathcal{T})$ (i.e., the nonleaf nodes of binary tree $\mathcal{T}$).

1: **for** $n = 1 \cdots N$ **do**
2: $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] \leftarrow \text{SVD}(\mathbf{X}_{(n)})$
3: $\mathbf{U}_n \leftarrow \mathbf{U}(:, 1 : R_n)$, i.e., set $\mathbf{U}_n$ equal to the $R_n$ left singular vectors of $\mathbf{X}_{(n)}$
4: **end for**
5: Starting from the root of tree $\mathcal{T}$, select a node $t$.
6: $t_l$ is the left child, $t_r$ is the right child.
7: $\mathbf{B}_t \leftarrow (\mathbf{U}_{t_l}^T \otimes \mathbf{U}_{t_r}^T)\mathbf{U}_t$
where $\mathbf{B}_t$ is an $r_{t_l} r_{t_r} \times r_t$ matrix, and $\mathbf{U}_t$ contains the $r_t$ left singular vectors of the $\mathcal{X}_{(t)}$ matricization.
8: Reshape $\mathbf{B}_t$ into an $r_{t_l} \times r_{t_r} \times r_t$ tensor.
9: Recurse on $t_l$ and $t_r$ until $t_l$ and $t_r$ are singletons.

very high-order tensors by imposing a parsimonious model. In contrast to H-Tucker, Tensor-Train does not require prior knowledge of a hierarchy of the modes. Tensor-Train decomposes a given tensor into a matrix, followed by a series of three-mode "transfer" core tensors (as in the case of the core tensors in H-Tucker), followed by a matrix. Each one of the core tensors is "connected" with its neighboring core tensor through a common reduced mode $R_i$. For a four-mode tensor, the Tensor-Train decomposition can be written as

$$\mathcal{X}(i, j, k, l) \approx \sum_{r_1, r_2, r_3} \mathbf{G}_1(i, r_1)\mathcal{G}_2(r_1, j, r_2)\mathcal{G}_3(r_2, k, r_3)\mathbf{G}_4(r_3, l).$$

A pictorial illustration of the four-mode Tensor-Train decomposition is shown in Figure 8. For the general $d$-mode case, we have

$$\mathcal{X}(i_1, \ldots, i_d) \approx \sum_{r_1, r_2 \cdots r_{d-1}} \mathbf{G}_1(i, r_1)\mathcal{G}_2(r_1, j, r_2) \cdots \mathcal{G}_{d-1}(r_{d-2}, d-1, r_{d-1})\mathbf{G}_d(r_{d-1}, d).$$

*3.5.2. Algorithms.* Algorithm 6 outlines the computation of the Tensor-Train decomposition, as introduced in Oseledets [2011]. Notice that the user is not required to explicitly define the reduced dimensions $R_i$; they are automatically calculated (line 7) by keeping the singular vectors that respect the approximation tolerance $\epsilon$ defined by the user. Of course, we can redefine the algorithm where instead of $\epsilon$, the user explicitly provides the $R_i$ dimensions if this is more appropriate for a given application. Also,

**ALGORITHM 6:** Tensor-Train Decomposition

**Input:** $N$-mode tensor $\mathcal{X}$ and approximation tolerance $\epsilon$.
**Output:** Tensor-Train decomposition $\mathbf{G}_1, \mathcal{G}_2 \cdots \mathcal{G}_{N-1} \cdots \mathbf{G}_N$.

Compute $\delta = \frac{\epsilon}{\sqrt{N-1}} \|\mathcal{X}\|_F$ as the truncation parameter.
$\mathbf{C} \leftarrow \mathcal{X}_{(1)}$ (select an arbitrary matricization of $\mathcal{X}$ for initialization).
$r_0 = 1$
**for** $k = 1 \cdots N$ **do**
  $\mathbf{C} \leftarrow \text{reshape}(\mathbf{C}, [r_{k-1} I_k, \frac{\text{numel}(\mathbf{C})}{r_{k-1} I_k}])$
  Compute a truncated SVD $[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}]$ of $\mathbf{C}$ such that the approximation error $e \leq \delta$.
  $r_k \leftarrow$ rank of the SVD that achieves the above approximation.
  $\mathcal{G}_k \leftarrow \text{reshape}(\mathbf{U}, [r_{k-1}, I_k, r_k])$
  $\mathbf{C} \leftarrow \mathbf{\Sigma}\mathbf{V}^T$
**end for**
$\mathbf{G}_N \leftarrow \mathbf{C}$

note that in Algorithm 6, for notational freedom, even if a tensor has one singleton dimension, we denote it as a tensor.

**Practitioner's Guide for Tensor-Train**
**Strengths**: Approximates well high-order tensors (with number of modes much larger than three) without suffering from the curse of dimensionality.
**Weaknesses**: Hard to interpret the core tensors, especially when they are dense.
**Utility** For approximating very-high order tensors and there is no prior hierarchical information on the modes.

### 3.6. Data Fusion and Coupled Matrix/Tensor Models

*3.6.1. Definition.* In a wide variety of applications, we have data that form a tensor and have side information or metadata that may form matrices or other tensors. For instance, suppose we have a (user, product, date) tensor that indicates which user purchased which product and when. Usually, stores also have some metadata on the user that can form a (user, features) matrix, as well as metadata on the products that form a (product, features) matrix. In this case, the tensor and the two matrices are *coupled* in the "user" and "product" mode, respectively, since there is a one-to-one correspondence of users in the tensor and the matrix (and accordingly for the products).

In Chemometrics, this concept was first introduced by Smilde et al. [2000] and Banerjee et al. [2007] apply this concept to data mining. There has been significant development of such a coupled model, either when matrices are coupled together [Singh and Gordon 2008; Acar et al. 2012] or when matrices and tensors are coupled (a tensor can be coupled with another matrix or even another tensor) [Lin et al. 2009; Zheng et al. 2010; Acar et al. 2011, 2012; Narita et al. 2012; Yokota et al. 2012; Wang et al. 2014; Ermiş et al. 2015]

One of the most popular models is the so-called Coupled Matrix-Tensor Factorization (CMTF) [Acar et al. 2011], where one or more matrices are coupled with a tensor. In the case where we have one coupled matrix in the "rows" mode, and we impose a CP model on the tensor, we have

$$\min_{\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r, \mathbf{d}_r} \left\| \mathcal{X} - \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \right\|_F^2 + \left\| \mathbf{Y} - \sum_{r=1}^{R} \mathbf{a}_r \mathbf{d}_r^T \right\|_F^2. \quad (5)$$

The CMTF decomposition with CP on the tensor consists of matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$. Notice that the factor matrix $\mathbf{A}$ corresponding to the rows of $\mathcal{X}$ and $\mathbf{Y}$ is the same. This

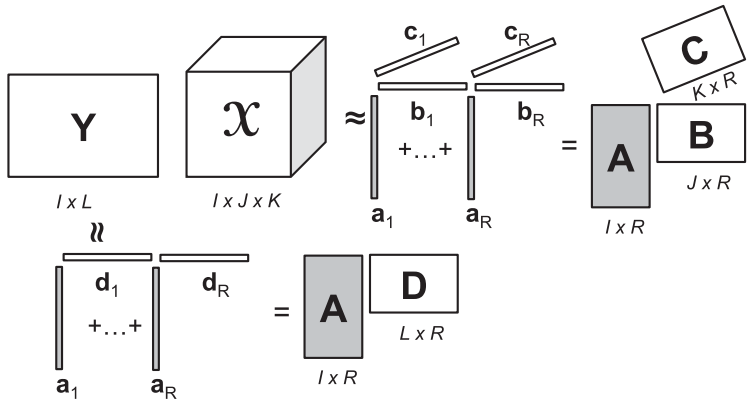


Fig. 9. Coupled Matrix-Tensor Factorization with a CP model on the tensor. Notice that the $\mathbf{a}_r$ vectors are the same between the tensor and the matrix factorization.

ensures that the two datasets are decomposed jointly and share the same latent space, effectively *fusing* the two datasets.

Instead of a CP model, we can have CMTF that assumes a Tucker model [Ermiş et al. 2015]:

$$\min_{\mathbf{U}_1,\mathbf{U}_2,\mathbf{U}_3,\mathcal{G},\mathbf{D}} \|\mathcal{X} - \mathcal{G} \times_3 \mathbf{U}_3 \times_2 \mathbf{U}_2 \times_1 \mathbf{U}_1\|_F^2 + \|\mathbf{Y} - \mathbf{U}_1\mathbf{D}^T\|_F^2. \tag{6}$$

An important issue when we have two or more heterogeneous pieces of data coupled with each other is the one we informally call "drowning," that is, when one of the datasets is vastly denser or larger than the other(s) and thus dominates the approximation error. In order to alleviate such problems, we have to weigh the different datasets accordingly [Wilderjans et al. 2009].

*3.6.2. Algorithms.* As in the case of CP, in CMTF we can also define an ALS algorithm, where we fix all but one of the matrices we are seeking to estimate. If, say, we seek to solve for $\mathbf{A}$, it turns out that we need to concatenate the two pieces of data, whose rows refer to matrix $\mathbf{A}$, that is, the matricized tensor $\mathbf{X}_{(1)}$ and matrix $\mathbf{Y}_1$, and we can then solve for $\mathbf{A}$ as

$$\mathbf{A} = \begin{bmatrix} \mathbf{X}_{(1)} \\ \mathbf{Y}_1 \end{bmatrix}^T \left( \begin{bmatrix} (\mathbf{B} \odot \mathbf{C}) \\ \mathbf{D} \end{bmatrix}^\dagger \right)^T.$$

Algorithm 7 shows the ALS algorithm for CMTF when a tensor $\mathcal{X}$ is coupled with three matrices $\mathbf{Y}_1, \mathbf{Y}_2, \mathbf{Y}_3$. We refer the user to Acar et al. [2011] for a gradient-based approach.

*3.6.3. Shared and Individual Components in Coupled Decompositions.* A fairly recent extension of the CMTF model has to do with shared and individual latent components between the datasets that are being jointly analyzed. For instance, in the coupled matrix-tensor case, we may assume that some of the latent components are exclusive to the tensor, some are exclusive to the matrix, and some of them are shared; this assumption gives relative freedom to the joint analysis to uncover a useful structure that is jointly present in both datasets, without falsely imposing structure from one dataset to the other that perhaps does not exist. Acar et al. [2013, 2014] introduce such flexible coupled decompositions. In a nutshell, they introduce distinct scalar weights on the components of the tensor and the matrix and they enforce a sparsity constraint on those weights, driving some of them to zero. A zero weight for, say, a component of

**ALGORITHM 7:** Alternating Least Squares (ALS) Algorithm for CMTF

**Input:** $\mathcal{X}$ of size $I \times J \times K$, matrices $\mathbf{Y}_i$, $i = 1 \cdots 3$, of size $I \times I_2$, $J \times J_2$, and $K \times K_2$, respectively, number of factors $F$.
**Output:** $\mathbf{A}$ of size $I \times F$, $\mathbf{B}$ of size $J \times F$, $\mathbf{C}$ of size $K \times F$, $\mathbf{D}$ of size $I_2 \times F$, $\mathbf{G}$ of size $J_2 \times F$, $\mathbf{E}$ of size $K_2 \times F$.

1: Initialize $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ using CP of $\mathcal{X}$. Initialize $\mathbf{D}$, $\mathbf{G}$, $\mathbf{E}$ as per the model (e.g., $\mathbf{D} = \mathbf{Y}_1(\mathbf{A}^\dagger)^T$).
2: **while** convergence criterion is not met **do**

$$3: \quad \mathbf{A} = \begin{bmatrix} \mathbf{X}_{(1)} \\ \mathbf{Y}_1 \end{bmatrix}^T \left( \begin{bmatrix} (\mathbf{B} \odot \mathbf{C}) \\ \mathbf{D} \end{bmatrix}^\dagger \right)^T$$

$$4: \quad \mathbf{B} = \begin{bmatrix} \mathbf{X}_{(2)} \\ \mathbf{Y}_2 \end{bmatrix}^T \left( \begin{bmatrix} (\mathbf{C} \odot \mathbf{A}) \\ \mathbf{G} \end{bmatrix}^\dagger \right)^T$$

$$5: \quad \mathbf{C} = \begin{bmatrix} \mathbf{X}_{(3)} \\ \mathbf{Y}_3 \end{bmatrix}^T \left( \begin{bmatrix} (\mathbf{A} \odot \mathbf{B}) \\ \mathbf{E} \end{bmatrix}^\dagger \right)^T$$

6: $\mathbf{D} = \mathbf{Y}_1(\mathbf{A}^\dagger)^T$, $\mathbf{G} = \mathbf{Y}_2(\mathbf{B}^\dagger)^T$, $\mathbf{E} = \mathbf{Y}_3(\mathbf{C}^\dagger)^T$
7: **end while**

the tensor indicates that this component is not present in the tensor and is individual to the matrix.

**Practitioner's Guide for Coupled Models**
**Strengths**: Jointly analyzes heterogeneous datasets that have one of their modes in common, and incorporates side information and metadata to the analysis.
**Weaknesses**: When the heterogeneous datasets are vastly different in terms of size and volume, we need to apply appropriate weighting in order to avoid one dataset "drowning" the rest.
**Utility** In applications where metadata are present and can be modeled as side information matrices, as well as when having two (or more) heterogeneous pieces of data that refer to the same physical entities.

### 3.7. PARAFAC2 and Decomposition of Multiset Data

*3.7.1. Definition.* Closely related to the coupled datasets that we talked about in the previous section is the concept of "multiset" data, which, however, merits its own discussion because of its prevalence. A multiset dataset is a collection of matrices $\{\mathbf{X}_k\}$ for $k = 1 \cdots K$ that have one mode in common. These matrices can be seen as nearly forming a tensor; however, the nonshared mode has different dimensions, and thus it has to be handled carefully. The PARAFAC2 decomposition, introduced in Harshman [1972], is specifically designed for such cases. Given a multiset dataset $\{\mathbf{X}_k\}$, where the matrices share the columns but have different numbers of rows, PARAFAC2 decomposes each matrix in the multiset as

$$\mathbf{X}_k \approx \mathbf{U}_k \mathbf{H} \mathbf{S}_k \mathbf{V}^T.$$

PARAFAC2 acknowledges the differences in the rows, by introducing a set of $\mathbf{U}_k$ row factor matrices, but imposes a joint latent structure on the columns and the third mode, similar to CP. A pictorial representation of PARAFAC2 is shown in Figure 10.

