Automatic Unsupervised Tensor Mining with Quality Assessment

Evangelos E. Papalexakis
Carnegie Mellon University
epapalex@cs.cmu.edu

Abstract

Tensor decomposition has been very popular in unsupervised modelling and multi-aspect data mining. In an exploratory setting, where no labels or ground truth are available how can we automatically decide how many components to extract? How can we assess the quality of our results, so that a domain expert can factor this quality measure in the interpretation of our results? In this paper, we introduce AUTOTEN, a novel automatic unsupervised tensor mining algorithm with minimal user intervention, which leverages and improves upon heuristics that assess the result quality. We extensively evaluate AUTOTEN's performance on synthetic data, outperforming existing baselines on this very hard problem. Finally, we apply AUTOTEN to a variety of real datasets, providing insights and discoveries.

1 Introduction

Tensor decompositions and their applications in mining multi-aspect datasets are ubiquitous and ever increasing in popularity. Data Mining application of these techniques has been largely pioneered by [21] where the authors introduce a topical aspect to a graph between webpages, and extend the popular HITS algorithm in that scenario. Henceforth, the field of multi-aspect/tensor data mining has witnessed rich growth with prime examples of applications being social networks [22, 25, 18], citation networks [22], and computer networks [22], to name a few.

Tensor decompositions are undoubtedly a very powerful analytical tool with a rich variety of applications. However there exist research challenges in the field of data mining that need to be addressed, in order for tensor decompositions to claim their position as a de-facto tool for practitioners.

One challenge, which has received considerable attention, is the one of making tensor decompositions scalable to today's web scale. However, recently and especially for sparse tensors, there has been a substantial body of work with the first pioneering step by Kolda et al. [21, 2] exploiting sparsity for scalability; subsequently there have been distributed approaches based on the latter formulation [19], and other scalable approaches [30, 11]. 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 main focus of this work, however, is on another, relatively less explored territory; that of assessing the *quality* of a tensor decomposition. In a great portion of tensor data mining, the task is exploratory and unsupervised: we are given a dataset, usually without any sort of ground truth, and we seek to extract *interesting* patterns or concepts from the data. It is crucial, therefore, to know whether a pattern that we extract actually models the data at hand, or whether it is merely modelling noise in the data. Especially in the age of Big Data, where feature spaces can be vast, it is imperative to have a measure of quality and avoid interpreting noisy, random variation that always exists in the data. Determining the number of components in a tensor is a very hard problem [16]. This is why, many seminal exploratory tensor mining papers, understandably, set the number of components manually [21, 32, 22]. When there is a specific task at hand, e.g. link prediction [9], recommendation [31], and supervised learning [14], that entails some measure of success, then we can use cross-validation for selecting a good number of latent components which unfortunately cannot generalize to the case where labels or ground truth are absent.

However, not all hope is lost. There exists highly influential work in the Chemometrics literature [5] that introduces heuristics for determining the quality of a decomposition, taking into account properties of the PARAFAC decomposition [13] and being *application independent*, requiring no prior knowledge about the data. Inspired by and drawing from [5], we provide a comprehensive method for mining large and potentially sparse multi-aspect datasets using tensor decompositions.

Our contributions are:

- **Technology Transfer** To the best of our knowledge, this is the first data mining work that employs the Core Consistency Diagnostic for the quality assessment of a tensor decomposition; our sincere hope is to popularize such approaches in the data mining community, conducting a technology transfer from the Chemometrics community.
- **Algorithms** We propose AUTOTEN, a comprehensive, parallelizable methodology on mining multi-aspect datasets using tensors, which minimizes manual trial-and-error intervention and provides quality characteri-

zation of the solution (Section 3.2). Furthermore, we extend the Core Consistency Diagnostic of [5] assuming KL-divergence loss, which is more effective in modelling highly sparse, count data [7] (Section 3.1).

- **Evaluation & Discovery** We conduct a large scale study on 10 real datasets, exploring the structure of hidden patterns within these datasets (Section 5.1). To the best of our knowledge, this is the first such broad study. As a data mining case study, we use AUTOTEN on real data discovering meaningful patterns (Section 5.2 and supplementary material[1]). Finally, we extensively evaluate our proposed method on synthetic data (Section 4).

In order to encourage **reproducibility**, most of the datasets used are public, and we make our code publicly available at `http://www.cs.cmu.edu/~epapalex/src/AutoTen.zip`.

2 Background

Table 1 provides an overview of the notation used in this and subsequent sections.

Symbol	Definition
$\underline{\mathbf{X}}, \mathbf{X}, \mathbf{x}, x$	Tensor, matrix, column vector, scalar
$\circ$	Outer product
$vec(\,)$	Vectorization operator
$\otimes$	Kronecker product
$*\ \oslash$	Element-wise multiplication and division
$\mathbf{A}^\dagger$	Moore-Penrose Pseudoinverse of $\mathbf{A}$
$D_{KL}(a\|\|b)$	KL-Divergence
$\|\mathbf{A}\|_F$	Frobenius norm
KRONMATVEC	Efficient computation of $\mathbf{y} = (\mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \cdots \otimes \mathbf{A}_n)\,\mathbf{x}$ [6]
$\mathbf{x}(i)$	i-th entry of $\mathbf{x}$ (same for matrices and tensors)
$\mathbf{X}(:,i)$	Spans the entire i-th column of $\mathbf{X}$ (same for tensors)
$\mathbf{x}^{(k)}$	Value at the k-th iteration
CP_ALS	Frobenius norm PARAFAC [3]
CP_APR	KL-Divergence PARAFAC [7]

