Université Claude Bernard Lyon 1

THESE d'HDR DE
L'UNIVERSITE CLAUDE BERNARD LYON 1

Soutenue publiquement le 30/06/2026, par :
Jérémy Cohen

Regularized low-rank approximations

Devant le jury composé de :

ACHARD, Sophie	Directrice de recherche CNRS, Laboratoire Jean Kuntzmann	Examinatrice
BOURGUIGNON, Sébastien	Professeur des universités, Ecole Centrale de Nantes	Rapporteur
FEVOTTE, Cédric	Directeur de recherche CNRS, Institut de Recherche en Informatique de Toulouse	Rapporteur
MASNOU, Simon	Professeur des universités, Université Lyon 1	Examinateur
SALMON, Joseph	Professeur des universités, Université de Montpellier	Rapporteur

Regularized low-rank approximations.

Jeremy E. Cohen

Apr 27, 2026

CONTENTS

Welcome to Jeremy Cohen's "Habilitation à Diriger des Recherches" (HDR) manuscript. This document is a synthesis of the work I have done since my PhD thesis, and took almost two years to complete. The main scientific topic of this manuscript is Low-Rank Approximations, and alongside my personal contributions, I introduce relevant existing results and tools from numerical optimization, having in mind to reuse this material for teaching.

The book contains code snippets with barebone implementations of algorithms, numerical simulations and visualisations. This was done so that most results shown in the manuscript are easily reproduced by the reader. The book is available in .pdf upon request, but it is advised to read the content from this website to get the best interface experience. The philosophy of the book design is further discussed in *How to read this manuscript*.

The scientific content of the book is summarized in the *Summary of HDR contents*. Here is the table of content for quick access.

Fig. 1: (Generated with ChatGPT)

Use of AI tools

Except for the picture above, generative AI has been used in this manuscript only for grammar and style checking, and generate javascript codes for interactive plots.

Many pages contain code executed upon compilation of the book for publication, below is their status.

Document	Modified	Method	Run Time (s)	Status
Howtouse/howtoread	2026-04-27 15:31	force	0.93	✓
introduction/summary	2026-04-27 15:31	force	5.02	✓
part1/AlternatingOptimization	2026-04-27 15:31	force	7.84	✓
part1/lra	2026-04-27 15:31	force	2.73	✓
part1/nnls	2026-04-27 15:31	force	3.09	✓
part2/Applications_of_rLRA/AMT	2026-04-27 15:32	force	37.98	✓
part2/Applications_of_rLRA/Single_pixel_spectral_imaging	2026-04-27 15:33	force	46.89	✓
part2/Applications_of_rLRA/intro	2026-04-27 15:33	force	2.23	✓
part2/Fast_algorithms_for_rLRA/CMTF	2026-04-27 15:33	force	0.64	✓
part2/Fast_algorithms_for_rLRA/NNParafac2	2026-04-27 15:33	force	14.44	✓
part2/Fast_algorithms_for_rLRA/UnrolledNMF	2026-04-27 15:33	force	5.55	✓
part2/Fast_algorithms_for_rLRA/inertial_BCD	2026-04-27 15:33	force	6.02	✓
part2/Fast_algorithms_for_rLRA/mSOM	2026-04-27 15:33	force	3.5	✓
part2/Fast_algorithms_for_rLRA/proco-als	2026-04-27 15:33	force	6.72	✓
part2/Fast_algorithms_for_rLRA/sparse_nnls	2026-04-27 15:34	force	43.14	✓
part2/Theory_of_rLRA/DLRA	2026-04-27 15:34	force	4.02	✓
part2/Theory_of_rLRA/DL_identifiability	2026-04-27 15:34	force	2.52	✓
part2/Theory_of_rLRA/HRSI_theory	2026-04-27 15:34	force	11.84	✓
part2/Theory_of_rLRA/MixedSparseCoding	2026-04-27 15:35	force	14.41	✓
part2/Theory_of_rLRA/multiple_dictionaries	2026-04-27 15:35	force	3.92	✓
part2/Theory_of_rLRA/onesparseDLRA	2026-04-27 15:35	force	2.73	✓
part3/KarpCoi	2026-04-27 15:35	force	1.63	✓

Part I

Introduction

CHAPTER

ONE

SUMMARY OF HDR CONTENTS

1.1 Some personal context

When I was six years old and starting to learn to play the piano, I had fun guessing the notes my tutor was playing while I was turning my back. This was a pleasant game as I discovered that I could hear, litteraly, the label of the pitch class of individual notes as they were played. I thought everyone else could also hear them and, therefore, transcribe music easily. It was only a few years later, when joining my first music school, that I discovered I was mistaken and that music transcription is actually hard. For everyone, in fact, as even with perfect pitch, transcribing polyphonic instruments requires pattern matching with template harmonies, chords and timbre stored in the listener's memory, and therefore requires extensive training.

As I grew older, I began to enjoy the broader concepts of sound and music processing, the Fourier transform, and their connections to how humans perceive sound, as well as the symbolic and theoretical description of music. It was with great interest that I realized, by the end of my PhD thesis, that many questions in the field of music information retrieval, which encompasses most of these concepts, could be addressed using a concept I had been working on: Low-Rank Approximations (LRA) of matrices and tensors. In particular, the design of regularizations and constraints was key to achieving reasonable performance, and I felt that, with my skill set, I could improve on the existing state of the art in that domain, particularly in automatic music transcription.

It turns out that LRA, which are the main topic of this manuscript, are quite heavily beaten in practice by deep learning approaches when it comes to transcribing piano pieces. This anecdote shows however that LRA are ubiquitous and critical to unsupervised machine learning. While some researchers specialize in one specific application of LRA, I am rather interested in the applied mathematical side of LRA, in particular in modeling and training aspects. I am convinced that LRA-based learning algorithms, in many applications such as automatic music transcription, can still be improved in light of the new results obtained over the last decade, a very modest part of which I am responsible for. It would bring me great joy to see anyone design a state-of-the-art automatic transcription algorithm that relies heavily on LRA, combined for instance with deep learning, for rare instruments with scarce training data. I firmly believe that this objective is within reach in the coming years.

I write this "Habilitation à Diriger des Recherches" (HDR) manuscript with the hope of demonstrating that regularized LRA (rLRA) is a rich topic with many exciting open questions. I also hope that fellow researchers who are willing to read through all this content will learn about some lesser-known results, tricks, and algorithmic details that can help them in their work on rLRA.

1.2 Why regularized low-rank approximations

LRA are workhorse methods in several unsupervised machine learning tasks such as dimensionality reduction [Golub and Van Loan, 1989, Hotelling, 1992, Tucker, 1966] and blind source separation [Harshman, 1970, Lee and Seung, 1999, Paatero, 1997]. The most prominent LRA method in machine learning is probably Principal Component Analysis (PCA), which numerically boils down to computing a *Singular Value Decomposition* (SVD), and is therefore efficiently computed.

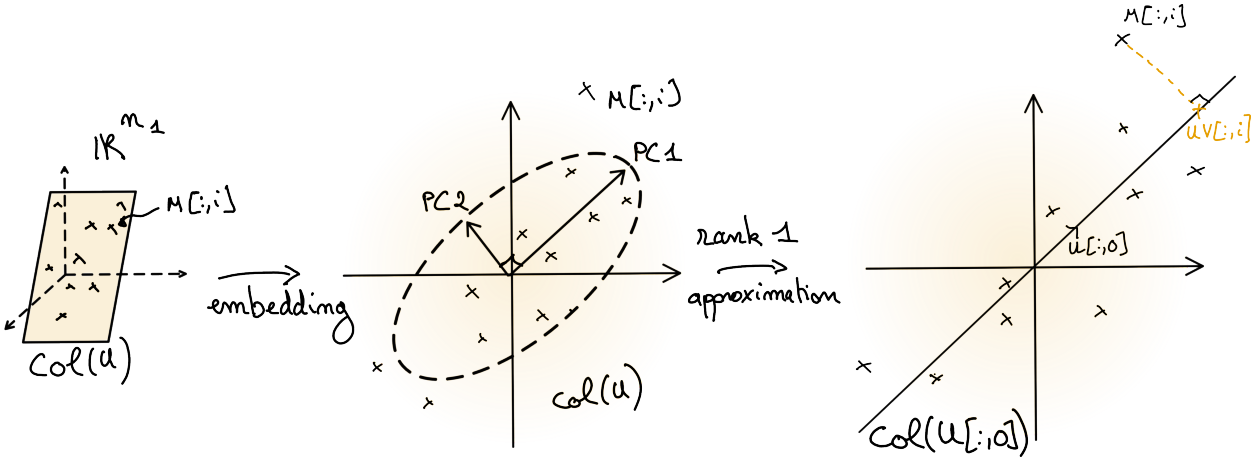


Fig. 1.1: Illustration of PCA. The centered data matrix M lies in a low-dimensional subspace $col(U)$. If the orthogonal matrix U is computed from the data matrix M using SVD, then the columns of U are the principal components (scaled in the figure). An LRA is obtained by projecting the data points on a subset of the principal components; in the right plot, on the first component.

Formally, for matrix data, a real LRA problem is an optimization problem of the form

$$\min_{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{n \times r}} f(Y, UV^T)$$

where integer r is the rank of the approximation $UV^T \approx Y$, with $r \ll m, n$, and f is a cost function, typically $f(Y, X) = \|Y - X\|_F^2$.

Remark

We present the case of matrix factorization here for simplicity, but the same logic applies to higher-order factorizations.

In unsupervised machine learning tasks, users often seek to give a physical meaning to the factor matrices U and V.

In this context, there exists a true pair (U^*, V^*) that the user seeks to recover, and LRA can be seen as an inverse problem. A necessary condition to correctly interpret a reconstructed pair is the uniqueness of the LRA solution. However, LRA, as defined above, is never unique, since any product UV can also be written $UPP^{-1}V$ for an invertible matrix P of size $r \times r$.

A widely used approach in signal processing and machine learning to restrict the set of solutions to an inverse problem is to incorporate prior information on the parameters in the form of constraints or regularizations. rLRA can be formulated as

$$\min_{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{r \times n}} f(Y, UV) + g_U(U) + g_V(V)$$

where g_U and g_V are regularizations promoting certain properties in solutions, such as smoothness, sparsity, or nonnegativity. Depending on the choice of regularizations, the solutions may be unique.

Note: Regularizations typically apply to each factor U and V independently. Indeed, the main identifiability issue in LRA is the rotation ambiguity $UV^T = UP\,(VP)^T$ for an orthogonal matrix P. A regularization $g(UV^T)$ would not discriminate between two solutions and therefore is not enough to ensure the uniqueness of rLRA solutions.

PCA is a constrained LRA model: factors are imposed to be orthogonal matrices. This allows us to obtain a model with essentially unique factors (under the mild condition that singular values must be distinct [Golub and Van Loan, 1989]), but orthogonality is often not a physically meaningful constraint. In the context of inverse problems, this means that the estimated matrices U and V may be poor approximations of the ground truth matrices U^* and V^*. On the other hand, Nonnegative Matrix Factorization (NMF), obtained by setting g_U and g_V to characteristic functions of the nonnegative orthant, can be unique [Gillis, 2020] while using more realistic assumptions. In many applications, elementwise nonnegativity is a natural assumption, which makes NMF particularly suited as a source separation/pattern mining model. A typical example of NMF usage is in spectral unmixing for remote sensing, as illustrated in Fig. 1.2, see also *Applications of rLRA: summary*.

Remark

Essential uniqueness is the uniqueness up to permutations and scaling ambiguities inherent to LRA models, see *Low-rank matrix and tensor models* for more details.

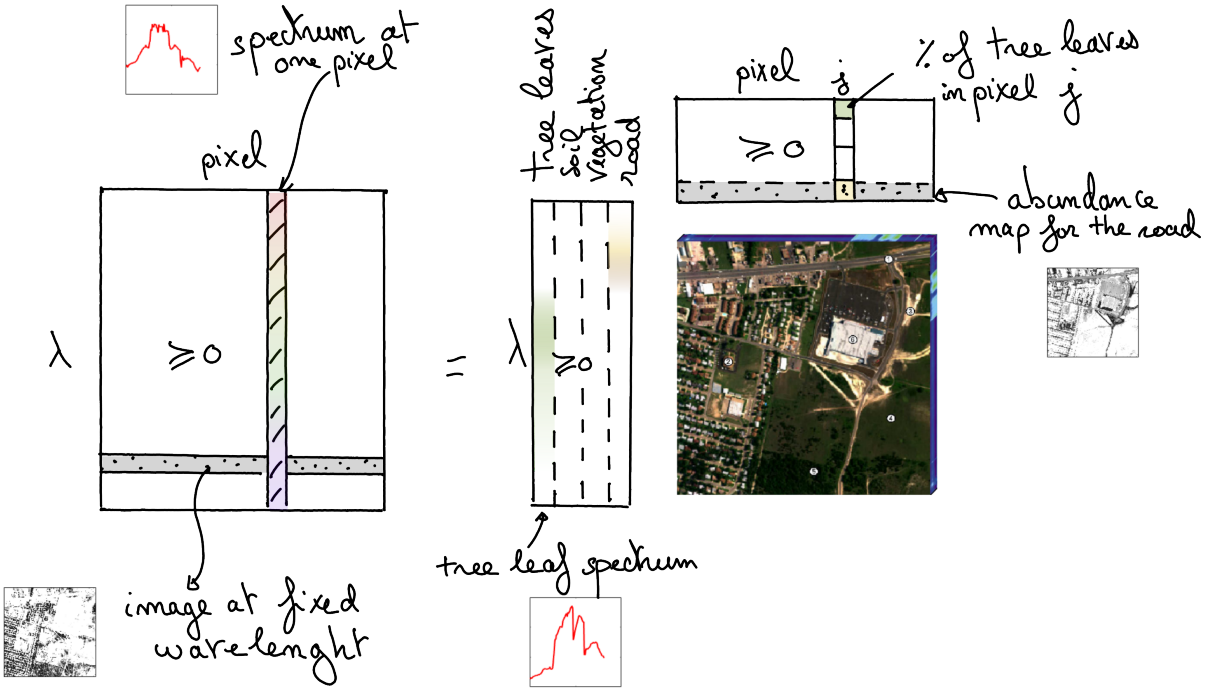


Fig. 1.2: NMF performs spectral unmixing on hyperspectral images, identifying essential spectra that are present in the scene, as well as their proportion in each pixel stored in abundance maps. The images are vectorized in the data matrix and the abundance maps.

An extension of matrix LRA is tensor LRA, which applies to input data with more than two dimensions. Tensor LRA has several interesting properties, including, in the case of the Canonical Polyadic Decomposition (CPD), identifiability without the need to regularize [Kruskal, 1977]. In practice, however, regularization, and in particular nonnegativity, is often useful to improve the interpretability of the estimated parameters from tensor LRA. Section *Low-rank matrix and tensor models* describes known results on matrix and tensor LRA in more detail.

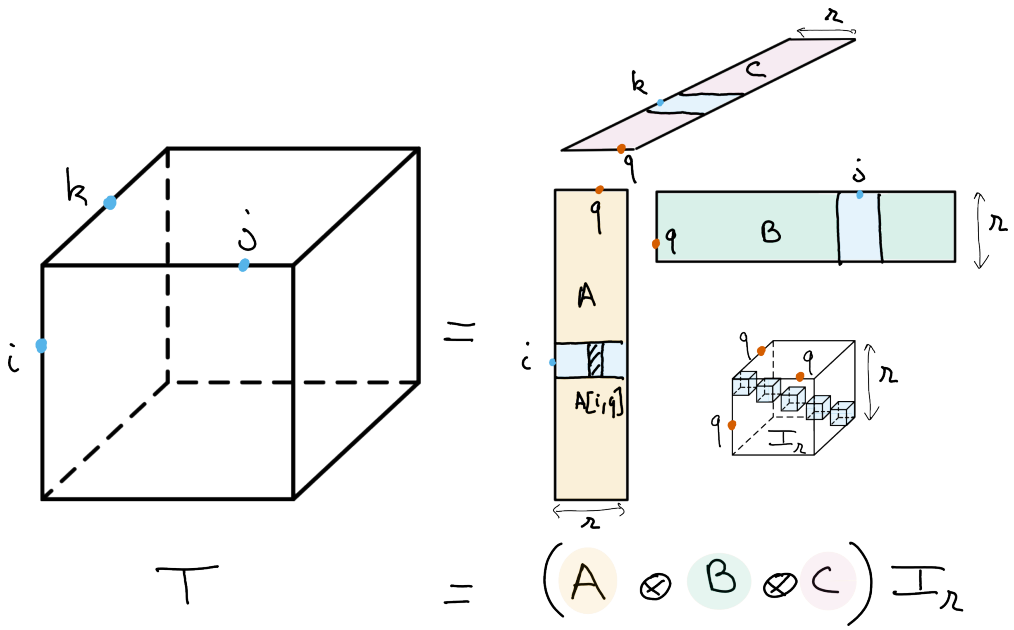


Fig. 1.3: CPD of rank r of a data tensor T. The CPD can be viewed as a multilinear diagonalization technique, with bases A, B and C respectively spanning the columns, rows and fibers of the tensor.

Once an LRA model has been chosen for a particular application, as in most machine learning problems, the model parameters need to be estimated so that the model fits the data. Optimization problems encountered in rLRA problems are often continuous but nonsmooth, for instance, due to the nonnegativity constraint, and nonconvex due to the presence of the product of variables in LRA. Convexity, however, often holds when the cost function is considered only with respect to one of the parameter matrices. Consider the Nonnegative Least Squares (NNLS) problem with Frobenius data fitting

$$\min_{U \in \mathbb{R}_+^{m \times r}} \|Y - UV^T\|_F^2,$$

which is essentially NMF where the matrix V is fixed. The cost with respect to U is convex, and tools from convex numerical optimization can be leveraged to compute NMF. Therefore, for the applied mathematician working on rLRA, it is important to be familiar with both nonsmooth, constrained and convex optimization and multi-block optimization often solved with Alternating Optimization (AO) strategies. The specific case of nonnegatively-constrained regressions is of critical importance to this manuscript and is covered in depth in *Nonnegative regressions: NNLS and NNKL*, while a summary of known results regarding AO strategies is provided in *Alternating optimization*.

There is a significant body of literature on rLRA with a myriad of applications, for instance in chemometrics [Bro, 1996], neuroscience [Cichocki, 2011], statistical inference [Anandkumar *et al.*, 2015], spectral unmixing for remote sensing [Bioucas-Dias *et al.*, 2012] or microscopy imaging [Hariga *et al.*, 2024], music information retrieval [Smaragdis and Brown, 2003], telecommunications [Sidiropoulos *et al.*, 2000], psychometry [Harshman, 1972] and more. There is no dedicated section to reviewing all these applications in this manuscript. Datasets from hyperspectral imaging and music information retrieval are used to illustrate the various contributions, and my contributions to these applications are detailed in *a separate chapter*.

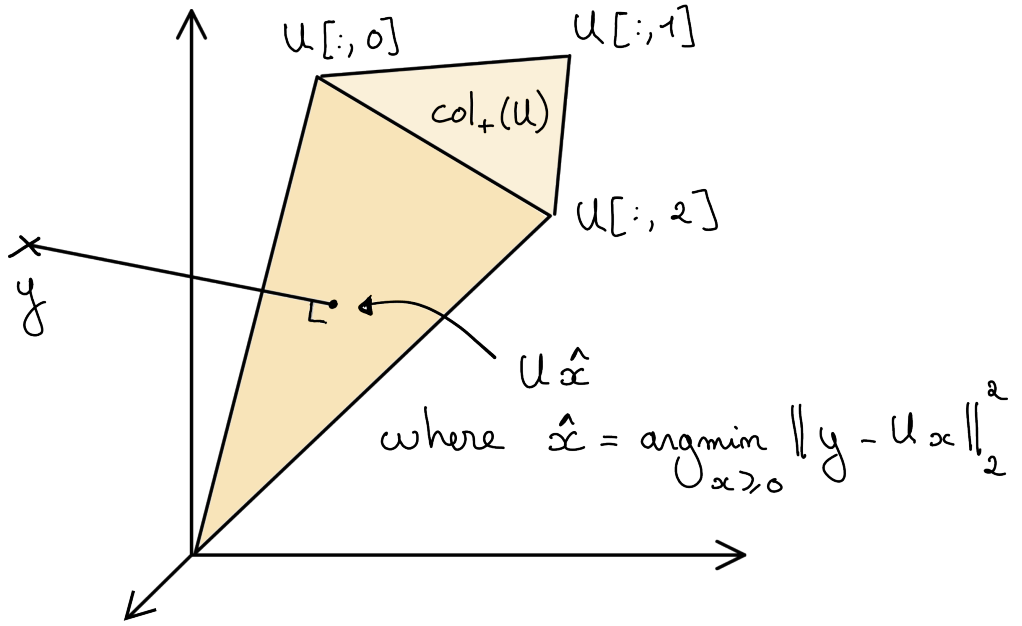


Fig. 1.4: Graphical representation of the NNLS problem. NNLS consists in the orthogonal projection of the data vector y on the cone $\text{col}_+(U)$ spanned by the nonnegative linear combinations of the columns of matrix U.

1.3 Contributions to regularized low-rank approximations

Since the start of my PhD thesis in 2013, I have focused my research on rLRA models as tools for extracting meaningful information out of matrices and tensors, using regularizations to enhance interpretability. These regularizations often include nonnegativity, which has become my specialty. My work covers a diverse range of topics, from understanding the properties of solutions to rLRA to proposing new applications for these models. A significant portion of my work is dedicated to algorithm design for efficiently computing solutions to rLRA across various setups.

1.3.1 Challenges in rLRA

Despite the large body of existing work on rLRA, its use by practitioners is often quite difficult. Indeed, there are many exciting theoretical and practical issues that remain to be dealt with:

- **Optimization problems** stemming from fitting rLRA models are multi-block, nonconvex, and often nonsmooth. Recent advances in numerical optimization, including acceleration, majorization-minimization, generalized projection operators, and AO, can all be leveraged to improve upon the state-of-the-art. It is also unclear, in practice, which optimization algorithm to use depending on the context.
- **Side information** is often available alongside the matrix or tensor data, which transforms the unsupervised rLRA learning problem into a semi-supervised or fully supervised problem. This information can take diverse forms: a known library of templates for the unknown sources, a downstream task with available training data, deep priors for the unknown sources such as provided by deep denoisers, or, among others, sparsity in well-known bases. Accounting for this side information in rLRA is often not straightforward, both from a modeling and from a training perspective.
- **Multimodality** has emerged as an important topic in source separation, where the same phenomenon is observed through different sensing devices. A typical example would be acquiring brain activity through EEG and fMRI jointly. The joint processing of the acquired dataset can be performed by several coupled rLRA. The research questions on joint rLRA typically revolve around the coupling model, and the design of flexible algorithms to solve

a wide range of multimodal source separation problems, in particular when the datasets have different dynamics, noise levels, and even different fields (such as floats vs count vs categorical data).

- The rLRA models themselves still have elusive properties. An important example is nonnegative Tucker decomposition, for which **identifiability** is unclear in many cases, despite recent advances on the topic [Saha *et al.*, 2025]. Generally, a user needs guarantees about the nature of rLRA solutions, and such theoretical guarantees are challenging to obtain, especially when the hypotheses must be verifiable in practice.
- The **scientific software ecosystem** is a dimension often overlooked but critical in practice. Available software for tensor decomposition is largely bloated; see the survey by Psarras *et. al.* from 2022, which lists over 70 tensor packages scattered across the internet [Psarras *et al.*, 2022]. However, software packages dedicated to, or capable of handling at least partially, rLRA are scarce. For tensor decompositions, many packages focus on the decomposition itself but do not use efficient large-scale contractions on CPUs or GPUs that are crucial for up-scaling [G. A. Smith and Gray, 2018, Li *et al.*, 2015, Smith and Karypis, 2015]. Another critical issue is the actual implementation of rLRA algorithms, that requires non-trivial caching of expensive operations and speed-memory trade-offs that are research topics by themselves [Kaya and Uçar, 2016].

1.3.2 Improving on theory, algorithms, and applications

My work has been dedicated to proposing (partial) solutions to these issues. I have grouped my contributions into three parts: theoretical contributions, application-oriented contributions, and algorithmic-focused contributions. However, in most of these works, all three aspects (theory, algorithms, applications) are intertwined, so this is not a strict segmentation.

The three main parts of this manuscript are

- ***Theory of rLRA***, where I summarize my contributions to the analysis of several sparse models: *dictionary-based LRA*, *sparse nonnegative least squares*. I also study *the impact of scaling ambiguity on rLRA solutions*, which can induce unexpected group-sparsity at the component level.
- ***Algorithms for rLRA***. My contributions concern *flexible algorithms for multimodal rLRA*, *heuristic inertial acceleration for AO algorithms*, and more recently, *the design of tight surrogates* for nonnegative optimization problems, including NMF in non-Euclidean settings. I have also worked on data-driven regularization for LRA, in particular on *unrolling multiplicative updates* for NMF. Finally, I revis a work from my PhD thesis on *projected least squares to compute nonnegative LRA*.
- ***Applications of rLRA***. My expertise in terms of applications of rLRA is geared towards both *spectral imaging* and music information retrieval, in particular *automatic transcription*. My interest in spectral imaging stems from the fact that LRA is intimately linked to additive mixtures, which accurately describe mixtures in spectral acquisition techniques such as fluorescence spectroscopy or hyperspectral remote sensing. Being a semi-professional pianist, composer and singer, I have a personal interest in music that drives my research in music information retrieval. Other applications, mostly related to biomedical imaging, are more punctual collaborations and are described *below*.

This manuscript dives quite deep into the details of the selected contributions. A general overview of my work can still be obtained by reading only the introductions of *Theory of rLRA*, *Algorithms for rLRA*, and *Applications of rLRA*, which contextualize and summarize the works detailed in each part.

There is no section in this manuscript dedicated to software development. Instead, simple implementations of the methods under scrutiny are provided inline, imported from tensorly, which I have co-developed since 2019, or imported from a local set of methods developed specifically for this manuscript. Software development is an important part of scientific contributions in rLRA and, more generally, in applied mathematics. In my opinion, it is important to show the exact code being run for experiments and illustrations so that readers can be more deeply convinced of the results' exactitude.

1.3.3 Other works

Several of my works are not discussed in details in this manuscript. They are often targeted toward specific applications or require dedicated mathematical tools that I want to avoid introducing here. Below is a short paragraph for each of these works.

Music segmentation

One of the most impactful work that I chose not to detail is related to the PhD thesis of Axel Marmoret [Marmoret *et al.*, 2020, Marmoret, 2022, Marmoret *et al.*, 2022]. Low-dimensional models such as Tucker decomposition and auto-encoders are used to compress information from a recording of a full song [Smith and Goto, 2018]. This compression highlights the similarities and dissimilarities between bars in the song, which then helps a dedicated dynamic program solver to perform an automatic segmentation of the song. Despite the method being unsupervised (a supervised beat-tracking algorithm is still used to cut the song into bars), the results are very encouraging if the parameters of the compressed models can be chosen optimally. This choice is difficult in practice, which limits the method's performance.

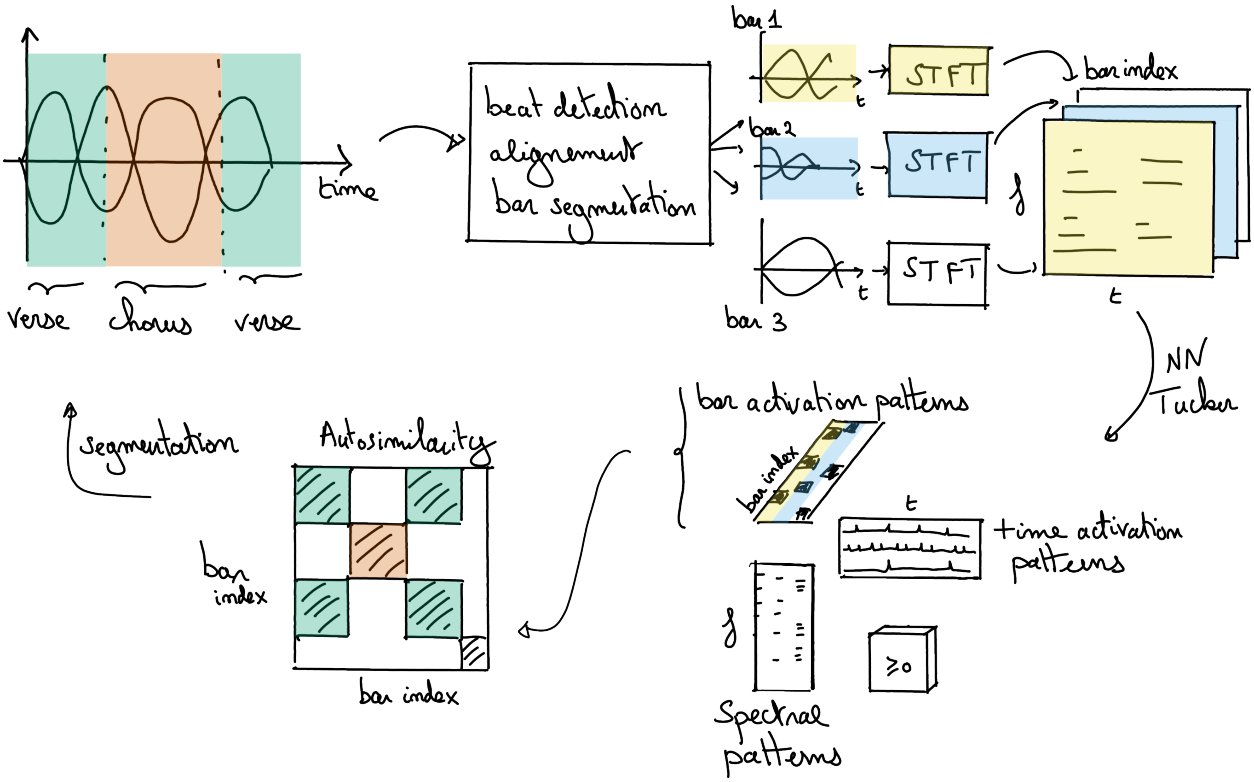


Fig. 1.5: Nonnegative Tucker decomposition can be used to extract meaningful patterns from a Time-Frequency-Bar tensor, obtained from an audio recording of a song by slicing the audio barwise and applying individual short-time Fourier transforms. The bar activation patterns are then used as low-dimensional features to perform automatic segmentation on an auto-similarity matrix, revealing the hidden structure of the song.

This work was rather visible in the music information retrieval community when released. It has launched the academic career of Axel Marmoret, who is today working, among other areas, on LRA in conjunction with deep learning and their applications to music information retrieval. His PhD manuscript describes our contributions in detail.

1.3.4 Birkhoff-von Neumann decomposition of stochastic matrices

The Birkhoff-von Neumann (BvN) decomposition of a stochastic matrix writes a bistochastic matrix M as a minimal number of permutation matrices P_i,

$$\text{Find the smallest } k \in \mathbb{N} \text{ such that } M = \sum_{i=1}^{k} \alpha_i P_i,$$

where α_i are nonnegative coefficients; both coefficients and permutation matrices have to be estimated from the input matrix. This long-standing NP-hard problem can be tackled by a variety of existing heuristics. In collaboration with Bora Uçar and Damien Lesens, we proposed viewing BvN as a sparse coding problem and adapting existing sparse coding solvers, such as Orthogonal Matching Pursuit (OMP), to compute solutions [Lesens *et al.*, 2024]. The contribution is to link the selection of the best atom (here, the best permutation matrix) to several graph matching algorithms, making the selection both fast and effective. Our method is competitive with the state-of-the-art. In a follow-up work driven by Damien Lesens, we extended BvN algorithms to the symmetric case [Lesens *et al.*, 2026]. We are the first to propose an implementable algorithm to solve symmetric BvN.

These works fall slightly outside the realm of rLRA. My role for the second contribution on symmetric BvN, in particular, was rather minor. Therefore, I chose not to discuss this work further. Damien Lesens is now pursuing a PhD co-supervised with Bora Uçar on optimization algorithms for nonnegative LRA, see the *Perspectives*.

1.3.5 Sparse separable NMF

Separable NMF is a regularized variant of NMF with nice identifiability and algorithmic properties, see *the dedicated section* in *Low-rank matrix and tensor models*. In a joint work with Nicolas Nadisic, Arnaud Vandaele, and Nicolas Gillis, we studied a slightly more general model called sparse separable NMF, which assumes that the data points are sparse convex combinations of a few data vectors [Nadisic *et al.*, 2021]. An algorithm is proposed ressembling OMP that identifies the components from the data directly, and compute sparse regressions for each data point using a previously proposed *sparse NNLS algorithm*. We show, however, that the problem is in general NP-hard, unlike separable NMF.

1.3.6 Optical Biopsy NMF

A contribution closely related to *Single-pixel hyperspectral image reconstruction and unmixing* performs segmentation of cancerous cells from videos of the brain during surgery using separable NMF [Caredda *et al.*, 2023], in collaboration with Bruno Montcel, Charly Carrera, and others. For this work, my contribution is to propose a robust variant of the well-known direct algorithm for rank-two exact separable NMF in the Frobenius norm. The components estimated from the rank-two NMF are then used to estimate the spatial oxygenated blood flows, which help locate the glioma.

1.3.7 Lever arm tensor decomposition

In a collaboration with Raphaël Dumas and Joris Claude, we studied a set of lever arm measurements from leg muscles acquired on patients with knee injuries during walking [Claude *et al.*, 2024]. This study revealed synergies among leg muscles. These muscle activations can be clustered to summarize the complex action of the human body during walking into a few descriptors. My contribution to this work was to help design the constrained tensor decomposition model. The data tensor is quite difficult to decompose, and running unconstrained CP decomposition yields inconsistent results across runs; using sparsity and nonnegativity priors helped reduce this variability and improve the interpretability of results. Joris Claude is now pursuing a PhD in biomechanical systems.

1.4 Perspectives

1.4.1 Research objectives

There have been at least two sources of frustration in my research work so far. First, I have always been very interested in the applied mathematics aspects of signal processing, and many of my current contributions are based on heuristics or lack a clean theoretical motivation. Second, numerical optimization is the bread and butter of rLRA, but there is a large body of literature on numerical optimization that I am still unfamiliar with. I have the impression that contributions to numerical optimization, with the practical challenges of rLRA and its applications in mind, are within reach and can be impactful. Therefore, in the coming years, my main objective is to dedicate more research time to understanding the intricacies of numerical optimization and to propose theoretically motivated algorithms both for convex and nonconvex problems.

An optimization problem that I believe is particularly interesting is the so-called NNKL problem described in *Nonnegative regressions: NNLS and NNKL*, namely, solving linear regressions under the Kullback-Leibler divergence (KL-divergence) loss. For an input nonnegative data vector $y \in \mathbb{R}^m_+$ and a linear observation matrix $A \in \mathbb{R}^{m \times n}_+$, NNKL can be formulated as

$$\operatorname*{argmin}_{x \geq 0} \mathcal{D}_{\text{KL}}\left(y, Wx\right),$$

where $\mathcal{D}_{\text{KL}}\left(y, z\right) = \sum_i y[i] \log(\frac{y[i]}{z[i]}) + z[i] - y[i]$ is the KL-divergence. This optimization problem is quite challenging for at least two reasons:

- The cost function is not Lipschitz-smooth at zero. Lipschitz-smoothness is a key property of cost functions leveraged in most convergence proofs of first-order methods. In practice, choosing a step-size for first-order methods can be challenging without Lipschitz-smoothness.
- When $y[i]$ is significantly smaller than $z[i]$, the loss is almost linear (the logarithmic term vanishes). This means that the cost function is not strongly convex, another important property that guarantees the practical speed of first-order methods.

```
# Showing plots for 1d Kullback Leibler with slider for data position
x = np.linspace(1e-5, 10, 300)

def kl_divergence(p, q):
    """Calculate the scaled KL divergence between two positive vectors."""
    return np.sum(p * np.log(p / q) + q - p)

y = np.zeros_like(x)
p = 2.5  # Initial data
for i, q in enumerate(x):
    y[i] = kl_divergence(p, q)
```

The optimization community has dedicated significant effort to proposing algorithms to solve problems such as NNKL. *Multiplicative Updates* (MU) is such an algorithm that works well in practice. However, few methods can solve constrained variants of NNKL. In the current state of signal processing, where data-driven methods such as plug-and-play and unrolling algorithms yield significant performance gains, there is a need for algorithms that solve such nonsmooth, non-Lipschitz-smooth, nonconvex problems. I also plan to work on improving the state-of-the-art for solving NNKL and related problems, as I believe that within the frameworks of majorization-minimization and second-order approximation, significant gains can be made. We have already started to propose *better approximations of the cost*, but more work is required in this direction to find faster algorithms, robust to data and parameter sparsity, that can scale to large datasets.

Finally, there is a dire need for a community-oriented optimization benchmark in rLRA. As mentioned above, the number of software libraries for performing LRA is extremely large; however, to this day, as far as I am concerned, it remains

unclear which optimization method should be used in which context, and which implementation is the best on the market. I will start making efforts towards such benchmarking tools, starting with NMF. The benchmark structure has already been completed using the benchopt framework, in collaboration with Cassio Fraga-Dantas, and we are working with several authors, in particular Damien Lesens, on filling up the benchmark with algorithms and datasets.

The perspectives of my research on NNKL, and more generally Poisson-distributed problems, are further detailed in *Numerical optimization for KL-based regularized inverse problems*.

A related, important line of work for the future focuses on Single-Pixel Imaging (SPI). Perspectives on this topic include

- A joint reconstruction and unmixing of single-pixel images with deep priors (such as plug-and-play and unrolling). This is the topic of the end of Serena Hariga's PhD thesis.
- Better algorithms for Poisson noise problems, as discussed above.
- A deeper understanding of Poisson-Gaussian equivalences, and estimation under Poisson-Gaussian noise. Anna Jezierska and co-authors have already worked on this problem in the context of inverse problems [Chouzenoux *et al.*, 2015], but I believe there could be additional discoveries to be made. In particular, known results about Poisson-Gaussian equivalences in high-count settings, although standard in the optics and statistics community [Curtis, 1975, Seifert *et al.*, 2023], are not easily summarized or put in use in a numerical optimization context. This work could be performed in collaboration with Valentin Debarnot, recently recruited to CREATIS.

These topics are also discussed in *Numerical optimization for KL-based regularized inverse problems*. Other interesting topics regarding SPI are discussed in *Other perspectives*. A first topic of interest is the *design of the acquisition operator*, putting the theory of compressive sensing to the test. A second topic is the derivation of *separable NMF heuristics for inverse problems*. The low-rank data matrix is not observed directly, therefore one may not simply pick columns from it as traditionally done in separable NMF. Finally, a last topic of interest is Borgen plots [Neymeyr and Sawall, 2018], and more generally, algorithmic tools for studying uniqueness of solutions in inverse problems. Borgen plots show all possible solutions of NMF graphically for rank three, but their generalization to higher dimensions and ranks, as well as to other inverse problems [Munier *et al.*, 2025, Sawall *et al.*, 2022], remains an open problem.

1.4.2 About the evolution of research on rLRA

In the long run, with the rise of deep learning and the success of end-to-end black-box modeling coupled with the increasing ease of collecting large training datasets or generating synthetic ones, one may fear that unsupervised machine learning techniques, and in particular LRA, could become obsolete. My honest answer to this observation is that there is indeed a risk that LRA models, and more generally physics-inspired models, may not be required to achieve state-of-the-art results across a wide range of applications. Looking at the current state of things, however, there are at least two reasons to hope for a future in signal processing and AI in which LRA still plays an important role.

1. Physics-driven AI is currently a promising research direction to "open the black box". By using physical models, for instance, as forward operators, the performance of deep learning systems may not improve, but can be more robust to distribution shifts [Lorente Mur *et al.*, 2022]. Unrolling optimization algorithms provides a means to guide the design of neural networks motivated by traditional optimization algorithms. Plug-and-play methods enable the use of black-box models within physics-driven iterative algorithms. Although the resulting algorithms can still work in mysterious ways.
2. LRA is a fundamental tool in numerical linear algebra, signal processing, and machine learning. Many researchers are already working on using LRA to improve the design or training of deep learning architectures. A famous example of this is LORA [Hu *et al.*, 2021], but other works have considered more involved usage of LRA models in deep learning [Borsoi *et al.*, 2025, Kossaifi *et al.*, 2020]. Another more personal view on this matter is that rLRA can always be used as a trustworthy baseline in the future, even if more advanced models may significantly outperform it. In applications such as *automatic transcription*, it can be refined in multiple ways so that its performance is close to that of deep learning [Marmoret *et al.*, 2020]. If the current generation can produce efficient and flexible software for rLRA, it is reasonable to assume that, in applications where rLRA performs well, it will be used as an unsupervised baseline for the foreseeable future. The manuscript and the codes furnished along the way are, hopefully, one small step towards this goal.

CHAPTER

TWO

ACKNOWLEDGEMENTS

Part II

About the manuscript

CHAPTER

THREE

HOW TO READ THIS MANUSCRIPT

Research in signal processing, applied mathematics and machine learning requires close interactions between numerical experiments and theoretical observations and results. In this manuscript, I did not want to hide the software/experiment development aspect of my research, and therefore chose to include, in the main body, large sections of commented and executable python code, producing most of the figures and experimental results upon the manuscript compilation. This has several advantages:

- The code can be tinkered with by either downloading the notebooks in .ipynb format (download button on top of each page), or running the code directly inside the navigator. This can be done by clicking on the rocket icon (visible in pages with executable code) and choosing "live code". A binder container is used with the correct package setup; the code from the cells are provided to a python interpreter in this container, and the results are returned live inside the browser. For instance, you can run the code cell below and change the numerical values of the variables.

```
a = 5  # you can edit this !!
b = 7  # you can edit this !!
print(a+b)
```

```
12
```

- The numerical experiments and plots reported in the manuscript are obtained upon the book compilation and computed on github servers. Therefore, I cannot tinker with the results as a post-processing, and the results are ensured to be reproducible. The code of the manuscript is fully available on github. This kind of transparency should be the norm in our communities to avoid practical imprecisions and mistakes in publications. This HDR manuscript was the perfect opportunity to promote my vision of open and reproducible research.

The main distribution support for this manuscript is the online version, however the book is also available as a .pdf document upon request. On the online version, it is possible to leave comments directly from the browser. Click on the arrow in the top-right corner. This expands the "hypothesis.is" panel, where you can create an account, log in, and leave comments by hovering text. To leave comments in a non-public group (preferred), follow this link. This should allow for constant improvement of this manuscript.

Practically, this manuscript can be read in any order. The *introduction* explain general concepts and challenges in regularized LRA. The book is then cut in three main parts, namely

- A technical introduction on *LRA*, *nonnegative problems in numerical optimization* and *alternating optimization algorithms*.
- A rather detailled collection of notebooks summarizing a subset of my contributions in the *theory*, *algorithms* and *applications* of LRA. Each of these themes have short summaries for a quick overview.
- Research perpectives on *Poisson noise inverse problems and numerical optimization* and *other research questions* related to LRA.

A *table of all the acronyms* is also provided to ease nonsequential reading.

CHAPTER

FOUR

TERMINOLOGY AND ACRONYMS REFERENCE

This document lists all acronyms, technical terms, and words with ambiguous spelling found in the manuscript.

4.1 Acronyms

- ADMM: Alternating Descent Method of MultipliersA
- ALS: Alternating Least Squares
- AMT: Automatic Music Transcription
- AMU: Alternating Multiplicative Updates
- ANLS: Alternating Nonnegative Least Squares
- AO: Alternating Optimization
- APGD: Alternating Projected Gradient Descent
- AS: Active-Set
- BCD: Block-Coordinate Descent
- BSUM: Block SUM
- cDL: complete Dictionary Learning
- CPD: Canonical Polyadic DecompositionA
- CMTF: Coupled Matrix and Tensor Factorization
- DL: Dictionary Learning
- DLRA: Dictionary-based Low-Rank Approximation
- EM: Expectation-Maximization
- HALS: Hierarchical Alternating Least Squares
- HDR: Habilitation à Diriger des Recherches
- HER: Heuristic Extrapolation with Restart
- HOMP: Hierarchical Orthogonal Matching Pursuit
- HRSI: Homogeneous Regularized Scale-Invariant problem
- GCMS: Gas Chromatography Mass Spectrometry
- KL-divergence: Kullback Leibler-divergence
- LCMS: Liquid Chromatography Mass Spectrometry

- LRA: Low-Rank Approximation(s)
- MAP: Maximum *A Posteriori*
- mSOM: median Second-Order Majorant
- MC-ALS: Maximum Correlation ALS
- ML-EM: Maximum Likelihood-Expectation-Maximization
- MM: Majorization Minimization
- MMV: Multiple Measurement VectorA
- MU: Multiplicative Updates
- MSC: Mixed Sparse Coding
- nCPD: Nonnegative CPD
- NMF: Nonnegative Matrix Factorization
- NNKL: Nonnegative Kullback-Leibler regression
- NNLS: Nonnegative Least Squares
- OMP: Orthogonal Matching Pursuit
- PCA: Principal Component Analysis
- pro-ALS: projected Alternating Least Squares
- rLRA: Regularized Low-Rank Approximation(s)
- RMSE: Root Mean Square Error
- SNPA: Successive Nonnegative Projection Algorithm
- SOM: Second-Order Majorant
- SPI: Single-Pixel Imaging
- SUM: Successive Upper-bound Minimization
- SVD: Singular Value Decomposition

CHAPTER

FIVE

NOTATION

5.1 General notation

- Matrices are typically represented by uppercase letters, such as M, while vectors and scalars are denoted by lowercase letters, like x.
- Scalar values are often indicated using lowercase Greek λ.
- The transpose of a matrix or vector is indicated with a superscript M^T.
- Slicing of tensors is performed as in python. For instance, the entry at row i and column j of a matrix M is expressed h as $M[i, j]$. To remove the jth column of a matrix M, we write $M[:, -j]$. To consider all columns and rows up to indices i and j, we write $M[: i, : j]$.
- Notation $\mathbb{R}_+$ denotes the set of nonnegative real numbers, and constraints like $x \geq 0$ indicate elementwise nonnegativity.
- The Frobenius norm is denoted as $\|M\|_F$.
- The support of a vector x, or the set of indices of its nonzero elements, is denoted by $S(x)$, or simply S when clear from context.
- The number of elements of a discrete set $\mathcal{K}$ is denoted by $\#\mathcal{K}$.
- The Kronecker product is symbolized by $\otimes_K$ to avoid confusion with the generic tensor product $\otimes$.
- The gradient of a function f is written as ∇f, and its Hessian as $\nabla^2 f$.
- The (left) Moore-Penrose pseudo-inverse of a matrix M is represented by $M^\dagger$.
- The Kullback-Leibler divergence between two nonnegative reals y and z is denoted as $\mathcal{D}_{\text{KL}}(y, z)$.
- The spark of a matrix M, or the smallest number of linearly dependent columns in M, is written as spark(W).
- The column space, or span, of a matrix U is the set of any vector that can be written as Ux and is denoted as col(U). The positive span of matrix U, obtained when x is elementwise nonnegative, is denoted by col$_+(U)$.
- The set of linear maps acting on vectors from a subspace E is denoted as $\mathcal{L}(E)$.
- The matrix of vector full of ones is written either 1_n where n is its dimension, or sometimes simply 1.

5.2 Tensor notation

The unfolding of a tensor T along mode i, denoted $T_{[i]}$, is obtained by row-first vectorization of all modes but i. The Khatri-Rao product of two matrices A and B with the same number r of columns is the columnwise Kronecker product,

$$A \odot B = [A[:, 1] \otimes_K B[:, 1], \ldots, A[:, r] \otimes_K B[:, r]] .$$

The multiway product can be defined for three matrices A, B, C multiplied on modes one, two and three with a tensor G as

$$(A \times_1 B \times_2 C \times_3 G)[i, j, k] = \sum_{r_1, r_2, r_3} A[i, r_1] B[j, r_2] C[k, r_3] G[r_1, r_2, r_3].$$

It can be implemented with tensor contractions, or with matrix-matrix products with unfoldings and reshapes involved.

The *CP decomposition* is represented using the Kruskal notation $[\![A, B, C]\!]$.

Part III

Introduction to rLRA

CHAPTER

SIX

LOW-RANK MATRIX AND TENSOR MODELS

Low-rank models are important tools in both unsupervised and supervised machine learning, signal processing, and more generally in applied mathematics. They are also the main mathematical object studied in this manuscript. This section briefly introduces matrix, and tensor, Low-Rank Approximations (LRA). A solid background in linear algebra is required. Some of the material presented in the following sections has been covered in more detail in the book chapter I wrote with Pierre Comon and Rasmus Bro [Cohen *et al.*, 2023]. The book of Golub and Van Loan provides a complete description of matrix low-rank models [Golub and Van Loan, 1989]. The recent book of Grey Ballard and Tammy Kolda provides a deeper introduction to tensor decompositions, with a focus on computational aspects [Ballard and Kolda, 2025].

6.1 Matrix low-rank factorizations and approximations

There are at least two important ways in which a low-rank factorization of an n_1 by n_2 real-valued matrix M can be understood.

As a linear dimensionality reduction technique

Consider that matrix M is a collection of n_2 vectors in $\mathbb{R}^{n_1}$. A low-rank model describes the information contained in matrix M if and only if these n_2 vectors belong to a smaller subspace of dimension $r \leq n_1$. A low-rank model is therefore a linear dimensionality reduction technique. Dimensionality reduction is at the heart of both signal processing and machine learning, as it offers a compact representation of the input matrix that can be used to either compress the data or to mine information in an unsupervised manner by identifying an important set of variables.

Mathematically, a low-rank factorization of matrix M may be written as

$$\exists U \in \mathbb{R}^{n_1 \times r},\ V \in \mathbb{R}^{n_2 \times r} \text{ such that } \forall j \leq n_2,\ M[:, j] = UV^T[:, j].$$

Matrix U encodes a basis of the low-dimensional subspace spanning the columns of matrix M, and $V^T[:, j]$ is the vector of coordinates of each column $M[:, j]$ in this basis. If the subspace dimension r is minimal, that is, there is no smaller subspace containing the n_2 columns of M, then the dimension r is called the rank of matrix M and is denoted rank (M). Notice that if $n_1 \geq n_2$, then one may always reduce the dimension of the data by projecting on the column space spanned by the n_2 columns. Therefore, this linear dimensionality-reduction interpretation is perhaps more meaningful when $n_1 < n_2$. However, formally, the low-rank model reduces the dimension of the data whenever $r \leq \min(n_1, n_2)$.

The computation of matrix U and coefficients V from the data matrix M can be performed using Singular Value Decomposition (SVD). Algorithms for the SVD rely on either QR decomposition or related tools from linear algebra; see the excellent book of Golub and Van Loan [Golub and Van Loan, 1989], and are implemented in the widely distributed LAPACK software [Anderson *et al.*, 1999].

When the data matrix M is centered, the (truncated) SVD returns a particular low-rank factorization (approximation) of the data matrix, known as the Principal Component Analysis (PCA). PCA is a cornerstone of unsupervised machine learning, used not only for dimensionality reduction but also for mining information from high dimensional dataset; see Fig. 6.1 for an illustration, and `low-rank-approximations` for a discussion on approximations.

As a factorization into a sum of rank-one matrices

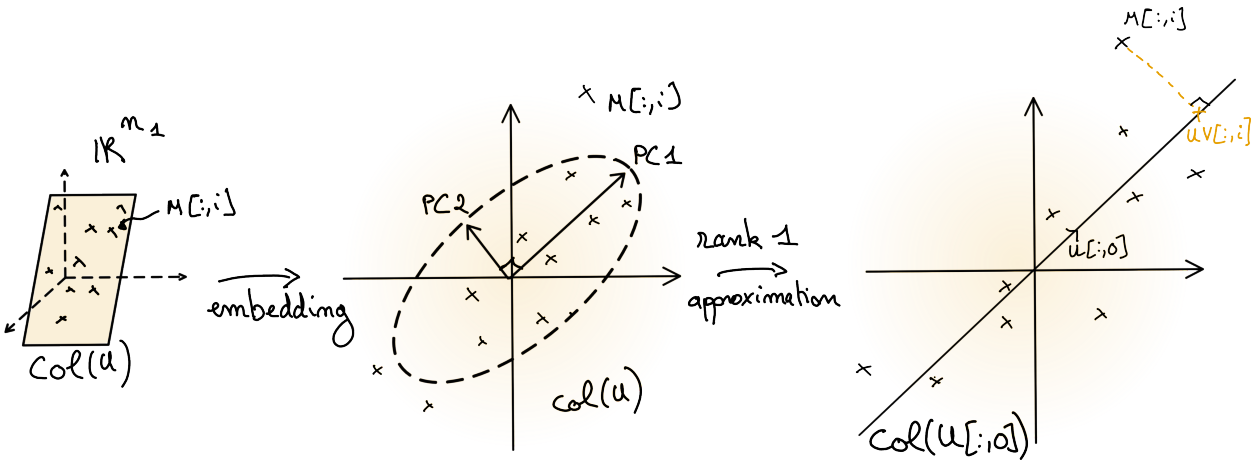


Fig. 6.1: Illustration of Principal Component Analysis. The centered data matrix M lies in a low-dimensional subspace $col(U)$. If the orthogonal matrix U is computed from the data matrix M, then the columns of U are the (unscaled) principal components. A low-rank approximation is obtained by projecting the data points on a subset of the principal components, here on the first component.

Another equivalent description views low-rank models as decompositions of the matrix M into a sum of rank-one matrices. Formally, for a given integer $r \leq \min(n_1, n_2)$,

$$\exists U \in \mathbb{R}^{n_1 \times r},\ V \in \mathbb{R}^{n_2 \times r} \text{ such that } M[i,j] = \sum_{q=1}^{r} U[i,q]V[j,q].$$

The roles of matrices U and V, often called **factor matrices**, are symmetric. Both factor matrices contain representations of rank-one terms $M_q := U[:,q]V^T[q,:]$ such that $M = \sum_{q=1}^{r} M_q$. Note that for fixed dimensions n_1 and n_2, rank-one matrices are simply the set $\mathcal{M}_1$ of outer products of any two nonzero vectors,

$$\mathcal{M}_1(n_1, n_2) = \{ab^T \mid a \in \mathbb{R}^{n_1},\ b \in \mathbb{R}^{n_2},\ a \neq 0,\ b \neq 0\}.$$

The rank of matrix M is the smallest number of terms r such that the decomposition into rank-one terms exists. The two definitions of the rank above are equivalent. If $M = UV^T$, then, equivalently, $M[i,j] = \sum_{q=1}^{r} U[i,q]V[j,q]$.

The decomposition of a matrix as the sum of rank-one matrices can be understood as an atomic decomposition, where the dictionary is the set of rank-one matrices $\mathcal{M}(n_1, n_2)$. The rank of a matrix is nonnegative and bounded above: any matrix M can be exactly described as a sum of rank-one matrices with at most $\min(n_1, n_2)$ terms. Therefore, the rank of a matrix is one possible measure of matrix complexity: rank-one matrices are simple because they can be described with few parameters (a single atom), while full-rank matrices are the most complicated. In this interpretation, low-rank matrices where rank $(M) \ll \min(n_1, n_2)$ are especially interesting because, while they live in the ambient space $\mathbb{R}^{n_1 \times n_2}$, they can be summarized with only r atoms, described by $(n_1 + n_2)r$ parameters.

Remark

An atomic decomposition in this context can be understood simply as combining simple elements, the atoms, from a given set, the dictionary.

The point of view of low-rank models as a decomposition into a minimal sum of rank-one terms establishes a clear link between inverse problems and LRA. It is also powerful when considering *extensions of the notion of matrix rank to tensors*.

Note: There are several other equivalent definitions of the rank for matrices that are widely used. An important definition is the dimension of the row-space or the column-space of a matrix M, both of which are in fact equal in dimension.

6.1.1 Low-rank approximations

In most practical applications of interest to this manuscript, the data matrices are not exactly low-rank. Instead, these data matrices are well approximated by low-rank matrices. A specific LRA is determined by the loss function, often the squared Euclidean distance, the input data matrix, and the rank of the approximation. A rank r approximation of a data matrix M, in the Euclidean distance, can be formulated as an optimization problem

$$\underset{\text{rank}(N)\leq r}{\text{argmin}} \ \|M-N\|_F^2$$

where $\|M-N\|_F^2 = \sum_{i,j}\left(M[i,j]-N[i,j]\right)^2$ is the (squared) Frobenius norm of the difference $M-N$, with N the low-rank approximation of data matrix M.

A different formulation of LRA is obtained if the approximation matrix is parameterized by two factor matrices U and V,

$$\underset{U\in\mathbb{R}^{n_1\times r},\, V\in\mathbb{R}^{n_2\times r}}{\text{argmin}} \ \|M-UV^T\|_F^2,$$

a formulation often coined as the Burer-Montero factorization [Burer and Monteiro, 2003].

Note that the exact low-rank factorization problem may also be written with a similar formalism,

$$\text{Find } U\in\mathbb{R}^{n_1\times r},\ V\in\mathbb{R}^{n_2\times r} \text{ such that } M=UV^T.$$

The Frobenius norm is the most widely used distance metric for LRA, as it yields a closed-form solution. Indeed, let $M=ASB^T$ be the SVD of the matrix M. Then the following result [Eckart and Young, 1936] shows that $U=A$ and $V^T=SB^T$ is optimal in the least squares sense.

Theorem 1 (SVD solves LRA with Frobenius norm (Eckart-Young theorem))

Assume data matrix M admits the singular value decomposition $M=ASB^T$. Then the optimization problem

$$\underset{\text{rank}(N)\leq r}{\text{argmin}} \ \|M-N\|_F^2$$

admits the truncated SVD as a solution, $N^*=A[:,:r]S[:r,:r]B^T[:r,:]$. The solution is unique if the r-th singular value of M has multiplicity one. The residual error is then exactly $\sum_{q=r+1}^{\min(n_1,n_2)} S[q,q]^2$.

