TensorClus: A python library for tensor (Co)-clustering

Rafika Boutalbi, Lazhar Labiod, Mohamed Nadif

TensorClus: A Python Library for Tensor (Co)-clustering

Rafika Boutalbi[a,*], Lazhar Labiod[a], Mohamed Nadif[b]

[a]*Institute for Parallel and Distributed Systems, Analytic Computing, University of Stuttgart*
[b]*Université de Paris, CNRS, Centre Borelli UMR 9010, F-75006 Paris, France*

Abstract

Tensor data analysis is the evolutionary step of data analysis to more than two dimensions. Dealing with tensor data is often based on tensor decomposition methods. The present paper focuses on unsupervised learning and provides a python package referred to as `TensorClus` including novel co-clustering algorithms of three-way data. All proposed algorithms are based on the latent block models and suitable to different types of data, sparse or not. They are successfully evaluated on challenges in text mining, recommender systems, and hyperspectral image clustering. `TensorClus` is an open-source Python package that allows easy interaction with other python packages such as NumPy and TensorFlow; it also offers an interface with some tensor decomposition packages namely `Tensorly` and `TensorD` on the one hand, and on the other, the co-clustering package `Coclust`. Finally, it provides CPU and GPU compatibility. The `TensorClus` library is available at `https://pypi.org/project/TensorClus/`[1].

Keywords: Tensors, (Co)-clustering, Multiple Graphs, Tensor Decomposition.

---

1. Introduction

The amount of data collected in fields, as different as social networks, online shopping, or medicine has grown exponentially over the last decade. Nowadays,

*Corresponding author

Email addresses: `rafika.boutalbi@gmail.com` (Rafika Boutalbi), `lazhar.labiod@u-paris.fr` (Lazhar Labiod), `mohamed.nadif@u-paris.fr` (Mohamed Nadif)

[1]Accepted `https://www.sciencedirect.com/science/article/abs/pii/S0925231221013941`

the extraction of knowledge from such data can be based on data organized in the form of tensors instead of matrices. A tensor is a multidimensional array, which is also known as the N-way and Nth-order tensor; a third-order tensor has three dimensions. The use of tensors in data analysis applications was pioneered by researchers in psychometrics and chemometrics [1]. Two recent effective open-source `Tensorly` [2] and `TensorD` [3] are available. They offer a state-of-the-art tensor decomposition approach, including algorithms such as PARAFAC and Tucker decomposition.

Here, we are interested in three-way data that are present in many applications. In medical domain, for instance, we could have a tensor $patients \times images \times features$, and the objective could be analyzing patient images based on extracted features. To deal with such data we focus on co-clustering that can be viewed as an extension of clustering [4] devoted to reorganizing a data matrix into homogeneous blocks. This objective has attracted many authors these two decades through different approaches based on information-theoretic [5], spectral co-clustering [6, 7], matrix factorization [8, 9], or probabilistic models [10, 11, 12, 13, 14]. The recent `Coclust` package [15] provides the implementation of co-clustering algorithms designed to efficiently handle count data matrices [15]. However, despite the great interest in co-clustering techniques on the one hand and the tensor decomposition methods on the other, few works tackle co-clustering from tensor data. To date, we can cite [16, 17] based on tensor-based decomposition while aiming to extract co-clusters. In contrast with these methods that require parameters tuning, in our proposal, the co-clustering objective is derived from flexible tensor latent block models which present many advantages described in [18] and illustrated in section 2.2. Previously, we proposed [19, 18] Tensor Latent Block Model (TLBM) for the co-clustering of tensor data as illustrated in Figure 1. TLBM exploits the flexibility of the latent block model [4] and is able to consider any type of data i.e. continuous, binary, count tables. We also showed that the derived algorithms can be also used for the clustering of multiple graphs or multi-view clustering. The package `TensorClus` that we propose is the first free python package for tensor (co)-clustering and it is open-source.