*3.7.2. Algorithms.* As in many tensor decompositions, the most popular algorithm for PARAFAC2 is based on ALS [Kiers 1993]. In Algorithm 8, we present an improvement upon the algorithm of Kiers [1993], which is proposed by Chew et al. [2007]

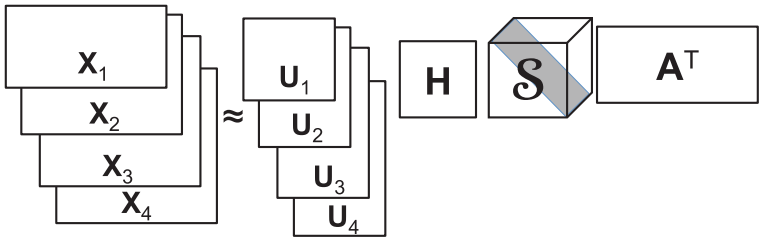


Fig. 10. The PARAFAC2 decomposition.

**ALGORITHM 8:** ALS Algorithm for PARAFAC2

**Input:** Multiset $\{\mathbf{X}_k\}$ for $k = 1 : K$ and rank $R$.
**Output:** PARAFAC2 Decomposition of $\{\mathbf{X}_k\}$: $\{\mathbf{U}_k\}$, $\mathbf{H}$, $\mathcal{S}$, $\mathbf{V}$.

Initialize:
  $\mathbf{V} \leftarrow R$ principal eigenvectors of $\sum_{k=1}^{K} \mathbf{X}_k^T \mathbf{X}_k$
  $\mathbf{H} \leftarrow \mathbf{I}$
**for** $k = 1 \cdots K$ **do**
  $\mathcal{S}(:, :, k) \leftarrow \mathbf{I}$, for $k = 1 \cdots K$.
**end for**
**while** convergence criterion is not met **do**
  **for** $k = 1 \cdots K$ **do**
    $[\mathbf{P}_k, \mathbf{\Sigma}_k, \mathbf{Q}_k] \leftarrow$ truncated SVD of $\mathbf{H}\mathbf{S}_k\mathbf{V}^T\mathbf{X}_k^T$ at rank $R$
    $\mathbf{U}_k \leftarrow \mathbf{Q}_k\mathbf{P}_k^T$
  **end for**
  **for** $k = 1 \cdots K$ **do**
    Compute $\mathcal{Y}(:, :, k) = \mathbf{U}_k^T\mathbf{X}_k$
  **end for**
  Run a single iteration of CP ALS (Algorithm 1) on $\mathcal{Y}$ and compute factors $\mathbf{H}$, $\mathbf{V}$, $\hat{\mathbf{S}}$.
  **for** $k = 1 \cdots K$ **do**
    $\mathcal{S}(:, :, k) \leftarrow Diag(\hat{\mathbf{S}}(k, :))$
  **end for**
**end while**

**Practitioner's Guide for PARAFAC2**
**Strengths**: Can jointly analyze heterogeneous pieces of data that cannot be expressed as a tensor.
**Utility** When we have a set of matrices that nearly form a tensor, but they do not match one of the modes.

### 3.8. Model Order Selection

An important issue in exploratory data mining using tensors is selecting the order of the model, in other words the rank of the decomposition in models such as CP, or the dimensions of the core tensor in Tucker-based models. This is generally a very hard problem, especially in the presence of noisy data; however, there exist heuristic methods that work well in practice.

*3.8.1. CP.* For the CP decomposition, considerations about the quality of the decomposition date as early as 1984 where Harshman [1984] outlines strategies that can reveal the quality of the decomposition. The most intuitive and popular heuristic for model order selection is the so-called Core Consistency Diagnostic or CORCONDIA [Bro 1998;

Bro and Kiers 2003]. CORCONDIA is based on the observation we highlighted in Section 3.2 and Figure 4, observing that CP can be written as a restricted Tucker model with a super-diagonal core. Based on that observation, CORCONDIA, given a tensor $\mathcal{X}$ and its CP decomposition $\mathbf{A}, \mathbf{B}, \mathbf{C}$, computes a Tucker model where $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are the factors (which are known) and $\mathcal{G}$ is the core tensor (which is unknown). The core tensor $\mathcal{G}$ holds important information: if it is exactly or nearly super-diagonal, then $\mathbf{A}, \mathbf{B}, \mathbf{C}$ is a "good" CP decomposition (hence the chosen model order is correct); otherwise, there is some problem either with the chosen rank or with the data lacking a multilinear structure. After computing $\mathcal{G}$, the Core Consistency diagnostic can be computed as

$$c = 100 \left(1 - \frac{\sum_{i=1}^{F} \sum_{j=1}^{F} \sum_{k=1}^{F} (\mathcal{G}(i,j,k) - \mathcal{I}(i,j,k))^2}{F}\right),$$

where $\mathcal{I}$ is a super-diagonal tensor with ones on the $(i,i,i)$ entries. For a perfectly super-diagonal $\mathcal{G}$ (i.e., perfect modeling), $c$ will be 100. For rank-one models, the metric will always be 100, because the rank-one component can produce a single scalar, which is a trivial super-diagonal core. Therefore, CORCONDIA can be used for ranks higher than 2. da Costa et al. [2008] extend the aforementioned idea, making it more robust in the presence of noise. For big tensor data, naive computation of CORCONDIA cannot scale. However, recently, Papalexakis and Faloutsos [2015] proposed a scalable algorithm for computing the diagnostic, especially for big sparse tensors.

In addition to CORCONDIA, various other methods have been proposed, ranging from Minimum Description Length (MDL) [Araujo et al. 2014; Metzler and Miettinen 2015] to Bayesian [Mørup and Hansen 2009; Zhao et al. 2015].

*3.8.2. Tucker.* For the Tucker decomposition, one of the first methods for determining the model order (i.e., the size of the core tensor) is the so-called DIFFIT (DIFFerence in FIT) method [Timmerman and Kiers 2000]. In a nutshell, DIFFIT tries different combinations of the dimensions $R_1, R_2, R_3$ of the core tensor, such that $R_1 + R_2 + R_3 = s$ for various values of $s \geq 3$. For each given $s$, DIFFIT finds the combination of dimensions that gives the best-fit complexity (as measured by the number of fitted parameters) tradeoff. Subsequently, DIFFIT compares different values of $s$, choosing the one $s_c = R_1 + R_2 + R_3$ that yields the best fit. DIFFIT requires the computation of multiple Tucker decompositions, a fact that may slow down the estimation of the model order. Kiers and Kinderen [2003] propose a method that computes a single Tucker decomposition and is shown to perform comparably to DIFFIT. Finally, as in the CP case, Mørup and Hansen [2009] can also estimate the model order of Tucker employing a Bayesian framework.

## 4. DATA MINING APPLICATIONS

Tensors are very powerful and versatile tools, as demonstrated by the long list of their applications in data mining. In this section, we cover a wide spectrum of such applications: social and collaboration network analysis, web mining and web search, knowledge bases, information retrieval, topic modeling, brain data analysis, recommendation systems, urban computing, healthcare and medical applications, computer networks, speech and image processing, and computer vision. For each application, we focus on what the problem formulation is, how a tensor is modeled, and which decomposition is used, and discuss the results.

### 4.1. Social and Collaboration Network Analysis

Social and collaboration network analysis can benefit from modeling data as tensors in various ways: when there exist multiple "views" of the network (e.g., who calls

whom, who texts whom and so on), this can be expressed as a three-mode tensor with each frontal slice being an adjacency matrix of the network for a particular view. Furthermore, tensors have been used in modeling time-evolving networks, where each frontal slice of the tensor is a snapshot of the network for a particular point in time.

Tensor applications in social networks date back to Acar et al. [2005], which is also one of the earliest tensor applications in data mining. In this particular work, the authors analyze IRC chat-room data and create a synthetic tensor data generator that mimics the properties of real chat-room data. The tensors are of the form (user, keyword, time), and for the purposes of demonstrating the utility of expressing the data as a tensor, the authors also create (user, keyword) and (user, time) matrices. Subsequently, they compare SVD (on the matrices), and Tucker and CP for noiseless and noisy data; Tucker seems to outperform the rest of the tools for noisy data because of its flexibility to have a different number of latent components per mode. The paper also touches upon two very important issues: (1) temporal granularity of the data, indicating cases where the analysis breaks down due to inappropriate resolution in the "time" mode, and (2) model order selection, highlighting that using solely the approximation error as a guide to select the number of components does not necessarily imply better interpretation of the data.

Bader et al. [2007] analyze the Enron email exchange data, creating an (employee, employee, month) tensor, recording the number of emails sent from one employee to another. Given the asymmetry of the relation, the authors use DEDICOM to decompose the tensor and identify *latent roles* for the employees (with the roles discovered being labeled as "Executive," "Government Affairs," "Legal," "Pipeline"), as well as the communication pattern between roles. The strength of DEDICOM is its ability to discover asymmetric communication patters (e.g., executives tend to receive more emails than they send).

Kolda and Sun [2008] model the DBLP collaboration network as a tensor of (author, keyword, conference) and use the Tucker decomposition to identify groups of authors who publish on similar topics and on similar conferences.

In addition to the social interactions, Lin et al. [2009] demonstrate that using the context behind those interactions can improve the accuracy in discovering communities. In this work, the authors define a "Metagraph," a graph that encodes the context of social interactions, and subsequently propose a Metagraph Factorization, which essentially boils down to a coupled tensor and matrix factorization, involving all the various tensors and matrices that capture relations in the Metagraph. They apply their algorithm to a dataset from Digg, where they record six relations in the data. They demonstrate that their proposed technique outperforms simple approaches that use the interaction frequency, as well as tensor methods that ignore the context of the interaction.

Dunlavy et al. [2011] consider the problem of *temporal link prediction*, that is, predicting future links based on past data. The problem is formulated as a CP decomposition of graphs over time. The tensor-based techniques are especially effective in terms of predicting periodic behaviors in the links. See also Acar et al. [2009].

Papalexakis et al. [2012, 2013] apply the CP decomposition to the Enron email exchange data, again forming an (employee, employee, month) tensor. They identify cliques of interaction, which concur with the roles identified by Bader et al. [2007]. Furthermore, Papalexakis et al. [2012] apply the CP decomposition to a Facebook Wall posts database that forms a (wall owner, wall poster, day) tensor and identify interesting patterns and their temporal evolution.

Anandkumar et al. [2014] show that orthogonal CP can be used to estimate latent factor models such as Latent Dirichlet Allocation (LDA) and Mixed Membership Block

Models (MMBMs) using the method of moments. Huang et al. [2013] use orthogonal CP for recovering a Mixed Membership Block Model that detects overlapping communities in social networks and apply it to a Facebook social network dataset (two-mode). This is a very interesting alternative perspective because the authors use the tensor decomposition to identify a model on a two-mode graph and do not use the tensor decomposition on the graph itself.

Papalexakis et al. [2013] introduce a CP-based community detection algorithm in multiview social networks. They analyze two different datasets: (1) "Reality Mining," a multiview social network with four views: phone call, text, Bluetooth (indicating proximity), and physical friendship, and (2) a network of researchers in DBLP having three views: coauthor, using same keywords in publications, and citation of each other's work. The algorithm is able to identify communities with better accuracy than approaches that do not use all the views or do not exploit the higher-order structure of the data.

Araujo et al. [2014] use a CP-based decomposition to identify communities in time-evolving graphs. They use Minimum Description Language (MDL) for identifying the number of communities that can be extracted from the data. They apply the algorithm to a dataset of phone calls over time for a very large city and identify various patterns of communities. Among the most frequent patterns were (1) "Flickering Stars," which are star-like communication patterns that have a varying number of receivers over time, and (2) "Temporal Bipartite Cores," which are near-bipartite cores of people communicating with each other over time.

Jiang et al. [2014] model user behavior over time and have two case studies: (1) behavior of academic researchers and (2) behavior of "mentions" in tweets. In order to identify user behavior over time, the authors model the problem as the decomposition of a series of tensors, each one for a particular point in time. For instance, for the tweets data, we have a tensor of (source user, target user, keyword), for each time tick. The authors propose to decompose each one of those tensors using a Tucker-like model and impose regularization to tackle sparsity. Furthermore, in order to reduce the computational complexity, the authors propose an algorithm that carries out the decomposition incrementally, using the model from the previous time point to estimate the new one. They showcase their algorithm in datasets from Microsoft Academic Search and Weibo, identifying interesting user behavior patterns over time.

Schein et al. [2015] use dyadic events between countries in order to discover multilateral relations among them. In particular, they propose a Bayesian version of the CP decomposition, postulating a Poisson distribution on the data, which has been shown to be more effective when dealing with sparse, count data. They apply their proposed decomposition to a four-mode tensor of (country A, country B, event type, timestamp) and evaluate the effectiveness of their approach by identifying well-documented multilateral country relations in recent history.

### 4.2. Web Mining and Web Search

Kolda et al. [2005] extend Kleinberg's HITS algorithm for authoritativeness and hubness scores of webpages, including context information on the link. In particular, for each link, they use the anchor text as the context. This creates a three-mode tensor of (webpage, webpage, anchor text), and the CP decomposition gives the authoritativeness and hubness scores (**A** denotes the authorities and **B** the hubs, and **C** encodes the topic of each set of hubs and authorities). This, in contrast to plain HITS (which can be seen as an SVD of the hyperlink matrix), provides more intuitive interpretation. The authors demonstrate the superiority of the approach in identifying topically coherent groups of webpages by applying it to a custom crawl of the web, emulating what a commercial search engine does.

Sun et al. [2005] personalize web search by using historic click-through data of users. They construct a (user, query, page) tensor that records the clicks of a user to a particular result of a query, and they use HOSVD to take a low-rank factorization of the click-through tensor. By reconstructing the tensor from its HOSVD, they fill in missing values, which can then be used as personalized result recommendations for a particular user.

Agrawal et al. [2015a] model the comparison between the results of different search engines using tensors. For a set of queries, they create a (query, keyword, date, search engine) tensor and use the CP decomposition to create latent representations of search engines in the same space. They apply this tool to compare Google and Bing web search and find that for popular queries, the two search engines have high overlap. Subsequently, Agrawal et al. [2015b] apply the same methodology to compare Google and Twitter-based search, finding that the overlap is lower, showing the potential for social-media-based web search.