Remark

Uniqueness is meant here in the sense that matrix N is the only best rank-r approximation of the data. This is compatible with the lack of identifiability of factor matrices U and V.

The Eckart-Young theorem is a remarkable and incredibly useful result in linear algebra, as LRA in Frobenius norm is one of the few nonconvex problems that admit a closed-form solution that can be computed efficiently. The same result holds (with a different residual) if the squared Frobenius norm is replaced by the spectral norm, but in general, the truncated SVD is not the solution to LRA with arbitrary loss functions. Another loss function often encountered in LRA is the KL-divergence discussed in *NNKL problem definition*.

6.1.2 Why are data matrices often (approximately) low-rank

After many years of research in signal processing and machine learning, I have observed an intriguing yet recurring phenomenon: data matrices encountered in various applications are often approximately low-rank. While there is probably no general explanation for this phenomenon, here are a few (non-orthogonal) reasons why this can happen.

1. "Nice" latent variable models are of low-rank [Udell and Townsend, 2019], meaning that for a large catalogue of generative models, the resulting data matrix can be well approximated (in entry-wise error) by a low-rank matrix.
2. The entries of a rank-one matrix are samples of a separable map in two variables,

$$f(x, y) = f_1(x)f_2(y).$$

   Separable maps are ubiquitous in physics since they allow for simple descriptions of multivariate functions.
3. Low-rank models $M = UV^T$ can be seen as linear source separation models. The rank of the LRA is then the number of underlying sources. Matrix U contains columnwise the templates for the sources, and matrix V contains the mixing coefficients for these sources in the data. Nonlinearities, measurement noise, and missing data corrupt the low-rank data matrix. Many physical processes can be described this way, see examples in audio and hyperspectral image processing in *Applications of rLRA: summary*.

Remark

The first claim however must be mitigated, the class of "nice" latent variable models is more restrictive than hinted in the initial publication by Udell and co-authors, see [Budzinskiy, 2025].

The second and third explanations also imply that in many applications, the factor matrices bear physical meaning: the practitionners ask of LRA for source separation to return good approximations of the underlying true factor matrices U and V. There are two important questions to answer regarding the estimation of the factor matrices:

1. Is the model essentially identifiable, *i.e.*, does equality of two low-rank models $M = U_1V_1^T = U_2V_2^T$ implies equality of the factors up to trivial ambiguities ?
2. Is the estimation of factor matrices robust in the presence of noise?

Note: Trivial ambiguities in LRA are the scaling ambiguity, $ab^T = \frac{1}{\lambda}a\lambda b^T$, and the permutation ambiguity $UV^T = U\Pi\Pi^TV^T$ for any nonzero scalar λ and any permutation matrix Π. These ambiguities arise from representing rank-one matrices as outer products but are not inherent to the atomic decomposition; the same atoms are used by two essentially equivalent low-rank factorizations. Essential uniqueness, meaning uniqueness up to permutations and scalings, is therefore considered.

The answer to the first question is negative: if $U_1V_1^T = U_2V_2^T$ with the same rank for both models, then one may only assume that the two bases describe the same subspace, and are therefore related by an invertible transformation $Q \in \mathbb{R}^{r\times r}$ such that $U_1 = U_2Q$. This poses a fundamental problem for applications of LRA in source separation, as essential identifiability (or at least guarantees on the set of LRA solutions) is required to interpret the output of LRA algorithms as approximations to the ground-truth factor matrices. The rest of this section is therefore devoted to introducing further matrix and tensor decomposition models that enjoy identifiability conditions.

The answer to the second question is well understood in the context of matrix low-rank approximations in the Frobenius norm: the robustness of LRA is directly related to the conditioning number of the data matrix.

Note: The algorithmic development for LRA is only briefly described in this section. Two other sections in the manuscript are dedicated to numerical optimization: *Nonnegative regressions: NNLS and NNKL* and *Alternating optimization*.

6.1.3 Nonnegative Matrix Factorization

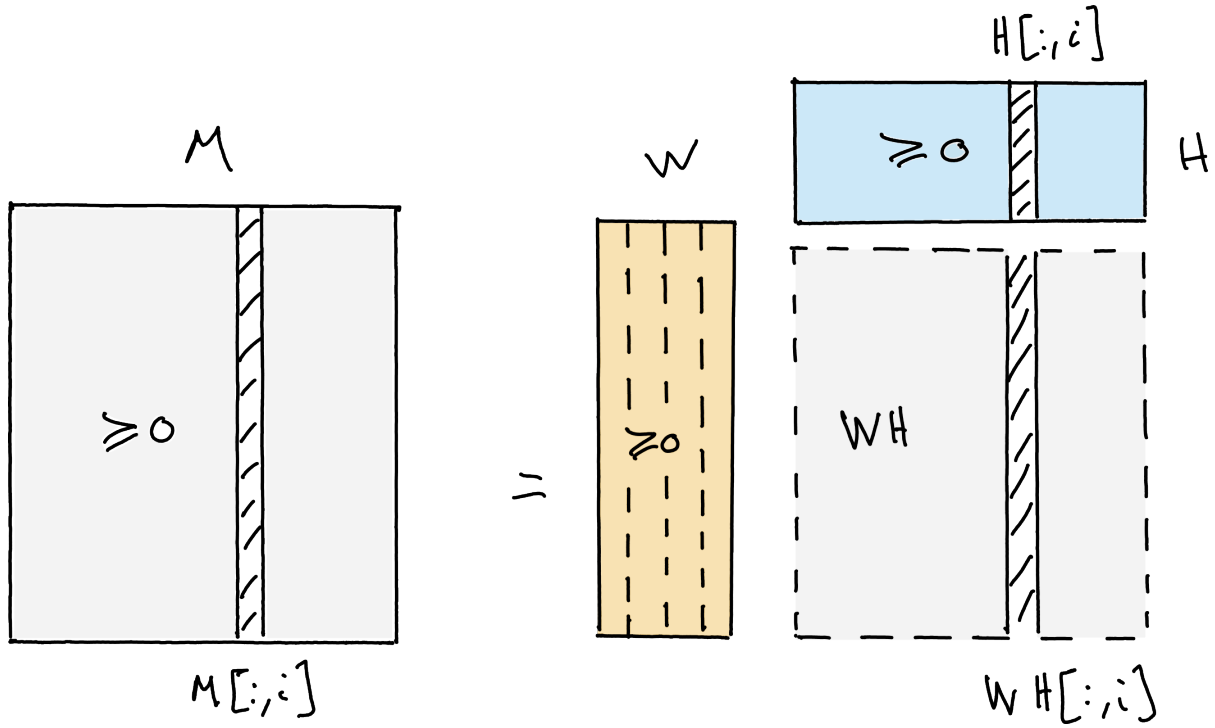


Fig. 6.2: Nonnegative Matrix Factorization (NMF) writes an elementwise nonnegative matrix M as the product of two elementwise nonnegative matrices W and H with respectively fewer columns and rows than M.

One possible route to make LRA parameters identifiable is to impose constraints on them. A constraint that is physically meaningful in many applications is nonnegativity. It applies naturally when factor matrices represent relative concentrations, spectral templates, images, probabilities, user ratings, energy, and other nonnegative physical quantities. The exact NMF model can be formulated as

$$\text{Find } W \in \mathbb{R}_+^{n_1 \times r},\ H \in \mathbb{R}_+^{n_2 \times r} \text{ such that } M = WH^T.$$

Remark

NMF is traditionally denoted with factors W and H. I therefore switch to this notation for this section and use this notation consistently throughout the manuscript.

Unlike unconstrained low-rank approximation, exact and approximate NMF do not admit, in general, closed-form solutions, although the best rank-one approximation may be obtained in closed form as discussed in *Best nonnegative rank-one approximations*. Computing (approximate) NMF amounts to solving an optimization problem. For instance, with the squared Frobenius norm, rank r approximate NMF computes

$$\underset{W \in \mathbb{R}_+^{n_1 \times r},\ H \in \mathbb{R}_+^{n_2 \times r}}{\operatorname{argmin}} \|M - WH^T\|_F^2.$$

Computing exact NMF is an NP-hard problem [Vavasis, 2010]. There exist many strategies, many of which are based on alternating optimization, as discussed in *Alternating optimization*, and, to the best of my knowledge, there is no single best method for computing exact or approximate NMF for all datasets. An algorithm that provides reasonably good performance across a large set of problems is Hierarchical Alternating Least Squares (HALS); see *Nonnegative regressions: NNLS and NNKL* and *Alternating optimization* for a detailed description.

Another difficulty with NMF, compared to unconstrained LRA, is determining the approximation rank. With LRA, one may simply compute the full SVD of the input matrix and truncate according to the distribution of the singular values. Dropping orthogonality implies that the deflation strategy of truncated SVD does not transfer to NMF, and in fact, no deflation strategy exists for NMF that is as simple and general as truncated SVD. In practice, the rank is often guessed by the user, or several approximate NMF models are computed with various ranks, and the best model is kept. The nonnegative rank of a matrix is defined as the minimal integer r such that it admits an exact NMF of rank r.

Geometric interpretation and identifiability of exact NMF

Remark

In this section, we discuss the case of exact NMF, where the equality $M = WH$ holds exactly. The noisy case can be analysed similarly, although one should be cautious of normalization for low-amplitude columns of M.

NMF can be interpreted geometrically as a linear dimensionality reduction technique, with ingredients similar to those in the geometric interpretation of *NNLS*. The columns of the data matrix M live in the cone $\text{col}_+(W)$ spanned by the columns of the matrix W, which are located in the nonnegative orthant, see Fig. 6.3.

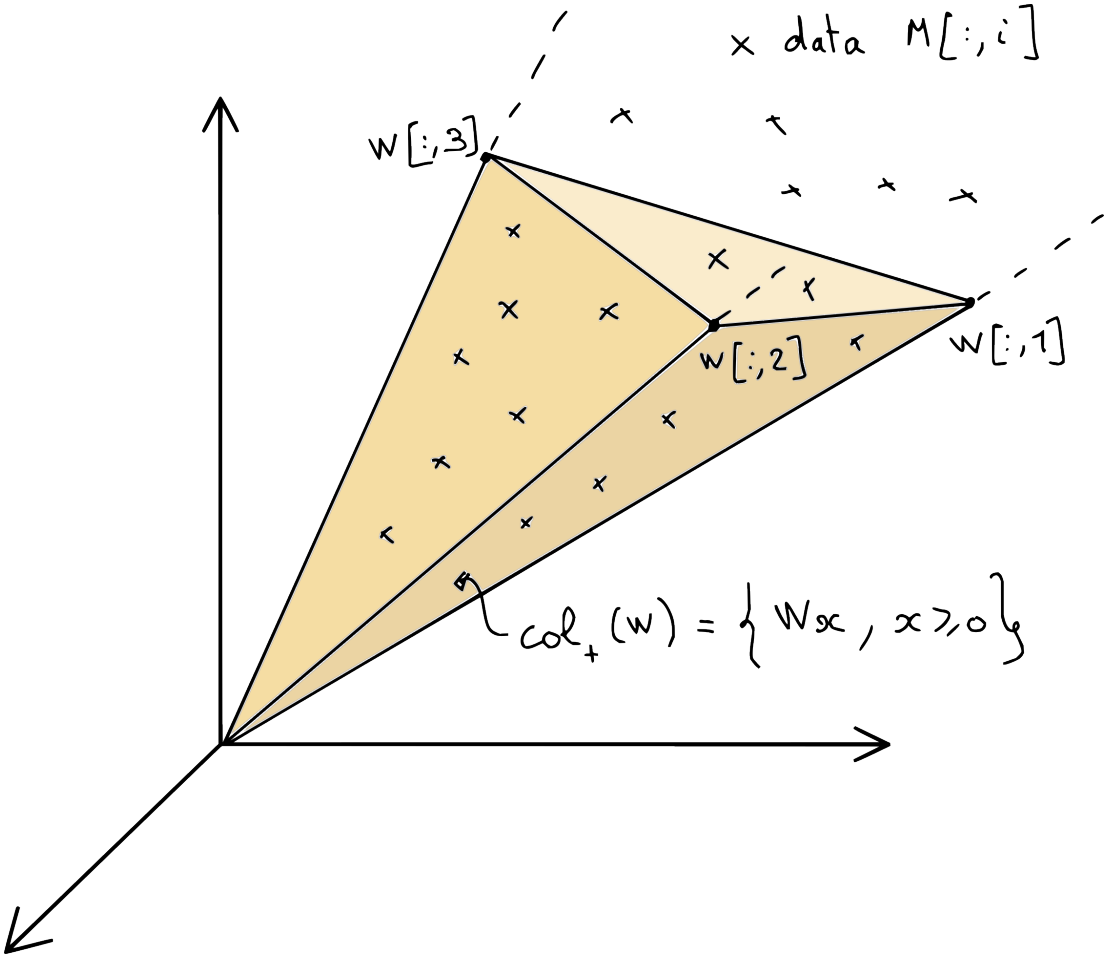


Fig. 6.3: A graphical illustration of one possible NMF of data matrix M. The cone $\text{col}_+(W)$ collects all the positive combinations of columns of matrix W, and must contain all the columns of matrix M.

With this interpretation, it is simple to deduce the non-uniqueness, in general, of NMF. Consider the dimension two and rank-two case, $n_1 = r = 2$, in Fig. 6.4. As soon as $M[:, 1]$ or $M[:, 2]$ is elementwise positive, the purple areas are non-trivial and there are infinitely many solutions. The only way NMF can be unique when n_1 equals the rank is if $W = I$ is the only solution (up to scaling and permutation and its columns), since $M = IM$ is always a valid NMF in that case.

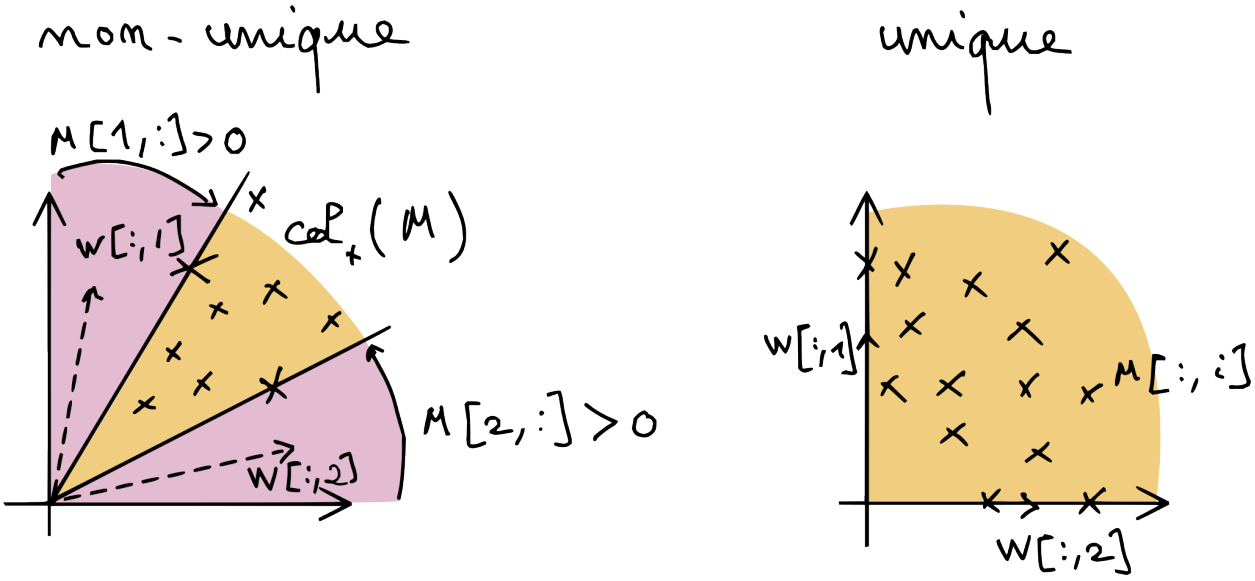


Fig. 6.4: A simple graphical explanation of why NMF without further regularization is generally non-unique. As long the purple areas around the cone spanned by the data points are sufficiently large, there may be infinitely many positive cones that contain the data. The situation is more complex in higher dimensions.

A slightly more involved geometric representation of exact NMF can be obtained by projecting the data and model onto the unit simplex. The data then lives in the intersection of the nonnegative orthant and the unit simplex, which is a polytope.

Remark

Projection of the data matrix M onto the data simplex implies that both matrices W and H have columns that sum to one; see [Gillis, 2020].

This geometric view allows for a finer description of identifiability in low-dimensional settings. Consider the case $n_1 = 3$ and $r = 2$ in Fig. 6.5. We see that identifiability can be achieved without further regularization only if the data points touch the boundary of the simplex.

A key concept for NMF identifiability: sufficiently scattered

One of the many interesting results regarding NMF identifiability states that NMF is identifiable when both columns of H^T and W^T are sufficiently scattered inside the simplex. The book written by Gillis [Gillis, 2020] describes this technical condition in great detail. Intuitively, the sufficiently scattered condition for matrix H^T states that the columns of the data matrix X, when looked from the inside of the cone spanned by matrix W, touch the border of that cone and are sufficiently close to the extreme rays. A different interpretation is that the volume of the convex hull of columns of H^T is large enough to contain the largest sphere contained in the simplex (in fact, a little more is required). This explains the intuition behind sparse NMF and minimum- and maximum-volume NMF, discussed shortly below.

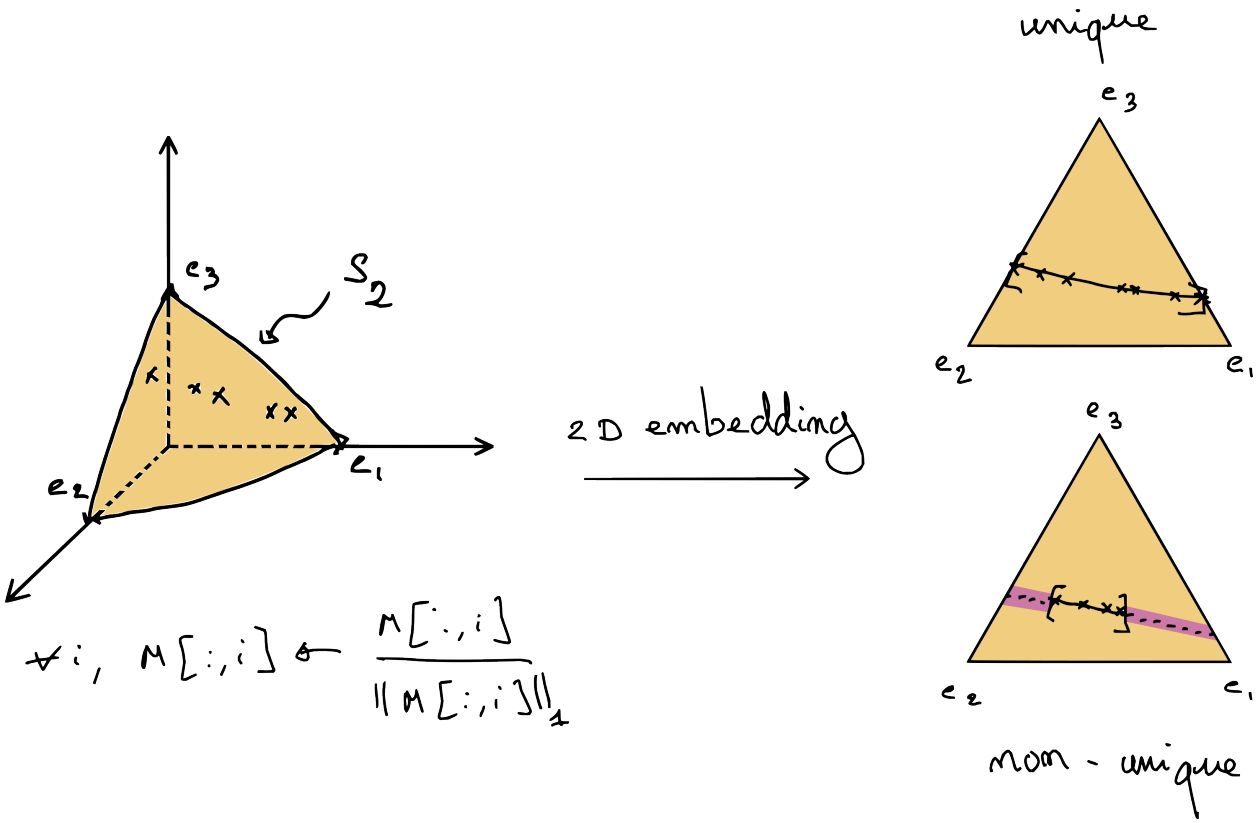


Fig. 6.5: Exact NMF in dimension 3 and rank 2. The data, after ℓ_1 normalization, lies on the simplex $\mathcal{S}_2$. This embedding allows to visualize NMF in dimension three in the 2D plane. The right-hand side shows two possible scenarios, where NMF is either unique if the data points touch the border of the simplex, or not unique if the data are located on a segment strictly contained in the simplex.

Regularized NMF

Given the above observations, it is reasonable to assume in practice that NMF may not be unique. As with other LRA models, regularizations can be added to the model to remove unrealistic solutions.

There exist at least three main choices for regularizing NMF. First, one may assume that the columns of W are contained in the columns of the data M. The resulting model, called separable NMF, writes

$$M = M[:, \mathcal{K}]H^T$$

where $\mathcal{K}$ is a subset of r indices in $[1, n_2]$. Separable NMF allows for restricting the set of NMF solutions drastically, since there is only a combinatorial number of column combinations. Separable NMF is interesting both for interpretability and algorithm design. A typical example is separable exact NMF in the rank-two case, which is solved in closed form using a simple algorithm that picks the largest vector in the data and then the vector farthest from that first component; see Fig. 6.6. Exact separable NMF in general can be solved in polynomial time with a variant of the QR with pivoting algorithm [Gillis, 2014, Gillis and Vavasis, 2014]. Separable NMF is incredibly useful in practice for initializing any NMF algorithm, since it is fast to compute. Because the templates W are extracted directly from the data, separable NMF often yields interpretable results that summarize key patterns in the data. Separable NMF, however, fails when the dataset contains no pure data points, which may occur when all data points are the result of non-trivial mixtures of several components.

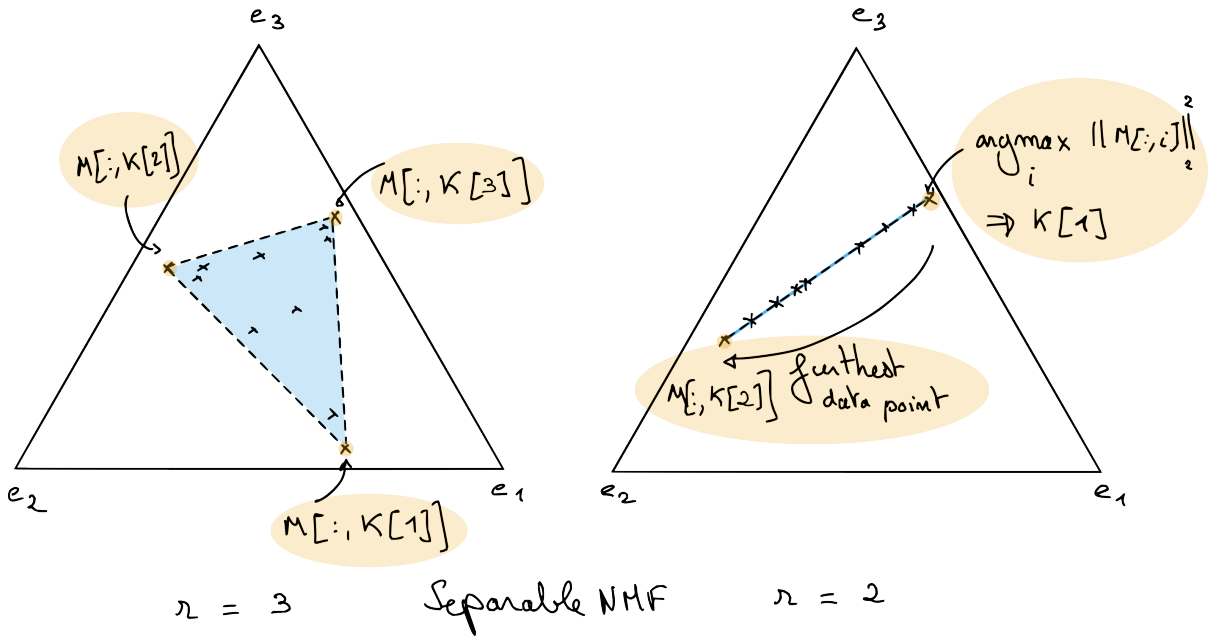


Fig. 6.6: Separable NMF finds the matrix W in the columns of the data matrix M. In the particular case of rank-two separable NMF (on the right), the solution can be obtained by picking for the first columns of matrix W the data point with largest ℓ_2 norm after ℓ_1 normalization, and then choosing the furthest data point as the second column. When the rank is larger, algorithms exist that can compute exactly separable NMF in polynomial time, such as the Successive Nonnegative Projection Algorithm (SNPA) [Gillis, 2014]; the underlying idea behind SNPA is similar to that of OMP.

A second variant of NMF is sparse NMF, in which the entries of the matrices W and/or H are pushed towards zero. A typical formulation of sparse NMF with ℓ_1 regularization writes

$$\underset{W \mathbb{R}_+^{n_1 \times r},\, H \in \mathbb{R}_+^{n_2 \times r}}{\operatorname{argmin}} \|Y - WH^T\|_F^2 + \lambda \left(\|W\|_F^2 + \|H\|_1 \right),$$

with λ a positive hyperparameter, that promotes sparsity in matrix H when large enough. Such regularized low-rank models are discussed further in *Implicit regularization in rLRA*. Sparse NMF can be unique when NMF is not, since sparsity pushes columns of matrix H towards the border of the simplex, maximizing volume.

A third regularized NMF model is the minimum-volume (or maximum-volume [Thanh and Gillis, 2026]) NMF. The idea behind minimal-volume NMF is to choose the matrix W to be as close as possible to the convex hull of the data. In a sense, it is a relaxation of the separability assumption that can still work when pure data do not exist. The minimization of the volume of the cone spanned by the matrix W can be related to the maximization of the volume of the columns of matrix H^T. The volume of the columns of W can be measured as $\log\det(W^TW)$, which is a concave function.

In many practical uses of NMF, it is important to consider whether regularized NMF should be considered instead. After working for several years with NMF on various applications, I believe many authors have underestimated the identifiability problem of NMF. A typical example is *automatic transcription*, where NMF has been used over the last two decades without, to the best of my knowledge, a proper discussion on the uniqueness of the results.

Fitting NMF

To fit NMF to a dataset, several Python libraries are available. Scikit-learn has a nice implementation of both *MU* and cyclic coordinate descent, which allow for sparsity-inducing regularization. Nimfa is a toolbox dedicated to NMF that implements many variants of NMF, including Bayesian and graph-regularized NMF. It has not been updated since 2019, and as far as I know, it is geared towards flexibility rather than performance.

A third and probably less advertised option is to use nonnegative tensor decomposition available in tensorly. The interface is rather simple and, like in Scikit-learn, two algorithms are proposed, namely MU and alternating *HALS*. tensorly, in fact, implements HALS as a NNLS solver, which can be handy as a building block for other nonnegative factorization problems. Since I am a co-developer of tensorly, let us demonstrate how to perform a toy NMF factorization with tensorly.

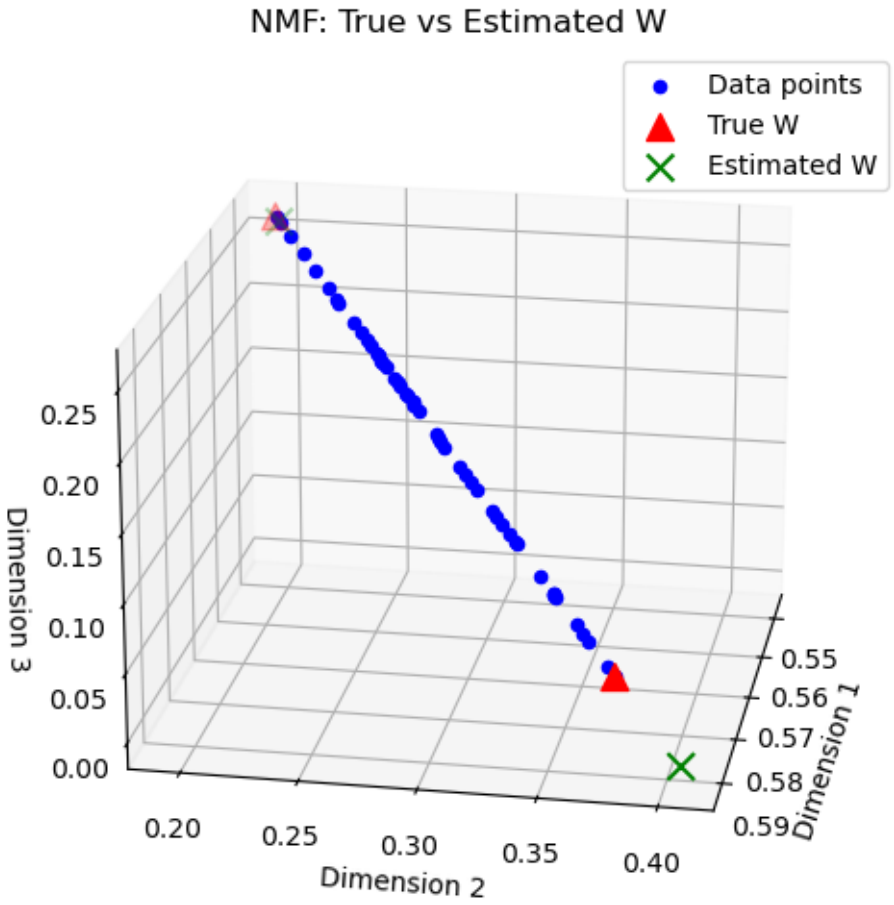
```
import numpy as np
from tensorly.decomposition import non_negative_parafac_hals
import matplotlib.pyplot as plt

# Dimensions
n1, n2 = [3,50]
rank = 2

# Seed
rng = np.random.default_rng(5)

# Generating a dummy nonnegative matrix with exact NMF
W_true = rng.random((n1, rank))
H_true = rng.random((n2, rank))
M = W_true@H_true.T

# Computing NMF
out = non_negative_parafac_hals(M, rank, init="svd", n_iter_max=100, tol=0)
We = out[1][0]
He = out[1][1]
Me = We@He.T

# Computing final error
print(f"Final mean reconstruction error: {np.linalg.norm(M-Me)/n1/n2}")

# Plotting the 3d data points, true W positions as triangles and estimated W positions
# a. normalization of data, data_e and W
We = We / np.sum(We, axis=0)
W_true = W_true / np.sum(W_true, axis=0)
M = M / np.sum(M, axis=0)
Me = Me / np.sum(Me, axis=0)

# b. 3d plotting
```

(continues on next page)

(continued from previous page)

```
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(projection='3d', elev=20, azim=10)
# 3d data
for i in range(n2):
    ax.scatter(M[0,i], M[1,i], M[2,i], c='blue', label='Data points' if i==0 else "")
# increase markersize for better visibility
ax.scatter(W_true[0,:], W_true[1,:], W_true[2,:], s=100, c='red', marker='^', label=
↪'True W')
ax.scatter(We[0,:], We[1,:], We[2,:], s=100, c='green', marker='x', label='Estimated W
↪')
ax.set_xlabel('Dimension 1')
ax.set_ylabel('Dimension 2')
ax.set_zlabel('Dimension 3')
plt.title('NMF: True vs Estimated W')
plt.legend()
plt.grid()
plt.show()
```

```
Final mean reconstruction error: 1.2146666877436951e-05
```

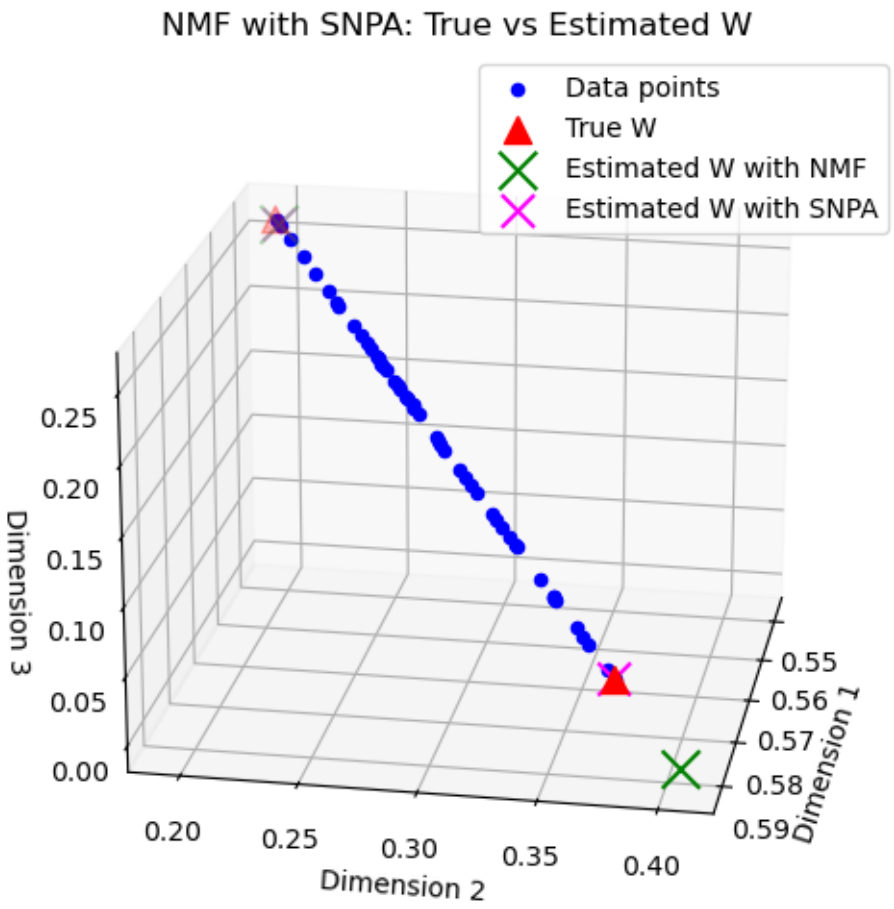


We can observe that the data matrix indeed follows an exact rank-two NMF, since all data points lie on a segment in the nonnegative orthant. Note that the true NMF with columns of matrix W close to the data points is not recovered by

our NMF algorithm despite the small fitting error. Here, separable NMF or minimum-volume NMF would allow us to recover almost exactly the ground truth. An implementation of the nonnegative variant of QR with pivoting for NMF, SNPA [Gillis, 2014], is provided with this manuscript. Indeed, separable NMF in this toy example can recover a better estimation of the true parameter matrix W, and therefore more accurate coefficients H.

```
from tensorly_hdr.sep_nmf import snpa

_, We_snpa, He_snpa = snpa(Me, rank)
He_snpa = He_snpa.T
Me_snpa = We_snpa@He_snpa.T
print(f"Final mean reconstruction error with SNPA, {np.linalg.norm(M-Me_snpa)/n1/n2}")

# Plotting the 3d data points, true W positions as triangles and estimated W␣
 ↪positions with SNPA
fig = plt.figure(figsize=(8,6))
ax = fig.add_subplot(projection='3d', elev=20, azim=10)
# 3d data
for i in range(n2):
    ax.scatter(M[0,i], M[1,i], M[2,i], c='blue', label='Data points' if i==0 else "")
# increase markersize for better visibility
ax.scatter(W_true[0,:], W_true[1,:], W_true[2,:], s=100, c='red', marker='^', label=
 ↪'True W')
ax.scatter(We[0,:], We[1,:], We[2,:], s=200, c='green', marker='x', label='Estimated␣
 ↪W with NMF')
ax.scatter(We_snpa[0,:], We_snpa[1,:], We_snpa[2,:], s=150, c='magenta', marker='x',␣
 ↪label='Estimated W with SNPA')
ax.set_xlabel('Dimension 1')
ax.set_ylabel('Dimension 2')
ax.set_zlabel('Dimension 3')
plt.title('NMF with SNPA: True vs Estimated W')
plt.legend()
plt.grid()
```

```
Final mean reconstruction error with SNPA, 1.9743863064816314e-05
```



Note that in the *rank-two case*, exact separable NMF can be instead computed exactly with a cheaper algorithm. The simulation above is merely a toy example, and SNPA is useful for all separable NMF problems with rank greater than two.

6.2 Tensor decompositions

Tensors are often loosely defined as multiway arrays, that is, extensions of matrices with more than two dimensions. In reality, there are several definitions of tensors in the world of applied mathematics that coincide:

- Tensors can be seen as multiway arrays.
- Tensors can be seen as representations of multilinear forms (like matrices represent bilinear forms).
- Tensors can be seen as representations of multilinear maps (like matrices encode linear operators). The notion of covariant/contravariant spaces is defined in this context.
- Tensors are vectors in a tensor space (like a matrix M equals $\sum_{i,j} M[i,j] e_i e_j^T$ with e_i, e_j canonical basis vectors), which is how I like to define them.

While for matrices, the coexistence of these concepts does not seem to pose significant issues in the scientific community, these definitions, while compatible, lead to serious misunderstandings between researchers of different scientific fields. In classical mechanics, tensors are generally regarded as multilinear maps, whereas in quantum mechanics, they are elements of tensor spaces of high order (one vector space per quantum particle in the system). In data science, they are generally considered multiway arrays, and in statistics, they are often used to describe higher-order moments. In algebra, tensors

are used to represent higher-order polynomials and their associated multilinear form. This leads to confusing discussions, especially regarding the basis of representation for tensors, which is critical in the operator representation point of view but largely irrelevant in the multiway array point of view. These different views even gave birth to a dedicated Wikipedia page that clarifies these various definitions.

In these fields, tensors are used for different reasons, have different characteristics, and represent data and operators. My personal advice to fellow researchers, when faced with scary tensor concepts and notation, is to go back to the matrix case that most of us are more familiar with and then extrapolate the concepts from matrices to tensors. Some authors have long advocated that tensors are a completely different object from matrices, and that their study requires specialized tools and mathematical intuition. While this statement is not false, in my opinion, it is in fact largely exaggerated. What is special is not often tensors but rather unconstrained LRA for matrices and the existence of SVD and the *Eckart-Young theorem*. Most problems encountered when working with NMF, such as identifiability, nonconvex optimization, or NP-hardness, hold for many tensor decompositions, both constrained and unconstrained. A particularity of tensors, however, is the necessity of efficient high-order contractions, that is, computing sums of multiway arrays across many indices as fast as possible; see, for instance, this Dagstuhl report to which I modestly contributed [Bientinesi *et al.*, 2022].

In the following, a minimal description of tensor decomposition is provided to understand the contributions summarized later in the manuscript. For a better reference on tensor decompositions, I advise reading either the recent book of Ballard and Kolda [Ballard and Kolda, 2025] for a quick practical understanding, or the book of Hackbush [Hackbusch, 2012] for a deep mathematical description of tensors. I also enjoyed reading an older reference from Laurent Schwartz [Schwartz, 1975] on tensor products.

6.2.1 Tensors and the tensor product

The tensor product

We start by defining a tensor space in terms of linear algebra. Tensor spaces emerge as a way to provide a "nice" vector space for bilinear objects. Consider the couple $(x, y) \in E \times F$ with E and F two given real (or complex) vector spaces. The carthesian product $E \times F$ is a vector space of dimension $\dim(E) + \dim(F)$, and scalar multiplication with a scalar λ write $\lambda(x, y) = (\lambda x, \lambda y)$. The Cartesian product simply juxtaposes the vector spaces E and F, but it does not allow much interaction between them. Notice, for instance, that there is no principled way in the Cartesian product to multiply each vector x and y by two different scalars. Vectors x and y in a Cartesian product are essentially living in two unrelated spaces.

In data mining and low-rank approximations in particular, there is a clear interest in extracting patterns that span across the two dimensions of a matrix. Think how NMF extracts patterns (column of matrix W) for each dimension jointly with the activations (columns of matrix H). Therefore, we need an underlying vector space that contains interactions between the two vector spaces, larger than the Cartesian product. One way to do this is to consider multilinear maps, such as the scalar product,

$$\langle x, y \rangle = u(x, y),$$

where u is a multilinear map acting on the couple (x, y). The core idea of tensor spaces and the tensor product is to find a canonical way to map the couple (x, y) to a vector $\otimes(x, y) := x \otimes y$, where $\otimes$ is the tensor product, such that the multilinear map u decomposes as a linear map w_u acting on the tensor product:

$$u(x, y) = w_u(\otimes(x, y)) = w_u(x \otimes y).$$

The tensor product is, therefore, a canonical multilinear map that can be used to make all multilinear maps linear. An important mathematical result, proved for instance in [Schwartz, 1975] and called the universality theorem, is that this canonical multilinear map $\otimes$ always exists and that for any multilinear map u and fixed tensor product $\otimes$, there is a **unique** linear map w_u such that $u(x, y) = w_u(x \otimes y)$. The set $E \otimes F := \{x \otimes y, (x, y) \in E \times F\}$ is a vector space, coined the tensor product space. Its dimension is $\dim(E)\dim(F)$ and it is generated by basis elements $e_i \otimes f_j$ for two bases $\{e_i\}_i$ and $\{f_j\}_j$ of respectively vector spaces E and F.

The tensor product itself is not unique. However, it can be shown that any two tensor products are related through an isomorphism. This fact is very well known to all data scientists. Take the outer product xy^T, and the Kronecker product $x \otimes_K y$. It turns out both are tensor products, related through a trivial isomorphism: row-first vectorization.

For most practical use cases, **one may identify vector spaces E and F with $\mathbb{R}^{n_1}$ and $\mathbb{R}^{n_2}$, and the tensor product with the outer product**

$$(x \otimes y)[i, j] = xy^T[i, j] = x[i]y[j].$$

Going back to the scalar product example, with the outer product as a tensor product, the linear map associated with the bilinear scalar product is the trace operator:

$$\langle x, y \rangle = \text{Tr}(xy^T).$$

One interesting piece of multilinear algebra theory, however, is that, in finite dimensions, the vector space of linear maps acting on a tensor space is also a tensor space, of the form $\mathcal{L}(E) \otimes \mathcal{L}(F)$ [Hackbusch, 2012]. The natural tensor product in linear operator spaces for matrix representation is the Kronecker product, and this can be used to generalize tensor decompositions to linear operators quite naturally, as done in the PhD thesis of Cassio Fraga Dantas [Dantas, 2019]. When considering linear operators acting on tensors, for instance A acting on x and B acting on y, we simply write them in the form $(A \otimes B)(x \otimes y) := Ax \otimes By$.

Important: This discussion holds not only for two vector spaces E and F, but for any number of vector spaces E_i, with $i \leq d$. A tensor is any element of the tensor space $E_1 \otimes \cdots \otimes E_d$; we call the integer d the order of the tensor, or the number of modes of the tensor. Mode i, or dimension i, denotes the particular vector space E_i.

Rank-one tensors are generators of the full tensor space

We have seen by construction that any vector $x \otimes y$ for $(x, y) \in E \otimes F$ belongs to the tensor space $E \otimes F$. Since $E \otimes F$ is a vector space, we may add an arbitrary number of such vectors and stay inside $E \otimes F$. This means that any vector of the form

$$\sum_{i \leq p} x_i \otimes y_i$$

with p some finite integer is in $E \otimes F$. We call elements of $E \otimes F$ of the form $x \otimes y$ **rank-one tensors**.

This construction raises two immediate questions of crucial importance

1. Can all elements of $E \otimes F$ be described as a sum of rank-one tensors? If so, how large must p be?
2. Are all elements of $E \otimes F$ rank-one tensors, and if not, what is the smallest number of terms that describes a given tensor?

The answer to the first question is positive: rank-one tensors are generators of the linear space $E \otimes F$. This is clear in finite dimension since for two bases $\{e_i\}_i$ and $\{f_j\}_j$ of respectively E and F, the set $\{e_i \otimes f_j\}_{i,j}$ is a basis for $E \otimes F$. With matrices and using the outer product, we are simply stating here that any matrix M can be written in the canonical basis

$$M = \sum_{i,j} M[i, j]e_i f_j^T$$

with e_i and f_j vectors null everywhere except in position respectively i and j where they contain one. This also shows that using more than $d = \dim(E)\dim(F)$ is not required.

Remark

The fact that $\{e_i \otimes f_j\}_{i,j}$ is a basis of $E \otimes F$ is often a direct consequence of the construction of $E \otimes F$, while the universality theorem is derived as a corollary. The converse is also possible, as proposed by Schwarz.

The answer to the second part of the second question is directly related to the canonical tensor decomposition described in the next section. Regarding the first half of the question, it is simple to exhibit a tensor that cannot be written as a rank-one tensor. Take $e_1 f_1^T + e_2 f_2^T$ with the previous bases notations. If there exists a rank-one tensor xy^T such that $xy^T = e_1 f_1^T + e_2 f_2^T$, then denoting z_x an orthogonal vector to x,

$$z_x^T x y^T = 0$$

for the left-hand side, while

$$z_x^T (e_1 f_1^T + e_2 f_2^T) = \langle z_x, e_1 \rangle f_1^T + \langle z_x, e_2 \rangle f_2^T .$$

Because f_1 and f_2 are linearly independent, the two equalities cannot happen simultaneously, and therefore $e_1 f_1^T + e_2 f_2^T$ cannot be written as a rank-one tensor (matrix). The counter-example holds for tensor spaces of higher order as well.

6.2.2 CP decomposition

The Canonical Polyadic decomposition (CP decomposition or CPD), also called PARAFAC decomposition, is the decomposition of a given tensor into a sum of rank-one tensors with as few terms as possible [Carroll and Chang, 1970, Harshman, 1970, Hitchcock, 1927]. This minimal number of terms is called the rank of a tensor, and is an extension of matrix rank. Indeed, matrix rank can be described, among all possible equivalent definitions, as

$$\text{rank}\,(M) = \min\{q \in \mathbb{N},\ M = \sum_{i \leq q} x_i \otimes y_i \text{ where } \ x_i \otimes y_i \in E \otimes F\}$$

when E and F are real finite-dimensional vector spaces.

Notice that the sum $\sum_{i \leq q} x_i \otimes y_i$ can be written in matrix format as $M = XY^T$, where matrices X and Y contain vectors x_i and y_i stacked columnwise. In the matrix case, the CP decomposition is therefore simply an exact unconstrained LRA model. Matrix CP decomposition is never unique, and can be computed efficiently with the SVD. An approximate CP decomposition, defined as an *approximate LRA model*, is also easy to compute with the truncated SVD.

From numerical linear algebra and data science perspectives, the CP decomposition for tensors with more than two modes is fundamentally different: the CP decomposition for higher-order tensors is identifiable under mild assumptions, but its computation can be extremely challenging.

CP decomposition for third-order tensor and identifiability

Let us now discuss the tensor CP decomposition for real multiway arrays, with the outer product $(x \otimes y \otimes z)[i,j,k] = x[i]y[j]z[k]$ as the tensor product. For a third-order tensor T in $\mathbb{R}^{n_1 \times n_2 \times n_3}$, the CP decomposition of T is a minimal decomposition

$$T[i,j,k] = \sum_{q=1}^{r} A[i,q]B[j,q]C[k,q]$$

where matrices A, B, and C are sometimes called the factor matrices of the CP decomposition, and the rank r is as small as possible.

The key property of the CP decomposition is that there is typically a unique set of rank-one tensors $A[:,q] \otimes B[:,q] \otimes C[:,q]$ that decomposes a given tensor T. If these factors can be computed numerically with guaranteed stability [Vannieuwenhoven, 2017] and if the CP decomposition matches a physically meaningful model, then the estimated factor matrices

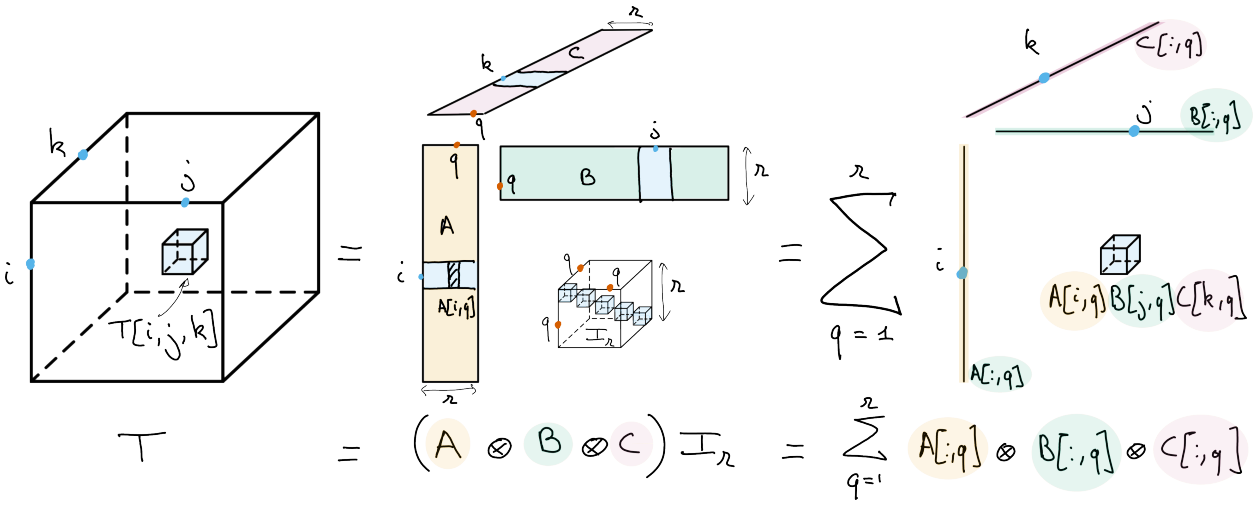


Fig. 6.7: Canonical Polyadic Decomposition (CPD) of rank r of a data tensor T. The CPD can be viewed as a multilinear diagonalization technique, with bases A, B and C respectively spanning the columns, rows and fibers of the tensor, or as an atomic decomposition into a sum of r rank-one tensors.

can be interpreted without the need for further regularization. This nice property can be contrasted with the matrix case, where regularizations, such as nonnegativity for NMF, are required to hope for identifiability. The most well-known identifiability conditions for the CP decomposition are probably the Kruskal condition; see, for instance, the accessible proof provided by Stegeman and Sidiropoulos [Stegeman and Sidiropoulos, 2007].

Why is CP decomposition identifiable while matrix LRA is not

When the CP decomposition was studied by Richard Harshman in psychometrics in 1970 [Harshman, 1970], he provided a simple proof from Jenrich of its parameter identifiability, albeit with strong assumptions. He had a very strong intuition for why the model could be identifiable. The core idea is that CP decomposition fixes rotation ambiguities observed in matrix LRA.

To see this, observe that the CP decomposition of a third-order tensor can be written in terms of coupled matrix factorization as

$$\forall k \leq n_3,\ T[:,:,k] = A\mathrm{Diag}\left(C[:,k]\right)B^T.$$

In other words, the CP decomposition is the joint diagonalization of each slice $T[:,:,k]$ of the tensor. Joint diagonalization means that the same bases for the column and row spaces A and B are used for all slices.

To explain further why joint diagonalization fixes rotation ambiguities, let us assume that there are only two slices in the tensor, $n_3 = 2$, and assume these two slices admit an exact CP decomposition with full rank matrices A and B and nonzero entries in one of the two columns of matrix C. It is possible to estimate A and B directly from the tensor T using Jenrich's algorithm. Observe that

$$T[:,:,1]T[:,:,2]^\dagger = A\mathrm{Diag}(\frac{C[:,1]}{C[:,2]})A^\dagger,$$

which is exactly the nonzero part of the eigenvalue decomposition of the rank-deficient symmetric matrix $T[:,:,1]T[:,:,2]^\dagger$. It is unique up to scaling and permutations, provided the singular values are distinct. Similarly, matrix B can be obtained from the eigenvalue decomposition of $T[:,:,1]^\dagger T[:,:,2]$. The estimation of matrix C can be performed by solving an overdetermined linear system, with a unique solution since matrices A and B are full column rank.

Jenrinch's algorithm shows that in many practical applications, the CP decomposition is essentially unique, even with

two slices; while a low-rank factorization of each slice has infinitely many solutions, the intersection of these solution sets is singular, up to permutations and scaling ambiguities.

Remark

The "proof", as reported by Harshman from Jenrich, is wrong as it assumes the uniqueness of the decomposition to prove the uniqueness of the decomposition itself. Jenrich's algorithm, however, provides a constructive proof of the uniqueness of CP decomposition when the factor matrices are of full rank. Jenrich was a collaborator of Harshman, who credited him with the Jenrich algorithm and the proof of uniqueness, but Jenrich did not co-sign Harshman's seminal work.

The CP decomposition model, like matrix LRA, has many applications. This is because multilinear maps are often encountered as simple yet efficient models in physics and mathematics for describing complex systems. The way tensor products are built on top of the multilinear tensor product ensures that the CP decomposition yields a sparse multilinear decomposition of the data. In other words, for each mode and for instance the first one, the map $A \mapsto T$ is linear. The CP decomposition is essentially the description of a tensor as a sum of a few simple, discrete multilinear maps, the discrete analogue of a decomposition

$$f(x,y,z) = \sum_{q=1}^{r} f_{1,q}(x) f_{2,q}(y) f_{3,q}(z).$$

The number of parameters in the CP decomposition is $(n_1 + n_2 + n_3)r$, much smaller than the $n_1 n_2 n_3$ entries in the tensor T when the rank is smaller than the product of any two dimensions. Therefore, CP decomposition can also be used purely as a dimensionality reduction tool.

Note: There exist several shorthand notations for CP decomposition. Many authors use the so-called Kruskal notation $T = [\![A, B, C]\!]$, which I will also use in this manuscript [Kolda and Bader, 2009]. However, in some settings, I prefer working with the more rigorous notation

$$T = (A \otimes B \otimes C)\, I_r$$

where $I_r[i,j,k] = \delta_{ir}\delta_{jr}\delta_{kr}$ is a diagonal tensor of ones. The tensor product $\otimes$ is here meant in the sense of linear operators. This is in practice equivalent to the multiway product notation

$$T = A \times_1 B \times_2 C \times_3 I_r$$

but also makes explicit the tensor structure of the linear operators acting on tensors in finite dimensions.

Approximate CP decomposition computation

On the surface, approximate CP decomposition is simply a particular case of approximate LRA. Let T be a data tensor to decompose. Approximate CP decomposition aims at finding a rank r tensor $[\![A, B, C]\!]$ as close as possible to T; the rank of the approximation r is fixed in advance. If the Frobenius norm $\|T\|_F^2 = \sum_{i,j,k} T[i,j,k]^2$ is used as an error metric, the approximate CP decomposition computes the best rank-r approximation by solving the optimization problem

$$\underset{A \in \mathbb{R}^{n_1 \times r},\, B \in \mathbb{R}^{n_2 \times r},\, C \in \mathbb{R}^{n_3 \times r}}{\operatorname{argmin}} \|T - [\![A, B, C]\!]\|_2^2.$$

Like most LRA-related optimization problems, this is a nonconvex, multiblock optimization problem that can be addressed in practice by *alternating optimization*. The workhorse algorithm for fitting approximate CP decomposition is the Alternating Least Squares (ALS) algorithm. It updates each matrix A, B, and C in sequence, which, in general, can

be done in closed form since the cost is quadratic in each parameter matrix. For instance, the update for the parameter matrix A is obtained by solving the linear system

$$T_{[1]}(B \odot C) = A(B^T B * C^T C),$$

where $\odot$ is the Khatri-Rao product (columwise Kronecker products stacked horizontally), and $*$ is the elementwise product. Matrix $T_{[1]}$ is an unfolding of the tensor, namely, stacked slices of the tensor along the first mode. This formula can be obtained by computing the gradient of the cost with respect to the matrix A and then setting it to zero. ALS is a simple algorithm to describe at a high level, but, as is often the case in tensor computations, its efficient implementation requires advanced tools from numerical linear algebra and high-performance computing. There exist **many** other algorithms to compute approximate CP decomposition based on gradient descent (alternating or not), preconditioning, acceleration, algebraic methods, or second-order optimization [Acar *et al.*, 2011, De Lathauwer *et al.*, 2004, Rajih *et al.*, 2008, Xu and Yin, 2013] (this list could be significantly longer; surveying this extensive literature is beyond the scope here).

Remark

Many cumbersome notations, such as unfoldings and the Khatri-Rao product, are used to describe optimization algorithms. I will make as little use as possible of these notations and therefore avoid defining them in this section. It is not important to understand them to follow the overall discussion; informal definitions are provided in the *notation* section. In fact, in tensorly, these operations are not actually performed, and tensor contractions are preferred for performance reasons.

Ill-posed approximation and nonnegative CP decomposition

Approximate CP decomposition is an ill-posed problem. The set of rank r tensors is not closed, therefore it may happen in theory that there exists no best rank r approximation [Hillar and Lim, 2013]. This causes practical issues that are difficult to predict and handle, such as slow convergence of ALS (so-called swamps) or even diverging components that cancel at infinity.

Since this manuscript is mostly concerned with regularized LRA, this problem is often avoided entirely. What is required to obtain the existence of a minimum in the approximation problem is coercivity of the cost on the domain definition. This is achieved when any coercive regularization is added, such as ℓ_2 regularization. The continuity of the cost, alongside coercivity, ensures that there exists a compact subset of the definition domain that contains the solution, which is therefore reached. In the context of CP decomposition, regularizations therefore not only help improve the interpretability of the parameters (although CP decomposition can be interpretable without such constraints, they often help in practice), but also make the approximation problem well-posed and easier to solve in terms of the optimization landscape.

