Clustering Boolean Tensors

Saskia Metzler and Pauli Miettinen

Max Planck Institut für Informatik
Saarbrücken, Germany
{saskia.metzler, pauli.miettinen}@mpi-inf.mpg.de

Abstract. Tensor factorizations are computationally hard problems, and in particular, often are significantly harder than their matrix counterparts. In case of Boolean tensor factorizations – where the input tensor and all the factors are required to be binary and we use Boolean algebra – much of that hardness comes from the possibility of overlapping components. Yet, in many applications we are perfectly happy to partition at least one of the modes. In this paper we investigate what consequences does this partitioning have on the computational complexity of the Boolean tensor factorizations and present a new algorithm for the resulting clustering problem. While future work aims at further tuning our algorithm for Boolean tensor clustering, it already now can obtain better results than algorithms solving different relaxations of the problem.

1 Introduction

Tensors become increasingly popular data representations in data mining. Ternary (or higher order) relations, for instance, can be represented as binary 3-way (or multi-way) tensors. Given such data, the question is whether there is any underlying structure or regularity in the data. To approach that question, typically tensor decomposition methods are applied. In this work, we restrict ourselves to binary data and also restrict the factors in the decomposition to be binary.

Tensor decompositions with similar restrictions have previously been studied: Cerf et al. [3] present an algorithm for the extraction of noise-tolerant itemsets in binary relations. Erdős and Miettinen [6] propose a scalable algorithm for Boolean CANDECOMP/PARAFAC (CP) and Tucker decompositions, and apply it to information extraction [5].

The novelty of the Boolean tensor decomposition approach we present is the restriction to non-overlapping factors in one mode. This takes apart complexity from the task and also often fits the structure of real-world data. For example in subject–relation–object data, the relations are non-overlapping: While a subject can be linked to multiple objects and vice versa, the relation is a property of the link between them. The algorithm we present has better approximability results, is simpler than previous algorithms, and also outperforms them.

Existing work relates to different aspects of our approach: Jegelka et al. [9] study the problem of clustering simultaneously all modes of a tensor (tensor co-clustering). In the context of formal concept analysis, Belohlavek et al. [2] use

triadic concepts to obtain an optimal decomposition of three-way binary data. That approach is extended to approximate solutions by Ignatov et al. [8]. Huang et al. [7] and Liu et al. [12] (among others) study the problem where only one mode is clustered and the remaining modes are represented using a low-rank approximation. The latter form is closer to what we study in this paper, but the techniques used in the continuous methods do not apply to the binary case.

2 Preliminaries

Throughout this paper, we indicate vectors as bold lower-case letters ($\boldsymbol{v}$), matrices as bold upper-case letters ($\boldsymbol{M}$), and tensors as bold upper-case calligraphic letters ($\boldsymbol{\mathcal{T}}$). Element (i, j, k) of a 3-way tensor $\boldsymbol{\mathcal{X}}$ is denoted as x_{ijk}. A colon in a subscript denotes taking that mode entirely; for example, $\boldsymbol{X}_{::k}$ is the kth *frontal slice* of $\boldsymbol{\mathcal{X}}$ (shorthand $\boldsymbol{X}_k$).

A tensor can be *unfolded* into a matrix by arranging its fibers (i.e. its columns, rows, or tubes in case of a 3-way tensor) as columns of a matrix. For a *mode-n matricization*, mode-n fibers are used as the columns and the result is $\boldsymbol{X}_{(n)}$.

The outer product of vectors is denoted by $\boxtimes$. For vectors $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$ of length n, m, and l, $\boldsymbol{\mathcal{X}} = \boldsymbol{a} \boxtimes \boldsymbol{b} \boxtimes \boldsymbol{c}$ is an n-by-m-by-l tensor with $x_{ijk} = a_i b_j c_k$.

The *Boolean tensor sum* of binary tensors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ is defined as $(\boldsymbol{\mathcal{X}} \vee \boldsymbol{\mathcal{Y}})_{ijk} = x_{ijk} \vee y_{ijk}$. For binary matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ where $\boldsymbol{X}$ has r columns and $\boldsymbol{Y}$ has r rows their *Boolean matrix product*, $\boldsymbol{X} \circ \boldsymbol{Y}$, is defined as $(\boldsymbol{X} \circ \boldsymbol{Y})_{ij} = \bigvee_{k=1}^{r} x_{ik} y_{kj}$. The *Boolean matrix rank* of a binary matrix $\boldsymbol{A}$ is the least r such that there exists a pair of binary matrices $(\boldsymbol{X}, \boldsymbol{Y})$ of inner dimension r with $\boldsymbol{A} = \boldsymbol{X} \circ \boldsymbol{Y}$.

