Non-negative Tensor Factorizations and Applications

Neriman Tokcan[1,2]

Abstract. In this brief report, we discuss the applications of non-negative factorizations explored during the "AGATES: Tensors in statistics, optimization and machine learning" semester. We provide an overview of non-negative tensor factorizations and their applications across diverse fields.

Key words. tensors, multi-dimensional data analysis, tensor decompositions.

1. Introduction.

Tensors have become more common in recent years because they offer a natural and powerful way to represent complex and high-dimensional data structures that arise in many fields, including computer vision, natural language processing, neuroscience, genomics, recommender systems, and social network analysis [17, 22, 26, 32, 10, 21, 33, 15]. Tensors can capture the interactions between multiple modes of data, such as time, space, and frequency. They can extract meaningful patterns, relationships, and dependencies that may be difficult to discover using traditional matrix-based methods. As a result, tensor-based methods have become an increasingly important tool for data analysis and machine learning in many fields, and are likely to grow in importance as the complexity and scale of data increase.

Several matrix-based techniques have been employed for analyzing multi-dimensional data. One approach is to unfold the data into a single large matrix, while another method is to slice it into multiple matrices before applying matrix factorization [17]. However, these techniques do not fully utilize the intrinsic multi-way structure of the data, resulting in the loss of significant information regarding the interplay between different modalities. Tensor factorizations, as a multi-way extension of matrix factorizations, address this issue by preserving the multi-way structure of the data, allowing for a more accurate representation of the data and a better identification of latent features. Popular tensor decomposition methods include Candecomp/Parafac, Tucker, and their variants, which can extract latent factors, discover patterns, and reduce the dimensionality of tensor data [17]. We can place certain limitations on the models depending on their intended usage. For many applications, imposing a non-negativity restriction is common practice as it can enhance the interpretability of the model.

In this report, we will focus on non-negative tensor factorization (NTF) which is a multi-way extension of non-negative matrix factorization (NMF). Non-negative Matrix Factorization has a rich history spanning over 40 years, originally referred to as positive matrix factorization or non-negative rank factorization [5, 6, 23]. NMF gained significant popularity when Lee and Seung [20] demonstrated its ability to decompose images of visual objects into meaningful parts, effectively "learning the parts of objects". Since then, NMF has been widely applied in various fields including image processing, audio signal processing, text mining, and bioinformatics, for purposes such as clustering, feature extraction, and pattern recognition [7, 13].

[1]Department of Mathematics, University of Massachusetts,Boston, Massachusetts
[2]The Eli and Edythe L. Broad Institute of MIT and Harvard, Cambridge, Massachusetts

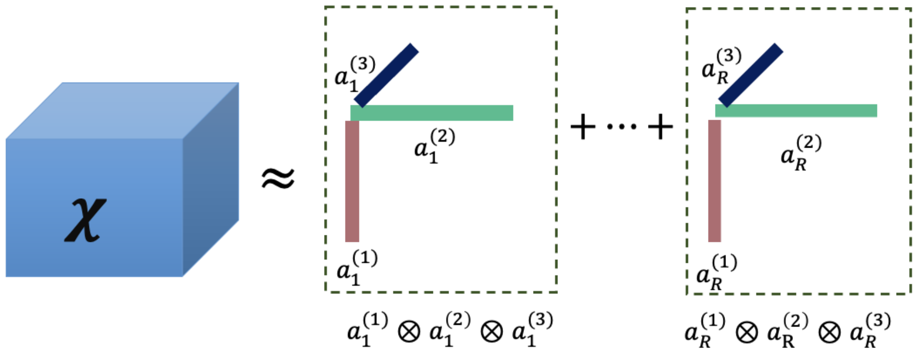

Figure 1.1. *CP decomposition of a 3-way tensor $\mathcal{X}$*

However, NMF is limited to handling two-dimensional data, while NTF can handle multi dimensional data. As previously discussed, applying NMF to higher-dimensional arrays can result in the loss of interactions between different modalities. The question of whether NMF is unique is closely related to whether the underlying latent factors are the sole interpretation of the data or if alternative interpretations are possible. However, NMF is generally non-unique, meaning that there can be multiple sets of factor matrices that can represent the same data [13]. Additional constraints can be added to help make the factorization unique [28, 24]. On the other hand, NTF is unique under milder conditions [7].