A classic example of regularized CP decomposition is approximate nonnegative CP Decomposition (nCPD), defined as the solution to the optimization problem

$$\underset{A \in \mathbb{R}_+^{n_1 \times r},\, B \in \mathbb{R}_+^{n_2 \times r},\, C \in \mathbb{R}_+^{n_3 \times r}}{\operatorname{argmin}} \|T - [\![A, B, C]\!]\|_2^2.$$

Notice that the matrices have nonnegative entries. Algorithms to compute nCPD are similar to algorithms for solving NMF, and rely on NNLS solvers described extensively in *Nonnegative regressions: NNLS and NNKL*. A generic alternating algorithms that solves NNLS problems with respect to each factor matrix is sometimes called Alternating Nonnegative Least Squares (ANLS).

Notice also that coercivity is achieved with nonnegativity constraints, since the components cannot cancel out, and the cost function increases toward infinity as the components grow. Therefore, the best nonnegative low-rank approximation of a tensor always exists.

6.2.3 Tucker decomposition

In the CP decomposition, the factor matrices A, B, and C can be seen as linear operators that map a diagonal tensor to the data tensor T. In fact, using the linear operator tensor space point of view, the map $A \otimes B \otimes C$ is a rank-one linear operator acting on tensors. The idea of Tucker decomposition is to generalize this observation by using rank-one linear operators to map large tensors to smaller tensors, thereby performing multilinear dimensionality reduction.

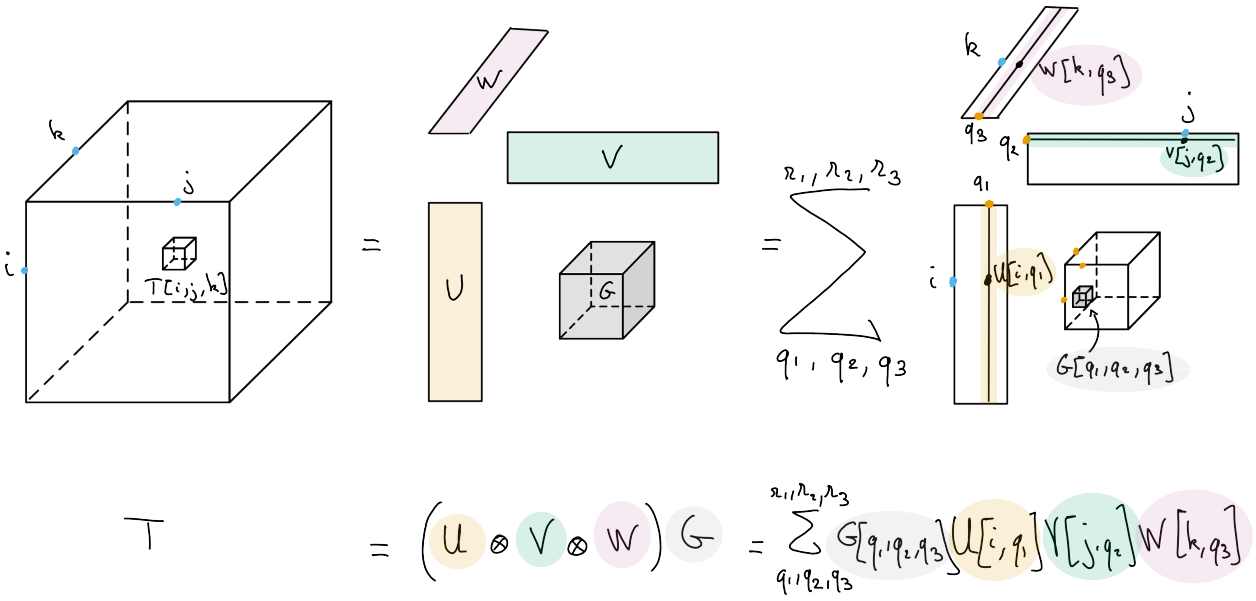


Fig. 6.8: Tucker decomposition of a data tensor T, with multilinear ranks r_1, r_2 and r_3. The Tucker decomposition can be viewed as a multilinear dimensionality reduction technique, with orthogonal bases U, V and W respectively spanning the columns, rows and fibers of the tensor. Tensor G stores the coefficients into a tensor of size $r_1 \times r_2 \times r_3$.

For a data tensor $T \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, the Tucker decomposition finds three matrices $U \in \mathbb{R}^{n_1 \times r_1}$, $V \in \mathbb{R}^{n_2 \times r_2}$, $W \in \mathbb{R}^{n_3 \times r_3}$ and a smaller core tensor $G \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ with $r_i \leq n_i$ such that

$$T = (U \otimes V \otimes W)\, G = U \times_1 V \times_2 W \times_3 G.$$

with $\times_i$ the multiway product *defined above*. Matrices U, V, and W as well as the core tensor G are the parameters of the model, which involves $n_1 r_1 + n_2 r_2 + n_3 r_3 + r_1 r_2 r_3$ parameters. This is significantly smaller than the number of entries in the tensor T when the so-called multilinear ranks r_i are smaller than the tensor dimensions. A tensor following an exact Tucker decomposition is written as

$$T = [\![U, V, W; G]\!].$$

The Tucker decomposition is more similar to matrix LRA than the CP decomposition in terms of algorithmic development and uniqueness properties. Tucker decomposition has rotation ambiguities in each mode, since $U \times_1 G = (UQ \times_1)(Q^T \times_1 G)$ and is therefore never unique without further regularizations. Orthogonality is typically employed on each mode [Lathauwer *et al.*, 2000], although nonnegativity is also a popular choice [Cohen *et al.*, 2017, Mørup *et al.*, 2008] that I have worked with, see *Summary of HDR contents*. The identifiability of Nonnegative Tucker decomposition is a longstanding problem that has seen some recent progress [Saha *et al.*, 2025].

Tucker decomposition with orthogonality constraints can be computed almost exactly with the HOSVD algorithm [De Lathauwer and Vandewalle, 2004].

Tucker decomposition vs Tucker format

Because the Tucker decomposition lacks identifiability, it is more often used as a means of dimensionality reduction. In some computer science and applied mathematics communities, it is therefore called the Tucker format, while decomposition is used for models with uniqueness properties. I agree with this observation, but still refer to the Tucker format as the Tucker decomposition in this manuscript, as it is more common in the data science community.

CHAPTER

SEVEN

NONNEGATIVE REGRESSIONS: NNLS AND NNKL

Note: This section is based partly on a doctoral course I taught at the doctoral school of ITWIST 2020, and partly on more recent works around the MU algorithm. It is quite dense and long; however, nonnegative optimization problems are crucial tools to understand the rest of this manuscript.

Nonnegative regression problems are central in nonnegative low-rank approximations. Many LRA algorithms are based on *alternating optimization*, and optimization methods for nonnegative LRA are often heavily inspired by algorithms designed for nonnegative regression. Nonnegative regression is a class of optimization problems defined as

$$\underset{x \geq 0}{\operatorname{argmin}} \; f(y, Wx)$$

where $f(y, z)$ is a nonnegative, separable, convex (with respect to the second variable) loss function comparing two vectors $y \in \mathbb{R}^m$ and $z \in \mathbb{R}^m$ elementwise, and $W \in \mathbb{R}^{m \times n}$ is an observation matrix. The constraint $x \geq 0$ is meant elementwise. Typically, both the observations y and the matrix W are also elementwise nonnegative. This section introduces two particular nonnegative regression problems, Nonnegative Least Squares (NNLS) and Nonnegative Kullback-Leibler regression (NNKL), that will be of particular interest for the rest of the manuscript.

7.1 Nonnegative Least Squares

NNLS is a classic quadratic optimization problem, maybe one of the simplest generalizations of ordinary least squares. I am unsure of its origin, but the oldest mention of NNLS I am aware of is in the book of Lawson and Hanson on optimization methods for least squares problems [Lawson and Hanson, 1974]. Unlike ordinary least squares, however, NNLS does not have, in general, a closed-form solution.

7.1.1 Problem description

NNLS can be formulated as the following optimization problem, with $f\left(y, Wx\right) = \|y - Wx\|_2^2$:

$$\text{Find } x^* \in \underset{x \geq 0}{\operatorname{argmin}} \; \|y - Wx\|_2^2.$$

In general, the projected unconstained least squares estimate, $\left[W^\dagger y\right]^+$, is **not** a solution to NNLS, see *Proco-ALS for fast nCPD* for more details.

The cost function of NNLS is coercive (with respect to z) and continuous and the set of admissible solutions is nonempty, which ensures the existence of a solution. If matrix $W \in \mathbb{R}^{m \times n}$ is full column rank, which is often the case in low-rank approximation problems, then the cost is also strongly convex (and Lipschitz-smooth), ensuring the NNLS solution is unique. When the matrix W is not full column rank, the discussion on the uniqueness is more difficult. One of the interests of NNLS over ordinary least squares, however, is that the solution for underdetermined systems can still be unique, see the *NNLS KKT conditions* below.

7.1.2 Geometric interpretation

When W has desirable properties such as elementwise nonnegativity, NNLS is equivalent to the orthogonal projection of the input vector y on a pointed cone. Indeed, defining $\text{col}_+(W) = \{Wx, x \geq 0\}$, we see that $\text{col}_+(W)$ is always a cone. Let $Wx_1, Wx_2 \in \text{col}_+(W)$, for any nonnegative weights λ_1 and λ_2, it holds that $\lambda_1 Wx_1 + \lambda_2 Wx_2 = W(\lambda_1 x_1 + \lambda_2 x_2)$ is in $\text{col}_+(W)$. NNLS finds the closest vector z to the input y in $\text{col}_+(W)$,

$$\underset{z=Wx,\, x\geq 0}{\text{argmin}} \quad \|y - z\|_2^2,$$

which is exactly the orthogonal projection on $\text{col}_+(W)$. Depending on the matrix W, the cone $\text{col}_+(W)$ can either be the whole search space $\mathbb{R}^m$, or a pointed cone. The latter is the commonly encountered case in nonnegative LRA problems, in particular when the matrix W is elementwise nonnegative. Below is an illustration in dimensions $m = n = 3$.

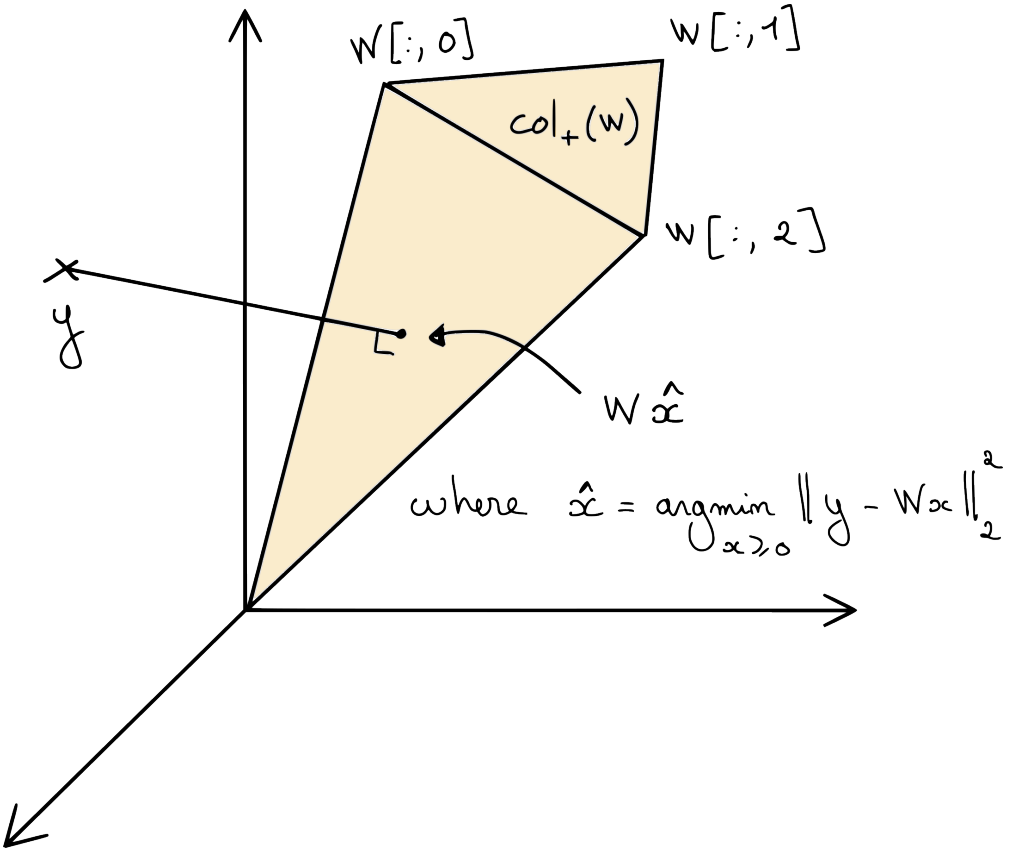


Fig. 7.1: A graphical illustration of the projection of data vector y on the cone $\text{col}_+(W)$ collecting all the positive combinations of columns of matrix W. The NNLS problem is equivalent to computing the coefficients of the projected data vector in the cone; note that when the data vector y is outside $\text{col}_+(W)$, as illustrated here, the solution is sparse since the projection is located on a facet of the cone.

From this geometric interpretation, we can deduce a few results:

- If the measurement vector y lies inside the cone $\text{col}_+(W)$, then an exact reconstruction is possible. We show with the *KKT conditions* that the least squares estimate $W^\dagger y$ is a NNLS solution. The solution may be non-unique.
- If matrix W has more columns than rows, but these columns are in general position, and vector y lies outside $\text{col}_+(W)$, then in general the NNLS solution is unique.
- When vector y lies outside the cone $\text{col}_+(W)$, the projection solution $z^* = Wx^*$ is located on a facet of the cone. In fact, it is the orthogonal projection of y on this facet. Hence, x^* may be sparse.

These results can be formalized using the KKT conditions.

7.1.3 KKT conditions and the night sky theorem

Deriving the Karush-Kuhn-Tucker conditions (see, *e.g.*, [Boyd and Vandenberghe, 2004]) for NNLS is not difficult, but very instructive. Denoting $\lambda \geq 0$ the dual variable for the nonnegativity constraint, the Lagrangian writes

$$\mathcal{L}(x, \lambda) = \|y - Wx\|_2^2 - \langle \lambda, x \rangle$$

and the optimality conditions are:

- Primal feasibility: $x^* \geq 0$.
- Dual feasibility: $\lambda^* \geq 0$
- Complementary slackness: $\lambda^*[i]x^*[i] = 0$ for all $i \leq n$.
- Fermat rule: $\nabla_x \mathcal{L}(x^*, \lambda^*) = 2W^T(Wx^* - y) - \lambda^* = 0$.

We can refine these conditions by introducing the support S^* of a solution; the support $S(x)$ is the set of indices of the nonzero entries of the vector x. The Fermat rule, combined with complementary slackness, then yields

$$W[:, S^*]^T(W[:, S^*]x[S^*] - y) = -W[:, S^*]^T r^* = 0,$$

with $r = y - W[:, S^*]x[S^*]$ the residual of the projection of y on the cone $\text{col}_+(W)$. When y is in the cone $\text{col}_+(W)$, the residual r is null, and little can be said about the matrix $W[:, S^*]$. However, when $y \notin \text{col}_+(W)$, it must hold that $r \neq 0$. Then any column $W[:, i]$ of matrix W where $i \in S^*$ is in the hyperplane $r^\perp$ of dimension $m - 1$.

Remark

The spark of a matrix W indicates the smallest number of columns from W that are linearly dependent. If the spark of a matrix is larger than n, then any n-sparse solution to linear systems involving W is obtained by the well-defined left pseudo-inverse of W restricted to the support of that solution.

If there is no group of m columns of matrix W that are linearly dependent, that is, if $\text{spark}(W) > m$, then there can be only a single subset of at most $m - 1$ linearly independent columns of matrix W in $r^\perp$. This can happen even if matrix W has more columns than rows; we only require here that no subset of m columns is linearly dependent (and in particular cannot belong to $r^\perp$). With this observation, we can conclude that S^* has size at most $m - 1$, *i.e.*, NNLS solutions, under some conditions, are generally sparse. Moreover, the Fermat rule gives $x^*[S^*] = W[:, S^*]^\dagger y$: knowing the support of the solution is enough to find the solution itself (up to solving a linear system).

Theorem 2 (Night sky theorem [Byrne 1981])

Suppose $y \notin \text{col}_+(W)$ and $\text{spark}(W) > m$. Then the NNLS problem admits a unique solution which has at most $m - 1$ nonzeros.

The nice name for this result, which emphasizes the sparsity of NNLS solutions, was suggested to me by Cédric Herzet and Clément Elvira. While I am unsure of the name's true origin, the result itself is proved in [Byrne, 1981]. The night sky theorem is useful to derive the *active-set algorithm* for NNLS.

Remark

Even if multiple solutions x^* to the NNLS exist, the residual $y - Wx^*$ must always be the same: the orthogonal projection on a convex set has a single solution.

7.2 Nonnegative Kullback-Leibler regression (NNKL)

Sometimes, using the Euclidean norm to measure discrepancies between the measurement vector y and the model Wx is undesirable. There are at least two reasons why the Euclidean norm might be avoided:

- Euclidean norm is sensitive to outliers. More generally, it penalizes large errors more than small errors. In applications such as *music information retrieval*, the data exhibit wide dynamic ranges, and small measurement values are as important as large ones.
- From a statistical estimation point of view, the solution of NNLS is the Maximum Likelihood Estimator (MLE) of variable x when the noise model is Gaussian. For other noise models such as Poisson,

$$y \sim \mathcal{P}(Wx),$$

  the MLE is obtained by the minimization of another cost function.

These two observations are connected: under the Poisson distribution, the larger the expected observation Wx, the larger the noise, but the variance also grows with Wx. This means that small data values are more likely to be accurate than large values (in absolute value, not in relative error).

7.2.1 NNKL problem definition

The MLE for Poisson noise implies the minimization of the KL-divergence between a measurement vector y and some vector $z \in \mathbb{R}^m_+$, defined as the separable function

$$\mathcal{D}_{\mathrm{KL}}(y, z) = \sum_{i=1}^{m} y[i] \log\left(\frac{y[i]}{z[i]}\right) + z[i] - y[i] = \sum_{i=1}^{m} z[i] - y[i]\log(z[i]) + \mathrm{cst}(z),$$

see *below* for the estimator derivation. The nonnegative regression problem with KL-divergence as a loss function, that is coined NNKL in this manuscript, is the optimization problem

$$\text{Find } x \in \operatorname*{argmin}_{x \in \mathbb{R}^n_+} \sum_i \mathcal{D}_{\mathrm{KL}}(y[i], W[i,:]x)$$

where we have set $f(y, Wx) = \sum_i \mathcal{D}_{\mathrm{KL}}(y[i], W[i,:]x)$. Below are two interactive plots showing the graph of KL-divergence in 1D and 2D.

```
# Showing plots for 1d Kullback Leibler with slider for data position
x = np.linspace(1e-5, 10, 300)

def kl_divergence(p, q):
    """Calculate the KL divergence between two distributions."""
    return np.sum(p * np.log(p / q) + q - p)

y = np.zeros_like(x)
p = 2.5  # Initial reference distribution parameter
for i, q in enumerate(x):
    y[i] = kl_divergence(p, q)
```

```
# Data grid
xmax=10
ymax=10
x = np.linspace(1e-5, xmax, 400)
```

(continues on next page)

(continued from previous page)

```
y = np.linspace(1e-5, ymax, 400)
X, Y = np.meshgrid(x, y)

# Initialize Z
Z = np.zeros_like(X)
ref_x, ref_y = 2.5, 2.5
for i in range(X.shape[0]):
    for j in range(X.shape[1]):
        q = np.array([X[i, j], Y[i, j]])
        p = np.array([ref_x, ref_y])
        Z[i, j] = kl_divergence(p, q)
```

7.2.2 Difficulties

Compared to NNLS, which is a quadratic program, NNKL is, in general, significantly more difficult.

Smoothness issue

Maybe the most commented-on difficulty for NNKL is the lack of Lipschitz-smoothness of the KL-divergence at zero. Lipschitz-smoothness is a crucial property in numerical optimization, used to derive descent conditions for first-order algorithms. In a nutshell, a function f which is twice differentiable over a convex set is l-Lipschitz-smooth if its Hessian can be bounded by lI. This implies, using the second-order Taylor expansion and majorizing the second-order terms and remainder [Beck, 2017], that for any vector x and local perturbation z the following descent lemma holds:

$$f(x+z) \leq f(x) + \langle \nabla f(x), z \rangle + \frac{l}{2}\|z\|_2^2.$$

Remark

The true definition of Lipschitz-smoothness and the descent lemma do not require the function f to be twice-differentiable, see, *e.g.*, [Beck, 2017].

Lipschitz-smoothness, therefore, ensures that **globally**, the slope of the function f does not change too fast. Combined with strong convexity arguments, Lipschitz continuity is the key ingredient for proving the convergence of the gradient descent algorithm with fixed stepsize. However, the second-order derivative of KL-divergence is $\frac{\partial^2 \mathcal{D}_{\mathrm{KL}}(y,z)}{\partial z^2} = \frac{y}{z^2}$, which is unbounded near zero. Therefore, solving NNKL with first-order algorithms is challenging because no stepsize selection rule guarantees convergence. In practice, for sparse measurement vectors y, the lack of Lipschitz-smoothness makes many out-of-the-box solvers inefficient for NNKL. Fig. 7.2 provides additional intuition.

Remark

Lipschitz-smoothness allows us to majorize the cost function f by an isotropic quadratic, while strong convexity lower bounds f with another isotropic quadratic. Both arguments together imply that f essentially behaves like an isotropic quadratic, allowing global linear convergence of the gradient descent algorithm for stepsizes smaller than the inverse of the Lipschitz constant l.

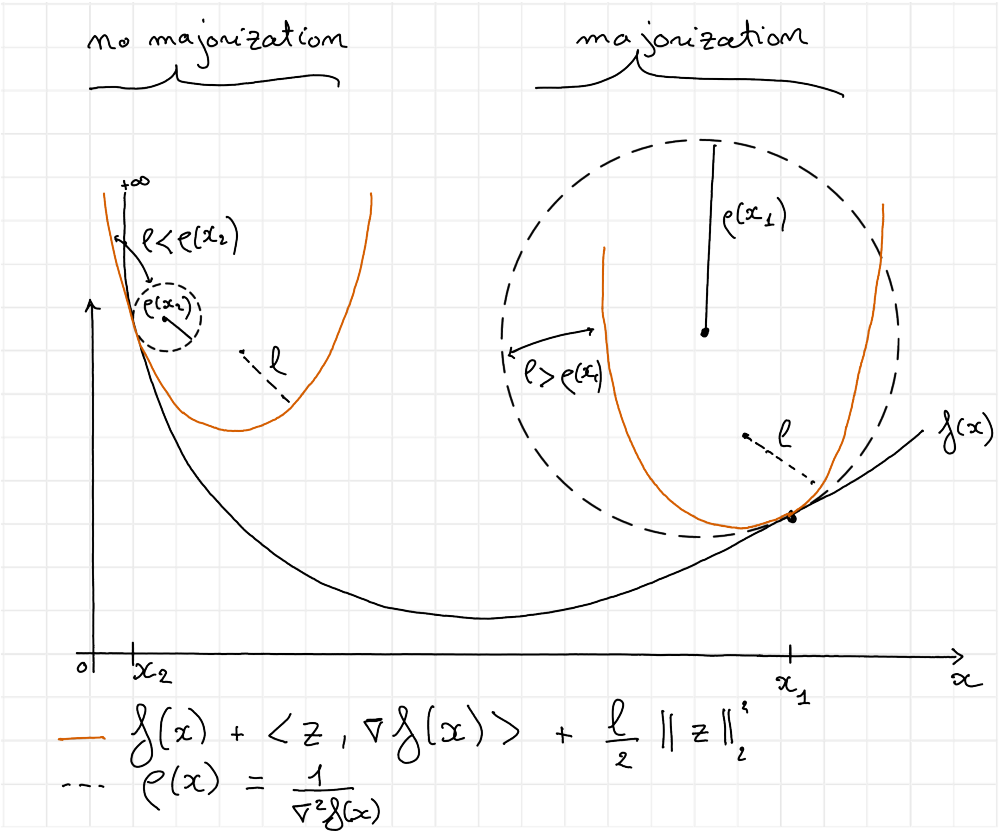


Fig. 7.2: A globally Lipschitz-smooth function can be bounded globally by a local quadratic approximation at any point with a fixed curvature l. In this illustration, the curvature $\rho(x)$ of the cost function f varies from flat (on the right) to steep (on the left). To ensure that a quadratic majorant can be defined everywhere with the same curvature l, because to loss here diverges toward infinity near zero, the curvature must be chosen arbitrarily small. This is therefore an example of a function which is not Lipschitz-smooth. A direct consequence of this lack of smoothness is that it is impossible to define a constant stepsize for a gradient descent algorithm which guarantees that the cost decreases at each iteration. The gradient step, that computes the minimum of a quadratic majorant at each iteration, may reach a value below zero if initialized too far on the right (where the local curvature allows large stepsizes), while it will lead to exploding behavior near zero since the gradient of the cost is arbitrarily large. Gradient descent for KL-divergence is therefore typically employed with tiny stepsizes and strict positivity constraints that yield slow convergence.

Convexity issue

Another important issue with the KL-divergence is that, although it is strictly convex, it is not strongly convex. Indeed, the KL-divergence is asymptotically linear when z is much larger than y. This causes problems in particular when the data measurement y is sparse or exhibits large dynamics. Strong convexity guarantees linear convergence rates for gradient descent; when the cost function is not strongly convex, first-order algorithms might converge sub-linearly [Beck, 2017]. In the extreme case where y has many zeros, a significant part of the cost is linear, and convex first-order optimization techniques are, in general, ill-suited for linear programming.

Remark

KL-divergence also does not satisfy the global Polyak-Lojasiewicz inequality, a more relaxed assumption than strong convexity [Karimi *et al.*, 2016]. It is unclear to me at the time of writing whether KL-divergence satisfies the Kurdyka-Lojasiewicz condition globally, or a local variant of the Polyak-Lojasiewicz condition useful for studying the convergence of first-order methods in nonconvex problems [Attouch and Bolte, 2009, Bolte *et al.*, 2014].

Smoothness and convexity issues combined imply that there may not exist a safe stepsize and a linear convergence rate for gradient descent applied to NNKL. *In the following*, we explore preconditionned first-order algorithms that partially avoid these issues.

Support issue

A third, more subtle issue with NNKL, compared with NNLS, is that even if the support of the solution x^* is known, the solution cannot be computed in closed form. Indeed, the KKT conditions write

- Primal feasibility: $x^* \geq 0$.
- Dual feasibility: $\lambda^* \geq 0$
- Complementary slackness: $\lambda^*[i]x^*[i] = 0$ for all $i \leq n$.
- Fermat rule: $\nabla_x \mathcal{L}(x^*, \lambda^*) = W^T(1 - \frac{y}{Wx^*}) - \lambda^* = 0$.

On the support S of the solution the slack variables λ are null, which yields

$$W[:, S]^T \left(1 - \frac{y}{W[:, S]x[S]^*}\right) = 0.$$

Unlike NNLS, to the best of my knowledge, this system cannot be solved in closed form in general. Interestingly, one may observe that the residual vector $r = 1 - \frac{y}{Wx^*}$, uniquely defined because Wx^* is unique [Tibshirani, 2013], must be sparse. The argumentation is in fact similar to the proof of the night sky theorem: the residual r belongs to the null space of $W[:, S]^T$; together with the spark condition on matrix W, this implies that S is of size at most $m - 1$.

Another observation is that we can rewrite the Fermat rule on the solution support as

$$W[:, S]^T \text{diag}\left(Wx^*\right)^{-1}(y - Wx^*) = 0.$$

This rule is similar to the projection on the cone $\text{col}_+(W)$ found in the KKT conditions for NNLS, with a preconditioning $\text{diag}\left(Wx^*\right)^{-1}$. We may therefore interpret NNKL as a non-orthogonal projection on the cone $\text{col}_+(W)$. If we further assume that $Wx^* \approx y$ and $y > 0$, NNKL is approximately the projection of y on the preconditionned cone $\text{col}_+\left(\text{diag}(y)^{-1}W\right)$. This point of view sheds light on why algorithms for solving NNKL often rely on a combination of preconditioning and first-order methods.

7.2.3 Optimal scaling

A last but important observation on NNKL is that the solution Wx^* must satisfy a simple scaling condition. Indeed, consider the optimal scaling problem

$$\underset{\lambda \geq 0}{\operatorname{argmin}}\ \mathcal{D}_{\mathrm{KL}}\left(y, \lambda W x\right) = -\log(\lambda) \sum_{i=1}^{m} y[i] + \lambda \sum_{i=1}^{m} W[i,:]x + \mathrm{cst}(\lambda)$$

Supposing the unconstrained solution can be positive, the Fermat rule from the KKT conditions gives

$$\lambda^* = \frac{\sum_i y[i]}{\sum_i W[i,:]x},$$

showing that an optimal solution x^* must satisfy $\sum_{i,j} W[i,j]x[j] = \sum_i y[i]$. We find that $\sum_j \left(\sum_i W[i,j]\right) x[j] = \sum_i y[i]$. By scaling the columns of matrix W to sum to one, we find that the marginals of x and y must match. Therefore, an algorithm that solves NNKL can be refined by scaling the initial and output estimates x.

Note: If nonnegativity constraints are dropped and the observation matrix W is right-invertible, the linear system $y = Wx$ has at least one exact solution, and the Euclidean norm or the KL-divergence minimization problems have a common solution $x^* = W^\dagger y$. This observation emphasizes the importance of the nonnegativity constraints in NNLS and NNKL, which act as an informative prior. Solutions to NNLS and NNKL can be arbitrarily far from the right pseudo-inverse and its projection on the nonnegative orthant, see the end of *Proco-ALS for fast nCPD* for an illustration for NNLS.

7.3 Algorithms for NNLS and NNKL

There is a vast literature on optimization algorithms for solving NNLS, NNKL, or both, some of which is included in later parts of this manuscript. Listing all these methods, with their advantages and disadvantages, is an important but tedious task to which I will devote time in the coming years; see the *discussion on research perspectives*. In the rest of this section, only the most useful NNLS and NNKL algorithms for the manuscript are introduced: Active-Set (AS), HALS, and MU.

7.4 Active-set (NNLS only)

A workhorse algorithm to solve NNLS is the AS algorithm proposed by Lawson and Hanson in 1974 [Bro and De Jong, 1997, Lawson and Hanson, 1974]. It is based on the observation that, knowing the support of the solution $S(x^*)$, the solution itself can be computed using the unconstrained least squares estimate restricted to this support. AS therefore iteratively searches for the support of the solution. It works similarly to the OMP algorithm [Pati *et al.*, 1993], where a candidate index is first added to the current estimation of the solution support, and the unconstrained least squares estimate restricted to the current support estimate is then computed. However, unlike greedy sparse approximation algorithms, AS includes a third step that thins the support. Below is a pseudo-code for AS and a simple example. We then describe each step in more detail. We will see that any step taken by the AS algorithm has to decrease the cost function.

```
def get_support(x):
   # returns the support of input vector x
   return list(np.where(x>0)[0])

def get_neg_support(x):
   # returns the support of input vector x
   return list(np.where(x<0)[0])
```

(continues on next page)

(continued from previous page)

```
def activeset(y,W,x):
   # Initialize the support and restricted LS solutions
   supp = get_support(x)
   res = y - W@x
   print(f"Residual: {np.linalg.norm(y - W@x)}, support {supp}")

   # main AS loop, stop when the support does not change
   iter = 0
   old_supp = [-1]  # anything different than initial support
   while supp != old_supp:
      iter += 1
      old_supp = copy(supp)
      print(f"Iteration {iter}")
      # 1. Support update: find the "most negative" Lagrange multiplier
      if len(supp) < len(x):  # if the support is full, do not update the support␣
↪(only happens at init)
         dual_params = W.T@res
         dual_params[supp] = -np.inf # ignore entries corresponding to the current␣
↪support
         idx = np.argmax(dual_params)
         bisect.insort(supp, idx)  # insert support index while keeping sorted

      # 2. Unconstrained Least Squares restricted to the support
      ls_sol = np.linalg.lstsq(W[:,supp], y, rcond=None)[0]
      # embed ls_sol into full space
      ls_sol_full = np.zeros_like(x)
      ls_sol_full[supp] = ls_sol
      neg_supp = get_neg_support(ls_sol_full)

      # 3. Support iterative refinement, to ensure ls_sol is nonnegative
      while len(neg_supp) > 0:
         # Move along the line from the previous to the current candidate estimate,␣
↪until the cone is crossed
         t = np.min(x[neg_supp]/(x[neg_supp]-ls_sol_full[neg_supp]))
         # Update x to decrease the cost along the line (x, ls_sol_full-x), until the␣
↪constraints are active
         x = x + t*(ls_sol_full - x)
         # Update the support (remove one element)
         supp = get_support(x)
         # 2. Recompute LS solution on the new support and count negative entries
         ls_sol = np.linalg.lstsq(W[:,supp], y, rcond=None)[0]
         ls_sol_full = np.zeros_like(x)
         ls_sol_full[supp] = ls_sol
         neg_supp = get_neg_support(ls_sol_full)
         print(f"Residual: {np.linalg.norm(y - W@x)}, support {supp}")

      # Update x with new nonnegative ls_sol
      x = ls_sol_full
      res = y - W@x

      # printing error for following AS progress
      print(f"Residual: {np.linalg.norm(res)}, support {supp}")

   return x, supp
```

Observe how in the following examples, the support grows and shrinks with AS iterations, and the NNLS cost decreases

every time vector x is updated.

```
np.random.seed(2)
m=6
n=50
W = np.random.rand(m,n)
support = [0,2,4]
xtrue = np.zeros(n)
xtrue[support] = np.random.rand(3)
y = W@xtrue + 0.1*np.random.randn(m)
x0 = np.zeros(n)
#x0 = np.random.rand(n)  # also works
x_opt, est_support = activeset(y,W,x0)
print("Optimal x:", x_opt)
print(f"Optimality criterion on estimated support: {np.linalg.norm(W[:,est_support].
 ↪T@(y-W@x_opt))}")
print(f"Optimality criterion on true support: {np.linalg.norm(W[:,support].T@(y-W@x_
 ↪opt))}")
```

```
Residual: 0.7332827438506135, support []
Iteration 1
Residual: 0.34036387671449636, support [49]
Iteration 2
Residual: 0.23756836129567335, support [41, 49]
Iteration 3
Residual: 0.16861002184543528, support [7, 41, 49]
Iteration 4
Residual: 0.15317817929560062, support [3, 7, 41]
Residual: 0.13413207652719195, support [3, 7, 41]
Iteration 5
Residual: 0.10556301170510979, support [3, 4, 7, 41]
Iteration 6
Residual: 0.0519536934584759, support [3, 4, 41, 44]
Residual: 0.04449646221989813, support [4, 41, 44]
Residual: 0.04207535623520433, support [4, 41, 44]
Iteration 7
Residual: 0.04207535623520433, support [4, 41, 44]
Residual: 0.04207535623520433, support [4, 41, 44]
Optimal x: [0.         0.         0.         0.         0.00500251 0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.25668643
 0.         0.         0.45111056 0.         0.         0.
 0.         0.        ]
Optimality criterion on estimated support: 3.122649825233599e-16
Optimality criterion on true support: 0.00401564956638645
```

7.4.1 Initialization of AS

Contrary to what is sometimes assumed, *e.g.*, on the Wikipedia page on NNLS, the AS algorithm can be initialized with any nonnegative vector x. If the initial guess is dense, the first step is skipped on the first iteration.

7.4.2 First step: selection

Assuming the current support of x is not $[1, .., n]$, the selection step finds a "reasonable" index of an entry of x to add to the current estimation of the optimal support. A locally optimal choice is to select the index of the column of W most correlated with the residual $r = y - Wx$ at the current iteration,

$$S \leftarrow S \cup j \quad \text{where} \quad j = \underset{i \notin S}{\operatorname{argmax}} \langle W[:, i], r \rangle.$$

The local optimality is meant in the following sense: the scalar product $\langle W[:, i], r \rangle$ is proportional to the orthogonal projection $\Pi_{W[:,i]}(r)$ of the residual on the line spanned by vector $W[:, i]$:

$$\langle W[:, i], r \rangle = \Pi_{W[:,i]}(r) \|W[:, i]\|_2^2.$$

Another point of view is to observe that in the NNLS KKT conditions, the quantity $W^T r$ must be null on the optimal support and negative outside the support. Therefore, the selection step selects the largest nonzero scalar product and assumes it should be null. The last estimate of the residual is always an orthogonal projection on the span of $W[:, S]$, hence no atom can be selected that is already in the support, or in the span of $W[:, S]$, even without the explicit support constraint. This allows matrix $W[:, S]$ to always be full column rank, except maybe at initialization.

This observation is also discussed in *One-sparse dictionary-based LRA*, but with ℓ_2-normalized atoms.

7.4.3 Second step: restricted least squares regression

The second step of AS is straightforward: one simply computes the unconstrained least squares estimate $z = W[:, S]^\dagger y$, knowing that matrix W is full column rank. Note that at this step, the cost function can only decrease since the support increased in size due to the selection step. If z is nonnegative, we can safely continue the AS algorithm by looping back to step 1 with $x = z$. However, if z is not an admissible solution, the AS algorithm enters a third step.

7.4.4 Third step: constraint satisfaction

AS always ensures that the estimated x at the current iterate is admissible, and that the cost function decreases at each iteration. If z from step 2 is not admissible, the cost will have decreased, but it may be lower than the NNLS global solution. The main idea of AS is then to move the previous estimated x towards z. This move will always decrease the cost because of the strong convexity of the map $t \mapsto \|y - W[:, S](x + t(z - x))\|_2^2$. AS chooses the largest possible scalar t that ensures $x + t(z - x)$ remains feasible, which is easily computed as

$$t^* = \min_{i \text{ such that } z[i]<0} (x[i]/(x[i] - z[i])).$$

By updating x as $x + t^*(z - x)$, at least one zero (and often, exactly one zero) is added in the current estimate x. The cost has strictly decreased (unless $t = 0$, which can be shown to only happen at optimality), and the updated x is still admissible. The support needs, however, to be updated by removing all indices at which the interpolation reaches the cone boundary.

Then the algorithm loops back to step 2. An important remark is that for each support visited by AS, either the unconstrained least squares solution is admissible, and the current cost decreases, or the solution is not admissible and the support is updated while also decreasing the cost. Therefore, the same support can never be visited twice unless the algorithm has reached the global optimum. This shows that the AS algorithm terminates in a non-polynomial but finite number of steps: it can at most visit all the possible supports. When initialized close to the solution, AS can therefore be a very effective algorithm for solving NNLS.

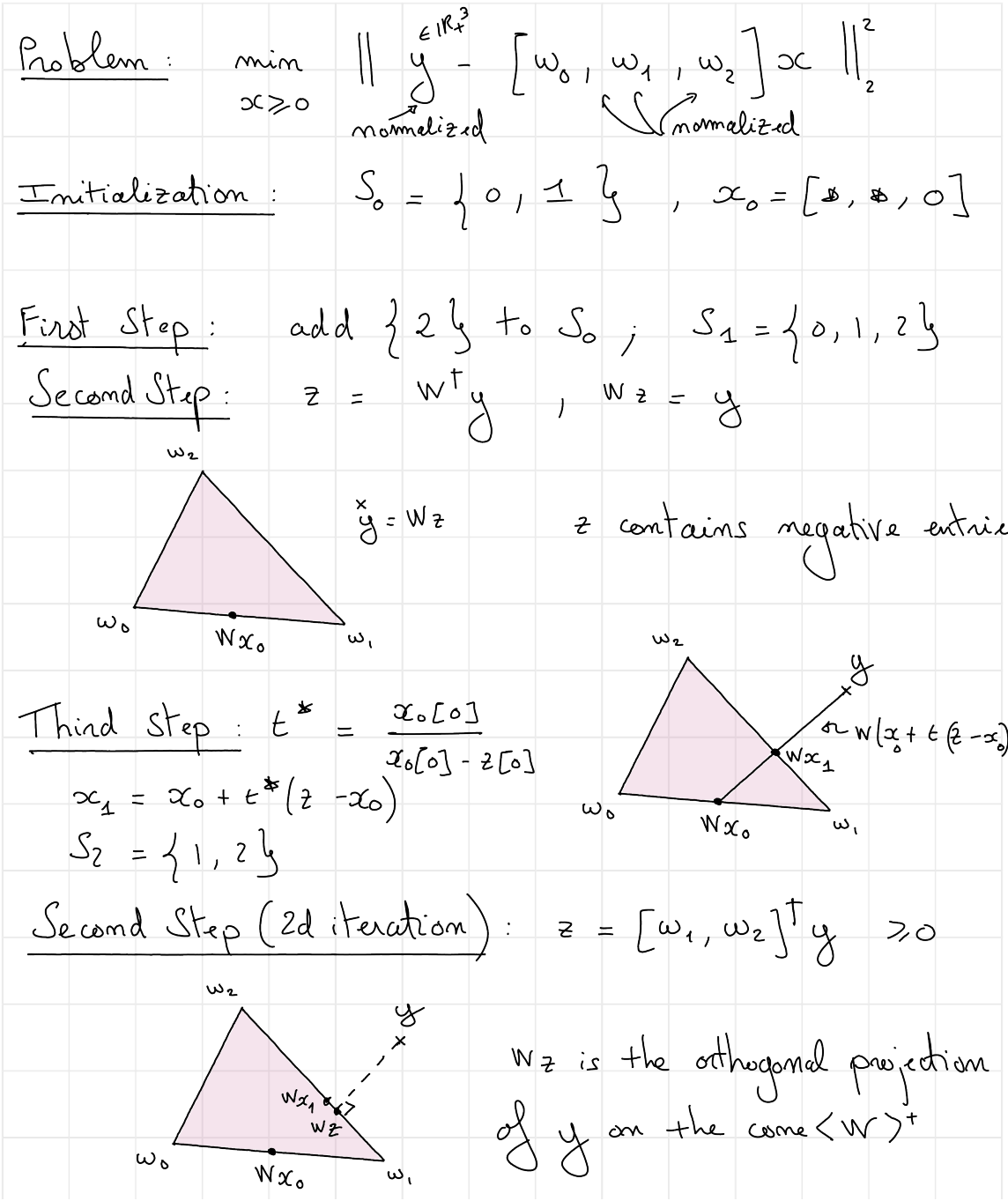


Fig. 7.3: An illustrated example of AS solving a three-dimensional NNLS problem.

7.5 HALS (NNLS only)

The AS algorithm is the workhorse method for efficiently solving NNLS when the data are a one-dimensional vector. However, it is not well-suited for batch computations when solving the matrix NNLS problem of the form

$$\underset{H\geq 0}{\operatorname{argmin}} \|Y - WH^T\|_F^2 = \underset{H\geq 0}{\operatorname{argmin}} \sum_{i\leq n} \|Y[:,i] - WH^T[:,i]\|_2^2.$$

Indeed, the AS algorithm would search for the optimal support of each column of the unknown matrix H^T. These supports can all be different. The costly operation in AS is solving the least squares systems restricted to the current support estimate, and for matrix NNLS, these operations must be performed independently for each column. The fast AS algorithm [Bro and De Jong, 1997] improves on this issue by precomputing the grammians W^TW and W^TY in the case where W is a tall matrix.

For matrix NNLS, it is, however, useful to design an algorithm that can update all columns $H^T[:,i]$ simultaneously. This is the rationale behind the HALS algorithm. HALS updates each row of matrix H sequentially, and this update is known in closed form. Indeed, notice that for any nonzero vector $a \in \mathbb{R}^m$ and any matrix $Z \in \mathbb{R}^{m\times n}$,

$$\underset{h\in\mathbb{R}^n_+}{\operatorname{argmin}} \|Z - ah^T\|_2^2 = \left[\frac{a^TZ}{\|a\|_2^2}\right]^+. \tag{7.1}$$

Indeed, this loss function is separable into n scalar problems

$$\min_{x\geq 0} \|Z[:,i]\|_2^2 - 2w^TZ[:,i]x + \|w\|_2^2x^2$$

for any column index $i \leq n$. The minimum of a scalar quadratic function under nonnegativity constraints is either the global minimum of the quadratic $h[i] = \frac{w^TZ[:,i]}{\|w\|_2^2}$, or, if the global minimiser is negative, zero, see Fig. 7.4.

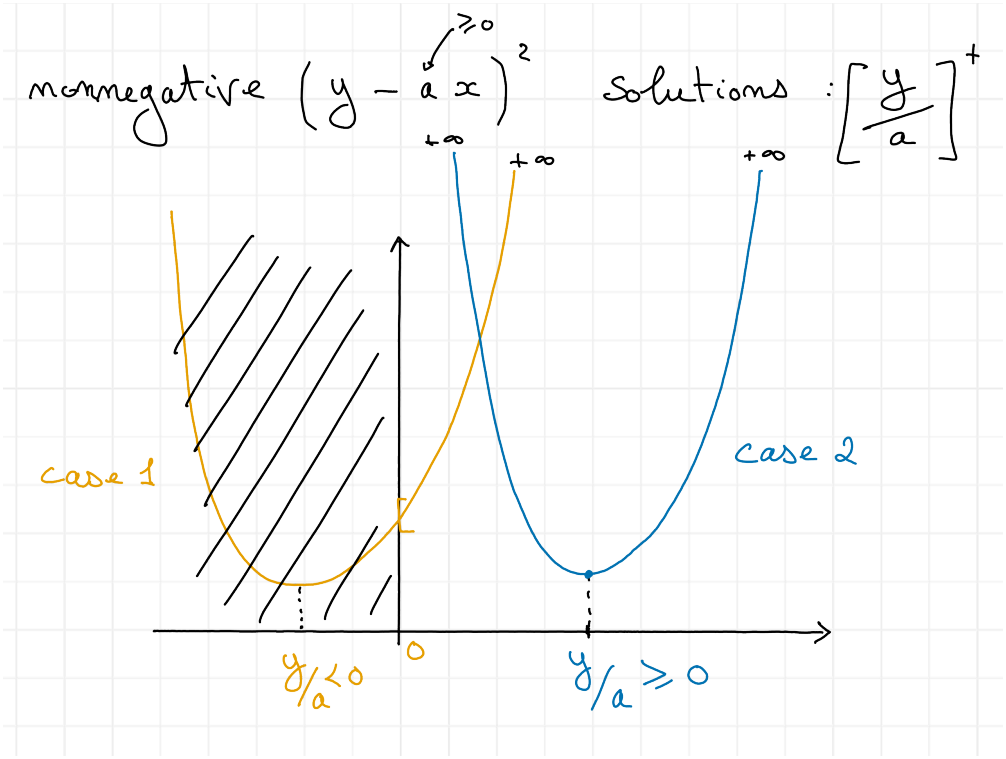


Fig. 7.4: Solutions to nonnegative least squares in 1d with positive terms.

HALS uses this strategy over each row j of matrix H^T, solving iteratively problem (7.1) with $Z = Y - W[:, -j]H^T[-j, :]$ and $w = W[:, j]$. Because each update is an exact minimization procedure, HALS is an instance of exact alternating optimization. As long as each block update is uniquely defined, which happens when W has nonzero columns, convergence to the global minimizer is ensured; see *Alternating optimization*. A simple implementation of HALS is provided below, and is also available in tensorly. The input of the algorithm is the grammians W^TW and W^TY to utilize faster inner products for high-order tensors.

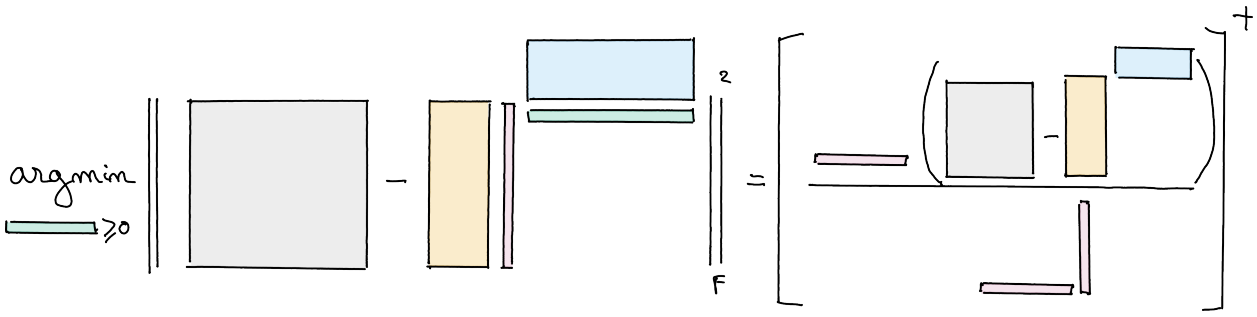


Fig. 7.5: HALS algorithm principle.

```
import tensorly as tl
import matplotlib.pyplot as plt

# Note that HT in the code stands for H^T

def hals_nnls(WtY, WtW, HT=None, n_iter_max=500, tol=1e-8, epsilon=0.0,␣
↪callback=None):
   rank, _ = tl.shape(WtY)
   if HT is None:
      HT = tl.solve(WtW, WtY)
      HT = tl.clip(HT, a_min=0, a_max=None)

   # For error computation, since W and Y are not known directly
   if callback is not None:
      callback(HT)

   for iteration in range(n_iter_max):
      for k in range(rank):
         if WtW[k, k]:
            num = WtY[k, :] - tl.dot(WtW[k, :], HT) + WtW[k, k] * HT[k, :]
            den = WtW[k, k]
            HT[k,:] = tl.clip(num / den, a_min=epsilon)

      if callback is not None:
         callback(HT)

   return HT
```

Here is a small example of HALS running on a synthetic matrix NNLS problem. We also show how to use a callback error function to compute reconstruction errors inside the HALS algorithm.

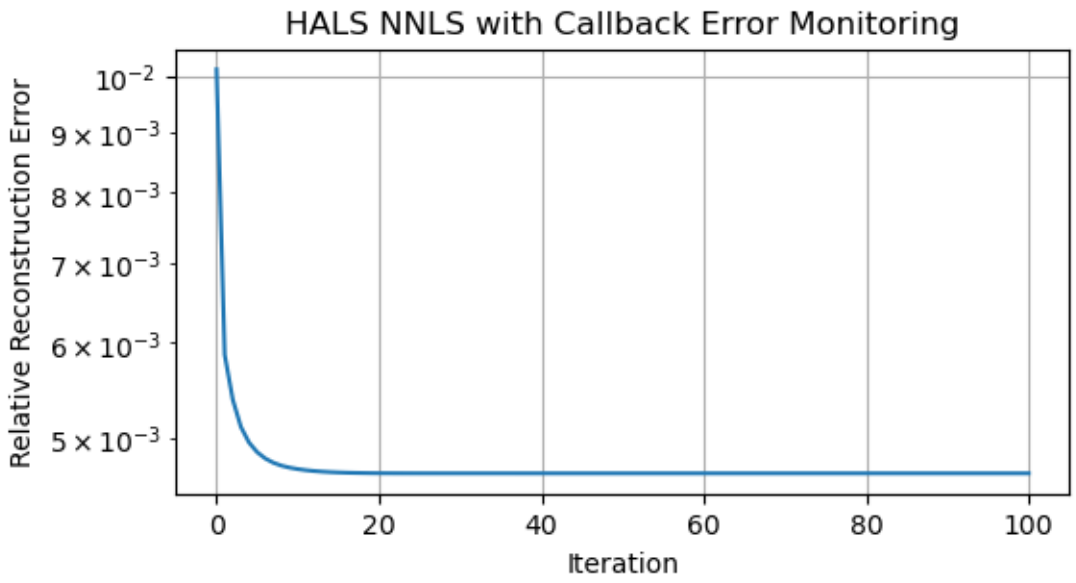
```
np.random.seed(2)
m=10
r = 6
n=20
W = np.random.rand(m,r)
```

(continues on next page)

(continued from previous page)

```
HTtrue = np.random.randn(r,n) # H^T
HTtrue[HTtrue<0] = 0  # sparsify true H
Y = W@HTtrue + 0.01*np.random.randn(m, n)
errs = []
def callback_err(HT):
   errs.append(np.linalg.norm(Y - W@HT)/np.linalg.norm(Y))
   return True
HTest = hals_nnls(W.T@Y, W.T@W, n_iter_max=100, tol=1e-6, callback=callback_err)
print(f"Relative Reconstruction error: {np.linalg.norm(Y - W@HTest)/np.linalg.norm(Y)}
 ↪")
print(f"Root mean squared error on H: {np.linalg.norm(HTtrue - HTest)/np.sqrt(r*n)}")

plt.figure(figsize=(6,3))
plt.semilogy(errs)
plt.xlabel("Iteration")
plt.ylabel("Relative Reconstruction Error")
plt.title("HALS NNLS with Callback Error Monitoring")
plt.grid()
plt.show()
```

```
Relative Reconstruction error: 0.004666660655171395
Root mean squared error on H: 0.010340649769157122
```



7.6 Multiplicative Updates

One of the most well-known iterative solvers for general nonnegative least squares problems (both NNLS and NNKL although it is primarily used for the latter) is the MU algorithm. The history of MU is complex. It has been proposed independently under various names in the literature; the oldest reference of MU that I am aware of is probably the so-called Richardson-Lucy iteration [Lucy, 1974, Richardson, 1972], which is specialized for convolutive models and stems from the computational imaging community. MU is also sometimes referred to as the Maximum-Likelihood-Expectation-Maximization (ML-EM) algorithm [Dempster *et al.*, 1977, Fessler and Hero, 1994] as it can be formulated as a particular case of the generic EM framework. MU was popularized in the source-separation community as an algorithm for solving NMF by Lee and Seung in several seminal papers [Lee and Seung, 1999] and was later generalized by Févotte and Idier for a wider class of loss functions [Févotte and Idier, 2011].

The general equations for the MU are, for the NNLS problem,

$$x^{(k+1)} = x^{(k)} * \frac{W^T y}{W^T W x^{(k)}}, \tag{7.2}$$

where the $*$ symbol in the MU updates denotes the Hadamard (element-wise) product. For the NNKL problem,

$$x^{(k+1)} = x^{(k)} * \frac{W^T \frac{y}{W x^{(k)}}}{W 1_n}. \tag{7.3}$$

These updates can be derived from several frameworks, as already underlined for the quadratic case in Gillis's book on NMF [Gillis, 2020]. However, for NNKL, it is important to note that MU is a special case of the EM algorithm; this fact is known to experts in NMF but not to practitioners in computational imaging, where EM for NNKL was originally derived. Below, I therefore provide a brief explanation of how to obtain MU from these various frameworks and the implications of these derivations.

Remark

It is, in fact, interesting to note that the Richardson-Lucy/ML-EM algorithm has seen many developments and improvements over the years in parallel with developments proposed in the source separation/numerical optimization literature. A few interesting reads are a constrained version of Richardson-Lucy called one-step late [Green, 1990] that amounts to MM with linearized regularizations, the general formulation of EM for likelihoods in the exponential family as a mirror gradient descent which allows to derive a generic linear convergence rate [Kunstner *et al.*, 2021] and the design of TV-regularized and Plug-and-Play variants with convergence guarantees (work under way in the team [Modrzyk *et al.*, 2025]).

7.6.1 As a gradient ratio

A simple way to obtain MU is to note that the multiplicative term is the ratio of the negative to the positive parts of the gradient. The gradients are easily computed as

$$\nabla_x f\left(y, Wx\right) = \begin{cases} 2W^T W x - 2W^T y & \text{for NNLS} \\ W^T 1_n - W^T \frac{y}{Wx} & \text{for NNKL .} \end{cases}$$

Note that the terms $W^T W x, W^T y, W^T \frac{y}{Wx}$ and $W^T 1_n$ are all nonnegative. Denoting ∇_x^+ the positive terms in the gradient computation and similarly for ∇_x^-, we may write rewrite the MU updates as

$$x^{(k+1)} = x^{(k)} * \frac{\nabla_x^- f\left(y, x\right)}{\nabla_x^+ f\left(y, x\right)}.$$

This interpretation of MU is, in my opinion, useful mostly to quickly remember the updates. While Gillis [Gillis, 2020] provides an interpretation of the gradient ratio related to the KKT conditions (namely, the ratio should get closer to one to satisfy the KKT conditions), deriving convergence results from this formulation is not straightforward. It is also unclear how this update rule will behave for regularized NNLS/NNKL problems.

7.6.2 As a preconditioned gradient descent algorithm

MU for NNLS can be cast exactly as a preconditioned descent algorithm with a diagonal preconditioner designed to upper-bound the true Hessian. The full description of this formulation is deferred to the *second part of the manuscript* since it deeply connects with a contribution regarding fast algorithms for NNLS and NNKL.

7.6.3 As a majorization-minimization algorithm

The MU algorithm was introduced in the source separation and machine learning communities by Lee and Seung in 1999 [Lee and Seung, 1999] as a majorization minimization algorithm. We follow in this paragraph the derivations of Févotte and Idier [Févotte and Idier, 2011] that work for the more general class of β-divergences. For simplicity, we restrict the presentation to the case of a convex loss function for $\beta \in [1, 2]$. We show in the next paragraph that, for the particular case of KL-divergence, the Majorization-Minimization (MM) derivations fall within the EM framework.