### 4.3. Knowledge Bases, Information Retrieval, and Topic Modeling

Chew et al. [2007] tackle the problem of Cross-language Information Retrieval, where we have parallel documents in different languages and we need to identify latent topics as in Latent Semantic Analysis (LSA) [Deerwester et al. 1990]; in a nutshell, LSA refers to taking a low-rank Singular Value Decomposition of a (term, document) matrix and exploiting the low-rank structure to identify synonyms. Doing simply LSA on the concatenated set of terms for all languages and documents ignores the fact that parallel documents have the same structure and ought to be clustered similarly to latent topics. To that end, the authors use the PARAFAC2 decomposition, which is applied to multiset data on a set of (term, document) matrices (terms can be different from language to language; that's why we don't have a tensor). The authors demonstrate the superiority of their approach by applying it on a set of translations of the Bible and achieving better performance than traditional LSA.

Kang et al. [2012] and Papalexakis et al. [2012] apply the CP decomposition to data coming from the Read the Web [RTW 2016] project. In particular, the data are in the form (noun-phrase, noun-phrase, context-phrase). Using the CP decomposition, the authors identify latent topics that are semantically and contextually coherent, coming from each one of the rank-one components of the decomposition. Furthermore, the factor matrices of the CP decomposition define latent embeddings of the noun-phrases to a concept space; the authors find similar noun-phrases in that concept space, which are essentially "contextual synonyms," for example, noun-phrases that can be used in the same semantics/context.

Jeon et al. [2015] apply the Tucker decomposition to a particular snapshot of the Freebase knowledge base that contains entities and relations about music. The authors use the factor matrices of Tucker as latent embeddings (as in the case of the CP factor matrices) and identify semantically coherent latent concept entities and relations (e.g., "Classic Music" and "Pop/Rock Music").

Nickel et al. [2012] model semantic data of type (entity1, entity2, relation) as a three-mode tensor. They analyze it using the RESCAL model, whose advantage is that it captures attribute and relationship correlations across multiple interconnections of entities and relations. They apply it to the YAGO [Suchanek et al. 2007] Knowledge Base and show better performance in (1) predicting unknown triples, (2) collective learning (defined as "automatic exploitation of attribute and relationship correlations across multiple interconnections of entities and relations"), and (3) learning taxonomies on the knowledge base.

A higher-order extension of LSA can be found in Chang et al. [2013], where the authors integrate multiple relations between words from different information sources.

They essentially use a Tucker decomposition and they absorb the $\mathbf{W}$ factor matrix of Figure 3 by multiplying it with the core tensor $\mathcal{G}$, creating a different core tensor $\mathcal{S}$, showing the analogy of this Tucker representation and the SVD (where now they have two factor matrices and one core tensor). The authors demonstrate the performance of their proposed LSA extension by identifying antonyms and is-a relations more accurately than the state of the art.

Following the examples of Nickel et al. [2012] and Chang et al. [2014] use a RESCAL-inspired decomposition to model and analyze knowledge base data. The new additions to the model are type constraints: each entity of a knowledge base has a known type (e.g., "person"); therefore, these type constraints are explicitly included in the model by excluding triples of entity-relation-entity with incompatible types from the optimization. This both saves computation and produces a more accurate result.

Finally, in a similar spirit to Huang et al. [2013], Anandkumar et al. [2014], and Huang et al. [2014] use the CP decomposition to compute the LDA [Blei et al. 2003] via the method of moments. In addition to showing how LDA is computed using CP, the authors provide a distributed algorithm on the distributed platform REEF.

### 4.4. Brain Data Analysis

Acar et al. [2007] analyze electroencephalogram (EEG) data from patients with epilepsy in order to localize the origin of the seizure. To that end, they model the EEG data using a three-mode (time samples, scales, electrodes) tensor (after preprocessing the EEG measurements via a wavelet transformation). In order to analyze the EEG tensor, they use the CP decomposition: when they identify a potential seizure (which has signatures on the time and frequency domains), they use the factor vector of the third mode (the "electrodes" mode) to localize that activity. In some cases, the data contain artifacts that may shadow the seizures from the CP decomposition (such as activity caused by the movement of the eyes) and therefore have to be removed. In order to remove those artifacts, the authors use the Tucker3 decomposition, which can capture the subspace variation for each mode better than CP (due to its increased degrees of freedom, which in turn make it harder to interpret). They identify the artifacts in the Tucker components and they use them to remove the artifacts from the data.

When dealing with brain measurements such as EEG or functional magnetic resonance imaging (fMRI), usually researchers average data from multiple trials; EEG measures different electrodes on the brain and fMRI measures 3D pixels (or voxels in the literature). If we have access to the measurement data from different trials over time, instead of averaging, we can model the measurements as a three-mode tensor (voxel, time, trial). In brain measurements, it is not unusual for time shifts to happen, due to physical phenomena in the brain. If we analyze the tensor without accounting for those shifts, the solution may be degenerate and therefore not very useful to interpret. To solve that, Mørup et al. [2008], following up on Harshman et al. [2003], propose a modification to the CP model, the shift invariant CP (Shift-CP), which takes the time shift into account. The model is

$$\mathcal{X}(i, j, k) = \sum_{r=1}^{R} \mathbf{A}(i, r)\mathbf{B}(j - \boldsymbol{\tau}(k), r)\mathbf{C}(k, r),$$

where $\boldsymbol{\tau}$ is a vector of time shifts that is also learned during the optimization. Mørup et al. [2008] show that this can recover the latent components of fMRI and EEG data more accurately, avoiding degenerate solutions.

A generalization of the aforementioned Shift-CP model is given in Mørup et al. [2011], where the authors propose the Convolutional CP (Conv-CP) model. The idea behind that model is that, unlike Shift-CP, which allows for a single time delay per

trial, Conv-CP can accommodate an arbitrary number of such delays per trial, within the length of the convolutive filter used ($T$). The Conv-CP decomposition is

$$\mathcal{X}(i, j, k) = \sum_{r=1}^{R} \sum_{\tau=1}^{T} \mathbf{A}(i, r)\mathbf{B}(j - \tau, r)\mathcal{C}(k, r, \tau).$$

In this case, $\mathcal{C}$, which serves as the convolutive filter for each trial, is a tensor. The authors show that for both simulated and real data, Conv-CP outperforms Shift-CP since it is a more general and flexible model.

Davidson et al. [2013] have fMRI measurements over time and wish to estimate regions of the brain and the connections between them. They model the data as a tensor (x-coordinates, y-coordinates, time) and assume that using a CP decomposition, each rank-one tensor gives a particular region of the brain (the first two modes in space and the third in time). In order to guide the decomposition to find the right structure in the brain, they use linear constraints for the spatial factors of the decomposition according to groups of voxels in the brain that are known to behave in a coherent manner. Applying those constraints, the authors are able to detect nodes and the network between those nodes with higher accuracy.

He et al. [2014] extend supervised learning models such as Support Vector Machines to operate on tensors as opposed to vectors or points. They leverage the fact that data such as fMRI brain scans have an inherent tensor structure that should be exploited and propose a Kernel SVM that uses the CP decomposition of the data points as a compact representation that preserves the structure. They apply their approach in classifying fMRI brain scans from patients with Alzheimer's disease, ADHD, and brain damage due to HIV, and demonstrate the effectiveness of exploiting the higher-order structure of the data rather than ignoring it.

Finally, Papalexakis et al. [2014] seek to identify coherent regions of the brain, among different individuals, that exhibit activity for groups of semantically similar stimuli. They use fMRI data from nine different people when shown 60 different simple English nouns (e.g., "dog," "airplane," "hammer") forming a (noun, voxel, person) tensor. They also use semantic features for those same nouns, represented by a (noun, feature) matrix, and they use Coupled Matrix-Tensor Factorization to identify latent clusters of nouns, voxels, people, and noun features. In an unsupervised way, they identify sets of semantically similar nouns that activate coherent regions of the brain such as the premotor cortex, which is activated when holding small things or picking things up.

### 4.5. Recommendation Systems

One of the first attempts to apply tensors to collaborative filtering and recommendation systems is Xiong et al. [2010]. The authors propose to extend Bayesian Probabilistic Matrix Factorization (BPMF) [Salakhutdinov and Mnih 2008], which is widely used in Recommendation Systems in the case where we have temporal information. They propose a Bayesian Probabilistic Tensor Factorization (BPTF) that is based on the CP model. In experimental evaluation, they show that BPTF outperforms BPMF, demonstrating that using temporal information and exploiting the higher-order structure it induces on the data prove beneficial for recommendation.

In a similar spirit as earlier and around the same time, Karatzoglou et al. [2010] propose to use context (such as time) in traditional user-item recommendation scenarios by modeling the data as a tensor. The difference is that they use HOSVD to take a low-rank decomposition of the data (only on the observed values) and use the reconstructed values for the missing values. The method beats Matrix Factorization techniques as well as other Context-Aware Techniques that may (partially) ignore the higher-order structure of the data that the tensor decomposition exploits.

Rendle and Schmidt-Thieme [2010] propose the Pairwise Interaction Tensor Factorization (PITF) model as a method for tag recommendation on the web. The scenario is as follows: users tag items (webpages, pictures, products, etc.) over time, and we would like to recommend new items they would like to tag. The proposed PITF model is a special case of both Tucker and CP. It explicitly models the pairwise interactions between users, tags, and items. In order to follow the original paper's notation, we rename the $\mathbf{A}, \mathbf{B}, \mathbf{C}$ CP factor matrices to $\mathbf{U}, \mathbf{I}, \mathbf{T}$, corresponding to the users, items, and tags, respectively. To model the pairwise interactions, we divide those factor vectors as $\mathbf{U} = [\mathbf{U}^{(I)} \ \mathbf{U}^{(T)}]$, where each one of the column blocks of $\mathbf{U}$ interacts with the corresponding columns of $\mathbf{I}$ and $\mathbf{T}$. Then, the PITF model is defined as

$$\mathcal{X}(u,i,t) = \sum_{r=1}^{R} \mathbf{U}^{(T)}(u,r)\mathbf{T}^{(U)}(t,r) + \sum_{r=1}^{R} \mathbf{I}^{(T)}(i,r)\mathbf{T}^{(I)}(t,r) + \sum_{r=1}^{R} \mathbf{U}^{(I)}(u,r)\mathbf{I}^{(U)}(i,r).$$

The authors evaluate the proposed model in comparison to CP and Tucker, where it obtains higher prediction accuracy faster. Furthermore, PITF won the ECML/PKDD Discovery Challenge 2009.

Rendle [2010] introduces Factorization Machines, a generalization of Support Vector Machines that parameterizes the data internally using a factorization model instead of using the raw data. The Factorization Machine can have degree 2 or higher. In the case of degree 2, the internal factorization is a bilinear matrix factorization, whereas for higher degrees, it uses a CP model. Factorization Machines combine the benefits of SVMs and Factorization Models, especially in scenarios of highly sparse data (such as in Collaborative Filtering) where simple SVM fails. In experiments, the author shows that Factorization Machines achieve the same recommendation quality as the PITF method described earlier.

Peng et al. [2010] work on a similar social tagging scenario to Rendle and Schmidt-Thieme [2010] where they design a tag recommendation system on a (user, item, tag) tensor. In this paper, the authors propose an HOSVD-based dimensionality reduction. They compare their proposed recommendation scheme against methods that do not fully exploit the inherent higher-order structure and demonstrate better performance.

Zheng et al. [2010, 2012] attack the problem of location-based activity recommendation to users, using side information for users, activities, and locations. In particular, the data in the problem include a tensor (user, location, activity), a (user, user) matrix of user similarities in a social network, a (location, feature), a (user, location) matrix that encodes information of a user being present at a location without a specified activity, and an (activity, activity) matrix containing the activities' similarities. The authors propose to jointly decompose these datasets using a flavor of Coupled Matrix-Tensor Factorization, imposing additional proximity constraints of the involved entities in the latent space. They show that by integrating all these pieces of auxiliary information, they outperform recommendation systems that include a part of these additional sources or none of them.

Finally, Pantraki and Kotropoulos [2015] work on image and tag recommendation on Flickr. They model the data as a multiset where we have three matrices: (image, feature), (image, tag keyword), and (image, user). Since the data do not strictly form a tensor, the authors apply PARAFAC2 to this dataset and demonstrate its ability to obtain a joint low-rank representation of this dataset, which can provide high-quality image and tag recommendations.

### 4.6. Urban Computing

Urban Computing refers to a class of applications that study human activity and mobility within a city, with the ultimate goal to improve the livability of an urban environment.

Mu et al. [2011] use historical data from a metropolitan area in order to forecast areas with potential future crime activity. They model the data as a four-mode tensor, where the first three modes are (longitude, latitude, time) and the fourth mode consists of features such as residential burglary information, social events, and offender data. They use a form of Tucker decomposition to obtain a low-rank representation of that tensor, and they use that representation for linear discriminant analysis to predict future crime activity.

In Wang et al. [2014], the problem the authors solve is the one of estimating the travel time of a particular trajectory in a city road network. In order to do so, they use real GPS data of travel times by a set of drivers. The issue that arises with these types of data is the high degree of sparsity, since many road segments may have not been traversed at all (or sometimes ever); thus, trying to estimate travel times for all road segments this way may result in inaccurate measurements. To alleviate this data sparsity, the authors propose to use historical data for those GPS trajectories, as well as side information about time slots and road segments, to fill in missing travel time values. In the heart of their proposed method lies a Coupled Matrix Tensor Factorization: they form a (road segment, driver, time slot) tensor, with each entry containing the travel time that a driver did on a particular road segment during a particular time slot; a (time slot, time slot) matrix capturing the similarities between time slots; and a (road segment, geographical features) matrix, providing additional information for the road segments. Interestingly, the proposed Coupled Matrix Tensor Factorization here imposes a Tucker model on the tensor part (as opposed to the CP model shown in Figure 9), and this is because the primary purpose of this low-rank decomposition is to complete missing values, where Tucker can capture more nontrilinear variation in the data. Using this tensor decomposition scheme to enrich the data, the authors show that their method outperforms state-of-the-art baselines.

