Big Data
Volume 4 Number 3, 2016

DOI: 10.1089/big.2016.0026

ORIGINAL ARTICLE

Unsupervised Tensor Mining for Big Data Practitioners

Evangelos E. Papalexakis[1,*,†] and Christos Faloutsos[1]

Abstract

Multiaspect data are ubiquitous in modern Big Data applications. For instance, different aspects of a social network are the different types of communication between people, the time stamp of each interaction, and the location associated to each individual. How can we jointly model all those aspects and leverage the additional information that they introduce to our analysis? Tensors, which are multidimensional extensions of matrices, are a principled and mathematically sound way of modeling such multiaspect data. In this article, our goal is to popularize tensors and tensor decompositions to Big Data practitioners by demonstrating their effectiveness, outlining challenges that pertain to their application in Big Data scenarios, and presenting our recent work that tackles those challenges. We view this work as a step toward a fully automated, unsupervised tensor mining tool that can be easily and broadly adopted by practitioners in academia and industry.

Keywords: big data analytics; data mining; machine learning

Introduction

Many real-world phenomena, especially in the age of Big Data, produce data and metadata that are inherently multiaspect. For instance, social interaction among individuals is a naturally multiaspect process. Social interaction has multiple modes or aspects: the means of interaction (e.g., who calls whom, who messages whom, and who is friends on Facebook with whom), the time of the interaction, the location, as well as the text and the language associated to it.

Such multiaspect data are ubiquitous in the modern interconnected world and there is imperative need for methods that model and process that data, and extract useful knowledge out of them, which can be used for decision support and scientific discovery.

A very powerful set of tools that are invaluable in that endeavor is tensors and tensor decompositions. A tensor is a multidimensional extension of a matrix and the number of dimensions of the tensor is called "order" or "modes." For instance, a matrix is a two-mode tensor, and a data cube is a three-mode tensor. Tensors are very expressive structures and they can naturally model multiaspect data such as the ones in our social interaction example: if we simply record the interactions between individuals, then we have a matrix (or two-mode tensor) of (person, person); if we additionally record the means of interaction, then we have a three-mode tensor of (person, person, means of interaction); if, on top of that, we have time-stamped events, this results in a four-mode tensor of (person, person, means of interaction, time); and if we have location information (which is now ubiquitous in most online social network platforms), we end up with a five-mode tensor of (person, person, means of interaction, time, location). Depending on what type of multiaspect data our application entails, we can have a corresponding tensor that models those data concisely.

Having successfully modeled our data using a tensor, how do we extract useful knowledge from the data? For this purpose, we use a tool called *tensor decomposition* or *tensor factorization* (both terms used interchangeably in this article). There exist multiple flavors of tensor decompositions that have different properties and we invite the interested reader to read[1] which contains excellent introductory material to the inner workings of tensor decompositions and[2] which is an excellent introduction to unsupervised data analysis using tensors.

[1]*Department of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania.*
[†]*Present address: Department of Computer Science and Engineering, University of California Riverside, Riverside, California.*

*Address correspondence to: *Evangelos E. Papalexakis, Department of Computer Science and Engineering, University of California Riverside, 355 Winston Chung Hall, Riverside, CA 92521,* E-mail: epapalex@cs.ucr.edu

In this article, we will focus on the so-called Canonical Decomposition, PARAFAC, or CP decomposition,[3] henceforth referred to as PARAFAC.

A pictorial representation of PARAFAC is shown in Figure 1. Essentially, each rank-one component of the decomposition corresponds to a dense "block" of data within the data tensor. This block need not be formed by consecutive rows, columns, and third-mode "fibers," but it can be visible after appropriately rearranging the rows, columns, and fibers.

Revisiting our social network example, suppose we have a three-mode tensor of (person, person, means of interaction) recording the amount of interactions between different people in a social network, taking its PARAFAC decomposition as shown in Figure 1 will result in a soft *co-clustering* of people and means of interaction: each latent component is a co-cluster, that is, as subset of people and means of interaction that exhibit very similar behavior. To familiarize the reader with the concept of co-clustering, in Figure 2 we present a very simple example of two co-clusters in a (user, movie) matrix that can conceptually contain movie ratings on Netflix by users. The main assumption of co-clustering is that postulating that a particular cluster of users enjoy viewing *all* movies the same is too restrictive. Instead, co-clustering relaxes this requirement and seeks to identify a group of users that have similar viewing behavior across a *subset* of the movies; thus, a co-cluster in a matrix is simply a subset of the rows (users) and columns (movies). In our particular example, there is one group of users who enjoy viewing horror movies, and a separate group of users who enjoy comedies. In reality, those co-clusters may very well overlap. Notice also that the "patch" in the matrix that denotes the co-cluster may not be immediately apparent to an analyst, since one needs to rearrange rows and columns appropriately to see it. When our data form a tensor, a co-cluster is a subset of rows, columns, and third-mode fibers, as shown in Figure 1. Such co-clusters manifest as blocks in the data, for which PARAFAC is ideal for uncovering. In fact, in Ref.[4] the authors showed that PARAFAC with additional sparsity constraints on the factor vectors essentially yields a co-clustering on the tensor data. We invite the interested reader to read the article[4] and references within for a more detailed treatment of co-clustering.

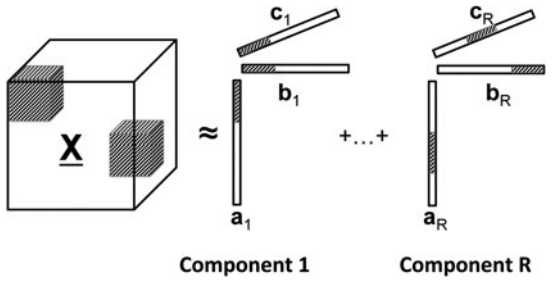


FIG. 1. Pictorial representation of the PARAFAC decomposition. Each rank-one component corresponds to a dense block in the data.

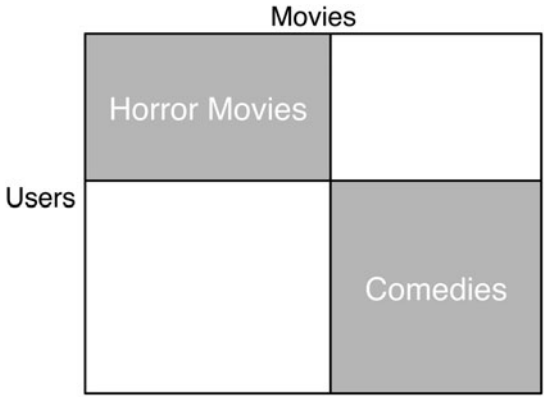


FIG. 2. Simple example of two co-clusters: a subset of people who enjoy watching a subset of movies (horror movies), and another subset who enjoy watching comedies.

This is very important for practitioners who wish to conduct exploratory analysis on the data and understand what latent clusters and patterns are present. There are also fundamental theoretical advantages of using tensors whenever data naturally possess multiple aspects; we refer the interested reader to Appendix 1 for a discussion.