Remark

β-divergences are a family of separable divergences that contain, in particular, the Euclidean distance ($\beta = 2$) and the (symmetrized) KL-divergence ($\beta = 1$). The general formula, prolonged by continuity for β in $\{0, 1\}$ is given by

$$\mathcal{D}_\beta = \frac{1}{\beta(\beta-1)} \left(x^\beta + (\beta-1)y^\beta - \beta x y^{\beta-1}\right).$$

The main idea of MM is to fix a current iterate $x^{(k)}$, build a global majorant of the cost $\xi(x, x^{(k)}) \leq f(y, x)$ tight and tangent to the cost at $x^{(k)}$, and then minimize this cost.

Remark

MM is often introduced without requiring that the tangent of the cost and the majorant be equal; see also *Alternating optimization*.

To obtain the MU algorithm, one may use the convexity inequality for the loss function, also called the Jensen inequality in this context:

$$\sum_{i \leq m} f\left(y[i], \sum_{j \leq n} \lambda_{i,j} W[i,j]x[j]\right) \leq \sum_{i \leq m,\, j \leq n} \lambda_{i,j} f\left(y[i], W[i,j]x[j]\right)$$

for nonnegative coefficients λ_i that sum to one and any positive vector $x \in \mathbb{R}^n_+$. The trick to obtaining MU using MM is to choose the specific values

$$\lambda_{i,j} = \frac{W[i,j]x^{(k)}[j]}{\sum_{j \leq n} W[i,j]x^{(k)}[j]} = \frac{W[i,j]x^{(k)}[j]}{\hat{y}[i]}.$$

The estimated observations at the current iterate $\hat{y} = Wx^{(k)}$ have been introduced to simplify the presentation. The lambda parameters are introduced in the summand of the second argument of the loss, setting $\sum_j W[i,j]x[i,j] = \sum_j W[i,j]x[j]\frac{\lambda_{i,j}}{\lambda_{i,j}}$. Applying the convexity inequality after observing that $\frac{W[i,j]}{\lambda_{i,j}} = \frac{\hat{y}[i]}{x^{(k)}[j]}$ yields a majorant of the cost

$$\xi(x, x^{(k)}) = \sum_{i \leq m,\, j \leq n} \frac{W[i,j]x^{(k)}[j]}{\hat{y}[i]} f\left(y[i], \frac{x[j]}{x^{(k)}[j]}\hat{y}[i]\right).$$

This majorant is separable and convex, so we can find its minimizer by setting the derivative with respect to each $x[j]$ to zero. Denoting $\frac{\partial}{\partial x_2} f$ the partial derivative of the function f with respect to its second argument, the MM algorithm

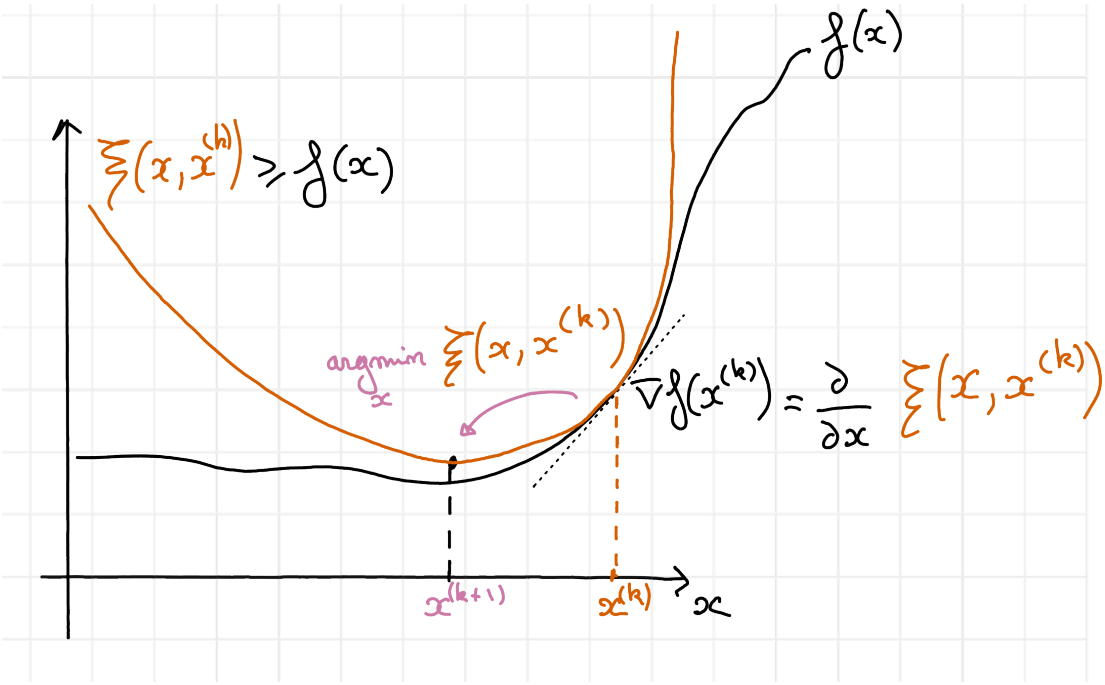


Fig. 7.6: MM is an algorithmic framework that succesively majorizes a cost function f globally, and then minimizes the majorant. In this figure, the majorant $\xi(x, x^{(k)})$, in red, is tight and tangent at the point $x^{(k)}$. Its minimum is attained at $x^{(k+1)}$, in purple.

updates x by solving

$$\forall j \leq n, \quad 0 = \sum_{i \leq m} W[i,j] \frac{\partial}{\partial x_2} f\left(y[i], \frac{x[j]}{x^{(k)}[j]} \hat{y}[i]\right). \tag{7.4}$$

For the Euclidean loss and the KL-divergence, respectively,

$$\frac{\partial}{\partial x_2} f\left(y[i], \frac{x[j]}{x^{(k)}[j]} \hat{y}[i]\right) = \frac{x[j]}{x^{(k)}[j]} \hat{y}[i] - y[i],$$
$$\frac{\partial}{\partial x_2} f\left(y[i], \frac{x[j]}{x^{(k)}[j]} \hat{y}[i]\right) = 1 - \frac{y[i] x^{(k)}[j]}{\hat{y}[i] x[j]},$$

both of which, when plugged into the stationary point equation (7.4), boil down to the MU updates (7.2) and (7.3).

The MM interpretation of MU is useful because it automatically guarantees that MU iterations always decrease the cost. Moreover, for NNLS with positive initialization, MU falls within the scope of the *SUM framework*, which guarantees convergence of the cost to a stationary point. However, additional hypotheses are required to guarantee that the limit point of the cost and the iterates are respectively stationary points and global minimizers of the cost function in the NNKL problem. These assumptions are summarized in [Gillis, 2020], and revolve around the fact that Lipschitz continuity of the cost is required to avoid arbitrarily small improvements of the iterates for a given cost decrease.

Positivity constraints

A workaround for the NNKL problem, that also guarantees the positivity required for convergence in NNLS, is to impose positivity constraints by introducing $\epsilon > 0$ such that $x \geq \epsilon$. Positivity avoids the non-Lipschitz smoothness of the KL-divergence at zero, and avoids the zero-locking phenomenon [Takahashi and Hibi, 2014]. The MU iterates

become

$$x^{(k+1)} = \max\left(x^{(k)} * \frac{W^T y}{W^T W x^{(k)}}, \epsilon\right),$$

for the NNLS problem, and

$$x^{(k+1)} = \max\left(x^{(k)} * \frac{W^T \frac{y}{Wx^{(k)}}}{W^T 1_n}, \epsilon\right).$$

for the NNKL problem, where the maximum is applied elementwise. The underlying algorithm here can be thought of as a generalized proximal gradient or preconditioned forward-backward algorithm; see, for instance, the variable metric forward-backward framework for more details [Chouzenoux *et al.*, 2014, Chouzenoux *et al.*, 2016].

Remark

The zero-locking phenomenon is a numerical instability that occurs when the entries of the vector x are numerically so close to zero that the computer stores them as actual zeros. Once a value in x is zero, it can never increase again in MU due to the elementwise multiplications at each iteration. This can prevent convergence in practice and lead to numerical instabilities due to division by zero.

7.6.4 As an EM algorithm

In the MM description of MU, the particular choice for the parameters $\lambda_{i,j}$, which is crucial to obtain the updates, could seem somewhat arbitrary. It turns out that using the EM framework for the NNKL problem, one can make sense of this choice. Below, we derive MU from EM in a way that slightly differs from earlier references from computational tomography [Dempster *et al.*, 1977] that I find personally hard to follow.

A statistical description of the NNKL problem is required to write the EM algorithm. We suppose that the data samples $y[i]$ are sampled independently from random variables $Y[i]$ following Poisson distributions conditionally to the knowledge of unknown parameters x, that is

$$\forall i \leq m, \quad Y[i] \mid x \sim \mathcal{P}\left(\sum_{j \leq n} W[i,j]x[j]\right).$$

The MLE in this setting finds the parameters x that maximize (minimize) the (negative) log-probability of the observations y,

$$\operatorname*{argmin}_{x \in \mathbb{R}^n} \; -\sum_{i \leq m} \log p(Y[i] = y[i] \mid x),$$

where the summation over i is due to the independence of the random variables in Y. For the Poisson distribution, one may observe that the log-likelihood is the KL-divergence up to constant terms with respect to x:

$$\log p(Y[i] = y[i] | x[i]) = y[i] \log \sum_{j \leq n} W[i,j]x[j] - \sum_{j \leq n} W[i,j]x[j] - \log(y[i]!)\,.$$

Therefore, computing the MLE amounts to solving the NNKL problem.

Basics of the EM algorithm

EM is a particular case of a majorization-minimization algorithm applied to a log-likelihood. There are several ways to understand EM besides the usual textbook presentation (including MM, proximal point algorithms, alternating optimization); see, for instance, the discussion in [Kunstner *et al.*, 2021]. The MM point of view connects EM with other optimization algorithms quite naturally; let us derive EM within the MM framework.

Remark

The EM algorithm is presented here with discrete probabilities for simplicity, but the general formulation for continuous probability densities is obtained in the same fashion.

EM can be obtained by combining two techniques: marginalization with respect to user-chosen latent variables and Jensen's/convexity inequality (similar to the derivations of MU within the MM framework) to approximate the cost. Introducing latent random variable(s) Z taking values z in the set $\mathcal{Z}$ and the observation vector y, using the total probability law, the log-likelihood writes

$$\log p(y \mid x) = \log \sum_{z \in \mathcal{Z}} p(y, z \mid x).$$

These latent variables can represent, for instance, intermediate values in the optimization problem (see the application of EM to NNKL below), or missing observations. A core idea of EM (and more generally variational Bayesian estimation) is to leverage Jensen's inequality to build a lower bound of $\log p(y \mid x)$ by introducing another well-chosen distribution. EM is an iterative algorithm in which, at iteration k, this distribution is chosen as $p(z \mid y, x^{(k)})$, which is, in essence, the distribution of the latent variables given the observed data and the current estimates of the unknown parameters. In many cases, this conditional distribution can be derived from the statistical description of the problem. More precisely, the log-likelihood is lower-bounded as follows:

$$\begin{aligned}
\log p(y \mid x) &= \log \sum_{z \in \mathcal{Z}} p(y, z \mid x) \frac{p(z \mid y, x^{(k)})}{p(z \mid y, x^{(k)})}, \\
&\geq \sum_{z \in \mathcal{Z}} p(z \mid y, x^{(k)}) \log \frac{p(y, z \mid x)}{p(z \mid y, x^{(k)})}, \\
&\geq \mathbb{E}_{Z|y,x^k} \left[\log p(y, z \mid x)\right] - \underbrace{\mathbb{E}_{Z|y,x^k} \left[\log p(z \mid y, x^{(k)})\right]}_{\text{constant w.r.t. } x}.
\end{aligned}$$

One can also easily show that this lower bound is tight for $x = x^{(k)}$ and that the first-order derivatives match. To minimize the negative log-likelihood, the EM algorithm therefore minimizes $\mathbb{E}_{z|y,x^k} \left[-\log p(y, z \mid x)\right]$ with respect to x. There are again many other equivalent formulations of the functional minimized by EM. One that works well for source separation problems is to introduce back the conditional likelihood and prior information on the latent variables:

$$\xi(x, x^{(k)}) = \mathbb{E}_{Z|y,x^k} \left[-\log p(y \mid z, x) - \log p(z \mid x)\right]. \tag{7.5}$$

Thus far, the EM algorithm is rather high-level and abstract. One personal difficulty in applying EM has always been to make sense of the probabilities and the conditional expectation in (7.5). For some problems, deriving these quantities can in fact be very challenging. Hopefully, this is not the case for NNKL, and the EM algorithm is derived in the next section.

Relationship between EM, MU, and mirror descent

Bauschke, Bolte, and Teboulle have proposed to solve NNKL using (proximal) mirror descent [Bauschke *et al.*, 2017]. They make use of a particular choice of potential, Burg's entropy $-\sum_i \log(x[i])$, and show that KL-divergence is smooth relative to this choice with constant $\frac{1}{\|y\|_1}$. In other words, KL-divergence can be upper-bounded using a first-order approximation and the Bregman divergence $\mathcal{D}_h$ obtained with Burg's entropy

$\mathcal{D}_{\mathrm{KL}}\left(y, Wx\right) \leq \mathcal{D}_{\mathrm{KL}}\left(y, Wx^{(k)}\right) + \langle \nabla_x \mathcal{D}_{\mathrm{KL}}\left(y, Wx^{(k)}\right), x - x^{(k)} \rangle + \frac{1}{2\|y\|_1} \mathcal{D}_h(x, x^{(k)})$,

that yields, after simple derivations found in [Bauschke *et al.*, 2017]

$\mathcal{D}_{\mathrm{KL}}\left(y, Wx\right) \leq \mathcal{D}_{\mathrm{KL}}\left(y, Wx^{(k)}\right) + \langle W^T\left(1_n - \frac{y}{Wx^{(k)}}\right), x - x^{(k)} \rangle + \frac{1}{2\|y\|_1} \sum_{j \leq n} \frac{x[j] - x^{(k)}[j]}{x^{(k)}[j]} - \log \frac{x[j] - x^{(k)}[j]}{x^{(k)}[j]}$.

Minimizing this upper-bound leads to unusual multiplicative updates

$x = x * \frac{1}{1 + \frac{1}{\|y\|_1} x \odot (W^T 1_n - W^T \frac{y}{Wx})}$.

Since these updates allow the use of (Bregman) proximal operators while ensuring convergence, they have been used in regularized NNKL problems [Hurault *et al.*, 2023]. However, it is unclear how these updates compare to the classical MU (7.3) and its convergence rate is unknown.

Recently, a formal link was established between the EM algorithm for the exponential family [Kunstner *et al.*, 2021] and mirror descent with Bregman divergences. It allows, in particular, the derivation of linear convergence rates for EM, and therefore for the usual MU in the NNKL problem. This also shows that other choices of potential, besides Burg entropy, lead to interesting update rules for NNKL. This was brought to my attention by Thibaut Modrzyk, who is currently investigating this observation. The relationship between MU and mirror gradient descent was also partially discussed in [Hien and Gillis, 2021] . Linear convergence of NNKL explains why the MU algorithm is an efficient method for solving NNKL.

MU is an instance of EM for NNKL

To apply EM to NNKL, on top of introducing the Poisson distribution on the observed variables $Y[i]$, we introduce latent variables that will allow us to derive the EM algorithm efficiently. The literature [Dempster *et al.*, 1977, Shepp and Vardi, 1982] informs us to define independent latent variables

$$Z[i, j] \sim \mathcal{P}(W[i, j]x[j]),$$

such that $\sum_{j \leq n} Z[i, j] = Y[i]$. In source separation, random variables $Z[i, j]$ model (parts of) the various components in an additive mixture and therefore bear physical meaning.

Remark

I denote with small letters, *e.g.* $z[i, j]$, the values taken by the random variables inside the probability function, and by capital letters the random variables. The probability $p(z[i, j])$ should be read as $p(Z[i, j] = z[i, j])$.

We can now expand the upper-bound ξ. First notice that the prior distribution of the latent variables $p(z \mid x)$ is separable into the product $\prod_{i,j} p(z[i, j] \mid x[j])$ since all latent variables are mutually independent, and the random variable $Z[i, j]$ depends only on the parameter $x[j]$. Therefore,

$$\xi(x, x^{(k)}) = \sum_{i \leq m} \mathbb{E}_{Z[i,:]|y[i],x^k} \left[-\log p(y[i] \mid z[i, :], x) - \sum_{j \leq n} \log p(z[i, j] \mid x[j]) \right].$$

The conditional likelihood $p(y[i] \mid z[i, :], x)$ has a particular shape. Because we defined the latent variables Z such that $Y[i] = \sum_j Z[i, j]$, conditionned on Z, Y is deterministic. To simplify, we thereafter treat this first term in the majorant as a constraint imposing that the latent variables indeed sum up to the observations, $y[i] = \sum_j Z[i, j]$ (otherwise the logarithm goes to $-\infty$), and ignore it in the definition of the cost. Therefore,

$$\begin{aligned} \xi(x, x^{(k)}) &= \sum_{i \leq m, j \leq n} \mathbb{E}_{Z[i,:]|y[i],x^k} \left[-\log p(z[i, j] \mid x[j]) \right] \text{ such that } \forall i \leq m, \ \sum_j Z[i, j] = y[i] \\ &= \sum_{i \leq m, j \leq n} \mathbb{E}_{Z[i,:]|\sum_j Z[i,j],x^k} \left[-\log p(z[i, j] \mid x[j]) \right]. \end{aligned}$$

After expending the priors,

$$-\log p(z[i,j] \mid x[j]) = W[i,j]x[j] - z[i,j]\log W[i,j]x[j] + \log z[i,j]!,$$

the upper-bound simplifies into

$$\xi(x, x^{(k)}) = \sum_{i\leq m, j\leq n} W[i,j]x[j] - \mathbb{E}_{Z[i,:]|y[i],x^k}\left[z[i,j]\right]\log W[i,j]x[j] + \mathbb{E}_{Z[i,:]|y[i],x^k}[\log z[i,j]!].$$

The last term involving the factorial of the latent variables is constant with respect to the parameters x and can be ignored. What remains to compute to obtain a simpler form for the majorant ξ is the expected value of $Z[i,j]$ conditioned to the knowledge of the sum of other Poisson variables $\sum_j Z[i,j]$ and the parameter values x^k of the Poisson laws, $\mathbb{E}_{Z[i,:]|y[i],x^k}\left[z[i,j]\right]$ with $y[i] = \sum_j Z[i,j]$.

The following Lemma allows us to conclude, its proof is simple [van Oosten, 2014].

Lemma 1 (Distribution of Poisson variables conditionned by their sum)

Let Z_1, Z_2 be two random variables distributed respectively as $\mathcal{P}(\lambda_1)$ and $\mathcal{P}(\lambda_2)$. Then the conditionned random variable $Z_1|Z_1+Z_2$ follows a Binomial distribution $\text{Bin}(Z_1+Z_2, \frac{\lambda_1}{\lambda_1+\lambda_2})$.

Applying *Lemma 1* with $Z_1 = Z[i,j]$ and $Z_2 = y[i] - Z[i,j]$ yields

$$\mathbb{E}_{Z[i,:]|y[i],x^k}\left[z[i,j]\right] = y[i]\frac{W[i,j]x^{(k)}[j]}{\sum_j W[i,j]x[j]} = y[i]\frac{W[i,j]x^{(k)}[j]}{\widehat{y}[i]}.$$

Notice that we recover the weights of the MM formulation of MU, $y[i]\lambda_{i,j} = \mathbb{E}_{Z[i,:]|y[i],x^k}\left[z[i,j]\right]$, which provides a nice interpretation of the ad-hoc choice for these parameters in the MM framework for NNKL.

We now observe that the majorant ξ is equal up to constant terms to the majorant obtained with the MM framework, and the gradients of the two convex separable majorants match:

$$\xi(x, x^{(k)}) = \sum_{i\leq m, j\leq n} W[i,j]x[j] - y[i]\frac{W[i,j]x^{(k)}[j]}{\widehat{y}[i]}.\log x[j] + \text{cst}(x),$$

$$\frac{\partial}{\partial x[j]}\xi(x, x^{(k)}) = \sum_{i\leq m} W[i,j] - \frac{x^{(k)}[j]}{x[j]}\sum_{i\leq m}\frac{W[i,j]y[i]}{\widehat{y}[i]}.$$

Therefore, the EM algorithm for NNKL is exactly MU.

MU may not be an instance of EM for NNLS

Importantly, while the EM algorithm can in principle be used to derive update rules for the NNLS problem given reasonable additional statistical assumptions on the variance of the variables $z[i]$, the update rules obtained with these assumptions on distributions do not match the MU updates (7.2). Instead, they are another particular case of diagonally preconditioned gradient descent; see, for instance, [van Oosten, 2014].

Equivalence of the MU formulations breaks with l1 and l2 regularizations

If additive ℓ_1 and ℓ_2 regularizations terms $\lambda_1\|x\|_1 + \frac{1}{2}\lambda_2\|x\|_2^2$ are added to the cost, the MU updates using the gradient ratio for both NNLS and NNKL would write

$$x^{(k+1)} = x^{(k)} * \frac{\nabla_x^- f\left(y, x\right)}{\nabla_x^+ f\left(y, x\right) + \lambda_1 + \lambda_2 x^{(k)}}.$$

since the gradients of the regularizations are always positive. For instance, for NNKL, MU obtained from the gradient ratio heuristic is

$$x^{(k+1)} = x^{(k)} * \frac{W^T \frac{y}{Wx^{(k)}}}{W1_n + \lambda_1 + \lambda_2 x^{(k)}}.$$

However, using the MM framework, the impact of the regularizations on the update rules changes for NNLS and NNKL. Note that a more complete analysis is provided in [Leplat *et al.*, 2021]. The updates within the MM framework are given by solving

$$\forall j \leq n, \quad 0 = \sum_{i \leq m} W[i,j] \frac{\partial}{\partial x_2} f\left(y[i], \frac{x[j]}{x^{(k)}[j]} \hat{y}[i]\right) + \lambda_1 + \lambda_2 x[j].$$

Injecting the gradients of the cost function yields

$$\text{for NNLS:} \quad \forall j \leq n, \quad 0 = \sum_{i \leq m} W[i,j] \left(\frac{x[j]}{x^{(k)}[j]} \hat{y}[i] - y[i]\right) + \lambda_1 + \lambda_2 x[j],$$

$$\text{for NNKL:} \quad \forall j \leq n, \quad 0 = \sum_{i \leq m} W[i,j] \left(\frac{y[i] x^{(k)}[j]}{\hat{y}[i] x[j]} - 1\right) + \lambda_1 + \lambda_2 x[j].$$

and the resulting MU updates for the NNLS problem are

$$x^{(k+1)} = \max\left(x^{(k)} * \frac{W^T y - \lambda_1}{W^T W x^{(k)} + \lambda_2}, \epsilon\right).$$

Note the difference with the gradient ratio heuristic that places the ℓ_1 regularization hyperparameter in the denominator. Moreover, these MU updates with $\lambda_1 \geq 0$ can in principle become negative or zero; therefore, positivity constraints should be enforced as discussed *above*.

For the NNKL problem, one needs to compute the positive roots of the second-order polynomials

$$\lambda_2 x[j]^2 + \left(\sum_{i \leq m} W[i,j] + \lambda_1\right) x[j] + x^{(k)}[j] \sum_{i \leq m} W[i,j] \frac{y[i]}{\hat{y}[i]}.$$

7.7 Best nonnegative rank-one approximations

We have mainly focused in this section on solving nonnegative regression problems. We will discuss algorithms for LRA throughout the rest of this manuscript, with a particular focus on alternating algorithms in *Alternating optimization*. However, there exist closed-form algorithms for the best rank-one nonnegative approximation, related to the discussion of NNLS and NNKL.

7.7.1 KL-divergence NMF

Consider the rank-one NMF approximation problem in the KL-divergence

$$\underset{w\in\mathbb{R}_+^m,\, h\in\mathbb{R}_+^n}{\operatorname{argmin}} \; \mathcal{D}_{\mathrm{KL}}\left(Y, wh^T\right).$$

The *optimal scaling condition* imposes that

$$h^*[j]\sum_{i=1}^m w^*[i] = \sum_{i=1}^m Y[i,j] \quad \forall j \leq n,$$
$$w^*[i]\sum_{j=1}^n h^*[j] = \sum_{j=1}^n Y[i,j] \quad \forall i \leq m.$$

We immediately deduce that solutions h^* and w^* are proportional to the marginals of the observation matrix Y. If we further assume that both vectors have the same scales, the closed-form solution writes

$$h^* = \frac{\sum_{i=1}^m Y[i,:]}{\sqrt{\sum_{i,j} Y[i,j]}},$$
$$w^* = \frac{\sum_{j=1}^n Y[:,j]}{\sqrt{\sum_{i,j} Y[i,j]}}.$$

In other words, the best nonnegative rank-one approximation in the KL-divergence is known in closed form, and its computation is straightforward. This result has probably been known for a long time, as it is related to the Sinkhorn algorithm [Sinkhorn and Knopp, 1967], to the estimation of coupled probabilities, and to information-geometric concepts [Ghalamkari *et al.*, 2025]. It has been introduced in the NMF community several times, in [Ho and Van Dooren, 2008] and in [Huang and Sidiropoulos, 2017]. The analogy with the scaling problem and the Sinkhorn algorithm in particular can be seen by noticing that

$$wh^T = \mathrm{diag}(w)1_{m\times n}\mathrm{diag}(h)$$

and that the Sinkhorn algorithm can be used to solve the optimization problem

$$\underset{w\in\mathbb{R}_+^m,\, h\in\mathbb{R}_+^n}{\operatorname{argmin}} \; \mathcal{D}_{\mathrm{KL}}\left(Y, \mathrm{diag}(w)X\mathrm{diag}(h)\right),$$

that is, scaling a matrix X to have the same marginals as another given matrix Y. In general, this problem has no closed-form solution.

7.7.2 Frobenius norm NMF

Computing the best rank-one approximation in the Frobenius norm, that is, solving

$$\underset{w\in\mathbb{R}_+^m,\, h\in\mathbb{R}_+^n}{\operatorname{argmin}} \; \|Y - wh^T\|_F^2$$

in general is NP-hard [Gillis and Glineur, 2014]. However, for nonnegative data $Y \in \mathbb{R}_+^{m\times n}$, the solution can be computed efficiently (in particular, in polynomial time) using a rank-one Singular Value Decomposition. Indeed, for positive matrices, the Perron-Frobenius theorem guarantees that the first singular vectors and the first singular value are always positive. If the data has zero entries, the Perron-Frobenius theorem does not hold in general, but it has been shown that the solution can still be obtained using the absolute values of the first components. Indeed, following [Gillis, 2020]:

$$\|X - w_{\mathrm{svd}}h_{\mathrm{svd}}^T\|_F^2 \geq \|X - |w_{\mathrm{svd}}||h_{\mathrm{svd}}|^T\|_F^2,$$

(this relationship holds for any couple (w,h) because the data matrix X is elementwise nonnegative), and by optimality of the SVD, for any couple (w,h) (in particular with nonnegative entries),

$$\|X - wh^T\|_F^2 \geq \|X - w_{\mathrm{svd}}h_{\mathrm{svd}}^T\|_F^2 \geq \|X - |w_{\mathrm{svd}}||h_{\mathrm{svd}}|^T\|_F^2.$$

We have therefore found an admissible solution with the lowest possible cost.

CHAPTER

EIGHT

ALTERNATING OPTIMIZATION

Many estimation problems considered in the manuscript rely on solving an optimization problem that involves d different blocks, of the form

$$\underset{x_1 \in \mathcal{X}_1, \, \ldots \, , x_d \in \mathcal{X}_d}{\operatorname{argmin}} \; f(x_1, x_2, \ldots, x_d)$$

where $f : \mathcal{X}_1 \times \ldots \times \mathcal{X}_d \mapsto \mathbb{R}_+$ is a nonnegative cost function and $\mathcal{X}_k$ are Euclidean spaces embedded in $\mathbb{R}^{n_k}$.

Remark

We assume the cost function f is nonnegative for simplicity and coherence with the rest of the manuscript. However it is sufficient to assume that f is lower-bounded so that coercivity and continuity imply the existence of a minimizer.

Indeed, LRA models are multifactor: each block x_k may represent a factor matrix in a low-rank model. For instance, solving a rank r approximate NMF problem in the presence of Gaussian noise may result in the following two-block optimization problem

$$\underset{X_1 \in \mathbb{R}_+^{n_1 \times r}, \; X_2 \in \mathbb{R}_+^{n_2 \times r}}{\operatorname{argmin}} \; \|Y - X_1 X_2^T\|_F^2$$

for a known data matrix $Y \in \mathbb{R}^{n_1 \times n_2}$. In this example, and in fact in most problems involving LRA, the cost has interesting blockwise properties. Here, the cost is not convex because of the product of variables, but it is block-convex, and often blockwise strongly convex in practice. This observation hints at the design of a class of convergent algorithms for computing solutions to LRA problems that update each block sequentially.

This section provides a brief overview of existing results on methods that update the blocks of variables in an alternative manner. These methods are clustered in two groups: Alternating Optimizations methods (AO) where the cost is exactly minimized with respect to each block alternatively, and Block Coordinate Descent methods (BCD) where the cost decreases with respect to each block alternatively. BCD methods considered here are all, in fact, particular cases of alternating MM algorithms. The literature on both AO and BCD is old and vast; I present mostly recent results of practical interest to the rest of the manuscript, but an interested reader should find in these recent references many acknowledgments of previous works. AO was especially intensively studied in the 1970s. This section is also similar to, and heavily inspired by, chapters 11 and 14 from the book of Amir Beck [Beck, 2017], and to Section 8.1.3 in the book of Nicolas Gillis [Gillis, 2020]. The book of Beck also constitutes a great introduction to convex optimization in general, which can be completed by the book of Dimitri Bertsekas [Bertsekas, 1999].

We build the rest of this section as follows. First, common assumptions are quickly recalled. Then AO and BCD are introduced sequentially, moving from the more general to the more specific results.

8.1 Summary of assumptions and convergence results

This section collects useful assumptions for the convergence results of AO and BCD. The definitions of convexity, differentiability, Lipschitz-smoothness, and related optimization concepts are not recalled and can be found, for instance, in [Beck, 2017]. The shorthand notation $x = (x_1, \dots, x_d)$ is used.

Here is a list of the assumptions required for proving the convergence of AO and BCD:

- (A0): f can be splitted as $f(x) = f_0(x) + \sum_{k=1}^{d} r_k(x_k)$.
- (A1): $\mathcal{X}_k$ are closed convex sets.
- (A2): f has bounded level sets $I_f(z) = \{x | f(x) \leq z\}$.
- (A3): f is proper, lower semicontinous.
- (A4): f is convex.
- (A5): f is continuous over its domain, that is, continuous strictly in the interior of the definition domain.
- (A6): f is differentiable in the interior of its domain.
- (A7): f is continuously differentiable.
- (A8): f is globally Lipschitz-smooth.
- (A9): f is block Lipschitz-smooth, meaning that the restrictions of function f on each block are Lipschitz-smooth.
- (A10): f admits directional differentials defined at x along direction d as $f'(x,d) = \liminf_{\lambda \to 0} \frac{f(x \lambda d) - f(x)}{\lambda}$ (notice that λ is positive). This includes nonsmooth functions such as the ℓ_1 norm.
- (A11): The block updates $\operatorname{argmin}_{x_k \in \mathcal{X}_k} f(x_1, \dots, x_k, \dots, x_d)$ have a unique minimizer.
- (A12(p)): f is block strictly quasiconvex with respect to p blocks. Quasiconvexity of the function f is equivalent to the assumption that any level set $I_f(z)$ is convex.
- (A13): f is non-increasing on the update path.
- (A14): f is regular at the limit points of the algorithm sequence.

We use the notation (A9)(f) to denote, for instance, assumption (A9) applied on the maps f. (A0) and (A4,A7)(f_0) mean that f can be split as f_0 and separable maps r_k, and that f_0 is convex and differentiable on the interior of its domain. When an assumption, *e.g.*, convexity, applies on a majorant u of f, the notation (A4)(u) is used. Table *List of convergence results* summarizes the various convergence results for AO and BCD with their assumptions.

Table 8.1: List of convergence results

Reference	Assumptions	Result
Theorem 3	(A2,A3,A5,A11)(f)	AO limit points are coordinatewise minimum
Theorem 4	(A1) (A0,A2,A3,A5,A11)(f) (A6)(f_0) (A3,A4)(r_k)	AO limit points are stationary points
Theorem 5	(A0,A2,A3,A5)(f) (A4,A6)(f_0) (A3,A4)(r_k)	AO limit points are stationary points
Theorem 6	(A1) (A3,A6,A11,A13)(f)	AO limit points are stationary points
Theorem 7	(A1) (A3,A5)(f)	$d = 2$ AO limit points are stationary points
Theorem 8	(A1) (A3,A5,A12(d-2))(f)	AO limit points are stationary points
Theorem 9	(A1) (A2,A3,A10)(f)	SUM sequence converges to a stationary point
Theorem 10	(A1) (A3,A10,A14)(f) (A11,A12(d))(u)	BSUM limit points are stationary points
Theorem 11	(A1) (A2,A3,A10,A14)(f) (A11)(u)	BSUM iterates converge to stationary points

Remark

A few remarks on these assumptions:

- Block Lischitz-smoothness (A9) does not imply global Lipschitz-smoothness (A8) for nonconvex functions.
- Boundness of the level sets of the cost is implied, for instance, by coercivity. This property is not trivially satisfied for LRA models, see `about-compact-sets-and-lra`.
- Continuity over the definition domain (A5) implies lower-semicontinuity only when the domain is closed.

This section also relies on the concept of MM, already discussed in the context of EM in *Nonnegative regressions: NNLS and NNKL*. Providing an overview of MM in this manuscript would be out of scope; the reader is referred to the excellent survey article by Sun, Babu, and Palomar [Sun *et al.*, 2017].

Note: Three assumptions often found in the numerical optimization literature,

- a function is lower-semicontinuous,
- a function is closed,
- a function has closed level sets,

are equivalent when considering extended real-valued functions from an Euclidean space, see [Beck, 2017] Theorem 2.6. The results in the book by Beck use the terminology "closed", but I prefer "lower-semicontinuous", which is more clearly related to continuity.

8.2 Alternating Optimization

AO, sometimes also called exact BCD, minimizes the cost function f sequentially for each block of variables, essentially performing cyclic updates

$$x_k^{(i+1)} = \underset{x_k \in \mathcal{X}_k}{\operatorname{argmin}} \, f(x_1^{(i+1)}, \ldots, x_{k-1}^{(i+1)}, x_k, x_{k+1}^{(i)}, \ldots, x_d^{(i)}) \tag{8.1}$$

at iteration $i+1$ and for the k-th block. It is immediate to observe that because the cost f is positive, and because each AO update diminishes the cost, the sequence of cost function values is always decreasing. Therefore, the AO algorithm converges, in cost, towards a positive value. Interesting questions, therefore, are not to know if the AO algorithm converges in cost, but rather:

- Does AO converge in cost to a stationary point, and at what rate?
- Are the sequences of iterates $\{x_k^{(i)}\}_{k \leq d}$ converging towards a stationary point in the nonconvex case, or the global minimizer in the convex case, and at what rate?

A famous counter-example for the convergence of AO was proposed by Powell in 1973 [Powell, 1973]. This counter-example occurs because the minimization subproblems with respect to each block do not have a unique solution (A11) [Beck, 2017]. We will see below that most AO convergence guarantees rely on this assumption.

One may note the similarity with the assumptions for the convergence of MM to stationary points. For $d = 2$ blocks, single-block MM and AO are in fact equivalent [Sun *et al.*, 2017]. Denoting

$$x_2(x_1) = \underset{x_2}{\operatorname{argmin}} \, f(x_1, x_2),$$

we get that

$$f(x_1, x_2(x_1^{(i)})) \geq f(x_1, x_2(x_1)).$$

Two blocks AO is exactly the computation of, first, $x_2(x_1^{(i)})$, and then the minimization of the majorant $f(x_1, x_2(x_1^{(i)}))$. This majoration can be loose for arbitrary functions without inter-block regularity, which can intuitively explain the slow convergence discussed in *different-flavours-of-convergence-of-ao-iterates*.

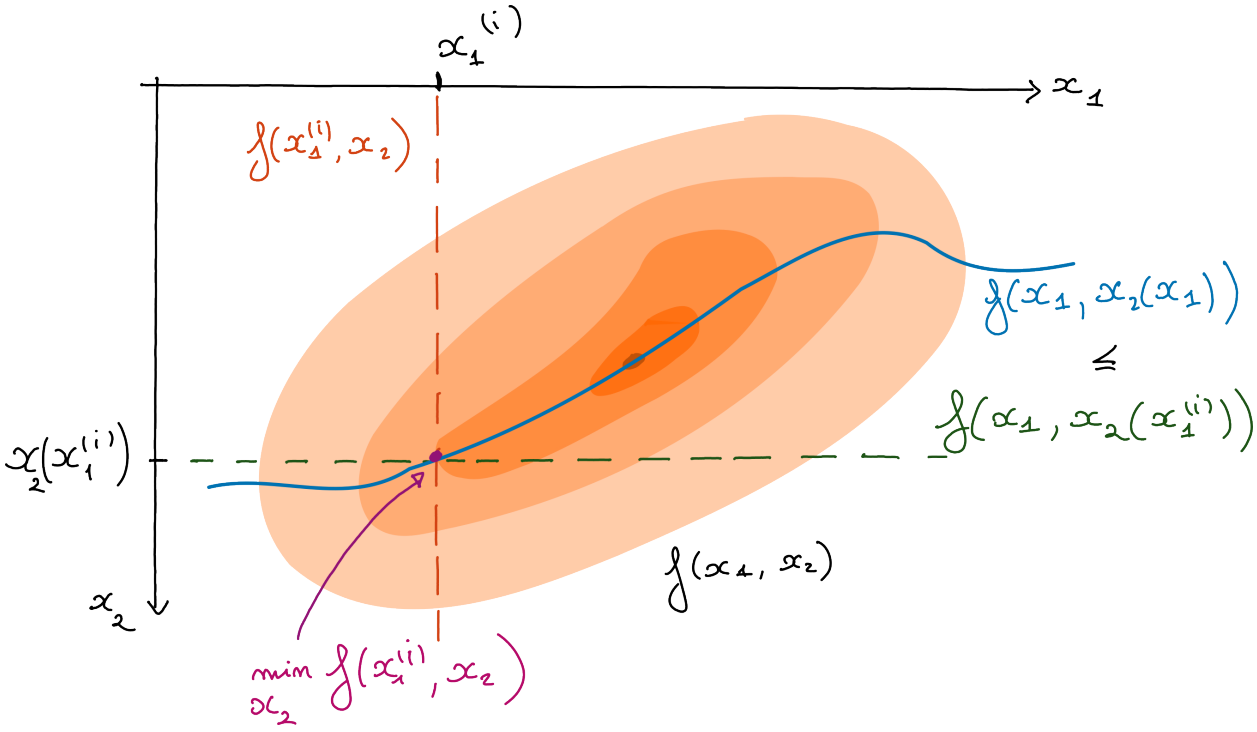


Fig. 8.1: AO seen as a MM technique. The colored surfaces represent level sets of the bivariate loss $f(x_1, x_2)$. The horizontal dashed line is the loss function restricted to the line $x_2 = x_2(x_1^{(i)})$ for some estimated $x_1^{(i)}$ at iteration i. Along this line the loss is always greater than the cost on the curve $f(x_1, x_2(x_1))$ which is always optimal with respect to x_2 for a fixed x_1. The minimization of the loss along the horizontal dashed line yields one iteration of AO.

Another immediate issue with AO is that, without additional assumptions, it may converge only to a coordinatewise minimum, namely a point at which the cost $f(x)$ is minimal along each coordinate. Coordinate-wise minima are not necessarily stationary points, and Beck shows that this holds even when the cost is convex. The root of the problem is that the Fermat rule $0 \in \partial f$ does not decompose into separate conditions for each block, unlike the gradient for differentiable functions: it is always true that $\nabla f = 0$ is equivalent to $\forall k \leq d,\ \frac{\partial}{\partial x_k} f = 0$, but for non-differentiable functions, $0 \in \partial f$ only implies $\forall k \leq d,\ 0 \in \partial f_k$ while the converse is not true. This explains why most works on AO that consider nonsmooth costs introduce assumption (A0), additive separable nonsmooth regularizations r_i to a differentiable multi-block cost f_0,

$$f(x_1, \dots, x_d) = f_0(x_1, \dots, x_d) + \sum_{k=1}^{d} r_k(x_k)$$

where f_0 is proper, lower-semicontinuous, and continuously differentiable over its domain (it can be nonconvex), and functions r_k are proper, convex, lower-semicontinuous. A typical example where the constraints are nonsmooth and non-separable is linear coupling constraints between variables; AO for these problems is bound to get stuck at a coordinatewise minimum, which may be irrelevant to the global optimization problem.

8.2.1 Different flavours of convergence of AO iterates

Probably the most well-known result on the convergence of AO is due to Dimitry Bertsekas and was published in his book Nonlinear Programming in 1995 [Bertsekas, 1999]. There have been several iterations of his result, in particular to distinguish between convergence to coordinatewise minima and stationary points. Below is a summary of Bertsekas's result and its variations.

Remark

The book has been edited three times, in 1995, 1999, and 2016.

Theorem 3 (Convergence of AO, nonconvex nonsmooth)

Assumptions (A2,A3,A5,A11)(f)

Assume that the function f is proper, lower-semicontinuous, and continuous over its domain. Assume that each block update (8.1) has a unique solution. Assume also that the level sets of the function f are bounded. Then the sequence of AO iterates is bounded, and any limit point is a coordinatewise minimum.

From [Beck, 2017, Bertsekas, 1999]

As discussed above, the convergence to coordinatewise minima is of little use for nonsmooth functions, since they are not necessarily stationary points. Therefore, while this first result does not require differentiability, in practice it should be used with caution in a non-differentiable setup; hence the splitting of smooth and nonsmooth terms in this second theorem.

Theorem 4 (Convergence of AO, nonconvex smooth + separable convex nonsmooth)

Assumptions (A1) (A0,A2,A3,A5,A11)(f) (A6)(f_0) (A3,A4)(r_k)

Assume that the function f is proper, lower-semicontinuous, and continuous over its domain. Assume that each block update (8.1) has a unique solution. Assume also that the level sets of the function f are bounded.

If moreover function f decomposes as $f(x) = f_0(x) + \sum_{k \leq d} r_k(x_k)$ with r_k proper, lower-semicontious, continuous over its domain and convex, and f_0 lower-semicontinuous, differentiable on the interior of a Cartesian product of closed convex sets, then any limit point of the AO iterates is a stationary point.

From [Beck, 2017]

There are two further versions of this result. First, one may want to relax the assumption that the block updates have unique solutions; this assumption can indeed be too strong in practical cases where convergence is yet observed. This assumption can be easily relaxed if the cost f is convex.

Theorem 5 (Convergence of AO, convex smooth + separable convex nonsmooth)

Assumptions (A0,A2,A3,A5)(f) (A4,A6)(f_0) (A3,A4)(r_k)

Assume that the function f is proper and continuous over its domain. Assume also that the level sets of the function f are bounded.

If moreover function f decomposes as $f(x) = f_0(x) + \sum_{k \leq d} r_k(x_k)$ with r_k proper, lower-semicontinuous, continuous over its domain and convex, and f_0 continuously differentiable and **convex**, then any limit point of the AO iterates is a stationary point.

From [Beck, 2017]

A second version that applies in the nonconvex case discards the splitting and the boundness of the level sets assumptions. To obtain convergence to a stationary point, differentiability is required. The level set assumption is replaced by the requirement that along the block updates, the cost is non-increasing, a property resembling pseudo-convexity often used in earlier AO convergence proofs. This is actually the original result by Bertsekas.

Theorem 6 (Convergence of AO, nonconvex smooth)

Assumptions (A1) (A3,A6,A11,A13)(f)

Assume that the function f is proper, lower-semicontinuous, and continuously differentiable over its domain, which is a Cartesian product of closed convex sets. Assume that each block update (8.1) has a unique solution.

Assume further that the cost is non-increasing in the interval

$$\left[(x_1^{(i+1)}, \dots, x_{k-1}^{(i+1)}, x_k^{(i)}, x_{k+1}^{(i)}, \dots, x_d^{(i)}), (x_1^{(i+1)}, \dots, x_{k-1}^{(i+1)}, x_k^{(i+1)}, x_{k+1}^{(i)}, \dots, x_d^{(i)})\right].$$

Then any limit point of the AO iterates is a stationary point.

From [Bertsekas, 1999]

It is important to note that the order of the AO block updates is not important in these results, as long as there exists an integer K such that each block is updated at least once in every K AO iterations [Gillis, 2020].

Convergence rate of AO

The convergence rate of the AO algorithm with arbitrarily many blocks is found in the book of Amir Beck [Beck, 2017]. AO converges sublinearly in the convex, Lipschitz-smooth plus separable convex nonsmooth case; in fact, AO has the same asymptotic convergence speed as first-order methods. Moreover, the convergence rate depends linearly on the Lipschitz constant L_{f_0} of the function f_0, and therefore on the worst Lipschitz constant over all blocks. In practice, however, AO can be faster than BCD. An example of this phenomenon for gradient descent performed over $\mathcal{X}_1 \times \dots \times \mathcal{X}_d$ is discussed in *Practical issues*.

AO with $d = 2$ blocks

An important special case is AO with two blocks. The convergence of AO with two blocks has been studied in the seminal paper of Grippo and Sciandrone [Grippo and Sciandrone, 2000].

Theorem 7 (Convergence of AO, $d = 2$, nonconvex smooth)

Assumptions (A1) (A3,A5)(f)

Assume that the function f is continuously differentiable, and assume that the sets $\mathcal{X}_1$ and $\mathcal{X}_2$ are closed and convex sets. Then the limit points of the AO iterates are stationary points.

Notice how the assumptions for the two-block case are milder than in the general case. Compared to *Theorem 6*, the two-block case requires neither the uniqueness of block updates nor the monotonicity of the cost along the update path. The two-block case is useful when considering the convergence of alternating algorithms for models such as NMF or Dictionary Learning (DL).

Convergence rate of AO with two blocks

The convergence rate of AO with two blocks is still sublinear under convexity and separable nonsmoothness assumptions, but the constant is now proportional to the **smallest** Lipschitz constant of f_0 for the two blocks, a sharp improvement with respect to the $d > 2$ blocks case [Beck, 2017]. Moreover, the function f_0 need not be Lipschitz-smooth and may be differentiable only on an open set.

Extension of the two-blocks case

Grippo and Scriandrone also provide a convergence result for $d > 2$ blocks based on blockwise strict quasiconvexity of the cost for $d - 2$ blocks and continuous differentiability of the cost.

Theorem 8 (Convergence of AO, block strictly quasiconvex, smooth)

Assumptions (A1) (A3,A5,A12(d-2))(f)

Assume that the function f is continuously differentiable, and that the sets $\mathcal{X}_k$ are closed and convex. Further assume that the function f is blockwise strictly quasiconvex with respect to $d - 2$ blocks. Then the limit points of the AO iterates are stationary points.

This result requires both blockwise strict quasiconvexity and the differentiability of the cost, and is weaker in that sense than the variants introduced above. Yet because it does not require the uniqueness of the block updates and does not require convexity with respect to all blocks, it can sometimes be applied in contexts where *Theorem 4* cannot.

8.3 Block-coordinate descent

AO is a simple framework for multiblock optimization problems, but its convergence guarantees can be hard to satisfy. In particular, computing the exact block updates and ensuring their uniqueness can be challenging. The convergence rate is also sublinear, and one may hope to find faster alternating algorithms. BCD is another algorithmic framework in which the blocks are updated sequentially (in any order), but each block update may only reduce the cost function value. The next sections introduce BCD frameworks based on the MM principle.

Remark

As for AO, in BCD, the blocks can be visited in any order, provided each block is visited at least once every K iterations for a fixed integer K. In particular, each block can be updated several times. If a block is updated many times, a BCD algorithm may approximately behave like an AO algorithm, see the discussion on the practical consequences of this observation in `practical-issues`.

8.3.1 Successive upper-bound minimization

The Successive Upper-bound Minimization framework (SUM), proposed by Razaviyayn, Hong, and Luo in 2013 [Razaviyayn *et al.*, 2013], is an extension of the MM principle, in which the majorant must be continuous and have the same first-order derivative as the cost at the majoration point in all directions. This requirement is rather weak in practice, as many MM algorithms rely on Taylor expansions to build the majorants, in which case this tangent condition holds.

The SUM framework is simple to describe and essentially the same as the MM framework. Consider a single block function $f(x)$.

1. At a given point $x^{(i)}$, build a continuous majorant $u(y, x^{(i)})$ of the cost f, tight and tangent.
2. Compute $x^{(i+1)}$ as a minimizer of the majorant $u(y, x^{(i)})$ over y.
3. Repeat 1. and 2. until convergence.

The convergence of SUM is simple to prove under mild assumptions.

Theorem 9 (Convergence of SUM, nonconvex nonsmooth)

Assumptions (A1) (A2,A3,A10)(f)

Assume that the function f admits directional derivatives at all points, and that the set $\mathcal{X}$ is closed and convex. Further assume that the majorant u in SUM satisfies the majoration conditions, namely it is tight, tangent in all directions, and upper-bounds the cost at all points. Then every limit point of the SUM iterates is a stationary point. If, further, the level set $\{x|f(x) \leq f(x^{(0)})\}$ is bounded, then the iterates converge to the set of stationary points.

From [Razaviyayn *et al.*, 2013].

A typical use of the SUM framework is when the cost decomposes as $f(x) = f_0(x) + r(x)$, with function f continuously differentiable and function r a proximable regularization with direction derivatives such as the ℓ_1 norm. Then SUM where f is majorized by a second-order Taylor expansion of f_0 with an isotropic quadratic term is exactly a proximal gradient descent algorithm. SUM also covers the MU algorithm discussed in *Nonnegative regressions: NNLS and NNKL*. In general, the convergence rate of SUM is therefore at best sublinear.

8.3.2 The block successive upper-bound minimization framework

The extension of SUM to a multiblock function is called Block SUM (BSUM) and is a generic framework that encompasses many other BCD algorithms, such as PALM [Bolte *et al.*, 2014], under the hypothesis that the cost admits directional derivatives. BSUM is exactly the application of SUM sequentially over all blocks, in any order.

Remark

While PALM is arguably more well-known in the French numerical optimization community, in my opinion, it offers little more than the BSUM framework. BSUM is more general than PALM. When the majorants in BSUM are blockwise second-order isotropic Taylor expansions of the differentiable part of the blockwise Lipschitz-smooth cost, BSUM is exactly PALM. The convergence guarantees of BSUM to coordinatewise minima (and to stationary points under regularity assumptions) hold, since the majorants are strongly convex and the updates have unique solutions. PALM leverages the Kurdyka-Łojasiewicz property and assumes a splitting as continuously differentiable plus separable nonsmooth terms to guarantee convergence towards a critical point without directional derivability or regularity assumptions. The proof techniques in SUM and BSUM are also arguably simpler than in PALM; these frameworks are thus more amenable to an introductory course on alternating optimization techniques.

Compared to SUM, BSUM requires additional assumptions for convergence, which come in two flavors. First, one may assume that the majorants are quasiconvex and that the block updates have unique solutions.

Theorem 10 (Convergence of BSUM, quasiconvex nonsmooth)

Assumptions (A1) (A3,A10,A14)(f) (A11,A12(d))(u)

Assume that the function f admits directional derivatives at all points, and that the sets $\mathcal{X}_k$ are closed and convex. Further assume that the majorants u in BSUM satisfy the majoration conditions for all blocks, namely the majorants are tight, tangent in all directions, and upper-bound the cost at all points. Additionally, assume that the majorants for each block are quasiconvex, and that the block updates have unique solutions.

Then every limit point of the BSUM iterates is a coordinatewise minimum. If, further, the cost is regular at all these limit points, then the limit points of the BSUM iterates are stationary points.

From [Razaviyayn *et al.*, 2013].

Remark

Regular in this context means that the Fermat rule implies stationarity.

A second version of this convergence result avoids the quasiconvexity assumption and shows convergence of the iterates, rather than of their limit points, towards the set of stationary points. It relies on the compacity of the level set $I_f(x^{(0)}) = \{x | f(x) \leq f(x^{(0)})\}$ where $x^{(0)}$ is the initialization of the BSUM iterates.

Theorem 11 (Convergence of BSUM, nonconvex nonsmooth)

Assumptions (A1) (A2,A3,A10,A14)(f) (A11)(u)

Assume that the cost function f admits directional derivatives at all points, and that the sets $\mathcal{X}_k$ are closed and convex. Further assume that the majorants u in BSUM satisfy the majoration conditions for all blocks, namely the majorants are tight, tangent in all directions, and upper-bound the cost at all points. Additionally, assume that the level set $I_f(x^0)$ is bounded, and that the block updates have unique solutions for at least $d-1$ blocks. Further, assume that the cost is regular at all the stationary points.

Then the BSUM iterates converge to the set of stationary points.

From [Razaviyayn *et al.*, 2013].

Remark

The original work of Razaviyayn states that the level sets must be compact, but the continuity of f implies that they are closed. Hence, compactness here requires only that the level sets be bounded.

It is important to note that, as with AO, the BSUM framework assumes that the constraint sets are disjoint. In particular, BSUM cannot handle linear couplings between blocks of variables, and in fact, BSUM, like AO, is bound to fail in this setup.

The convergence rate of BSUM was studied in the convex case [Hong *et al.*, 2017]. BSUM, PALM, and other similar first-order BCD algorithms have, in general, sublinear convergence rates, see also chapter 11 of the book of Beck [Beck, 2017].

BCD with extrapolation

An important research direction to accelerate BCD algorithms is to leverage extrapolation of iterates, inspired by the seminal work of Nesterov on the fast gradient algorithm [Nesterov, 1983]. Going into the details of all the proposed extrapolated BCD methods would be out of scope. Instead, an interested reader will find below a list of a few useful works on this topic. These methods typically also have sublinear rates, but improved from $\mathcal{O}(i^{-1})$ to $\mathcal{O}(i^{-2})$ where i is the iteration index.

- Alternating Proximal Gradient Descent (APGD) [Xu and Yin, 2013] is a general framework for alternating algorithms, that covers AO and PALM with extrapolation. APGD assumes that the cost is the sum of a strongly convex term (for AO) or Lipschitz smooth (for PALM with extrapolation) and separable convex nonsmooth regularizations. The originality lies in the general conditions with asymptotic rates and the extrapolation for PALM.
- iPALM is an extension of PALM with extrapolation [Pock and Sabach, 2016]. It is similar to APGD but relaxes a condition on the monotonicity of the cost along the update path.
- TITAN [Hien *et al.*, 2023] is essentially the BSUM framework with extrapolation. TITAN combines ingredients from PALM, such as the Kurdyka-Łojasiewicz property, with heavy-ball acceleration. A downside of TITAN is that it introduces, among others, the nearly sufficiently decreasing property that may not be satisfied when the majorants are not strongly convex. This condition can be relaxed, provided other assumptions on the extrapolation parameters, by using the framework of mirror gradient descent to include the extrapolation term in the majorant definition [Hien *et al.*, 2025].

8.4 Practical issues

From the convergence properties summarized above, it is far from obvious whether AO or BCD should be preferred in practice. What's more, AO and BCD are optimization frameworks, but their implementations can significantly affect performance. For BCD in particular, the order in which the blocks are visited and the number of times each block is updated before switching to a different block are important topics.

8.4.1 AO vs BCD vs approximate AO

Let us discuss a concrete example of NMF with the Frobenius loss; see *Low-rank matrix and tensor models* and *Nonnegative regressions: NNLS and NNKL* for details on the NMF model and classical solvers. The solver we consider is *HALS*. Denoting W and H the two factors of the NMF $Y \approx WH^T$, HALS can be seen as a BCD algorithm where the blocks are the columns of matrices W and H. Each block update is computed in closed form.

The HALS updates for the columns of matrix H involve computing the quantities W^TW and W^TY. In fact, in high-dimensional settings, these operations are the computational bottleneck of HALS. Because these quantities are independent of matrix H, it is more efficient in terms of computation cost to update the columns of matrix H several times, sequentially, before switching to matrix W. The number of times n_{inner} the columns of matrix H (and similarly for W) are updated is thus a hyperparameter of HALS. The outer iterations are incremented once both matrices have been updated.