Definition 1 (Boolean tensor rank). *The* Boolean rank *of a 3-way binary tensor $\boldsymbol{\mathcal{X}}$,* $\operatorname{rank}_B(\boldsymbol{\mathcal{X}})$*, is the least integer r such that there exist r triplets of binary vectors ($\boldsymbol{a}_i$, $\boldsymbol{b}_i$, $\boldsymbol{c}_i$) with $\boldsymbol{\mathcal{X}} = \bigvee_{i=1}^{r} \boldsymbol{a}_i \boxtimes \boldsymbol{b}_i \boxtimes \boldsymbol{c}_i$.*

A binary matrix $\boldsymbol{X}$ is a *cluster assignment matrix* if each row of $\boldsymbol{X}$ has exactly one non-zero element. In that case the Boolean matrix product corresponds to the regular matrix product, $\boldsymbol{X} \circ \boldsymbol{Y} = \boldsymbol{X}\boldsymbol{Y}$.

For a tensor $\boldsymbol{\mathcal{X}}$, $|\boldsymbol{\mathcal{X}}|$ denotes its number of non-zero elements. The Frobenius norm of a 3-way tensor $\boldsymbol{\mathcal{X}}$ is $\|\boldsymbol{\mathcal{X}}\| = \sqrt{\sum_{i,j,k} x_{ijk}^2}$. If $\boldsymbol{\mathcal{X}}$ is binary, $|\boldsymbol{\mathcal{X}}| = \|\boldsymbol{\mathcal{X}}\|^2$. The *similarity* between two n-by-m-by-l binary tensors $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ is defined as $\mathsf{sim}(\boldsymbol{\mathcal{X}}, \boldsymbol{\mathcal{Y}}) = nml - |\boldsymbol{\mathcal{X}} - \boldsymbol{\mathcal{Y}}|$.

Let $\boldsymbol{X}$ be n_1-by-m_1 and $\boldsymbol{Y}$ be n_2-by-m_2 matrix. Their *Kronecker (matrix) product*, $\boldsymbol{X} \otimes \boldsymbol{Y}$, is the $n_1 n_2$-by-$m_1 m_2$ matrix defined by

$$\boldsymbol{X} \otimes \boldsymbol{Y} = \begin{pmatrix} x_{11}\boldsymbol{Y} & x_{12}\boldsymbol{Y} & \cdots & x_{1m_1}\boldsymbol{Y} \\ x_{21}\boldsymbol{Y} & x_{22}\boldsymbol{Y} & \cdots & x_{2m_1}\boldsymbol{Y} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n_1 1}\boldsymbol{Y} & x_{n_1 2}\boldsymbol{Y} & \cdots & x_{n_1 m_1}\boldsymbol{Y} \end{pmatrix}.$$

The *Khatri–Rao (matrix) product* of $\boldsymbol{X}$ and $\boldsymbol{Y}$ is defined as 'column-wise Kronecker'. That is, $\boldsymbol{X}$ and $\boldsymbol{Y}$ must have same number of columns ($m_1 = m_2 =$

m), and their Khatri–Rao product $\boldsymbol{X} \odot \boldsymbol{Y}$ is the $n_1 n_2$-by-m matrix defined as $\boldsymbol{X} \odot \boldsymbol{Y} = \big(\boldsymbol{x}_1 \otimes \boldsymbol{y}_1, \boldsymbol{x}_2 \otimes \boldsymbol{y}_2, \ldots, \boldsymbol{x}_m \otimes \boldsymbol{y}_m\big)$. Notice that if $\boldsymbol{X}$ and $\boldsymbol{Y}$ are binary, so are $\boldsymbol{X} \otimes \boldsymbol{Y}$ and $\boldsymbol{X} \odot \boldsymbol{Y}$.

The Boolean tensor CP decomposition mirrors the standard tensor CP decomposition.