Zheng et al. [2014] analyze noise complaint data from New York City in order to identify the major sources of noise pollution in the city for different times of the day and the week. The issue with the human-generated complaints is that they result in very sparse data where some regions are overrepresented and some are underrepresented or not present at all. In a similar spirit as Wang et al. [2014], in order to overcome the data sparsity problem, the authors form a tensor out of the noise complaint data with modes (region, noise category, time slot) and couple it with matrices with additional information for regions, noise categories, and time slots. Subsequently, they decompose it using a Coupled Matrix Tensor Factorization with a Tucker model on the tensor and complete missing values of noise information in areas and time slots with few or no complaints at all. The resulting dataset is an accurate representation of New York City's noise signature.

Finally, the work of Zhang et al. [2015] addresses the problem of exploring, analyzing, and estimating drivers' refueling behavior in an urban setting for better planning (e.g., placement of gas stations) and recommendation of nearby gas stations with minimal wait time. As before, the problem with the existing data created by volunteer drivers is sparsity, and the authors tackle it through low-rank factorization and reconstruction of a (gas station, hour, day) tensor. In particular, they propose a flavor of HOSVD decomposition, which also incorporates context features including features of the gas station and the weather. The proposed HOSVD extension is

$$\mathcal{X}(i,j,k) = \sum_{r_1,r_2,r_3} \mathbf{G}(i,r_1)\mathbf{H}(j,r_2)\mathbf{D}(k,r_3)\mathcal{S}(r_1,r_2,r_3) + \sum_{l=1}^{L} \mathbf{B}(l,c_l),$$

where $\mathbf{B}(l,c_l)$ captures the effect of feature $l$ under condition $c_l$ in the reconstructed tensor.

### 4.7. Healthcare and Medical Applications

Ho et al. [2014] use a tensor-based technique to automatically derive phenotype candidates from electronic health records. Intuitively, the problem is to automatically identify groups of patients who have similar diagnoses and have undergone similar procedures. In order to do that, they propose a tensor decomposition based on the CP model, where each phenotype candidate is a rank-one component of the CP decomposition of a (patient, diagnosis, procedure) tensor. Interestingly, the proposed decomposition, MARBLE, has a twist from the traditional CP model. In addition to the sum of rank-one tensors, the proposed model also contains a rank-one "bias" tensor, as shown in the following equation:

$$\mathcal{X} = \left( \sum_{r=1}^{R} \lambda_r \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r \right) + \left( \mathbf{u}^{(1)} \circ \mathbf{u}^{(2)} \circ \mathbf{u}^{(3)} \right).$$

In this model, the $\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r$ factors are constrained to be sparse, and the bias factor vectors are normally dense. In that sense, this model can be seen as an $(R+1)$ rank CP model, where some of the columns of the factor matrices are constrained. The role of the rank-one bias tensor is to capture global and constant variation in the data that is not specific to a particular phenotype. Having this rank-one bias tensor proves instrumental in identifying good local structure in the data. In addition to the proposed model, Ho et al. [2014] also propose to fit this model under a KL-divergence loss and impose nonnegativity constraints, resulting in sparse and nonnegative factors that are easier to interpret.

The work of Perros et al. [2015] is the first data mining application of the Hierarchical Tucker (H-Tucker) model of 3.4. In particular, the authors propose a sparse version of H-Tucker (which is presented in more detail in Section 5.3) and apply it to a disease phenotyping problem, much like the one addressed by Ho et al. [2014]. The difference is that here, the authors use co-occurrence data of patients exhibiting the same disease within a large database of medical records. The need for using H-Tucker, which handles tensors of very high order more gently than Tucker or CP, is because the number of disease types for which the authors compute co-occurrences is as high as 18 (which is the top-level number of disease types as defined by the International Classification of Diseases hierarchy). Thus, the tensor they operate on is as high order as 18-mode. Using H-Tucker and interpreting the factor matrices on the leaves, the authors are able to identify meaningful phenotypes for diseases, concurring with medical experts.

Mohammadi et al. [2016] consider the problem of network alignment with a goal of preserving triangles across the aligned graphs. This can be expressed as a tensor eigenvalue problem, and the method of Kolda and Mayo [2011] can be used to solve it. This has applications in comparative interactomics in biology. For instance, the problem of aligning the human and yeast interactomes is considered, and the tensor-based method outperforms competitors in terms of quality. Although this is considered in the context of biological network alignment, it is a general-purpose method that can be used in other contexts.

### 4.8. Computer Networks

Maruhashi et al. [2011] use the CP decomposition to analyze network traffic data from Lawrence Berkeley National Labs (LBNL) forming a tensor of (source IP, destination IP, port #, timestamp) where each entry indicates a connection between the two IP addresses on a given port for a given time tick. Using the factor matrix corresponding to the "timestamp" mode, the authors propose a spike detection algorithm on the temporal profile of the latent components of the decomposition, identifying anomalous connections in the data.

Subsequently, Papalexakis et al. [2012] analyze the same LBNL dataset as in Maruhashi et al. [2011], identifying components of normal activity, as well as anomalous components, such as one that indicates a port scanning network attack (where the attacker serially probes a wide range of ports in a machine, aiming to discover potential security holes).

Finally, Mao et al. [2014] analyze two types of network data: (1) Honeynet Data of (source IP, destination IP, timestamp) and (2) Intrusion Detection System (IDS) logs of (event type, timestamp, target IP). They apply the CP decomposition, and using clustering methods on the factor matrices, for different temporal resolutions, they are able to identify anomalous events, outperforming state-of-the-art IDS systems.

### 4.9. Speech and Image Processing and Computer Vision

Nion et al. [2010] use the CP decomposition to conduct Blind Source Separation (BSS). BSS is the problem of estimating a set of signals that are mixed by an unknown channel (hence "blind"), using solely the information on the receiver's end as measured by a set of sensors. This problem arises in scenarios such as speech separation, where each signal refers to an individual's speech, and sensors refer to microphones. The authors show that using CP can outperform other baselines, and furthermore, CP's uniqueness properties can give guarantees for the harder, underdetermined version of the problem where we have more speakers (signals) than microphones (sensors).

Liu et al. [2013] propose a tensor-based method for completing missing values in series of images. In order to do so, they define the trace norm for the tensor case and extend matrix completion algorithms that use the matrix trace norm. The authors compare their proposed method against Tucker, CP, and doing SVD on each tensor slice separately, and they demonstrate that their proposed method achieves superior performance.

Pioneering the use of tensors in computer vision, Vasilescu and Terzopoulos [2002] introduce TensorFaces, a methodology that uses tensors to analyze collections of face images into principal components across many different modes of a picture (e.g., different pose, different illumination, or different facial expression). Twenty-eight male subjects were photographed in five poses of three illuminations and three expressions, with 7,943 pixels per image. They apply HOSVD to this ensemble of images and identify the principal variation of every picture with respect to all the modes of the data.

Finally, Tao et al. [2008] propose a Bayesian version of HOSVD and use it in order to tackle model 3D face data. In particular, using their proposed decomposition in ensembles of 3D face images, they compute a parsimonious representation of the 3F faces in the form of core tensor, which captures the latent characteristics of the 3D faces, which is sufficient to later reconstruct any particular face with a given expression.

## 5. SCALING UP TENSOR DECOMPOSITIONS

With the advent of big data, tensor decompositions have faced a set of challenges that needed to be addressed before tensors could be used for truly big data applications. In recent years, this particular subfield of designing scalable tensor decompositions has witnessed remarkable growth and is currently at a sufficiently mature stage, where tensor decompositions can be deployed in the big data scale. In this section, we present such advances, mostly in chronological order, and draw high-level abstractions, summarizing the key ideas and insights behind each method.

At a high level, the algorithms that we survey here can be categorized as using one or more of the following strategies in order to achieve scalability and speed:

—**Compression:** Coming up with a compressed version of the tensor and decomposing it instead of the full data is one of the earliest approaches, which has also been a recurring theme in recent works.

—**Exploiting Sparsity:** In many applications, the tensor is highly sparse (i.e., many of the values are 0) because of the very nature of the application (e.g., in Facebook, people are usually friends with a few hundred people, out of the 2 billion people who use Facebook; therefore, such a graph would have very few connections). Exploiting this sparsity in various ways, either by redesigning the algorithm so that it carries out sparse matrix multiplications or using it in conjunction with sampling (which is the next major category), has been a very popular way of achieving scalability and efficiency.
—**Sampling:** Either via drawing a sample of a few entries of the tensor or through extracting sampled subtensors from the data, sampling has been instrumental in creating approximate algorithms that achieve good accuracy (often comparable to exact algorithms) while being much more lightweight in terms of computation.
—**Parallel/Distributed Computation:** In conjunction with the rest of the techniques, many recent works have taken advantage of parallelization of the algorithms, or the use of existing high-end distributed computation environments (e.g., Hadoop/MapReduce), and thus achieve scalability.

In the next few lines, we will summarize the advances for the CP, Tucker, and H-Tucker decompositions, for which there exist scalable algorithms. Furthermore, Tables II and III classify the algorithms we summarize with respect to the aforementioned strategies.

### 5.1. CP

Perhaps the earliest method that, by using compression, speeds up the CP decomposition is the one proposed in Bro and Andersson [1998]. The authors use the observation that for a given CP model of a three-mode tensor, $[\mathbf{A}, \mathbf{B}, \mathbf{C}]$, the factor matrices span the subspaces of that tensor; if we call $[\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3]$ the bases for those subspaces, we can then write $\mathbf{A} = \mathbf{U}_1\tilde{\mathbf{A}}$, $\mathbf{B} = \mathbf{U}_2\tilde{\mathbf{B}}$, and $\mathbf{C} = \mathbf{U}_2\tilde{\mathbf{C}}$, where $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{C}}$ are much smaller factor matrices. This gives rise to the algorithm that first computes a Tucker decomposition on the original tensor $\mathcal{X}$, obtaining core $\mathcal{G}$ and subspace basis matrices $[\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3]$. The core is a compressed version of $\mathcal{X}$. The algorithm then computes a CP decomposition on $\mathcal{G}$, obtaining the compressed factors $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{C}}$, and then using the Tucker matrices, it projects those factors back to the original subspace of the tensor. This works because it can be shown that this is a CANDELINC [Carroll et al. 1980] model, essentially a CP model with linear constraints on the factors, and it can proved that matrices $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}, \tilde{\mathbf{C}}$ preserve the variation of the data. Subsequently, Cohen et al. [2015], based on the same principle, derive a Tucker-compression-based algorithm to speed up CP with nonnegativity constraints.

Subsequently, Kolda et al. [2005] define a greedy algorithm for CP that is based on the computation of multiple rank-one components. In order to do so, they define the following recursive equations:

$$\mathbf{x}^{(t+1)} = \mathcal{X} \times_2 \mathbf{y}^{(t)} \times_3 \mathbf{z}^{(t)}$$
$$\mathbf{y}^{(t+1)} = \mathcal{X} \times_1 \mathbf{x}^{(t+1)} \times_3 \mathbf{z}^{(t)}$$
$$\mathbf{z}^{(t+1)} = \mathcal{X} \times_1 \mathbf{x}^{(t+1)} \times_2 \mathbf{y}^{(t)}.$$

The previous recursion is called a higher-order power method and converges to vectors $\mathbf{x}^*, \mathbf{y}^*, \mathbf{z}^*$, which can be shown to be equal to the rank-one CP decomposition of $\mathcal{X}$, assuming that $\mathcal{X}$ is rank-one and noise-free. The power method for tensors dates back to Kofidis and Regalia [2002], wherein the authors propose that method for the symmetric case of $\mathbf{x} = \mathbf{y} = \mathbf{z}$. The power method is very advantageous for sparse tensors, since its complexity is in the order of number of nonzero entries in $\mathcal{X}$.

This gives us a way of computing a single rank-one component; therefore, Kolda et al. [2005] adopt a "deflation" technique where they compute a rank-one component at every iteration, remove it from the data, and continue iterating until all $R$ components are extracted. This approach is not optimal, since CP factors are not orthogonal; however, it has been shown that for sparse factors, deflation numerically converges to the same factors as if they were calculated all at the same time [Papalexakis et al. 2013].

Later, Araujo et al. [2014] used the same principle of rank-one component calculation via the higher-order power method and deflation, with the addition of a Minimum Description Length (MDL) cost function, which dictates the termination of the deflation (and chooses the number of components) automatically.

A significant portion of the literature is devoted to scaling up the ALS algorithm for CP. In Section 3.1.2 and Equation (3), we define the operation MTTKRP as

$$\mathbf{Y} = \mathbf{X}_{(1)}(\mathbf{C} \odot \mathbf{B}).$$

When the dimensions $I \times J \times K$ of $\mathcal{X}$ are big, materializing the Khatri-Rao product $(\mathbf{C} \odot \mathbf{B})$ and carrying out MTTKRP can be prohibitive. This turns out to be a scalability problem of the Alternating Least Squares algorithm, which we will henceforth refer to as "Intermediate Data Blowup" or "Intermediate Data Explosion" (a term first coined by Kolda and Sun [2008] and later by Kang et al. [2012]), since $(\mathbf{C} \odot \mathbf{B})$ is an interim piece of data which is a by-product of the algorithm and not the output, which, however, requires immense storage and computational resources.

The "Intermediate Data Explosion" issue was first identified and addressed by Bader and Kolda [2007], wherein the authors define a suite of operations for sparse tensors with a specific focus in Matlab. This later became the widely used Tensor Toolbox for Matlab [Bader and Kolda 2015], which specializes in the manipulation of sparse tensors. Sparsity is key in order to achieve scalability, since it allows for the manipulation of the nonzero values of $\mathcal{X}$ in operations such as MTTKRP, which leads to far more efficient algorithms. In particular, Bader and Kolda [2007] show how instead of computing the Khatri-Rao product, this expensive operation can be carried out for every column of $\mathbf{C}$ and $\mathbf{B}$ independently. In this column-wise view of the operation, the computation simply reduces to $n$-mode products of $\mathcal{X}$ and the columns of $\mathbf{C}$ and $\mathbf{B}$:

$$\mathbf{Y}(:, r) = \mathcal{X} \times_3 \mathbf{C}(:, r) \times_2 \mathbf{B}(:, r)$$

for $r = 1 \cdots R$. The $n$-mode products of a sparse tensor $\mathcal{X}$ with the columns of $\mathbf{C}$ and $\mathbf{B}$ can be carried out efficiently, without ever computing $(\mathbf{C} \odot \mathbf{B})$, which makes the proposed algorithm in Bader and Kolda [2007] very efficient for sparse tensors.