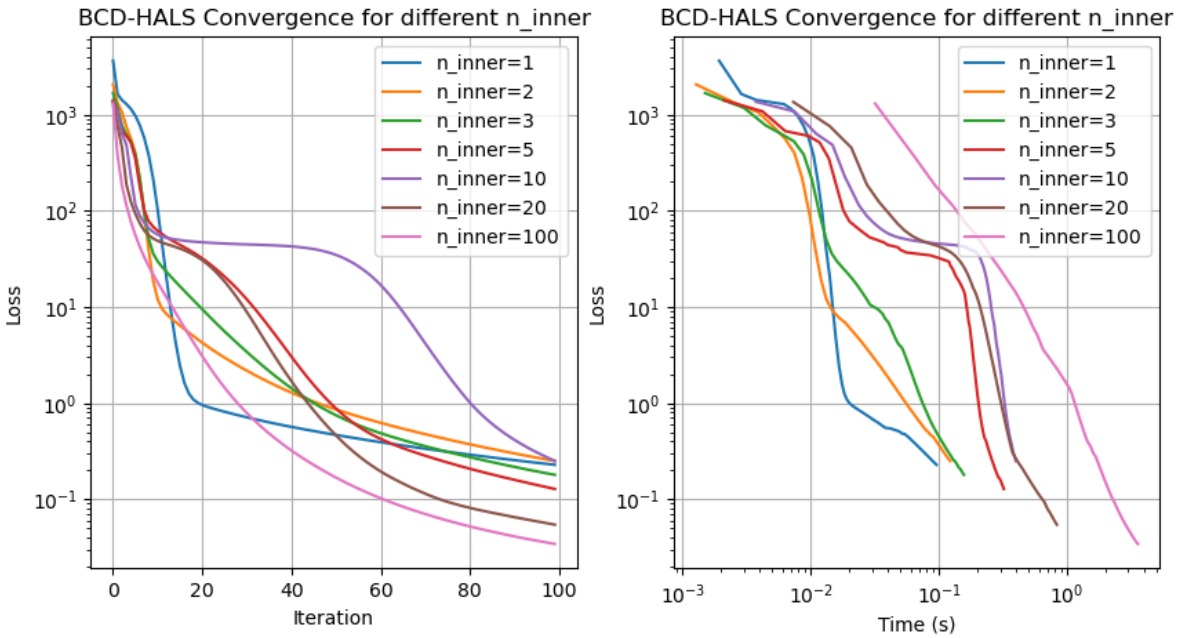
If the number of inner iterations n_{inner} is set to a large number, the HALS algorithm is essentially an (approximate) AO algorithm, ANLS, solving the NNLS problem for matrices W and H alternatively. One may wonder if setting the number of inner iterations n_{inner} to a low number leads to a sharper decrease of the cost **per outer iteration**, meaning that BCD has an advantage over AO in terms of convergence speed, and if this translates into a computation time advantage as well. Let us test the performance of HALS with various inner iterations n_{inner} numerically in a noiseless setting. The HALS implementation is taken from tensorly.

```
import tensorly as tl
import numpy as np
```

(continues on next page)

(continued from previous page)

```
from tensorly.solvers import hals_nnls
from time import perf_counter

def BCD_hals(Y,Winit,Hinit,n_inner,n_outer):
    W = np.copy(Winit)
    H = np.copy(Hinit)
    start = perf_counter()
    loss = []
    time = []

    for it in range(n_outer):
        # Update H
        UtM = W.T @ Y
        UtU = W.T @ W
        H = hals_nnls(UtM, UtU, V=H.T, n_iter_max=n_inner, tol=0).T

        # Update W
        VtM = H.T @ Y.T
        VtV = H.T @ H
        W = hals_nnls(VtM, VtV, V=W.T, n_iter_max=n_inner, tol=0).T

        loss.append(np.linalg.norm(Y - W@H.T,'fro')**2)
        time.append(perf_counter() - start)
    return W, H, loss, time


# Generate toy synthetic nonnegative tensor data
np.random.seed(0)
shape = [2000, 50]
rank = 3
Wtrue, Htrue = [np.random.rand(s, rank) for s in shape]
Y = Wtrue @ Htrue.T
Winit = np.random.rand(shape[0], rank)
Hinit = np.random.rand(shape[1], rank)
results = {}

for n_inner in [1, 2, 3, 5, 10, 20, 100]:
    West, Hest, loss, time_ = BCD_hals(Y, Winit=Winit, Hinit=Hinit, n_inner=n_inner,␣
 ↪n_outer=100)
    results[n_inner] = (loss, time_)
```



We may observe that:

- Setting a large number of inner iterations leads to lower reconstruction error per outer iteration.
- With respect to time, however, for a data matrix of size 2000 by 50, using one to five inner iterations leads to faster runtime to reach a high precision.

Remark

We could also solve the NNLS problems exactly using AS instead of HALS, but the runtime would be even slower due to the sequential processing of the rows of matrices W and H.

In this example, therefore, AO is a better strategy than BCD per outer iteration. We can only implement approximate AO, and its iterations are costly (solving the NNLS problem exactly requires many inner iterations of HALS); therefore, BCD should be used instead.

Note that this simulation is seeded. Running this experiment several times with different datasets and initializations yields different conclusions about the optimal number of inner iterations, both in terms of iterations and time. The dimensions of the input matrix are also an important factor, and for larger datasets (which makes this notebook slow to run and the whole manuscript, therefore, too slow to compile), setting the number of inner iterations higher can be beneficial. It can also be beneficial to run a different number of inner iterations for each matrix when their dimensions are vastly different; see, for instance, [Gillis and Glineur, 2012].

8.4.2 About continuous differentiability

In this manuscript, we primarily use the squared Frobenius norm to measure discrepancies between the LRA model and the dataset. However, we also sometimes use β-divergences, and the KL-divergence in particular. It is then important to note that the KL-divergence is not differentiable at zero, and that, therefore, some of the results introduced above do not apply to the convergence of AO and BCD. *Theorem 4*, on the contrary, applies to KL-divergence loss since differentiability is only required in the interior of the domain of the smooth part f_0 of the cost.

8.4.3 About compact sets and LRA

One may object that many of the AO and BCD convergence results rely on the assumption that the sets $\mathcal{X}_k$ are bounded or compact. In the context of LRA, due to scaling ambiguity, the block variables, which are typically the factors of the LRA model, do not belong to compact sets because these sets are unbounded. This problem is addressed in Gillis's book on NMF, p. 267. His solution applies to most LRA models [Gillis, 2020]. In a nutshell, one may simply upper-bound the norms of the blocks with a large enough constant that depends on the data and the initialization. The trick also applies to guarantee that the level set $\{x|f(x) \leq f(x^{(0)})\}$ is compact as long as the cost is coercive.

Part IV

Theory of rLRA

CHAPTER

NINE

THEORY OF RLRA: SUMMARY

Despite their widespread use in the signal processing and machine learning communities, rLRA models remain rather poorly understood to this day. There are two kinds of theoretical questions in the literature for which there are no clear answers.

- **Identifiability**: given an exact factorization model, *e.g.*, $M = X_1 X_2$, and a set of constraints $\{\mathcal{C}_i\}_i$ such that $X_i \in \mathcal{C}_i$, are solutions to the factorization problem under these constraints essentially unique?
- **Solution characterization**: given a constrained or regularized optimization problem, for instance

$$\underset{X_1, X_2}{\operatorname{argmin}} \|M - X_1 X_2^T\|_F^2 + g_1(X_1) + g_2(X_2)$$

  for some regularization functions g_1 and g_2, what are the properties satisfied by the solutions of this problem?

These two questions are related at first glance but differ fundamentally. The identifiability of rLRA models is a cornerstone of source separation techniques, as it allows one to give a physical meaning to the estimated factor matrices. Characterizing the solutions to an optimization problem amounts to ensuring that certain properties hold for the estimated factor matrices, regardless of whether the solution is unique. For instance, the user may care only about the sparsity level of the factor matrices or their orthogonality.

Identifiability of rLRA has received significant attention, in particular for matrix factorization models such as DL, NMF, or tensor decomposition models such as nonnegative CPD and, more recently, nonnegative Tucker decomposition, see *Low-rank matrix and tensor models*. My contribution to the identifiability question has been to provide deterministic sufficient conditions for the *identifiability of complete DL* (cDL), improving upon other works and correcting mistakes in the literature. I also show how to *compute complete DL* in practice with tensorly.

Solution characterization, on the other hand, has not been considered much in the literature, in particular in contrast to the significant body of work that concerns regularized regression problems such as ridge regression or LASSO [Foucart and Rauhut, 2013]. For instance, it is still unclear whether solutions to a sparse NMF problem

$$\underset{X_1 \geq 0,\, X_2 \geq 0}{\operatorname{argmin}} \|M - X_1 X_2^T\|_F^2 + \mu \left(\|X_1\|_1 + \|X_2\|_1\right)$$

are indeed sparse or not, and under which conditions. This might be due to the relative difficulty of studying the solutions of these problems. The problem with a single factor admits a closed-form solution via the soft-thresholding operator, whereas the sparse NMF problem with two factors is nonconvex, nonsmooth, and NP-hard. As a first step towards understanding the properties of the solution or rLRA, together with Valentin Leplat, we showed that, for a large class of rLRA problems, explicit regularization, such as ℓ_1-ℓ_1, leads to implicit regularization due to the scale-invariance of rLRA models. Our contributions, both on the implicit regularization and its algorithmic implications, are summarized in *Implicit regularization in rLRA*.

Because rLRA solutions have ambiguous properties, in particular for homogeneous regularization such as the ℓ_1 norm, I studied, in another line of work, DL and dictionary-based LRA under cardinality constraints. Cardinality constraints impose sparsity explicitly on the solution. I have studied in particular the identifiability and the conception of dedicated algorithms for a class of models that includes the following dictionary-based NMF

$$\underset{X_2 \geq 0,\, S \in \{0,1\},\, \|S[:,q]\|_0 \leq 1}{\operatorname{argmin}} \|M - DSX_2^T\|_F^2$$

where D is a known dictionary of template components. These contributions are summarized in *Dictionary-based LRA*.

CHAPTER

TEN

IMPLICIT REGULARIZATION IN RLRA

Reference

[Cohen and Leplat, 2025] J. E. Cohen, V. Leplat, "Efficient Algorithms for Regularized Nonnegative Scale-invariant Low-rank Approximation Models", SIAM Journal of Mathematics on Data Science, 2025 [arxiv]

Consider the ridge NMF problem with Frobenius loss:

$$\underset{X_1\geq 0, X_2\geq 0}{\operatorname{argmin}} \|M - X_1X_2^T\|_F^2 + \lambda_1\|X_1\|_F^2 + \lambda_2\|X_2\|_F^2.$$

with $M \in \mathbb{R}_+^{m_1\times m_2}$ and the nonnegative rank is r.

Such a ridge-regularized model could be preferred by a user on the basis of ridge-regression: avoiding overfitting (in the case of NMF, this would translate as choosing a particular solution with low variance) or improving the numerical stability of the optimization algorithm (the optimization problem is strictly bi-convex when $\lambda_i > 0$).

It turns out that, maybe surprisingly at first, the ridge-regularized NMF problem is equivalent (in the sense that the minimizers are related by a trivial mapping) to a variant of the nuclear-norm low-rank sensing problem with nonnegativity constraints,

$$\underset{X_1\geq 0, X_2\geq 0}{\operatorname{argmin}} \|M - X_1X_2^T\|_F^2 + \frac{\sqrt{\lambda_1\lambda_2}}{2}\sum_{q=1}^{r}\|X_1[:,q]\otimes X_2[:,q]\|_F$$

This second formulation is insightful in many ways.

- The regularization in the second, implicit formulation, takes the form of a group-sparse regularization of the rank-one terms in NMF. Without nonnegativity constraints, this constraint is exactly the nuclear norm of the low-rank matrix $X_1X_2^T$. The ridge penalty on each factor matrix, therefore, implies rank-minimization implicitly.
- Tuning each parameter λ_1 and λ_2 individually yields counterintuitive results. For instance, when λ_1 is much larger than λ_2, the regularization on the first factor X_1 is not stronger than on the second factor X_2. Only the product of the two regularization hyperparameters governs the regularization intensity.

In the work conducted with Valentin Leplat [Cohen and Leplat, 2025], we generalize this observation for a large class of regularized LRA models, namely Homogeneous Regularized Scale-Invariant models (HRSI), that account for many LRA models regularized with homogeneous positive-definite regularizations such as any ℓ_p norm. The implicit rank-selection effect observed for ridge regularization is shown to be due to the scale-ambiguity of LRA models, and therefore pertains to most ℓ_p norms. We provide the *main theoretical result* below, and provide more examples, namely ℓ_1-ℓ_1 and ℓ_1-ℓ_2 regularizations.

We present numerical simulations in the next section, along with algorithm-oriented concerns, such as leveraging scale-invariance to optimally balance the factors to minimize regularization, which provably allows one to escape the so-called "scaling swamp" [Papalexakis *et al.*, 2013].

10.1 Scale invariance main result

While the scale of the factor matrices leaves the low-rank approximation intact, namely $X_1 X_2^T$ and $X_1 \Lambda \Lambda^{-1} X_2^T$ are equivalent low-rank representations for any nonzero diagonal matrix Λ, implicit regularization in LRA essentially occurs because the regularizations are not invariant with respect to such a balancing of columns and rows of the factor matrices. Solving for the optimal scales in rLRA modifies the penalization. Further, the optimal scales are determined by the homogeneity degree of the regularizations and the value of the regularization hyperparameters, and essentially balance the rank-one components.

In what follows, we first introduce the general framework of HRSI, and then derive the main result that relates scale-invariance with implicit regularization.

10.1.1 Homogeneous Regularized Scale-Invariant models

Consider the following optimization problem

$$\underset{\forall i \leq n,\, X_i \in \mathbb{R}^{m_i \times r}}{\operatorname{argmin}} \; f(\{X_i\}_{i \leq n}) + \sum_{i=1}^{n} \mu_i \sum_{q=1}^{r} g_i(X_i[:, q]), \tag{10.1}$$

where f is a continuous map from the Cartesian product $\times_{i=1}^{n} \mathbb{R}^{m_i \times r} \cap \operatorname{dom}(g_i)$ to $\mathbb{R}_+$, $\{\mu_i\}_{i \leq n}$ is a set of positive regularization hyperparameters, and $\{g_i\}_{i \leq n}$ is a set of lower semicontinous regularization maps from $\mathbb{R}^{m_i}$ to $\mathbb{R}_+$. We assume that the total cost is coercive, ensuring the existence of a minimizer. Furthermore, we assume the following assumptions hold:

- **A1**: The function f is invariant to columnwise scaling of the parameter matrices. For any sequence of positive diagonal matrices $\{\Lambda_i\}_{i \leq n}$,

  $$f(\{X_i \Lambda_i\}_{i \leq n}) = f(\{X_i\}_{i \leq n}) \;\text{ if }\; \prod_{i=1}^{n} \Lambda_i = I_r,\; \Lambda_i > 0.$$

  In other words, scaling any two columns $X_{i_1}[:, q]$ and $X_{i_2}[:, q]$ inversely proportionally has no effect on the value of f.

- **A2**: Each regularization function g_i is positive and homogeneous of degree p_i, that is for all $i \leq n$ there exists $p_i > 0$ such that for any $\lambda > 0$ and any $x \in \mathbb{R}^{m_i}$,

  $$g_i(\lambda x) = \lambda^{p_i} g_i(x).$$

  This property holds in particular for any ℓ_p^p norm.

- **A3**: Each function g_i is positive-definite, meaning that for any $i \leq n$, $g_i(x) = 0$ if and only if x is null.

We refer to the optimization problem (10.1) with assumptions **A1**, **A2**, and **A3**, the Homogeneous Regularized Scale-Invariant problem (HRSI).

> **Remark**
>
> Assumption **A3** limits the framework's applicability by excluding positive homogeneous regularizations such as Total Variation.

HRSI is quite general and encompasses many instances of regularized LRA problems. Specifically, *ridge NMF* is an HRSI problem with $f(\{X_i\}_{i \leq n}) = \|M - X_1 X_2^T\|_F^2$, $g_1(x) = \|x\|_2^2 + \eta_{\mathbb{R}_+^{m_1}}(x)$ and $g_2(x) = \|x\|_2^2 + \eta_{\mathbb{R}_+^{m_2}}(x)$.

10.1.2 Solutions to HRSI also solve an implicit regularized problem, and are balanced

We proved the following result in [Cohen and Leplat, 2025], relying on an identity for the geometric mean.

Theorem 12 (HRSI solutions characterization and implicit equivalent formulation)

If $\mu_i > 0$ for all i, any solution $\{X_i^*\}_{i\leq n}$ to the HRSI problem (10.1) under assumptions **A1**, **A2** and **A3** satisfies

$$p_i \mu_i g_i(X_i^*[:,q]) = \beta_q$$

for all $i \leq n$ and $q \leq r$, where

$$\beta_q = \left(\prod_{i\leq n} \left(p_i \mu_i g_i(X_i^*[:,q])\right)^{\frac{1}{p_i}} \right)^{\frac{1}{\sum_{i\leq n} \frac{1}{p_i}}}.$$

Moreover, the solutions to the HRSI problem are, up to scaling, solutions to the implicit HRSI problem

$$\underset{\forall i\leq n,\, X_i \in \mathbb{R}^{m_i \times r}}{\operatorname{argmin}} f(\{X_i\}_{i\leq n}) + \tilde{\mu} \sum_{q=1}^{r} \left(\prod_{i=1}^{n} g_i(X_i[:,q])^{\frac{1}{p_i}} \right)^{\frac{1}{\sum_{i=1}^n \frac{1}{p_i}}},$$

where $\tilde{\mu} = \left(\prod_{i=1}^n (p_i \mu_i)^{\frac{1}{p_i}} \right)^{\frac{1}{\sum_{i=1}^n \frac{1}{p_i}}} \left(\sum_{i=1}^n \frac{1}{p_i} \right)$.

From this theorem, we observe that

- Only the product of regularization hyperparameters impacts the regularity of the solution.
- The sum of columnwise regularization on the factor matrices in the explicit formulation becomes a product of regularizations in the implicit formulation, yielding group-sparsity on the rank-one components.
- At optimality, the scales of the components must satisfy the balancing condition $p_i \mu_i g_i(X_i^*[:,q]) = \beta_q$. This means, in particular, that LRA models regularized with homogeneous functions do not suffer from scale ambiguity.

10.2 Special cases with ℓ_1 and ℓ_2 regularizations

To make our main result *Theorem 12* more explicit, let us instantiate the implicit HRSI formulation and the optimal scales for a couple of classical problems in rLRA. Based on the observation that only the product of regularization hyperparameters matters, the same regularization hyperparameter is used for all factor matrices.

10.2.1 Ridge-regularized nonnegative CPD

Define the ridge nCPD formally as the optimization problem

$$\underset{X_i \geq 0}{\operatorname{argmin}} \|\mathcal{T} - \mathcal{I}_r \times_1 X_1 \times_2 X_2 \times_3 X_3\|_F^2 + \mu \left(\|X_1\|_F^2 + \|X_2\|_F^2 + \|X_3\|_F^2 \right).$$

The implicit HRSI problem shows that ridge-regularization yields a componentwise group-sparsity implicit regularization,

$$\underset{\{\mathcal{L}_q\}_{q\leq r},\, \operatorname{rank}(\mathcal{L}_q)\leq 1}{\operatorname{argmin}} \|\mathcal{T} - \sum_{q=1}^{r} \mathcal{L}_q\|_F^2 + 3\mu \sum_{q=1}^{r} \|\mathcal{L}_q\|_F^{\frac{2}{3}}$$

where tensors $\mathcal{L}_q$ are rank-one tensors $X_1[:,q] \otimes X_2[:,q] \otimes X_3[:,q]$. We experimentally validate in *the experiments subsection* that tuning the regularization hyperparameter μ indeed helps us to select the rank of the nCPD factorization automatically. Moreover, the optimal solution of ridge nCPD verifies the balancing equation

$$1 = \sqrt{2}\|X_i^*[:,q]\|_2^{-\frac{1}{3}} \prod_{j \neq i} \|X_j^*[:,q]\|_2^{\frac{2}{3}}.$$

This balancing identity will be used in *the experiments subsection* to refine the iterations of an iterative optimization algorithm to solve ridge nCPD.

10.2.2 Sparse NMF L1-L1

Another interesting case of an implicit regularization effect concerns the (double) sparse NMF model

$$\underset{X_1 \geq 0, X_2 \geq 0}{\operatorname{argmin}} \|M - X_1 X_2^T\|_F^2 + \mu \left(\|X_1\|_1 + \|X_2\|_1 \right).$$

Intuitively, a practitioner using sparse NMF with ℓ_1 penalizations on both factors would expect sparse entries in both matrices, leveraging the usual behavior of the ℓ_1 norm in regression problems. The implicit formulation

$$\underset{L_q \in \mathbb{R}_+^{m_1 \times m_2},\ \operatorname{rank}(L_q) \leq 1}{\operatorname{argmin}} \|M - \sum_{q=1}^{r} L_q\|_F^2 + 2\sqrt{\mu_1 \mu_2} \sum_{q=1}^{r} \sqrt{\|L_q\|_1}$$

shows that, while both factors are indeed penalized to be sparse, there is another group-sparse effect that prunes entire components. The problem with this formulation is that both effects (elementwise sparsity and componentwise sparsity) are not controlled individually but rather by the same regularization hyperparameter. This yields unexpected component pruning as shown in the simulation below with mixtures of Gaussians. It is also unclear from the implicit formulation whether both factors will be sparse elementwise.

```
import numpy as np
import tensorly as tl
import matplotlib
import matplotlib.pyplot as plt

# Generating toy image
n1 = 30
n2 = 40
r = 3
sig = 0.2  # 0.15 # 0.05

# Components will be Gaussian distributions, here is the macro to generate them
def gauss(x, m, sig, thresh=0.1):
    # max is 1
    out = np.exp(-(x-m)**2/sig**2)
    out[out < thresh] = 0
    return out

# Generating the data as a mixture of separable Gaussians
x = np.linspace(0, n1-1, n1) # 1D abcisse
y = np.linspace(0, n2-1, n2)
W = np.zeros([n1, r])
W[:,0] = gauss(x, 5, 3)
W[:,1] = gauss(x, 15, 5)
W[:,2] = gauss(x, 25, 10)
H = np.zeros([n2, r])
H[:, 0] = gauss(y, 10, 5)
```

(continues on next page)

(continued from previous page)

```
H[:, 1] = gauss(y, 20, 6)
H[:, 2] = gauss(y, 25, 5)
trueNMF = (None, [W, H])

Y = W@H.T  # NMF model
rng = np.random.default_rng(1246)
Yn = Y + sig*rng.random((n1, n2))  # corruption with gaussian noise

# initialization
W0 = 0*W + 0.5*rng.random(W.shape)
H0 = 0*H + 0.5*rng.random(H.shape)

# Storing output factors for various regularization in dictionaries
We = dict()
He = dict()

# Chose the grid values for the hyperparameter
lambset = [0, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.6, 0.7, 0.75, 0.78, 0.8, 1, 1.2, 1.5,␣
↪2, 3, 4, 5, 5.2, 6, 20]

import copy
for lamb in lambset:
    # Computing the sparse NMF with tensorly (the algorithm is HALS, which is the␣
↪state of the art for this problem)
    out = tl.decomposition.non_negative_parafac_hals(Yn, r, sparsity_
↪coefficients=[lamb, lamb], init=copy.deepcopy((None, [W0, H0])))
    # Estimated factors
    We[lamb] = out[1][0]
    He[lamb] = out[1][1]
    # Set estimates as new initialization for the next iteration (warm start)
    W0 = We[lamb]
    H0 = He[lamb]

    # rescale We so that when a column of W is 0, so is the corresponding column of H
    norms = tl.max(We[lamb], axis=0)
    for q in range(r):
        if norms[q] > 1e-8:
            We[lamb][:, q] = We[lamb][:, q]/norms[q]
            He[lamb][:, q] = He[lamb][:, q]*norms[q]
        else:
            We[lamb][:, q] = We[lamb][:, q]*0
            He[lamb][:, q] = He[lamb][:, q]*0
```

Observe in particular that the sparsity level of the components does not evolve smoothly with the regularization hyperparameter. Components vanish brutally at $\mu = 0.75$ and $\mu = 5.2$.

10.3 The sparse NMF using l1-l2 regularization

The explicit ℓ_1-ℓ_2 sparse NMF model writes

$$\underset{X_1 \geq 0, X_2 \geq 0}{\operatorname{argmin}} \|M - X_1 X_2^T\|_F^2 + \mu \left(\|X_1\|_1 + \|X_2\|_2^2 \right).$$

The implicit HRSI formulation of sparse NMF can be easily derived:

$$\underset{L_q \in \mathbb{R}_+^{m_1 \times m_2},\, \operatorname{rank}(L_q) \leq 1}{\operatorname{argmin}} \|M - \sum_{q=1}^{r} L_q\|_F^2 + \frac{3}{2^{\frac{2}{3}}} \mu \sum_{q=1}^{r} \|L_q\|_{1,2}^{\frac{2}{3}}.$$

where the mixed matrix norm is defined as $\|L_q\|_{1,2} = \sqrt{\sum_j \left(\sum_i L_q[i,j] \right)^2}$ and we used the identity

$$\|x \otimes y\|_{1,2} = \|x\|_1 \|y\|_2$$

that holds for any two vectors x and y. One may observe that the implicit regularization acts as a group-LASSO norm on the rows of the rank-one components, effectively enforcing sparsity elementwise in the factor matrix X_1. The group-LASSO effect on the rank-one components still appears, however, in the implicit regularization, and similarly to the ℓ_1-ℓ_1 case, the two effects may occur simultaneously.

It is still open to fully characterize the link between the regularized formulation and the constrained formulation

$$\underset{X_1 \geq 0, X_2 \geq 0}{\operatorname{argmin}} \|M - X_1 X_2^T\|_F^2 + \mu \|X_1\|_1 \text{ s.t. } \|X_2[:,q]\|_2 = 1 \ \forall q \leq r.$$

Interestingly, Marmin, Goulard, and Févotte show equivalence between the constrained formulation and the implicit HRSI formulation of sparse NMF, hinting towards the equivalence between the three formulations [Marmin *et al.*, 2023].

10.4 Balanced and non-Euclidean algorithms for regularized LRA

In our work with Valentin Leplat, in addition to the theoretical analysis of HRSI discussed in *Implicit regularization in rLRA*, we tackle a more practical problem. We observe, both numerically and theoretically, the existence of scaling swamps that considerably slow the convergence of alternating algorithms. Can the balancing equation $p_i \mu_i g_i(X_i^*[:,q]) = \beta_q$ be leveraged to escape these swamps?

In what follows, we first show the existence of scaling swamps in a toy problem numerically ([Cohen and Leplat, 2025] formally proves the sublinear convergence rate of ALS for this problem). A simulation then shows the rank-selection capabilities of ridge nCPD, along with the importance of balancing.

10.4.1 Scaling swamps

In the tensor decomposition literature, swamps refer to many consecutive iterations of optimization algorithms in which the cost function remains nearly constant while the parameters change significantly. This phenomenon occurs for many algorithms and is conjectured to be due to the ill-posedness of tensor LRA [Hillar and Lim, 2013], [Comon *et al.*, 2009], [Mohlenkamp, 2019], [Vermeylen *et al.*, 2025].

Another kind of swamp was observed by Papalexakis and Sidiropoulos [Papalexakis *et al.*, 2016] for ℓ_1-ℓ_1 sparse NMF. It is called a scaling-swamp because, as in traditional swamps, the cost improves very slowly over many iterations. Unlike in swamps from the tensor decomposition literature, however, the cost does not decrease any faster after a certain number of iterations, and scaling swamps also occur in matrix factorization problems such as NMF, even though the low-rank manifold is closed.

We can construct a simple numerical example in one dimension in which the ALS algorithm is provably slow. Consider the loss function

$$f(x_1, x_2) = (y - x_1 x_2)^2 + \lambda(x_1^2 + x_2^2). \tag{10.2}$$

where x_1 and x_2 are real values and the regularization hyperparameter λ and the data y are positive real values.

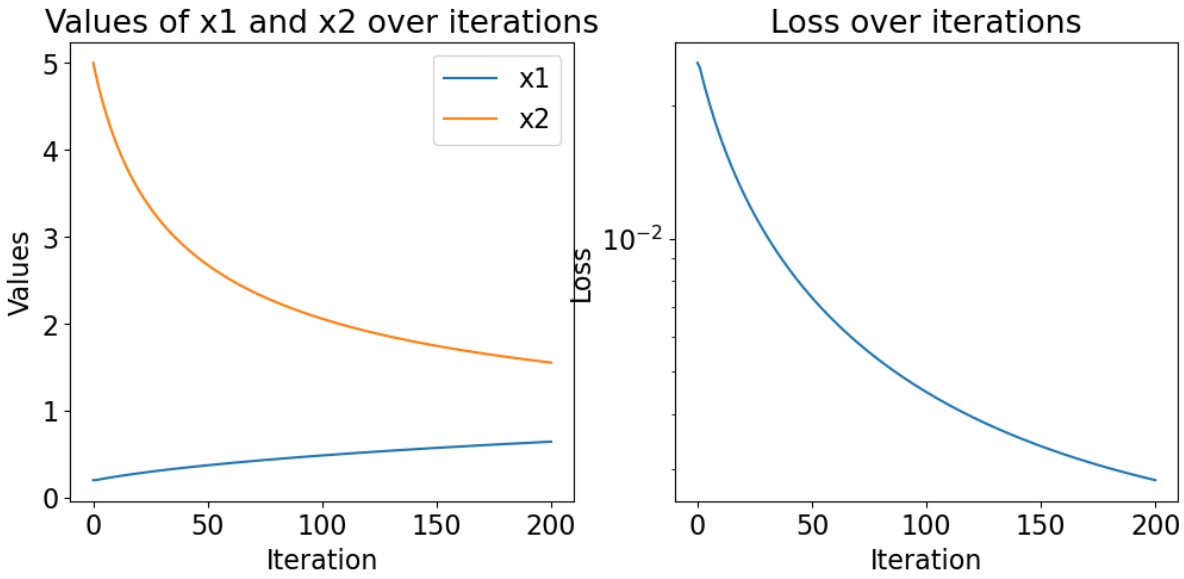
Using a perturbation analysis, it can be observed that the ALS algorithm, whose updates at iteration $k+1$ are given by

$$x_1^{(k+1)} = \frac{x_2^{(k)} y}{x_2^{2(k)} + \lambda} \quad \text{and} \quad x_2^{(k+1)} = \frac{x_1^{(k+1)} y}{x_1^{2(k+1)} + \lambda}.$$

has a convergence rate for the iterates asymptotically close to $1 - 4\lambda/y$, which is arbitrarily close to one (sublinear convergence) for small regularizations. The following numerical simulation shows that scaling swamps are more pronounced when regularization is small relative to the data magnitude.

```
from matplotlib import axes
import numpy as np
import matplotlib.pyplot as plt

# A barebone implementation of the ALS (Alternating Least Squares) algorithm
def ALS(y,x1,x2, lamb, itermax=200):
    loss = [(y - x1*x2)**2 + lamb * (x1**2 + x2**2)]
    x1_store = [x1]
    x2_store = [x2]
    for k in range(itermax):
        x1 = x2*y/(x2**2+lamb)
        x2 = x1*y/(x1**2+lamb)
        loss.append((y - x1*x2)**2 + lamb * (x1**2 + x2**2))
        x1_store.append(x1)
        x2_store.append(x2)
    return x1, x2, loss, x1_store, x2_store

# Example usage, you can play with the values of y and lamb !
y = 1
lamb = 0.001

# Initialization
x1 = 0.2
x2 = 5

# Prints and Plots
x1, x2, loss, x1store, x2store = ALS(y, x1, x2, lamb)
print("After 200 iterations:")
print(f"x1: {x1}, x2: {x2}, Optimal values: {np.sqrt(y-lamb)}")
print(f"Loss: {loss[-1]}, Optimal loss: {(y-np.sqrt(y-lamb))**2+ lamb * 2 * (y-lamb)}
 ↪")
```

```
After 200 iterations:
x1: 0.642558430874947, x2: 1.5525184957484308, Optimal values: 0.999499874937461
Loss: 0.0028290328046436134, Optimal loss: 0.00199825012507818
```



Interestingly, the toy problem (10.2) has a simple closed form solution that can be obtained by optimally balancing the two scalar values x_1 and x_2 as defined in *Theorem 12*, namely

$$|x_1| = |x_2|.$$

Then the optimal solution is simply (up to sign ambiguities) $x_1 = x_2 = \sqrt{y - \lambda}$.

This suggests that balancing the estimates optimally before, after, or within an optimization algorithm solving HRSI could help avoid the scaling swamp phenomenon. We explore here only the balancing of initialization and/or outputs of an algorithm for simplicity.

Balancing initial or final estimates of an algorithm is a straightforward operation using *Theorem 12*. First, compute the columnwise geometric mean

$$\beta_q = \left(\prod_{i \leq n} (p_i \mu_i g_i(X_i^*[:, q]))^{\frac{1}{p_i}} \right)^{\frac{1}{\sum_{i \leq n} \frac{1}{p_i}}}.$$

Then scale all columns such that their regularization is exactly $\frac{1}{p_i \mu_i} \beta_q$. For the problem of ridge-regularized nCPD, this amounts to performing for all i in $\{1, 2, 3\}$ the balancing procedure

$$X_i[:, q] \leftarrow \left(\prod_{j \leq 3} \|X_j[:, q]\|_2^{\frac{1}{3}} \right) \frac{X_i[:, q]}{\|X_i[:, q]\|_2}.$$

An example barebone implementation is shown below.

```
import matplotlib.pyplot as plt
import tensorly as tl
from copy import deepcopy
from tensorly.solvers.penalizations import scale_factors_fro

def optimal_balancing(factors):
    rank = factors[0].shape[1]
    beta = []
    for q in range(rank):
        beta.append(tl.prod([tl.norm(factor[:,q]) ** (1 / 3) for factor in factors]))
```

(continues on next page)

(continued from previous page)

```
    balanced_factors = []
    for i in range(len(factors)):
        balanced_factor = factors[i].copy()
        for q in range(rank):
            balanced_factor[:, q] =  factors[i][:, q] * beta[q] / tl.
↪norm(factors[i][:, q])
        balanced_factors.append(balanced_factor)
    return balanced_factors
```

The simulation below illustrates the importance of balancing initial (and sometimes final) estimates. In [Cohen and Leplat, 2025], we also show that balancing at every outer iteration in the HALS algorithm further improves the convergence speed empirically.

Recall the ridge-regularized nCPD problem

$$\underset{X_i \geq 0}{\operatorname{argmin}} \|\mathcal{T} - \mathcal{J}_r \times_1 X_1 \times_2 X_2 \times_3 X_3\|_F^2 + \mu \left(\|X_1\|_F^2 + \|X_2\|_F^2 + \|X_3\|_F^2\right).$$

The reader can play around with the regularization value: high values lead to more component pruning, but the effect of balancing is less pronounced; low values lead to no component pruning, but the importance of balancing is more visible, even after the optimization algorithm has converged.

Also note that balancing only optimizes the factor scales with respect to the regularization terms. To scale the factors also according to the data fitting term and therefore start the algorithm at the best scaled position, it can be useful to scale the initial guess by solving the polynomial minimization problem

$$\underset{\lambda \geq 0}{\operatorname{argmin}} \|\mathcal{T} - \lambda^3 \mathcal{J}_r \times_1 X_1 \times_2 X_2 \times_3 X_3\|_F^2 + \mu\lambda^2 \left(\|X_1\|_F^2 + \|X_2\|_F^2 + \|X_3\|_F^2\right),$$

which amounts to evaluating the cost at all the positive roots of the polynomial

$$P(\lambda) = -6\lambda^2 \langle \mathcal{T}, \mathcal{J}_r \times_1 X_1 \times_2 X_2 \times_3 X_3 \rangle + 6\lambda^5 \|\mathcal{J}_r \times_1 X_1 \times_2 X_2 \times_3 X_3\|_F^2 + 2\mu\lambda \left(\|X_1\|_F^2 + \|X_2\|_F^2 + \|X_3\|_F^2\right).$$

```
# Example usage of the optimal_balancing function
rank = 3
dims = [10, 11, 12]
ndims = len(dims)
noise = 0.1
itermax = 100

# Regularization hyperparameter
ridge_reg = 0.1

# Create random factors for a 3-way tensor decomposition
np.random.seed(22)  # For reproducibility
true_factors = [tl.tensor(np.random.rand(dims[i], rank)) for i in range(ndims)]

# Creating the data (from factors or balanced factors, it leads to the same result)
CPtensor = tl.cp_tensor.CPTensor(([10**(-i) for i in range(ndims)], true_factors))
data = CPtensor.to_tensor() + noise * tl.tensor(np.random.randn(*dims))  # Adding␣
↪some noise

# Initialization
rank_e = rank + 3  # overestimating rank for initialization
init = [10**(-i) * tl.tensor(np.random.rand(dims[i], rank_e)) for i in range(ndims)]
CPinit = tl.cp_tensor.CPTensor((None, init))
```

(continues on next page)

(continued from previous page)

```
# Optimal scaling of the initialization
t_init_unscaled = CPinit.to_tensor()
CPinit_scaled, scale = scale_factors_fro(CPinit, data, [ridge_reg]*ndims, [0]*ndims,␣
↪nonnegative=True)
init_scaled = CPinit_scaled.factors

# Balancing the unscaled initialization factors
balanced_init = optimal_balancing(init)
CPinit_balanced = tl.cp_tensor.CPTensor((None, balanced_init))

# Balancing the scaled initialization factors
balanced_scaled_init = optimal_balancing(init_scaled)
CPinit_scaled_balanced = tl.cp_tensor.CPTensor((None, balanced_scaled_init))
```

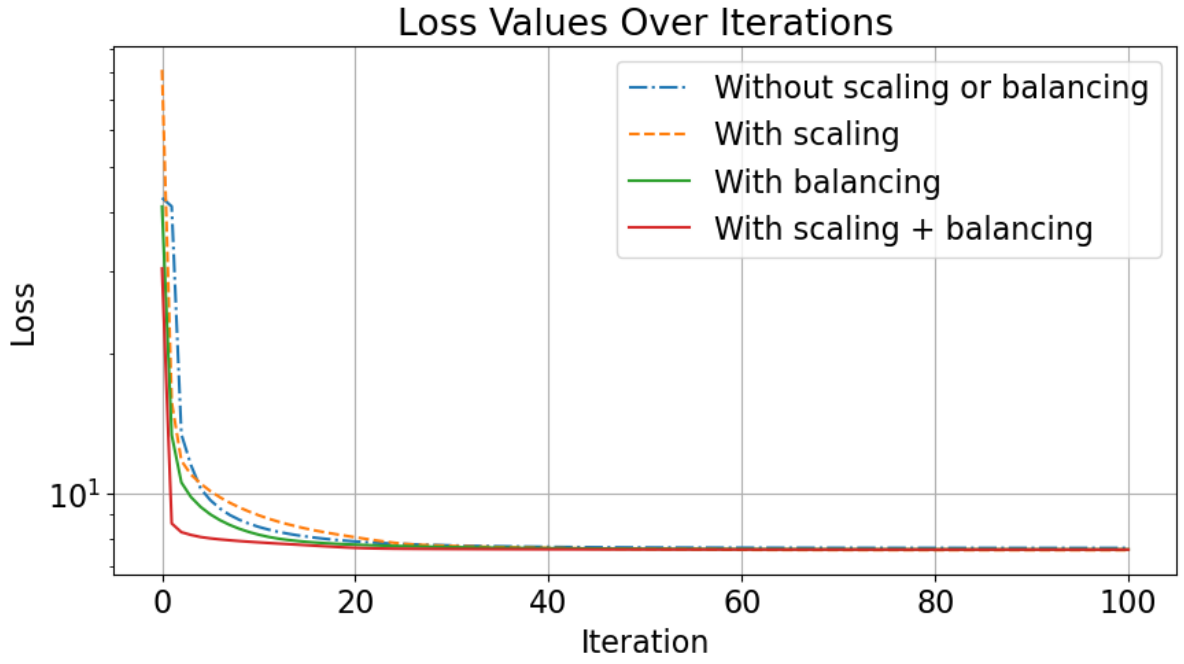
```
Optimal scaling factors: 5.290233987029234
Initial factors:
Factor 0: shape (10, 6), norm 4.299994791774854
Factor 1: shape (11, 6), norm 0.4074799825930501
Factor 2: shape (12, 6), norm 0.04968474562511675

Scaled factors:
Factor 0: shape (10, 6), norm 22.74797859149603
Factor 1: shape (11, 6), norm 2.1556644529478346
Factor 2: shape (12, 6), norm 0.2628439299428947

Balanced factors:
Factor 0: shape (10, 6), norm 0.43822144179578987
Factor 1: shape (11, 6), norm 0.43822144179579
Factor 2: shape (12, 6), norm 0.4382214417957899

Balanced scaled factors:
Factor 0: shape (10, 6), norm 2.3182939652330408
Factor 1: shape (11, 6), norm 2.3182939652330408
Factor 2: shape (12, 6), norm 2.3182939652330408
```

```
# Fetching the loss values with callback
callback_loss = []
def callback(factors, unnormalized_rec_errors):
    loss = (unnormalized_rec_errors**2)/2 + sum([ridge_reg*tl.norm(factors[1][i])**2␣
↪for i in range(3)])
    callback_loss.append(loss)

# Running nonnegative CP decomposition with various balancing/scaling setups
from tensorly.decomposition import non_negative_parafac_hals

CPe = non_negative_parafac_hals(data, rank_e, n_iter_max=itermax, tol=0,␣
↪init=deepcopy(CPinit), verbose=False, ridge_coefficients=ridge_reg,␣
↪callback=callback)
loss = np.copy(callback_loss)

callback_loss = []
CPe_scaled = non_negative_parafac_hals(data, rank_e, n_iter_max=itermax, tol=0,␣
↪init=deepcopy(CPinit_scaled), verbose=False, ridge_coefficients=ridge_reg,␣
↪callback=callback)
```

(continues on next page)

(continued from previous page)

```
loss_scaled = np.copy(callback_loss)

callback_loss = []
CPe_balanced = non_negative_parafac_hals(data, rank_e, n_iter_max=itermax, tol=0,␣
↪init=deepcopy(CPinit_balanced), verbose=False, ridge_coefficients=ridge_reg,␣
↪callback=callback)
loss_balanced = np.copy(callback_loss)

callback_loss = []
CPe_scaled_balanced = non_negative_parafac_hals(data, rank_e, n_iter_max=itermax,␣
↪tol=0, init=deepcopy(CPinit_scaled_balanced), verbose=False, ridge_
↪coefficients=ridge_reg, callback=callback)
loss_scaled_balanced = callback_loss
```



We can also look at the loss function with and without balancing of the final estimates to observe that balancing reduces the final error slightly, and the effect is more prononced for the output of algorithms that did not scale and balance the initial guess.

```
# Balancing the final estimated factors
CPe_fbalance = tl.cp_tensor.CPTensor((None, optimal_balancing(CPe.factors)))
CPe_scaled_fbalance = tl.cp_tensor.CPTensor((None, optimal_balancing(CPe_scaled.
↪factors)))
CPe_balanced_fbalance = tl.cp_tensor.CPTensor((None, optimal_balancing(CPe_balanced.
↪factors)))
CPe_scaled_balanced_fbalance = tl.cp_tensor.CPTensor((None, optimal_balancing(CPe_
↪scaled_balanced.factors)))

def loss(data, CPestim, ridge_reg):
    regs = sum([ridge_reg * tl.norm(factor) ** 2 for factor in CPestim.factors])
    return 0.5 * (tl.norm(data - CPestim.to_tensor()) ** 2) + regs
```

```
Final Loss values before | after rebalancing:
Without balancing or scaling:
7.640703504963442 | 7.640136375087222
With scaling:
7.558884490291016 | 7.558384623717727
With balancing:
7.58741959225447 | 7.5869139361472975
With scaling and balancing:
7.581340753614704 | 7.581238791301618
```

CHAPTER

ELEVEN

IDENTIFIABILITY OF COMPLETE DICTIONARY LEARNING

Reference

[Cohen and Gillis, 2019] J. E. Cohen, N. Gillis, "Identifiability of Complete Dictionary Learning", SIAM Journal of Mathematics on Data Science, vol.1 issue 3, pp. 518-536, 2019. pdf`

A core issue in source separation models is ensuring that a user can uniquely identify the output of a separation algorithm with the ground-truth sources. We say that a model is identifiable if it features this uniqueness property. To illustrate what a lack of identifiability means, let a nonnegative matrix X be generated as the product of nonnegative matrices WH. If an NMF $X = \tilde{W}\tilde{H}^T$ is computed, in general, the estimated sources $\tilde{W}, \tilde{H}$ are not essentially the same as the ground truth sources W, H (*i.e.*, up to permutation and scaling ambiguities), see *Low-rank matrix and tensor models*. In what follows, we summarize our contribution [Cohen and Gillis, 2019], in which we study the identifiability of cDL. Note that this work has been extended for sparse NMF [Abdolali *et al.*, 2024, Abdolali and Gillis, 2021].

cDL is a sparse matrix factorization model, where n vectors $M[:, i]$ of dimensions p stacked in a matrix $M \in \mathbb{R}^{p\times n}$ are decomposed as a sparse combinations with at most k nonzero coefficients of r vectors $D[:, q]$ called atoms and stacked in a dictionary matrix $D \in \mathbb{R}^{p\times r}$. cDL can be formalized as a matrix factorization problem,

$$\text{Find } D \in \mathbb{R}^{p\times r} \text{ and } B \in \mathbb{R}^{r\times n} \text{ such that } M = DB \text{ and } \forall i \leq n,\ \|B[:, i]\|_0 \leq k,$$

where matrix B is the coefficients matrix. We consider the blind setup where both D and B are unknown. In short, the goal of cDL is to estimate the dictionary D and the coefficients matrix B solely from the knowledge of X, the number of atoms r, and the sparsity level k. In what follows, we set the dimensions such that $p \leq r$ and suppose that D has full column rank. This means that the dictionary is not over-complete (which is a usual setup of interest, but harder to study), thus the name cDL. In this setup, without loss of generality in the noiseless setting, we can assume $p = r$ and will do so in the following.

11.1 Identifying $r - 1$ dimensional facets

Our identifiability result relies on the geometric interpretation of cDL. cDL can be seen as a subspace clustering problem, since the data points $M[:, i]$ all lie on the union of subspaces of dimension at most k. Indeed, denoting S_i the support of column $B[:, i]$, any data point writes as a linear combination of at most k atoms

$$M[:, i] = \sum_{q\leq k} D[:, S_i[q]]B[S_i[q], i].$$

Since B is unknown, so is the support S_i but we still know that $M[:, i]$ must live in one of the subspaces spanned by any k columns of D, therefore for any $i \leq n$,

$$M[:, i] \in \cup_{\#S\leq k} \text{col}\left(D[:, S]\right).$$

Informally, our theorem states the following. Suppose there exists a cDL factorisation $M = DB$. Then for any sparsity value k, the dictionary atoms define $r-1$-dimensional subspaces, called facets, $F_i = \text{col}(D[:, -i])$, that contain the data points. These facets can be used to uniquely recover the atoms in dictionary D up to scalings using

$$D[:, i] = \cap_{j \leq i} F_i.$$

If the sparsity level k is smaller than $r-1$, these facets might seem oversized tools for studying cDL, since the data points $M[:, i]$ then belong to $k < r-1$-dimensional subspaces. It turns out they are just the right objects to study for identifiability. Indeed, if a unique set of facets can be collected that covers the data points, then D is identifiable (which is not true for lower-dimensional subspaces). The proof idea is straightforward: count how many data points with full dimensionality lie on each facet, and verify that there are enough to uniquely characterize each facet. We can optimize this count using the fact that data points $M[:, i]$ lie on at least $r-k$ facets F_i simultaneously. The minimal number of points we need to identify a single facet is obtained by saturating all facets with as many points as possible, while no facet is identifiable (which happens when a facet contains $r-1$ points). This leads to the following result.

Theorem 13 (Identifiability of cDL)

A cDL $M = DB$ is essentially unique if on each facet F_i there are $\lfloor \frac{r(r-2)}{r-k} \rfloor + 1$ data points with spark r.

This is maybe better understood with a visualisation. First, note that we may normalize the dictionary D columnwise without loss of generality, because of scale invariance in the product DB. Therefore, we may suppose that atoms live on the simplex. Furthermore, we can normalize the columns of M by the ℓ_1 norm, and it can be shown [Gillis, 2020] that this amounts to normalizing the columns of B by the ℓ_1 norm as well. In other words, for an r-dimensional cDL problem, we can assume that the data points and the atoms are located on the $r-1$ dimensional simplex. This means we can visualize the case $r = 3$ in two dimensions, which is convenient. In the figures below, we set $k = 2$, and the white circles and squares are two sets of correct dictionary atoms $D[:, i]$ that may be used to write the data point $M[:, i]$ (black dots) as a 2-sparse combination of these atoms.

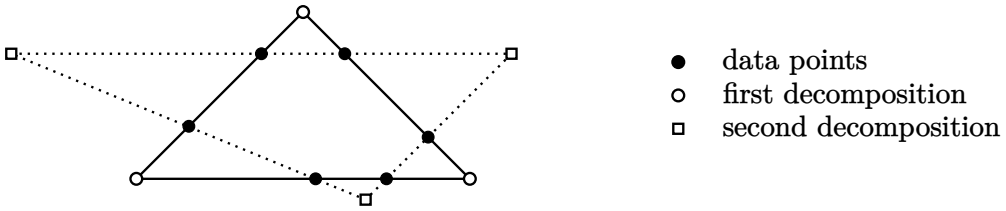


Fig. 11.1: Two points per facet.

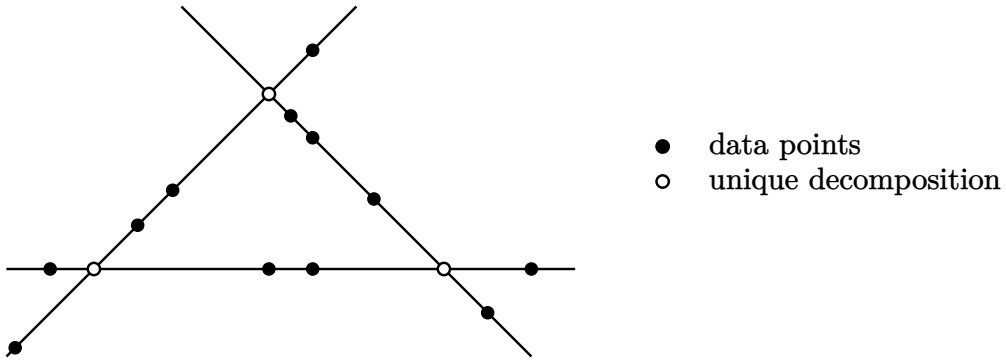


Fig. 11.2: Four points per facet.

The first case is not identifiable because there are only two points per facet, while in the second example, four points per facet ensure identifiability. This is proven by *Theorem 13* since in three dimensions with $k = 2$ and $r = 3$, $\frac{r(r-2)}{r-k}$ amounts

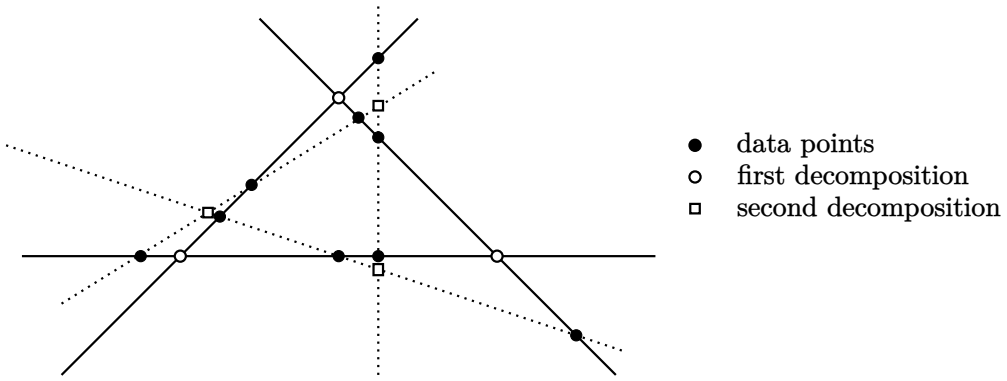


Fig. 11.3: Three points per facet, not identifiable.

to 3. In the third case, we were unlucky; generically, there should be a single solution, but we carefully chose a setup in which the data points lie at the intersection of two triangles.

11.2 Relation to existing results

Our result complements the existing literature in several ways:

- It contradicts an existing result from Georgiev et. al. [Georgiev *et al.*, 2005]. These authors incorrectly assumed that facets can be uniquely identified using r data points (when $k = r - 1$), but we provide a counterexample in Fig. 11.3. When $r = 3$ and $k = 2$, we need in fact $r + 1$ data points on the facets.
- It is fully deterministic, therefore it holds with slightly more generality than many existing results based on a Bayesian description of the model, see for instance [Gribonval *et al.*, 2015] for an overview.

11.3 Computation of complete Dictionary Learning with tensorly

It is possible to compute a cDL with tensorly using the constrained CP decomposition function. Indeed, one may compute a solution to the cDL problem by solving the optimization problem

$$\min_{D,B} \|M - DB\|_F^2 + \eta_{\|B[:,i]\|_0 \leq k}.$$

Tensorly implements an AO algorithm where each block (here the dictionary D and the coefficients X respectively) are updated using a few iterations of Alternating Descent Method of Multipliers (ADMM). This algorithm was proposed as a flexible optimization framework for constrained matrix and tensor problems by Huang, Sidiropoulos and Liavas [Huang *et al.*, 2016] and implemented in tensorly in collaboration with Caglayan Tuna in PR#284.

This algorithm is demonstrated below on a simulated example. Convergence speed and runtime can be improved using dedicated software and algorithms.

```
import numpy as np
import tensorly as tl
from tensorly.decomposition._constrained_cp import constrained_parafac
from tensorly.cp_tensor import cp_permute_factors

# ------------- Parameters ---------------
k = 3 # true number of nonzeros in columns of X
kest = 3 # number of nonzeros in the computed solution Xe
```

(continues on next page)

(continued from previous page)

```
noise = 0 # how much noise in the data
rank = 8 # rank of the factorization (number of atoms)
sig = 0.02 # how far from the solution we initialize
oversampling = 2.0 # oversampling factor, <1. has lots of chances to fail

# Getting dimensions from Theorem 1 bound
bound_theorem1 = (np.floor(rank*(rank-2)/(rank-k))+1)*(rank/(rank-k))
print(f"Theorem 13 ensures identifiability with more than {bound_theorem1} data␣
↪samples")
dims = [rank, int(np.floor(oversampling*bound_theorem1))]
print(f"There are {dims[1]} data samples used in this experiment")

# ------------ Data generation -----------
D = np.random.randn(dims[0],rank)
D = D/tl.norm(D,axis=0)
X = np.random.randn(rank,dims[1])
# sparsify X, Bernouilli Gaussian model
for i in range(X.shape[1]):
    X[:-k,i] = 0
    np.random.shuffle(X[:,i])

M = D@X
Mnoise = M + noise*np.random.randn(*M.shape)

# Init close to solution
D0 = D+sig*np.random.randn(*D.shape)
X0 = X+sig*np.random.randn(*X.shape)
init = (None,[D0,X0.T])

# ------- Decomposition with tensorly -----
out, err = constrained_parafac(Mnoise, rank, hard_sparsity_rowwise={1:kest},␣
↪verbose=False, init=init, n_iter_max=500, return_errors=True, tol_outer=0)
print(f"Initial cost was {err[0]}, Final cost is {err[-1]}, {len(err)} iterations␣
↪were used")
# postprocess estimate by permuting the components optimally
try:
    out, _ = cp_permute_factors((None,[D,X.T]), out)
except:
    print("no permutation can be computed because a zero component is present in the␣
↪true or estimated coefficients")
Xe = out[1][1].T
De = out[1][0]

# ----------- Error metrics --------------
# Computing True False Positives, True False Negatives
tp = tl.sum((X!=0) & (Xe!=0))
fp = tl.sum((X==0) & (Xe!=0))
fn = tl.sum((X!=0) & (Xe==0)) # always equal to fp if k is chosen optimally
tn = tl.sum((X==0) & (Xe==0))
# Precision, Recall, Accuracy, Fmetric
precision = tp/(tp+fp)
recall = tp/(tp+fn)
accuracy = (tp+tn)/(tp+tn+fn+fp)
fmet = 2*precision*recall/(precision+recall)
print(f"Support estimation: Precision {precision:.2f}, Recall {recall:.2f}, F metric
↪{fmet:.2f}, Accuracy {accuracy:.2f}")
```

(continues on next page)

(continued from previous page)

```
# Estimation of D and X
Denorms = tl.norm(De, axis=0)
De = De/Denorms
Xe = (Denorms*Xe.T).T
rmse_D = tl.norm(D-De)/tl.sqrt(tl.prod(D.shape))
rmse_X = tl.norm(X-Xe)/tl.sqrt(tl.prod(X.shape))
print(f"Relative error on the dictionary: {rmse_D}, on the sparse factor: {rmse_X}")
```

```
Theorem 13 ensures identifiability with more than 16.0 data samples
There are 32 data samples used in this experiment
```

```
Initial cost was 0.00991585462821329, Final cost is 0.00028673444509207563, 500␣
↪iterations were used
Support estimation: Precision 0.99, Recall 0.99, F metric 0.99, Accuracy 0.99
Relative error on the dictionary: 9.822218589886557e-05, on the sparse factor: 0.
↪0006667703985511948
```

Feel free to play with the parameters. One may observe for instance that as soon as the initial factors are chosen too far from the true solution, most of the time of support is poorly estimated. Similarly, the true support is only recovered in small noise regimes. These two properties, respectively local convergence and robustness, have been studied in the literature, see for instance [Liang *et al.*, 2022] and [Gribonval *et al.*, 2015] and references therein. However the bounds on the initial error and noise levels are in general hard to compute explicitly.

We can illustrate the previously presented result on identifiability with this simulation. The data points are located uniformly on each facet with sparsity exactly k. Each facets requires $\lfloor \frac{r(r-2)}{r-k} \rfloor + 1$ points at least and there are $r - k$ facets. Each point lives in k facets. Therefore we need more than $(\lfloor \frac{r(r-2)}{r-k} \rfloor + 1)\frac{r}{r-k}$ data points to satisfy *Theorem 13*.