Definition 2 (Boolean CP). *Given an n-by-m-by-l binary tensor $\boldsymbol{\mathcal{X}}$ and an integer r, find binary matrices $\boldsymbol{A}$ (n-by-r), $\boldsymbol{B}$ (m-by-r), and $\boldsymbol{C}$ (l-by-r) such that they minimize $|\boldsymbol{\mathcal{X}} - \bigvee_{i=1}^{r} \boldsymbol{a}_i \boxtimes \boldsymbol{b}_i \boxtimes \boldsymbol{c}_i|$.*

Following Kolda and Bader [11], we use $[[\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]]$ to denote the normal 3-way CP and $[[\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]]_B$ for the Boolean CP. We can also write the Boolean CP as matrices using unfolding. The matrix product has to be the Boolean matrix product while the Khatri–Rao product is closed under the Boolean algebra:

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

Both problems, finding the least error Boolean CP decomposition and deciding the Boolean tensor rank, are NP-hard [13].

3 Problem Definition

We consider the variation of *tensor clustering* where the idea is to cluster one mode of a tensor and potentially reduce the dimensionality of the other modes.

Assuming a 3-way tensor and that we do the clustering in the last mode, we can express the *Boolean CP clustering* (BCPC) problem as follows:

Definition 3 (BCPC). *Given a binary n-by-m-by-l tensor $\boldsymbol{\mathcal{X}}$ and an integer k, find matrices $\boldsymbol{A} \in \{0,1\}^{n \times k}$, $\boldsymbol{B} \in \{0,1\}^{m \times k}$, and $\boldsymbol{C} \in \{0,1\}^{l \times k}$ such that $\boldsymbol{C}$ is a cluster assignment matrix and that the tuple $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ maximizes $\mathsf{sim}(\boldsymbol{\mathcal{X}}, [[\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}]]_B)$*

To understand what BCPC does, we use the unfolding rules (1) and write $\boldsymbol{X}_{(3)} \approx \boldsymbol{C}(\boldsymbol{B} \odot \boldsymbol{A})^T$, where we can see that we have restricted the type of cluster centroids: While in a general clustering problem, we would aim to cluster the frontal slices of $\boldsymbol{\mathcal{X}}$ into k clusters each represented by an n-by-m matrix, in this setting each cluster representative has to be of type $(\boldsymbol{b} \otimes \boldsymbol{a})^T$. This restriction on the cluster centroids plays a crucial role in the decomposition, as we shall see shortly.

4 Solving Maximum-Similarity BCPC

Given a tensor $\boldsymbol{\mathcal{X}}$, for the optimal solution to BCPC, we need matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$ that maximize $\mathsf{sim}(\boldsymbol{X}_{(3)}, \boldsymbol{C}(\boldsymbol{B} \odot \boldsymbol{A})^T)$. If we replace $\boldsymbol{B} \odot \boldsymbol{A}$ with an arbitrary binary matrix, this would equal the hypercube segmentation problem defined in [1]: Given a set S of l vertices of the discrete d-dimensional cube $\{0,1\}^d$, find k vertices $P_1, \ldots, P_k \in \{0,1\}^d$ and a partition of S into k segments to maximize $\sum_{i=1}^{k} \sum_{c \in S} \mathsf{sim}(P_i, c)$. Therefore we employ an algorithm that resembles those for hypercube segmentation, with the added restrictions to our centroid vectors.

Algorithm 1 `SaBoTeur` algorithm for the BCPC

Input: A 3-way binary tensor $\boldsymbol{\mathcal{X}}$, number of clusters k, number of samples r.
Output: Binary factor matrices $\boldsymbol{A}$ and $\boldsymbol{B}$; cluster assignment matrix $\boldsymbol{C}$.

1: **function** `SaBoTeur`($\boldsymbol{\mathcal{X}}, k, r$)
2: **repeat**
3: Sample k rows of $\boldsymbol{X}_{(3)}$ into matrix $\boldsymbol{Y}$
4: Find binary matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ that maximize $\mathsf{sim}(\boldsymbol{Y}, (\boldsymbol{B} \odot \boldsymbol{A})^T)$
5: Cluster $\boldsymbol{C}$ by assigning each row of $\boldsymbol{X}_{(3)}$ to its closest row of $(\boldsymbol{B} \odot \boldsymbol{A})^T$
6: **until** r resamples are done
7: **return** best $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$
8: **end function**