2. `TensorClus` package

`TensorClus` is a Python library composed of five modules dedicated to each step of three-way data analysis, from data loading to the visualization of results. Figure 2 shows the structures of the library and the packages that interface

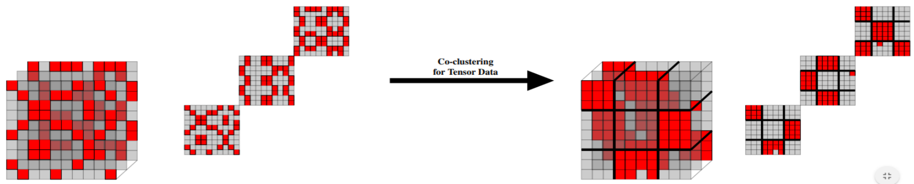

Figure 1: Objective of Tensor Co-clustering.

with `TensorClus`, namely `Tensorly`, `TensorD`, and `Coclust` available in Python. Next, we describe in details the five modules depicted in Figure 2.

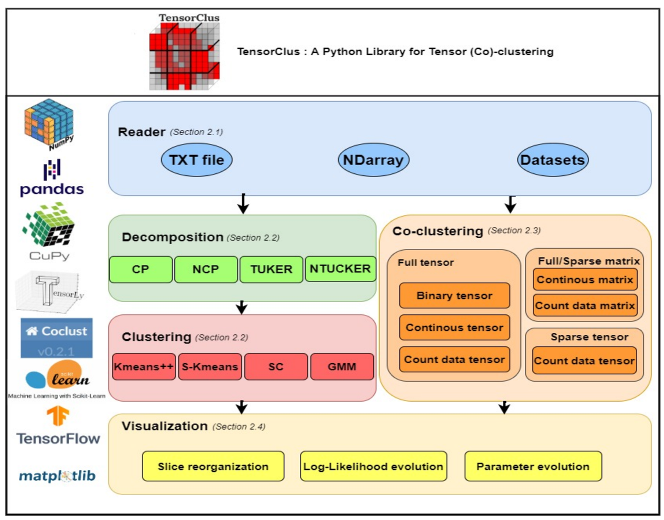

Figure 2: `TensorClus` library structure.

2.1. *Reader module*

To load tensor data, we built a `Reader` module that interacts with `NumPy` and `Pandas` packages. The module allows the following three ways of data loading:

- Load data from a text file: The user should save the tensor in a text file where the three first columns represent the tensor indices of entries and the last column the value of each tensor entry. For this, we can use the `read_txt_tensor` function.

- Load data from datasets: The user can import tensor datasets. We illustrate this step with datasets having different characteristics (see Table 1). The true partitions are also available for all datasets; they will be used just to evaluate the algorithms in this package in terms of clustering. For loading a dataset, the user can use the function `load_dataset` by specifying the dataset name.

- Create a `NumPy` array: The user can also create tensor data as an `NDarray` using the `NumPy` package and use it for tensor clustering.

Table 1: Characteristics of datasets.

Datasets	Type	#Slices	#Node	#Cluster
DBLP1	Text	4	1995	3
DBLP2	Text	4	2223	3
PubMed-Diabets-4K	Text	4	4354	3
Nus-Wide-8	Text+Images	6	2738	8
Amazon-Products-10	Text+Images	7	9897	10

The detailed description of the integrated datasets is available in a public github repository [2].

2.2. *Decomposition and clustering modules*

There are four popular implemented tensor decomposition methods, namely Parafac, Non-negative Parafac, Tucker decomposition, and Non-negative Tucker decomposition [20, 21]. Note that these methods are not devoted to clustering, however, they return factor matrices that can be used for clustering. The `decomposition_with_clustering` function is dedicated for this task. It has an argument `algorithm` for choosing which clustering algorithm among a list of suitable algorithms for the clustering of continuous data: `Kmeans++`, `Spherical Kmeans`, `Spectral clustering (SC)`, and the EM algorithm derived from *Gaussian Mixture Model* (`GMM`) available in the `Scikit-Learn` package.

Notice that, both learning representations and clustering tasks are performed successively –not simultaneously–. In contrast with these techniques, in our proposal with `TensorClus` the clustering procedure is carried out directly on three-way data and therefore does not require any learning representations.