In the above simulation, setting the noise level to zero, if there are fewer points than the bound of *Theorem 13*, the dictionary and the sparse factors are never recovered because of a lack of identifiability. This is more visible when starting midly far from the true solution. Indeed when the model is not identifiable, the true solution is still a minimizer of the problem, but there are generally infinitely many solutions which may be obtained instead. Try running the simulation several times with the following parameter set:

```
k = 3 # true number of nonzeros in columns of X
kest = 3 # number of nonzeros in the computed solution Xe
noise = 0 # how much noise in the data
rank = 8 # rank of the factorization (number of atoms)
sig = 0.03 # how far from the solution we initialize
oversampling = .8 # oversampling factor, <1. has lots of chances to fail
```

and observe that the RMSE are generally not small, even when perfect support estimation occurs.

When more samples are drawn, because of the stochastic nature of the repartition of samples on the facets, it is possible that the model is still not identifiable, but as the number of points grows this becomes less likely. The algorithm still falls in local minimizers, but sometimes it finds a solution very close to the true, global solution. Try running the simulation with the same parameters but more samples:

```
oversampling = 2.
```

and notice how the RMSE on the dictionary and factor are often smaller then in the undersampled regime, and sometimes

close to machine precision.

CHAPTER

TWELVE

DICTIONARY-BASED LRA

Reference

[Cohen and Gillis, 2018] J. E. Cohen, N. Gillis, "Dictionary-based Tensor Canonical Polyadic Decomposition", IEEE Transactions on Signal Processing, vol.66 issue 7, pp. 1876-1889 2018. pdf

[Cohen and Gillis, 2018] J. E. Cohen, N. Gillis, "Spectral Unmixing with Multiple Dictionaries", IEEE Geoscience and Remote Sensing Letters, vol.15 issue 2, pp. 187-191, 2018. pdf

[Cohen, 2022] J. E. Cohen, "Dictionary-based low-rank approximation and the mixed sparse coding problem", Frontiers in Applied Mathematics and Statistics, 2022. frontiers arxiv codes

[Cohen, 2020] J. E. Cohen, "Computing the proximal operator of the l1 induced matrix norm", arxiv 2005.06804, 2020

Note: the publication in the Frontiers journal was in a special issue with editors that I know and trust, with the intention to start a new journal dedicated to tensors. The review process was comparable to that of a medium-level publication venue. I would not repeat the experiment, and the editors have since stopped collaboration with Frontiers.

When decomposing matrices and tensors, a reasonable assumption is that the factor matrices, although unknown, are generated from a known set of templates. Let us provide two typical examples using a simple matrix factorization $Y = AB^T.

- Factors contain smooth components, say the first factor matrix A. A simple way to account for smoothness in a decomposition is to fix a set of smooth splines stored as columns of a dictionary matrix D, and assume that each component writes as a parsimonious sum of these splines. Each column of factor A then writes $A = DX$ with $\|X[:,i]\|_0 \leq k$ for each atom i, where the integer k is the number of smooth splines that can add up to build each component.
- The components are contained in a large dictionary D, but it is not known which column of $D \in \mathbb{R}^{m\times d}$ must be picked exactly. This can be challenging, in particular when the number of atoms d is much larger than the decomposition rank r. Again in the case of factor matrix A, this translates into a relationship $A[:,i] = D[:,\sigma_i]$ for all column index i, where σ is an injection of $[1,d]$ into $[1,r]$. Equivalently, one may write $A = DX$ with $\|X[:,i]\|_0 = 1$ for all atoms i and $X \in \{0,1\}^{m\times d}$. This may happen in applications such as spectral unmixing, where the components in the spectral mode represent reflectance spectra that are typically known, whereas the components present in the data may be unknown. The decomposition problem is to unmix the sources and identify their components using the known spectral signatures stored in the dictionary.

An optimization problem that formalizes both these problems, using the Frobenius norm as a metric, writes

$$\min_{B\in\Omega_B,\ X\in\Omega_X,\ \|X[:,i]\|_0\leq k\ \forall i\leq r} \|Y - DXB^T\|_F^2,$$

where k is the known sparsity level of columns of the matrix X. The matrix B lives in a set Ω_B that we may adapt so that the above problem covers both tensor decompositions and matrix decompositions. For instance, by imposing that B

is the Khatri-Rao product of two factor matrices, in the case of order-three CP decomposition. Similarly, the constraint set Ω_X is used to impose additional constraints on the matrix X, such as elementwise nonnegativity. This problem is referred to as the Dictionary-based Low-Rank Approximation (DLRA) in the following.

One may observe that this minimization problem, in general, is difficult. It is a generalization of sparse coding, which is obtained when B is set to the identity and $r = 1$; therefore DLRA is NP-hard. Moreover, it generalizes matrix factorization problems such as NMF when D is the identity, $k = m$, and the constraint sets are the nonnegative orthants. Even if we know the support of the solution X, one still needs to estimate both the nonzeros in X and the right-hand-side matrix B, which is challenging [Zheng *et al.*, 2023].

Therefore, to better understand the nature of the solutions to DLRA and to design efficient heuristics to solve this problem, we have studied three particular cases in three different publications:

- The special case of *1-sparse DLRA* when $k = 1$ [Cohen and Gillis, 2018]
- The subproblem of *estimating factor matrix* X with fixed B [Cohen, 2022], and its usage in solving the DLRA problem with alternating optimization.
- *A third contribution* [Cohen and Gillis, 2018] has also been proposed in the context of spectral unmixing under the pure pixels assumption.

These first two problems are studied in the following subsections. Python packages for running experiments are located in two dedicated repositories: dlra and dlraos-essentials.

12.1 One-sparse dictionary-based LRA

Reference

[Cohen and Gillis, 2018] J. E. Cohen, N. Gillis, "Dictionary-based Tensor Canonical Polyadic Decomposition", IEEE Transactions on Signal Processing, vol.66 issue 7, pp. 1876-1889 2018. pdf

Note: Out of simplicity, we present only the matrix case, but both one-sparse dictionary-based matrix and tensor low-rank decompositions are studied in our work [Cohen and Gillis, 2018]. In particular, we demonstrate that one-sparse DLRA is a well-posed approximate tensor decomposition problem, in contrast to approximate CPD. Also note that this work has been studied against global optimization solutions in 2023 [Dache *et al.*, 2023].

We are interested in solving the following matrix factorization problem:

$$\min_{B \in \Omega_B, \mathcal{K} \in [1,d]^r} \|Y - D[:, \mathcal{K}]B^T\|_F^2$$

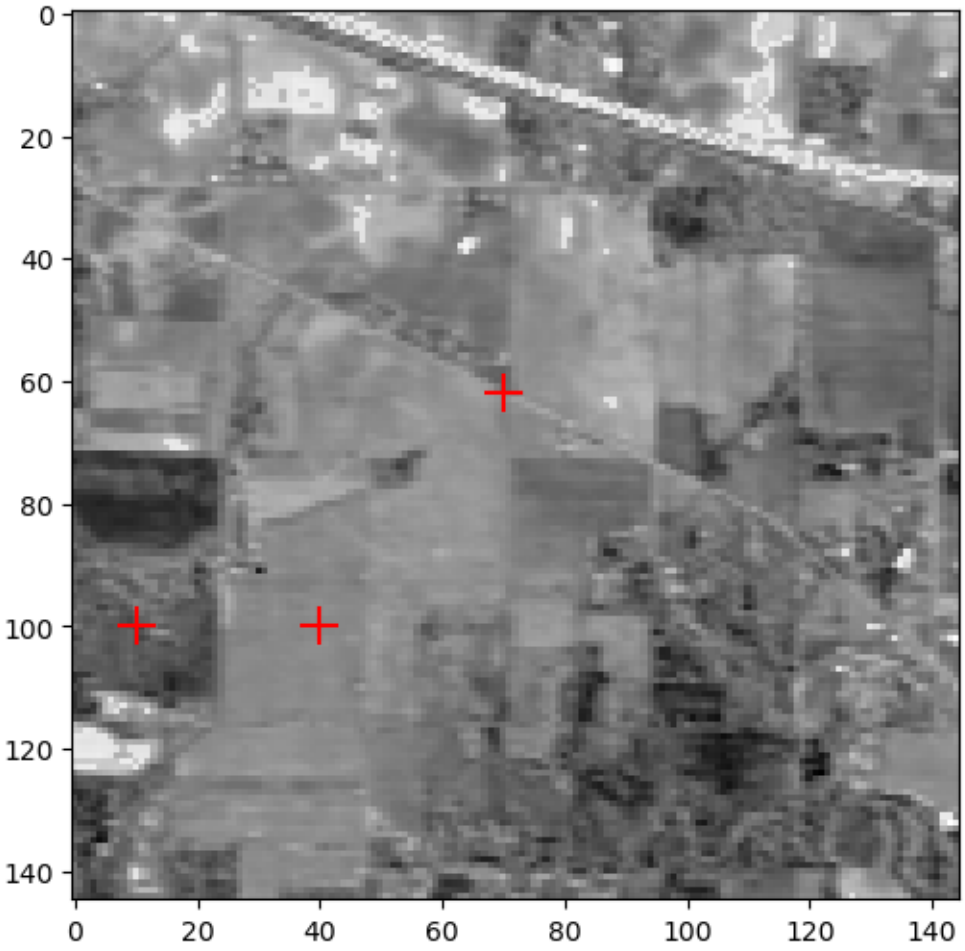
where Ω_B is typically $\mathbb{R}^{n \times r}$ or $\mathbb{R}_+^{n \times r}$ with $r \leq m, n$.

This problem, coined one-sparse DLRA, consists of finding the columns of the dictionary D indexed in the set $\mathcal{K}$ such that the data matrix Y is well approximated by a low-rank model $D[:, \mathcal{K}]B^T$. One can also rewrite one-sparse DLRA using a selection binary matrix S whose columns have a single nonzero element at each $\mathcal{K}[q]$ for $q \in [1, r]$, this yields $D[:, \mathcal{K}]B^T = DSB^T$.

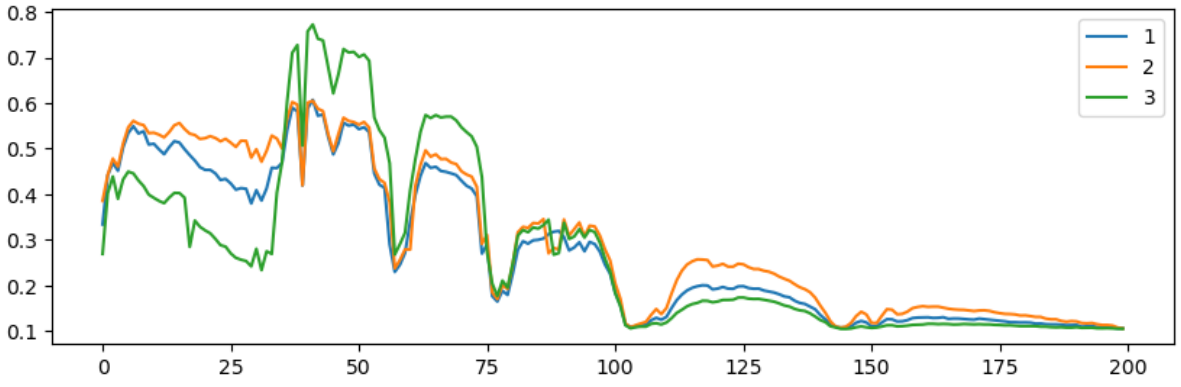
The code snippet below shows how to generate such a data matrix Y without noise. We use a hyperspectral image available in `tensorly.dataset`, and assume that a few pixels are pure. This means that the source matrix A is contained in the data itself; in other words, $A = D[:, \mathcal{K}]$ but $\mathcal{K}$ is unknown.

```
import numpy as np
import tensorly as tl
from tensorly.datasets import data_imports
import matplotlib.pyplot as plt
from tensorly_hdr.image_utils import convert_to_pixel, convert_to_index

# Import an hyperspectral image
image_bunch = data_imports.load_indian_pines()
image_t = image_bunch.tensor # x by y by wavelength
image_t = image_t/tl.max(image_t) # maximum value set to 1

# Vectorize the tensor into a m\times n matrix (wavelength \times pixels, y vary␣
↪first)
image = tl.unfold(image_t, 2)

# Simulation parameters
# For illustration purposes, we can pick a few spectra and build a synthetic data␣
↪matrix accordingly.
band = 70
rank = 3
# hand-picked road, field, and forest
Kx = [70,40,10]
Ky = [62,100,100]
K = convert_to_index(Kx, Ky, image_t.shape[1])
```

```
Showing the hyperspectral image at band indexed by 70
Red crosses signal the position of the hand-picked pure pixels
```




```
Showing 3 pixels (spectra) indexed by [9060, 14540, 14510]
```




```
from tensorly_hdr.image_utils import sparsify
Bt = np.random.rand(rank, image.shape[1]) # elementwise nonnegative
# We sparsify B so that not all pixels use all three spectra, this is optional
Bt = sparsify(Bt, s=0.5)
Y = image[:,K]@Bt
```

The columns of the matrix Y are now indeed linear combinations of only three columns in the dictionary D (which is the

original hyperspectral image itself in this particular case). The DLRA problem consists of finding the pure pixel indices $\mathcal{K}$ and the abundances B^T.

Note: The one-sparse DLRA model under scrutiny in this section has strong connections to other matrix and tensor factorization models, most prominently Multiple Measurements Vector (MMV), also known as collaborative sparse coding. The main different is that MMV does not assume that the data is low-rank. On the other hand, MMV transforms the bivariate problem into a single convex problem over the SB^T matrix, which can be solved optimally but typically involves significantly more variables and requires guarantees for the correctness of the convex relaxation.

12.1.1 Identifiability of one-sparse DLRA

There are at least two questions of interest regarding one-sparse DLRA:

- In general, low-rank approximations are not interpretable because of the rotation ambiguity. For the one-sparse DLRA model, does the sparsity constraint lead to identifiability?
- May an algorithm be designed that finds optimal solutions in a reasonable time?

We first answer the identifiability question positively under mild conditions on the dictionary.

Theorem 14

Let Y be a real $m \times n$ matrix of rank r, and let D be a real $m \times d$ matrix with spark$(D) > r$. If there exists a full column rank permutation matrix S (columns are one sparse and entries are binary) and a matrix $B \in \mathbb{R}^{n \times r}$ with nonzero columns such that $Y = (DS)B^T$, then S and A are unique up to permutation ambiguity.

See [Cohen and Gillis, 2018] for the proof.

Remark

Recall that spark(D) is the minimum integer k such that at least one subset of k columns of matrix D is rank-defficient.

In other words, for a well-built dictionary with incoherent atoms, the dictionary constraint leads to uniqueness of the matrix factorization problem.

12.1.2 A simple heuristic for one-sparse DLRA

Designing an optimization algorithm with performance guarantees for one-sparse DLRA is not straightforward since one-sparse DLRA is a generalization of sparse coding (sparse coding is recovered when $n = 1$). While we can consider convex relaxations for the sparsity constraints, empirically, we found that a heuristic based on ALS works rather well. In what follows, we introduce this heuristic, which we coin Maximum Correlation ALS (MC-ALS), and showcase its performance in spectral unmixing. However, we defer the formal analysis of MC-ALS performance to the *k-sparse DLRA problem*.

Remark

The original name of MC-ALS in [Cohen and Gillis, 2018] was Matching Pursuit ALS, which is somewhat misleading, since it involves only one Matching Pursuit iteration and the overall algorithm is not greedy.

MC-ALS works as follows: we view the submatrix $D[:, \mathcal{K}]$ as factor matrix A, and compute the least squares estimate

$$\hat{A} = \underset{A}{\operatorname{argmin}} \|Y - AB^T\|_F^2.$$

If, in the application at hand, the matrix A is nonnegative (like the spectra of our running hyperspectral image example), we may use a *NNLS* solver instead.

Then given an estimation $\hat{A}$, the closest atom in the dictionary D is computed for each column of $\hat{A}$

$$\mathcal{K}[i] = \underset{j \leq d}{\operatorname{argmin}} |D[:, j]^T \hat{A}[:, i]|.$$

Remark

This formula assumes that the dictionary has ℓ_2-norm normalized columns.

Note: Using the largest absolute scalar product between the atoms and the estimated factor matrix $\hat{A}$ may lead to selecting the same atom twice, *e.g.*, if two columns of $\hat{A}$ are similar. This problem can be alleviated by rather selecting indices $\mathcal{K}$ as the optimal linear assignement of atoms in the dictionary D to columns of matrix $\hat{A}$:

$$\mathcal{K} = \underset{\mathcal{K}}{\operatorname{argmin}} \operatorname{Tr}(|D[:, \mathcal{K}]^T A|)$$

This problem is solved efficiently using, for instance, the Hungarian algorithm [Kuhn, 1955]. In the original work [Cohen and Gillis, 2018], we did not use a linear assignment problem solver, but we do here because it avoids the singularity of the estimated factor.

This yields the following routine to estimate the matrix A given an estimate of matrix B.

```
from scipy.optimize import linear_sum_assignment
# scipy is a dependency of tensorly

def MC_ALS_Aestimation(Y, D, B, nonnegative=True):
    # Should ensure the dictionary has normalized columns
    if nonnegative:
        Ahat = tl.solvers.hals_nnls(B.T@(Y.T), B.T@B, n_iter_max=20).T
    else:
        Ahat = tl.solve(B.T@B,B.T@(Y.T)).T
    K, _ = linear_sum_assignment(tl.abs(D.T@Ahat), maximize=True)
    Ahat2 = D[:,K]
    return K, Ahat2, Ahat

D = image/np.linalg.norm(image,axis=0)
B = Bt.T
out = MC_ALS_Aestimation(Y, D, B)
print(f"Selected pixels by MC-ALS: {list(out[0])}, ground truth: {K}")
```

```
Selected pixels by MC-ALS: [9060, 14510, 14540], ground truth: [9060, 14540, 14510]
```

In the code snippet above, we can recover the indices of the atoms that we hand-picked earlier. However, we assumed that the matrix B is known, and no noise was added to the system.

The full algorithm requires estimating the matrix B. Because we work here with hyperspectral images and both spectra and abundances are nonnegative, we use a NNLS solver to estimate B rather than ordinary least squares.

```
def MC_ALS(Y, D, B_0, itermax=100):
    errs = []
    B = np.copy(B_0)
    for it in range(itermax):
        # estimate A
        K, A, _ = MC_ALS_Aestimation(Y, D, B)
        # estimate B
        #B = tl.solve(A.T@A, A.T@Y).T
        B = tl.solvers.hals_nnls(A.T@Y, A.T@A, n_iter_max=20).T
        # compute error
        errs.append(tl.norm(Y - A@B.T)**2/tl.prod(Y.shape))
        # stop if error has not changed at all (algorithm is stuck)
        if it>1 and errs[-2]==errs[-1]:
            print("Final estimated indices", K)
            print("MSE", errs[-1])
            break
    return (None, [A, B]), errs, K

out = MC_ALS(Y, D, np.random.rand(*B.shape))
```

```
Final estimated indices [ 4116 14510 14540]
MSE 7.602023946323768e-06
```

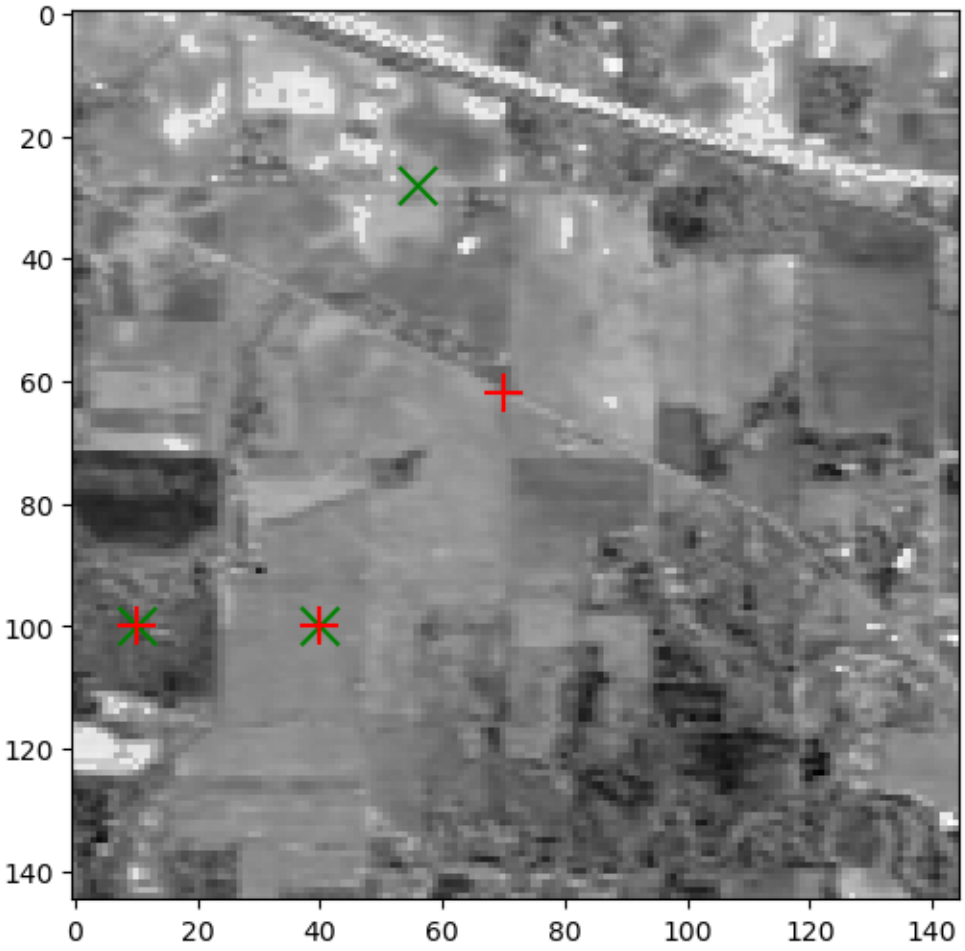
We can compare the indices of the atoms selected by MC_ALS with the (synthetic) ground truth.

```
print(K, out[2])
```

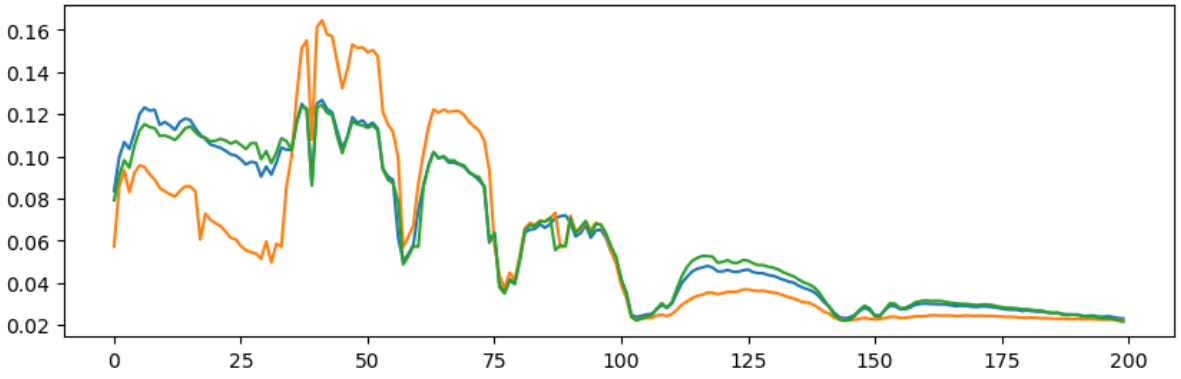
```
[9060, 14540, 14510] [ 4116 14510 14540]
```

If one runs the experiment several times, one might observe that the ground truth is often only partially recovered, even in the absence of noise. Moreover, the results are surprisingly consistent across runs. We can place the selected pixels on the hyperspectral image (in green) and plot the corresponding spectra. Notice how the selected pixels sometimes still appear to belong to the same class as the ground truth (road, forest, field).

```
Kxe, Kye = convert_to_pixel(out[2], image_t.shape[1])
plt.figure(figsize=(8,6))
plt.imshow(image_t[:,:,band], cmap="Greys")
# Also showing the pixels we hand-pick as pure
plt.scatter(Kxe, Kye, s=200, c='green', marker='x')
plt.scatter(Kx, Ky, s=200, c='red', marker='+')
plt.show()
# Show spectra from selected pixels
plt.figure(figsize=(10,3))
print(f"Showing {rank} pixels (spectra) indexed by {out[2]}")
plt.plot(out[0][1][0])
plt.show()
```




```
Showing 3 pixels (spectra) indexed by [ 4116 14510 14540]
```



A common issue with MC-ALS is that the algorithm often gets stuck after only a few iterations. Indeed, during an iteration in which A changes slightly but the selected atoms remain the same as in the previous iteration, the estimates A and B are identical to those of the previous iteration. We can tackle this issue by using a more flexible formulation or a convex relaxation, but, in our experience, the empirical performance of these variants is similar or worse. This phenomenon can be observed in the toy experiment above; see below the printed number of iterations before the MC-ALS algorithm got stuck.

```
Number of iterations of MC-ALS: 7
```

12.2 Mixed sparse coding

Reference

[Cohen, 2022] J. E. Cohen, "Dictionary-based low-rank approximation and the mixed sparse coding problem", Frontiers in Applied Mathematics and Statistics, 2022. frontiers arxiv codes

[Cohen, 2020] J. E. Cohen, "Computing the proximal operator of the l1 induced matrix norm", arxiv 2005.06804, 2020

Let us now look at the DLRA problem when the sparsity level is greater than one. As we have seen *earlier*, the DLRA problem is tricky to solve. To design alternating optimization algorithms for DLRA, it is necessary to first study the subproblem of estimating the columnwise k-sparse matrix X. The resulting estimation problem, coined Mixed Sparse Coding (MSC), is the following:

$$\min_{X \in \Omega_X,\, \|X[:,q]\|_0 \leq k \;\forall q \leq r} \|Y - DXB^T\|_F^2$$

where matrix B is assumed to be full column rank. Recall that Ω_X typically stands for $\mathbb{R}^{d \times r}$ or $\mathbb{R}_+^{d \times r}$.

This problem is studied in particular in [Cohen, 2022]. I provide a short and incomplete overview of the results obtained in this work. It also relates to the *sparse NNLS problem* discussed later.

12.2.1 A few observations

The MSC problem is obviously related to the sparse coding problem, which is recovered when $r = n = 1$. MSC shares a few properties with sparse coding:

- If the support of matrix X is fixed/known in advance, it is possible to compute the nonzero values of X exactly. Indeed, MSC can be reformulated in a vectorized way as $\text{vec}(Y) \approx (D \otimes B)\text{vec}(X)$, and therefore a linear system can be built by selecting the columns of $D \otimes B$ which correspond to nonzero elements in $\text{vec}(X)$. Whether or not this linear system is overdetermined depends on the sparsity level k, the sizes m and d, and also the spark of the dictionary D.
- Solutions to MSC are generically unique if the dictionary D has spark strictly greater than $2k$. The proof is, in fact, the same as in sparse coding because B is full column rank.

There are also some important differences between MSC and sparse coding. For instance, when the dictionary is orthogonal, unless the rank is exactly one, hard-thresholding does not solve MSC. Similarly, the case $k = 1$, which is covered in the previous section, is not trivial, while in sparse coding it is solved by computing the maximum correlation between atoms and the data matrix Y.

Since MSC and sparse coding are not the same problem, it is relevant to propose dedicated algorithms for solving MSC that are inspired by sparse coding. Two families of algorithms are typically proposed for sparse coding: greedy algorithms and convex relaxations. We propose algorithms from both families in the following.

12.2.2 Heuristics for MSC

A retrospective look at MC-ALS

A naive heuristic for solving MSC assumes noise is absent. In that case, introducing the right pseudo-inverse of B as matrix C, we known that

$$YC = DX$$

which allows us to compute the optimal matrix X by solving a sparse coding problem, for instance, with OMP [Pati *et al.*, 1993]. In particular, when $k = 1$, the optimum is obtained by a single iteration of matching pursuit. This heuristic, coined Trick-OMP, is in fact exactly the approach used in MC-ALS *described earlier* (not the linear assignment variant). While this is correct in the noiseless case, it is a priori unclear how robust this method can be. The following theorem is one possible characterization of the robustness of Trick-OMP. I show that when the noise level is small compared to the conditioning of both matrices B and D, the Trick-OMP procedure can identify the support of the solution.

Theorem 15

Let X, X' two columnwise k-sparse matrices and $\epsilon > 0,\ \delta > 0$ such that $\|Y - DXB\|_F^2 \leq \epsilon$ and $\|YB(B^TB)^{-1} - DX'\|_F^2 \leq \delta$. Further suppose that spark$(D) > 2k$ and that B has full column rank. Then

$$\|X - X'\|_F \leq \frac{1}{\sigma_{\min}^{(2k)}(D)} \sqrt{\delta + \frac{\epsilon}{\sigma_{\min}^2(B)}}$$

where $\sigma_{\min}(B)$ is the smallest nonzero singular value of matrix B and $\sigma_{\min}^{(2k)}$ is the smallest nonzero singular value of all submatrices of D constructed with $2k$ columns.

Under these hypotheses, supposing that X and X' are exactly columnwise k-sparse, if

$$\min_{j \leq r} \sqrt{\min_{i \in S(X_j)} {X_{ij}}^2 + \min_{i' \in S(X'_j)} {X'_{i'j}}^2} > \frac{1}{\sigma_{\min}^{(2k)}(D)} \sqrt{\delta + \frac{\epsilon}{\sigma_{\min}^2(B)}},$$

then matrices X and X' have the same support, $S(X) = S(X')$.

See [Cohen, 2022] for the proof.

Remark

Matrix X' stands for the estimate produced by the MC-ALS algorithm, while X is the ground truth solution to the mixed sparse coding problem.

This result shows that Trick-OMP is theoretically robust; in practice, it performs poorly at medium and high noise levels. Since we aim to solve MSC within an alternating algorithm, we cannot expect that $Y = DXB^T$ holds even approximately at each iteration. Using Trick-OMP as an MSC solver for DLRA is consequently ill-advised unless an excellent initialization can be provided.

Another greedy approach: Hierarchical OMP

Another algorithm we proposed is Hierarchical Orthogonal Matching Pursuit (HOMP). It is a direct adaptation of OMP for MSC. It is indeed possible to show that MSC can be solved column by column as a sparse coding problem, *e.g.*, using OMP. Both Trick-OMP and HOMP are implemented in the library dlra available online. The implementation of these algorithms is slightly too complex to detail here.

A tightest convex relaxation approach

Popular approaches to solve sparse coding problems are to relax the ℓ_0 function using its convex envelope, the ℓ_1 norm. The optimization problem then becomes convex and easier to solve, but the challenge is to assess whether the solutions to the relaxed problem are the solutions of the original sparse coding problem [Foucart and Rauhut, 2013].

Remark

The ℓ_0 function is not a norm, and not even a pseudo-norm in the strict mathematical sense, because it is not homogeneous. I prefer to call it the ℓ_0 function, or ℓ_0 "norm", to avoid ambiguity.

In the MSC problem, it is possible to show that we are, in fact, solving a problem of the form

$$\min_{X \in \Omega_X} \ell_{0,0}(X) \text{ such that } \|Y - DXB^T\|_F^2 \leq \epsilon$$

for some positive value ϵ. We defined the loss function as

$$\ell_{0,0}(X) = \max_{\|z\|_0=1} \|Xz\|_0 = \max_i \|X[:, i]\|_0.$$

In other words, MSC can be seen as a min-max problem in which the solution must have the smallest possible number k of nonzero elements per column to fit the data, but each column may use up to k nonzero elements.

Note: The $\ell_{p,q}$ matrix norm here is defined as $\ell_{p,q}(X) = \sup_{\|z\|_p=1} \|Xz\|_q$, which is the convention used in the linear algebra community, but not in the machine learning community.

A first strategy to solve MSC is to mimic the sparse coding case and compute the tightest convex relaxation of the $\ell_{0,0}$ function. This can be done, and I showed that this relaxation is exactly the $\ell_{1,1}$ matrix norm defined as

$$\ell_{1,1}(X) = \max_i \|X[:, i]\|_1.$$

Remark

The proof is purely analytic and rather straightforward; it relies on computing twice the convex envelope of the $\ell_{0,0}$ function.

In an unpublished note [Cohen, 2020], I showed how to compute the proximity operator of the $\ell_{1,1}$ matrix norm to propose an adaptation of ISTA to the MSC problem. Dokmanic and co-authors have also proposed an algorithm to compute this proximity operator, which is both faster and simpler, and therefore should be used instead [Béjar *et al.*, 2022]. But an immediate issue with this regularization term is that it only penalizes columns of X whose ℓ_1 norms are saturated to the largest value across all columns. In other words, one cannot regularize columns of X which have vastly different ℓ_1 norms, while this may be the case in the ground truth solution. This can be formalized with the following result, which hints that some (possibly all but one!) columns of the regularized solution may not be sparse.

Theorem 16

Suppose the $\ell_{1,1}$ relaxed problem has a unique solution X^*. Let I the set of indices such that for any i in I, $\|X^*[:,i]\|_1 = \|X^*\|_{1,1}$. Denote S the support of X^*. Then there exists at least one index i in I such that $D[:, S[i]]$ is full column rank and $\|X^*[:,i]\|_0 \leq n$.

Experiments conducted in [Cohen, 2022] moreover showed that this approach performs worse than another simpler approach described next.

A simpler convex relaxation

A simpler strategy consists of penalizing each column of matrix X with the ℓ_1 norm. We then solve a problem of the form

$$\min_{X \in \Omega_X} \|Y - DXB^T\|_F^2 + \sum_{i=1}^{r} \lambda_i \|X[:,i]\|_1$$

This second approach, coined Block-LASSO, is simple because we can compute gradients and apply the soft-thresholding operator (the proximity operator of the ℓ_1 norm) column by column. However, it requires introducing one regularization parameter $\lambda_i > 0$ for each column of the matrix X. This is tedious, in particular since designing homotopy algorithms that can automatically select these values is not straightforward at all.

To tune all the λ_i parameters, we assume that the desired sparsity level k for each column is known. Parameters λ_i can then be tuned dynamically using a simple heuristic to reach k nonzero entries within an iterative algorithm. The resulting algorithm is coined Block-FISTA and is a proximal fast gradient algorithm with heuristically adaptive regularization weights.

We may compare the three proposed methods (Trick-OMP, HOMP, Block-FISTA) as MSC solvers, as is done in [Cohen, 2022]. However, to remain concise, we show below an example of the use of an alternating algorithm with Block-FISTA as the MSC solver, applied to smooth CPD.

12.2.3 DLRA for Smooth CPD

One possible application of DLRA is to impose smoothness constraints in tensor decompositions with splines. The idea of using splines in the CP decomposition is due to Timmerman and Kiers [Timmerman and Kiers, 2002]. They proposed to constrain one of the factors of the CP decomposition of an input tensor $T \in \mathbb{R}_+^{n_1 \times n_2 \times n_3}$, say matrix A, to be decomposed exactly in a basis $U \in \mathbb{R}_+^{n_1 \times p}$ of B-splines, such that $A = UZ$. The columns of matrix U are the individual splines that compose this basis. The user has to choose the splines manually, by placing knots in the 1D plane and choosing the polynomial degrees. There are typically only a few B-splines, such that $p \leq n_1$, and the smoothness constraint allows for dimensionality reduction. Numerically, Timmerman computes the QR decomposition of $U = QR$ and projects the data tensor, such that $Q^T A \times_1 T$ is the new compressed data.

Remark

For low-dimensional linear models of the parameter matrices in most rLRA models, computing the QR decomposition of the basis of the low-dimensional linear space is often a useful numerical trick, which works beyond smoothness and B-splines.

The problem of choosing the B-spline by hand is mild for an experienced user with precise knowledge of the expected outcome of the CP decomposition, but it can plague the practical usage of this method by novice users. Using DCPD, it is possible to automatically select the relevant splines from a large dictionary of polynomials by building an explicit dictionary D that stacks all candidate splines. The sparse regression then picks the best splines numerically. Another advantage of

this method is that different components (columns of matrix A) may use different splines. This comes at the cost of solving a large DLRA problem. This probably makes the method realistically impractical for usage in chemometrics, but the goal of this experiment is rather to showcase one possible application of DLRA. Below, we perform DLRA for a rank-three CP decomposition with smoothness on the second mode, using 251 splines for a tensor of size $18 \times 251 \times 21$ with sparsity level $k = 6$.

As a baseline comparison, we also perform nonnegative CP decomposition without sparsity of smoothness constraints, and then compute either NNLS or sparse NNLS to encode the estimated factor matrix in the spline basis. The results show that DLRA, in general, outperforms these two-step procedures in terms of MSE. We also use data smoothing with QR decomposition, but since there are as many splines as dimensions on the second mode, the smoothing has essentially no effect.

```
import numpy as np
import tensorly as tl
import tlviz.visualisation
import matplotlib.pyplot as plt
import urllib.request
from tensorly_hdr.image_utils import generate_splines_DLRA

rank = 3
subsample = False # False or int

D = generate_splines_DLRA(251, [30,10,30], [4,8,4]) # generates some splines, this is␣
↪a handcrafted function, not intended for general use
if subsample:
    D = D[:,:subsample] # taking only first 60 atoms, to make the dictionary tall/
↪undercomplete
    # Then the choice of the splines matters a lot.

#plt.plot(D)
#plt.show()

# Loading the data from online repo https://gitlab.com/tensors/tensor_data_eem
# and move it to the dataset folder of tensorly_hdr
# urllib.request.urlretrieve('https://gitlab.com/tensors/tensor_data_eem/-/raw/master/
↪EEM18.mat?ref_type=heads', '../tensorly_hdr/dataset/eem.mat')
# The dataset is also distributed with the code directly

# Time for a little dance with MATLAB and Python dataset formats
import scipy.io
import xarray
import tlviz
data = scipy.io.loadmat('../../tensorly_hdr/dataset/eem.mat')
tensor = data['X']['data']
tensor = tl.tensor(tensor[0,0])
tensor = tensor/tl.max(tensor)
# adding noise
tensor_noised = tensor + 0.2*np.random.randn(*tensor.shape)
# Creating a dataset for tlviz
dataset = xarray.DataArray(
    data = tensor_noised,
    coords={
        "Sample index": np.linspace(1,18,18),
        "Emission wavelength": np.linspace(250, 500, 251),
        "Excitation wavelength": np.linspace(210, 310, 21)
    },
    dims = ["Sample index", "Emission wavelength", "Excitation wavelength"]
```

(continues on next page)

(continued from previous page)

```
)
# Plotting raw data
from tensorly.decomposition import non_negative_parafac_hals
raw_cp = non_negative_parafac_hals(tensor, rank, n_iter_max=100, init='svd') #␣
 ↪noiseless 'GT'
mse_raw = tl.norm(tensor - raw_cp.to_tensor())**2/tl.prod(tl.shape(dataset.data))

# Computing nonnegative CP on noisy data
out = non_negative_parafac_hals(dataset.data, rank, n_iter_max=100, init='svd')
out.normalize()
mse = tl.norm(tensor - out.to_tensor())**2/tl.prod(tl.shape(dataset.data))

# Computing (not nonnegative!) CP with preconditioning using the splines (Timmerman␣
 ↪et. al.)
Q, R = np.linalg.qr(D)
precond_data = tl.tenalg.mode_dot(dataset.data, Q@Q.T, mode=1)
out_precond = non_negative_parafac_hals(precond_data, rank, n_iter_max=100, init='svd
 ↪')
mse_precond = tl.norm(tensor - out_precond.to_tensor())**2/tl.prod(tl.shape(dataset.
 ↪data))
# too many splines, need to pick some by hand...
# also same splines for all components!

# Running Sparse Coding as a post processing, with fista
from tensorly.solvers.nnls import fista, hals_nnls
from copy import deepcopy
out_ls = deepcopy(out)
Xls = hals_nnls(D.T@out_ls[1][1], D.T@D, V=Q.T@out_ls[1][1]) # nonnegative least␣
 ↪squares
out_ls[1][1] = D@Xls
mse_ls = tl.norm(tensor - out_ls.to_tensor())**2/tl.prod(tl.shape(dataset.data))

out_sc = deepcopy(out)
lamb = 0.1
Xsc = fista(D.T@out_sc[1][1], D.T@D, sparsity_coef=lamb, tol=0, n_iter_max=1000) #␣
 ↪fista
out_sc[1][1] = D@Xsc
mse_sc = tl.norm(tensor - out_sc.to_tensor())**2/tl.prod(tl.shape(dataset.data))

# Testing the proposed DLRA algorithm
from dlra.algorithms import dlra_parafac
k = 6
perm_data = tl.transpose(dataset.data, [1,0,2])
init_dlra = tl.cp_tensor.CPTensor((None, [np.copy(out_precond[1][i]) for i in [1,0,
 ↪2]])) # initialized with Timmerman approach
X0 = np.copy(Xsc)
out_dlra, _, _ = dlra_parafac(perm_data, rank, [D], [k], nonnegative=True, lamb_rel =␣
 ↪[1e-3], tau=5, X0=[X0], tol=0, n_iter_max = 20, init = init_dlra)
out_dlra = tl.cp_tensor.CPTensor((None, [out_dlra[1][i] for i in [1,0,2]]))
mse_dlra = tl.norm(tensor - out_dlra.to_tensor())**2/tl.prod(tl.shape(dataset.data))
```

```
MSE for:
 raw data 1.1122985684446354e-05,
 noisy data 0.0002501098178444264,
 smoothed data 0.00023286010332691673,
 nnls on D 0.00010307638113572705,
```

(continues on next page)

(continued from previous page)

```
ALS + sparse coding 0.0003177281042646095,
DLRA 0.00013877809428615277
```

12.3 Multiple dictionaries

Reference

[Cohen and Gillis, 2018] J. E. Cohen, N. Gillis, "Spectral Unmixing with Multiple Dictionaries", IEEE Geoscience and Remote Sensing Letters, vol.15 issue 2, pp. 187-191, 2018. pdf

In the *one-sparse DLRA section*, we have seen that in the context of spectral unmixing, we may use the whole hyperspectral image Y as a dictionary. Then computing one-sparse DLRA means identifying pure pixels in the image Y whose spectra correspond to a single material.

While this approach can, in principle, detect pure pixels, the optimisation problem is in fact difficult to solve: the dictionary contains as many atoms as there are pixels, and these atoms are highly correlated. A potential workaround, proposed in [Cohen and Gillis, 2018], is to consider a subset of the image. The user might select regions on a projected view of Y, typically in an RGB visualization, believing they contain pure pixels.

In fact we may go further, and assume that the user selects p regions $D_i := Y_n[:, S_i]$ in the image where S_i collects the indices of the pixel in the ith zone from the columnwise ℓ_2 normalized image Y_n, and that the user guesses how many pure pixels d_i are to be found in each zone, at most. Finding the pure pixels in each zone and the corresponding abundances can then be formalized as follows:

$$\min_{\mathcal{K}_i \subset [1, \#S_i], B \in \mathbb{R}_+^{n \times r}} \|Y - \left[D_1[:, \mathcal{K}_1], \ldots, D_p[:, \mathcal{K}_p]\right] B^T\|_F^2 \text{ s.t. } \#\mathcal{K}_i \leq d_i, \sum_{i=1}^{p} \#\mathcal{K}_i = r$$

where $\mathcal{K}_i$ is the set of selected pure pixels in the user-defined zone S_i, and $\#\mathcal{K}$ stands for the number of elements in the set $\mathcal{K}$, see the illustration below.

```
import tensorly as tl
from tensorly.datasets import data_imports
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from myst_nb import glue

# Import an hyperspectral image
image_bunch = data_imports.load_indian_pines()
image_t = image_bunch.tensor # x by y by wavelength
image_t = image_t/tl.max(image_t) # maximum value set to 1

# Define zones with pure pixels
# Indian Pines is 145 by 145 (pixels) by 200 (bands)
n1, n2, n3 = tl.shape(image_t)
rank = 5
zx = [[100, 120], [50, 60], [100, 110], [10, 130]]
zy = [[110, 130], [80,90], [15, 25], [4, 90]]
d = [1, 1, 1, 2] # we aim to find 1, 1, 1 and 1 pure pixels in each zone
D = [tl.unfold(image_t[zx[i][0]:zx[i][1], zy[i][0]:zy[i][1], :], 2) for i in␣
 ↪range(len(zx))] # unnormalized

# Vectorize the tensor into a n_1\times n_2 matrix (wavelength \times pixels, y vary␣
 ↪first)
image = tl.unfold(image_t, 2)


# Show the hyperspectral image at a given band
fig1, ax1 = plt.subplots()
```

(continues on next page)

(continued from previous page)

```
band = 70  # which spectral band to show
ax1.imshow(image_t[:,:,band], cmap="Greys")
colors = ['b', 'y', 'g', 'm', 'r', 'k']
for i in range(len(zx)):
    rect = patches.Rectangle((zx[i][0], zy[i][0]), zx[i][1]-zx[i][0], zy[i][1] -␣
 ↪zy[i][0],
                             linewidth=3, edgecolor=colors[i], facecolor='none')
    ax1.add_patch(rect)
plt.close()
```

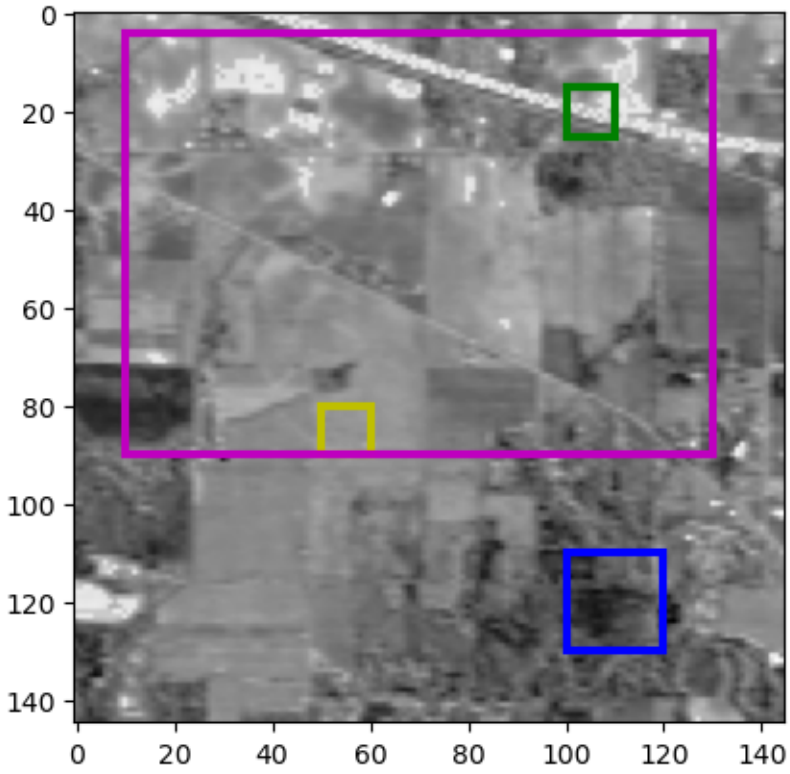


Fig. 12.1: Showing the hyperspectral image at the 70th band. Colored zones mark the position of the hand-picked pure pixel zones. The dictionaries D_i have shapes [(200, 400), (200, 100), (200, 100), (200, 10320)]. We seek [1, 1, 1, 2] atoms in each dictionary.

Solving this multiple-dictionary one-sparse DLRA problem seems daunting. However, it is not much more difficult than solving a one-sparse DLRA if an alternating optimization strategy is used. Solving for matrix B with fixed pure pixel indices is a simple NNLS. When fixing matrix B, estimating pure pixel indices $\mathcal{K}_i$ for each dictionary is combinatorial. Nevertheless, we can adapt the *MC-ALS* strategy to handle this problem:

- First, given an estimate for matrix B, compute $A := \underset{A \geq 0}{\text{argmin}} \|Y - AB^T\|_F^2$ with a NNLS solver.
- Then, fixing A, we find the pure pixel indices by solving the following optimization problem:

$$\underset{\mathcal{K}_i \subset [1, \#S_i] \; \forall i \leq p}{\text{argmin}} \quad \|A - \left[D_1[:, \mathcal{K}_1], \ldots, D_p[:, \mathcal{K}_p]\right]\|_F^2.$$

This quadratic cost can be turned into a linear cost since the dictionaries are normalized columnwise and all quantities

are nonnegative.

$$\underset{\mathcal{K}_i \subset [1, \#S_i] \ \forall i \leq p}{\operatorname{argmax}} \langle A, [D_1[:, \mathcal{K}_1], \ldots, D_p[:, \mathcal{K}_p]] \rangle$$

where $\langle A, D \rangle = \operatorname{Tr}(A^T D)$ is the usual scalar product for matrices. This problem can be seen as a linear sum assignment problem, for which efficient solvers are known [Kuhn, 1955]. First, we may assume that all d_i are equal to one, since we can always duplicate any dictionary D_i to handle $d_i > 1$. When $d_i = 1$ for all $i \leq p$, for each dictionary D_i, only one atom can be selected. It is therefore natural to define a distance between a column j of matrix A and the whole dictionary D_i, here stored in a matrix $C \in \mathbb{R}^{p \times r}$

$$C[i, j] = \max_{l \leq \#S_i} \langle A[:, j], D_i[:, l] \rangle.$$

Then the problem of finding pure pixels with the precomputed matrix A boils down to solving

$$\underset{\Pi \in \mathcal{P}(p,r)}{\operatorname{argmax}} \langle A, C\Pi \rangle$$

where $\mathcal{P}(p, r)$ is the set of surjective permutations from $[1, p]$ to $[1, r]$. This is a classical formulation of the linear sum assignment problem (with p possibly larger than r).

Remark

We used here the fact that minimization over all $\mathcal{K}_i$ can be decomposed in two steps: minimization over $\mathcal{K}_i$ for a single i and each column of A, yielding the ith row of cost matrix C, then minimization over all $i \leq p$ and all columns of A, which is done by the linear sum assignement solver (typically dedicated linear program solvers such as the Hungarian algorithm).

Note: Building on top of the above derivation, we may in fact use any reasonable distance between atoms in dictionaries D_i and columns of A to compute a cost matrix C, such as the Spectral Angular Mapper [Boardman, 1993, Kruse *et al.*, 1993]. The MC-ALS is a heuristic; in particular, the choice of this distance need not be driven by the noise model for the data Y.

We may implement the resulting Multiple dictionaries MC-ALS (M2C-ALS) similarly to MC-ALS. Below is the algorithm for estimating the pure pixels. We will assume nonnegativity is always satisfied by the data and estimated parameters.

```
from scipy.optimize import linear_sum_assignment
import tensorly as tl
import numpy as np

def M2C_ALS_Aestimation(Y, D, B, d, A0=None):
    # D = [D_1, ..., D_p] is a list of normalized dictionaries
    # d = [d_1, ..., d_p] is the list containing the bounds on the number of pure␣
 ↪pixels

    if A0 is None:
        Ahat = tl.solvers.hals_nnls(B.T@(Y.T), B.T@B, n_iter_max=20).T
    else:
        Ahat = tl.solvers.hals_nnls(B.T@(Y.T), B.T@B, n_iter_max=20, V=A0.T).T
    Ahatn = Ahat/tl.norm(Ahat, axis=0)
    n, r = tl.shape(Ahat)

    p = sum(d)
```

(continues on next page)

(continued from previous page)

```
    C = tl.zeros([p, r])
    best_atoms_i = tl.zeros([sum(d), r], dtype=int)
    best_atoms_idx = tl.zeros([sum(d), r], dtype=int)

    # Compute the distances between dictionaries D_i and columns of A, duplicating␣
↪when d_i>1
    # Keep in memory the position of the atoms in the dictionaries that maximize the␣
↪distance
    idx_1 = 0
    for i in range(len(d)):
        idx_0 = idx_1
        idx_1 = idx_1+d[i]
        cross_product = D[i].T@Ahatn
        for j in range(r):
            idx_max = tl.argmax(cross_product[:,j])
            C[idx_0:idx_1, j] = cross_product[idx_max, j]
            best_atoms_i[idx_0:idx_1, j] = i
            best_atoms_idx[idx_0:idx_1, j] = idx_max

    # solve the linear assignement problem
    K, _ = linear_sum_assignment(C, maximize=True)

    # Recover the atoms in each dictionary to fill in A
    Ahat2 = tl.copy(Ahat)
    Kis = []
    for ik, k in enumerate(K[:r]):
        i = best_atoms_i[k, ik]
        idx_max = best_atoms_idx[k, ik]
        Ahat2[:, ik] = D[i][:, idx_max]
        # store the atoms in the order they appear in A
        Kis.append((i, idx_max))
    return Kis, Ahat2, Ahat
```

For a proof of concept of this sub-algorithm, we provide random abundances and select pure pixels in the zones.

```
# normalize dictionaries and pick random pixels
pix = []
atom_list = []
for i in range(len(D)):
    D[i] = D[i]/tl.norm(D[i], axis=0)
    [pix.append(np.random.randint(D[i].shape[1])) for j in range(d[i])]
    [atom_list.append(D[i][:, pix[-j-1]]) for j in range(d[i])]

A0 = tl.tensor(atom_list).T
B = np.random.rand(n1*n2, rank) #tl.solvers.hals_nnls(A0.T@image, A0.T@A0).T
image_toy = A0@B.T
out = M2C_ALS_Aestimation(image_toy, D, B, d)
```

The true selected pixels are [204, 99, 74, 870, 6942] (indices are local for each dictionary D_i). The ones provided by the algorithm, without hints, are [204, 99, 74, 6942, 870]. Now we can test the full alternating algorithm on Indian Pines and see what it gives, when compared to naive separable NMF such as one-sparse DLRA or SNPA.

```
def M2C_ALS(Y, D, B_0, d, itermax=100, verbose=False):
    errs = []
    B = np.copy(B_0)
```

(continues on next page)

(continued from previous page)

```
    A = None
    for it in range(itermax):
        # estimate A
        K, A, _ = M2C_ALS_Aestimation(Y, D, B, d, A0=A)
        if verbose:
            print(f"At iteration {it}, the pure pixel indices in each dictionary are
↪{K}")
        # estimate B
        B = tl.solvers.hals_nnls(A.T@Y, A.T@A, n_iter_max=20).T
        # compute error
        errs.append(tl.norm(Y - A@B.T)**2/tl.prod(Y.shape))
        # stop if error has not changed at all (algorithm is stuck)
        if it>1 and errs[-2]==errs[-1]:
            print("Final estimated indices", K)
            print("MSE", errs[-1])
            break
    return (None, [A, B]), errs, K

B0 = np.eye(*B.shape)
#B0 = np.random.rand(*B.shape)  #TODO better init
ABest, errs, Ke = M2C_ALS(image, D, B0, d)
```

```
Final estimated indices [(0, 119), (1, 13), (2, 64), (3, 2150), (3, 2289)]
MSE 0.0002204785844333469
```

It may be noted that very few iterations are performed in practice; the algorithm gets stuck in a local minimum because of the discrete-continuous nature of the optimization problem. This problem is inherited from the MC-ALS algorithm. Nevertheless, the residuals are rather small, and we have found good candidates for pure pixels within each dictionary D_i. Below, we plot the obtained spectra and abundances, along with their positions within the entire image. We compare with a classical separable NMF approach, SNPA [Gillis and Vavasis, 2014].

```
# Compare with SNPA
from tensorly_hdr import sep_nmf
from tensorly_hdr.image_utils import convert_to_pixel, convert_to_pixel_from_patches,␣
↪convert_to_index
Ksep, Asep, Bsep = sep_nmf.snpa(image, rank)
Kxsep, Kysep = convert_to_pixel(Ksep, image_t.shape[1])
# Permute pairs of estimated matrices for better visualisation; also permute the Ksep
# TODO

# Converting estimated indices from M2C-ALS to pixel indices, and vectorized indices
Kx, Ky = convert_to_pixel_from_patches(Ke, zx, zy)
Kee = convert_to_index(Kx, Ky, image_t.shape[1])

print("Selected Pure Pixels based on separable NMF", Ksep)
print("Selected Pure Pixels based on Multiple-Dictionary NMF", Kee)
```

```
Selected Pure Pixels based on separable NMF [13225, 19579, 17966, 259, 2516]
Selected Pure Pixels based on Multiple-Dictionary NMF [16794, 11798, 3149, 3165,␣
↪3354]
```

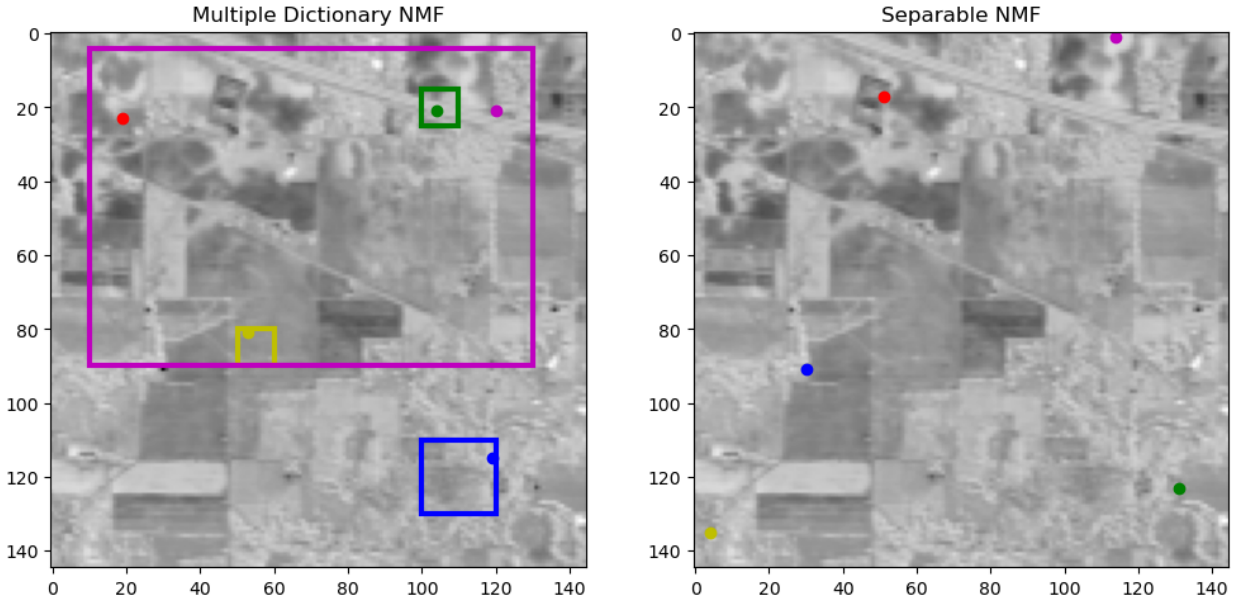


Fig. 12.2: Selected pure pixels from separable NMF (right) and multiple dictionaries (left). Notice that the pure pixels are located in the user-defined areas with the multiple dictionaries model.