4.1 The Algorithm

Alon et al. [1] gave an algorithm for the hypercube segmentation problem that obtains similarity within $(1 - \varepsilon)$ of the optimum. The running time of the algorithm is $\mathrm{e}^{O((k^2/\varepsilon^2)\ln k)}nml$ for n-by-m-by-l data. While technically linear in data size, the first term turns the running time unfeasible even for moderate values of k (the number of clusters) and ε. We therefore base our algorithm on the simpler algorithm by Kleinberg et al. [10] that is based on random sampling. This algorithm obtains an approximation ratio of $0.828 - \varepsilon$ with constant probability and running time $O(nmlk(9/\varepsilon)^k \ln(1/\varepsilon))$. While the running time is still exponential in k, it is dominated by the number of samples we do: each sample takes time $O(nmlk)$ for k clusters and n-by-m-by-l data. For practical purposes, we can keep the number of samples constant (with the cost of losing approximation guarantees, though).

Our algorithm `SaBoTeur` (SAmpling for BOolean TEnsor clUsteRing), Algorithm 1, considers only the unfolded tensor $\boldsymbol{X}_{(3)}$. In each iteration, it samples k rows of $\boldsymbol{X}_{(3)}$ as the initial, unrestricted centroids. It then turns these unrestricted centroids into the restricted type in line 4, and then assigns each row of $\boldsymbol{X}_{(3)}$ to its closest restricted centroid. The sampling is repeated multiple times, and in the end, the factors that gave highest similarity are returned.

The algorithm is extremely simple, which is an asset as it gives a very fast algorithm that, as we shall see in Section 5, also performs very well. In line 3 the algorithm samples k rows of the data as its initial centroids. Kleinberg et al. [10] proved that among the rows of $\boldsymbol{X}_{(3)}$ that are clustered into the same optimal cluster, one is a good approximation of the (unrestricted) centroid of the cluster. Intuitively, then, if we sample one row from each cluster, the sample has a high probability of inducing a close-optimal clustering.

4.2 Binary Rank-1 Matrix Decompositions

The final piece of the `SaBoTeur` algorithm is to turn the unrestricted centroids into the restricted format required by the BCPC problem (line 4). We start by showing that this problem is equivalent to finding the maximum-similarity binary rank-1 decomposition of a binary matrix:

Algorithm 2 Approximate maximum-similarity binary rank-1 decompositions

Input: An n-by-m binary matrix $\boldsymbol{X}$.
Output: Binary vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

```
1: function A(X)
2:     for all rows x_i of X do
3:         Let b = x_i
4:         Find a maximizing sim(X, ab^T)
5:     end for
6:     return best vectors a and b
7: end function
```

Definition 4 (Binary rank-1 decomposition). *Given an n-by-m binary matrix $\boldsymbol{X}$, find an n-dimensional binary vector $\boldsymbol{a}$ and an m-dimensional binary vector $\boldsymbol{b}$ that maximize* $\mathsf{sim}(\boldsymbol{X}, \boldsymbol{a} \boxtimes \boldsymbol{b})$.

Lemma 1. *Given an k-by-nm binary matrix $\boldsymbol{X}$, finding n-by-k and m-by-k binary matrices $\boldsymbol{A}$, $\boldsymbol{B}$ that maximize* $\mathsf{sim}(\boldsymbol{X}, (\boldsymbol{B} \odot \boldsymbol{A})^T)$ *is equivalent to finding the most similar binary rank-1 approximation of each row $\boldsymbol{x}$ of $\boldsymbol{X}$, where the rows are re-shaped as n-by-m binary matrices.*

Proof. If $\boldsymbol{x}_i$ is the ith row of $\boldsymbol{X}$ and $\boldsymbol{z}_i$ is the corresponding row of $(\boldsymbol{B} \odot \boldsymbol{A})^T$, then $\mathsf{sim}(\boldsymbol{X}, (\boldsymbol{B} \odot \boldsymbol{A})^T) = \sum_{i=1}^{k} \mathsf{sim}(\boldsymbol{x}_i, \boldsymbol{z}_i)$, and hence we can solve the problem row-by-row. Let $\boldsymbol{x} = (x_{1,1}, x_{2,1}, \ldots, x_{n,1}, x_{1,2}, \ldots, x_{n,m})$ be a row of $\boldsymbol{X}$. Re-write $\boldsymbol{x}$ as an n-by-m matrix in column major order.