Table 1: Table of symbols

2.1 Brief Introduction to Tensor Decompositions

Given a tensor $\underline{\mathbf{X}}$, we can decompose it according to the CP/PARAFAC decomposition [13] (henceforth referred to as PARAFAC) as a sum of rank-one tensors:

$$\underline{\mathbf{X}} \approx \sum_{f=1}^{F} \mathbf{a}_f \circ \mathbf{b}_f \circ \mathbf{c}_f$$

where the (i,j,k) entry of $\mathbf{a}_f \circ \mathbf{b}_f \circ \mathbf{c}_f$ is $\mathbf{a}_f(i)\mathbf{b}_f(j)\mathbf{c}_f(k)$. Usually, PARAFAC is represented in its matrix form $[\mathbf{A}, \mathbf{B}, \mathbf{C}]$, where the columns of matrix $\mathbf{A}$ are the $\mathbf{a}_f$ vectors (and accordingly for $\mathbf{B}, \mathbf{C}$). The PARAFAC decomposition is especially useful when we are interested in extracting the true latent factors that generate the tensor. In this work, we choose the PARAFAC decomposition as our tool, since it admits a very intuitive interpretation of its latent factors: each component can be seen as soft co-clustering of the tensor, using the high values of vectors $\mathbf{a}_f, \mathbf{b}_f, \mathbf{c}_f$ as the membership values to co-clusters.

Another very popular Tensor decomposition is Tucker3 [23], where a tensor is decomposed into outer product of factor vectors multiplied by a core tensor:

$$\underline{\mathbf{X}} \approx \sum_{p=1}^{P}\sum_{q=1}^{Q}\sum_{r=1}^{R} \underline{\mathbf{G}}(p,q,r)\mathbf{u}_p \circ \mathbf{v}_q \circ \mathbf{w}_r.$$

The Tucker3 model is especially used for compression. Furthermore, PARAFAC can be seen as a restricted Tucker3 model, where the core tensor $\underline{\mathbf{G}}$ is super-diagonal, i.e. non-zero values are only in the entries where $i = j = k$ and $p = q = r$.

2.2 Brief Introduction to Core Consistency Diagnostic

As outlined in the Introduction, in the chemometrics literature, there exists a very intuitive heuristic by the name of Core Consistency Diagnostic or CORCONDIA (henceforth used interchangeably) [5], which can serve as a guide in judging how well a given PARAFAC decomposition is modelling a tensor. In a nutshell, the idea behind the Core Consistency Diagnostic is the following: Given a tensor $\underline{\mathbf{X}}$ and its PARAFAC decomposition $\mathbf{A}, \mathbf{B}, \mathbf{C}$, one could imagine fitting a Tucker3 model where matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are the factors of the Tucker3 decomposition and $\underline{\mathbf{G}}$ is the core tensor (which we need to solve for). Since, as we already mentioned, PARAFAC can be seen as a restricted Tucker3 decomposition with super-diagonal core tensor, if our PARAFAC modelling of $\underline{\mathbf{X}}$ using $\mathbf{A}, \mathbf{B}, \mathbf{C}$ is modelling the data well, the core tensor $\underline{\mathbf{G}}$ should be as close to super-diagonal as possible. If there are deviations from the super-diagonal, then this is a good indication that our PARAFAC model is somehow flawed (either the decomposition rank is not appropriate, or the data do not have the appropriate structure). We can pose the problem as the following least squares problem:

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

with the least squares solution:

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

After computing $\underline{\mathbf{G}}$, the Core Consistency diagnostic can be computed as $c = 100\left(1 - \frac{\sum_{i=1}^{F}\sum_{j=1}^{F}\sum_{k=1}^{F}\left(\underline{\mathbf{G}}(i,j,k) - \underline{\mathbf{I}}(i,j,k)\right)^2}{F}\right)$, where $\underline{\mathbf{I}}$ is a super-diagonal tensor with ones on the (i,i,i) entries. For a perfectly super-diagonal $\underline{\mathbf{G}}$ (i.e. perfect modelling), c will be 100. One can see that for rank-one models, the metric will always be 100, because the rank one component can trivially produce a single element "super-diagonal" core; thus, CORCONDIA is applicable

[1] `http://www.cs.cmu.edu/~epapalex/papers/sdm16-autoten-supplementary.pdf`

for rank two or higher. According to [5], values below 50 show some imperfection in the modelling or the rank selection; the value can also be negative, showing severe problems with the modelling. In [5], some of the chemical data analyzed have perfect, low rank PARAFAC structure, and thus expecting $c > 50$ is reasonable. In many data mining applications, however, due to high data sparsity, data cannot have such perfect structure, but an *approximation* thereof using a low rank model is still very valuable. Thus, in our case, we expand the spectrum of acceptable solutions with reasonable quality to include smaller, positive values of c (e.g. 20 or higher).

2.3 Scaling Up CORCONDIA As we mention in the Introduction, CORCONDIA as introduced in [4] is suitable for small and dense data. However, this contradicts the area of interest of the vast majority of data mining applications. To that end, very recently the authors of [28] extended CORCONDIA to the case where the data are large but sparse, deriving a fast and efficient algorithm. Key behind [28] is avoiding to pseudoinvert $(\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C})$ and alternatively carrying out a series of very efficient KRONMATVEC computations [6]: $\mathbf{y} = (\mathbf{A}_1 \otimes \mathbf{A}_2 \otimes \cdots \otimes \mathbf{A}_n)\,\mathbf{x}$.

What [28] has achieved thus far is extending the CORCONDIA to large and sparse data, assuming Frobenius norm loss. This assumption postulates that the underlying data distribution is Gaussian. However, recently [7] showed that for sparse data that capture counts (e.g. number of messages exchanged), it is more beneficial to postulate a Poisson distribution, therefore using the KL-Divergence as a loss function. This has been more recently adopted in [17] showing very promising results in medical applications. Therefore, one natural direction, which we follow in the first part of the next section, is to extend CORCONDIA for this scenario.