1.1. Notation and Preliminaries.

We will introduce the key concepts and notations related to tensors, which will serve as the foundation for the rest of the paper. Most of our notation is borrowed from [17].

Tensors are multi-dimensional (multi-way) arrays, which extend the matrices to higher dimensions. The number of dimensions (modes or ways) of a tensor is referred to as its *order*. Tensors with an order of 1 are vectors, and those with an order of 2 are matrices. Tensors with an order greater than 2 are referred to as higher-order tensors. Throughout this paper, we use lowercase letters $x \in \mathbb{R}^p$ to represent vectors, uppercase letters $X \in \mathbb{R}^{p\times q}$ to represent matrices and capital calligraphic letters $\mathcal{X} \in \mathbb{R}^{p_1\times p_2\times\ldots\times p_d}$ to represent higher-order tensors. We denote the $(i_1, i_2, \ldots, i_d)$-th entry of an order-d tensor $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_d}$ as $\mathcal{X}_{i_1 i_2\ldots i_d}$.

The Frobenius norm of a tensor is similar to the matrix Frobenius norm, as it is defined as the square root of the sum of the squares of all its elements:

$$\|\mathcal{X}\|_F = \sqrt{\sum_{i_1=1}^{I_1}\sum_{i_2=1}^{I_2}\cdots\sum_{i_d=1}^{I_d} \mathcal{X}^2_{i_1 i_2\ldots i_d}}. \tag{1.1}$$

A rank-1 matrix $X \in \mathbb{R}^{p\times q}$ can be written as an outer product of two vectors $u \in \mathbb{R}^p$ and $v \in \mathbb{R}^q$ such that $X = u \otimes v = uv^T$. Similarly , a *rank-1 tensor* $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_d}$ can be expressed as the outer product d vectors, i.e., $\mathcal{X} = u_1 \otimes u_2 \otimes \ldots \otimes u_d$, $u_i \neq 0 \in \mathbb{R}^{I_i}$, $1 \leq i \leq d$. Such rank-1 tensors are called *simple* or *pure*. The process of decomposing a tensor into the sum of rank-1 tensors is called *tensor decomposition*, which is an extension of the concept of matrix factorization. There are many different tensor decomposition methods, such as the Canonical Polyadic (CP) decomposition, the Tucker decomposition, the PARAFAC2 decomposition, and the Hierarchical Tucker decomposition, among others [17, 8]. Canonical Polyadic (CP) decomposition, also known as Parallel Factor Analysis(PARAFAC), is a common method for decomposing tensors. It aims to express a tensor as a sum of rank-1 tensors. For a tensor $\mathcal{X} \in \mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_d}$, CP decomposition can be written as:

$$\mathcal{X} \approx \sum_{r=1}^{R} \lambda_r a_r^{(1)} \otimes a_r^{(2)} \otimes \ldots \otimes a_r^{(d)} \tag{1.2}$$

where $\lambda_r \in \mathbb{R}$ and $a_r^{(i)} \in \mathbb{R}^{I_i}, 1 \leq i \leq d, 1 \leq r \leq R$. The smallest such integer r is called the *rank (real rank)* of the tensor. For a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_d}$, the *Tucker decomposition* can be written as:

$$\mathcal{X} = \sum_{i_1=1}^{R_1} \sum_{i_2=1}^{R_2} \ldots \sum_{i_d=1}^{R_d} \mathcal{G}_{i_1,i_2,\ldots,i_d} a_{i_1}^{(1)} \otimes a_{i_2}^{(2)} \otimes \ldots \otimes a_{i_d}^{(d)} + \mathcal{E}, \tag{1.3}$$

where $A^{(i)} = [a_1^{(i)} \ldots a_{R_i}^{(i)}] \in \mathbb{R}^{I_i \times R_i}$ is the *factor matrix* along the $i-$th mode, $R_i \leq I_i, 1 \leq i \leq d$. The error-tensor $\mathcal{E}$ has the same size as the original tensor. The reduced *core tensor* $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times \ldots \times R_d}$ can be considered as "generalized singular values" as it gives the multilinear relations between different modalities. We can place certain limitations on the core and factor matrices depending on their usage. The CP decomposition can be viewed as a specific instance of the Tucker decomposition, where $R_1 = R_2 = \ldots = R_d$ and $\mathcal{G}_{i_1,i_2,\ldots,i_d} \neq 0$ only when $i_1 = i_2 = \cdots = i_d$ (core is super-diagonal).