2.3. *Co-clustering module*

Before describing the functions available in this module, we briefly present some essential points. From TLBM, different derived co-clustering algorithms are implemented. TLBM considers a three-way tensor data $\mathcal{X} = [\mathbf{x}_{ij}] \in \mathbb{R}^{n \times d \times v}$ where n, d, and v are the dimensions; $\mathbf{x}_{ij}$ is $(v \times 1)$ vector (Figure 3).

[2] https://github.com/boutalbi/TensorClus/blob/master/data_description.md

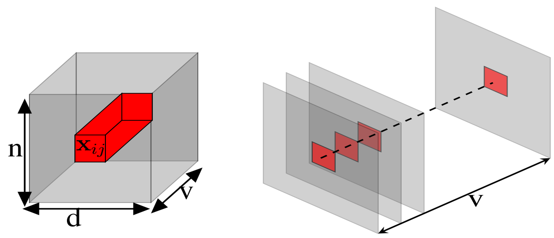

Figure 3: Three-way Data structure

To estimate the parameters of TLBM, we rely on variational EM, which optimizes the lower band of log-likelihood [22, 19, 18]. The implemented algorithms take as input the tensor $\mathcal{X}$ and the number of row clusters g and columns clusters m. It alternates two steps E and M (Algorithm 1), until the objective function value change is small or there is no change. The Expectation (E) step consists in computing the posterior probabilities $\mathbf{Z}^{(t)} = (z_{ik}) \in [0,1]^{n\times g}$ with $\sum_{k=1}^{g} z_{ik} = 1$ and $\mathbf{W}^{(t)} \in [0,1]^{d\times m}$ with $\sum_{\ell=1}^{m} w_{j\ell} = 1$, and Maximization (M) step consists in updating model parameters $\mathbf{\Omega}^{(t)}$. The parameter $\mathbf{\Omega}$ is formed by proportions of row clusters $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_g)$, proportions of column clusters $\boldsymbol{\rho} = (\rho_1, \ldots, \rho_m)$, and Θ which depends on the chosen probability distribution. Finally, at convergence, the algorithms return the row and column partitions and the estimated parameters $\mathbf{\Omega}$.

Algorithm 1: `TLBM`

Input: $\mathcal{X}$, g, m.
Initialization: Randomly generate $(\mathbf{Z}^{(0)}, \mathbf{W}^{(0)})$ and compute $\mathbf{\Omega}^{(0)}$
repeat
  E-Step: Compute the posterior probabilities $\mathbf{Z}^{(t)}$ and $\mathbf{W}^{(t)}$
  M-Step: Update parameters $\mathbf{\Omega}^{(t)}$
until *Convergence*;
return Z, W, $\mathbf{\Omega}$

With `TensorClus`, binary, continuous, and count data can be analyzed from Bernoulli, Gaussian, and Poisson models respectively. The `co-clustering` module provides the three following functions: `tensorCoclusteringBernoulli`, `tensorCoclustringGaussian`, and `tensorCoclusteringPoisson` that have the following arguments:

- `n_clusters` denotes the number of clusters.

- `init_row` and `init_col` are the initial partitions $\mathbf{Z}$ and $\mathbf{W}$, respectively. This means that the partitions are not randomly generated.

- `max_iter` denotes the number of iterations.

- `fuzzy` is a boolean value to choose if the final partition is hard or soft partition.

- `gpu` is a boolean value to select the type of execution, with or without GPU.

Note that `TensorClus` interfaces with `Coclust`. Therefore the user can also consider carrying out a co-clustering by slice. `Coclust` has been designed to complete and easily interface with popular machine learning libraires such as scikit-learn. Using the `sliceMatrixCoclustering` function of the `co-clustering` module, the user can perform different co-clustering algorithms with `Coclust`. This is achieved by specifying the index of slices and the selected algorithm.

Furthermore, a version dedicated for sparse three-way data referred to as `TSPLBM` is also proposed. The `TSLBM` algorithm tackles the clustering of multiple graphs. It is devoted to co-clustering of a three-way sparse data. Given $\mathcal{X} = [\mathbf{x}_{ij}] \in \mathbb{R}^{n \times n \times v}$ where n is the number of nodes, and v the number of graphs (slices). We can view the derived algorithm as an implicit consensus clustering for multiple graphs. With the `co-clustering` module, `sparseTensorCoclustering` allows to apply sparse tensor co-clustering.