3 Proposed Methods

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

In order to attack the above problem, first, in Section 3.1 we describe how we can derive a fast and efficient measure of quality for KL-Divergence loss. Finally, in 3.2, we introduce AUTOTEN, our unified algorithm for automatic tensor mining with minimal user intervention, and quality characterization of the solution.

3.1 Quality Assessment with KL-Divergence As we saw in the description of the Core Consistency Diagnostic with Frobenius norm loss, its computation requires solving a least squares problem. In the case of the `CP_APR` modelling, where the loss function is the KL-Divergence, the minimization problem that we need to solve is:

$$\min_{\mathbf{x}} D_{KL}(\mathbf{y}||\mathbf{W}\mathbf{x}),\ \mathbf{W} = \mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C}. \tag{3.1}$$

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* [15]. 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 Eq. 3.1, [12] and subsequently [7], employ the following update rule for the problem, which is used iteratively until convergence to a stationary point.

$$\mathbf{x}(j)^{(k)} = \mathbf{x}(j)^{(k-1)} \left(\frac{\sum_i \mathbf{W}(i,j)\left(\frac{\mathbf{y}(j)}{\tilde{\mathbf{y}}(j)^{(k-1)}}\right)}{\sum_i \mathbf{W}(i,j)}\right) \tag{3.2}$$

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 which we should exploit. Additionally, suppose that we have a $10^4 \times 10^4 \times 10^4$ tensor; then, the large dimension of $\mathbf{W}$ will be 10^{12}. If we attempt to materialize, store, and use $\mathbf{W}$ throughout the algorithm, it will be catastrophic to the algorithm's performance. We can exploit the Kronecker structure of $\mathbf{W}$ so that we break down Eq. 3.2 into pieces, each one which can be computed efficiently, given the structure of $\mathbf{W}$. The first step is to decompose the expression of the numerator of Eq. 3.2. 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 $\mathbf{z}_1 = \mathbf{y} \oslash \tilde{\mathbf{y}}$. 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 three above 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:

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

Proof. We can rewrite the row sums $\sum_{i=1}^{I} \mathbf{A}(i,:) = \mathbf{i}_I^T \mathbf{A}$ and $\sum_{j=1}^{J} \mathbf{B}(j,:) = \mathbf{i}_J^T \mathbf{B}$ where $\mathbf{i}_I$ and $\mathbf{i}_J$ are all-ones column vectors of size I and J respectively. For the Kronecker product of the row sums and by using properties of the Kronecker product, and calling $\mathbf{A} \otimes \mathbf{B} = \mathbf{W}$ we have

$$\left(\mathbf{i}_I^T \mathbf{A}\right) \otimes \left(\mathbf{i}_J^T \mathbf{B}\right) = \left(\mathbf{i}_I \otimes \mathbf{i}_J\right)^T (\mathbf{A} \otimes \mathbf{B}) = \mathbf{i}_{IJ}^T \mathbf{W} = \sum_{i=1}^{IJ} \mathbf{W}(i,:)$$

which concludes the proof. ■

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

Putting everything together, we end up with Algorithm 1 which is an efficient solution to the minimization problem of Equation 3.1. As in the naive case, we also use Iterative Majorization in the efficient algorithm; we iterate updating $\mathbf{x}^{(k)}$ until we converge to a local optimum or we reach a maximum number of iterations. After computing the core tensor, we then calculate the Core Consistency Diagnostic as before, by measuring its deviation from the super-diagonal tensor.

Algorithm 1: Efficient Majorization Minimization for solving $\min_{\mathbf{x}} D_{KL}(\mathbf{y} || (\mathbf{A} \otimes \mathbf{B} \otimes \mathbf{C})\,\mathbf{x})$

Input: Vector $\mathbf{y}$ and matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$.
Output: Vector $\mathbf{x}$

```
Initialize x^(0) randomly
ỹ = KronMatVec({A, B, C}, x^(0))
s = (Σ_i A(i,:)) ⊗ (Σ_j B(j,:)) ⊗ (Σ_n C(n,:))
while convergence criterion is not met do
    z_1 = y ⊘ ỹ
    z_2 = KronMatVec({A^T, B^T, C^T}, z_1)
    x^(k) = x^(k-1) * z_2
    ỹ = KronMatVec({A, B, C}, x^(k))
end while
Normalize x^(k) using s
```

3.2 AutoTen: Automated Unsupervised Tensor Mining

At this stage, we have the tools we need in order 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 towards making tensor mining a fully automated tool, used as a black box by academic and industrial practitioners. 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. A sketch of AUTOTEN follows, and is also outlined in Algorithm 2.

Solution Search The user provides a data tensor, as well as a maximum rank that reflects the budget that she is willing to devote to AUTOTEN's search. We neither have nor require any prior knowledge whether the tensor is highly sparse, or dense, contains real values or counts, hinting whether we should use, say, `CP_ALS` postulating Frobenius norm loss, or `CP_APR` postulating KL-Divergence loss.

Fortunately, our work in this paper, as well as our previous work [28] has equipped us with tools for handling all of the above cases. Thus, we follow a data-driven approach, where we let the data show us whether using `CP_ALS` or `CP_APR` is capturing better structure. For a grid of values for the decomposition rank (bounded by the user provided maximum rank), we run both `CP_ALS` and `CP_APR`, and we record the quality of the result as measured by the Core Consistency diagnostic into vectors $\mathbf{c}_{Fro}$ and $\mathbf{c}_{KL}$ truncating negative values to zero.