The optimization problem given in Equation (1.3) is typically solved using iterative algorithms, such as multiplicative updates, alternating least squares, or gradient descent [17, 1, 29]. These algorithms iteratively update the factor matrices and vectors until convergence. Classical tensor decomposition methods rely on maximum likelihood estimation (MLE), which provides a single-point estimate of the underlying parameters and does not account for parameter uncertainty [25, 12]. MLE can also be sensitive to outliers and may not be robust to deviations from the assumed distributional form. Bayesian approaches for higher-order arrays have additional advantages over MLE, such as incorporating prior knowledge, performing model selection, and quantifying uncertainty in the parameter estimates [25, 12]. Although probabilistic approaches are promising, their details are beyond the scope of this report.

2. Applications of Non-Negative Tensor Factorizations.

Non-negative tensor factorization (NTF) is a powerful method for decomposing high-dimensional data into a set of interpretable latent factors that capture the underlying patterns in the data [15]. NTF enforces non-negativity constraints on both the core tensor and the latent factors in Equation (1.3). This constraint is especially useful in applications such as image and signal processing, where the factors represent physical quantities that must be non-negative. The focus of this section will be primarily on the applications of tensor methods in image processing and genomics.

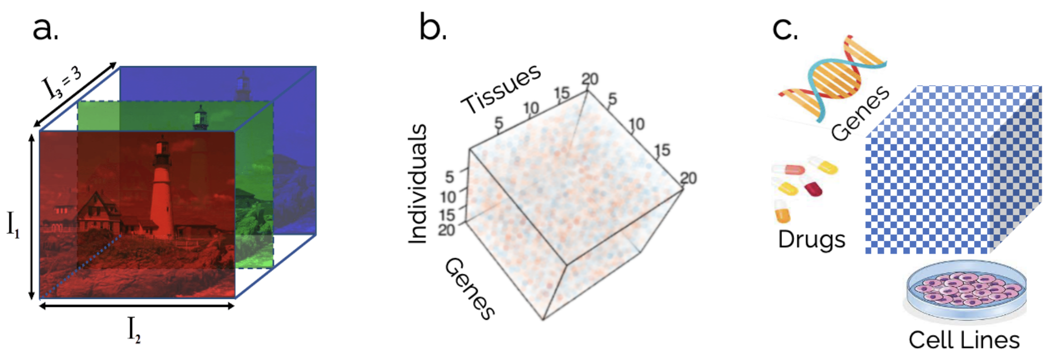

Figure 1.2. *Examples for 3-way tensors.a)Colored images.b) Multi-tissue and multi-individual gene expression data (Image source: [32]). c) Perturbational dataset for drug, cell line, and gene interactions.*

2.1. Image Processing.

In image processing, NTF can be used for image compression, denoising, and feature extraction [33].

Let $A_t,\ t = 1, ..., k$ be training images of dimension $n \times m$. Non-negative Matrix Factorization is a commonly used technique to decompose the training images of an object class into a basis of local parts. The first step is vectorizing the images. The V be the $(n \times m) \times k$ matrix whose columns correspond to vectorized images. Then NMF factorizes V such that:

$$V \approx WH,\ W \geq 0, H \geq 0. \tag{2.1}$$

The columns of W form the basis factors (latent factors) and due to non-negative constraint, both the basis factors and mixing coefficient tend to be sparse.

However, vectorizing images results in the loss of local image structure or spatial redundancy, hence using NMF on the vectorized image may not yield the underlying generative parts, even in the case of a perfect fit.