Consider the row of $(\boldsymbol{B} \odot \boldsymbol{A})^T$ that corresponds to $\boldsymbol{x}$, and notice that it can be written as $(\boldsymbol{b} \otimes \boldsymbol{a})^T$, where $\boldsymbol{a}$ and $\boldsymbol{b}$ be the columns of $\boldsymbol{A}$ and $\boldsymbol{B}$ that correspond to $\boldsymbol{x}$. As $(\boldsymbol{b} \otimes \boldsymbol{a})^T = (b_1 \boldsymbol{a}^T, b_2 \boldsymbol{a}^T, \cdots, b_m \boldsymbol{a}^T)$, re-writing it similarly as $\boldsymbol{x}$ we get $(\boldsymbol{b} \otimes \boldsymbol{a})^T = (a_1 b_1, a_2 b_1, \ldots, a_n b_1, a_1 b_2, \ldots, a_n b_m) = \boldsymbol{a}\boldsymbol{b}^T = \boldsymbol{a} \boxtimes \boldsymbol{b}$. Thus, we obtain $\mathsf{sim}(\boldsymbol{x}, (\boldsymbol{b} \otimes \boldsymbol{a})^T) = \mathsf{sim}(\boldsymbol{Y}, \boldsymbol{a} \boxtimes \boldsymbol{b})$. □

We present a simple, deterministic algorithm that approximates the maximum similarity within 0.828, Algorithm 2. It is similar to the algorithm for hypercube segmentation based on random sampling presented by Kleinberg et al. [10]. The algorithm considers every row of $\boldsymbol{X}$ as a potential vector $\boldsymbol{b}$ and finds the best $\boldsymbol{a}$ given $\boldsymbol{b}$. Using Lemma 3.1 of [10] it is straight forward to show that the algorithm achieves the claimed approximation ratio:

Lemma 2. *Algorithm 2 approximates the optimum similarity within* 0.828 *in time $O(nm \min\{n, m\})$.*

Proof. To prove the approximation ratio, let $\boldsymbol{a}^*(\boldsymbol{b}^*)^T$ be the optimum decomposition. Consider the rows in which $\boldsymbol{a}^*$ has 1. Per Lemma 3.1 of [10], selecting one of these rows, call it $\boldsymbol{b}$, gives us $\mathsf{sim}(\boldsymbol{X}, \boldsymbol{a}^*\boldsymbol{b}^T) \geq (2\sqrt{2} - 2)\mathsf{sim}(\boldsymbol{X}, \boldsymbol{a}^*\boldsymbol{b}^T)$ (notice that $\boldsymbol{a}^*\boldsymbol{b}^T$ agrees with the optimal solution in rows where $\boldsymbol{a}^*$ is zero). Selecting $\boldsymbol{a}$ that maximizes the similarity given $\boldsymbol{b}$ can only improve the result, and the claim follows as we try every row of $\boldsymbol{X}$.

If $n < m$, the time complexity follows as for every candidate $\boldsymbol{b}$ we have to make one sweep over the matrix. If $m < n$, we can operate on the transpose. □

4.3 Discussion

Lemma 1 gives us yet another way of interpreting BCPC, namely, in BCPC each centroid must be a binary rank-1 matrix. One could define a more general variant where the centroids are arbitrary-rank binary matrices. Between these two extrema is a problem where the (Boolean) ranks of the centroids are bounded from above by some constant $r < \min\{n, m\}$. For such a problem, however, finding the centroids is even harder than it is now, as it essentially requires us to solve the Boolean matrix factorization problem which is a hard problem even to approximate [14].

5 Experimental Evaluation

5.1 Other Methods and Evaluation Criteria

We decided to compare `SaBoTeur` to other Boolean tensor CP methods, and for some real-world experiments we also used a continuous CP method.

The Boolean CP methods we used for comparison were `BCP_ALS` [13] and `Walk'n'Merge` [6]. `BCP_ALS` is based on iteratively updating the factor matrices one at a time (similarly to the classical alternating least squares optimizations), while `Walk'n'Merge` is a recent algorithm for highly scalable Boolean tensor factorization in sparse binary tensors. We did not use `Walk'n'Merge` on synthetic data as `BCP_ALS` is expected to perform better on smaller and denser tensors [6] but we used it on larger real-world tensors; `BCP_ALS`, on the other hand, does not scale well to larger tensors and hence we had to omit it from some experiments.

