




TLViz: Visualising and analysing tensor decomposition models with Python

Marie Roald ^{1,2¶} and Yngve Mardal Moe ^{*3}

1 Simula Metropolitan Center for Digital Engineering, Norway 2 Oslo Metropolitan University, Norway 3 Independent Researcher, Norway ¶ Corresponding author

DOI: [10.21105/joss.04754](https://doi.org/10.21105/joss.04754)

Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Fabian-Robert Stöter](#) 

Reviewers:

- [@sara-02](#)
- [@yiitozer](#)

Submitted: 26 May 2022

Published: 04 November 2022

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

Introduction

Multi-way data, also known as tensor data or data cubes, occur in many applications, such as text mining ([Bader et al., 2008](#)), neuroscience ([Andersen & Rayens, 2004](#)) and chemical analysis ([Bro, 1997](#)). Uncovering the meaningful patterns within such data can provide crucial insights into the data source, and tensor decompositions have proven an effective tool for this task. In particular, the PARAFAC model, also known as CANDECOMP/PARAFAC (CP) or the canonical polyadic decomposition (CPD), has shown great promise for extracting interpretable components. PARAFAC has, for example, extracted topics from an email corpus ([Bader et al., 2008](#)) and chemical spectra from fluorescence spectroscopy data ([Bro, 1997](#)). For a thorough introduction to tensor methods, we refer the reader to ([Tamara G. Kolda & Bader, 2009](#)) and ([Bro, 1997](#)). The goal of TensorLy-Visualisation (TLViz) is to provide utilities for analysing, visualising and working with tensor decompositions for data analysis in Python.

Statement of need

Python has become a language of choice for data science for both industrial and academic research. Open source tools, such as scikit-learn ([Pedregosa et al., 2011](#)) and Pandas ([McKinney, 2010](#)) have made a variety of machine learning methods accessible within Python. Recently, TensorLy has made tensor methods also available in Python ([Kossaifi et al., 2019](#)), providing seamless integration of multi-way data mining methods within the python scientific environment. However, while TensorLy is an open-source, community-driven package for calculating tensor decompositions, it does not include tools for analysing or visualising the tensor decomposition models. Because tensor decompositions provide powerful tools to extract insight from multi-way data, effective visualisations are crucial as they are needed to communicate this insight. Furthermore, visualisation and evaluation are essential steps in the multi-way analysis pipeline — without tools for these steps, we cannot find suitable models.

There is, to our knowledge, no free open source software (FOSS) that facilitates all these steps. Some tools cover part of this scope, such as Tensor Toolbox for MATLAB ([Tamara G. Kolda & Bader, 2006](#)), which provides some functionality for model evaluation; and the N-Way toolbox for MATLAB ([Andersson & Bro, 2000](#)) and Higher Order Tensor ToolBOX (HOTTBOX) for Python ([Kisil et al., 2021](#)), which both provide limited functionality for model evaluation and visualisation. Finally, PLSToolbox covers most of our scope, but it is a closed-source commercial software. There is, therefore, a growing need for FOSS tools for the visualisation and evaluation of tensor decompositions.

*Co-first author

Example

The PARAFAC model is straightforward, but there are several pitfalls to consider. Two main pitfalls are the scaling and permutation indeterminacy (Bro, 1997). The order of the components does not matter, and the magnitude of one factor matrix can be scaled arbitrarily so long as another factor matrix is inversely scaled (an even number of components may even change sign). Therefore, it can be time-consuming and cumbersome to go from having fitted a PARAFAC model to visualising it. TLViz takes care of these hurdles in a transparent way. The code below shows how easily we can use TLViz and TensorLy to analyse a fluorescence spectroscopy dataset.

```
import tlviz
import matplotlib.pyplot as plt
from tensorly.decomposition import import parafac

def fit_parafac(dataset, num_components, num_inits):
    model_candidates = [
        parafac(dataset.data, num_components, init="random", random_state=i)
        for i in range(num_inits)
    ]
    model = tlviz.multimodel_evaluation.get_model_with_lowest_error(
        model_candidates, dataset
    )
    return tlviz.postprocessing.postprocess(model, dataset)

data = tlviz.data.load_aminoacids()
cp_tensor = fit_parafac(data, 3, num_inits=3)
tlviz.visualisation.components_plot(cp_tensor)
plt.show()
```

Loading Aminoacids dataset from:

Bro, R, PARAFAC: Tutorial and applications, Chemometrics and Intelligent
↪ Laboratory Systems, 1997, 38, 149-171

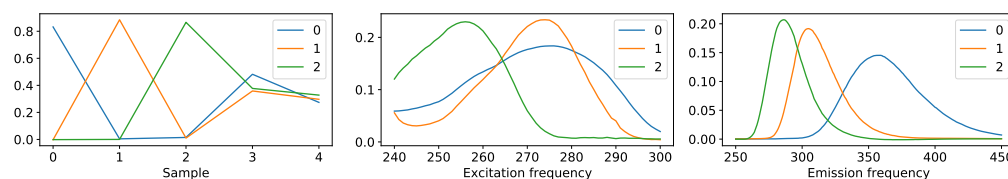


Figure 1: An example figure showing the component vectors of a three-component PARAFAC model fitted to a fluorescence spectroscopy dataset

The above code uses TensorLy to fit five three-component PARAFAC models to the data. Then it uses TLViz to do the following steps:

1. Select the model that gave the lowest reconstruction error.
2. Normalise the component vectors, storing their magnitude in a separate weight-vector.
3. Permute the components in descending weight (i.e. signal strength) order.
4. Flip the components, so they point in a logical direction compared to the data.
5. Convert the factor matrices into Pandas DataFrames with logical indices.
6. Plot the components using matplotlib.