Tensor decompositions have been shown to be effective in numerous fields. It is impressive that there exist multiple research communities that develop tensor algorithms and applications and demonstrate benefits of those approaches in their respective fields. There also exist multiple, cross-disciplinary meetings with sole topic that of tensor decompositions, which transcend different scientific disciplines, such as Psychology, Chemometrics, Signal Processing, and Data Mining.[5–7]

With this article, our hope is to familiarize the readers of this magazine with the concepts behind tensors and tensor decompositions, placing specific emphasis on a Big Data practitioner's point of view. To do so,

in the next section (Tensor Applications), we briefly explore a few of the numerous applications where tensors have been successful in data science and outline the difficulties that unsupervised tensor mining entails. In the Challenges in Unsupervised Tensor Mining section, we draw solutions from the field of Chemometrics and demonstrate how we can extend them for Big Data applications. Subsequently, in the Automatic Unsupervised Tensor Mining section, we describe an automatic, data-driven framework for tensor mining, outlining its inner workings and demonstrating its effectiveness. The Case Study section contains an indicative example of using tensor decomposition in conjunction with our framework for analyzing real data. Finally, in the Conclusions section, we conclude our discussion.

Tensor Applications

Tensor decompositions have been applied with great success in a broad spectrum of various data science applications. In this section, we survey a small sample of such applications, drawn from different domains.

One of the earliest tensor applications is found in Ref.[8] where the authors extend the extremely popular and successful Hyperlink-Induced Topic Search (HITS) algorithm by Jon Kleinberg,[9] which finds web pages that are hubs (point to many good pages), and authorities (are referenced by a lot of good pages). In particular, Ref.[8] adds a third dimension to the problem, recording the anchor text of a hyperlink, creating a (page, page, anchor text) tensor. By decomposing that tensor, the authors compute the topical-HITS, which finds hubs and authorities that pertain to a particular topic, an extension that is very important in understanding how different topics emerge on the web. The power of topical-HITS lies in its *interpretability*, because it attaches additional semantic information to a set of tightly connected (and presumably high-quality) web pages that can enable semantic search over those groups of topically coherent pages.

A different web application of tensors is found in Refs.[10,11] where the authors use tensor decompositions to measure the semantic similarity of the results of different search engines. They collect results of different search engines for a given set of queries over a period of time and form a four-mode tensor of (query, result keyword, date, and search date). Using the PARAFAC decomposition (which readily extends to four modes), they obtain an assignment of search engines to latent co-clusters; using those assignments, they determine the similarity of two search engines by looking at how similar their co-cluster assignments are. If two search engines end up in similar co-clusters, this indicates that their results are semantically similar, and if not, the results are dissimilar. Indicatively, they find that Google and Bing tend to produce very similar results for highly popular (so-called head) queries.

Moving from the web to social networks[12] demonstrates that using different views of a social network (e.g., who calls whom and who messages whom) results in more accurate communities and outperforms baseline methods that ignore this "high-order" structure of the data. More specifically,[12] decomposes a (person, person, means of interaction) using PARAFAC. It then uses the **a** vectors of PARAFAC, which provide assignment of persons to latent co-clusters to determine the co-cluster that each person belongs to; in this context, a co-cluster of people is a community within the social network.

Related to social network applications are urban computing applications. Such an example is Ref.[13] where the authors apply tensor analysis to estimate travel times of different trajectories of an urban road network, using real Global Positioning System (GPS) travel times measured only for small subsets of the road network. Because many road segments have not been traversed at all, or very rarely, this results in very sparse data, and thus, the main task of the authors[13] is to use tensor decomposition to fill in *missing values* of a (road segment, driver, time-slot) tensor. Completion of missing values is a fortuitous benefit of tensor decomposition; by reconstructing the tensor using the low-rank decomposition, many of the missing entries are estimated by leveraging the fact that the tensor decomposition discovers latent similarities between road segments, drivers, and time slots. Therefore, if two road segments are highly similar for a subset of the time slots and for a subset of the drivers (the same way that in Fig. 2 some users' preferences are similar for horror movies, and others' are similar for comedies), these latent similarities guide the completion of missing values. By estimating the missing values, on the tensor,[13] recovers very accurate travel times that can help the accuracy of navigation systems.

Tensors have been very successful in analyzing brain data. One of the earliest applications of tensors in brain data is in Ref.[14] where the authors analyze electroencephalogram (EEG) data from patients with epilepsy, to localize the origin of the seizure. To that end, they model the EEG data using a three-mode tensor (time samples, scales, electrodes) and tensor (after preprocessing the EEG measurements via a wavelet transformation).

To analyze the EEG tensor, they use the PARAFAC 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 the study by Davidson et al.,[15] the authors use fMRI scans and impose constraints to the PARAFAC decomposition that are inspired by neuroscience, to discover the so-called functional connectivity of the brain, that is, the information flow between different brain regions. In the study by Papalexakis et al.,[16] the authors discover semantically coherent brain regions among various human subjects in an unsupervised way. To do that, they analyze a tensor of fMRI scans for all the human subjects that records their responses for a variety of simple English nouns; they jointly analyze this tensor along with a set of semantic properties for those nouns, identifying groups of semantically coherent nous and the associated brain regions they activate.

Due to its ability to discover dense blocks within the data, the PARAFAC decomposition has been successful in network anomaly and intrusion detection. The reason for this is that various network attacks (such as denial of service from a botnet) manifest as dense bipartite cores in the data, which translate into dense blocks within the data. The works of Maruhashi et al. and Mao et al.[17,18] are such examples of successful application, identifying suspicious and malicious patterns in real network traffic data and intrusion detection system logs.

Finally, another very interesting and important line of work that tensors have been shown to be very effective is that of analyzing medical data. For example,[19] form a tensor of (patient, diagnosis, procedure) from Electronic Health Records that record diagnoses and procedures for different patients, and use the PARAFAC decomposition to identify phenotype candidates. Intuitively, the phenotypes that the authors seek to discover are groups of patients who have similar diagnoses and have undergone a similar procedure. Their approach is shown to discover meaningful phenotypes according to physicians, thereby demonstrating that tensor analysis can be instrumental in assisting medical professionals in decision-making.

Challenges in Unsupervised Tensor Mining

As we have demonstrated, tensors are very powerful tools and have enjoyed success in a wide variety of applications so far. However, to make tensors a de-facto analytical tool for today's Big Data practitioners, there still exist two major challenges that need to be addressed.

