Tensor Co-clustering: a Parameter-less Approach (DISCUSSION PAPER)

Elena Battaglia and Ruggero G. Pensa

University of Turin, Dept. of Computer Science, Turin, Italy
{elena.battaglia,ruggero.pensa}@unito.it

Abstract. Tensors co-clustering has been proven useful in many applications, due to its ability of coping with high-dimensional data and sparsity. However, setting up a co-clustering algorithm properly requires the specification of the desired number of clusters for each mode as input parameters. To face this issue, we propose a tensor co-clustering algorithm that does not require the number of desired co-clusters as input, as it optimizes an objective function based on a measure of association across discrete random variables that is not affected by their cardinality. The effectiveness of our algorithm is shown on real-world datasets, also in comparison with state-of-the-art co-clustering methods.

1 Introduction

Tensors are widely used mathematical objects that well represent complex information such as gene expression data, social networks, heterogenous information networks, time-evolving data, behavioral patterns, and multi-lingual text corpora. In general, every n-ary relation can be easily represented as a tensor. From the algebraic point of view, in fact, they can be seen as multidimensional generalizations of matrices and, as such, can be processed with mathematical and computational methods that generalize those usually employed to analyze data matrices, e.g., non-negative factorization, singular value decomposition, itemset and association rule mining, clustering and co-clustering.

Clustering, in particular, is by far one of the most popular unsupervised machine learning techniques since it allows analysts to obtain an overview of the intrinsic similarity structures of the data with relatively little background knowledge about them. However, with the availability of high-dimensional heterogenous data, co-clustering has gained popularity, since it provides a simultaneous partitioning of each mode. Despite its proven usefulness the correct application of tensor co-clustering is limited by the fact that it requires the specification of a congruent number of clusters for each mode, while, in realistic analysis scenarios, the actual number of clusters is unknown. Furthermore, matrix/tensor

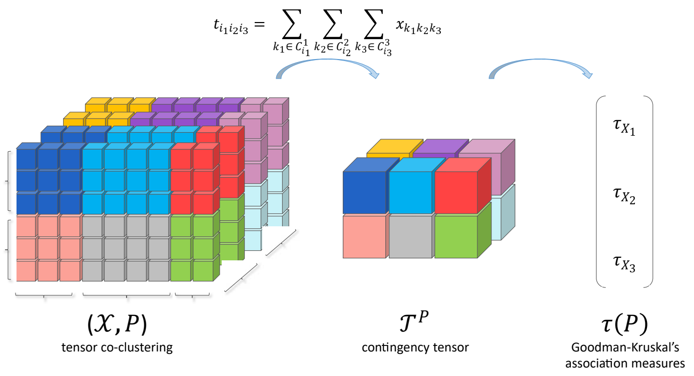

$t_{i_1 i_2 i_3} = \sum_{k_1 \in C^1_{i_1}} \sum_{k_2 \in C^2_{i_2}} \sum_{k_3 \in C^3_{i_3}} x_{k_1 k_2 k_3}$
τ_{X_1}
τ_{X_2}
τ_{X_3}
$(\mathcal{X}, P)$
tensor co-clustering
$\mathcal{T}^P$
contingency tensor
$\tau(P)$
Goodman-Kruskal's association measures

Fig. 1: An example of tensor co-clustering with the related contingency tensor and the associated Goodman-Kruskal's τ measures.

(co-)clustering is often based on a preliminary tensor factorization step or on latent block models [14, 3, 13] that, in their turn, require further input parameters (e.g., the number of latent factors/blocks within each mode). As a consequence, it is merely impossible to explore all combinations of parameter values in order to identify the best clustering results.

The main reason for this problem is that most clustering algorithms (and tensor factorization approaches) optimize objective functions that strongly depend on the number of clusters (or factors). Hence, two solutions with two different numbers of clusters can not be compared directly. Although this reduces considerably the size of the search space, it prevents the discovery of a better partitioning once a wrong number of clusters is selected. In this paper, we address this limitation by proposing a new tensor co-clustering algorithm that optimizes an objective function that can be viewed as n-mode extension of an association measure called Goodman-Kruskal's τ [5], whose local optima do not depend on the number of clusters. Compared with state-of-the-art techniques that require the desired number of clusters in each mode as input parameters, our approach achieves similar or better results on several real-world datasets.