Result Selection At this point, for both `CP_ALS` and `CP_APR` we have points in two dimensional space (F_i, c_i), reflecting the quality and the corresponding number of components. Given points (F_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 multi-objective optimization problem, where we need to maximize both c_i and F_i. However, if we, say, get the Pareto front of those points (i.e. the subset of all non-dominated points), we end up with a family of solutions without a clear guideline on how to select one. We propose to use the following, effective, two-step maximization algorithm that gives an intuitive *data-driven* solution:

- **Max c step**: Given vector $\mathbf{c}$, run 2-means clustering on its values. This will essentially divide the vector into a set of good/high values and a set of low/bad ones. If we call m_1, m_2 the means of the two clusters, then we select the cluster index that corresponds to the maximum between m_1 and m_2.
- **Max F step**: Given the cluster of points with maximum mean, we select the point that maximizes the value of F. We call this point (F^*, c^*).

After choosing the "best" points (F^*_{Fro}, c^*_{Fro}) and (F^*_{KL}, c^*_{KL}), at the final step of AUTOTEN, we have to select between the results of `CP_ALS` and `CP_APR`. In order to do so, we select the results that produce the maximum value between F^*_{Fro} and F^*_{KL}. For more potential strategies, we refer the reader to the supplementary material.

Discussion As we've mentioned above, the maximum number of components F_{max} is chosen by the user according to the computational budget. However, there also exist rigorous theoretical bounds on F_{max} that can help guide the

Algorithm 2: AUTOTEN: Automatic Unsupervised Tensor Mining

Input: Tensor $\underline{\mathbf{X}}$ and maximum budget for component search F_{max}
Output: PARAFAC decomposition $\mathbf{A}, \mathbf{B}, \mathbf{C}$ of $\underline{\mathbf{X}}$ and corresponding quality metric c^*.
1: **for** $f = 2 \cdots F_{max}$, in parallel **do**
2: Run `CP_ALS` for f components. Update $\mathbf{c}_{Fro}(f)$ using Algorithm in [28].
3: Run `CP_APR` for f components. Update $\mathbf{c}_{KL}(f)$ using Algorithm 1.
4: **end for**
5: Find (F^*_{Fro}, c^*_{Fro}) and (F^*_{KL}, c^*_{KL}) using the two-step maximization as described in the text.
6: Choose between `CP_ALS` and `CP_APR` using the strategy described in the text.
7: Output the chosen c^* and the corresponding decomposition.

choice. In particular, the main result in [8] states that for a tensor of dimensions $I \times J \times K$, assuming $I \leq J \leq K$, the maximum number of components that can be *uniquely* identified using the PARAFAC decomposition is $F_{max} \leq \frac{(I+1)(J+1)}{16}$, which is an upper bound to the choice of the F_{max} parameter in AUTOTEN. We point out that lines 2-3 of Algorithm 2, i.e. all the F_{max} computations, can be run *entirely in parallel*, speeding up the computation of each individual decomposition. Finally, it is important to note that AUTOTEN not only seeks to find a good number of components for the decomposition, combining the best of both worlds of `CP_ALS` and `CP_APR`, but furthermore is able to provide quality assessment for the decomposition: if for a given F_{max} none of the solutions that AUTOTEN sifts through yields a satisfactory result, the user will be able to tell because of the very low (or zero in extreme cases) c^* value.

4 Experimental Evaluation

We implemented AUTOTEN in Matlab, using the Tensor Toolbox [3], which provides efficient manipulation and storage of sparse tensors. We use the public implementation for the algorithm of [28], and we make our code publicly available[2]. The online version of our code contains a test case that uses the same code that we used for the following evaluation. All experiments were carried out on a workstation with 4 Intel(R) Xeon(R) E7- 8837 and 512Gb of RAM.

4.1 Evaluation on Synthetic Data 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 $50 \times 50 \times 50$ in the two following ways that model realistic data mining scenarios where we have highly sparse data: 1) using the function `create_problem` of the Tensor Toolbox for Matlab as a standardized means of generating synthetic tensors, we generate sparse random factors with integer values, with total number of non-zeros equal to 500, 2) following the synthetic data generation methodology of [7], which generates poisson distributed sparse factors. We generate these for true rank F_o ranging from 2-5.

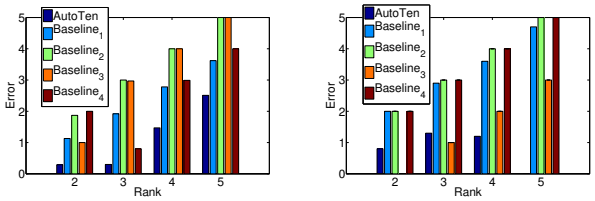
We compare AUTOTEN against four baselines:

- **Baseline 1**: A Bayesian tensor decomposition approach, as introduced very recently in [36] which automatically determines the rank.
- **Baseline 2**: This is a very simple heuristic approach where, for a grid of values for the rank, we run `CP_ALS` and record the Frobenius norm loss for each solution. If for two consecutive iterations the loss does not improve more than a small positive number ϵ (set to 10^{-6} here), we declare as output the result of the previous iteration.
- **Baseline 3**: Same as Baseline 2 with sole difference being that we use `CP_APR` and accordingly instead of the Frobenius norm reconstruction error, we measure the log-likelihood, and we stop when it stops improving more than ϵ. We expect Baseline 3 to be more effective than Baseline 2 in sparse data, due to the more delicate and effective treatment of sparse, count data by `CP_APR`.
- **Baseline 4**: A Bayesian framework based on Automatic Relevance Determination (ARD) that is adapted to the rank estimation of PARAFAC and Tucker models [27]. According to [27] this baseline works comparably to Core Consistency in the cases the authors examined.