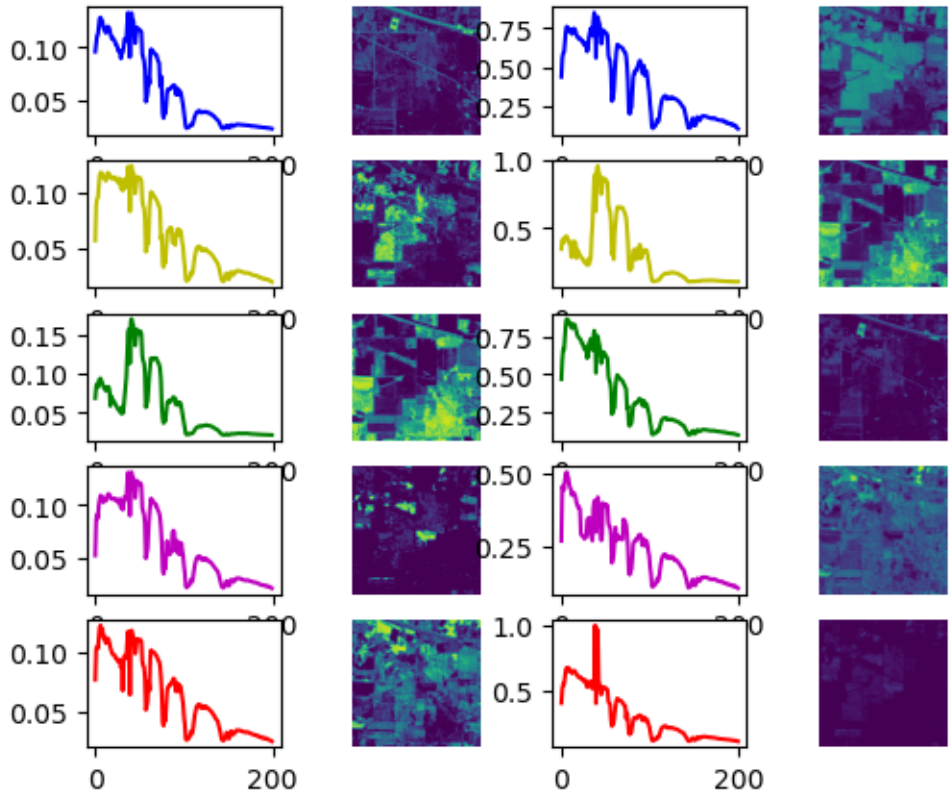


Fig. 12.3: Estimated spectra and abundance maps from separable NMF (right) and multiple dictionaries M2C-ALS (left)

Part V

Fast algorithms for rLRA

CHAPTER

THIRTEEN

FAST ALGORITHMS: SUMMARY

13.1 Optimization challenges in rLRA

Once a rLRA model has been designed for a targeted application, the computation of the rLRA parameters, often called training, parameter estimation, or model fitting, boils down to solving an optimization of the form

$$\underset{\forall q \leq d,\, x_q \in \mathbb{R}^{n_q}}{\operatorname{argmin}} \; f(x_1, x_2, \ldots, x_d) + \sum_q g_q(x_q)$$

with function f a data-fitting term, and g_q mode-wise regularization.

This particular family of problems lies at the intersection of two separate research fields in numerical optimization:

- Multiblock optimization, which is concerned with problems of the form

  $$\underset{\forall q \leq d,\, x_q \in \mathbb{R}^{n_q}}{\operatorname{argmin}} \; f(x_1, x_2, \ldots, x_d).$$

  AO and BCD algorithms are frameworks usually employed to solve these problems, when the cost function f has "nice" blockwise properties (convexity, smoothness, closed-form minimization…). The current state-of-the-art on AO and BCD is summarized in *Alternating optimization*. Algorithms that update all the parameters simultaneously can also be efficient, but are not explored in this manuscript [Acar *et al.*, 2011, Marmin *et al.*, 2023, Takahashi *et al.*, 2025].

- Nonsmooth optimization, encountered when regularizations g_q are nonsmooth. Typical examples in signal processing and machine learning are the ℓ_1 norm, the ℓ_0 function, or nonnegativity constraints.

Nonsmooth optimization has received a lot of attention in the signal processing optimization community over the last twenty years, as sparse approximations have become more prominent and have required better optimization tools. Among the most significant mathematical tools for nonsmooth first-order optimization is the proximity operator, a generalization of projections onto convex sets. The proximity operator is defined for a convex, proper, closed (lower-semicontinous) function g as

$$\operatorname{prox}_{\lambda g}(x) = \underset{u \in \mathbb{R}^n}{\operatorname{argmin}} \; g(u) + \frac{1}{2}\|u - x\|_2^2$$

where λ is a positive scaling parameter. The proximity operator of many classic regularizations is known in closed form or can be efficiently computed, a list is available online. The proximity operator of the characteristic function of a convex set, which penalizes to $+\infty$ any vector outside that set, is exactly the projection on that convex set. The particular case of nonnegativity constraints has been studied extensively and is summarized in *Nonnegative regressions: NNLS and NNKL*.

A third field of research in optimization, sometimes encountered when fitting rLRA models, concerns cost functions that are not Lipschitz-smooth. A typical example is the KL-divergence, which plays a central role in nonnegative LRA. First-order algorithms are hard to derive in this context, see the discussion in .

Because rLRA leads to optimization problems that mix multiblock, nonsmooth, and sometimes non-Lipschitz-smooth optimization, fitting rLRA models efficiently is often challenging.

13.2 Contributions

My contributions in the last ten years on numerical optimization have been geared towards two goals:

- Making optimization algorithms for rLRA faster (for a fixed computational power).
- Making optimization algorithms for rLRA more flexible.

13.2.1 Faster algorithms for rLRA

Faster optimization algorithms are often the focus of numerical optimization research, as they reduce energy and time consumption for end users. In applications where the rLRA model must be fitted in real time, training speed is a key factor that can even drive model choice, at the cost of increased reconstruction error. A typical example is separable NMF, which can be fitted with polynomial-time algorithms but yields higher errors than fully blind NMF.

I have several contributions in this direction. First, in the context of nonnegative LRA, in collaboration with Mai Quyen Pham, we proposed a tight second-order approximation technique coined *median Second-Order Majorant* (mSOM) that is similar to MU in theory, but has provable linear convergence and speeds up convergence in practice [Pham *et al.*, 2025]. This contribution is important in my opinion since it proposes a novel idea for constructing local majorants of loss functions with nonnegative Hessian matrices, that can be developed in several ways we have yet to explore; see the discussion in *Numerical optimization for KL-based regularized inverse problems*. It is also a step towards algorithmic design for computational imaging, a family of applications that I grew interested in after my mutation to CREATIS, see *Single-pixel hyperspectral image reconstruction and unmixing*.

Second, in a collaboration with Christophe Kervazo, we have studied unrolled algorithms for NMF. The key idea in unrolling is to leverage training pairs of inputs and outputs from an optimization algorithm to tune certain parameters in the model, or hyperparameters in the algorithm itself. The gradient of a supervision loss is computed after the optimization algorithm has converged, and any first-order method can then be used to upgrade these parameters. Our contribution is to *unroll the MU algorithm for NMF*. This is challenging because multiplicative updates have no hyperparameters and because of NMF's multiblock nature. While earlier works had considered unrolling the updates for a single block, we unroll the updates for both blocks by introducing trainable masked matrices. Unrolled MU is not only more precise when trained properly, but it is also significantly faster to converge than vanilla MU.

Third, in a collaboration with Andersen Man Shun Ang, we proposed a *heuristic extrapolation strategy for AO*. After each block update, the estimated parameters are extrapolated from the current and previous iterates. The actual implementation includes two important details: restarting and pairing sequences. Restarting cancels any extrapolation if it increases the cost, thereby guaranteeing that the cost decreases at each iteration. Pairing sequences are used in analogy to Nesterov's fast gradient. While this work was novel when released, more recent works have performed similar extrapolation strategies but with convergence guarantees [Hien *et al.*, 2025].

Fourth, in collaboration with Nicolas Nadisic, Arnaud Vandaele, and Nicolas Gillis, we studied branch-and-bound algorithms to compute the exact solution to medium-scale *sparse NNLS problems*. These problems typically arise in source separation, where each mixture at each measurement involves only a small number of spectra. Solving sparse NNLS exactly as fast as possible using combinatorial techniques is useful to design further algorithms such as *sparse separable NMF*.

In my PhD thesis, I was already concerned with fast algorithms for nonnegative tensor decomposition. Because my understanding of this problem has evolved significantly since then, I also included a section analysing one of my first contributions, *projected constrained ALS*.

13.2.2 Flexible algorithms for rLRA

From my experience, end users of rLRA algorithms and software often try various constraints, ranks, data preprocessing, and postprocessing before settling on a setup they feel comfortable with. Therefore, designing algorithms for rLRA that support a wide range of constraints, shapes, and data types is crucial in practice. My contribution to this goal is within the context of multimodality, in which rLRA is applied to multiple matrices or tensors simultaneously. Leveraging earlier work on ADMM for rLRA [Huang *et al.*, 2016], in a series of collaborations with Rasmus Bro, Evrim Acar, Carla Schenker, and Marie Roald, we proposed an ADMM-based algorithm for coupled factorizations, which also allows for flexible coupling designs between the multimodal dataset. The main concepts in coupled factorizations are described in *Constrained coupled matrix and tensor factorization*, while a more specific work on nonnegative PARAFAC2 is detailed in *Nonnegative PARAFAC2*.

CHAPTER

FOURTEEN

PROCO-ALS FOR FAST NCPD

Reference

[Cohen *et al.*, 2015] J. E. Cohen, R. C. Farias, and P. Comon, "Fast decomposition of large nonnegative tensors," IEEE Signal Processing Letters, vol. 22, no. 7, pp. 862-866, 2015. pdf

14.1 Projected least squares for solving NNLS

In the *NNLS* section, we overlooked one possible naive idea for computing an approximate solution: first compute the unconstrained least squares solution, then project it onto the nonnegative orthant. Given a NNLS problem $\underset{x \geq 0}{\operatorname{argmin}} \|b - Ax\|_2^2$, the following two lines of code compute this estimate:

```
    x_hat = tl.solve(A,b)
    x_hat[x_hat<0] = 0
```

While this solution is not optimal (in fact, *there are NNLS instances* for which it is arbitrarily far from the solution), it can be much faster to compute, in particular for large and/or structured A for which highly optimized least squares solvers exist.

A naive algorithm for nonnegative LRA then consists of using this solve-then-project routine as the ALS inner solver. A pitfall to avoid is that sign ambiguity may cause columns of a factor to be all negative at some iteration after the solve operation. Then the projection on the nonnegative orthant yields a zero vector, which makes the next solve ill-posed. A workaround is to flip negative vectors when they occur. The obtained algorithm is coined pro-ALS (projected ALS).

```
def pro_als_basic(T, r, init, itermax = 100, callback=None):
    # Input tensor T, rank of cp decomposition rank r, init is a cp_tensor
    cp_e = deepcopy(init) # estimated cp_tensor
    norm_tensor = tl.norm(T, 2)
    # compute initial error
    err = tl.norm(T - cp_e.to_tensor(), 2)
    callback(cp_e, err)
    for i in range(itermax): # iterative algorithm loop
        for mode in range(T.ndim): # alternating algorithm loop
            # form MTTKRP, pseudo-inverse and solve least squares
            mttkrp = unfolding_dot_khatri_rao(T, cp_e, mode)
            pseudo_inverse = get_pseudo_inverse(cp_e, r, mode, T)
            factor = tl.transpose(tl.solve(tl.transpose(pseudo_inverse), tl.
↪transpose(mttkrp)))
            # Projection
            # if factor is full negative, flip it
```

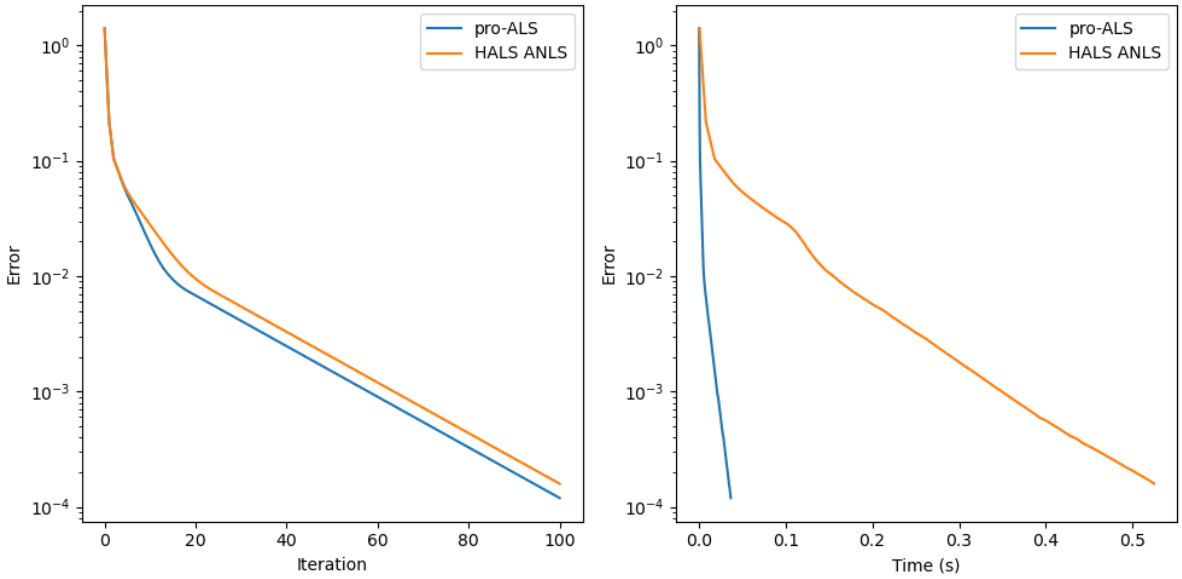
(continues on next page)

(continued from previous page)

```
            support = factor<0
            for q in range(r):
                if support[:,q].all():
                    factor[:,q] = -factor[:,q]
                else:
                    factor[:,q][support[:,q]] = 0
            cp_e[1][mode]=factor

        # compute error
        err = err_calc_simple(cp_e,mttkrp,norm_tensor)
        callback(cp_e, err)

    return cp_e, err
```

We can test this algorithm for computing nonnegative CP on a simple synthetic example.

```
# Hyperparameters
rank=2
dims = [10,10,10]
# Generate a random CP tensor
np.random.seed(0) # for reproducibility
cp_true = tl.random.random_cp(dims, rank) # rank 2, dim 10x10x10
cp_true[0] = None # no weights
init_cp = tl.random.random_cp(dims, rank)
init_cp[0] = None
for i in range(3):
    cp_true[1][i] = tl.maximum(cp_true[1][i],0)
    init_cp[1][i] = tl.maximum(init_cp[1][i],0)
T = cp_true.to_tensor()
# For error and time tracking
time_0 = time.perf_counter()
times_pro = []
errs_pro = []
def err_ret_pro(_,b):
    errs_pro.append(b)
    times_pro.append(time.perf_counter() - time_0)
# Run pro-ALS
cp_e, err = pro_als_basic(T, rank, init_cp, callback=err_ret_pro)

# Comparison with tensorly nn_cp hals
time_0 = time.perf_counter()
errs_hals = []
times_hals = []
def err_ret_hals(_,b):
    errs_hals.append(b) # returns error unnormalized, convert to error
    times_hals.append(time.perf_counter() - time_0)
# Run HALS ANLS
out2 = non_negative_parafac_hals(T, rank, init=deepcopy(init_cp), callback=err_ret_
↪hals)
```



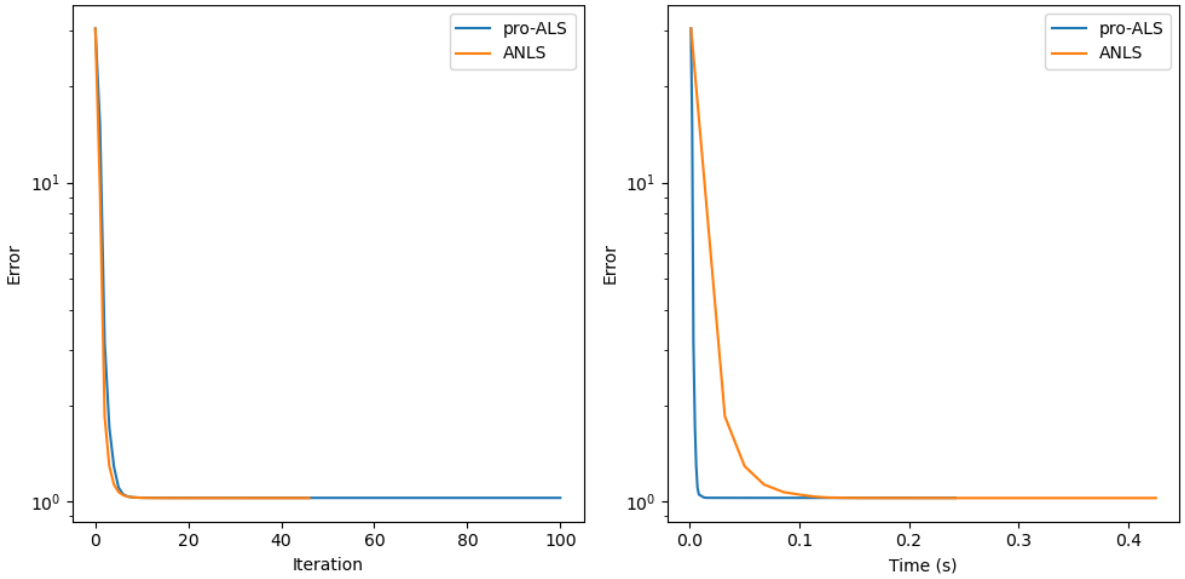
The cost function at each pro-ALS iteration empirically goes down, although this cannot be guaranteed. The algorithm should, in principle, be slower than HALS per iteration, but each iteration is cheap. In practice, for simple problems, it has about the same performance per iteration. For more difficult problems, pro-ALS can, however, struggle to decrease the cost at each iteration. Nevertheless, it performs well on the following realistic example.

```
from tensorly_hdr.load_eem import load_eem
from tensorly.solvers.penalizations import scale_factors_fro

# Load fluorescence chemometrics data
metadata, noisy_data, clean_data = load_eem()
rank = 3

# Random initialization
np.random.seed(12)  # for reproducibility
init_cp = tl.random.random_cp(clean_data.shape, rank)
init_cp[0] = None
for i in range(3):
    init_cp[1][i] = tl.abs(init_cp[1][i])

# For error and time tracking
time_0 = time.perf_counter()
errs_hals = []
times_hals = []
def err_ret_hals(_,b):
    errs_hals.append(b)
    times_hals.append(time.perf_counter() - time_0)
# Run HALS ANLS
out2 = non_negative_parafac_hals(clean_data, rank, init=deepcopy(init_cp),␣
 ↪callback=err_ret_hals)
# Also run pro-ALS
time_0 = time.perf_counter()
errs_pro = []
times_pro = []
def err_ret_pro(_,b):
    errs_pro.append(b)
    times_pro.append(time.perf_counter() - time_0)
```

(continues on next page)

(continued from previous page)

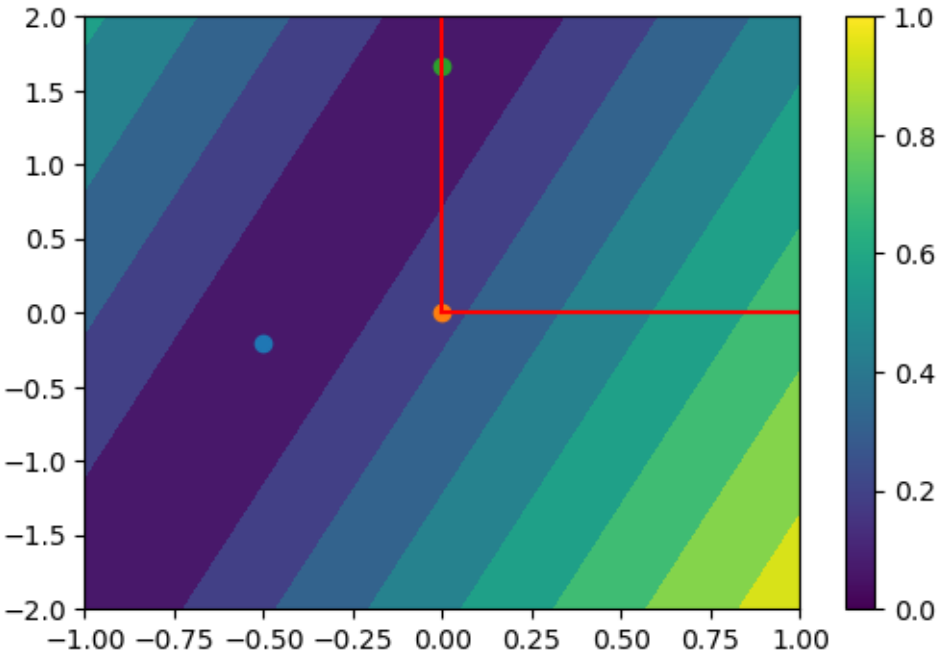
```
# Run pro-ALS
cp_e, err = pro_als_basic(clean_data, rank, init=deepcopy(init_cp), callback=err_ret_
↪pro)
```



14.2 Note on the suboptimality of projected least squares estimates

Using $[A^\dagger b]_+$ as the solution to a NNLS problem $\min_{x\geq 0} \|Ax-b\|_2^2$ can be a terrible idea for some problem instances. We can use NumPy to generate examples in which projected least squares solutions are arbitrarily far from the true NNLS solutions, even in dimensions two. This may happen in particular when the linear system is poorly conditioned. In the plot below, the projected least squares solution is always zero, but the NNLS solution can be made arbitrarily large by stretching and rotating the mixing matrix A as desired.

```
import numpy as np
import matplotlib.pyplot as plt
# Generate data
grid_x = np.meshgrid(np.linspace(-1,1,50),np.linspace(-2,2,50))
grid_z = np.copy(grid_x[0])
theta = 3.14/12
A = np.array([[1,0],[0,0.01]])@np.array([[np.cos(theta), -np.sin(theta)],[np.
↪sin(theta), np.cos(theta)]]) #scale and rotate
x_LS = np.array([-0.5,-0.2])
b = A@x_LS

# LS, Pro-LS, NNLS sols
x_pro_LS = np.maximum(x_LS,0)
x_NNLS = tl.solvers.nnls.active_set_nnls(A.T@b,A.T@A)
```



Color stands for the contour lines of the cost $\|Ax - b\|_2$. Red lines mark the nonnegative orthant. The solution to the LS problem is at the center of the darkest ellipse.

14.3 Proco-ALS

14.3.1 Tucker compression for faster CP-Decomposition

The true interest of pro-ALS lies in its use in conjunction with Tucker compression. The idea of Tucker compression is to first compute an orthogonal, approximate Tucker decomposition with small inner dimensions using, *e.g.*, HOSVD, then work on the resulting smaller core tensor to compute the CP decomposition.

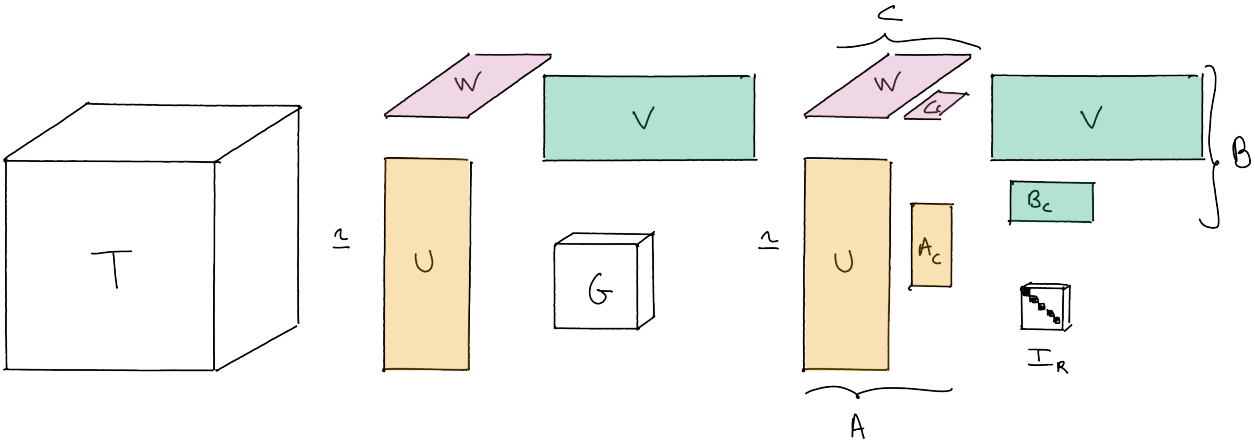


Fig. 14.1: Illustration of Tucker compression used before the computation of the CP decomposition. The core tensor G is significantly smaller than the original tensor T, and therefore the CP decomposition of tensor G is obtained faster than for T. If the Tucker decomposition is accurate and computed efficiently, Tucker compression before CP decomposition can lead to a significant speed-up of the CP decomposition of tensor T.

Mathematically, this makes sense because of the associativity of the multiway tensor product:

$$T = (A \otimes B \otimes C)\, I_r = (UA_c \otimes VB_c \otimes WC_c)\, I_r = (U \otimes V \otimes W)\, (A_c \otimes B_c \otimes C_c)\, I_r$$

where the tensor $(A_c \otimes B_c \otimes C_c)\, I_r \in \mathbb{R}^{r_1 \times r_2 \times r_3}$ can be understood as a core tensor G with a CP decomposition of rank r and small factors A_c, B_c, C_c.

In tensorly, we can perform Tucker compression (also called CANDELINC [Carroll *et al.*, 1980]) easily as follows

```
cp_true = tl.random.random_cp([100,100,100], 3)
T = cp_true.to_tensor()

# Perform Tucker on T
t_tensor = tl.decomposition.tucker(T, [3,3,3])
# Perform CP on G
G = t_tensor[0]
small_cp_est = tl.decomposition.parafac(G, 3)
# Recover larger cp by chain rule
cp_est = tl.cp_tensor.CPTensor((small_cp_est[0], [t_tensor[1][i]@small_cp_est[1][i]␣
↪for i in range(T.ndim)]))

# Comparing with direct cp algorithm
cp_est_direct = tl.decomposition.parafac(T, 3)
```

```
Relative reconstruction error of ALS with Tucker compression 0.0002231681889191562
Relative reconstruction error of ALS without Tucker compression 0.
↪00022316818891919473
```

In this noiseless experiment, Tucker compression is lossless.

14.3.2 Tucker compression for Nonnegative CP

This procedure is standard for unconstrained CP as it offers little inconvenience when high precision is not required or when the data is noiseless. It is particularly useful when several CP decompositions of the same tensor are to be computed.

However, for nonnegative CP, when working on the core tensor G, one has to keep in mind that nonnegativity applies to the original factors, not the small compressed ones. It means that when updating, *e.g.*, factor A_c during the CP decomposition of G, the problem to solve is

$$\min_{UA_c \geq 0} \|G_{[1]} - A_c\, (B_c \odot C_c)^T\, \|_F^2$$

which is not a typical NNLS problem. It is still a quadratic program; we can solve it up to machine precision with a variety of algorithms. However, these algorithms might be costly. Consider a projected gradient descent algorithm

$$A_c^{k+1} = \Pi_{Ux \geq 0} \left[A_c - \eta \left(G_{[1]}\, (B_c \odot C_c) + (B_c^T B_c * C_c^T C_c) \right) \right]$$

The cost of the gradient step is low due to Tucker compression, but the cost of the projection onto the positive cone of U can be consequential if U is a large matrix (which is exactly the setup we consider for Tucker compression). This projection is in fact exactly a collection of n_1 NNLS problems of dimensions r, with n_1 the dimension in the first mode of the original tensor.

In earlier work [Cohen *et al.*, 2015], we proposed to use the pro-ALS idea to avoid resorting to costly NNLS solvers. This gave birth to the proco-ALS algorithm detailed below. We first solve the least squares problem unconstrained, then project onto the constraint set $UA_c \geq 0$. As mentioned above, such a projection is also costly. We use a heuristic approximate projection instead,

$$\hat{\Pi}(x) = U^T\, [Ux]_+ .$$

This projector $\hat{\Pi}$ is one iteration of alternating projection on convex sets (projection on the nonnegative orthant, and on $\mathrm{col}(U)$ since U is a frame). In particular, it is a non-expensive operator. This means that iterating $\hat{\Pi}$ several times would converge towards $\Pi_{Ux\geq 0}$. However, for performance reasons, we recommend running it only once.

```
def proco_als(G, r, init, comp_facs, itermax=100):
    # Input tensor G, rank of cp decomposition rank r, init is a cp_tensor, comp_facs␣
 ↪are U,V,W
    cp_e = deepcopy(init) # estimated cp_tensor
    norm_tensor = tl.norm(G)
    err = []
    for i in range(itermax): # iterative algorithm loop
        for mode in range(G.ndim): # alternating algorithm loop
            # form MTTKRP, pseudo-inverse and solve least squares
            mttkrp = unfolding_dot_khatri_rao(G, cp_e, mode)
            pseudo_inverse = get_pseudo_inverse(cp_e, r, mode, G)
            factor = tl.transpose(tl.solve(tl.transpose(pseudo_inverse), tl.
 ↪transpose(mttkrp)))

            # Projection on UA_c >= 0
            # First decompress
            factor_big = comp_facs[mode]@factor
            # if factor is full negative, flip it, otherwise project
            support = factor_big<0
            for q in range(r):
                if support[:,q].all():
                    factor_big[:,q] = -factor_big[:,q]
                else:
                    factor_big[:,q][support[:,q]] = 0
            # recompress
            factor = comp_facs[mode].T@factor_big
            cp_e[1][mode]=factor

        # compute error in the compressed domain
        err.append(err_calc_simple(cp_e,mttkrp,norm_tensor))

    return cp_e, err
```

Again we can test this proposed proco-ALS algorithm on a simple synthetic dataset.

```
cp_true = tl.random.random_cp([100,100,100], 3)
init_cp = tl.random.random_cp([100,100,100], 3)
for i in range(3):
    cp_true[1][i] = tl.maximum(cp_true[1][i],0)
    init_cp[1][i] = tl.maximum(init_cp[1][i],0)
cp_true[0] = None
init_cp[0] = None
T = cp_true.to_tensor()

# Perform Tucker on T
t_tensor = tl.decomposition.tucker(T, [3,3,3])
# Perform CP on G
G = t_tensor[0]
init_small = tl.cp_tensor.CPTensor((None,[t_tensor[1][i].T@init_cp[1][i] for i in␣
 ↪range(T.ndim)]))
small_cp_est, err = proco_als(G, 3, init_small, t_tensor[1])
# Recover larger cp by chain rule
cp_est = tl.cp_tensor.CPTensor((small_cp_est[0], [t_tensor[1][i]@small_cp_est[1][i]␣
```

(continues on next page)

(continued from previous page)

```
 ↪for i in range(T.ndim)]))

# Comparing with direct cp algorithm
cp_est_direct = tl.decomposition.non_negative_parafac_hals(T, 3)
```

```
Relative reconstruction error of ALS with Tucker compression 4.566109438447164e-06
Relative reconstruction error of ALS without Tucker compression 0.17070784831547361
```

In this example without noise and with perfectly hand-picked multilinear ranks, Tucker compression using proco-ALS helps converging toward the optimal nCP much faster.

Note on the Projection Π

We use the KKT conditions to show that solving

$$\min_{Ux \geq 0} \frac{1}{2}\|x - \hat{x}\|_2^2$$

is equivalent to a NNLS problem

$$\min_{z \geq 0} \frac{1}{2}\|U^T z + \hat{x}\|_2^2.$$

This equivalence also has a geometric interpretation: we either project $-\hat{x}$ on the dual cone of U (second problem) or find the closest element to $\hat{x}$ in the intersection of half planes (first problem).

The Lagrangian for the projection problem writes $L(x, \mu) = \frac{1}{2}\|x - \hat{x}\|_2^2 - \mu^T U x$ which when minimized w.r.t. x yields $x^* = \hat{x} + U^T \mu^*$. The dual problem is therefore formalized as $\min_{\mu \geq 0} -\frac{1}{2}\|U^T \mu\|_2^2 + \|U^T \mu\|_2^2 + \mu^T U \hat{x}$ which has the same minimizer than $\min_{z \geq 0} \frac{1}{2}\|U^T z + \hat{x}\|_2^2$

To summarize, we may compute the projection $\Pi_{U.}(y)$ by first solving several small NNLS problems with mixing matrix U^T. This yields z. Then we compute the projection by using the primal-dual relationship $\Pi_{Ux}(y) = y + U^T z$.

CHAPTER

FIFTEEN

EXACT SPARSE ℓ_0 NONNEGATIVE LEAST SQUARES

Reference

[Nadisic *et al.*, 2020] N. Nadisic, J. E. Cohen, A. Vandaele, N. Gillis, "Exact Sparse Nonnegative Least Squares", ICASSP2020, pdf

Throughout this manuscript, the reader encounters many instances of NNLS. Therefore, efficient solvers for NNLS, in particular with matrix unknowns and warm-start, are instrumental to designing efficient algorithms for nonnegative LRA. Recall a generic formulation of (matrix) NNLS as follows,

$$\min_{V \in \mathbb{R}_+^{r \times n}} \|Y - UV^T\|_F^2$$

with $Y \in \mathbb{R}^{m \times n}$ the input data matrix, and $U \in \mathbb{R}^{m \times r}$ the mixing matrix, which may both in principle have positive and negative entries. The reader may refer to the *dedicated section regarding NNLS* in the first part of this manuscript.

An important variant of NNLS is the sparse NNLS problem (sNNLS), in which the coefficients are constrained to have only a few nonzero entries. In the following, we assume this sparsity is enforced columnwise on matrix V^T, although we have also worked on a sparsity constraint applied on all entries of V^T [Nadisic *et al.*, 2022], not discussed in this manuscript. Columnwise sparsity on the coefficient matrix V^T is frequently encountered in applications of rLRA related to source separation. For instance, the matrix V^T may denote source activations in a mixture; it is often reasonable to assume that not all sources are activated at all times or positions. *An example* is provided below in *spectral unmixing*. The (matrix) sNNLS problem can be formulated as

$$\min_{V \in \mathbb{R}_+^{n \times r}} \|Y - UV^T\|_2^2 \text{ such that } \forall q \leq n,\ \|V[q,:]\|_0 \leq k$$

for a known fixed sparsity level $k < r$. Geometrically, sNNLS is the projection of a data vector y on the facets of the cone spanned by columns of U. Data points can lie on exterior facets of the cone $\text{col}_+\left(\left(\right)U\right)$, or on interior facets.

Note that, unlike matrix NNLS, for which we can design algorithms such as HALS using matrix-matrix products which leverage the native parallelism of these operations, we have not found a better solution for sNNLS than applying a vector sNNLS solver looping over all columns. Therefore, we may simply study the following (vector) sNNLS problem,

$$\min_{v \in \mathbb{R}_+^r} \|y - Uv\|_2^2 \text{ such that } \|v\|_0 \leq k \tag{15.1}$$

with y a column of input matrix Y and v a column of matrix V^T.

15.1 sNNLS is NP-hard, but the rank is typically small

Looking at sNNLS (15.1), the reader may recognize a particular case of the sparse coding problem with nonnegativity constraints. In fact if we could solve sNNLS in polynomial time of r, we could solve sparse coding with the algorithm since a sparse matrix-vector product Ax can be rewritten as a sparse nonnegative matrix-vector product $[A, -A][x_+, x_-]$ splitting the positive and negative parts of sparse vector x. This proves that sNNLS is NP-hard. Computing global solutions to sparse coding is therefore bound to be expensive when the dimension r grows.

Nevertheless, in rLRA problems, dimension r refers to the rank of the rLRA model, which is typically reasonably small. This opens the door to considering algorithms that explore the entire set of solutions. Indeed, the sNNLS problem reduces to finding an optimal support for the solution v^*, since

$$\min_{v\in\mathbb{R}^r_+} \|y - Uv\|_2^2 \;\text{ such that }\; \|v\|_0 \leq k \;=\; \min_{\#S\leq k,\; S\subset[1,r]} \;\min_{v_S\in\mathbb{R}^k_+} \|y - U[:,S]v_S\|_2^2 \tag{15.2}$$

where the inner problem wrt. v_S is a simple NNLS of reduced dimension.

15.2 Branch and Bound strategy

Considering equation (15.2), we could solve the (vector) sNNLS problem by computing $\binom{r}{k}$ NNLS problems of size k. This brute force approach is tractable for small k and r and has the advantage of computing the global solution to sNNLS. Nevertheless, let us consider a more subtle approach that still solves sNNLS globally.

The key observation is that the cost function $\|y - Uv\|_2^2$ at optimality decreases with the sparisty constraint k. Formally, if $S_1 \subset S_2 \subset [1, r]$ are two supports with S_1 strictly included in S_2 (corresponding to a sparser solution v_1^*), then

$$\min_{v\geq 0} \|y - U[:,S_1]v\|_2^2 \geq \min_{v\geq 0} \|y - U[:,S_2]v\|_2^2.$$

In other words, the optimal cost necessarily increases as the constraint set grows. This observation suggests a Branch-and-Bound (BaB) algorithm for searching the combinatorial space of supports. We may first compute a guess for the optimal sNNLS solution using a heuristic algorithm, such as nonnegative OMP [Nguyen *et al.*, 2017, Pati *et al.*, 1993]. The cost function value at this solution v_{UB} is an upper-bound UB of the actual minimal cost of k-sparse sNNLS. Second, we may compute all NNLS problems with one zero element. For illustration purposes, consider the problem where the first element in v is set to zero, and denote the optimal cost for this problem s^*. Because of the previous observation regarding monotonicity of the costs with increasing constraints, cost s^* is a lower bound LB on the optimal costs of all subsequent sNNLS problems where the first entry in the unknown v is zero. This fact applies to any set of constraints (any node in the tree of possible supports, see illustration below). The BaB algorithm allows us to save computation time when we encounter the case $UB < LB$, in which case we know that the current proposed solution v_{UB} will always be better than the solutions at the current node and its children. This allows for safely pruning the set of supports to explore. Importantly, when a new solution for an r-sparse NNLS problem is computed, if its loss is lower than the current UB, we have obtained a new tighter UB, and update v_{UB} and UB accordingly. This procedure is perhaps better understood using a tree visualisation.

We named the proposed BaB algorithm for solving sNNLS `arborescent`.

Note: Despite the BaB algorithm pruning nodes, thus avoiding some computations, the total number of nodes in the graph – and therefore the total number of NNLS problems to solve if no pruning is achieved – is $\sum_{p=k}^{r} \binom{r}{p}$. Therefore in some problem instances, the brute force algorithm can be faster if well implemented.

Note: The proposed arborescent algorithm, in fact, solves all p-sparse NNLS problems for $p \geq k$. Indeed, one can show that the tree obtained after running arborescent can be cut at any floor optimally. This is useful if the sparsity level is not known in advance, in which case arborescent returns all p-sparse solutions.

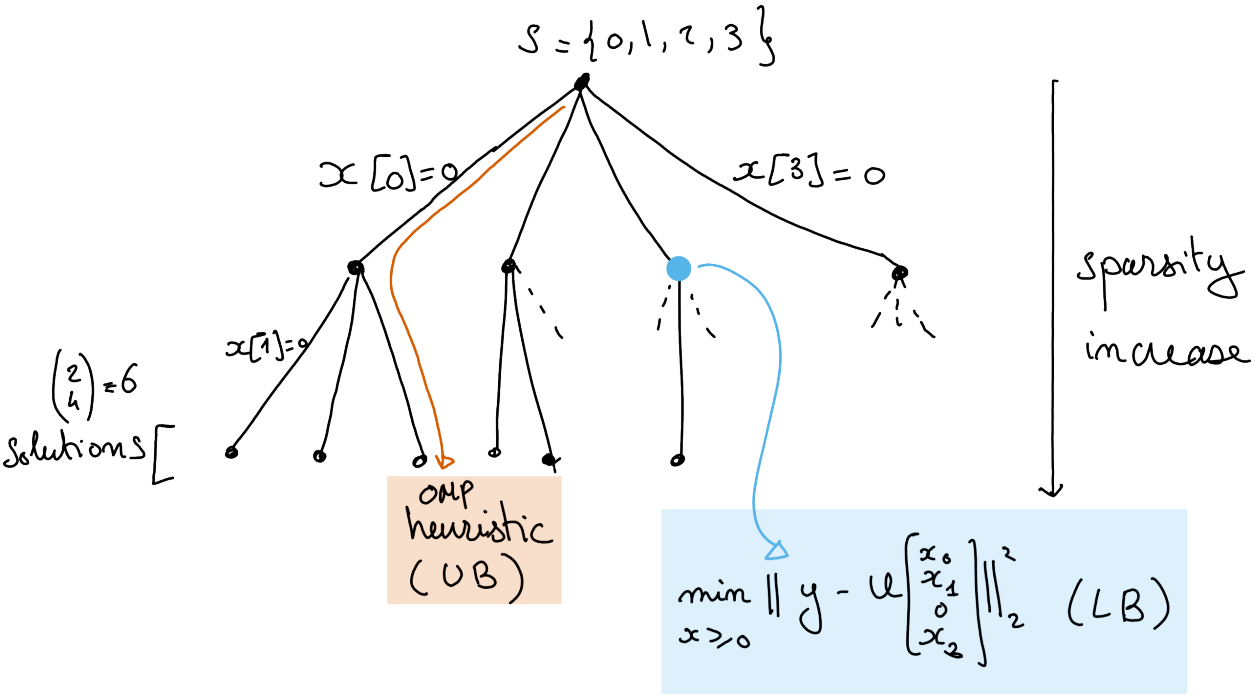


Fig. 15.1: A tree representation of the Branch and Bound algorithm for solving sNNLS with $r = 2$ and $k = 4$. The deeper the node the sparser the solution, and the higher the cost. The leafs are the k choose n solutions. OMP is a heuristic that selects one particular leaf, which loss is larger than the loss at the optimal sNNLS solution, while any node in the tree has a loss lower than all its children.

15.2.1 A few details on the actual computations

Upper-bound. We either use OMP, or compute a full NNLS at the root node, sort the coefficients, prune the smallest ones, and then recompute the resulting r-sparse sNNLS. This is equivalent to a depth-first, left-first exploration of the search tree. The nodes are then explored left to right (with the largest initial nonnegative coefficients first).

NNLS solver. To avoid spending too much time in the NNLS solver at each node, we use a warm start with the AS algorithm run to global convergence. In this setting, warm starts are extremely effective, drastically reducing computation time while keeping an exact solver and avoiding errors from early stopping.

Implementation. The method is implemented in Julia because loops and recursion are efficiently compiled.

15.3 Example of matrix sparse Nonnegative Least Squares

```
import tensorly as tl
from tensorly.datasets import data_imports
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from tensorly_hdr.sep_nmf import snpa
from tensorly_hdr.giantly.arborescent import ksparse_nnls
from tensorly.solvers.nnls import hals_nnls

# Import an hyperspectral image
image_bunch = data_imports.load_indian_pines()
image_t = image_bunch.tensor # x by y by wavelength
```

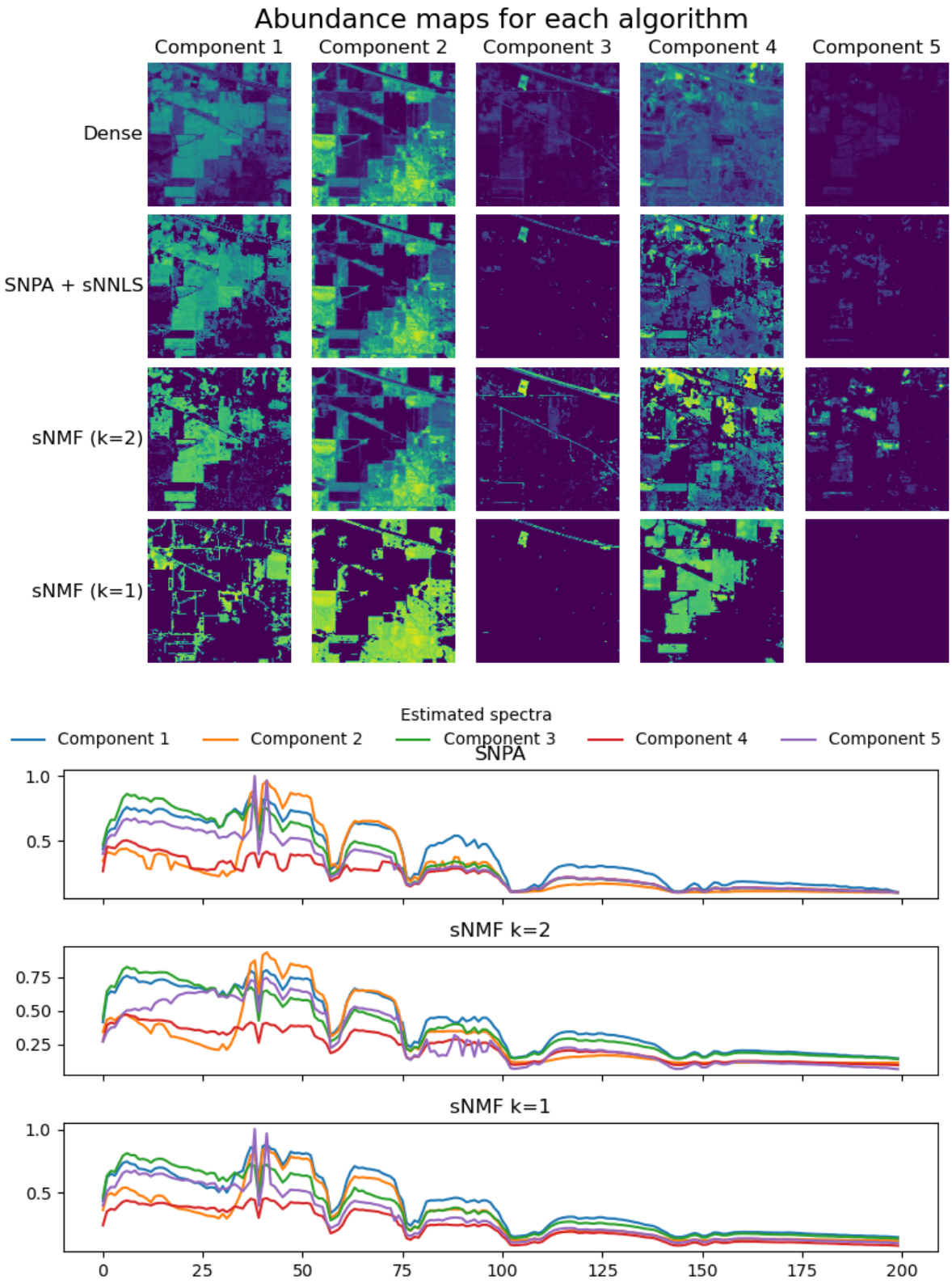
(continues on next page)

(continued from previous page)

```
image_t = image_t/tl.max(image_t) # maximum value set to 1
# Vectorize the tensor into a n_1\times n_2 matrix (wavelength \times pixels, y vary␣
↪first)
image = tl.unfold(image_t, 2)

# Indian Pines is 145 by 145 (pixels) by 200 (bands)
n1, n2, n3 = tl.shape(image_t)
rank = 5

# select pure pixels and compute dense abundances
_, U, VTdense = snpa(image, rank)

# compute k sparse coefficients with sNNLS
VTsparse = tl.copy(VTdense)
imagetimage = tl.norm(image, axis=0)**2
k = 2

VTsparse = ksparse_nnls(U.T@image, U.T@U, imagetimage, VTsparse, k)
```

We can also implement a bare-bones alternating algorithm for NMF with k-sparse coefficients as follows. This algorithm is guaranteed to converge since it is an exact alternating algorithm with unique and attained subproblem solutions (supposing full-rank factors at each iteration), see *Alternating optimization*.

```
def barebone_aNNLS_sNMF(Y, U, VT, rank, k, itermax=10, tol=1e-8):
    # inplace modification of initial U and V
    YtY = tl.norm(image, axis=0)**2
    for i in range(itermax):
        print(f"Running iteration {i}")
        VT = ksparse_nnls(U.T@image, U.T@U, YtY, VT, k) # r x n1n2
        U = hals_nnls(VT@image.T, VT@VT.T, U.T).T
        rmse = tl.norm(Y - U@VT)/tl.sqrt(n1*n2*n3)
        print(f"RMSE is {rmse}")
        if rmse<tol:
            break
    return U, VT

Usnmf, VTsnmf = barebone_aNNLS_sNMF(image, tl.copy(U), tl.copy(VTdense), rank, k)
# Clustering with k=1
Usnmf1, VTsnmf1 = barebone_aNNLS_sNMF(image, tl.copy(U), tl.copy(VTdense), rank, 1)
```


Abundance maps for each algorithm
Component 1
Component 2
Component 3
Component 4
Component 5
Dense
SNPA + sNNLS
sNMF (k=2)
sNMF (k=1)
Estimated spectra
Component 1
Component 2
Component 3
Component 4
Component 5
SNPA
1.0
0.5
sNMF k=2
0.75
0.50
0.25
sNMF k=1
1.0
0.5
0
25
50
75
100
125
150
175
200

CHAPTER

SIXTEEN

MEDIAN SECOND-ORDER MAJORANT FOR FASTER NNLS

Reference

[Pham *et al.*, 2025] M-Q. Pham, J. E. Cohen, T. Chonavel, "A fast Multiplicative Updates algorithm for Nonnegative Matrix Factorization", accepted at TMLR, 2026 arxiv reviews

16.1 Separable quadratic majorization minimization

In *Nonnegative regressions: NNLS and NNKL*, various equivalent formulations of MU have been discussed. There is, however, one more equivalent procedure that leads to MU updates for both Frobenius loss and KL-divergence loss that we leverage in [Pham *et al.*, 2025] to derive more efficient algorithms for NMF (or nonnegative tensor decomposition). This procedure is not limited to matrix and tensor factorizations.

Consider the optimization problem

$$\underset{x\in\mathcal{C}}{\operatorname{argmin}}\ f(x)$$

where $f:\mathbb{R}^n\mapsto\mathbb{R}_+$ is twice-differentiable, and $\mathcal{C}$ is a convex set. We also require that the Hessian matrix $\nabla^2 f(x)$ is nonnegative elementwise.

Remark

We typically work with convex sets for which the projection is easy and cheap to compute.

Since f is twice differentiable, we may consider its truncated Taylor expansion for two vectors x, y in $\mathbb{R}^n$:

$$\phi_x(y) = f(x) + \langle\nabla f(x), y-x\rangle + \frac{1}{2}\langle\nabla^2 f(x)(y-x), y-x\rangle,$$

and $\phi_x(y)\approx f(x)$ when vectors y and x are close.

The traditional approach to second-order iterative algorithms, and in particular the Newton method, is, at iteration index k, to minimize $\phi_{x^{(k)}}(y)$, denote $x^{(k+1)}$ the minimizer, and repeat this procedure until convergence [Bertsekas, 1999]. While the Newton algorithm features super-linear convergence near a stationary point of f, each iteration is computationally expensive. Moreover, Newton's method typically does not account for nonsmooth constraints. Indeed, without constraints,

$$\underset{y\in\mathbb{R}^m}{\operatorname{argmin}}\ \phi_x(y) = x - [\nabla^2 f(x)]^{-1}\nabla f(x),$$

while with constraints, the miniminization of ϕ has no closed form expression. In the particular case where $\mathcal{C}$ is the nonnegative orthant, minimizing the second-order approximation ϕ amounts to solving a NNLS problem.

Therefore, significant efforts have been made towards designing simpler algorithms than Newton's method that still utilise second-order information to some extent. Among others, Gauss-Newton, Levenberg-Marquardt, and LBFGS are popular methods.

Remark

More can be read about Newton and quasi-Newton algorithms in [Bertsekas, 1999].

Maybe the simplest way to change the quadratic estimation of the cost function is to replace the Hessian matrix by a diagonal matrix $A(x)$,

$$\psi_x(y) = f(x) + \langle \nabla f(x), y - x \rangle + \frac{1}{2} \langle A(x)(y - x), y - x \rangle.$$

The quadratic function ψ_x is separable, therefore finding the minimum of ψ_x is cheap even under the presence of constraints:

$$\operatorname*{argmin}_{y \in \mathcal{C}} \psi_x(y) = \operatorname*{argmin}_{y \in \mathcal{C}} \sum_{i=1}^{n} \nabla f(x)[i] y[i] + \frac{1}{2} A[i,i](y[i] - x[i])^2$$

where $\Pi_{\mathcal{C}}$ denotes the projection on the convex set $\mathcal{C}$. For many constraint sets, including nonnegativity and cardinality constraints (more generally, for any separable prior, stable by elementwise nonnegative scaling), the solution is given in closed form as

$$\Pi_{\mathcal{C}} \left(x - \frac{\nabla f(x)}{\operatorname{diag}(A(x))} \right).$$

The core design choice is the construction of the diagonal matrix $A(x)$. Inspired by the proof technique of Lee and Seung for the convergence of MU, we observed with Mai Quyen Pham that for any positive vector $u \in \mathbb{R}^n$ and for elementwise nonnegative Hessian matrices, any matrix of the form

$$A_u(x) = \operatorname{Diag} \left(\frac{\nabla^2 f(x) u}{u} \right)$$

is a majorant of the Hessian, namely $A_u(x) - \nabla^2 f$ is positive semi-definite. The proof is straightforward and can be found in [Pham *et al.*, 2025]. If matrix $A(x)$ is a majorant of the Hessian, we obtain a majorization minimization algorithm as long as f is a quadratic function, because the truncated second-order Taylor expansion is exact. When f is not quadratic, the proposed procedure, which, as far as we know, has not been explored in the optimization literature (at least for computing NMF), is coined Second-Order Majorant (SOM), because we minimize a majorant of a separable second-order approximation of the cost function.

Remark

For a quadratic loss function f, the SOM algorithm is guaranteed to converge since each iteration decreases the cost. However, this form of convergence is very weak: we only know that the values of the cost function will stagnate asymptotically, but the estimated minimizer need not be the true minimizer. This result can be refined with the *SUM framework* [Razaviyayn *et al.*, 2013]. Moreover, the convergence rate of SOM in the general case is unknown. In our work, we prove linear convergence of the iterates for the specific problem of NMF with beta-divergence loss.

16.2 mSOM: choosing an optimal majorant to the quadratic approximation of f

The core idea in the joint work with Mai Quyen Pham and Thierry Chonavel [Pham *et al.*, 2025] is to select a vector u so that $A_u(x)$ is as small as possible. This amounts to choosing the inverse of the gradient stepsizes as small as possible. We discuss several possible metrics to measure the magnitude of $A_u(x)$, but one design choice that leads to a closed-form expression is to minimize the median values in $A_u(x)$, namely

$$u_{mSOM} = \underset{u>0}{\text{argmin}} \, \| \frac{\nabla^2 f(x) u}{u} \|_1.$$

Remark

Another interesting choice of criterion is the min-max over the diagonal elements of the preconditioner. We show in this work that this is equivalent to Gradient Descent with an optimal stepsize.

It can be shown that when $\nabla^2 f(x)$ is nonnegative, the solution is in fact trivial:

$$u_{mSOM} = 1_n,$$

and the resulting algorithm is given by

$$x^{(k+1)} = \underset{y \in \mathcal{C}}{\text{argmin}} \sum_{i=1}^{n} \nabla f(x)[i] y[i] + \frac{(y[i] - x[i])^2}{\sum_j \nabla^2 f(x)[i,j]}.$$

Remark

Nonnegativity of the Hessian of the cost at any point is a strong assumption in general, but is satisfied in problems of interest to this manuscript, *e.g.*, for NMF with beta-divergence loss.