2.4. Visualization module

`TensorClus` also offers a module for data visualization to illustrate and analyze the results of co-clustering. Figure 5 shows the three visualizations proposed by the `Visualization` module.

- `plot_logLikelihood_evolution` plots the log-likelihood in function of iterations.

- `plot_parameter_evolution` provides the evolution of Θ at each iteration. At the convergence, this allows to compare and interpret the obtained co-clusters.

- `plot_slice_reorganisation` reorganizes each slice of a three-way data according the obtained co-clusters.

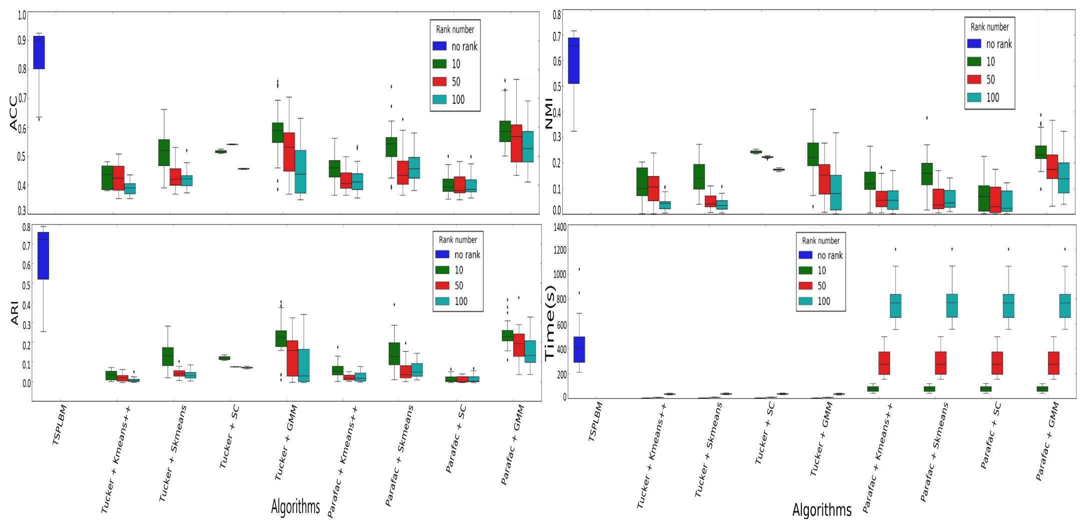

Figure 4: Comparison results using DBLP1 dataset.

3. Comparison of tensor co-clustering and tensor decomposition

This package allows to evaluate different algorithms dedicated to three-way data in terms of clustering. To reach this objective, we rely on datasets where a partition of one dimension is available, this is the case of the used three-way datasets. We propose to use external measurements such as accuracy (ACC), Normalized Mutual Information (NMI) [23], Adjusted Rand Index (ARI) [24]. These last two are less sensitive to heavily imbalanced clusters. These measures are equal to 1 if the resulting clustering is identical to the true one.

Applied on **DBLP1**, we compared the sparse tensor co-clustering algorithm `TSPLBM` with `Parafac` and `Tucker decomposition` combined with clustering algorithms. We use different ranks (10, 50, and 100) for tensor decomposition. We performed 30 runs with random initializations. Then we computed ACC, NMI, ARI, and computing time by averaging all runs. All experiments were performed using a PC with the following characteristics: Intel® Core 9e gen,a RAM(64 Gb), and GPU NVIDIA® GeForce® GTX 1650 Max-Q. Figure 4 shows the performances of `TSPLBM` and the two algorithms Parafac and Tucker decomposition (with different ranks) followed with the four clustering algorithms.