AUTOTEN as well as Baselines 2 & 3 require a maximum bound F_{max} on the rank; for fairness, we set $F_{max} = 2F_o$ for all three methods. In Figures 1(a) and 1(b) we show the results for both test cases. The error is measured as $|F_{est} - F_o|$ where F_{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. We calculated statistical significance of our results ($p < 0.01$) using a two-sided sign test. We observe that AUTOTEN generally achieves lower error in estimating the true number of components. There is a single instance in Fig. 1(b) where the log likelihood criterion (Baseline 3) works slightly better than the proposed method, and this is probably because the criterion of Baseline 3 is highly in sync with the generative model of the synthetic data, however, overall we conclude that AUTOTEN largely outperforms the baselines 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 practical

[2]Download our code at `http://www.cs.cmu.edu/~epapalex/src/AutoTen.zip`

solution that, as we demonstrate in the next Section, can help data mining practitioners.



(a) Sparse count data with integer factors (b) Data generated as described in [7]

Figure 1: Rank estimation error on synthetic data.

5 Data Mining Case Study

Section 5.1 takes 10 diverse real datasets shown in Table 2 and investigates their rank structure. In Section 5.2 we apply AUTOTEN to one of the datasets of Table 2 and we analyze the results, as part of an exploratory data mining study.

5.1 Rank Structure of Real Datasets Since exploration of the rank structure of a dataset, using the Core Consistency diagnostic, is an integral part of AUTOTEN, we deem necessary to dive deeper into that process. In this case study we are analyzing the rank structure of 10 real datasets, as captured by the Core Consistency under Frobenius norm loss (using our algorithm from [28], as well as Core Consistency with KL-Divergence loss (introduced here). Most of the datasets we use are *publicly available*. `ENRON`[3] is a social network dataset, recording the number of emails exchanged between employees of the company for a period of time, during the company crisis. `Reality Mining` [10] is a multi-view social network dataset, recording relations between MIT students (who calls whom, who messages whom, who is close to whom and so on). `Facebook` [33] is a time evolving snapshot of Facebook, recording people posting on other peoples' Walls. `Taxi`[4] is a dataset of taxi trajectories in Beijing; we discretize latitude and longitude to a 100×100 grid. `DBLP` is a dataset recording which researcher to researcher connections, from three different viewpoints (first view is co-authorship, second view is citation, and third view records whether two authors share at least three keywords in their title or abstract of their papers). `Netflix` comes from the Netflix prize dataset and records movie ratings by users over time. `Amazon co-purchase` data records items bought together, and the category of the first of the two products. `Amazon metadata` records customers who reviewed a product, and the corresponding product category. `Yelp` contains reviews of Yelp users for various businesses (from the data challenge[5]). Finally, `Airport`[6] contains records of flights between different airports, and the operating airline.

We ran our algorithms for $F = 2 \cdots 50$, and truncated negative values to zero. For KL-Divergence and datasets `Facebook`, `Netflix`, `Yelp`, and `Airport` we used smaller versions (first 500 rows for `Netflix` and `Yelp`, and first 1000 rows for `Facebook` and `Airport`), due to high memory requirements of Matlab; this means that the corresponding figures describe the rank structure of a smaller dataset, which might be different from the full one. Figure 2 shows the Core Consistency when using Frobenius norm as a loss, and Fig. 3 when using KL-Divergence. The way to interpret these figures is the following: assuming a `CP_ALS` (Fig. 2) or a `CP_APR` (Fig. 3) model, each figure shows the modelling quality of the data for a given rank. This sheds light to the rank structure of a particular dataset (although that is not to say that it provides a definitive answer about its true rank). For the given datasets, we observe a few interesting differences in structure: for instance, `ENRON` and `Taxi` in Fig. 2 seem to have good quality for a few components. On the other hand, `Reality Mining`, `DBLP`, and `Amazon metadata` have reasonably acceptable quality for a larger range of components, with the quality decreasing as the number gets higher. Another interesting observation is that `Yelp` seems to be modelled better using a high number of components. Figures that are all-zero merely show that no good structure was detected for up to 50 components, however, this might indicate that such datasets (e.g. `Netflix`) have an even higher number of components. Finally, contrasting Fig. 3 to Fig. 2, we observe that in many cases using the KL-Divergence is able to discover better structure than the Frobenius norm (e.g. `ENRON` and `Amazon co-purchase`).

5.2 AutoTen in practice We used AUTOTEN to analyze the `Taxi` dataset shown in Table 2.The data we have span an entire week worth of measurements, with temporal granularity of minutes. First, we tried quantizing the latitude and longitude into a 1000×1000 grid; however, AUTOTEN warned us that the decomposition was not able to detect good and coherent structure in the data, perhaps due to extreme sparsity. Subsequently, we modelled the data using a 100×100 grid and AUTOTEN was able to detect good structure. In particular, AUTOTEN output 8 rank-one components, choosing Frobenius norm as a loss function.