The first challenge in tensor mining, which has received considerable attention in the last few years, is the one of making tensor decompositions scalable to today's web scale. For instance, Facebook has around 2 billion users at the time of writing of this article and is ever growing, and making tensor decompositions able to work on even small portions of the entire Facebook network is imperative for the adoption of these techniques by such big industry players. Very frequently, data that fall under the aforementioned category turn out to be highly sparse; the reason is that, for example, each person on Facebook interacts with only a few hundreds of the users. Computing tensor decompositions for highly sparse scenarios is a game changer and exploiting sparsity is key in scalability. The work of Kolda et al.[8,20] introduced the first such approach of exploiting sparsity for scalability. Later on, distributed approaches based on the latter formulation,[21] or other scalable approaches[22–24] have emerged. The majority of the scalability work on tensor has focused on sparse data. However, tensor data can also be dense. For instance, when we are dealing with sensor measurements, where each sensor is producing a value for each time-tick, the resulting data will be fully dense. Handling such big and dense tensors is another great challenge that outlines very interesting research problems. To the best of our knowledge, there are only a few works that are able to deal with such data; in Ref.[25] the authors propose an MPI framework that is specifically tailored for large and dense tensors with very encouraging scalability results, and in Ref.[24] the authors propose a compression-based parallel framework that does not require the tensor to be sparse. By no means do we claim that scalability is a solved problem, however, we point out that there has been significant attention to it.

The second important challenge is the one of unsupervised quality assessment. In exploratory data mining applications, the case is very frequently the following: we are given a piece of (usually very large) data that is of interest to a domain expert, and we are asked to identify regular and irregular patterns that are potentially useful to the expert who is providing the data. Very often, this is done in an entirely unsupervised way since ground truth and labels are either very expensive or hard to obtain. In our context of tensor data mining, our problem, thus, is given a potentially very large and sparse tensor, and its R-component decomposition computes a quality

measure for that decomposition. Subsequently, using that quality metric, we would like to identify a good number of R components, and throughout this process, we would like to minimize human intervention and trial-and-error fine tuning.

This problem is extremely hard. In fact, even computing the rank of a tensor has been shown to be an NP-hard problem (in stark contrast to the matrix rank that can be easily computed in polynomial time). Fortunately, there exist heuristics that are able to assist with the above problem and have been shown to work well in practice, in the field of Chemometrics. Such a powerful and intuitive heuristic is the so-called Core Consistency Diagnostic[26]; in the following lines, we briefly describe how it works and we outline why this heuristic cannot be applied as-is in Big Data applications. We subsequently show how we can tackle those issues, summarizing our recent work by Papalexakis and Faloutsos[27] and Papalexakis.[28]

Notation preliminaries: Tensors are denoted by boldface underlined capital letters, for example, $\underline{\mathbf{X}}$, matrices by boldface capital letters, for example, $\mathbf{X}$, and vectors by boldface lower case letters, for example, $\mathbf{x}$. The rest of notation is defined when used. Given a tensor $\underline{\mathbf{X}}$, the PARAFAC decomposition is written as follows:

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

where the symbol $\circ$ denotes the outer product and the (i, j, k) element of $\mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$ is equal to $\mathbf{a}_r(i)\mathbf{b}_r(j)\mathbf{c}_r k$. The PARAFAC decomposition essentially decomposes the tensor into a sum of R outer products (each such outer product is also an elementary tensor of rank one*). We can also denote PARAFAC in its matrix form, where we gather all $\mathbf{a}_r$ vectors as columns of matrix $\mathbf{A}$: $\mathbf{A} = [\mathbf{a}_1 \quad \mathbf{a}_2 \quad \cdots \quad \mathbf{a}_R]$ and likewise for $\mathbf{B}$ and $\mathbf{C}$.

Core consistency diagnostic

The Core Consistency Diagnostic[26] is a heuristic method that takes as input a tensor $\underline{\mathbf{X}}$ and its R-component PARAFAC decomposition **A, B, C** and outputs a number c. This number can be 100 if the decomposition perfectly captures the data in $\underline{\mathbf{X}}$ and (usually) lower in the presence of noise. If c is very low (typically below 60–70) then our decomposition is somehow flawed, and either the data do not contain the block structure that PARAFAC decomposition seeks to find (recall Fig. 1) or the choice of R is wrong.

In a nutshell, the idea behind the Core Consistency Diagnostic[26] is as follows: Given a tensor $\underline{\mathbf{X}}$ and its PARAFAC decomposition **A, B, C**, one can imagine that each rank-one component has a number λ_r attached to it, which can absorb the scaling of the three vectors. Since there is a single number associated to each component, we can view these scalars λ_r as entries of an $R \times R \times R$ "core" tensor $\underline{\mathbf{G}}$ that is superdiagonal, that is, has nonzero entries only on its (i, i, i) coefficients. Imagine now that we ignore the fact that theoretically this core tensor $\underline{\mathbf{G}}$ ought to be superdiagonal, and we compute its value using $\underline{\mathbf{X}}$, **A, B, C**. If the tensor $\underline{\mathbf{G}}$ we estimate is exactly superdiagonal, or very close to being one, then our PARAFAC decomposition is a proper representation of the data; if on the contrary, $\underline{\mathbf{G}}$ is far from being superdiagonal, then our decomposition is degenerate and does not represent the data in $\underline{\mathbf{X}}$ properly. Figure 3 shows this pictorially.

To find the core tensor $\underline{\mathbf{G}}$, we need to solve the following minimization problem:

$$\min_{\underline{\mathbf{G}}} \| \, vec(\underline{\mathbf{X}}) - (\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C}) vec(\underline{\mathbf{G}}) \, \|_F^2$$

where $vec(\ )$ is the vectorization operation, stacking all elements of its input into a vector, $\| \ \|_F$ is the Frobenius norm, which is essentially the square root of the sum of squares of its input, and $\otimes$ is the Kronecker product, defined as follows:

DEFINITION 1 (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:*

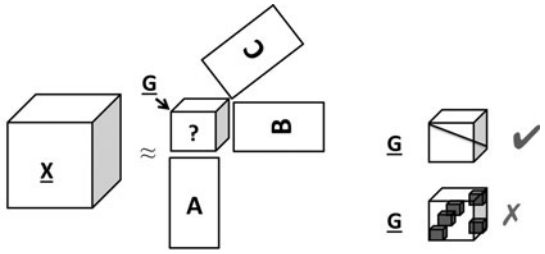


FIG. 3. The key idea behind the core consistency diagnostic.

*In fact, the rank of a tensor is defined as the minimum number of such outer products that are needed to add up to the original tensor.

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

The least squares solution for the above problem is as follows:

$$vec(\underline{\mathbf{G}}) = (\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C})^{\dagger} vec(\underline{\mathbf{X}})$$

where † is the Moore–Penrose pseudoinverse.

After computing $\underline{\mathbf{G}}$, the core consistency diagnostic can be computed as follows:

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

where $\underline{\mathbf{I}}$ is a superdiagonal tensor of dimensions $R \times R \times R$ and with ones in each superdiagonal entry. The above equation is effectively measuring how far is $\underline{\mathbf{G}}$ from a superdiagonal tensor.