The experiments were performed using CPU version. It should be emphasized that `TSPLBM` gives better results, in terms of NMI than tensor-based decomposition algorithms combined with clustering. Tucker decomposition with a rank equal to

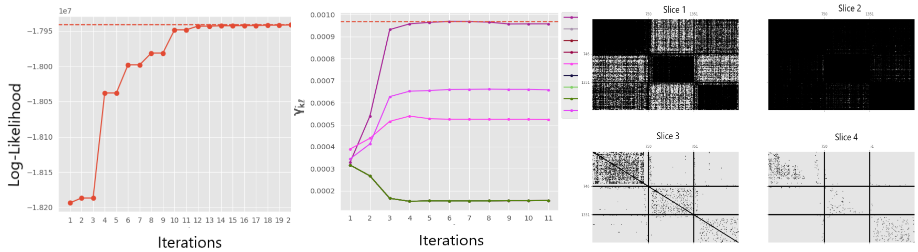

Figure 5: Tensor co-clustering results analysis for DBLP1 dataset.

10, combined with the GMM algorithm, achieves the best results after TSPLBM. In terms of time execution, TSPLBM is equivalent to Parafac combined with each clustering algorithm using a rank number equal to 50 and better than using the rank number equal to 100. Figure 5 shows the pictures obtained by the visualization module. We observe the log-likelihood increase at each iteration, and the algorithm converges at the 15th iteration (the plot on the left). In the middle figure, we observe the density evolution of co-clusters (densities of 3 diagonal co-clusters and one common density on outside of these co-clusters) given by `plot_parameter_evolution`. Finally, the figure on the right represents the slice reorganization based on the obtained co-clustering. We note that the three co-clusters with higher parameter values in the previous plot, are the three diagonal co-clusters; for details see [18].

TensorClus offers CPU and GPU compatibility. The CPU version uses the classical matrix operations from NumPy package. And for GPU, we rely to CuPy package wich is a NumPy-compatible array library accelerated by CUDA [25]. We compared the CPU and GPU versions of TSPLBM to evaluate computing time with both versions. In Figure 6 are reported the obtained results of CPU and GPU

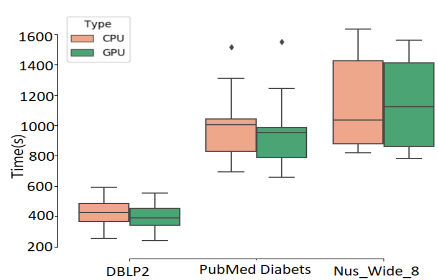

Figure 6: Comparison results of CPU and GPU version of TSPLBM on the three datasets.

versions in performing 10 runs for each version. We observe that `TensorClus` shows a slight performance using GPU implementation. These performances can be improved using a more powerful GPU. The experimentation's source code is available in a public github repository of TensorClus.

4. Conclusion

`TensorClus` is a Python library for three-way co-clustering. It is convenient and straightforward by proposing a panel of the tensor (co)-clustering methods, under a permissive license. It is simple and provides several tools for data loading and visualization. The library offers some illustrative examples to compare `TensorClus` with tensor-decomposition approaches combined to popular clustering methods. Thereby, the proposed implementation allows to easily interface with other python packages such as `Numpy`, `Tensorly`, `TensorD` and `Coclust`. For future work, we intend to extend the library by introducing tri-clustering methods and targeting further improvements in performance using GPU computations.

References

[1] A. Smilde, R. Bro, P. Geladi, Multi-way analysis: applications in the chemical sciences, John Wiley & Sons, 2005.

[2] J. Kossaifi, Y. Panagakis, A. Anandkumar, M. Pantic, Tensorly: Tensor learning in python, J. Mach. Learn. Res. 20 (1) (2019) 925–930.

[3] L. Hao, S. Liang, J. Ye, Z. Xu, TensorD: A tensor decomposition library in tensorflow, Neurocomputing 318 (2018) 196 – 200.

[4] G. Govaert, M. Nadif, Co-clustering: models, algorithms and applications, John Wiley & Sons, 2013.

[5] I. S. Dhillon, S. Mallela, D. S. Modha, Information-theoretic co-clustering, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, p. 89–98.

[6] L. Labiod, M. Nadif, Co-clustering under nonnegative matrix tri-factorization, in: International Conference on Neural Information Processing, 2011, pp. 709–717.