For the tensor approach, we do not vectorize images. We will form a tensor $\mathcal{X}$ of size $(n \times m) \times k$ such that $\mathcal{X}[:, :, i] = A_i,\ 1 \leq i \leq k$. We want to factorize $\mathcal{X}$ and consider the following least-squares problem:

$$min_{u_i, v_i, w_i} ||\mathcal{X} - \sum_{i=1}^{R} u_i \otimes v_i \otimes w_i||_F. \tag{2.2}$$

such that $u_i, v_i, w_i \geq 0,\ 1 \leq i \leq R$. We get three-factor matrices: $U = [u_1, u_2, ..., u_R], V = [v_1, v_2, ..., v_R], W = [w_1, w_2, ..., w_R]$. We can easily capture the approximation to our 2D images as:

$$A_t = U \Lambda_t V^T, \text{where } \Lambda_t = diag(w_{1t}, w_{2t}, \dots, w_{rt}). \tag{2.3}$$

In general, NMF is not unique. Additional constraints can be added to help make the factorization unique, e.g., sparsity or minimum determinant [28, 24]. On the other hand, NTF

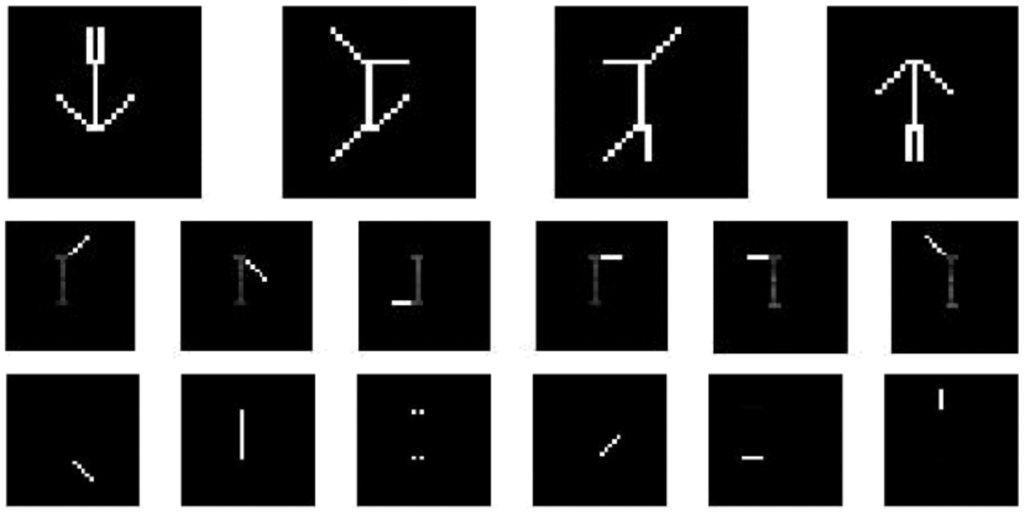

Figure 2.1. *In [11], a set of 256 images from the Swimmer Library are used to compare the factors generated by Nonnegative Matrix Factorization (NMF) and Nonnegative Tensor Factorization (NTF) methods. The sample image is presented in the top row, with the NMF factors displayed in the middle row and the NTF factors in the bottom row. The NMF factors reveal ghosts of the invariant parts, particularly the torso, which contaminates the sparse representation.*

is unique under milder conditions [7]. Both NMF and NTF have the potential to produce sparse factor matrices due to the non-negativity constraint. However, NTF has the added benefit of being a separable decomposition, factors are more compressed than NMF factors.

In [11], a comparison of NMF and NTF methods is presented using a Swimmer image set consisting of 256 images of dimensions 32 x 32. Each image contains a central "torso" of 12 pixels and four "limbs" of 6 pixels, which can be in one of four positions. The NMF method, which attempts to find 17 factors running over the image set, successfully resolves the local parts but fails on the invariant torso, which appears in the same position throughout the entire set and thus appears as a "ghost" in all the factors (Figure 2.1). In contrast, the NTF method provides a unique factorization and correctly resolves all 17 parts, including the invariant torso (Figure 2.1).