The above heuristic is extremely useful and works very well in practice, in applications of Chemometrics. However, it cannot be applied "off-the-shelf" in applications of interest to Big Data practitioners, mainly due to the following two challenges:

- *Scalability and data size:* The core consistency diagnostic was originally designed and computed without considerations for dealing with very big and potentially sparse data, such as the ones created by an online social network. As a result, the state-of-the-art algorithms that compute the core consistency diagnostic are not able to scale to big tensors. We tackle this problem in Ref.[27] Being fairly mathematical, we outline the key contributions and results in the Appendix 1, so that they do not distract the reader from the high-level message of this article, of how tensor decompositions can benefit Big Data practitioners.
- *Modeling assumptions:* As originally defined, the core consistency diagnostic is minimizing the Frobenius norm (i.e., the sum of squares). This choice implies that we assume that the underlying data distribution is Gaussian. However, recent work[29] has demonstrated that in Big Data application scenarios where we have sparse counts (e.g., the amount of social interactions between two parties), a more accurate assumption is that of a Poisson distribution., which implies the use of the KL-divergence as our error function, instead of the Frobenius norm (which is essentially the squared error). This has been more recently adopted[19] showing very promising results in medical applications. In Ref.[28] we extend the core consistency diagnostic assuming Poisson distributed data, and in the Appendix we present the main contributions. Henceforth, we shall refer to the decomposition that follows the Poisson distribution as $\text{PARAFAC}_{\text{KL}}$, and as $\text{PARAFAC}_{\text{Fro}}$ to the one that follows the normal distribution.

Automatic Unsupervised Tensor Mining

At this stage, we have the tools we need to design an automated tensor mining algorithm that minimizes human intervention and provides quality characterization of the solution. We call our proposed method AUTOTEN, and we view this as a step toward making tensor mining a fully automated tool, used as a black box by academic and industrial practitioners. This method originally appeared in Ref.[28] and here we present a high-level description of it, from a Big Data practitioner's point of view, and demonstrate its effectiveness experimentally. The code for AUTOTEN is publicly available at www.cs .cmu.edu epapalex/src/AutoTen.zip.

Algorithm Description

AUTOTEN is solving the following problem:

PROBLEM 1.

Given: *A data tensor* $\underline{\mathbf{X}}$ *and a maximum search budget* $R_{\max}$

Return: *The* R^**-component PARAFAC decomposition of* $\underline{\mathbf{X}}$ *and its associated quality (Core Consistency)* c^**, where both* R^* *and* c^* *are maximized.*

AUTOTEN is a two-step algorithm, where we first search through the solution space and at the second step, we automatically select a good solution based on its quality and the number of components it offers. In the following lines and Figure 4, we present a description of AUTOTEN:

Step 1: Solution Search. The user provides a data tensor, as well as a maximum rank $R_{\max}$ that they are willing to search for. We follow a data-driven approach, where we let the data show us whether using $\text{PARAFAC}_{\text{Fro}}$ or $\text{PARAFAC}_{\text{KL}}$ is capturing better structure: For a grid of values of R, we run both $\text{PARAFAC}_{\text{Fro}}$ and $\text{PARAFAC}_{\text{KL}}$ in parallel and store the result quality using the core consistency diagnostic into vectors $\mathbf{c}_{Fro}$ and $\mathbf{c}_{KL}$.

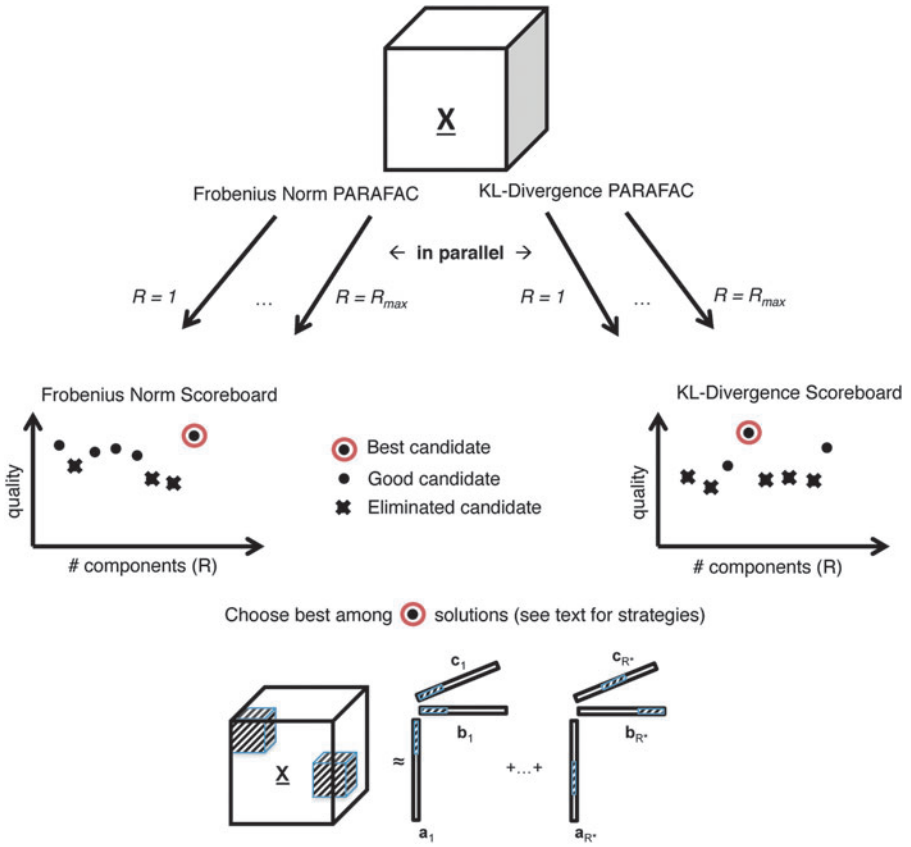


FIG. 4. Pictorial outline of the AUTOTEN algorithm.

Step 2: Result Selection. For both $\text{PARAFAC}_{\text{Fro}}$ and $\text{PARAFAC}_{\text{KL}}$, we have points in two-dimensional space (R_i, c_i), reflecting the quality and the corresponding number of components. These are the "scoreboards" shown in Figure 4. Given points (R_i, c_i), we need to find one that maximizes the quality of the decomposition, as well as finding as many hidden components in the data as possible. Intuitively, we are seeking a decomposition that discovers as many latent components as possible, without sacrificing the quality of those components. Essentially, we have a multiobjective optimization problem, where we need to maximize both c_i and R_i. We use the following, parameter-free, two-step maximization algorithm that gives an intuitive *data-driven* solution:

- **Max c step**: Run 2-means clustering on vector $\mathbf{c}$. This divides the coefficients of $\mathbf{c}$ into a set of good/high values and a set of low/bad ones. If m_1, m_2 are the centroids of the two clusters, we choose the cluster index that corresponds to the maximum between m_1 and m_2.
- **Max R step**: Given the cluster of points with maximum centroid, we select the point that maximizes the value of R. We call this point (R^*, c^*).

After choosing the "best" points (R^*_{Fro}, c^*_{Fro}) and (R^*_{KL}, c^*_{KL}), we have to select between the results of $\text{PARAFAC}_{\text{Fro}}$ and $\text{PARAFAC}_{\text{KL}}$. In order do so, there are a number of strategies to follow:

1. Calculate $s_{Fro} = \sum_r \mathbf{c}_{Fro}(r)$ and $s_{KL} = \sum_r \mathbf{c}_{KL}(r)$, and select the method that gives the largest sum. The intuition behind this data-driven strategy is choosing the loss function that is able to discover results with higher quality on aggregate, for more potential ranks.
2. Select the results that produce the maximum value between c^*_{Fro} and c^*_{KL}. This strategy is conservative and aims for the highest quality of

results, possibly to the expense of components of lesser quality that could still be acceptable for exploratory analysis.