All these steps are well documented with references to the literature. This makes it easy for new practitioners to analyse multi-way data without falling for known pitfalls.

Overview

TLViz follows the procedural paradigm, and all of TLViz's functionality lies in functions separated over 8 public modules:

1. `tlviz.data` - various open datasets
2. `tlviz.factor_tools` - transforms and compares PARAFAC models without using reference data
3. `tlviz.model_evaluation` - evaluates a PARAFAC model
4. `tlviz.multimodel_evaluation` - compares and evaluates multiple models at once
5. `tlviz.outliers` - finds data points that may be outliers
6. `tlviz.postprocessing` - post-processes PARAFAC models, usually used before visualising
7. `tlviz.utils` - general utilities that can be useful (e.g. forming dense tensors from decompositions)
8. `tlviz.visualisation` - visualising component models

A core design choice behind TLViz is how to store metadata. Consider the example above. It is necessary to know the values along the x-axis to interpret these components. Therefore, we use xarray DataArrays to store data tensors (Hoyer & Hamman, 2017), keeping the correct indices for each tensor mode (i.e. axis), and Pandas DataFrames to store factor matrices. However, TensorLy works with NumPy arrays. TLViz, therefore, provides useful tools to add the coordinates from an xarray DataArray onto the factor matrices obtained with TensorLy. Furthermore, all functions of TLViz support both labelled and unlabelled decompositions (i.e. DataFrames and NumPy arrays) and will use the labels whenever possible.

The visualisation module uses matplotlib to create the plots, and the goal of this module is to facilitate fast prototyping and exploratory analysis. However, TLViz can also seamlessly convert factor matrices into tidy tables, which are better suited for visualisation libraries such as Seaborn (Waskom, 2021) and Plotly Express, thus making it painless to combine tensor decomposition with the plotting library that best suits the user's specific needs.

To be easy to use, scientific software should have thorough and accurate documentation. For TLViz, this means two things: Explaining what the code does and why this is important. Therefore, we have taken care to review the literature, citing original sources wherever possible. By gathering the references together with the API documentation and examples, we make it straightforward for researchers new to the field to discover suitable references for their analysis.

The gallery of examples provided by TLViz explains the tools included in the package and how to use them. The gallery contains, among others, examples that explain how to select the number of components in PARAFAC models, how to detect outliers and how to combine TLViz with Plotly to get interactive visualisations. All examples include relevant references, making it easy for new practitioners to get started.

Acknowledgements

We want to thank Jean Kossaifi for valuable feedback and for creating the TensorLy project.

References

- Andersen, A. H., & Rayens, W. S. (2004). Structure-seeking multilinear methods for the analysis of fMRI data. *NeuroImage*, 22(2), 728–739. <https://doi.org/10.1016/j.neuroimage.2004.02.026>
- Andersson, C. A., & Bro, R. (2000). The n-way toolbox for MATLAB. *Chemometrics and Intelligent Laboratory Systems*, 52(1), 1–4. [https://doi.org/10.1016/S0169-7439\(00\)00001-0](https://doi.org/10.1016/S0169-7439(00)00001-0)

00071-X

- Bader, B. W., Berry, M. W., & Browne, M. (2008). Discussion tracking in enron email using PARAFAC. In *Survey of text mining II* (pp. 147–163). Springer. https://doi.org/10.1007/978-1-84800-046-9_8
- Bro, R. (1997). PARAFAC. Tutorial and applications. *Chemometrics and Intelligent Laboratory Systems*, 38(2), 149–171. [https://doi.org/10.1016/S0169-7439\(97\)00032-4](https://doi.org/10.1016/S0169-7439(97)00032-4)
- Hoyer, S., & Hamman, J. (2017). Xarray: ND labeled arrays and datasets in python. *Journal of Open Research Software*, 5(1). <https://doi.org/10.5334/jors.148>
- Kisil, I., Calvi, G. G., Dees, B. S., & Mandic, D. P. (2021). HOTTBOX: Higher order tensor ToolBOX. *arXiv Preprint arXiv:2111.15662*. <https://doi.org/10.48550/arXiv.2111.15662>
- Kolda, Tamara G., & Bader, B. W. (2006). *MATLAB tensor toolbox, version 00*. <https://www.osti.gov/biblio/1230898>
- Kolda, Tamara G., & Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Review*, 51(3), 455–500. <https://doi.org/10.1137/07070111X>
- Kossaifi, J., Panagakis, Y., Anandkumar, A., & Pantic, M. (2019). TensorLy: Tensor learning in python. *Journal of Machine Learning Research*, 20(26), 1–6. <http://jmlr.org/papers/v20/18-277.html>
- McKinney, Wes. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61). <https://doi.org/10.25080/Majora-92bf1922-00a>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, Édouard. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85), 2825–2830. <http://jmlr.org/papers/v12/pedregosa11a.html>
- Waskom, M. L. (2021). Seaborn: Statistical data visualization. *Journal of Open Source Software*, 6(60), 3021. <https://doi.org/10.21105/joss.03021>