In Figure 4 we show 4 representative components of the decomposition. In each sub-figure, we overlay the map of Beijing with the coordinates that appear to have high activity in the particular component; every sub-figure also shows the temporal profile of the component. The first two components

[3] `http://www.cs.cmu.edu/~enron/`

[4] `http://research.microsoft.com/apps/pubs/?id=152883`

[5] `https://www.yelp.com/dataset_challenge/dataset`

[6] `http://openflights.org/data.html`

Table 2: Datasets analyzed

Name	Description	Dimensions	Number of nonzeros
ENRON	(sender, recipient, month)	$186 \times 186 \times 44$	9838
Reality Mining [10]	(person, person, means of communication)	$88 \times 88 \times 4$	5022
Facebook [33]	(wall owner, poster, day)	$63891 \times 63890 \times 1847$	737778
Taxi [35, 34]	(latitude, longitude,minute)	$100 \times 100 \times 9617$	17762489
DBLP [29]	(paper, paper, view)	$7317 \times 7317 \times 3$	274106
Netflix	(movie, user, date)	$17770 \times 252474 \times 88$	50244707
Amazon co-purchase [24]	(product, product, product group)	$256 \times 256 \times 5$	5726
Amazon metadata [24]	(product, customer, product group)	$10000 \times 263011 \times 5$	441301
Yelp	(user, business, term)	$43872 \times 11536 \times 10000$	10009860
Airport	(airport, airport, airline)	$9135 \times 9135 \times 19305$	58443

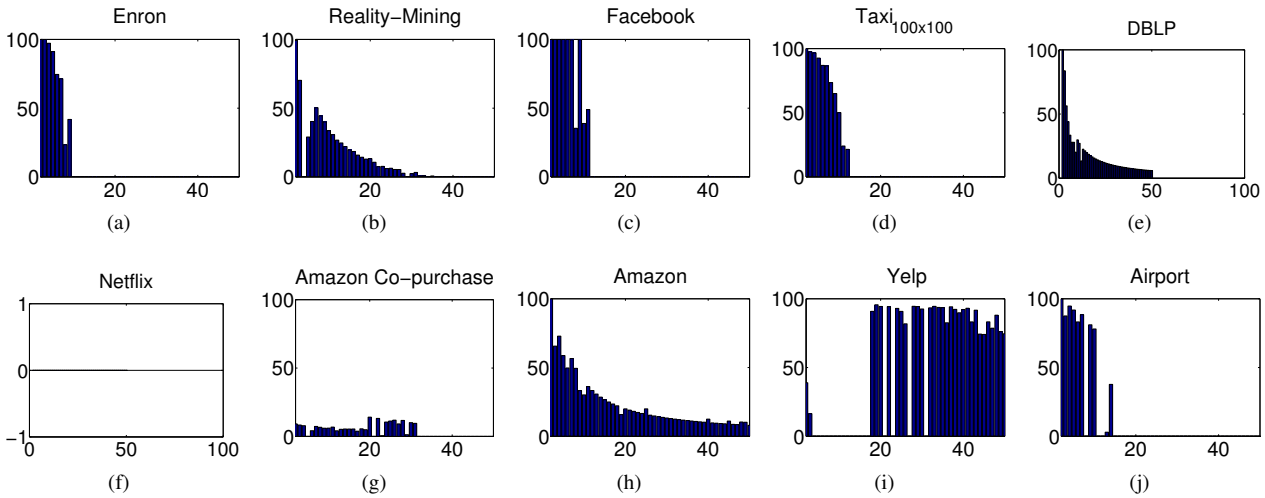


Figure 2: Core Consistency for CP_ALS

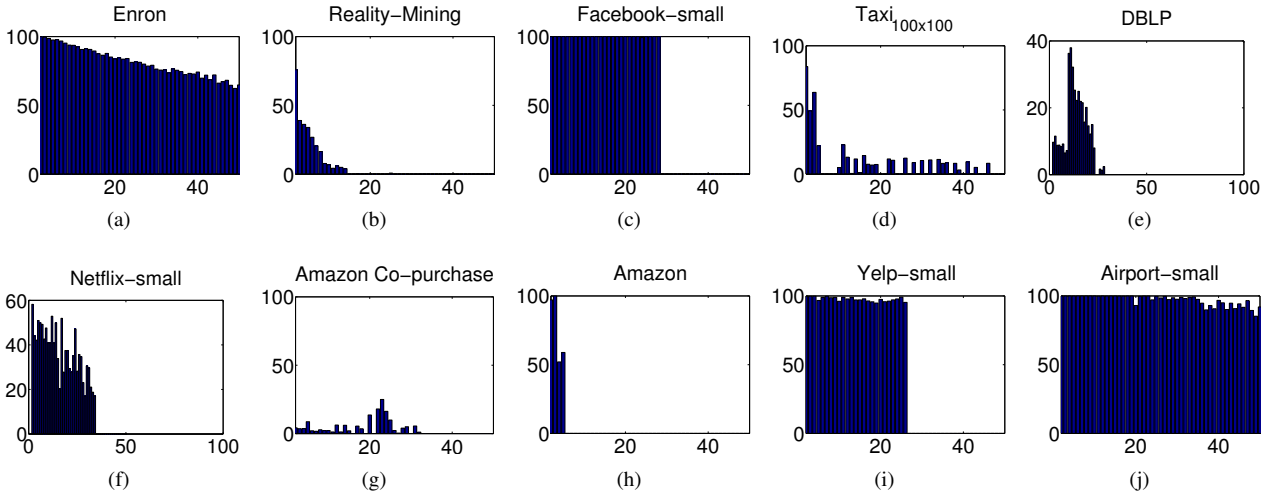
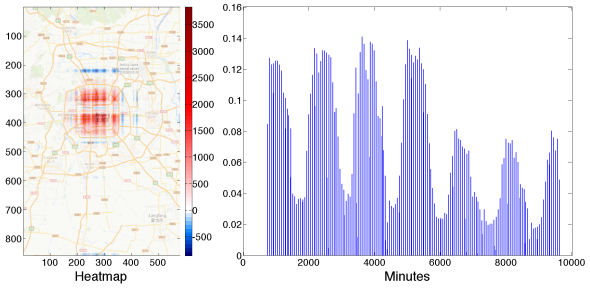
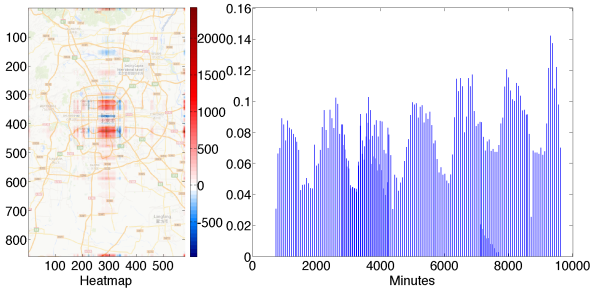