3. Select the results that produce the maximum value between R^*_{Fro} and R^*_{KL}. Contrary to the previous strategy, this one is more aggressive, aiming for the highest number of components that can be extracted with acceptable quality.

Empirically, the last strategy seems to give better results, however, they all perform very closely in synthetic data. Particular choice of strategy depends on the application needs, for example, if quality of the components is imperative to be high, then strategy 2 should be preferred over strategy 3.

Note here that AUTOTEN not only seeks to find a good number of components for the decomposition, combining the best of both worlds of $\text{PARAFAC}_{\text{Fro}}$ and $\text{PARAFAC}_{\text{KL}}$, but is also able to provide quality assessment for the decomposition; if for a given R_{max} there is no good solution that captures the structure in the data, the user will be able to tell because of the very low c^* value.

Effectiveness

In this section, we empirically measure AUTOTEN's ability to uncover the true number of components hidden in a tensor. We create synthetic tensors of size $50 \times 50 \times 50$ where we control the exact number of latent components R_o, ranging from two to five. We compare AUTOTEN against four baselines:

- **Baseline 1**: A Bayesian tensor decomposition approach, as introduced very recently in Ref.[30], which automatically determines the rank.
- **Baseline 2**: For R ranging from 1 to $2R_o$, we run $\text{PARAFAC}_{\text{Fro}}$ and measure the Frobenius norm loss for each solution. If for two consecutive iterations the error does not improve more than a small positive number $\in$ (equal to 10^{-6} here), we output the result of the previous iteration.
- **Baseline 3**: Same as Baseline 2; here we use $\text{PARAFAC}_{\text{KL}}$ and instead of the Frobenius norm, we measure the log likelihood, and we stop when it stops improving more than $\in$.
- **Baseline 4**: A Bayesian framework based on automatic relevance determination that is adapted to the rank estimation of PARAFAC.[31]

AUTOTEN as well as Baselines 2 and 3 require a maximum bound $R_{\max}$ on the rank; for fairness, we set $R_{\max} = 2R_o$ for all three methods. In Figure 5 we show the result. We measure the error as $|R_{est} - R_o|$, where R_{est} is the estimated number of components by each method. Due to the randomized nature of the synthetic data generation, we ran 100 iterations and we show the average results. Our results are statistically significant (with $p < 0.01$ using a two-sided sign test). We observe that AUTOTEN generally achieves lower error in estimating the true number of components in synthetic data that emulates realistic tensor mining applications. The problem at hand is an extremely hard one, and we are not expecting any *tractable* method to solve it perfectly. Thus, the results we obtain here are very encouraging and show that AUTOTEN is a framework that can be adopted by Big Data practitioners.

Case study

In the previous section, we demonstrated the effectiveness of AUTOTEN in synthetic data. Here we demonstrate how it can help a Big Data practitioner in using tensor decompositions for real-world applications. In Ref.[28] we analyze a human mobility data set that records the location of taxis in the city of Beijing for an entire week and here we present our results from point of view of a Big Data practitioner. The data were first introduced in Refs.[13,32] and are available for download.[33] The raw format of the data is (latitude, longitude, minute), which conceptually forms a three-mode tensor. Before we can actually form the tensor, we need to discretize the GPS coordinates. In our first attempt, we created a 1000×1000 grid to discretize the coordinates and used AUTOTEN to find a good solution. However, the best solution returned by AUTOTEN had extremely low quality, which was an indication that the particular choice of grid resulted in poorly structured

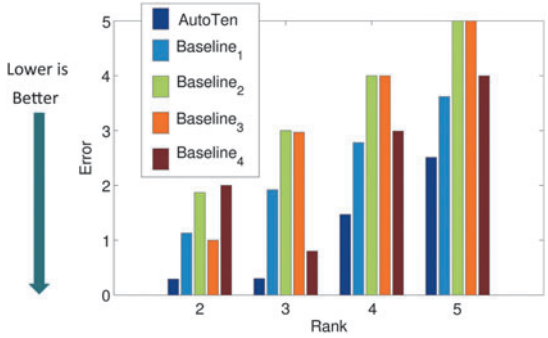


FIG. 5. AUTOTEN outperforms all baselines in estimating the true number of components in a tensor.

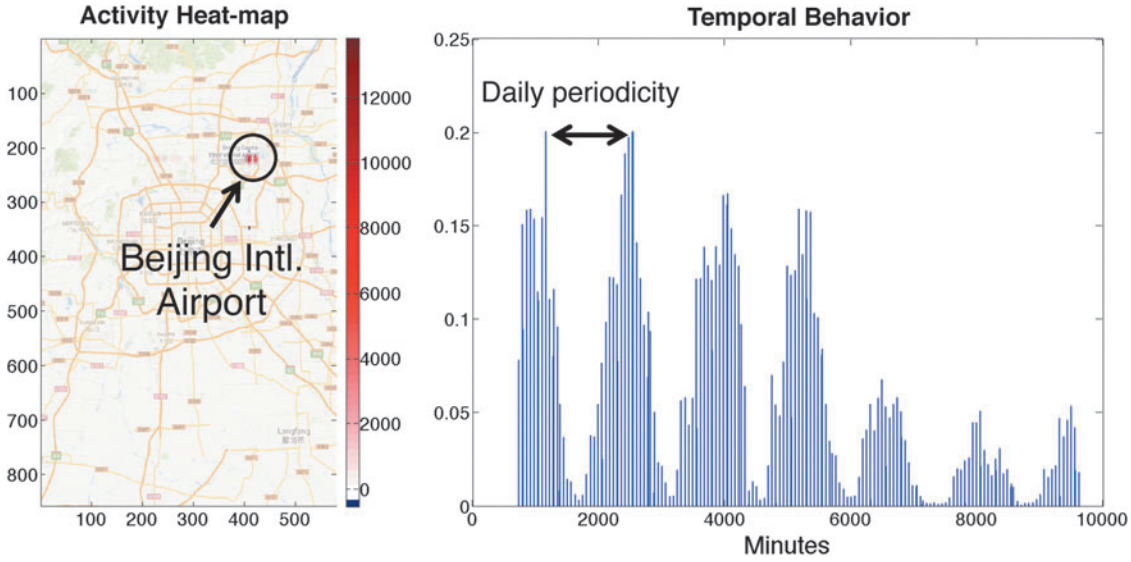


FIG. 6. High activity of taxis concentrated around Beijing's International airport (shown in the heat map on the left); the temporal activity of this co-cluster exhibits a daily periodicity that interestingly fades toward the weekend, a fact probably caused by reduced traffic to and from the airport those days.

data (perhaps extremely sparse data for which there was barely any dense structure within). We subsequently formed a 100×100 grid and AUTOTEN was able to detect good structure. In particular, AUTOTEN output eight rank-one components, choosing Frobenius norm as a loss function.