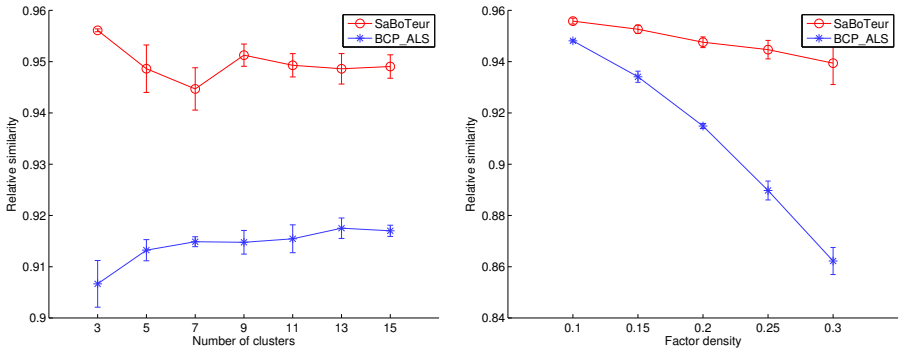
Of the continuous methods we used `CP_APR` [4] (implementation from the Matlab Tensor Toolbox v2.5[1]), an alternating Poisson regression algorithm that is specifically developed for sparse (counting) data.

For synthetic data, we report the relative similarity, that is, the fraction of the elements where the data and the clustering agree. For real-world data, we report the error measured using the squared Frobenius norm. This norm however can help the real-valued methods, as it scales all errors less than 1 down, but at the same time, small errors cumulate unlike with fully binary data. To alleviate this problem, we also rounded the reconstructed tensors from `CP_APR` to binary tensors. From different rounding thresholds between 0 and 1 we selected the one that gave the lowest (Boolean) reconstruction error.

5.2 Synthetic Experiments

To test the `SaBoTeur` algorithm in a controlled environment we created synthetic data sets that measured the algorithm's response to (1) different numbers of clusters, (2) different density of data, and (3) different levels of noise. All tensors were 700-by-500-by-50. All data sets were created by first creating ground-truth binary factor matrices $\boldsymbol{A}$, $\boldsymbol{B}$, and $\boldsymbol{C}$. The default number of clusters was 7 and

[1] `http://www.sandia.gov/~tgkolda/TensorToolbox/`



(a) Varying number of clusters. (b) Varying density of the factors.

Fig. 1. Synthetic experiment results. Markers are at the mean over five random tensors and the width of the errorbars is twice the standard deviation.

the default density of $\boldsymbol{A}$ and $\boldsymbol{B}$ was 0.2. A symmetric random noise was applied to the tensor and flipped 4% of the elements by default.

We varied each of the three features one at a time keeping the others in their default values, and created 5 random copies on each parameter combination. The results we report are mean values over these five random copies. In all experiments, the number of clusters (or factors) was set to the true number of clusters used to create the data. The number of re-samples in `SaBoTeur` was set to $r = 20$ in all experiments. We only used `BCP_ALS` to compare against in the synthetic experiments.

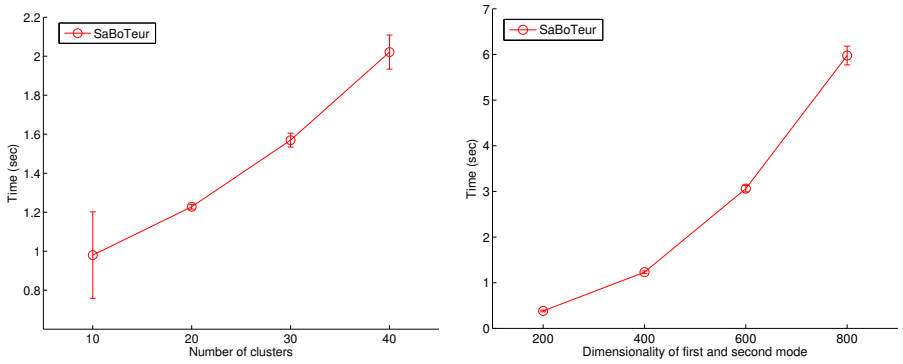
Varying the number of clusters. The number of clusters varied from 3 to 15 with steps of 2. The results are shown in Figure 1a. Perhaps the most surprising result here is how much better `SaBoTeur` is compared to `BCP_ALS`, especially given that `SaBoTeur`'s answer is a valid Boolean CP decomposition.

Varying the density. The density of the factor matrices varied from 10% to 30% with steps of 5%. The results can be seen in Figure 1b. Again `SaBoTeur` is better than `BCP_ALS`: it has gradually declining slope for increased density, whereas `BCP_ALS`'s results dive much faster.