Subsequently, Kang et al. [2012] propose a mathematically equivalent way of avoiding the Intermediate Data Explosion that is more appropriate for distributed computation, proposing the first CP ALS algorithm for the Map/Reduce framework. Instead of using $n$-mode products, the authors decouple the Khatri-Rao product between $\mathbf{C}$ and $\mathbf{B}$, separately compute products of each column $\mathbf{C}(:, r)$ and $\mathbf{B}(:, r)$ with $\mathbf{X}_{(1)}$, and then combine the results using Hadamard (element-wise) products. The reason Kang et al. [2012] follows this way is because it is more amenable to the Map/Reduce programming model, which they use in order to distribute the computation of the ALS algorithm among a cluster of machines. In addition to implementing an efficient MTTKRP operation, Kang et al. [2012] also introduce Map/Reduce optimizations that further enable scalability for tensors that do not fit in memory. Recently, they follow up with Jeon et al. [2015], which is more efficient than Kang et al. [2012] in the Map/Reduce framework.

More recently, Choi and Vishwanathan [2014] provide another efficient algorithm for computing the MTTKRP operation for sparse tensors. In particular, the authors show that the MTTKRP of Equation 3 can be equivalently computed in a column-wise

manner as follows:

$$\mathbf{M} = \mathbf{X}_{(2)}^T \mathbf{B}(:, r)$$
$$\mathbf{Y}(:, r) = \mathbf{MC}(:, r).$$

All of the aforementioned operations are very efficient assuming that $\mathcal{X}$ is sparse. Furthermore, the authors implement the ALS algorithm using this version of MTTKRP in a shared-memory parallel architecture, efficiently scaling up the decomposition.

Ravindran et al. [2014] provide a memory-efficient algorithm for MTTKRP that exploits sparsity and has memory requirements in the order of the nonzero entries of the tensor.

The latest work that is improving upon the MTTKRP operation is Smith et al. [2015], wherein the authors, instead of computing the result of MTTKRP in a column-wise fashion, as the rest of the existing methods do, compute each row $\mathbf{Y}(i, :)$ at a time, which has the advantage that it requires a single traversal of the elements of the sparse tensor $\mathcal{X}$, and indeed results in more efficient computation compared to the previous solutions to MTTKRP.

In a different spirit, Phan and Cichocki [2009] propose a block CP decomposition, where they partition the original tensor into smaller subtensors, distribute the computation of the subtensors potentially into different machines in a shared-memory architecture, and finally merge the factor matrices resulting from each individual subtensor by introducing multiplicative updates. The idea behind Phan and Cichocki [2009] tackles the Intermediate Data Explosion by reducing the dimensions of the tensor in a way that the intermediate data created by the ALS algorithm are no longer an issue for scalability. A potential issue, however, with Phan and Cichocki [2009] is, especially in the case of very sparse tensors, that it is very likely that many subtensors will be almost entirely zero or rank deficient, which may result in degenerate solutions.

A few years later, Papalexakis et al. [2012] introduced a parallel CP decomposition that uses biased sampling to extract subtensors from the data. Suppose that we have a sample of rows that is a set of indices $\mathcal{I}$ and, accordingly, $\mathcal{J}$ and $\mathcal{K}$ for the columns and third-mode fibers. Then, each subtensor $\mathcal{X}_p$ is defined as

$$\mathcal{X}_p = \mathcal{X}(\mathcal{I}, \mathcal{J}, \mathcal{K}).$$

The algorithm extracts many different subtensors $\mathcal{X}_p$, which are in turn decomposed in parallel. In the end, Papalexakis et al. [2012] merge the individual partial results from the subtensors by filling in the values in the indices of the factors that have been sampled in the first step. This has the fortuitous by-product that the resulting factors will be sparse by construction, which is very desirable both for storage and for interpretability. Furthermore, Papalexakis et al. [2012] provably guarantee that the different samples will be merged to the correct corresponding components. The fundamental difference from Phan and Cichocki [2009] is that sampling selects subtensors that are more likely to lead to high-quality solutions. Furthermore, Papalexakis et al. [2012] process fewer subtensors than Phan and Cichocki [2009], especially when the tensor is sparse. The idea of Papalexakis et al. [2012] is later on extended in the case of Coupled Matrix-Tensor Factorization in Papalexakis et al. [2014], showing significant speedups.

In the realm of Boolean tensor decompositions, Erdos and Miettinen [2013] propose an algorithm that bears a few similarities with Papalexakis et al. [2012] in the sense that it uses randomization to select dense blocks within a tensor and decomposes those blocks. In contrast to Papalexakis et al. [2012] and Erdos and Miettinen [2013] use random walks to identify dense blocks in a binary tensor. The authors define a graph where the nodes are the nonzero elements of the tensor, and the edges connect elements

that share at least two indices (in a three-mode tensor). The blocks that the random walk finds correspond to rank-one components of a Boolean CP decomposition; thus, the method returns the Boolean decomposition of the most important blocks in the data.

As we saw, Papalexakis et al. [2012] use sampling to reduce the dimensionality of the data and parallelize the decomposition. In a follow-up work, Sidiropoulos et al. [2014] propose an alternative scheme, where instead of sampling, they use random projection matrices $\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3$ (not to be confused with the Tucker matrices of Bro and Andersson [1998]) and compress the original tensor $\mathcal{X}$ into a smaller tensor $\mathcal{X}_p$ as

$$\mathcal{X}_p = \mathcal{X} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3.$$

As in Papalexakis et al. [2012], they create multiple compressed tensors $\mathcal{X}_p$, which are decomposed in parallel, and at the end, solving a least squares problem, they are able, under mild conditions, to identify the true factors of $\mathcal{X}$ up to column permutations and scaling, which is a very important theoretical guarantee of correctness. Sidiropoulos et al. [2014] build upon the result of Sidiropoulos and Kyrillidis [2012], where the authors show that one can reconstruct the original $\mathbf{A}, \mathbf{B}, \mathbf{C}$ from a single compressed replica, provided that the latent factors are sparse (requirement that is not posed by Sidiropoulos et al. [2014]). Subsequent work by Ravindran et al. [2014] shows how the compression of $\mathcal{X}$ into $\mathcal{X}_p$ can be done in a memory-efficient fashion.

In De Almeida and Kibangou [2014], we find a parallel approach that builds upon the idea of breaking down the tensor into a grid, such as in Phan and Cichocki [2009], and parallelizing the computation. The novelty in De Almeida and Kibangou [2014] is the fact that the communication between different machines that are working on separate tensor blocks is very important, especially so that machines that work on blocks that correspond to the same part of the latent factors can collaborate. Choosing the connectivity correctly, by using multilayer graphs to define connectivity between machines, can speed up the ALS algorithm and end up in solutions that under assumptions are identifiable.

Most of the aforementioned methods, with the exception of Papalexakis et al. [2012] and Cohen et al. [2015], are focused on the "vanilla" CP decomposition, without imposing any type of constraints. Liavas and Sidiropoulos [2015] authors propose a parallelizable algorithm for CP based on the Alternating Direction of Multipliers Method (ADMM) [Boyd et al. 2011], which can accommodate a wide variety of constraints and is particularly well suited for distributing parts of the optimization variables over different machines.

So far, all the scalable algorithms that we have seen, either explicitly or implicitly, use the ALS algorithm. The work of Beutel et al. [2014], however, introduces a Map/Reduce algorithm based on Distributed Stochastic Gradient Descent. The algorithm first splits the tensor into disjoint blocks, or "strata," which correspond to disjoint parameters in the factor matrices. Each stratum is distributed to a different computer, and for each stratum, the algorithm uses Stochastic Gradient Descent (SGD) updates to estimate the parameters of the factors corresponding to that stratum. In order to correctly cover all the data, the algorithm creates different sets of disjoint blocks and iterates over those block configurations. SGD is a very efficient, stochastic algorithm that uses a randomly selected data point at every different update instead of using the entire data, as other gradient based approaches do. As a framework , SGD, and as a result that of Beutel et al. [2014], is very flexible and can accommodate different objective functions (e.g., KL-Divergence instead of Frobenius norm) or regularizations such as $\ell_1$ norm penalties. One subtle issue to note here is that SGD is sampling from the "observed" values at random, and there is an inherent assumption that 0 values in the data are considered "unobserved" or "missing," a fact that we also touch upon in Section 3.1.3.

Table II. Classification of Scalable Algorithms for CP with Respect to the Strategies They Employ

| Algorithm | Compression | Exploits Sparsity | Sampling | Dist. / Parallel |
|---|---|---|---|---|
| Bro and Andersson [1998] | ✓ | | | |
| Kolda et al. [2005] | | ✓ | | |
| Bader and Kolda [2007] | | ✓ | | |
| Phan and Cichocki [2009] | | | | ✓ |
| Kim and Candan [2012] | | | | ✓ |
| Kang et al. [2012] | | ✓ | | ✓ |
| Papalexakis et al. [2012] | | ✓ | ✓ | ✓ |
| Erdos and Miettinen [2013] | | ✓ | ✓ | ✓ |
| De Almeida and Kibangou [2014] | | | | ✓ |
| Choi and Vishwanathan [2014] | | ✓ | | ✓ |
| Araujo et al. [2014] | | ✓ | | |
| Beutel et al. [2014] | | ✓ | ✓ | ✓ |
| Sidiropoulos et al. [2014] | ✓ | | | ✓ |
| Liavas and Sidiropoulos [2015] | | | | ✓ |
| Shin and Kang [2014] | | ✓ | | ✓ |
| Ravindran et al. [2014] | | ✓ | | |
| Smith et al. [2015] | | ✓ | | ✓ |
| [Cohen et al. 2015] | ✓ | | | |
| Jeon et al. [2015] | | ✓ | | ✓ |

Subsequently, Shin and Kang [2014] propose two distributed methods for Map/Reduce, one based on ALS and one based on Coordinate Descent. The ALS-based method works on sets of columns of the factor matrices and updates entire rows at a time. Coordinate Descent, on the other hand, updates individual coefficients of the factor matrices in a column-wise fashion. The authors demonstrate that both methods scale well, both in terms of tensor dimensionality and number of nonzeros, and show that their ALS-based method is more efficient in terms of convergence speed, whereas the Coordinate Descent method offers significant memory gains.

Finally, the work of Kim and Candan [2012] is quite different in flavor and draws from the domain of Relational Databases. The basic idea behind the proposed method is as follows: an $N$-mode tensor can be seen as a "relation" or table in a relational database, where each mode is a different column. A widely used operation in databases is the so-called normalization, where one splits up a relation with many columns (or alternatively a high-order tensor) into smaller relations, where one of those columns is shared. This is exactly what Kim and Candan [2012] do to split up a large and high-order tensor into smaller ones, compute the CP decomposition on the "normalized" relations/tensors, and, in the end, merge the factors using a "joining," another widely used database operation, which takes two relations with a column in common and joins the remaining columns based on matching entries for that common column.

### 5.2. Tucker

As we mentioned in the previous subsection, the work of Kolda and Sun [2008] was the first to mention the issue of Intermediate Data Blowup. In the particular case of the Tucker decomposition, the problem refers to the operation of line 4 of Algorithm 3:

$$\mathcal{Y} \leftarrow \mathcal{X} \times_N \mathbf{U}_N^T \cdots \times_{n+1} \mathbf{U}_{n+1}^T \times_{n-1} \mathbf{U}_{n+1}^T \cdots \times_1 \mathbf{U}_1^T.$$

This operation creates a series of intermediate results, the first of which is $\mathcal{X} \times_N \mathbf{U}_N^T$. If tensor $\mathcal{X}$ has large dimensions, then this intermediate piece of data will be dense (since the Tucker factors are dense matrices) and large, which creates the Intermediate Data Blowup problem. In order to tackle this problem, the authors propose to handle these

Table III. Classification of Scalable Algorithms for Tucker with Respect to the Strategies They Employ (We Omit Compression, Since This Has Been an Overarching Theme for Scaling Up CP (Often Using Tucker to Obtain Such Compression))

| Algorithm | Exploits Sparsity | Sampling | Dist. / Parallel |
|---|---|---|---|
| Kolda and Sun [2008] | ✓ | | |
| Caiafa and Cichocki [2010] | | ✓ | |
| Tsourakakis [2010] | ✓ | ✓ | |
| Jeon et al. [2015] | ✓ | | ✓ |
| Austin et al. [2016] | | | ✓ |

$N$-mode products of the previous operation element-wise for one or more modes of the tensor.

Especially in the cases where the tensor is sparse, executing the product element-wise for a given mode can be done more efficiently, since the computational complexity depends on the number of nonzero entries in the data. As a result of this, the algorithm in Kolda and Sun [2008] requires much less storage and is able to compute the Tucker decomposition of very large tensors. A side effect of handling one or more modes element-wise is that in some cases, this algorithm tends to be slower than Algorithm 3; however, the gains in terms of memory requirements are up to 1,000-fold.

Subsequently, the work of Tsourakakis [2010] uses sparsification of a tensor in order to achieve scalability. In particular, the author proves that by randomly sampling nonzero entries of the tensor and scaling appropriately, the expected approximation error does not suffer a lot. Thus, by sparsifying the tensor, the author demonstrates that we can compute the Tucker decomposition (using either Algorithm 2 or Algorithm 3) much faster while still capturing the variation of the full data.

Caiafa and Cichocki [2010] propose an extension to the matrix CUR decomposition [Drineas et al. 2006] for tensors, different from and more efficient than the Tensor-CUR decomposition [Mahoney et al. 2008]; in a nutshell, CUR decomposes a matrix $\mathbf{X} \approx \mathbf{CUR}$, where $\mathbf{C}$ contains sampled columns of the $\mathbf{X}$, $\mathbf{R}$ contains sampled rows, and $\mathbf{U}$ is computed such that the squared error is minimized. In this work, the authors introduce an adaptive algorithm for selecting the rows, columns, and third-mode fibers of the tensor that can be done efficiently, resulting in the fast computation of a Tucker-like model.