In Figure 6, we show one of the latent components of the PARAFAC decomposition. Each latent component here can be seen as a *hotspot of taxi activity*, since it contains the coordinates and the time for which there is high activity in the city. The first two modes (vectors **a** and **b**) correspond to the latitude and longitude, and we thus project them back on the map of Beijing. The intensity of activity is denoted by the color map in the figure (where deep red indicates high activity). The third mode vector (**c**) holds the temporal profile of that hotspot. The particular hotspot we show has a high spatial activity that is concentrated around Beijing's International airport. The temporal activity is also interesting since it exhibits a daily periodicity that fades toward the weekend, possibly because fewer people (particularly business people) travel during those days.

Our analysis is able to shed light into human mobility data by identifying hotspots of activity, which can in turn help policy makers understand traffic patterns in the city, reallocate the taxi fleet, and identify neighborhoods that are under- or overserved. Furthermore, such analysis can be used by the industry. For instance, Uber may be able to use historic data of rides to identify the temporal profiles of various hotspots and accordingly adapt their surge pricing scheme.

Conclusions

In this article, our aim is to popularize tensors and tensor decomposition to a wider audience of Big Data practitioners. To that end, we demonstrated their effectiveness in a wide variety of applications and we outlined the challenges posed by today's data. One major challenge is that of scaling up tensor decompositions has received considerable attention. However, the second major challenge of unsupervised quality assessment is fairly underexplored. We discussed heuristics that already exist in the literature and outlined their shortcomings with respect to their application in Big Data scenarios. We presented our recent contributions in tackling those shortcomings and demonstrated our data-driven proposed solution AUTOTEN, which is able to automate unsupervised tensor mining and is appropriate for Big Data practitioners.

Acknowledgments

This work was carried out while the first author was at Carnegie Mellon University. Research was supported by the National Science Foundation Grant No. IIS-1247489. 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.

Author Disclosure Statement

No competing financial interests exist.

References

1. Kolda TG, Bader BW. Tensor decompositions and applications. SIAM Rev. 2009;51:455–500.
2. Acar E, Yener B. Unsupervised multiway data analysis: A literature survey. IEEE Trans Knowl Data Eng. 2009;21:6–20.
3. Harshman RA. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. Los Angeles, CA: University of California at Los Angeles, 1970.
4. Papalexakis EE, Sidiropoulos ND, Bro R. From k-means to higher-way co-clustering: Multilinear decomposition with sparse latent factors. IEEE Trans Signal Process. 2013;61:493–506.
5. Tricap: ThRee-way methods in chemistry and psychology 7th edition, 2015. Available online at http://people.ece.umn.edu nikos/TRICAP_ home.html (last accessed May 21, 2016).
6. Workshop on tensor decompositions and applications. Available online at www.esat.kuleuven.be/stadius/TDA2016 (last accessed May 21, 2016).
7. Dagstuhl perspectives workshop: Tensor computing for internet of things. Available online at www.dagstuhl.de/en/program/calendar/semhp/?semnr=16152 (last accessed: May 21, 2016).
8. Kolda TG, Bader BW, Kenny JP. Higher-order web link analysis using multilinear algebra. In ICDM, New Orleans, LA, IEEE, 2005. 8 pp.
9. Kleinberg JM. Authoritative sources in a hyperlinked environment. J ACM. 1999;46:604–632.
10. Agrawal R, Golshan B, Papalexakis E. A study of distinctiveness in web results of two search engines. In 24th International Conference on World Wide Web, Web Science Track, Florence, Italy, ACM, May 2015.
11. Agrawal R, Golshan B, Papalexakis E. Whither social networks for web search? In 21st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Sydney, Australia, August 2015.
12. Papalexakis EE, Akoglu L, Ienco D. Do more views of a graph help? Community detection and clustering in multi-graphs. In FUSION, Istanbul, Turkey, IEEE, 2013. pp. 899–905.
13. Wang Y, Zheng Y, Xue Y. Travel time estimation of a path using sparse trajectories. In KDD, New York, NY, USA: ACM, 2014. pp. 25–34.
14. Acar E, Aykut-Bingol C, Bingol H, et al. Multiway analysis of epilepsy tensors. Bioinformatics. 2007;23:i10–i18.
15. Davidson I, Gilpin S, Carmichael O, Walker P. Network discovery via constrained tensor analysis of fmri data. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, ACM, 2013. pp. 194–202.
16. Papalexakis EE, Mitchell TM, Sidiropoulos ND, et al. Turbo-smt: Accelerating coupled sparse matrix-tensor factorizations by 200x. In SIAM SDM, 2014. pp. 118–126.
17. Maruhashi K, Guo F, Faloutsos C. Multiaspectforensics: Pattern mining on large-scale heterogeneous networks with tensor analysis. In Proceedings of the Third International Conference on Advances in Social Network Analysis and Mining. IEEE, 2011.
18. Mao CH, Wu CJ, Papalexakis EE, et al. Malspot: Multi2 malicious network behavior patterns analysis. In Advances in Knowledge Discovery and Data Mining, Tainan, Taiwan, Springer, 2014. pp. 1–14.
19. Ho JC, et al. Marble: High-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In KDD, New York, NY, ACM, 2014. pp. 115–124.
20. Bader BW, Kolda TG. Efficient MATLAB computations with sparse and factored tensors. SIAM J Sci Comput. 2007;30:205–231.
21. Kang U, et al. Gigatensor: Scaling tensor analysis up by 100 times-algorithms and discoveries. In KDD, Beijing, China, ACM, 2012. pp. 316–324.
22. Papalexakis EE, Faloutsos C, Sidiropoulos ND. Parcube: Sparse parallelizable tensor decompositions. In Machine Learning and Knowledge Discovery in Databases, Bristol, UK, Springer, 2012. pp. 521–536.
23. Erdos D, Miettinen P. Walk'n'merge: A scalable algorithm for Boolean tensor factorization. In ICDM, Dallas, TX, IEEE, 2013. pp. 1037–1042.
24. Sidiropoulos N, Papalexakis E, Faloutsos C. Parallel randomly compressed cubes: A scalable distributed architecture for big tensor decomposition. IEEE Signal Process Mag. 2014;31:57–70.
25. Aggour KS, Yener B. A parallel PARAFAC implementation and scalability testing for large-scale dense tensor decomposition. Technical report, Rensselaer Polytechnic Institute. 2016.
26. Bro R, Kiers HAL. A new efficient method for determining the number of components in PARAFAC models. J Chemom. 2003;17:274–286.
27. Papalexakis EE, Faloutsos C. Fast efficient and scalable core consistency diagnostic for the PARAFAC decomposition for big sparse tensors. In ICASSP. IEEE, 2015.
28. Papalexakis EE. Automatic unsupervised tensor mining with quality assessment. In SIAM SDM, 2016.
29. Chi EC, Kolda TG. On tensors, sparsity, and nonnegative factorizations. SIAM J Matrix Anal Appl. 2012;33:1272–1299.
30. Zhao Q, Zhang, L, Cichocki A. Bayesian CP factorization of incomplete tensors with automatic rank determination. IEEE Trans Pattern Anal Mach Intell. 2015;37:1751–1763.
31. Mørup M, Hansen LK. Automatic relevance determination for multi-way models. J Chemom. 2009;23:352–363.
32. Yuan J, et al. Driving with knowledge from the physical world. In KDD, San Diego, CA, ACM, 2011. pp. 316–324.
33. T-Drive trajectory data sample. Available online at http://research .microsoft.com/apps/pubs/?id=152883 (accessed August 29, 2016).
34. Kruskal JB. Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. Linear Algebra Appl. 1977;18:95–138.
35. Stegeman A, ten Berge JMF, De Lathauwer L. Sufficient conditions for uniqueness in CANDECOMP/PARAFAC and INDSCAL with random component matrices. Psychometrika. 2006;71:219–229.
36. Chiantini L, Ottaviani G. On generic identifiability of 3-tensors of small rank. SIAM J Matrix Anal Appl. 2012;33:1018–1037.
37. Bader BW, Kolda TG, et al. Matlab tensor toolbox version 2.6. Available online at www.sandia.gov/~tgkolda/TensorToolbox/ (accessed August 29, 2016).
38. Andersson CA, Bro R. The n-way toolbox for matlab. Chemom Intell Lab Syst. 2000;52:1–4.
39. Heiser WJ. Convergent computation by iterative majorization: Theory and applications in multidimensional data analysis. In: Recent Advances in Descriptive Multivariate Analysis. Clarendon Press, 1995. pp. 157–189.
40. Févotte C, Idier J. Algorithms for nonnegative matrix factorization with the β-divergence. Neural Comput. 2011;23:2421–2456.