Varying the noise. In the final synthetic experiment, we varied the noise level between 2% and 10% with steps of 2%. As is to be expected, increasing the noise decreases the results of both algorithms. Both algorithms exhibit roughly linear decrease in the similarity w.r.t. noise level, but `SaBoTeur` is again consistently the better of the two (results not shown).

Scalability. `SaBoTeur` is implemented in Matlab[2] and we run the scalability tests on a dedicated machine with 8 Intel Xeon E5530 2.4GHz processors and 48GB

[2] The code is available from `http://www.mpi-inf.mpg.de/~pmiettin/btc/`



(a) Varying the number of clusters. (b) Varying the dimensionality of the data.

Fig. 2. Scalability experiment results. Markers are at the mean over five random tensors and the width of the errorbars is twice the standard deviation.

of main memory. All reported times are wall-clock times. For these experiments, we created a new set of tensors. First, we tested the effect the number of clusters has to the algorithm. The data was 400-by-400-by-80 and the number of clusters varied from 10 to 40 with steps of 10. As can be seen in Figure 2a, we observe that the algorithm scales almost-linearly with the number of clusters. The slight non-linearity is due to the increasing number of non-zeros in the data in higher values of k. For the second experiment, we varied the dimensionality of the first and second mode between 200 and 800 with steps of 200. The results can be seen in Figure 2b, where we observe close-to-quadratic behavior, in line with the theoretical running time of the algorithm.

Discussion. The synthetic experiments confirm that `SaBoTeur` is capable of recovering the latent cluster structure from the synthetic data sets. Arguably the most surprising result of the synthetic experiments was that `SaBoTeur` was consistently better than `BCP_ALS`, even though the latter has more freedom to obtain better solutions.

5.3 Real-World Data

Datasets and earlier experiments. We tested `SaBoTeur` with three real-world data sets: The Resolver data contains entity–relation–entity tuples from the TextRunner open information extraction algorithm[3] [15]. A sample of size 343-by-360-by-200 (entity-by-entity-by-relation) was used for the experiments. The Enron data[4] (146-by-146-by-38) contains information about who sent e-mail to whom (rows and columns) per months (tubes). The TracePort data set[5] (10 266-by-8 622-by-501)

[3] `http://www.cis.temple.edu/~yates/papers/jair-resolver.html`
[4] `http://www.cs.cmu.edu/~enron/`
[5] `http://www.caida.org/data/passive/passive_2009_dataset.xml`

Table 1. Reconstruction errors rounded to the nearest integer. '—' denotes that the experiment was not conducted. Part of the results are from [6] and [13].

Algorithm	Enron	TracePort	Resolver
`SaBoTeur`	1 765	10 946	1 488
`BCP_ALS`	1 850	—	1 492
`Walk'n'Merge`	1 753	10 968	—
`CP_APR`	1 619	11 069	1 497
`CP_APR` $_{0/1}$	1 833	11 121	1 543

contains anonymized passive traffic traces (source and destination IP and port numbers) from 2009. With Enron and TracePort data sets, for `Walk'n'Merge`, `CP_APR` and `ParCube` we used the results from [6]. The results for `BCP_ALS` and Resolver are from [13]. The number of clusters/factors was set to $k = 15$ except for Enron data, for which it was $k = 12$.

Results. The results with the real-world data sets can be seen in Table 1. `SaBoTeur` continues its impressive results, being roughly on par with the other binary methods and with TracePort even better than the continuous method, `CP_APR`. In conjunction with what we observed in the synthetic setting, `SaBoTeur` consistently outperforms `BCP_ALS` despite solving a more restricted problem.

6 Conclusions and Future Work

We have studied the problem of clustering one mode of a 3-way binary tensor while simultaneously reducing the dimensionality in the two other modes. This problem bears close resemblance to the Boolean CP tensor decomposition, but the additional clustering constraint makes the problem significantly different. The main source of computational complexity, the consideration of overlapping factors in the tensor decomposition, does not play a role in BCPC. This lets us design algorithms with provable approximation guarantees better than what is known for the Boolean matrix and tensor decompositions.

Our experiments show that the algorithm for BCPC, `SaBoTeur`, is better than the dedicated (Boolean) tensor decomposition algorithms in building a Boolean CP decomposition of a tensor. Sometimes `SaBoTeur` also outperforms continuous methods for non-Boolean CP decomposition.