The algorithm presented in this paper has been fist introduced in [1]. An interested reader could refer to it for further theoretical and experimental details.

2 An association measure for tensor co-clustering

The objective function optimized by our tensor co-clustering algorithm is an association measure, called Goodman and Kruskal's τ [5], that evaluates the dependence between two discrete variables and has been used to evaluate the quality of 2-way co-clustering [10]. Goodman and Kruskal's τ estimates the strength of the link between two discrete variables X and Y according to the proportional reduction of the error in predicting one of them knowing the other: $\tau_{X|Y} = \frac{e_X - \mathbb{E}[e_{X|Y}]}{e_X}$, where e_X is the error in predicting X (estimated as the probability that two different observations from the marginal distribution of X fall in different categories) and $\mathbb{E}[e_{X|Y}]$ is the expected value of the conditional error taken with respect to the distribution of Y. Conversely, the proportional reduction of the error in predicting Y while X is known is $\tau_{Y|X} = \frac{e_Y - \mathbb{E}[e_{Y|X}]}{e_Y}$.

In order to use this measure for the evaluation of a tensor co-clustering, we need to extend it so that τ can evaluate the association of n distinct discrete variables $X_1, \ldots, X_n$. Reasoning as in the two-dimensional case, we can define the reduction in the error in predicting X_i while $(X_j)_{j \neq i}$ are all known as

$$\tau_{X_i} = \tau_{X_i|(X_j)_{j\neq i}} = \frac{e_{X_i} - \mathbb{E}[e_{X_i|(X_j)_{j\neq i}}]}{e_{X_i}}$$

for all $i \leq n$. When $n = 2$, the measure coincides with Goodman-Kruskal's τ.

We will now see how to use function τ to evaluate a tensor co-clustering. Let $\mathcal{X} \in \mathbb{R}_+^{m_1 \times \cdots \times m_n}$ be a tensor with n modes and non-negative values. Let us denote with $x_{k_1 \ldots k_n}$ the generic element of $\mathcal{X}$, where $k_i = 1, \ldots, m_i$ for each mode $i = 1, \ldots, n$. A co-clustering $\mathcal{P}$ of $\mathcal{X}$ is a collection of n partitions $\{\mathcal{P}_i\}_{i=1,\ldots,n}$, where $\mathcal{P}_i = \cup_{j=1}^{c_i} C_j^i$ is a partition of the elements on the i-th mode of $\mathcal{X}$ in c_i groups, with $c_i \leq m_i$ for each $i = 1, \ldots, n$. Each co-clustering $\mathcal{P}$ can be associated to a tensor $\mathcal{T}^{\mathcal{P}} \in \mathbb{R}_+^{c_1 \times \cdots \times c_n}$, whose generic element is the sum of all entries of $\mathcal{X}$ in the same co-cluster. We can look at $\mathcal{T}^{\mathcal{P}}$ as the contingency n-modal table that empirically estimates the joint distribution of n discrete variables $X_1, \ldots, X_n$, where each X_i takes values in $\{C_1^i, \ldots C_{c_i}^i\}$. From this contingency table, we can derive the marginal probabilities of each variable X_i (i.e., the probability that a generic element of mode i falls in a particular cluster on that mode) and we can use these probabilities to compute Goodman and Kruskal's τ functions: in this way we associate to each co-clustering $\mathcal{P}$ over $\mathcal{X}$ a vector $\tau^{\mathcal{P}} = (\tau_{X_1}^{\mathcal{P}}, \ldots, \tau_{X_n}^{\mathcal{P}})$ that can be used to evaluate the quality of the co-clustering. The overall co-clustering schema is depicted in Figure 1.