Fairly recently, Jeon et al. [2015] follow up the work in Kang et al. [2012], which is CP ALS for Map/Reduce, by introducing a unified decomposition framework with operations that are applicable both for CP and Tucker decompositions. In particular, Jeon et al. [2015] introduce a scalable and distributed implementation of the $N$-mode product, which is essential to both CP (as Bader and Kolda [2007] demonstrate) and Tucker computations, by decoupling its steps in a way that is suited for the Map/Reduce framework. As an extension of Jeon et al. [2015], the same research group provides a scalable implementation of Coupled Matrix-Tensor Factorization in Jeon et al. [2016].

Finally, the most recent work that is speeding up the Tucker decomposition is by Austin et al. [2016], who propose the first distributed memory implementation of Algorithms 2 and 3. They identify the basic bottleneck algorithms, essentially the $N$-mode product and the computation of the $R$ leading singular vectors of a matricized tensor (which they do by computing the eigenvectors of a symmetric matrix created from the matricized tensor). The proposed implementation is very scalable and was able to decompose (and ultimately compress, by using Tucker) multiple terabytes of scientific data expressed as high-order tensors.

### 5.3. H-Tucker

To the best of our knowledge, Perros et al. [2015] provide the first (and currently the only) scalable algorithm for the Hierarchical Tucker Decomposition. The main

computational bottleneck, especially for very high-order tensors, is the fact that the matricizations that Algorithm 5 is doing will have a prohibitively large dimension, since it will be the product of the sizes of all but one of the dimensions. This, in turn, makes the computation of the leading singular vectors per matricization an extremely hard problem, which makes Algorithm 5 unable to scale for very large and high-order tensors. The main idea behind Perros et al. [2015] is to use a sampling approach inspired by CUR, where instead of using the very high-dimensional matricizations, the authors sample a small number of columns (and make sure that the sampling is consistent across matricizations). Assuming that the original tensor is very sparse, the algorithm in Perros et al. [2015] ensures that all the data in the algorithm, both intermediate and results, are sparse, thus making the computations much faster and scalable.

## 6. CONCLUSIONS AND FUTURE CHALLENGES

Tensor decompositions are very versatile and powerful tools, ubiquitous in data mining applications. As we saw, they have been successfully integrated in a rich variety of real-world applications, and due to the fact that they can express and exploit higher-order relations in the data, they tend to outperform approaches that ignore such structure. Furthermore, recent advances in scaling up tensor decompositions have employed practitioners with a strong arsenal of tools that can be applied to many big multiaspect data problems.

The success that tensors have experienced in data mining during the last few years by no means indicates that all challenges and open problems have been addressed. Quite to the contrary, there exist challenges, some of which we summarize in the next few lines, which delineate very exciting future research directions:

—**Modeling space and time:** What is the best way to exploit spatial or temporal structure that exists in the data? We saw examples in Davidson et al. [2013], where the authors impose linear constraints that guide the decomposition according to prior knowledge of the spatial structure; Mørup et al. [2008, 2011], where the authors deal with temporal issues in brain data; and Sun et al. [2008], where the authors use a wavelet decomposition to represent time. Is there a generic way to incorporate such modifications in a tensor model and enable it to handle spatiotemporal data effectively? Furthermore, another open problem in spatiotemporal data modeling is selecting the right granularity for the space and time modes.

—**Unsupervised model selection:** In a wide variety of applications, ground truth is not easy to obtain; however, we need to have unsupervised means of understanding which tensor decomposition is more appropriate (e.g., CP vs. Tucker vs. DEDICOM etc.) and, given a decomposition, what model order is most appropriate for the data. As we outline in Section 3.8, there exists work in that respect; however, there is significant room for improvement and innovation.

—**Dealing with high-order data:** Many real-world applications involve data that can be represented as very high-order tensors. Works that use H-Tucker [Perros et al. 2015] have shown the utility of such approaches, and in the future, work on understanding and improving decompositions such as H-Tucker and Tensor-Train, in the context of data mining, will be very important.

—**Connections with Heterogeneous Information Networks:** In data mining, there exists a very rich line of work on Heterogeneous Information Networks (HINs), which are graphs between different types of nodes, connected with various types of edges. An HIN can be represented as a tensor, and in fact, a multiview social network is such an HIN. In the HIN mining literature, there exist concepts such as the "Meta-path" [Sun et al. 2011], which is a path within the network that traverses

multiple types of nodes, in the same spirit as a random walk, aiming to find similar nodes in the network (and it has also been used for clustering). Outlining connections between such works and tensor decompositions is a very interesting future direction that aims toward unifying different data mining approaches.

## ACKNOWLEDGMENTS

We thank Tamara G. Kolda for comments on earlier versions of this manuscript and identifying several additional references. The majority of this work was carried out while E. Papalexakis was at Carnegie Mellon University. Research was supported by the National Science Foundation Grants No. IIS-1247489 and IIS-1247632. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties.

## REFERENCES

Evrim Acar, Canan Aykut-Bingol, Haluk Bingol, Rasmus Bro, and Bülent Yener. 2007. Multiway analysis of epilepsy tensors. *Bioinformatics* 23, 13 (2007), i10–i18. DOI:http://dx.doi.org/10.1093/bioinformatics/btm210

Evrim Acar, Seyit A. Çamtepe, Mukkai S. Krishnamoorthy, and Bülent Yener. 2005. Modeling and multiway analysis of chatroom tensors. In *Intelligence and Security Informatics*. Springer, 256–268. DOI:http://dx.doi.org/ 10.1007/11427995_21

Evrim Acar, Daniel M. Dunlavy, and Tamara G. Kolda. 2009. Link prediction on evolving data using matrix and tensor factorizations. In *Proceedings of the 2009 IEEE International Conference on Data Mining Workshops (ICDMW'09)*. 262–269. DOI:http://dx.doi.org/10.1109/ICDMW.2009.54

Evrim Acar, Daniel M. Dunlavy, and Tamara G. Kolda. 2011. A scalable optimization approach for fitting canonical tensor decompositions. *Journal of Chemometrics* 25, 2 (2011), 67–86. DOI:http://dx.doi.org/10.1002/cem.1335

Evrim Acar, Daniel M. Dunlavy, Tamara G. Kolda, and Morten Mørup. 2010. Scalable tensor factorizations with missing data. In *SIAM International Conference on Data Mining (SDM)*. SIAM, 701–712. DOI:http://dx.doi.org/10.1137/1.9781611972801.61

Evrim Acar, Daniel M. Dunlavy, Tamara G. Kolda, and Morten Mørup. 2011. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems* 106, 1 (March 2011), 41–56. DOI:http://dx.doi.org/10.1016/j.chemolab.2010.08.004

Evrim Acar, Gözde Gurdeniz, Morten A. Rasmussen, Daniela Rago, Lars O. Dragsted, and Rasmus Bro. 2012. Coupled matrix factorization with sparse factors to identify potential biomarkers in metabolomics. In *IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 1–8. DOI:http://dx.doi.org/10.1109/icdmw.2012.17

Evrim Acar, Tamara G. Kolda, and Daniel M. Dunlavy. 2011. All-at-once optimization for coupled matrix and tensor factorizations. In *Proceedings of Mining and Learning with Graphs (MLG'11)*. https://www.cs.purdue.edu/mlg2011/papers/paper_4.pdf.

Evrim Acar, Anders J. Lawaetz, Morten Rasmussen, and Rasmus Bro. 2013. Structure-revealing data fusion model with applications in metabolomics. In *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC'13)*. IEEE, 6023–6026. DOI:http://dx.doi.org/10.1109/embc.2013.6610925

Evrim Acar, Evangelos E. Papalexakis, Morten A. Rasmussen, Anders J. Lawaetz, Mathias Nilsson, and Rasmus Bro. 2014. Structure-revealing data fusion. *BMC Bioinformatics* 15, 1 (2014), 239. DOI:http://dx.doi.org/10.1186/1471-2105-15-239

Evrim Acar, George Plopper, and Bülent Yener. 2012. Coupled analysis of in vitro and histology tissue samples to quantify structure-function relationship. *PloS One* 7, 3 (2012), e32227. DOI:http://dx.doi.org/10.1371/journal.pone.0032227

Evrim Acar and Bülent Yener. 2009. Unsupervised multiway data analysis: A literature survey. *IEEE Transactions on Knowledge and Data Engineering,* 21, 1 (2009), 6–20. DOI:http://dx.doi.org/10.1109/tkde.2008.112

Rakesh Agrawal, Behzad Golshan, and Evangelos Papalexakis. 2015a. A study of distinctiveness in web results of two search engines. In *24th International Conference on World Wide Web, Web Science Track*. ACM. DOI:http://dx.doi.org/10.1145/2740908.2743060

Rakesh Agrawal, Behzad Golshan, and Evangelos Papalexakis. 2015b. Whither social networks for web search? In *21st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Sydney, Australia. DOI:http://dx.doi.org/10.1145/2783258.2788571

Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M. Kakade, and Matus Telgarsky. 2014. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research* 15, 1 (2014), 2773–2832.

Miguel Araujo, Spiros Papadimitriou, Stephan Günnemann, Christos Faloutsos, Prithwish Basu, Ananthram Swami,
Evangelos E. Papalexakis, and Danai Koutra. 2014. Com2: Fast automatic discovery of temporal ('comet') communities. In *Advances in Knowledge Discovery and Data Mining*. Springer, 271–283. DOI:http://dx.doi.org/10.1007/ 978-3-319-06605-9_23

Woody Austin, Grey Ballard, and Tamara G. Kolda. 2016. Parallel tensor compression for large-scale scientific data. In *Proceedings of the 30th IEEE International Parallel & Distributed Processing Symposium (IPDPS'16)*.

Brett W. Bader, Richard A. Harshman, and Tamara G. Kolda. 2007. Temporal analysis of semantic graphs using ASALSAN. In *7th IEEE International Conference on Data Mining, 2007 (ICDM'07)*. IEEE, 33–42. DOI:http://dx.doi.org/10.1109/icdm.2007.54

Brett W. Bader and Tamara G. Kolda. 2007. Efficient MATLAB computations with sparse and factored tensors. *SIAM Journal on Scientific Computing* 30, 1 (Dec. 2007), 205–231. DOI:http://dx.doi.org/10.1137/060676489

Brett W. Bader and Tamara G. Kolda. 2015. MATLAB Tensor Toolbox Version 2.6. Available online. (February 2015). http://www.sandia.gov/~tgkolda/TensorToolbox/.

Jonas Ballani, Lars Grasedyck, and Melanie Kluge. 2013. Black box approximation of tensors in hierarchical Tucker format. *Linear Algebra and Its Applications* 438, 2 (2013), 639–657. DOI:http://dx.doi.org/10.1016/j.laa.2011.08.010

Arindam Banerjee, Sugato Basu, and Srujana Merugu. 2007. Multi-way clustering on relation graphs. In *SIAM International Conference on Data Mining (SDM)*. Vol. 7. SIAM, 145–156. DOI:http://dx.doi.org/10.1137/1.9781611972771.14

Alex Beutel, Abhimanu Kumar, Evangelos Papalexakis, Partha Pratim Talukdar, Christos Faloutsos, and Eric P. Xing. 2014. FLEXIFACT: Scalable flexible factorization of coupled tensors on hadoop. In *SIAM International Conference on Data Mining (SDM)*. SIAM. DOI:http://dx.doi.org/10.1137/1.9781611973440.13

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation. *Journal of Machine Learning Research* 3 (2003), 993–1022.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning* 3, 1 (2011), 1–122. DOI:http://dx.doi.org/10.1561/2200000016

John W. Brewer. 1978. Kronecker products and matrix calculus in system theory. *IEEE Transactions on Circuits and Systems* 25, 9 (1978), 772–781. DOI:http://dx.doi.org/10.1109/tcs.1978.1084534

Rasmus Bro. 1998. *Multi-way Analysis in the Food Industry: Models, Algorithms, and Applications*. Ph.D. Dissertation. Københavns Universitet'Københavns Universitet', LUKKET: 2012 Det Biovidenskabelige Fakultet for Fødevarer, Veterinærmedicin og NaturressourcerFaculty of Life Sciences, LUKKET: 2012 Institut for FødevarevidenskabDepartment of Food Science, 2012 Institut for Fødevarevidenskab, 2012 Kvalitet og TeknologiDepartment of Food Science, Quality & Technology.

Rasmus Bro and Claus A. Andersson. 1998. Improving the speed of multiway algorithms: Part II: Compression. *Chemometrics and Intelligent Laboratory Systems* 42, 1 (1998), 105–113. DOI:http://dx.doi.org/10.1016/s0169-7439(98)00011-2

Rasmus Bro and Henk A. L. Kiers. 2003. A new efficient method for determining the number of components in PARAFAC models. *Journal of Chemometrics* 17, 5 (2003), 274–286. DOI:http://dx.doi.org/10.1002/cem.801

Cesar F. Caiafa and Andrzej Cichocki. 2010. Generalizing the column–row matrix decomposition to multi-way arrays. *Linear Algebra Applications* 433, 3 (2010), 557–573. DOI:http://dx.doi.org/10.1016/j.laa.2010.03.020

J. Douglas Carroll and Jih-Jie Chang. 1970. Analysis of individual differences in multidimensional scaling via an N-way generalization of 'Eckart-Young' decomposition. *Psychometrika* 35, 3 (1970), 283–319. DOI:http://dx.doi.org/10.1007/bf02310791

J. Douglas Carroll, Sandra Pruzansky, and Joseph B. Kruskal. 1980. CANDELINC: A general approach to multidimensional analysis of many-way arrays with linear constraints on parameters. *Psychometrika* 45, 1 (1980), 3–24. DOI:http://dx.doi.org/10.1007/bf02293596

Kai-Wei Chang, Wen-tau Yih, and Christopher Meek. 2013. Multi-relational latent semantic analysis. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP'13)*. 1602–1612.

Kai-Wei Chang, Wen-tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP'14)*. 1568–1579. DOI:http://dx.doi.org/10.3115/v1/d14-1165

Peter A. Chew, Brett W. Bader, Tamara G. Kolda, and Ahmed Abdelali. 2007. Cross-language information retrieval using PARAFAC2. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 143–152. DOI:http://dx.doi.org/10.1145/1281192.1281211

Eric C. Chi and Tamara G. Kolda. 2012. On tensors, sparsity, and nonnegative factorizations. *SIAM Journal of Matrix Analysis & Applications* 33, 4 (2012), 1272–1299. DOI:http://dx.doi.org/10.1137/110859063

Luca Chiantini and Giorgio Ottaviani. 2012. On generic identifiability of 3-tensors of small rank. *SIAM Journal of Matrix Analysis & Applications* 33, 3 (2012), 1018–1037. DOI:http://dx.doi.org/10.1137/110829180

Joon Hee Choi and S. Vishwanathan. 2014. DFacTo: Distributed factorization of tensors. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, 1296–1304. http://papers.nips.cc/paper/5395-dfacto-distributed-factorization-of-tensors.pdf.