Figure 3: Core Consistency for CP_APR

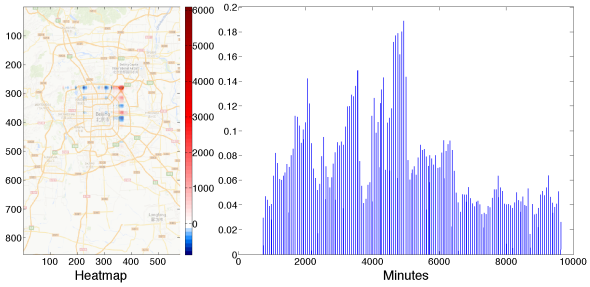
(Fig. 4(a), (b)) spatially refer to a similar area, roughly corresponding to the tourist and business center in the central rings of the city. The difference is that Fig. 4(a) shows high activity during the weekdays and declining activity over the weekend (indicated by five peaks of equal height, followed by two smaller peaks), whereas Fig. 4(b) shows a slightly inverted temporal profile, where the activity peaks over the weekend; we conclude that Fig. 4(a) most likely captures business traffic that peaks during the week, whereas Fig. 4(b) captures tourist and leisure traffic that peaks over the weekend. The third component (Fig. 4(c)) is highly active around the Olympic Center and Convention Center area, with peaking activity in the middle of the week. Finally, the last component (Fig. 4(d)) shows high activity only outside of Beijing's international airport, where taxis gather to pick-up customers; on the temporal side, we see daily



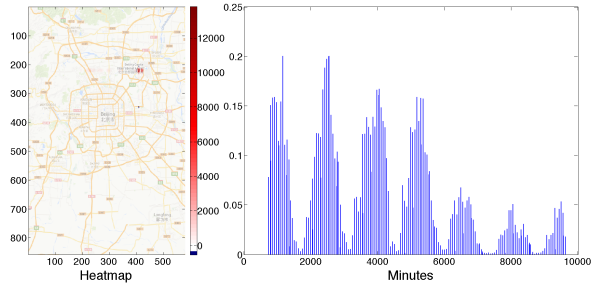
(a) **Tourist & Business Center**: High activity during weekdays, low over the weekend



(b) **Downtown**: Consistent activity throughout the week



(c) **Olympic Center**: Activity peak during the week



(d) **Airport**: High activity during weekdays, low over the weekend

Figure 4: Latent components of the `Taxi` dataset, as extracted using AUTOTEN.

peaks of activity, with the collective activity dropping during the weekend, when there is significantly less traffic of people coming to the city for business. By being able to analyze such trajectory data into highly interpretable results, we can help policymakers to better understand the traffic patterns of taxis in big cities, estimate high and low demand areas and times and optimize city planning in that respect. There has been very recent work [34] towards the same direction, and we view our results as complementary.

6 Related Work

Model order selection for Tensors As we have mentioned throughout the text, CORCONDIA [5] is using properties of the PARAFAC decomposition in order to hint towards the right number of components. In [28], the authors introduce a scalable algorithm for CORCONDIA (under the Frobenius norm). Moving away from the PARAFAC decompostion, Kiers and Kinderen [20] introduce a method for choosing the number of components for Tucker3. There has been recent work using Minimum Description Length (MDL): In [1] the authors use MDL in the context of community detection in time-evolving social network tensors, whereas in [26], Metzler and Miettinen use MDL to score the quality of components for a binary tensor factorization. Finally, there have also been recent advances using Bayesian methods in order to automatically decide the number of components: in particular [36] does so for the PARAFAC decomposition, and [27] (based on Automatic Relevance Determination) does so for both PARAFAC and Tucker models.

7 Conclusions

In this paper, we work towards an automatic, unsupervised tensor mining algorithm that minimizes user intervention. We encourage *reproducibility* by making our code publicly available at `http://www.cs.cmu.edu/~epapalex/src/AutoTen.zip`. Our main contributions are:

- **Technology Transfer** This is the first work to apply ideas such as the Core Consistency Diagnostic [5] in data mining, aiming to popularize it within the community.
- **Algorithms** We propose AUTOTEN, a novel automatic, parallel, and unsupervised tensor mining algorithm, which can provide quality characterization of the solution. We extend the Core Consistency Diagnostic of [5] for KL-Divergence loss and provide a novel and efficient algorithm.
- **Evaluation & Discovery** We evaluate our methods in synthetic data, showing their superiority compared to the baselines, as well as a wide variety of real datasets. Finally, we apply AUTOTEN to two real datasets discovering meaningful patterns (Section 5.2 and supplementary material).

Acknowledgements

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. The author would like to thank Professors Christos Faloutsos, Nicholas Sidiropoulos, and Rasmus Bro for valuable conversations, and Miguel Araujo for sharing the `Airport` dataset.

References

[1] Miguel Araujo et al. Com2: Fast automatic discovery of temporal ('comet') communities. In *Advances in Knowledge Discovery and Data Mining*, pages 271–283. Springer, 2014.