3 A stochastic local search approach to co-clustering

Our co-clustering approach can be formulated as a multi-objective optimization problem: given a tensor $\mathcal{X}$ with n modes and dimension m_i on mode i, an optimal co-clustering $\mathcal{P}$ for $\mathcal{X}$ is one that is not dominated by any other co-clustering (i.e. does not exist any other co-clustering $\mathcal{Q}$ with $\tau_{X_j}^{\mathcal{Q}} >= \tau_{X_j}^{\mathcal{P}}$ for all $j = 1, \ldots, n$).

Since we do not fix the number of clusters, the space of possible solutions is huge (for example, given a very small tensor of dimension $10 \times 10 \times 10$, the number of possible partitions is 1.56×10^{16}): it is clear that a systematic exploration of all possible solutions is not feasible for a generic tensor $\mathcal{X}$. For this reason we need to find a heuristic that allows us to reach a "good" partition of $\mathcal{X}$, i.e. a partition $\mathcal{P}$ with high values of $\tau_{X_k}^{\mathcal{P}}$ for all modes k. With this aim, we propose a stochastic local search approach to solve the maximization problem.

Algorithm 1: $\tau TCC(\mathcal{X}, N_{iter})$

```
Input: A $m_1 \times \cdots \times m_n$ tensor $\mathcal{X}$, $N_{iter}$
Result: $\mathcal{P}_1, \ldots, \mathcal{P}_n$
1  Initialize $\mathcal{P}_1, \ldots, \mathcal{P}_n$ with discrete partitions;
2  $i \leftarrow 0$;
3  $iter\_whithout\_moves \leftarrow 0$ ;
4  while $i \leq N_{iter}$ & $iter\_whithout\_moves < max_{j=1,\ldots,n}(m_j)$ do
5      for $k = 1$ to $n$ do
6          if $iter\_whithout\_moves < t$ then
7              Randomly choose $C_b^k$ in $\mathcal{P}_k$ and $x$ in $C_b^k$;
8          else
9              $x \leftarrow next(k)$ //Select the element following the one selected at
                iteration $i-1$ on mode $k$;
10             $C_b^k \leftarrow$ Cluster of $x$;
11         end
12         for $C_k^j$ in $\mathcal{P}_k \cup \emptyset$ do
13             $\mathcal{Q}_k^j \leftarrow (\mathcal{P}_k \setminus \{C_b^k, C_j^k\}) \bigcup \{C_b^k \setminus \{x\}, C_j^k \cup \{x\}\}$;
14             $\mathcal{Q}^j \leftarrow (\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_{k-1}, \mathcal{Q}_k, \mathcal{P}_{k+1}, \ldots, \mathcal{P}_n)$;
15             Compute contingency tensor $T^j$ associated to $\mathcal{Q}^j$ and $\tau^{\mathcal{Q}_j}$;
16         end
17         $e \leftarrow SelectBestPartition(k, b, (\tau^{\mathcal{Q}_j})_{j=1,\ldots,|\mathcal{P}_k \cup \emptyset|})$;
18         $\mathcal{P}_k \leftarrow Q_k^e$;
19         if $e == b$ then
20             $iter\_whithout\_moves \leftarrow iter\_whithout\_moves + 1$;
21         else
22             $iter\_whithout\_moves \leftarrow 0$;
23         end
24         $i \leftarrow i + 1$;
25     end
26 end
```

Algorithm 1 provides the general sketch of our tensor co-clustering algorithm, called τTCC. It repeatedly considers modes one by one, sequentially, and it tries to improve the quality of the co-clustering by moving one single element from its original cluster C_b^k to another cluster on the same mode, C_e^k, which most improves the quality of the partition, according to a criterion chosen to measure the quality of the partition. When all the n modes have been considered, the i-th iteration of the algorithm is concluded. If all objects have been tried but no move is possible, the algorithm ends. At the end of each iteration, one of the following possible moves has been done on mode k:

- an object x has been moved from cluster C_b^k to a pre-existing cluster C_e^k: in this case the final number of clusters on mode k remains the same (let'call it c_k) if C_b^k is non-empty after the move. If C_b^k is empty after the move, it will be deleted and the final number of clusters will be $c_k - 1$;

– an object x has been moved from cluster C_b^k to a new cluster $C_e^k = \emptyset$: the final number of clusters on mode k will be $c_k + 1$ (the useless case when x is moved from $C_b^k = \{x\}$ to $C_e^k = \emptyset$ is not considered);
– no move has been performed and the number of clusters remains c_k.