Cite this article as: Papalexakis EE, Faloutsos C (2016) Unsupervised tensor mining for big data practitioners. *Big Data* 4:3, 179–191, DOI: 10.1089/big.2016.0026.

Abbreviation Used

EEG = electroencephalogram
GPS = Global Positioning System
HITS = Hyperlink-Induced Topic Search

(*Appendix follows* →)

Appendix 1

Theoretical Advantages of Using Tensors

Compared against techniques that work on two-mode data (matrices), PARAFAC analysis enjoys very fortuitous theoretical properties that make it a very powerful exploratory tool. Specifically, the PARAFAC decomposition of a tensor is *unique* up to scaling and permutation of the components.[34–36] In plain terms, this means that when we have computed the decomposition, we have guarantees that there is no other set of factors that yield the same approximation to the tensor; this is the unique set of factors that decompose the data, and thus, we can interpret them without risking ignoring another, equivalent set of factors that may yield a different clustering in our data. To the contrary, matrix decompositions suffer from rotational ambiguity. In particular, given a matrix $\mathbf{X}$, we can decompose it into $\mathbf{AB}^T$, however, the following equivalence of solutions holds:

$$\mathbf{X} \approx \mathbf{AB}^T = \mathbf{AQQ}^{-1}\mathbf{B}^T = \tilde{\mathbf{A}}\tilde{\mathbf{B}}^T$$

which implies that using the above model for exploratory purposes may be misleading, since there are multiple solutions that have the same reconstruction error, but impose widely different clustering on the data.

Extending the Core Consistency Diagnostic

In the main text, we outlined the two major challenges faced by the core consistency diagnostic in Big Data applications. Here we provide the details of our contributions.

Scalable core consistency diagnostic

Although simple and elegant, the solution of the least squares problem that lies in the heart of CORCONDIA suffers in the case of high-dimensional data. In particular, this straightforward solution requires to first compute and store $(\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C})$ and then pseudoinvert it. Consider a $10^5 \times 10^5 \times 10^5$ tensor; even for an extremely low rank decomposition of $R = 10$, the aforementioned Kronecker product will be of size $10^{15} \times 10^3$, a fact that renders computing and storing such a matrix highly impractical (if not outright impossible), and subsequently, computing its pseudoinverse intractable. We stress that the main problem here is that materializing $(\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C})$ is what makes computation intractable. Therefore, even if we substituted the pseudoinverse with a more efficient computation of a least squares problem (such as the conjugate gradient method), this would still be intractable, since we cannot materialize matrix $(\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C})$.

Our "wish-list" of properties for our core consistency diagnostic algorithm is as follows:

- Avoid materializing any Kronecker product.
- Avoid directly pseudoinverting the (potentially huge) aforementioned Kronecker product.
- Exploit any sparse structure in the factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and/or the tensor $\underline{\mathbf{X}}$.

To achieve the above, we need to reformulate the computation of the core consistency diagnostic, according to the following Lemma.

LEMMA 1 ([27]) *The pseudoinverse* $(\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C})^\dagger$ *can be rewritten as:*

$$(\mathbf{V_a} \otimes \mathbf{V_b} \otimes \mathbf{V_c})\left(\Sigma_\mathbf{a}{}^{-1} \otimes \Sigma_\mathbf{b}{}^{-1} \otimes \Sigma_\mathbf{c}{}^{-1}\right)\left(\mathbf{U_a}{}^T \otimes \mathbf{U_b}{}^T \otimes \mathbf{U_c}{}^T\right)$$

where $\mathbf{A} = \mathbf{U_a}\Sigma_\mathbf{a}\mathbf{V_a}^\mathbf{T}$, $\mathbf{B} = \mathbf{U_b}\Sigma_\mathbf{b}\mathbf{V_b}^\mathbf{T}$, *and* $\mathbf{C} = \mathbf{U_c}\Sigma_\mathbf{c}\mathbf{V_c}^\mathbf{T}$ (*i.e., the respective Singular Value Decompositions*).

Given the above Lemma, we rewrite the solution for the core consistency diagnostic as follows:

$$vec(\underline{\mathbf{G}}) = (\mathbf{V_a} \otimes \mathbf{V_b} \otimes \mathbf{V_c})\left(\Sigma_\mathbf{a}{}^{-1} \otimes \Sigma_\mathbf{b}{}^{-1} \otimes \Sigma_\mathbf{c}{}^{-1}\right)$$

$$\left(\mathbf{U_a}{}^T \otimes \mathbf{U_b}{}^T \otimes \mathbf{U_c}{}^T\right) vec(\underline{\mathbf{X}})$$

Notice that the above equation can be broken down into a series of three operations:

$$\mathbf{z} = \left(\mathbf{U_a}{}^T \otimes \mathbf{U_b}{}^T \otimes \mathbf{U_c}{}^T\right) vec(\underline{\mathbf{X}})$$

$$\mathbf{w} = \left(\Sigma_\mathbf{a}{}^{-1} \otimes \Sigma_\mathbf{b}{}^{-1} \otimes \Sigma_\mathbf{c}{}^{-1}\right)\mathbf{z}$$

$$vec(\underline{\mathbf{G}}) = (\mathbf{V_a} \otimes \mathbf{V_b} \otimes \mathbf{V_c})\mathbf{w}$$