[2] Brett W. Bader and Tamara G. Kolda. Efficient MATLAB computations with sparse and factored tensors. *SIAM Journal on Scientific Computing*, 30(1):205–231, December 2007.

[3] Brett W. Bader, Tamara G. Kolda, et al. Matlab tensor toolbox version 2.6. Available online, February 2015.

[4] Rasmus Bro. *Multi-way analysis in the food industry: models, algorithms, and applications*. PhD thesis, 1998.

[5] Rasmus Bro and Henk AL Kiers. A new efficient method for determining the number of components in PARAFAC models. *Journal of chemometrics*, 17(5):274–286, 2003.

[6] Paul E Buis and Wayne R Dyksen. Efficient vector and parallel manipulation of tensor products. *ACM Transactions on Mathematical Software (TOMS)*, 22(1):18–23, 1996.

[7] Eric C Chi and Tamara G Kolda. On tensors, sparsity, and nonnegative factorizations. *SIAM Journal on Matrix Analysis and Applications*, 33(4):1272–1299, 2012.

[8] Luca Chiantini and Giorgio Ottaviani. On generic identifiability of 3-tensors of small rank. *SIAM Journal on Matrix Analysis and Applications*, 33(3):1018–1037, 2012.

[9] Daniel M Dunlavy, Tamara G Kolda, and Evrim Acar. Temporal link prediction using matrix and tensor factorizations. *TKDD*, 5(2):10, 2011.

[10] Nathan Eagle, Alex Sandy Pentland, and David Lazer. Inferring friendship network structure by using mobile phone data. *Proceedings of the National Academy of Sciences*, 106(36):15274–15278, 2009.

[11] Dóra Erdos and Pauli Miettinen. Walk'n'merge: A scalable algorithm for Boolean tensor factorization. In *ICDM*, pages 1037–1042. IEEE, 2013.

[12] Cédric Févotte and Jérôme Idier. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural Computation*, 23(9):2421–2456, 2011.

[13] R.A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis. 1970.

[14] Lifang He et al. Dusk: A dual structure-preserving kernel for supervised tensor learning with applications to neuroimages. In *SDM*. SIAM, 2014.

[15] Willem J Heiser. Convergent computation by iterative majorization: theory and applications in multidimensional data analysis. *Recent advances in descriptive multivariate analysis*, pages 157–189, 1995.

[16] Christopher J Hillar and Lek-Heng Lim. Most tensor problems are NP-hard. *Journal of the ACM (JACM)*, 60(6):45, 2013.

[17] Joyce C Ho et al. Marble: high-throughput phenotyping from electronic health records via sparse nonnegative tensor factorization. In *KDD*, pages 115–124. ACM, 2014.

[18] Meng Jiang et al. Fema: flexible evolutionary multi-faceted analysis for dynamic behavioral pattern discovery. In *KDD*, pages 1186–1195. ACM, 2014.

[19] U Kang et al. Gigatensor: scaling tensor analysis up by 100 times-algorithms and discoveries. In *KDD*, pages 316–324. ACM, 2012.

[20] Henk AL Kiers and Albert Kinderen. A fast method for choosing the numbers of components in Tucker3 analysis. *British Journal of Mathematical and Statistical Psychology*, 56(1):119–125, 2003.

[21] Tamara G Kolda, Brett W Bader, and Joseph P Kenny. Higher-order web link analysis using multilinear algebra. In *ICDM*, pages 8–pp. IEEE, 2005.

[22] Tamara G Kolda and Jimeng Sun. Scalable tensor decompositions for multi-aspect data mining. In *ICDM*, pages 363–372. IEEE, 2008.

[23] Pieter M Kroonenberg and Jan De Leeuw. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, 45(1):69–97, 1980.

[24] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[25] Yu-Ru Lin et al. Metafac: community discovery via relational hypergraph factorization. In *KDD*, pages 527–536. ACM, 2009.

[26] Saskia Metzler and Pauli Miettinen. Clustering Boolean tensors. *Data Mining and Knowledge Discovery 29(5)*, page 13431373, 2015.

[27] Morten Mørup and Lars Kai Hansen. Automatic relevance determination for multi-way models. *Journal of Chemometrics*, 23(7-8):352–363, 2009.

[28] Evangelos E. . Papalexakis and C. Faloutsos. Fast efficient and scalable core consistency diagnostic for the PARAFAC decomposition for big sparse tensors. In *ICASSP*. IEEE, 2015.

[29] Evangelos E Papalexakis, Leman Akoglu, and Dino Ienco. Do more views of a graph help? community detection and clustering in multi-graphs. In *FUSION*, pages 899–905. IEEE, 2013.

[30] Evangelos E Papalexakis, Christos Faloutsos, and Nicholas D Sidiropoulos. Parcube: Sparse parallelizable tensor decompositions. In *Machine Learning and Knowledge Discovery in Databases*, pages 521–536. Springer, 2012.

[31] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *WSDM*, pages 81–90. ACM, 2010.

[32] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *KDD*, pages 374–383. ACM, 2006.

[33] Bimal Viswanath et al. On the evolution of user interaction in facebook. In *ACM SIGCOMM Workshop on Social Networks (WOSN'09)*, August 2009.

[34] Yilun Wang, Yu Zheng, and Yexiang Xue. Travel time estimation of a path using sparse trajectories. In *KDD*, pages 25–34, New York, NY, USA, 2014. ACM.

[35] Jing Yuan et al. Driving with knowledge from the physical world. In *KDD*, pages 316–324. ACM, 2011.

[36] Q Zhao, L Zhang, and A Cichocki. Bayesian CP factorization of incomplete tensors with automatic rank determination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37:1751 – 1763, 2015.