Below is a comparison table for the properties of NMF and NTF for images:

	NMF	**NTF**
Representation	Vectorizes images	It represents the images the collection as a 3-way array
Uniqueness	Not unique	Unique under mild conditions
Sparse	Yes (can result in sparse factors)	Yes (can result in sparse factors)
Separable	Not separable	Separable

Table 2.1
Comparision of NMF and NTF for image processing

3. Genomics.

NTF is a widely-used tool in genomics for analyzing large-scale data sets, including microarrays, single-cell RNA sequencing data, and bulk-RNA sequencing data [27, 16, 32]. These data sets typically consist of gene expression values across multiple samples, each representing a different biological condition or state. By applying NTF to these data tensors, researchers can identify the biological factors contributing to gene expression patterns observed across samples.

The non-negativity constraint of NTF is particularly useful in genomics, as it ensures that the resulting factor matrices contain only positive values corresponding to gene expression levels. This constraint allows researchers to interpret the factor matrices as representing biologically relevant processes, such as metabolic or signaling pathways. In addition, NTF is capable of handling missing or noisy data, which are commonly encountered in genomic data sets [3, 9]. NTF can also identify co-regulated genes or co-occurring mutations, providing insights into the genetic mechanisms underlying disease [27].

3.1. Multi-Tissue Multi-Individual Genomics Data.

A typical multi-tissue experiment collects gene expression profiles from different individuals in a number of different tissues, and variation in expression levels often results from complex interactions among genes, individuals, and tissues.

Clustering has emerged as a valuable technique for identifying hidden patterns in high-dimensional expression data. Conventional approaches such as K-means, principal component analysis (PCA), and t-distributed stochastic neighbor embedding (t-SNE) have been commonly used for this purpose [32, 30]. However, these methods rely on the assumption that gene expression patterns remain consistent across different contexts or that samples are independent and homogeneous and they arrange the high-dimensional data into matrices.

Structuring the high-dimensional genomics data as matrices poses several challenges [19, 32]:

- It may limit the ability to identify tissue-specific and individual-specific patterns.
- Inferring gene modules separately for each tissue could overlook shared characteristics among tissues and hinder the identification of differentially-expressed genes.
- Disregarding individual heterogeneity, including biological factors such as race, gender, and age, can undermine the accuracy of estimating correlations between genes and/or tissues.

Tensor based approaches have been proposed to handle heterogeneity in each mode and to learn the clustering patterns across different modes of the data in an unsupervised manner analogous to PCA and SVD [16, 32, 27].

The study described in [32] utilized NTF on RNA-seq data from the Genotype-Tissue Expression (GTEx) project, which encompassed samples from 544 individuals across 53 human tissues at the time of the study. The results demonstrated the successful identification of three-way interactions with high accuracy and robustness.

Gene expression data from n_G genes, n_I individuals, and n_T tissues is modeled by a 3-way tensor $\mathcal{Y} \in \mathbb{R}^{n_G \times n_I \times n_T}$, and a low rank r approximation is obtained through NTF to identify latent factors:

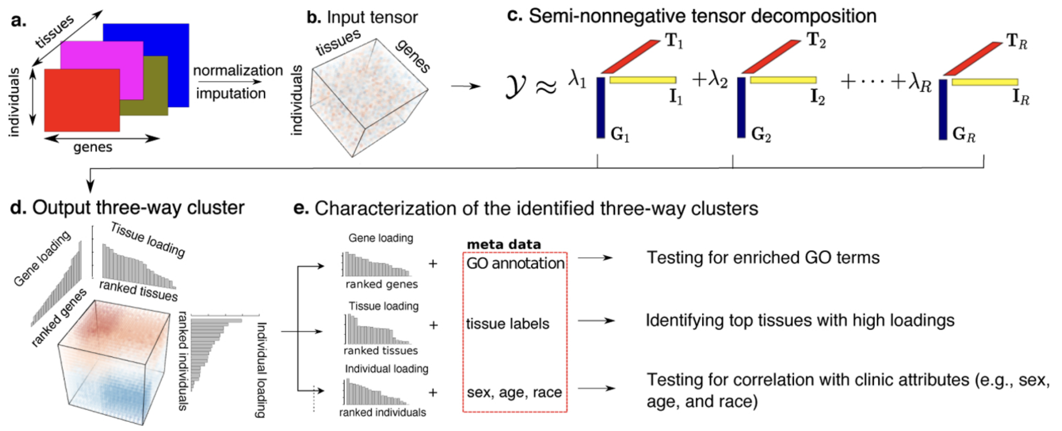

Figure 3.1. *Tensor factorization based pipeline for high-dimensional genomics data analysis given in [32].(a) Multi-tissue multi-individual gene expression data.(b) Input expression tensor after normalization and imputation.(c) Non-negative factorization of expression tensor into latent factors along gene, tissue, and individual modalities. (d) Representation of three-way clusters corresponding to each modality. (e) Identification of sources of variation in each latent factor by utilizing metadata, such as gene ontology (GO) annotation, tissue labels, and individual-level covariates.*

$$\mathcal{Y} = \sum_{r=1}^{R} \lambda_r G_r \otimes I_r \otimes T_r + \mathcal{E}, \tag{3.1}$$

where $\lambda_r \in \mathbb{R}_+$ and G_r, I_r, T_r are norm-1 unit vectors; and $\mathcal{E}$ is a noise tensor with each entry $\mathcal{E}_{i,j,k}$ i.i.d $N(0, \sigma_e^2)$. Each factor G_i is associated with a set of weights, which indicate the importance of each gene in the corresponding metagene. G_r corresponds to “eigen-genes”, representing the underlying biological pathways or processes that contribute to the observed gene expression patterns. I_r corresponds to “eigen-individuals”, capturing individual-specific characteristics that affect gene expression, and T_r corresponds to “eigen-tissues”, representing the shared biological attributes across different tissues.

The paper [32], highlights cases where PCA and matrix-based methods fail to capture mode-specific grouping and three-way interactions. It is noted that matrix PCA does not provide much insight into gene/tissue clustering due to the destruction of the three-way structure encoded in higher-order tensor data caused by matricization.

4. Toolboxes.

There are several open-source tensor decomposition toolboxes available for various programming languages [17]. In this section, we provide an overview of some of the most popular toolboxes along with their features and supported decompositions.

Tensor Toolbox [4]: Tensor Toolbox is a MATLAB toolbox that enables users to perform various tensor computations and decompositions. It supports various tensor decompositions such as CANDECOMP/PARAFAC (CP), Tucker, and Higher-Order Singular Value Decomposition (HOSV). The toolbox also includes alternative decompositions such as Poisson

Tensor Factorization via alternating Poisson regression (APR), Generalized CP (GCP) tensor factorization, and symmetric CP tensor factorization. Tensor Toolbox has been around for almost two decades and has been widely used and cited in numerous research studies, which contributes to its popularity and reputation as a reliable and robust tool for tensor analysis.

TensorLy [18]: TensorLy is an open-source, BSD-licensed library for tensor computations in Python. It is compatible with multiple backends such as NumPy, PyTorch, TensorFlow, JAX, Apache MXNet, and CuPy. It has minimal dependencies and offers thorough documentation. It supports various core tensor operations, decompositions, and tensor regressions. It also supports tensorized deep learning with TensorLy-Torch. It has an easy-to-use API, which makes it a popular choice for researchers and practitioners.

Tensorlab [31]: Tensorlab is a MATLAB toolbox that provides various tools for tensor computations, coupled tensor decompositions, and complex optimization. It supports dense, incomplete, sparse, and structured datasets and offers a library of pre-implemented transformations and structures. Tensor decompositions of both real and complex tensors such as the canonical polyadic decomposition (CPD), multilinear rank-$(L_r, L_r, 1)$ terms, low multilinear rank approximation (LMLRA), multilinear singular value decomposition (MLSVD), and block term decomposition (BTD) can be computed. Tensorlab also offers a tensorization framework, structured data fusion (SDF), and complex optimization algorithms. Demos have been developed to illustrate the power of the toolbox in different case studies.