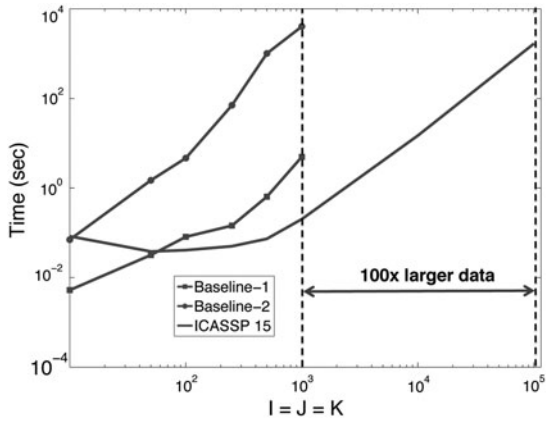
Each one of the above operations is in the following form:

$$(\mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \cdots \otimes \mathbf{A}_k)\mathbf{x}.$$

which we henceforth refer to as

$$\text{KRONMATVEC}\ (\mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \cdots \otimes \mathbf{A}_k), \mathbf{x}$$

and the key property of this operation is that it can be done *without materializing* $(\mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \cdots \otimes \mathbf{A}_k)$. What we have achieved is we have derived a mathematically equivalent expression for computing the core consistency diagnostic, which carries out the computations



Appendix FIG. 1. Computation of the core consistency diagnostic for two orders of magnitude larger data.

"on-the-fly," and does not suffer from the explosion of the intermediate Kronecker product matrix.

We implemented our algorithm in MATLAB, using the Tensor Toolbox,[37,†] which provides efficient manipulation and storage of sparse tensors. For comparisons, we used two baselines:

- **Baseline 1**: Implementation of the PLS Toolbox,[‡] which is commercial software and is considered the state of the art for computing CORCONDIA.
- **Baseline 2**: Implementation of the N-way Toolbox for MATLAB,[38,§] which is freely available and open source.

All experiments were run on a workstation with 4 Intel(R) Xeon(R) E7-8837 and 1TB of RAM.

Appendix Figure 1 shows the execution time as a function of the tensor mode dimension I for $I \times I \times I$ for sparse synthetic tensors with I nonzero values. We clearly see that our proposed algorithm is generally much faster and more scalable than both baselines (note that the figures are in log-scales), while it keeps working for $I = 10^5$ (*two orders of magnitude larger data*) where the baselines run out of memory.

Core consistency diagnostic for sparse count data

Recent seminal work[29] has shown that postulating a Poisson distribution in tensors that record sparse count data is more beneficial in capturing the structure in the data. The authors of Ref.[29] introduce a version of the PARAFAC decomposition that does that, referred to as $\text{PARAFAC}_{\text{KL}}$.

To solve for $\text{PARAFAC}_{\text{KL}}$, which postulates a Poisson distribution in the data, we have to minimize the KL-divergence of the tensor $\underline{\mathbf{X}}$ and the PARAFAC model. The KL-divergence is defined as follows:

$$D_{KL}(p||q) = \sum_i p(i) \log \frac{p(i)}{q(i)}$$

This is in contrast to the traditional PARAFAC decomposition that minimizes the Frobenius norm of the error (henceforth referred to as $\text{PARAFAC}_{\text{Fro}}$). To do the same for the core consistency diagnostic, we need to minimize the following:

$$\min_{\mathbf{x}} D_{KL}(\mathbf{y}||\mathbf{W}\mathbf{x}), \mathbf{W} = \mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C}. \quad (1)$$

and in our case $\mathbf{y} = vec(\underline{\mathbf{G}})$ and $\mathbf{x} = vec(\underline{\mathbf{X}})$.

Unlike the Frobenius norm case, where the solution to the problem is the least squares estimate, in the KL-divergence case, the problem does not have a closed form solution. Instead, iterative solutions apply. The most prominent approach to this problem is via an optimization technique called *Majorization–Minimization* (MM) or *Iterative Majorization*.[39] In a nutshell, in MM, given a function that is hard to minimize directly, we derive a "majorizing" function, which is always greater than the function to be minimized, except for a support point where it is equal; we minimize the majorizing function and iteratively update the support point using the minimizer of that function. This procedure converges to a local minimum. For the problem of Equation (1),[29,40] use the following update rule for the problem, which is used iteratively until convergence to a stationary point.

$$\mathbf{x}(j)^{(k)} = \mathbf{x}(j)^{(k-1)} \left(\frac{\sum_i \mathbf{W}(i,j) \left(\frac{\mathbf{y}(j)}{\tilde{\mathbf{y}}(j)^{(k-1)}} \right)}{\sum_i \mathbf{W}(i,j)} \right) \quad (2)$$

where $\tilde{\mathbf{y}}^{(k-1)} = \mathbf{W}\mathbf{x}^{(k-1)}$, and k denotes the k-th iteration index.

The above solution is generic for any structure of $\mathbf{W}$. Remember, however, that $\mathbf{W}$ has very specific Kronecker structure that we should exploit. Recall the previous subsection, where we gave an example of how large this matrix $\mathbf{W} = \mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C}$ can get, for tensors of moderately large size, and how this can affect scalability. We thus need to exploit this special structure. In particular, we break down Equation (2) into pieces, each one that can be computed efficiently, given the structure of $\mathbf{W}$ using KronMatVec operations.

[†]Tensor Toolbox available at: www.sandia.gov tgkolda/TensorToolbox/
[‡]PLS Toolbox available at: www.eigenvector.com/software/pls_toolbox.htm
[§]N-way Toolbox available at: www.models.life.ku.dk/nwaytoolbox

The first step is to decompose the expression of the numerator of Equation (2). In particular, we equivalently write

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} * \mathbf{z}_2$$

where

$$\mathbf{z}_2 = \mathbf{W}^T \mathbf{z}_1$$

and $\mathbf{z}_1 = \mathbf{y} \oslash \tilde{\mathbf{y}}$, where $*$ denotes element-wise multiplication, and $\oslash$ denotes element-wise division.

Due to the Kronecker structure of $\mathbf{W}$:

$$\mathbf{Z}_2 = \text{KRONMATVEC}(\{\mathbf{A}^T, \mathbf{B}^T, \mathbf{C}^T\}, \mathbf{z}_1)$$

Therefore, the update to $\mathbf{x}^{(k)}$ is efficiently calculated in the above three steps. The normalization factor of the equation is equal to $\mathbf{s}(j) = \sum_i \mathbf{W}(i,j)$. Given the Kronecker structure of $\mathbf{W}$, however, the following holds:

LEMMA 2 ([28]) *The row sum of a Kronecker product matrix* $\mathbf{A} \otimes \mathbf{B}$ *can be rewritten as* $\left(\sum_{i=1}^{I} \mathbf{A}(i,:)\right) \otimes \left(\sum_{j=1}^{J} \mathbf{B}(j,:)\right)$

Thus,

$$\mathbf{s} = \left(\sum_i \mathbf{A}(i,:)\right) \otimes \left(\sum_j \mathbf{B}(j,:)\right) \otimes \left(\sum_n \mathbf{C}(n,:)\right).$$

By manipulating the MM update formula as we showed in the last few lines, we can compute the core consistency diagnostic using the KL-divergence efficiently, without suffering from scalability problems.