Andrzej Cichocki, Danilo Mandic, Lieven De Lathauwer, Guoxu Zhou, Qibin Zhao, Cesar Caiafa, and Huy Anh Phan. 2015. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *IEEE Signal Processing Magazine* 32, 2 (2015), 145–163. DOI:http://dx.doi.org/10.1109/msp.2013.2297439

Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and S. Amari. 2009. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*. John Wiley & Sons.

Jeremy E. Cohen, Rodrigo Cabral Farias, and Pierre Comon. 2015. Fast decomposition of large nonnegative tensors. *IEEE Signal Processing Letters* 22, 7 (2015), 862–866. DOI:http://dx.doi.org/10.1109/lsp.2014.2374838

Joao Paulo C. L. da Costa, Martin Haardt, and F. Romer. 2008. Robust methods based on the HOSVD for estimating the model order in PARAFAC models. In *5th Sensor Array and Multichannel Signal Processing Workshop, 2008 (SAM'08)*. IEEE, 510–514. DOI:http://dx.doi.org/10.1109/sam.2008.4606923

Ian Davidson, Sean Gilpin, Owen Carmichael, and Peter Walker. 2013. Network discovery via constrained tensor analysis of fMRI data. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 194–202. DOI:http://dx.doi.org/10.1145/2487575.2487619

André L. F. De Almeida and Alain Y. Kibangou. 2014. Distributed large-scale tensor decomposition. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'14)*. DOI:http://dx.doi.org/10.1109/icassp.2014.6853551

Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. 2000. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications* 21, 4 (2000), 1253–1278. DOI:http://dx.doi.org/10.1137/s0895479896305696

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science* 41, 6 (Sept. 1990), 391–407. DOI:http://dx.doi.org/10.1002/(SICI)1097-4571(199009)41:6⟨391::AID-ASI1⟩3.0.CO;2-9

Petros Drineas, Ravi Kannan, and Michael W. Mahoney. 2006. Fast Monte Carlo algorithms for matrices III: Computing a compressed approximate matrix decomposition. *SIAM Journal on Computing* 36, 1 (2006), 184. DOI:http://dx.doi.org/10.1137/s0097539704442702

Daniel M. Dunlavy, Tamara G. Kolda, and Evrim Acar. 2011. Temporal link prediction using matrix and tensor factorizations. *ACM Transactions on Knowledge Discovery from Data* 5, 2 (Feb. 2011). DOI:http://dx.doi.org/10.1145/1921632.1921636

Dóra Erdos and Pauli Miettinen. 2013. Walk'n'merge: A scalable algorithm for boolean tensor factorization. In *2013 IEEE 13th International Conference on Data Mining (ICDM'13)*. IEEE, 1037–1042. DOI:http://dx.doi.org/10.1109/icdm.2013.141

Beyza Ermiş, Evrim Acar, and A. Taylan Cemgil. 2015. Link prediction in heterogeneous data via generalized coupled tensor factorization. *Data Mining and Knowledge Discovery* 29, 1 (2015), 203–236. DOI:http://dx.doi.org/10.1007/s10618-013-0341-y

Lars Grasedyck. 2010. Hierarchical singular value decomposition of tensors. *SIAM Journal of Matrix Analysis & Applications* 31, 4 (2010), 2029–2054. DOI:http://dx.doi.org/10.1137/090764189

Lars Grasedyck, Daniel Kressner, and Christine Tobler. 2013. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen* 36, 1 (2013), 53–78. DOI:http://dx.doi.org/10.1002/gamm.201310004

Wolfgang Hackbusch and Stefan Kühn. 2009. A new scheme for the tensor representation. *Journal of Fourier Analysis and Applications* 15, 5 (2009), 706–722. DOI:http://dx.doi.org/10.1007/s00041-009-9094-9

Samantha Hansen, Todd Plantenga, and Tamara G. Kolda. 2015. Newton-based optimization for Kullback-Leibler nonnegative tensor factorizations. *Optimization Methods and Software* 30, 5 (April 2015), 1002–1029. DOI:http://dx.doi.org/10.1080/10556788.2015.1009977

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

Richard A. Harshman. 1972. PARAFAC2: Mathematical and technical notes. *UCLA Working Papers in Phonetics* 22 (1972), 30–44.

Richard A. Harshman. 1978. Models for analysis of asymmetrical relationships among N objects or stimuli. In *1st Joint Meeting of the Psychometric Society and the Society for Mathematical Psychology*. McMaster University, Hamilton, Ontario.

Richard A. Harshman. 1984. How can I know if its real? *A Catalog of Diagnostics for Use with Three-Mode Factor Analysis and Multidimensional Scaling.* 566–591.

Richard A. Harshman, Sungjin Hong, and Margaret E. Lundy. 2003. Shifted factor analysis—Part I: Models and properties. *Journal of Chemometrics* 17, 7 (2003), 363–378. DOI:http://dx.doi.org/10.1002/cem.808

Lifang He, Xiangnan Kong, S. Yu Philip, Ann B. Ragin, Zhifeng Hao, and Xiaowei Yang. 2014. Dusk: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages. *Matrix* 3, 1 (2014), 2. DOI:http://dx.doi.org/10.1137/1.9781611973440.15

Frank L. Hitchcock. 1927. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics* 6, 1 (1927), 164–189. DOI:http://dx.doi.org/10.1002/sapm192761164

Joyce C. Ho, Joydeep Ghosh, and Jimeng Sun. 2014. Marble: High-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 115–124. DOI:http://dx.doi.org/10.1145/2623330.2623658

Furong Huang, Sergiy Matusevych, Anima Anandkumar, Nikos Karampatziakis, and Paul Mineiro. 2014. Distributed latent Dirichlet allocation via tensor factorization. *Neural Information Processing Systems (NIPS) Optimization Workshop 2014*.

Furong Huang, U. N. Niranjan, Mohammad Umar Hakeem, and Animashree Anandkumar. 2013. Fast detection of overlapping communities via online tensor methods. *arXiv preprint arXiv:1309.0787* (2013).

ByungSoo Jeon, L. S. I. Jeon, and U. Kang. 2016. SCouT: Scalable coupled matrix-tensor factorization algorithm and discoveries. In *International Conference on Data Engineering (ICDE)*. IEEE.

Inah Jeon, Evangelos E. Papalexakis, U. Kang, and Christos Faloutsos. 2015. Haten2: Billion-scale tensor decompositions. In *International Conference on Data Engineering (ICDE)*. DOI:http://dx.doi.org/10.1109/icde.2015.7113355

Meng Jiang, Peng Cui, Fei Wang, Xinran Xu, Wenwu Zhu, and Shiqiang Yang. 2014. FEMA: Flexible evolutionary multi-faceted analysis for dynamic behavioral pattern discovery. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1186–1195. DOI:http://dx.doi.org/10.1145/2623330.2623644

Tao Jiang and Nicholas D. Sidiropoulos. 2004. Kruskal's permutation lemma and the identification of CANDECOMP/PARAFAC and bilinear models with constant modulus constraints. *IEEE Transactions on Signal Processing* 52, 9 (2004), 2625–2636. DOI:http://dx.doi.org/10.1109/tsp.2004.832022

U. Kang, Evangelos Papalexakis, Abhay Harpale, and Christos Faloutsos. 2012. Gigatensor: Scaling tensor analysis up by 100 times-algorithms and discoveries. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 316–324. DOI:http://dx.doi.org/10.1145/2339530.2339583

Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. 2010. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM, 79–86. DOI:http://dx.doi.org/10.1145/1864708.1864727

Henk A. L. Kiers. 1993. An alternating least squares algorithm for PARAFAC2 and three-way DEDICOM. *Computational Statistics & Data Analysis* 16, 1 (1993), 103–118. DOI:http://dx.doi.org/10.1016/0167-9473(93)90247-q

Henk A. L. Kiers and Albert Kinderen. 2003. A fast method for choosing the numbers of components in Tucker3 analysis. *British Journal of Mathematical and Statistical Psychology* 56, 1 (2003), 119–125. DOI:http://dx.doi.org/10.1348/000711003321645386

Henk A. L. Kiers. 1997. Weighted least squares fitting using ordinary least squares algorithms. *Psychometrika* 62, 2 (June 1997), 215–266. DOI:http://dx.doi.org/10.1007/BF02295279

Mijung Kim and K. Selçuk Candan. 2012. Decomposition-by-normalization (DBN): Leveraging approximate functional dependencies for efficient tensor decomposition. In *Proceedings of the 21st ACM*

*International Conference on Information and Knowledge Management*. ACM, 355–364. DOI:http://dx.doi.org/10.1145/2396761.2396809

Eleftherios Kofidis and Phillip A. Regalia. 2002. On the best rank-1 approximation of higher-order supersymmetric tensors. *SIAM Journal on Matrix Analysis & Applications* 23, 3 (2002), 863–884. DOI:http://dx.doi.org/10.1137/s0895479801387413

Tamara G. Kolda and Brett W. Bader. 2009. Tensor decompositions and applications. *SIAM Reviews* 51, 3 (2009). DOI:http://dx.doi.org/10.1137/07070111x

Tamara G. Kolda, Brett W. Bader, and Joseph P. Kenny. 2005. Higher-order web link analysis using multilinear algebra. In *5th IEEE International Conference on Data Mining*. IEEE, 8–pp. DOI:http://dx.doi.org/10.1109/icdm.2005.77

Tamara G. Kolda and Jackson R. Mayo. 2011. Shifted power method for computing tensor eigenpairs. *SIAM Journal on Matrix Analysis & Applications* 32, 4 (Oct. 2011), 1095–1124. DOI:http://dx.doi.org/10.1137/100801482

Tamara G. Kolda and Jimeng Sun. 2008. Scalable tensor decompositions for multi-aspect data mining. In *8th IEEE International Conference on Data Mining, 2008 (ICDM'08)*. IEEE, 363–372. DOI:http://dx.doi.org/10.1109/icdm.2008.89

Daniel Kressner and Christine Tobler. 2012. Htucker–A matlab toolbox for tensors in hierarchical Tucker format. *Mathicse, EPF Lausanne* (2012).

Joseph B. Kruskal. 1977. Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra and Its Applications* 18, 2 (1977), 95–138. DOI:http://dx.doi.org/10.1016/0024-3795(77)90069-6

Daniel D. Lee and H. Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401, 6755 (1999), 788–791. DOI:http://dx.doi.org/10.1038/44565

Athanasios P. Liavas and Nicholas D. Sidiropoulos. 2015. Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers. *IEEE Transactions on Signal Processing,* 63, 20 (2015), 5450–5463. DOI:http://dx.doi.org/10.1109/tsp.2015.2454476

Yu-Ru Lin, Jimeng Sun, Paul Castro, Ravi Konuru, Hari Sundaram, and Aisling Kelliher. 2009. Metafac: Community discovery via relational hypergraph factorization. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 527–536. DOI:http://dx.doi.org/10.1145/1557019.1557080

Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. 2013. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 35, 1 (2013), 208–220. DOI:http://dx.doi.org/10.1109/iccv.2009.5459463

Haiping Lu, Konstantinos N. Plataniotis, and Anastasios N. Venetsanopoulos. 2011. A survey of multilinear subspace learning for tensor data. *Pattern Recognition* 44, 7 (2011), 1540–1551. DOI:http://dx.doi.org/10.1016/j.patcog.2011.01.004

Machine Learning Department. 2016. Carnegie Mellon University. Read the Web. http://rtw.ml.cmu.edu/rtw/. (Last accessed: 2/13/2016).

Michael W. Mahoney, Mauro Maggioni, and Petros Drineas. 2008. Tensor-CUR decompositions for tensor-based data. *SIAM Journal on Matrix Analysis & Applications* 30, 3 (2008), 957–987. DOI:http://dx.doi.org/10.1137/060665336

Ching-Hao Mao, Chung-Jung Wu, Evangelos E. Papalexakis, Christos Faloutsos, and Tien-Cheu Kao. 2014. MalSpot: Multi2 malicious network behavior patterns analysis. In *Pacific Asia Conference on Knowledge Discovery and Data Mining (PAKDD) 2014*. DOI:http://dx.doi.org/10.1007/978-3-319-06608-0_1

K. Maruhashi, F. Guo, and C. Faloutsos. 2011. MultiAspectForensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In *Proceedings of the 3rd International Conference on Advances in Social Network Analysis and Mining*. DOI:http://dx.doi.org/10.1109/asonam.2011.80

Saskia Metzler and Pauli Miettinen. 2015. Clustering boolean tensors. *Data Mining and Knowledge Discovery* 29, 5 (2015), 1343–1373. DOI:http://dx.doi.org/10.1007/s10618-015-0420-3

Pauli Miettinen. 2011. Boolean tensor factorizations. In *2011 IEEE 11th International Conference on Data Mining (ICDM'11)*. IEEE, 447–456. DOI:http://dx.doi.org/10.1109/icdm.2011.28

Shahin Mohammadi, David Gleich, Tamara Kolda, and Ananth Grama. 2016. Triangular alignment (TAME): A tensor-based approach for higher-order network alignment. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (2016).