The essential piece, the maximum-similarity binary rank-1 approximation achieves an approximation ratio of 0.828 in $O(nm \min\{n, m\})$ time, and with that dominates the running time. Faster algorithms for the rank-1 approximation (with better approximation guarantees) would have an instant impact on `SaBoTeur`.

For the rank-1 approximation, the running time of $O(nm \min\{n, m\})$ is ascribed to the fact that every row of $\boldsymbol{X}$ (resp. every column if $n < m$) is tried as candidate. Because $\boldsymbol{a}^* \boldsymbol{b}^T$ already agrees with the optimal solution in rows where $\boldsymbol{a}^*$ is zero, it would be enough to take the non-zero elements into account rather than the complete rows. This modification is in particular beneficial to

sparse settings. Further improvement in terms of speed could be gained from parallelization of the algorithm, possibly using a MapReduce model. Each of the two concatenated loops in Algorithm 2 could be executed in parallel. Also early stopping might be advantageous if no rows are left that could improve the result. This might however require an additional step of sorting the rows according to their number of non-zero elements. Future investigations need to show which parallel configuration is most beneficial. Another aspect of refinement concerns the introduction of data structures and operations tailored to Boolean algebra. Storing the data as bit vectors and the use of native bit operations might yield additional speedup. At the same time this reduces the space needed to store the data compared to the representation as vectors of numbers.

Overall, this paper covered an extreme of the Boolean tensor clustering: each centroid was restricted to a rank-1 binary matrix. In future research, we hope to better cover the spectrum between this problem and the other extreme, clustering the frontal slices of the tensor with no reduction on the other modes.

References

1. N. Alon and B. Sudakov. On two segmentation problems. *J. Algorithm*, 33:173–184, 1999.
2. R. Belohlavek, C. Glodeanu, and V. Vychodil. Optimal Factorization of Three-Way Binary Data Using Triadic Concepts. *Order*, 30(2):437–454, Mar. 2012.
3. L. Cerf, J. Besson, K.-N. T. Nguyen, and J.-F. Boulicaut. Closed and noise-tolerant patterns in n-ary relations. *Data Min. Knowl. Discov.*, 26(3):574–619, 2013.
4. E. C. Chi and T. G. Kolda. On Tensors, Sparsity, and Nonnegative Factorizations. *SIAM J. Matrix Anal. Appl.*, 33(4):1272–1299, Dec. 2012.
5. D. Erdős and P. Miettinen. Discovering Facts with Boolean Tensor Tucker Decomposition. In *CIKM '13*, pages 1569–1572, 2013.
6. D. Erdős and P. Miettinen. Walk'n'Merge: A Scalable Algorithm for Boolean Tensor Factorization. In *ICDM '13*, pages 1037–1042, Dec. 2013.
7. H. Huang, C. Ding, D. Luo, and T. Li. Simultaneous tensor subspace selection and clustering: The equivalence of high order svd and k-means clustering. In *KDD '08*, pages 327–335, Aug. 2008.
8. D. I. Ignatov, S. O. Kuznetsov, J. Poelmans, and L. E. Zhukov. Can triconcepts become triclusters? *Int. J. Gen. Syst.*, 42(6):572–593, Aug. 2013.
9. S. Jegelka, S. Sra, and A. Banerjee. Approximation Algorithms for Tensor Clustering. In *ALT '09*, pages 368–383. Springer Berlin Heidelberg, 2009.
10. J. M. Kleinberg, C. H. Papadimitriou, and P. Raghavan. Segmentation problems. *J. ACM*, 51(2):263–280, Mar. 2004.
11. T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, 2009.
12. X. Liu, L. De Lathauwer, F. Janssens, and B. De Moor. Hybrid Clustering of Multiple Information Sources via HOSVD. In *ISNN '10*, pages 337–345. Springer Berlin Heidelberg, 2010.
13. P. Miettinen. Boolean Tensor Factorizations. In *ICDM '11*, pages 447–456, 2011.
14. P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila. The Discrete Basis Problem. *IEEE Trans. Knowl. Data En.*, 20(10):1348–1362, Oct. 2008.
15. A. Yates and O. Etzioni. Unsupervised methods for determining object and relation synonyms on the web. *J. Artif. Intell. Res.*, 34:255–296, 2009.