[7] L. Labiod, M. Nadif, A unified framework for data visualization and coclustering, IEEE transactions on neural networks and learning systems 26 (9) (2014) 2194–2199.

[8] S. Affeldt, L. Labiod, M. Nadif, Ensemble block co-clustering: A unified framework for text data, in: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, pp. 5–14.

[9] M. Nadif, F. Role, Unsupervised and self-supervised deep learning approaches for biomedical text mining, Briefings in Bioinformatics 22 (2) (2021) 1592–1603.

[10] M. Ailem, F. Role, M. Nadif, Model-based co-clustering for the effective handling of sparse data, Pattern Recognition 72 (2017) 108 – 122.

[11] M. Ailem, F. Role, M. Nadif, Sparse poisson latent block model for document clustering, IEEE Transactions on Knowledge and Data Engineering 29 (7) (2017) 1563–1576.

[12] G. Govaert, M. Nadif, Mutual information, phi-squared and model-based co-clustering for contingency tables, Advances in Data Analysis and Classification 12 (3) (2018) 455–488.

[13] A. Salah, M. Nadif, Directional co-clustering, Adv. Data Anal. Classif. 13 (3) (2019) 591–620.

[14] H.-H. Bock, Co-clustering for object by variable data matrices, in: Advanced Studies in Behaviormetrics and Data Science, Springer, 2020, pp. 3–17.

[15] F. Role, S. Morbieu, M. Nadif, Coclust: A python package for co-clustering, Journal of Statistical Software 88 (7) (2019) 1–29.

[16] T. Wu, A. R. Benson, D. F. Gleich, General tensor spectral co-clustering for higher-order data, in: Advances in Neural Information Processing Systems 29, Curran Associates, Inc., 2016, pp. 2559–2567.

[17] S. Feizi, H. Javadi, D. Tse, Tensor biclustering, in: Advances in Neural Information Processing Systems 30, 2017, pp. 1311–1320.

[18] R. Boutalbi, L. Labiod, M. Nadif, Tensor latent block model for co-clustering, International Journal of Data Science and Analytics 10 (2020) 161 – 175.

[19] R. Boutalbi, L. Labiod, M. Nadif, Sparse tensor co-clustering as a tool for document categorization, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2019, pp. 1157–1160.

[20] L. R. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika 31 (3) (1966) 279–311.

[21] Z. Xu, F. Yan, Y. Qi, Infinite tucker decomposition: Nonparametric bayesian models for multiway data analysis, in: International Conference on International Conference on Machine Learning, 2012, p. 1675–1682.

[22] R. Boutalbi, L. Labiod, M. Nadif, Co-clustering from tensor data, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2019, pp. 370–383.

[23] A. Strehl, J. Ghosh, Cluster ensembles—a knowledge reuse framework for combining multiple partitions, Journal of machine learning research 3 (Dec) (2002) 583–617.

[24] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, Yong ; Ma, Properties of the hubert-arabie adjusted rand index., IEEE Transactions on Pattern Analysis and Machine Intelligences 35 (1) (2013) 171 – 184.

[25] R. Okuta, Y. Unno, D. Nishino, S. Hido, C. Loomis, Cupy: A numpy-compatible library for nvidia gpu calculations, in: Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS), 2017.

Current code version

Nr.	Code metadata description	Please fill in this column
C1	Current code version	V0.0.1
C2	Permanent link to code/repository used of this code version	https://github.com/boutalbi/TensorClus
C3	Legal Code License	BSD 3-Clause License
C4	Code versioning system used	Git
C5	Software code languages, tools, and services used	Python (>= 3.6)
C6	Compilation requirements, operating environments & dependencies	Python (>= 3.6); packages: scikit-learn, co-clust, tensorflow, numpy, pandas, matplotlib, tensorly, tensorD. It supports major operating systems namely Microsoft Windows, MacOS, and Ubuntu.
C7	If available Link to developer documentation/manual	For example: https://tensorclus.readthedocs.io/en/latest/ and https://pypi.org/project/TensorClus/
C8	Support email for questions	boutalbi.rafika@gmail.com

Table 2: Code metadata of TensorClus