Thus, during the iterative process, the updating procedure is able to increase or decrease the number of clusters at any time. This is due to the fact that, contrary to other measures, such as the loss in mutual information [4], τ measure has an upper limit which does not depend on the number of co-clusters and thus enables the comparison of co-clustering solutions of different cardinalities.

As mentioned above, the choise of the best cluster on mode k in which the selected element x should be moved depends on the way in which we decide to measure the increase in the quality of the tensor partition. We can define different measures, corresponding to different ways to implement function $SelectBestPartition$ in Algorithm 1. According to our experiments, the one with best performances in terms of speed of convergence and quality of the identified co-clusters is the following. Suppose we want to move an object on mode k: we consider only those moves that improve (or at least do not worsen) τ_{X_k} and, among them, we choose the one with the greatest value of $avg(\tau) = \frac{1}{n}\sum_{j=1}^{n} \tau_{X_j}$. It can be proven that algorithm τTCC with this selection strategy converges to a Pareto local optimum.

4 Experiments and discussion

In this section, we test the effectiveness of our co-clustering algorithm through experiments on the following three real-world datasets: the "four-area" DBLP dataset[1]; the "hetrec2011-movielens-2k" dataset[2] [2], from which we extraxt two different tensors, MovieLens1 and MovieLens2; the Yelp dataset[3], from which we extract two tensors, YelpTOR and YelpPGH. To assess the quality of the clustering performances, we consider two measures commonly used in the clustering literature: normalized mutual information (NMI) [11] and adjusted rand index (ARI) [8].

We compare our results with those of other state-of-the-art tensor co-clustering algorithms. **nnCP** is the non-negative CP decomposition [6] and can be used to co-cluster a tensor, as done by [14], by assigning each element in each mode to the cluster corresponding to the latent factor with highest value. **nnTucker** is the non-negative Tucker decomposition [12]. **nnCP+kmeans** and **nnT+kmeans** combine CP (or Tucker) decomposition with a post-processing phase in which k-means is applied on each of the latent factor matrices, similarly as what has been done by [7] and [3]. **SparseCP** consists of a CP decomposition with non-negative sparse latent factors [9]. Finally, **TBM** performs tensor co-clustering

[1] `http://web.cs.ucla.edu/~yzsun/data/DBLP_four_area.zip`
[2] `https://grouplens.org/datasets/hetrec-2011/`
[3] `https://www.yelp.com/dataset`

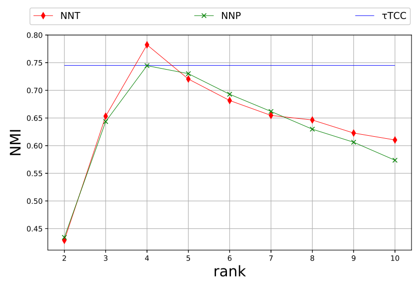
Table 1: Results achieved by the co-clustering algorithms on the real-world datasets (average values over 5 runs). NMI and ARI are computed for the main mode (authors in DBLP, movies in MovieLens and restaurants in Yelp).

Dataset	**DBLP**		**MovieLens1**		**MovieLens2**		**yelpTOR**		**yelpPGH**	
Algorithm	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI	NMI	ARI
τTCC	0.74	0.80	**0.66**	**0.71**	**0.40**	**0.51**	0.42	**0.44**	**0.38**	**0.28**
nnTucker	**0.78**	**0.84**	0.42	0.53	0.24	0.17	**0.43**	0.39	0.28	0.19
nnCP	0.74	0.80	0.38	0.34	0.11	0.03	0.42	0.37	0.10	0.06
SparseCP	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.15	0.11	0.04
TBM	-	-	0.00	0.01	0.01	0.01	0.02	0.01	0.10	0.04
nnCP+kmeans	0.24	0.08	0.31	0.21	0.15	0.07	0.11	0.01	0.09	0.03
nnT+kmeans	0.25	0.06	0.38	0.31	0.17	0.09	0.09	0.01	0.09	0.03