Morten Mørup and Lars Kai Hansen. 2009. Automatic relevance determination for multi-way models. *Journal of Chemometrics* 23, 7–8 (2009), 352–363. DOI:http://dx.doi.org/10.1002/cem.1223

Morten Mørup, Lars Kai Hansen, Sidse Marie Arnfred, Lek-Heng Lim, and Kristoffer Hougaard Madsen. 2008. Shift-invariant multilinear decomposition of neuroimaging data. *NeuroImage* 42, 4 (2008), 1439–1450. DOI:http://dx.doi.org/10.1016/j.neuroimage.2008.05.062

Morten Mørup, Lars Kai Hansen, and Kristoffer Hougaard Madsen. 2011. Modeling latency and shape changes in trial based neuroimaging data. In *2011 Conference Record of the 45th Asilomar Conference on Signals, Systems and Computers (ASILOMAR'11)*. IEEE, 439–443. DOI:http://dx.doi.org/10.1109/acssc.2011.6190037

Yang Mu, Wei Ding, Melissa Morabito, and Dacheng Tao. 2011. Empirical discriminative tensor analysis for crime forecasting. In *Knowledge Science, Engineering and Management*. Springer, 293–304. DOI:http://dx.doi.org/10.1007/978-3-642-25975-3_26

Atsuhiro Narita, Kohei Hayashi, Ryota Tomioka, and Hisashi Kashima. 2012. Tensor factorization using auxiliary information. *Data Mining and Knowledge Discovery* 25, 2 (2012), 298–324. DOI:http://dx.doi.org/10.1007/978-3-642-23783-6_32

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on Machine Learning (ICML'11)*. 809–816.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing YAGO: Scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web*. ACM, 271–280. DOI:http://dx.doi.org/10.1145/2187836.2187874

Dimitri Nion, Kleanthis N. Mokios, Nicholas D. Sidiropoulos, and Alexandros Potamianos. 2010. Batch and adaptive PARAFAC-based blind separation of convolutive speech mixtures. *IEEE Transactions on Audio, Speech, and Language Processing,* 18, 6 (2010), 1193–1207. DOI:http://dx.doi.org/10.1109/tasl.2009.2031694

Dimitri Nion and Nicholas D. Sidiropoulos. 2009. Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor. *IEEE Transactions on Signal Processing,* 57, 6 (2009), 2299–2310. DOI:http://dx.doi.org/10.1109/tsp.2009.2016885

Ivan V. Oseledets. 2011. Tensor-train decomposition. *SIAM Journal on Scientific Computing* 33, 5 (2011), 2295–2317. DOI:http://dx.doi.org/10.1137/090752286

Pentti Paatero. 1997. A weighted non-negative least squares algorithm for three-way "PARAFAC" factor analysis. *Chemometrics and Intelligent Laboratory Systems* 38, 2 (Oct. 1997), 223–242. DOI:http://dx.doi.org/10.1016/S0169-7439(97)00031-2

Evangelia Pantraki and Constantine Kotropoulos. 2015. Automatic image tagging and recommendation via PARAFAC2. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP'15)*. IEEE, 1–6. DOI:http://dx.doi.org/10.1109/mlsp.2015.7324363

Evangelos E. Papalexakis and Christos Faloutsos. 2015. Fast efficient and scalable core consistency diagnostic for the PARAFAC decomposition for big sparse tensors. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'15)*. IEEE. DOI:http://dx.doi.org/10.1109/icassp.2015.7179011

Evangelos E. Papalexakis, Leman Akoglu, and Dino Ienco. 2013. Do more views of a graph help? Community detection and clustering in multi-graphs. In *2013 16th International Conference on Information Fusion (FUSION'13)*. IEEE, 899–905.

Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D. Sidiropoulos. 2012. ParCube: Sparse parallelizable tensor decompositions. In *Machine Learning and Knowledge Discovery in Databases*. Springer, 521–536. DOI:http://dx.doi.org/10.1007/978-3-642-33460-3_39

Evangelos E. Papalexakis, Tom M. Mitchell, Nicholas D. Sidiropoulos, Christos Faloutsos, Partha Pratim Talukdar, and Brian Murphy. 2014. Turbo-SMT: Accelerating coupled sparse matrix-tensor factorizations by 200x. In *SIAM International Conference on Data Mining (SDM)*. SIAM. DOI:http://dx.doi.org/10.1137/1.9781611973440.14

Evangelos E. Papalexakis, Nicholas D. Sidiropoulos, and Rasmus Bro. 2013. From k-means to higher-way co-clustering: Multilinear decomposition with sparse latent factors. *IEEE Transactions on Signal Processing,* 61, 2 (2013), 493–506. DOI:http://dx.doi.org/10.1109/tsp.2012.2225052

Jing Peng, Daniel Dajun Zeng, Huimin Zhao, and Fei-yue Wang. 2010. Collaborative filtering in social tagging systems based on joint item-tag recommendations. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. ACM, 809–818. DOI:http://dx.doi.org/10.1145/1871437.1871541

Ioakeim Perros, Robert Chen, Richard Vuduc, and Jimeng Sun. 2015. Sparse hierarchical tucker factorization and its application to healthcare. In *2015 IEEE 15th International Conference on Data Mining (ICDM'15)*. IEEE. DOI:http://dx.doi.org/10.1109/icdm.2015.29

Anh-Huy Phan and Andrzej Cichocki. 2009. Block decomposition for very large-scale nonnegative tensor factorization. In *2009 3rd IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP'09)*. IEEE, 316–319. DOI:http://dx.doi.org/10.1109/camsap.2009.5413268

Anh-Huy Phan, Petr Tichavský, and Andrzej Cichocki. 2013. Low complexity damped Gauss–Newton algorithms for CANDECOMP/PARAFAC. *SIAM Journal on Matrix Analysis & Applications* 34, 1 (Jan. 2013), 126–147. DOI:http://dx.doi.org/10.1137/100808034

Niranjay Ravindran, Nicholas D. Sidiropoulos, Shaden Smith, and George Karypis. 2014. Memory-efficient parallel computation of tensor and matrix products for big tensor decomposition. In *2014 48th Asilomar Conference on Signals, Systems and Computers*. IEEE, 581–585. DOI:http://dx.doi.org/10.1109/acssc.2014.7094512

Steffen Rendle. 2010. Factorization machines. In *2010 IEEE 10th International Conference on Data Mining (ICDM'10)*. IEEE, 995–1000. DOI:http://dx.doi.org/10.1109/icdm.2010.127

Steffen Rendle and Lars Schmidt-Thieme. 2010. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*. ACM, 81–90. DOI:http://dx.doi.org/10.1145/1718487.1718498

Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th International Conference on Machine Learning*. ACM, 880–887. DOI:http://dx.doi.org/10.1145/1390156.1390267

Aaron Schein, John Paisley, David M. Blei, and Hanna Wallach. 2015. Bayesian poisson tensor factorization for inferring multilateral relations from sparse dyadic event counts. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1045–1054. DOI:http://dx.doi.org/10.1145/2783258.2783414

Amnon Shashua and Tamir Hazan. 2005. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the 22nd International Conference on Machine Learning*. ACM, 792–799. DOI:http://dx.doi.org/10.1145/1102351.1102451

Kijung Shin and U. Kang. 2014. Distributed methods for high-dimensional and large-scale tensor factorization. In *2014 IEEE International Conference on Data Mining (ICDM'14)*. IEEE, 989–994. DOI:http://dx.doi.org/10.1109/icdm.2014.78

Nicholas Sidiropoulos, Evangelos Papalexakis, and Christos Faloutsos. 2014. Parallel randomly compressed cubes: A scalable distributed architecture for big tensor decomposition. *IEEE Signal Processing Magazine* 31, 5 (2014), 57–70. DOI:http://dx.doi.org/10.1109/msp.2014.2329196

Nicholas D. Sidiropoulos and Rasmus Bro. 2000. On the uniqueness of multilinear decomposition of N-way arrays. *Journal of Chemometrics* 14, 3 (2000), 229–239. DOI:http://dx.doi.org/10.1002/1099-128X(200005/06)14:3⟨229::AID-CEM587⟩3.0.CO;2-N

Nicholas D. Sidiropoulos and Anastasios Kyrillidis. 2012. Multi-way compressed sensing for sparse low-rank tensors. *IEEE Signal Processing Letters* 19, 11 (2012), 757–760. DOI:http://dx.doi.org/10.1109/lsp.2012.2210872

Ajit P. Singh and Geoffrey J. Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 650–658. DOI:http://dx.doi.org/10.1145/1401890.1401969

Age K. Smilde, Johan A. Westerhuis, and Ricard Boqué. 2000. Multiway multiblock component and covariates regression models. *Journal of Chemometrics* 14, 3 (2000), 301–331. DOI:http://dx.doi.org/10.1002/1099-128x(200005/06)14:3⟨301::aid-cem594⟩3.0.co;2-h

Shaden Smith, Niranjay Ravindran, Nicholas D. Sidiropoulos, and George Karypis. 2015. SPLATT: Efficient and parallel sparse tensor-matrix multiplication. In *29th IEEE International Parallel & Distributed Processing Symposium*. DOI:http://dx.doi.org/10.1109/ipdps.2015.27

A. Stegeman, J. M. F. ten Berge, and L. De Lathauwer. 2006. Sufficient conditions for uniqueness in CANDECOMP/PARAFAC and INDSCAL with random component matrices. *Psychometrika* 71, 2 (2006), 219–229. DOI:http://dx.doi.org/10.1007/s11336-006-1278-2

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge unifying wordnet and wikipedia. In *16th International World Wide Web Conference (WWW'07)*. 697–706. DOI:http://dx.doi.org/10.1145/1242572.1242667

Jimeng Sun, Dacheng Tao, and Christos Faloutsos. 2006. Beyond streams and graphs: Dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 374–383. DOI:http://dx.doi.org/10.1145/1150402.1150445

Jimeng Sun, Charalampos E. Tsourakakis, Evan Hoke, Christos Faloutsos, and Tina Eliassi-Rad. 2008. Two heads better than one: Pattern discovery in time-evolving multi-aspect data. *Data Mining and Knowledge Discovery* 17, 1 (2008), 111–128. DOI:http://dx.doi.org/10.1007/978-3-540-87479-9_19

Jian-Tao Sun, Hua-Jun Zeng, Huan Liu, Yuchang Lu, and Zheng Chen. 2005. CubeSVD: A novel approach to personalized web search. In *Proceedings of the 14th International Conference on World Wide Web*. ACM, 382–390. DOI:http://dx.doi.org/10.1145/1060745.1060803

Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. 2011. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *International Conference on Very Large Data Bases (VLDB)* (2011).

Dacheng Tao, Mingli Song, Xuelong Li, Jialie Shen, Jimeng Sun, Xindong Wu, Christos Faloutsos, and Stephen J. Maybank. 2008. Bayesian tensor approach for 3-D face modeling. *IEEE Transactions on Circuits and Systems for Video Technology,* 18, 10 (2008), 1397–1410. DOI:http://dx.doi.org/10.1109/tcsvt.2008.2002825

Jos M. F. ten Berge and Nikolaos D. Sidiropoulos. 2002. On uniqueness in CANDECOMP/PARAFAC. *Psychometrika* 67, 3 (2002), 399–409. DOI:http://dx.doi.org/10.1007/bf02294992

Marieke E. Timmerman and Henk A. L. Kiers. 2000. Three-mode principal components analysis: Choosing the numbers of components and sensitivity to local optima. *British Journal of Mathematical and Statistical Psychology* 53, 1 (2000), 1–16. DOI:http://dx.doi.org/10.1348/000711000159132

Giorgio Tomasi and Rasmus Bro. 2005. PARAFAC and missing values. *Chemometrics and Intelligent Laboratory Systems* 75, 2 (2005), 163–180. DOI:http://dx.doi.org/10.1016/j.chemolab.2004.07.003

Giorgio Tomasi and Rasmus Bro. 2006. A comparison of algorithms for fitting the PARAFAC model. *Computational Statistics & Data Analysis* 50, 7 (2006), 1700–1734. DOI:http://dx.doi.org/10.1016/j.csda.2004.11.013

Charalampos E. Tsourakakis. 2010. MACH: Fast randomized tensor decompositions. In *SIAM International Conference on Data Mining (SDM)*. SIAM, 689–700. DOI:http://dx.doi.org/10.1137/1.9781611972801.60

L. R. Tucker. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika* 31, 3 (1966), 279–311. DOI:http://dx.doi.org/10.1007/bf02289464

Alex M. Vasilescu and Demetri Terzopoulos. 2002. Multilinear analysis of image ensembles: Tensorfaces. *Computer Vision ECCV 2002* (2002), 447–460. DOI:http://dx.doi.org/10.1007/3-540-47969-4_30

Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'14)*. ACM, New York, NY, 25–34. DOI:http://dx.doi.org/10.1145/2623330.2623656

Tom Wilderjans, Eva Ceulemans, and Iven Van Mechelen. 2009. Simultaneous analysis of coupled data blocks differing in size: A comparison of two weighting schemes. *Computational Statistics & Data Analysis* 53, 4 (2009), 1086–1098. DOI:http://dx.doi.org/10.1016/j.csda.2008.09.031

Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G. Schneider, and Jaime G. Carbonell. 2010. Temporal collaborative filtering with Bayesian probabilistic tensor factorization. In *SIAM International Conference on Data Mining (SDM)*. Vol. 10. SIAM, 211–222. DOI:http://dx.doi.org/10.1137/1.9781611972801.19

Tatsuya Yokota, Andrzej Cichocki, and Yukihiko Yamashita. 2012. Linked PARAFAC/CP tensor decomposition and its fast implementation for multi-block tensor analysis. In *Neural Information Processing*. Springer, 84–91. DOI:http://dx.doi.org/10.1007/978-3-642-34487-9_11

Fuzheng Zhang, Nicholas Jing Yuan, David Wilkie, Yu Zheng, and Xing Xie. 2015. Sensing the pulse of urban refueling behavior: A perspective from taxi mobility. *ACM Transactions on Intelligent Systems and Technology (TIST)* 6, 3 (2015), 37. DOI:http://dx.doi.org/10.1145/2644828

Q. Zhao, L. Zhang, and A. Cichocki. 2015. Bayesian CP factorization of incomplete tensors with automatic rank determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 37 (2015), 1751–1763. DOI:http://dx.doi.org/10.1109/tpami.2015.2392756

Vincent Wenchen Zheng, Bin Cao, Yu Zheng, Xing Xie, and Qiang Yang. 2010. Collaborative filtering meets mobile recommendation: A user-centered approach. In *AAAI International Conference on Artificial Intelligence (AAAI)*. Vol. 10. 236–241.

Vincent W. Zheng, Yu Zheng, Xing Xie, and Qiang Yang. 2012. Towards mobile intelligence: Learning from GPS history data for collaborative recommendation. *Artificial Intelligence* 184 (2012), 17–37. DOI:http://dx.doi.org/10.1016/j.artint.2012.02.002

Yu Zheng, Tong Liu, Yilun Wang, Yanmin Zhu, Yanchi Liu, and Eric Chang. 2014. Diagnosing New York City's noises with ubiquitous data. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 715–725. DOI:http://dx.doi.org/10.1145/2632048.2632102

Received February 2016; accepted April 2016