Other toolboxes: Multiple toolboxes are available for tensor computations, including the N-way Toolbox, TT-Toolbox, and Probabilistic Tensor Decomposition Toolbox. The N-way Toolbox [2] is capable of handling multi-modal data and offers various methods for fitting models such as PARAFAC, Tucker, N-PLS, GRAM, and TLD. The TT-Toolbox [14] is a MATLAB-based toolbox for tensor computations using the Tensor Train (TT) format, which is an efficient way of representing high-dimensional tensors with low-parametric methods. The probabilistic tensor decomposition toolbox [12] is also a MATLAB toolbox for tensor decomposition using Variational Bayesian inference and Gibbs sampling.

5. Acknowledgements. This work is partially supported by the Thematic Research Programme "Tensors: geometry, complexity and quantum entanglement", University of Warsaw, Excellence Initiative – Research University and the Simons Foundation Award No. 663281 granted to the Institute of Mathematics of the Polish Academy of Sciences for the years 2021-2023.

REFERENCES

[1] E. Acar, T. Kolda, D. M. Dunlavy, *All-at-once optimization for coupled matrix and tensor factorizations* (2011), arXiv:1105.3422

[2] C. A. Andersson and R. Bro, *The N-way toolbox for MATLAB, Chemometrics and Intelligent Laboratory Systems* (200), 52, pp. 1–4, https://www.mathworks.com/matlabcentral/fileexchange/1088-the-n-way-toolbox.

[3] T. S. Andrews, and M. Hemberg, *False signals induced by single-cell imputation*(2019), F1000Research, 7, 1740. doi: 10.12688/f1000research.16613.2

[4] B. W. Bader, T. G. Kolda and others, *MATLAB Tensor Toolbox, Version 3.5*, available online at https://www.tensortoolbox.org, 20XX.

[5] S. L. Campbell and G. D. Poole, *Computing nonnegative rank factorizations* (1981), Linear Algebra Appl.,

vol. 35, pp. 175–182.

[6] J.-C. Chen, *The nonnegative rank factorizations of nonnegative matrices*(1984) Linear Algebra Appl., vol. 62, pp. 207–217.

[7] A. Cichocki, R. Zdunek, A. H. Phan, and S. Amari, *Nonnegative matrix and tensor factorizations:* (2009), John Wiley & Sons.

[8] K. Fonał and R. Zdunek, *Fast Recursive Nonnegative Standard and Hierarchical Tucker Decomposition*,(2019), IEEE Signal Processing Letters, vol. 26, no. 9, pp. 1265-1269, doi: 10.1109/LSP.2019.2926845.

[9] J. Dauwels, L. Garg, A. Earnest, and L.K. Pang, *Tensor factorization for missing data imputation in medical questionnaires*(2012), In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 2109-2112). doi: 10.1109/ICASSP.2012.6288327.

[10] E. Frolovand I. Oseledets, I., *Tensor methods and recommender systems* (2017), WIREs Data Mining Knowl Discov, 7: e1201. https://doi.org/10.1002/widm.1201.

[11] T. Hazan, S. Polak and A. Shashua, *Sparse image coding using a 3D non-negative tensor factorization*, Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1, Beijing, China(2005), pp. 50-57 Vol. 1, doi: 10.1109/ICCV.2005.228.

[12] J. L. Hinrich, K. H. Madsen, and M. Mørup, *The probabilistic tensor decomposition toolbox* (2020), Machine Learning: Science and Technology, available online at https://github.com/JesperLH/prob-tensor-toolbox.

[13] K. Huang, N.D. Sidiropoulos, and A. Swami, *Non-Negative Matrix Factorization Revisited: Uniqueness and Algorithm for Symmetric Decomposition*(2014), Signal Processing, IEEE Transactions on. 62. 211-224. 10.1109/TSP.2013.2285514.

[14] O. Ivan, *Tensor-Train Decomposition* (2011), SIAM J. Scientific Computing. 33. 2295-2317. 10.1137/090752286, available at https://github.com/oseledets/TT-Toolbox.

[15] Y. Ji, Q. Wang, X. Li, and J. Liu, *A Survey on Tensor Techniques and Applications in Machine Learning* (2019), in IEEE Access, vol. 7, pp. 162950-162990, doi: 10.1109/ACCESS.2019.2949814.