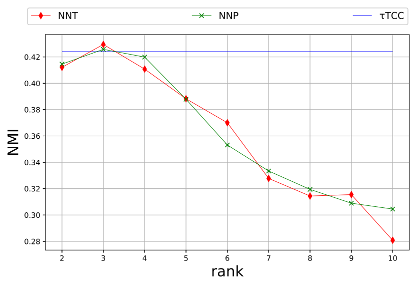
via the Tensor Block Model [13]. The last four methods require as input parameters the number of clusters on each mode: we set these numbers equal to the true number of classes on the three modes of the tensor. When further parameters are needed, we follow the instructions suggested in the original papers for setting them. Algorithms **nnCP**, **nnTucker** and their variant with k-means are applied trying different ranks of the decomposition: we report the best result obtained. In this way we are giving a big advantage to our competitors: we choose the rank of the decomposition and the number of clusters by looking at the actual number of categories, which are unknown in standard unsupervised settings. Despite this, as shown in Table 1, τTCC outperforms the other algorithms on all datasets but one (DBLP) and has comparable results on another (YelpTOR). In these cases, non-negative Tucker decomposition (with the number of latent factors set to the correct number of embedded clusters) achieves the best results and non-negative CP decomposition obtains results that are comparable with those of τTCC. However, when we modify, even if slightly, the number of latent factors (see Figure2), the results get immediately worse than those of τTCC.

The number of clusters identified by τTCC is usually close to the correct number of embedded clusters: on average, 5 instead of 4 for DBLP, 5 instead of 3 for MovieLens1, the correct number 3 for MovieLens2, 5 instead of 3 for YelpPGH. Only YelpTOR presents a number of clusters (13) that is far from the correct number of classes (3). However, more than the 85% of the objects are classified in 3 large clusters, while the remaining objects form very small clusters: we consider these objects as candidate outliers. The same beahviour is even more pronounced in DBLP, where four clusters contain the 99.9% of the objects and only 2 objects stay in the "extra cluster".

Lastly, we provide some insights about the quality of the clusters identified by our algorithm. To this purpose, we choose a co-clustering of the MovieLens1 dataset. This dataset contains 181 movies and all the tags assigned by the users to each movie. We construct a $(215 \times 181 \times 142)$-dimensional tensor, where the three modes represent users, movies and tags. The movies are labelled in three categories (Animation, Horror and Documentary). Algorithm τTCC identifies five clusters of movies, instead of the three categories we consider as labels. The tag clouds in Figure 3, illustrate the 30 movies with more tags for each cluster (text size depends on the actual number of tags): it can be easily observed that

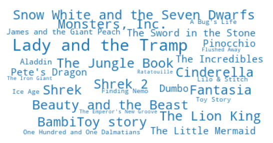


(a) DBLP

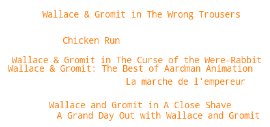


(b) YelpTOR

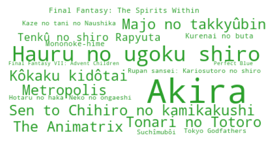
Fig. 2: Variation of nnTucker/nnCP results w.r.t. the rank of the decomposition in DBLP and YelpTOR datasets.



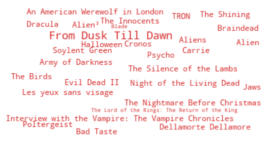
(a) Cl1 - Cartoons


Wallace & Gromit in The Wrong Trousers
Chicken Run
Wallace & Gromit in The Curse of the Were-Rabbit
Wallace & Gromit: The Best of Aardman Animation
La marche de l'empereur
Wallace and Gromit in A Close Shave
A Grand Day Out with Wallace and Gromit

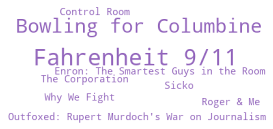
(b) Cl2 - Wallace&Gromit



(c) Cl3 - Anime



(d) Cl4 - Horror



(e) Cl5 - Documentaries

Fig. 3: First 30 movie in each cluster identified by τTCC on dataset MoveiLens1.

the first cluster concerns animated movies for children, mainly Disney and Pixar movies; the second one is a little cluster containing animated movies realized with the claymation technique (mainly Wallace and Gromit saga's movies or other films by the same director); the third cluster is still a subset of the animated movies, but it contains anime and animated films from Japan. The fourth cluster is composed mainly by horror movies and the last one contains only documentaries. On the tag mode, our algorithm finds thirteen clusters. Six of them contain more than 90% of the total tags and only 10 uninformative tags are partitioned in other 7 very small clusters, and could be considered as outliers. There is a one-to-one correspondence between four clusters of movies (Cartoons, Anime, Wallace&Gromit and Documentary) and four of the tag clusters; cluster Horror, instead, can be put in relation with two different tag clusters, the first containing names of directors, actors or characters of popular horror movies, the second composed by adjectives tipically used to describe disturbing films.

5 Conclusions

Our experimental validation has shown that our approach is able to identify meaningful clusters. Moreover, it outperforms state-of-the-art methods for most datasets. Even when our algorithm is not the best one, we have found that the competitors can not work properly without specifying a correct number of clusters for each mode of the tensor. As future work, we will design a specific algorithm for sparse tensors with the aim of reducing the overall computational complexity of the approach. Finally, we will further investigate the ability of our method to identify candidate outliers as small clusters in the data.

References

1. Battaglia, E., Pensa, R.G.: Parameter-less tensor co-clustering. In: Proceedings of DS 2019. pp. 205–219 (2019)
2. Cantador, I., Brusilovsky, P., Kuflik, T.: 2nd Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011). In: Proc. RecSys 2011 (2011)
3. Cao, X., Wei, X., Han, Y., Lin, D.: Robust face clustering via tensor decomposition. IEEE Trans. Cybernetics **45**(11), 2546–2557 (2015)
4. Dhillon, I.S., Mallela, S., Modha, D.S.: Information-theoretic co-clustering. In: Proc. ACM SIGKDD 2003. pp. 89–98 (2003)
5. Goodman, L.A., Kruskal, W.H.: Measures of association for cross classification. Journal of the American Statistical Association **49**, 732–764 (1954)
6. Harshman, R.A.: Foundation of the parafac procedure: models and conditions for an" explanatory" multimodal factor analysis. UCLA Working Papers in Phonetics **16**, 1–84 (1970)
7. Huang, H., Ding, C.H.Q., Luo, D., Li, T.: Simultaneous tensor subspace selection and clustering: the equivalence of high order svd and k-means clustering. In: Proc. ACM SIGKDD 2008. pp. 327–335 (2008)
8. Hubert, L., Arabie, P.: Comparing partitions. J. Classif. **2**(1), 193–218 (1985)
9. Papalexakis, E.E., Sidiropoulos, N.D., Bro, R.: From K-means to higher-way co-clustering: Multilinear decomposition with sparse latent factors. IEEE Trans. Signal Processing **61**(2), 493–506 (2013)
10. Robardet, C., Feschet, F.: Efficient local search in conceptual clustering. In: Proceedings of DS 2001. pp. 323–335 (2001)
11. Strehl, A., Ghosh, J.: Cluster ensembles – A knowledge reuse framework for combining multiple partitions. Journal of Machine Learning Research **3**, 583–617 (2002)
12. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika **31**, 279–311 (1966)
13. Wang, M., Zeng, Y.: Multiway clustering via tensor block models. In: Proc. of NeurIPS 2019. pp. 713–723 (2019)
14. Zhou, Q., Xu, G., Zong, Y.: Web co-clustering of usage network using tensor decomposition. In: Proceedings of ECBS 2009. pp. 311–314 (2009)