[16] I. Jung, M. Kim, S. Rhee, S. Lim, S. Kim, *MONTI: A Multi-Omics Non-negative Tensor Decomposition Framework for Gene-Level Integrative Analysis* (2021), Frontiers in genetics, 12, 682841. https://doi.org/10.3389/fgene.2021.682841.

[17] T. G. Kolda and B. W. Bader, *Tensor decompositions and applications*, SIAM review **51** (2009), no. 3, 455–500.

[18] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic, *TensorLy: Tensor Learning in Python*, Journal of Machine Learning Research (2019), Volume: 20, Issue: 26, Pages: 16, available online at https://tensorly.org/stable/index.html.

[19] D. Lähnemann, J. Köster, E. Szczurek, D. J. McCarthy, S. C. Hicks, M. D. Robinson, C. A. Vallejos, K. R. Campbell, and others, *Eleven grand challenges in single-cell data science*(2020),Genome biology, 21, 1–35.

[20] D. D. Lee and H. S. Seung, *Learning the parts of objects by non-negative matrix factorization* (1999) ,Nature, vol. 401, no. 6755, pp. 788–791.

[21] N. Marin, E. Makhneva, M. Lysyuk, V. Chernyy, I. Oseledets, and E. Frolov, *Tensor-based Collaborative Filtering With Smooth Ratings Scale*(2022), arXiv preprint arXiv:2205.05070.

[22] A. Özdemir, M. A. Iwen, and S. Aviyente, *A multiscale approach for tensor denoising*(2016), IEEE Statistical Signal Processing Workshop (SSP), pages 1–5, June 2016. 7

[23] P. Paatero and U. Tapper, *Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values*, (1994), Environmetrics, vol. 5, no. 2, pp. 111–126.

[24] R. Schachtner, G. Pöppel, and E. W. Lang, *Towards unique solutions of non-negative matrix factorization problems by a determinant criterion*,(2011), Digit. Signal Process., vol. 21, no. 4, pp. 528–534.

[25] A. Schein, J. Paisley, D. Blei, David H. Wallach, *Bayesian Poisson Tensor Factorization for Inferring Multilateral Relations from Sparse Dyadic Event Counts*(2015), In Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15), https://doi.org/10.1145/2783258.278341410.1145/2783258.2783414.

[26] J. Schreiber, T. Durham, J. Bilmes, and W. S. Noble,*Avocado: a multi-scale deep tensor factorization method learns a latent representation of the human epigenome*(2020), Genome Biology, 21(1):81, 2020.

[27] M. Shi, R. Sherpa, L. Klindziuk, S. Kriel, and S. Mollah, *A Non-Negative Tensor Factorization Approach*

to Deconvolute Epigenetic Microenvironment in Breast Cancer (2020), bioRxiv 2020.12.01.406249; doi: https://doi.org/10.1101/2020.12.01.406249.

[28] F. J. Theis, K. Stadlthanner, and T. Tanaka, *First results on uniqueness of sparse non-negative matrix factorization* (2005), in Proc. 13th Eur. SignalProcess. Conf. (EUSIPCO'05), Antalya, Turkey, Sep. 4–8, 2005.

[29] G. Tomasi and R. Bro, *A comparison of algorithms for fitting the parafac model*(2006), Comput. Stat. Data Anal. 50 1700–34.

[30] L.Van Der Maaten and G. Hinton *Visualizing data using t-SNE* (2008), Journal of machine learning research.

[31] N. Vervliet, O.Debals, L. Sorber, V M. Van Barel, and L. De Lathauwer, Tensorlab 3.0 (2016), available online at https://www.tensorlab.net/.

[32] M. Wang, J. Fischer, and Y. S. Song, *Three-way clustering of multi-tissue multi-individual gene expression data using semi non-negative tensor decomposition*(2019), The annals of applied statistics, 13(2), 1103–1127. https://doi.org/10.1214/18-aoas1228

[33] H. Yang, G. He, and Y. Dong, *Nonnegative Tensor Decomposition and It's Applications in Image Processing*, Proceedings of the International Conference on Computer Science and Information Technology, ICCSIT (2008), 212 - 217. 10.1109/ICCSIT.2008.99