In many particular cases, such as nonnegativity constraints, this update further simplifies to

$$x^{(k+1)} = \Pi_{\mathcal{C}} \left(x^{(k)} - \frac{\nabla f(x)}{\nabla^2 f(x) 1_n} \right).$$

Because this update rule will be used inside an alternating optimization framework, it can be useful not to minimize the majorant but rather to take a larger step by introducing, for $\gamma \in [0, 2]$, the update

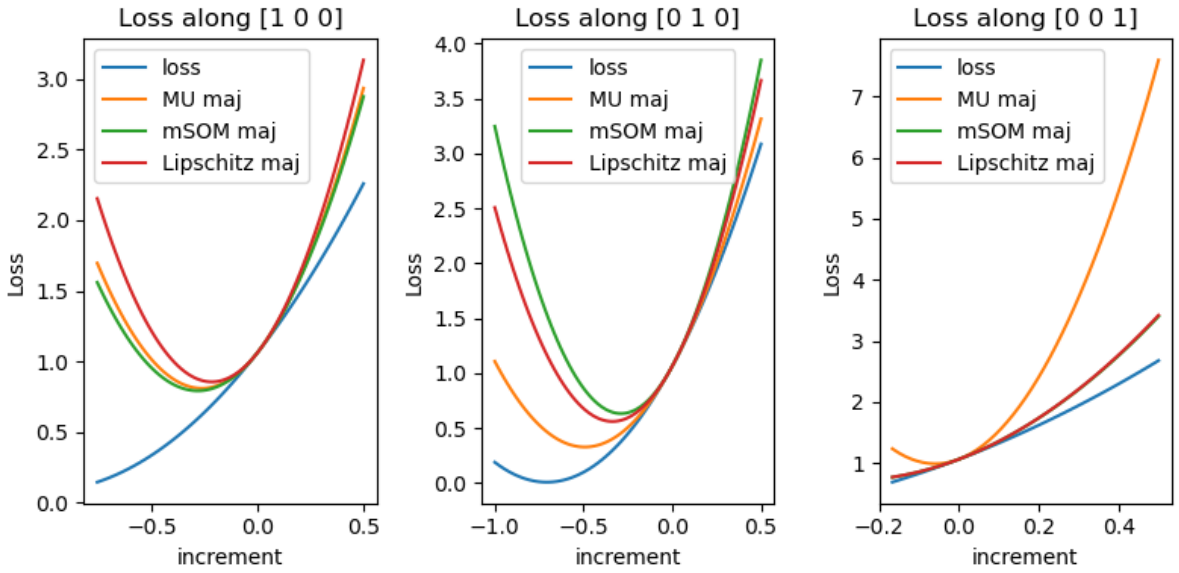
$$x^{(k+1)} = \Pi_{\mathcal{C}} \left(x^{(k)} - \gamma \frac{\nabla f(x)}{\nabla^2 f(x) 1_n} \right). \tag{16.1}$$

We call the algorithm with update rule (16.1) the median SOM algorithm (mSOM). One has to be mindful of a few properties of mSOM:

- For non-quadratic loss functions, it is not guaranteed to decrease the cost function. In fact, mSOM can diverge if initialized poorly. We prove in [Pham *et al.*, 2025] that for $f(x) = \mathcal{D}_{\text{KL}}(y, Wx)$ (or any β-divergence with $\beta \in [1, 2[$) and in the noiseless case, linear convergence happens but is only local. In practice, we observe convergence problems in the first few iterations that can be avoided by using a *properly scaled initialization*, either by using a few iterations of another algorithm as initialization or by checking numerically that the costs decrease.
- For quadratic loss, the mSOM algorithm is a proper MM algorithm, and we show that it converges linearly globally for any γ in $[0, 2]$.

We can visualize the different majorants of the cost (MU majorant, mSOM majorant, and usual gradient descent with optimal stepsize) on a numerical example.

```
import numpy as np
import matplotlib.pyplot as plt

# Data generation
np.random.seed(0)
[n,m] = [3,4]
W = np.random.rand(m, n)
# worse conditionning makes mSOM and Gradient descent more visibly distinct
W[:,2] = 0.1*W[:,2]+0.45*W[:,0] + 0.45*W[:,1]  # col 3 is col 1 + col 2
xmin = np.random.rand(n)
y = W@xmin

def maj(xk,x,W,y, case='MU'):
    """ Majorant function for the Frobenius NNLS loss at xk, obtained with the MU␣
 ↪algorithm.
    returns MU, mSOM and GD majorants
    """
    Wxk = W@xk
    WtW = W.T@W
    Wty = W.T@y
    if case=='MU':
        Precond = WtW@xk/xk
    elif case=='mSOM':
        Precond = WtW@np.ones(n)
    elif case=='GD':
        Precond = np.linalg.norm(WtW, 2)*np.ones(n)
    return np.sum((Wxk - y)**2) + (x - xk).T@(2*WtW@xk - 2*Wty) + np.sum(Precond * (x␣
 ↪- xk)**2),


def loss_compute(x0, v, t, W, y):
    """ Compute the Frobenius NNLS loss at x0 + t*v, restricted to the nonnegative␣
 ↪orthant.
    """
    steps = [v*t[i] for i in range(len(t))]
    xs = [x0 + steps[i] for i in range(len(t))]
    return [ [np.linalg.norm(y - W@xs[i])**2 for i in range(len(t)) if np.min(xs[i])>
 ↪=0],
             [maj(x0, xs[i], W, y, case="MU") for i in range(len(t)) if np.min(xs[i])>
 ↪=0],
             [maj(x0, xs[i], W, y, case="mSOM") for i in range(len(t)) if np.
 ↪min(xs[i])>=0],
             [maj(x0, xs[i], W, y, case="GD") for i in range(len(t)) if np.min(xs[i])>
 ↪=0],
             ], [t[i] for i in range(len(t)) if np.min(xs[i])>=0]

# We plot the Frobenius NNLS loss on three 1d slices of the 3d cost
v1 = np.array([1, 0, 0])
v2 = np.array([0, 1, 0])
v3 = np.array([0, 0, 1])

x0 = xmin + np.array([0.2, 0.5, 0.1])  # we plot the loss centered on this point

t = np.linspace(-1, 0.5, 100)
[L1, Lmu1, LmSOM1, LGD1], ts1 = loss_compute(x0, v1, t, W, y)
[L2, Lmu2, LmSOM2, LGD2], ts2 = loss_compute(x0, v2, t, W, y)
[L3, Lmu3, LmSOM3, LGD3], ts3 = loss_compute(x0, v3, t, W, y)
```



Notice that, in this toy example, while mSOM is designed to be sharper than MU in median along the line $t[0, 1, 0]$, MU is tighter.

16.2.1 MU algorithm as quadratic majorant minimization

The SOM algorithm reduces the design of the diagonal matrix $A_u(x)$ to choosing a positive vector u. Any choice of u ensures that $A_u(x)$ is a majorant of the Hessian and therefore that the obtained algorithm is principled. It turns out that for a NNLS problem, the MU algorithm is a particular case of SOM when $u = x$. Indeed, for $f(x) = \frac{1}{2}\|y - Wx\|_2^2$ and nonnegativity constraints, the Hessian matrix writes $\nabla^2 f(x) = W^T W$, and the majorant matrix $A_u(x)$ with $u = x$ is

$$A_x(x) = \text{Diag}\left(\frac{W^T W x}{x}\right).$$

Recall from *Nonnegative regressions: NNLS and NNKL* that this is exactly the diagonal preconditioner that MU utilizes, when understanding MU as a preconditioned gradient descent algorithm. Therefore, setting $u = x$ yields the MU update with a projection operator

$$x^{(k+1)} = \Pi_{\mathcal{C}}\left(x^{(k)} \frac{W^T y}{W^T W x}\right).$$

16.3 Alternating mSOM for NMF

Equipped with the mSOM solver for nonnegative convex problems, one may factorize NMF using an alternating optimization strategy, with mSOM as the inner solver. The following code implements the resulting Alternating mSOM (AmSOM) for NMF with the Frobenius loss and compares it with Alternating MU (AMU) and Alternating Projected Gradient Descent (APGD) on a toy synthetic dataset. The AmSOM updates rules for this problem formalized as $\underset{W,H\geq\epsilon}{\text{argmin}} \|Y - WH^T\|_F^2$ are

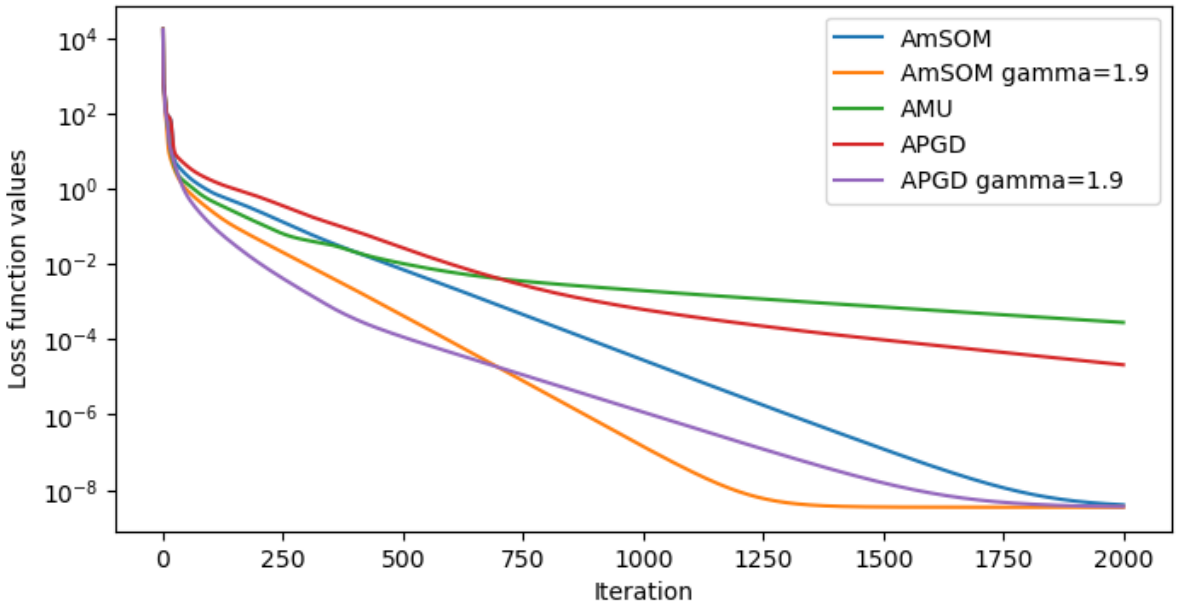
$$H \leftarrow \max\left(H - \gamma \frac{1}{1_{n\times r} W^T W}\left(HW^T W - Y^T W\right), \epsilon\right),$$
$$W \leftarrow \max\left(W - \gamma \frac{1}{1_{m\times r} H^T H}\left(WH^T H - YH\right), \epsilon\right).$$

```python
import numpy as np
import matplotlib.pyplot as plt


def mSOM_update(HtH, YH, W, precond, gamma=1.9, epsilon=0):
    ''' Computes the mSOM update rules for the NNLS problem min_{W\geq 0} ||Y - WH^
↪T||_F^2
    '''
    W = np.maximum(W - gamma * precond * (W @ HtH - YH), epsilon)
    return W

def MU_update(HtH, YH, W, epsilon=0):
    ''' Computes the mSOM update rules for the NNLS problem min_{W\geq 0} ||Y - WH^
↪T||_F^2
    '''
    W = np.maximum(W * (YH / (W @ HtH)), epsilon)
    return W

def AmSOM(Y, Winit, Hinit, method="mSOM", niter=100, gamma=1.9, epsilon=0):
    ''' Alternating mSOM for the NNLS problem min_{W,H\geq 0} ||Y - WH^T||_F^2
    Using 10 inner iterations for the mSOM updates of W and H.
    '''
    W = np.copy(Winit)
    H = np.copy(Hinit)
    loss = [1/2*np.linalg.norm(Y - W @ H.T, 'fro')**2]
    for it in range(niter):
        HtH = H.T @ H
        if method=="GD":
            step = 1/np.linalg.norm(HtH,2)
        YH = Y @ H
        precond_W = 1/np.sum(HtH, axis=1)  # makes use of broadcasting
        for _i in range(10):
            if method == "MU":
                W = MU_update(HtH, YH, W, epsilon=epsilon)
            elif method=="mSOM":
                W = mSOM_update(HtH, YH, W, precond_W, gamma=gamma, epsilon=epsilon)
            elif method=="GD":
                W = np.maximum(W - gamma*step*(W@HtH - YH), epsilon)

        WtW = W.T @ W
        if method=="GD":
            step = 1/np.linalg.norm(WtW,2)
        YW = Y.T @ W
        precond_H = 1/np.sum(WtW, axis=1)  # makes use of broadcasting
        for _i in range(10):
            if method == "MU":
                H = MU_update(WtW, YW, H, epsilon=epsilon)
            elif method=="mSOM":
                H = mSOM_update(WtW, YW, H, precond_H, gamma=gamma, epsilon=epsilon)
            elif method=="GD":
                H = np.maximum(H - gamma*step*(H@WtW - YW), epsilon)

        loss.append(1/2*np.linalg.norm(Y - W @ H.T, 'fro')**2)
    return W, H, loss

# Exemple usage on a toy dataset
np.random.seed(27)
```

(continues on next page)

(continued from previous page)

```
n, m, r = 100, 80, 5
niter = 2000
Wtrue = np.random.rand(n, r)
Htrue = np.abs(np.random.randn(m, r))
Y = Wtrue @ Htrue.T + 1e-6 * np.random.randn(n, m)

Winit = np.abs(np.random.randn(n, r))
Hinit = np.abs(np.random.randn(m, r))

W, H, err_msom = AmSOM(Y, Winit, Hinit, method="mSOM", niter=niter, gamma=1,␣
↪epsilon=0)
Wg, Hg, err_msomg = AmSOM(Y, Winit, Hinit, method="mSOM", niter=niter, gamma=1.9,␣
↪epsilon=0)
W_mu, H_mu, err_mu = AmSOM(Y, Winit, Hinit, method="MU", niter=niter, gamma=1,␣
↪epsilon=0)
W_gd, H_gd, err_gd = AmSOM(Y, Winit, Hinit, method="GD", niter=niter, gamma=1,␣
↪epsilon=0)
W_gdg, H_gdg, err_gdg = AmSOM(Y, Winit, Hinit, method="GD", niter=niter, gamma=1.9,␣
↪epsilon=0)
```



16.4 Other loss functions?

The mSOM framework also applies to NMF with beta-divergences as loss functions. Convergence guarantees are only local, and the empirical results are less significant. Improving mSOM for non-quadratic losses and sparse datasets is a future research direction.

CHAPTER

SEVENTEEN

EXTRAPOLATED BCD ALGORITHMS FOR UNCONSTRAINED MATRIX AND TENSOR DECOMPOSITIONS

Reference

[Ang *et al.*, 2019] A. Ang, J. E. Cohen, L. K. Hien, N. Gillis, "Extrapolated Alternating Algorithms for Approximate Canonical Polyadic Decomposition", ICASSP2020, pdf.

[Ang *et al.*, 2020] A. M. S. Ang, J. E. Cohen, N. Gillis, L. T. K. Hien, "Accelerating Block Coordinate Descent for Nonnegative Tensor Factorization", Numerical Linear Algebra Appl., 2021;e2373. pdf

Among various techniques to speed up first-order optimization algorithms for convex smooth problems, extrapolated (also called inertial) algorithms are especially popular. The theory of extrapolated algorithms such as Nesterov projected gradient descent (also called FISTA or inertial Forward Backward) is now rather well understood: while the convergence rate of gradient descent for smooth strongly convex cost functions is at worse $\mathcal{O}(\frac{1}{k})$, Nesterov gradient descent converges with rate $\mathcal{O}(\frac{1}{k^2})$. This improvement comes at almost no additional cost in terms of memory or computation. In essence, extrapolated first-order algorithms minimize a cost function $f(x)$, where x must belong to a convex set $\mathcal{C}$, with iterates of the form

$$x_{k+1} = \Pi_{\mathcal{C}} \left[y_k - \eta \nabla_f(y_k) \right]$$
$$y_{k+1} = x_{k+1} + \beta_k (x_{k+1} - x_k)$$

where y_k and x_k are the auxiliary and principal sequences of iterations at iteration k, η is the stepsize, $\Pi_{\mathcal{C}}$ is the projection on convex set $\mathcal{C}$ and α_k is an extrapolation parameter dependent on the iteration index. Various formulations have been proposed for the choice of α_k. The convergence of such a scheme was proved first by Nesterov [Nesterov, 1983], but later analyzed more generally as discretized second-order differential equations [Attouch and Peypouquet, 2016]. Interestingly, the cost function may increase with Nesterov gradient descent. O'Donoghue and Candès have proposed a restart strategy to further speed up convergence [O'Donoghue and Candes, 2015]. Restart consists in resetting the value of β_k to ensure that the cost decreases.

17.1 Extrapolated block-coordinate algorithms for LRA

Extending inertial extrapolation strategies for LRA is not straightforward. Indeed, extrapolated algorithms were initially proposed for convex problems, but LRA optimization problems are, in general, multiblock and nonconvex. A simple approach would be to use extrapolated algorithms to solve each block in an alternating algorithm. In [Ang *et al.*, 2019] and [Ang *et al.*, 2020], we proposed an empirical strategy to perform extrapolation for BCD algorithms after the BCD updates, at the outer loop level. In other words, we extrapolate the estimated block of variables after the update, using the previous block estimate. This is fundamentally different from extrapolation within each block. This work has spurred a series of more principled works, in which I was not involved, led by Lee Ti Hien Khan and Nicolas Gillis, that propose extrapolated BCD algorithms with convergence guarantees and good empirical performance [Hien *et al.*, 2023], [Hien

et al., 2025]. Other works in the literature have studied extrapolated block coordinate descent; see, for instance, the iPALM algorithm [Pock and Sabach, 2016]. The main difference between iPALM and subsequent works by Khan, in particular TITAN, is that iPALM alternates between extrapolated gradient steps, whereas TITAN studies a broader class of alternating inertial MM algorithms, including iPALM.

Another work, published almost simultaneously and closely related to ours, proposes an extrapolated ALS algorithm for CPD with restart [Mitchell *et al.*, 2020].

17.2 Heuristic Extrapolation with Restart

The Heuristic Extrapolation with Restart (HER) framework takes the idea of Nesterov projected gradient but applies it to a full BCD cycle. For simplicity, let us apply the HER framework to a specific model: nonnegative CPD,

$$\underset{X_i \geq 0}{\operatorname{argmin}} \|\mathcal{T} - \mathcal{I}_r \times_1 X_1 \times_2 X_2 \times_3 X_3\|_F^2.$$

A vanilla BCD algorithm for nCPD is ANLS, using a generic NNLS solver, which writes as follows for the k-th iteration:

$$\begin{aligned}
X_1^{k+1} &= \underset{X_1 \geq 0}{\operatorname{argmin}} \|T_{[1]} - X_1 \left(X_2^k \odot X_3^k\right)\|_F^2 \\
X_2^{k+1} &= \underset{X_2 \geq 0}{\operatorname{argmin}} \|T_{[2]} - X_2 \left(X_1^{k+1} \odot X_3^k\right)\|_F^2 \\
X_3^{k+1} &= \underset{X_3 \geq 0}{\operatorname{argmin}} \|T_{[3]} - X_3 \left(X_1^{k+1} \odot X_2^{k+1}\right)\|_F^2.
\end{aligned}$$

In contrast, HER-ANLS extrapolates the estimates X^{k+1} based on the NNLS estimation and the previous value, both taken in a pairing sequence Y^k, as follows:

$$\begin{aligned}
Y_1^{k+1} &= \underset{Y_1 \geq 0}{\operatorname{argmin}} \|T_{[1]} - Y_1 \left(X_2^k \odot X_3^k\right)\|_F^2 \\
X_1^{k+1} &= \max\left(Y_1^{k+1} + \beta_k \left(Y_1^{k+1} - Y_1^k\right), 0\right) \\
Y_2^{k+1} &= \underset{Y_2 \geq 0}{\operatorname{argmin}} \|T_{[2]} - Y_2 \left(X_1^{k+1} \odot X_3^k\right)\|_F^2 \\
X_2^{k+1} &= \max\left(Y_2^{k+1} + \beta_k \left(Y_2^{k+1} - Y_2^k\right), 0\right) \\
Y_3^{k+1} &= \underset{Y_3 \geq 0}{\operatorname{argmin}} \|T_{[3]} - Y_3 \left(X_1^{k+1} \odot X_2^{k+1}\right)\|_F^2 \\
X_3^{k+1} &= \max\left(Y_3^{k+1} + \beta_k \left(Y_3^{k+1} - Y_3^k\right), 0\right).
\end{aligned}$$

To ensure that the cost function does not increase significantly, HER also performs a restart operation and updates the extrapolation hyperparameters in consequence. The following code implements HER-ANLS, simplified from the presentation in [Ang *et al.*, 2020], which used an additional pairing sequence to ensure that the cost function decreases.

```
# HER-ANLS
def HER_ANLS(data, rank, cp_e, beta=0.1, gamma=1.05, eta=2.0, itermax=100):
    """
    HER-ANLS: Heuristic Extrapolated Alternating Nonnegative Least Squares for␣
 ↪Nonnegative Canonical Polyadic Decomposition (nCPD).
    """
    # Initialize factor matrices and auxilliary factor matrices
    cp_y = deepcopy(cp_e)

    # Initialize hyperparameters
    norm_tensor = tl.norm(data, 2)
    err = [tl.norm(data - cp_e.to_tensor())]  # Initial error.
```

(continues on next page)

(continued from previous page)

```
    # Outer iteration loop
    for k in range(itermax):
        # loop on the modes of the tensor
        for mode in range(data.ndim):
            # Computing MTTKRP and pseudo-inverse
            pseudo_inverse = get_pseudo_inverse(cp_e, rank, mode, data)
            mttkrp = unfolding_dot_khatri_rao(data, cp_e, mode)

            factor_old = np.copy(cp_y[1][mode])
            # Call the hals resolution with nnls, optimizing the current mode,␣
↪inplace update
            hals_nnls(
                tl.transpose(mttkrp),
                pseudo_inverse,
                tl.transpose(cp_y[1][mode]),
                n_iter_max=100,
            )

            # Extrapolation step, store temporarily to check for restart
            cp_e[1][mode] = tl.maximum(
                cp_y[1][mode] + beta * (cp_y[1][mode] - factor_old), 0
            )

        # Check for restart
        current_err = err_calc_simple(cp_e, mttkrp, norm_tensor)
        if k>0 and current_err > err[-1]:
            ## Restart the extrapolation, cost increased
            cp_e = deepcopy(cp_y)
            beta = beta/eta  # decrease beta
            err.append(tl.norm(data - cp_e.to_tensor())/norm_tensor)
        else:
            ## More aggressive extrapolation
            beta = tl.minimum(beta*gamma, 1)  # increase beta
            err.append(current_err)

    return cp_e, err
```

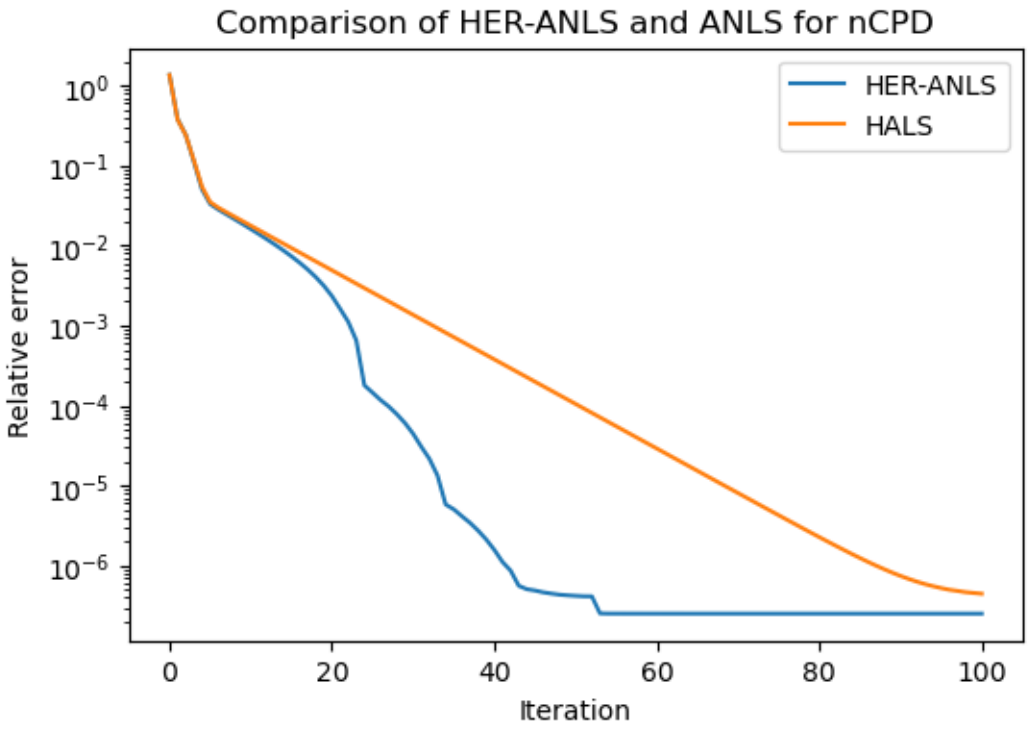
Let us test the HER-ANLS algorithm on a simple nCPD problem with small dimensions and almost no noise.

```
# Hyperparameters
rank = 2
dims = [6,6,6]
noise = 1e-7
# Generate a random nCPD problem
np.random.seed(42)  # For reproducibility
cp_true = tl.random.random_cp(dims, rank) # rank 2, dim 10x10x10
cp_true[0] = None # no weights
init_cp = tl.random.random_cp(dims, rank)
init_cp[0] = None
for i in range(3):
    cp_true[1][i] = tl.maximum(cp_true[1][i],0)
    init_cp[1][i] = tl.maximum(init_cp[1][i],0)
T = cp_true.to_tensor() + noise*tl.random.random_tensor(shape=dims)  # Add some noise
# Computing the nCPD with HER-ANLS
cp_e, err = HER_ANLS(T, rank, deepcopy(init_cp), itermax=100, beta=0.01, gamma=1.2,␣
```

(continues on next page)

(continued from previous page)

```
 ↪eta=5)
# Comparison with tensorly nn_cp hals
err_hals = []
def err_ret(_,a):
    # non_negative_parafac_hals returns a tuple (cp, err, loss) with a 1/2␣
 ↪coefficient for the Frobenius norm
    err_hals.append(a)
out2 = non_negative_parafac_hals(T, rank, init=deepcopy(init_cp), callback=err_ret, n_
 ↪iter_max=100, tol=0)
```



Note that the main hyperparameters for HER-ANLS are

- γ: the amount by which the extrapolation parameter is multiplied at each iteration.
- η: the amount by which the extrapolation parameter is divided when restart occurs.

CHAPTER

EIGHTEEN

CONSTRAINED COUPLED MATRIX AND TENSOR FACTORIZATION

Reference

[Cabral Farias *et al.*, 2016] R. Cabral Farias, J. E. Cohen, and P. Comon, "Exploring multimodal data fusion through joint decompositions with flexible couplings," IEEE Transactions on Signal Processing, vol. 64, pp. 4830-4844, September 2016. pdf

[Cohen and Bro, 2018] J. E. Cohen, R. Bro, "Nonnegative PARAFAC2, a flexible coupling approach", LVA/ICA 2018, arXiv:1802.05035

[Schenker *et al.*, 2021] C. Schenker, J. E. Cohen, E. Acar, "An optimization framework for regularized linearly coupled matrix-tensor factorization", EUSIPCO2020, 2021 pdf

[Schenker *et al.*, 2021] C. Schenker, J. E. Cohen, E. Acar, ``A Flexible Optimization Framework for Regularized Matrix-Tensor Factorizations with Linear Couplings", IEEE Journal on Selected Topics in Signal Processing, 2020. arxiv IEEE Xplore Supplementary Materials Matlab code

[Roald *et al.*, 2021] M. Roald, C. Schenker, J. E. Cohen, E. Acar, "PARAFAC2 AO-ADMM: Constraints in all modes", EUSIPCO2021, arxiv

[Roald *et al.*, 2022] M. Roald, C. Schenker, V. D. Calhoun, T. Adali, R. Bro, J. E. Cohen, E. Acar, "An AO-ADMM approach to constraining PARAFAC2 on all modes", SIAM Journal of Mathematics on Data Science, 2022. arxiv code SIMODS

[Chatzis *et al.*, 2025] C. Chatzis, C. Schenker, J. E. Cohen, E. Acar, "DCMF: Learning Interpretable Evolving Patterns from Temporal Multiway Data" EUSIPCO 2025. arxiv

18.1 Background on joint factorization models

Joint factorization models are collections of matrix or tensor factorization problems in which some of the variables are explicitly related. There are at least two motivations for considering this family of problems.

- **Multimodal acquisitions:** Several datasets are acquired informing on the same phenomenon but with different modalities, *e.g.*, EEG+FMRI, NMR+LCMS [Acar and Yener, 2009], occulometry+EEG [Rivet and Cohen, 2016]. Joint factorization techniques allow the extraction of latent information from each dataset while leveraging the shared information across datasets to reduce estimation error and enhance uniqueness [Sorensen and De Lathauwer, 2015], thereby improving interpretability.
- **Alternative formulations of tensor decompositions:** Joint factorization models are useful for rewriting the classical tensor models *such as CP decomposition* and proposing new extended models such as PARAFAC2 [Harshman, 1972] [Kiers *et al.*, 1999], Shift/Conv NMF/CP, PARATUCK2 [Usevich, 2025] and so on. These formulations are also useful for *identifiability proofs*, as they relate tensor decompositions to matrix LRA.

Let us describe a few important joint factorization models, namely *CPD*, *CMTF*, and *PARAFAC2*.

18.1.1 Joint diagonalisation

CP decomposition is a particular case of coupled matrix factorization. Indeed, if a tensor $T \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ follows a rank r CP decomposition with factors A, B, C, then for each slice index $k \leq n_3$

$$T[:, :, k] = A\text{Diag}(C[:, k])B^T.$$

All the slices $T[:, :, k]$ are therefore jointly diagonalized with left and right bases A and B. There is therefore a strong connection between CP decomposition and joint diagonalisation methods such as GSVD, SOBI [Belouchrani *et al.*, 1997], JBSS [Lahat and Jutten, 2018].

18.1.2 CMTF (direct coupling)

One of the most well-known examples of joint decompositions is coined Coupled Matrix and Tensor Factorization (CMTF) [Acar *et al.*, 2013]. CMTF couples a matrix and a tensor on a single mode, where the matrix Y is low-rank, and the tensor T has low CP-rank. CMTF may be formulated as an optimization problem

$$\underset{A,B,C,D}{\text{argmin}} \ \|T - I_r \times_1 A \times_2 B \times_3 C\|_F^2 + \|Y - AD^T\|_F^2.$$

The first factor of the low-rank approximations of the tensor and the matrix is the same. This means that the same underlying patterns are sought in both datasets along the first mode. In practice, the two terms in the cost function should be balanced based on the SNR of each dataset [Cohen *et al.*, 2016].

CMTF assumes that the parameter matrix A is shared across measurement modalities. Therefore, CMTF lacks flexibility in many applications, such as genomics and chemometrics, where some components cannot be observed by certain modalities due to physical constraints. It is then necessary to modify the CMTF model to share only a subset of components. This variant of CMTF is referred to as ACMTF [Acar *et al.*, 2017]. Another extension of CMTF assumes that the two modalities share a parameter matrix through a known linear relationship. This allows modeling possible variations in the sampling rates by linear interpolation [Cabral Farias *et al.*, 2016], or coupling only the derivatives of the components [Rivet *et al.*, 2015]. This work is summarized *below*.

18.1.3 PARAFAC2 and variants

Starting from the joint diagonalization formulation of the CPD, one may observe that CPD imposes the coupled slices/matrices $T[:, :, k]$ to have the same row and column factors. In some applications, it could be interesting to relax this assumption and suppose that only a single mode, say the first one, has a shared factor. Then the coupled factorization model becomes

$$T[:, :, k] = A\text{Diag}(C[k, :])B_k^T$$

where each slice $T[:, :, k]$ has a different factor matrix $B_k \in \mathbb{R}^{n_{2,k} \times r}$. The issue with this relaxed formulation is that it is equivalent to an unconstrained matrix factorization of the stacked slices,

$$\begin{aligned}[T[:, :, 1], \ldots, T[:, :, n_3]] &= A\left[\text{Diag}(C[1, :])B_1^T, \ldots, \text{Diag}(C[n_3, :])B_{n_3}^T\right], \\ T_{[1]} &= A\tilde{B}^T.\end{aligned}$$

The factor matrix $\tilde{B}$ is virtually unconstrained, as any matrix can be written as a stacked matrix of products. This observation means that to obtain a multiway decomposition that does not reduce to unconstrained low-rank matrix approximation, further constraints must be applied on the parameter matrices B_k. There are several ways to do so, for instance, imposing some shift-invariance with respect to the slice index (Shift-PARAFAC) [Harshman *et al.*, 2003], or imposing further low-rank structure on the factors (PARATUCK2) [Usevich, 2025]. A popular extension of CPD based on this construction is the PARAFAC2 model, obtained by imposing that the cross product of the B_k matrices is constant,

$$\forall k \leq n_3, \ B_k^T B_k = \Delta^T \Delta$$

with Δ of size r by r.

There are several reasons for assuming that the correlation matrices of the second-mode factors are constant across slices. A simple explanation may be obtained by observing that this constraint is equivalent to the reparameterization

$$B_k = P_k \Delta$$

with P_k a left-orthogonal matrix. Therefore, all the slices have the same rank r factor matrix, transformed from slice to slice by a rotation matrix. Orthogonal linear coupling between slices models many linear transformations, such as circular shifts or diffeomorphisms [Cohen *et al.*, 2018]. The PARAFAC2 model has therefore been used extensively in chemometrics applications where such transformations occur, in particular LCMS and GCMS data [Cohen and Bro, 2018]. The computation of PARAFAC2 with nonnegativity constraints is a topic I have worked on, detailed in *Nonnegative PARAFAC2*.

18.2 General formulation: Linearly-Coupled Constrained CMTF

My earliest contribution on joint factorization models was to consider a generalization of CMTF that allows for more complicated coupled relationships [Cabral Farias *et al.*, 2016]. If CMTF supposes that the same factor matrix can be extracted from several datasets, Linearly-Coupled CMTF (LC-CMTF) assumes that the shared components are linked through a linear relationship. For the particular case of a joint matrix and tensor decomposition, LC-CMTF can be formalized as the optimization problem

$$\underset{A_1,B,C,A_2,D,A}{\operatorname{argmin}} \|T - I_r \times_1 A_1 \times_2 B \times_3 C\|_F^2 + \|M - A_2 D^T\|_F^2.$$
$$\text{s.t. } \vec{A}_1 = H_1 \vec{A} \text{ and } \vec{A}_2 = H_2 \vec{A}$$

for known linear coupling matrices H_1 and H_2 and a shared, unknown latent factor matrix A which size may differ from A_1 and A_2.

This model is flexible enough to express several realistic scenarios, such as partially shared components and unaligned data. Two particular linear couplings often encoutered are $[A_2; A_2] = [H_1; H_2]A$ (allows for resampling on a common grid) and $[A_1, A_2] = A[H_1, H_2]$. The latent variable A is introduced to enable the model to easily extend to more than two coupled tensors. In general, a necessary condition for identifiability of $\vec{A}$ is that the matrix $[H_1; H_2]$ is invertible.

In collaboration with Carla Schenker and Evrim Acar [Schenker *et al.*, 2021], we formalized the linear couplings more clearly and proposed an algorithm based on *AO*, where each subproblem is solved by the ADMM algorithm [Huang *et al.*, 2016]. ADMM is chosen for its flexibility, as we can also impose constraints on the parameter matrices, including the coupled ones. It is also possible to use other loss functions than the Frobenius norm. A MATLAB implementation is available here. Sadly, there are no available Python implementations, so I cannot show examples of the AO-ADMM algorithm in action in this manuscript. A python and julia implementation is under way in collaboration with David Hong and Evrim Acar.

18.2.1 Other related works

There are a couple of related published works of mine that are worth mentioning here.

Temporal-aware CMTF

An extension of the *regularized PARAFAC2 work* includes modeling temporal dynamics. A simple way to model the evolution over time of a multivariate vector is through a linear dynamical system, which essentially acts as a linear coupling across slices of a tensor of data. In collaboration with Christos Chatzis, we have studied the results of the time-varying CMTF model and proposed an algorithm to estimate its parameters when the linear dynamical system is known.

EEG artifact removal using ocular measurements

In a collaboration with Bertrand Rivet and Rodrigo Cabral-Farias, we used eye-movement signals recorded with an oculometer to remove artifacts in EEG signals [Cohen *et al.*, 2018]. The signals were acquired by Emmanuelle Kristensen. The goal was to detect eye movements and remove EEG peaks synchronized with them. To identify eye saccades, the data need to be aligned in the temporal domain. The problem may be formulated as a coupled matrix factorization problem, where the sacades are identified by low-rank factorization, and the coupled sacades are related by a time warping.

Instead of using PARAFAC2, which implicitly models time warping, we explicitly model time warping via a diffeomorphism. Our contribution was to propose an algorithm to estimate the parameters of the diffeomorphism along with the low-rank parameter matrices. The low-rank parameter matrices are estimated as in CMTF, while the diffeomorphism parameters are one-dimensional and estimated by binary search. This work was a proof of concept, but the algorithm is too unstable to use in practice, and our earlier work (to which I contributed modestly) using non-parametric coupling with dynamic time warping proved more convenient in the long run [Rivet and Cohen, 2016].

18.3 Nonnegative PARAFAC2

Reference

[Cohen and Bro, 2018] J. E. Cohen, R. Bro, "Nonnegative PARAFAC2, a flexible coupling approach", LVA/ICA 2018, arXiv:1802.05035

[Roald *et al.*, 2021] M. Roald, C. Schenker, J. E. Cohen, E. Acar, "PARAFAC2 AO-ADMM: Constraints in all modes", EUSIPCO2021, arxiv

[Roald *et al.*, 2022] M. Roald, C. Schenker, V. D. Calhoun, T. Adali, R. Bro, J. E. Cohen, E. Acar, "An AO-ADMM approach to constraining PARAFAC2 on all modes", SIAM Journal of Mathematics on Data Science, 2022. arxiv code SIMODS

18.3.1 Why Nonnegative PARAFAC2

The PARAFAC2 model is often used in applications such as Liquid/Gas Chromatography Mass Spectroscopy (LCMS and GCMS, respectively) to extract mass spectra and time elution profiles for various experimental setups. The mixture of interest is injected into a long tube where the various chemical compounds glide at different speeds, and therefore reach the end of the tube at different times. By measuring the mass spectrum at the output of the tube with a mass spectrometer, one may acquire spectra at various time instants. One acquisition, therefore, yields a data matrix with intensities collected at various coordinates (mass-to-charge (m/z) x time). Repeating this acquisition several times for different chemical compounds yields a tensor dataset, with the third dimension indexing the experiments. An example GCMS dataset is available below, with different wine samples from various producers measured individually. The full dataset is described in [Ballabio *et al.*, 2008].

One interesting feature of this dataset, visible in the top-right plot, is that the time elution profiles shift slightly with the experiment index.

Formally, given a collection of mass spectra over time $X[:,:,k]$ where $k \leq K$ stands for the experiment index, supposing elution profiles and mass spectra can be extracted from each experiment using a low-rank approximation, and further supposing that these components are shared across all experiments, estimating these components amounts to computing

$$X[:,:,k] = AD_kB^T,$$

that is by computing a CPD from the data tensor X. The factors bear physical meaning (mass spectra, elution time profiles, and experiment contribution); it is reasonable to impose nonnegativity on all factors.

However, in practice, as observed in this dataset, the injection of chemical compounds into the chromatography system is done manually, which impacts the elution time. Elution time may also fluctuate due to other experimental conditions, such as temperature. Therefore, the time profiles of components B are slightly modified across experiments indexed by k. As discussed in *PARAFAC2 and variants*, further constraints are required to link the resulting B_k matrices, and PARAFAC2 ensure in particular that

- The columns of factors B_k are transformed similarly (the same transformation matrix P_k applies to each column $B[:,q]$).
- The cross product of the B_k factors is constant.

A technical difficulty in applying PARAFAC2 to GCMS data is designing an optimization algorithm that allows for nonnegativity constraints on the B_k matrices. Indeed, the classical algorithm to fit PARAFAC2 proposed by Kiers and Bro [Kiers *et al.*, 1999] relies on the reparameterization $B_k = P_kB$. It can be summarized as follows:

1. Compute the optimal P_k matrices by solving K orthogonal Procrustes problems.
2. Compute the CP decomposition of the tensor obtained by stacking $X[:,:,k]P_k$.
3. Loop the first two steps until convergence

The matrices B_k do not appear explicitly in this algorithm, and therefore, we need to rethink the optimization procedure to include these constraints.

18.3.2 Formalisation of the Nonnegative PARAFAC2 problem

In our first work with Rasmus Bro, we proposed to use a flexible constraint formulation for the Nonnegative PARAFAC2 [Cohen and Bro, 2018]. While this work had a good impact on the chemometrics community, it relies on a relaxation of the PARAFAC2 constraint. It is, however, possible to use the AO-ADMM framework discussed in *Constrained coupled matrix and tensor factorization* to design an alternating iterative algorithm that solves the nonnegative PARAFAC2 problem. The core idea is to introduce a "PARAFAC2 constraint" on the matrices B_k that imposes that their cross products are constant. A collection of matrices $\{B_k\}_{k\leq K}$ satisfies the PARAFAC2 constraint if for all $k \leq K$, $B_k^TB_k$ is constant. In that case we write that $\{B_k\}_{k\leq K} \in \mathcal{C}_{\text{P2}}$.

Nonnegative PARAFAC2 then boils down to the optimization problem

$$\underset{A\geq 0,\ B_k\geq 0,\ C\geq 0}{\text{argmin}} \|X[:,:,k] - A\text{Diag}(C[k,:])B_k^T\|_F^2 \text{ s.t. } \{B_k\}_{k\leq K} \in \mathcal{C}_{\text{P2}}.$$

This problem can be handled by AO-ADMM with splitting, provided we can project onto the set of sets of matrices satisfying the PARAFAC2 constraint. We explore this projection next.

18.3.3 Projection on the PARAFAC2 constraint

A full understanding of the projection under the PARAFAC2 constraint has yet to be reached, and it remains an active research topic. It connects with Riemmanian geometry and optimization, but has not been studied in this literature as such, to the best of my knowledge.

In the work conducted with Marie Roald and co-authors, we used an empirical approach to this projection, relying on a few mathematical observations. First, notice that if a set $\{B_k\}_{k\leq K}$ satisfies the PARAFAC2 constraint, it can be represented as points on the orbit

$$\{PB \mid P^TP = I\} = \mathcal{O}(B).$$

Projection on the PARAFAC2 constraint therefore means finding a matrix B such that the total distance between each point B_k and the orbit $\mathcal{O}(B)$ is as small as possible:

$$\operatorname*{argmin}_B \sum_{k\leq K} d(B, B_k).$$

The distance d we propose to use is the Euclidean distance to the closest point P_kB on $\mathcal{O}(B)$ to B_k, that is

$$d(B, B_k) = \operatorname*{argmin}_{P_k \text{ orthogonal}} \|B_k - P_kB\|_F^2.$$

Projection on the PARAFAC2 constraint, therefore, boils down to solving

$$\operatorname*{argmin}_{B, P_k \text{ orthogonal}} \sum_{k\leq K} \|B_k - P_kB\|_F^2. \tag{18.1}$$

From the point of view of algorithm design, we seek a procedure to compute exact solutions to the problem (18.1) and plug them into the AO-ADMM algorithm. Maybe surprisingly, the following simple alternating algorithm typically converges in a few iterations:

1. Compute orthogonal matrices P_k solving K orthogonal Procrustes problems.
2. Compute the geodesic mean B as $\operatorname*{argmin}_B \sum_k \|P_k^TB_k - B\|_F^2 = \frac{1}{K}\sum_k P_k^TB_k$.

We can observe the procedure's fast convergence empirically in the experiment below.

```
# Hyperparameters
dims = [10,10,5]  # Dimensions of the tensor X
rank = 3  # Rank of the factorization

# Generate a collection of (nonnegative) matrices Bk randomly
Bks = [np.random.rand(dims[1],rank) for _ in range(dims[2])]

def project_parafac2(Bks, itermax=10):
    B = sum(Bks) / len(Bks)  # Initial guess as the mean of the input matrices
    cost = [sum([np.linalg.norm(Bks[i] - B)**2 for i in range(len(Bks))])]  # Initial␣
↪cost
    Pks = [np.eye(0) for _ in range(len(Bks))]  # Initialize Pk matrices

    for _ in range(itermax):
        # Solve for Pk rotation matrices
        for i in range(len(Bks)):
            # Project each matrix Bk onto the PARAFAC2 constraints
            U,s,V = np.linalg.svd(Bks[i]@B.T)
            Pks[i] = U @ V
```

(continues on next page)

(continued from previous page)

```
        # Update B matrix as the mean
        B = sum([Pks[i].T @ Bks[i] for i in range(len(Bks))]) / len(Bks)

        # Recompute the cost
        cost.append(sum([np.linalg.norm(Bks[i] - Pks[i] @ B)**2 for i in␣
↪range(len(Bks))]))

        print(f"Iteration {_+1}, Cost: {cost[-1]:.8f}")

    return B, Pks, cost

out = project_parafac2(Bks, itermax=5)
```

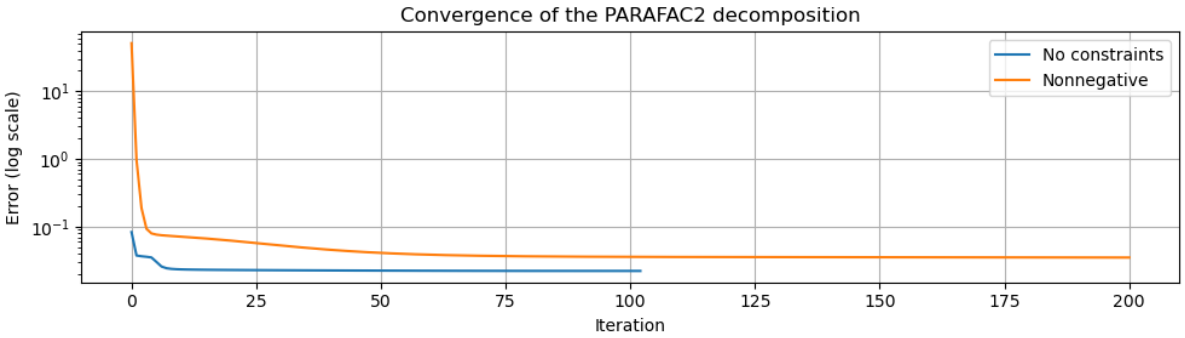
```
Iteration 1, Cost: 2.06434972
Iteration 2, Cost: 2.04302005
Iteration 3, Cost: 2.04300580
Iteration 4, Cost: 2.04300579
Iteration 5, Cost: 2.04300578
```

The convergence speed of this alternating optimization algorithm is suspiciously fast, and we suspect that there may exist a similar procedure that converges in a single iteration, or at least a theoretical analysis guaranteeing the fast convergence of this heuristic projection. What is known for now is that this alternating algorithm converges to the solution of the problem, as a two-block alternating optimization with exact and unique solutions to each subproblem, see *Alternating optimization*.

18.3.4 Nonnegative PARAFAC2 for GCMS data

We may now compute PARAFAC2 with and without nonnegativity constraints on the time elution mode. The AO-ADMM implementation of PARAFAC2 with constraints is found in the matcouply package[Roald, 2023], developed by Marie Roald at Simula (Norway) during her PhD thesis. For PARAFAC2 without nonnegativity on the second mode (but with nonnegativity on the first mode to avoid sign ambiguities), we use the implementation available in tensorly, which uses reparameterization and relies on the CP decomposition.

```
import matcouply as mcl
from matcouply.decomposition import parafac2_aoadmm
from tensorly.decomposition import parafac2

# With nonnegativity constraint
cmf_nn, diagnostics_nn = parafac2_aoadmm(
    tensor, 3, n_iter_max=200, non_negative=True, verbose=False, return_errors=True,␣
↪tol=1e-9, random_state=5
)
weightsnn, (Ann, B_isnn, Cnn) = cmf_nn

# tensorly unconstrained PARAFAC2
pf2, rec_err = parafac2(
    tensor, 3, n_iter_max=200, return_errors=True, nn_modes=[0], random_state=0,␣
↪tol=1e-9, verbose=False
)
weights, (A, B, C), P_is = pf2
B_is = [P_i @ B for P_i in P_is]
```



One may observe that the nonnegative PARAFAC2 model can recover elution peaks that overlap significantly with the baseline. Without nonnegativity constraints, the extracted elution profiles are somewhat mixed, and the baseline is found in all three components.

In a collaboration with Marie Roald, Carla Schenker, and Evrim Acar (SimulaMet, Oslo), we have extended the proposed algorithms for nonnegative PARAFAC2 to other constraints [Roald *et al.*, 2022]. The algorithm works similarly by projecting onto the PARAFAC2 constraint and applying the proximity operator of the constraints to the B_k matrices, after splitting.

CHAPTER

NINETEEN

UNROLLED MU FOR DATA-DRIVEN NMF

Reference

[Kervazo *et al.*, 2024] C. Kervazo, A. Chetoui and J. E. Cohen, "Deep unrolling of the multiplicative updates algorithm for blind source separation, with application to spectral unmixing", EUSIPCO 2024. hal

[Kervazo and Cohen, 2025] C. Kervazo, J. E. Cohen, "Unrolled Multiplicative Updates for Nonnegative Matrix Factorization applied to Hyperspectral Unmixing, submitted pdf

19.1 Data-driven NMF principles

19.1.1 Data-driven NMF through regularization

Low-rank approximation models such as NMF are designed to perform unsupervised learning: given a data matrix Y, their goal is to compute two possibly constrained matrices W, H such that $Y \approx WH^T$. In practical applications, as discussed in *Applications of rLRA: summary*, however, performing LRA is not the final goal. The estimated factor matrices are instead further processed for a downstream task. These post-processing operations include clustering of the components, thresholding activations, or performing linear regression. Often, additional side information is available on the expected outcome of this post-processing. In the language of machine learning, we could say that this additional information takes the form of training data stored in a matrix M.

A simple way to make use of this additional data M is to modify the LRA cost to incorporate this data. This procedure has been named "Supervised LRA" in the literature, see [Lock and Li, 2018] and references therein. For NMF, linear regression from a matrix W to additional data M can, for instance, drive the supervision task. The "supervised" NMF problem formulation is then given by

$$\underset{W\geq 0,\, H\geq 0,\, \theta}{\operatorname{argmin}} \; \mathcal{D}\left(Y, WH^T\right) + \lambda \|W - M\theta\|_F^2$$

for a regularization hyperparameters $\lambda > 0$, a data fitting term $\mathcal{D}$ such as the squared Frobenius norm $\|Y - WH^T\|_F^2$, and regression parameters θ trained jointly with the NMF factors.

One could argue that this approach does not really leverage training data to train a model in the way usually understood in supervised learning. Regression parameters θ are learnt from scratch for each pair of data matrices (Y, M). Matrix M must also be known at inference time.

A modification of this supervised LRA framework can be considered, in which the parameters θ are shared across p LRA problems. In the above example of supervised NMF with linear regression, given a collection of data matrices $\{(Y_i, M_i)\}_{i\leq p}$, parameters θ are trained to reconstruct matrix M_i from the jointly estimated matrix W_i:

$$\underset{\forall i\leq p,\; W_i\geq 0, H_i\geq 0, \theta}{\operatorname{argmin}} \; \sum_{i=1}^{p} \mathcal{D}\left(Y_i, W_i H_i^T\right) + \lambda \|W_i - M_i\theta\|_F^2.$$

Parameters θ^*, trained on the training dataset consisting of pairs (Y_i, M_i), could then be used at inference time when only a single matrix Y is known to estimate matrix $M := \theta^* W^\dagger$ after obtaining matrix W with NMF, or by solving the NMF and the regression problem

$$\operatorname*{argmin}_{W\geq 0, H\geq 0, M} \mathcal{D}\left(Y, WH\right) + \lambda \|W - M\theta^*\|_F^2.$$

We can still go further. In the formulations of supervised LRA above, while the post-processing parameters θ and the parameter matrices are optimized jointly and move the solution of the supervised LRA problem away from the best low-rank approximation, the model applied to Y is still a low-rank approximation. Modern-day machine learning often relies on black-box models based on neural network architectures that lack strong inductive biases, such as bilinearity and low-rankness. Therefore, it is tempting to also train the model, in some sense, to better fit the training data. Formally, we may assume that the forward model is a map $\mathcal{M}(W, H, \theta)$ and solve the optimization problem

$$\operatorname*{argmin}_{\forall i\leq p,\ W_i\geq 0, H_i\geq 0, \theta} \sum_{i=1}^{p} \mathcal{D}\left(Y_i, \mathcal{M}(W_i, H_i, \theta)\right) + \lambda \|W_i - M_i\theta\|_F^2. \tag{19.1}$$

An immediate issue with this formulation is that it is unclear how to even define such a map $\mathcal{M}$ while ensuring that the matrices W and H remain interpretable in practice. Another issue is that computing the minimizers using first-order methods requires computing the derivatives of $\mathcal{M}$ with respect to both the matrix W and the parameters θ, which can be challenging and time-consuming depending on the definition of the model $\mathcal{M}$. As far as I know, supervised LRA models in the same form as Equation (19.1) have not been studied in the literature. Rather, a more convenient way of designing data-driven LRA models is through the lens of bilevel optimization.

19.1.2 Bilevel formulation for data-driven NMF

On top of the issues discussed above, a fundamental problem with supervised LRA as defined in Equation (19.1) is also the choice of hyperparameter λ. Why should the user compromise between the quality of the forward pass of the model and the training of that model?

Bilevel formulations address the trade-off between model inference and parameter updates during training. There is, however, not a single canonical bilevel formulation for unrolling LRA. On the above example of supervised NMF, a naive formulation that separates the model computation (forward pass) and the actual training of the model (backward pass, *i.e.*, updating model parameters θ to reduce a training loss) writes

$$\operatorname*{argmin}_{\theta} \sum_{i=1}^{p} \|M_i - W_i^*\theta\|_F^2 \quad \text{such that} \quad H_i^*, W_i^* = \operatorname*{argmin}_{W_i\geq 0, H_i\geq 0} \mathcal{D}(Y_i, W_i H_i^T).$$

Such a bilevel optimization problem is not particularly interesting because the two optimization problems are essentially decoupled and can be solved sequentially. We are back to simply post-processing the estimated parameter matrices of NMF. Following data-driven or task-driven dictionary learning [Mairal *et al.*, 2011, Sprechmann *et al.*, 2014], the symmetry between matrices W and H may be broken. We then solve a bilevel problem of the form

$$\operatorname*{argmin}_{\theta,\, W} \sum_{i=1}^{p} \mathcal{L}(M_i, H_i^*(W), \theta) \quad \text{such that} \quad H_i^*(W) = \operatorname*{argmin}_{H_i\geq 0} \mathcal{D}(Y_i, W H_i^T).$$

Hence, the dictionary W is now trained to reduce the training loss $\mathcal{L}$ while only the scores H_i are computed at the inner level. Updating matrix W means computing gradients through minimizers $H_i^*(W)$, which is tractable analytically depending on the choice of the model, loss $\mathcal{D}$, and in the presence of regularizations such as sparsity [Mairal *et al.*, 2011]. It is, however, not obvious how to form these gradients for NMF.

The core idea of unrolling is to replace the inner optimization problem with a numerical algorithm that computes solutions to this problem:

$$\operatorname*{argmin}_{\theta,\, W} \sum_{i=1}^{p} \mathcal{L}(M_i, H_i^*(W), \theta) \quad \text{such that} \quad H_i^*(W, \theta) = \mathcal{A}(Y_i, W, \theta),$$

where $\mathcal{A}$ is a parametric algorithm to compute approximately a solution to NMF. Algorithm $\mathcal{A}$ is chosen to be a fixed number of iterations of a truncated iterative algorithm. Note that the forward model / unrolled algorithm $\mathcal{A}$ may depend on parameters θ for greater flexibility.

One interesting property of unrolling is that even without training, the iterative algorithm that solves the problem, here NMF, typically works rather well for minimizing the supervision loss. In many problem instances, earlier contributions have addressed the problems at hand in a fully unsupervised manner. Therefore, finding a good initialization for the parameters θ and the matrix W is often straightforward.

19.1.3 Unrolling the Multiplicative Updates algorithm

Existing unrolled NMF algorithms break the symmetry between matrices W and H. Therefore, they are not well-suited to make use of training data in the form of pairs $(Y_i, (W_i^{gt}, H_i^{gt}))$, where W_i^{gt} and H_i^{gt} are ground-truth factors. A typical use case is source separation in remote sensing, where spectra and abundance maps may be provided alongside hyperspectral images. It is also possible to generate a synthetic training dataset in which one can produce both ground-truth matrices W and H.

We therefore propose to formulate data-driven NMF where both matrices W and H are outputs of the parametric algorithm, solving

$$\operatorname*{argmin}_{\theta} \sum_{i=1}^{p} \mathcal{L}(W_i^{gt}, H_i^{gt}, H_i(\theta), W_i(\theta)) \quad \text{such that} \quad \forall i \leq p,\ H_i(\theta), W_i(\theta) = \mathcal{A}(Y_i, \theta).$$

The main design choices for the unrolled algorithm are

- The (truncated) iterative algorithm $\mathcal{A}$
- The trained parameters θ.

In a series of works with Christophe Kervazo, we proposed unrolling a workhorse algorithm for NMF, the MU algorithm; see *Nonnegative regressions: NNLS and NNKL* for a detailed presentation. Other algorithms could be considered, but MU poses an interesting challenge: there are no obvious trainable parameters θ in the algorithm. For instance, the MU update for matrix W with Frobenius loss writes

$$W \leftarrow W * \frac{YH}{WH^TH}.$$

Unrolling strategies typically train a stepsize or a linear operator in the log-prior (*e.g.*, the finite difference operator). Strategies to unroll MU previously proposed by Nasser [Nasser *et al.*, 2022] replace both matrix H and the cross product H^TH with trainable matrices. However, this strategy is not suited for an alternating procedure since the dependence on H is lost.

We proposed introducing trainable parameters that are multiplied elementwise with the updates. At iteration k, the proposed Non-Adaptive Linearize MU (NALMU) is given by

$$W^{k+1} = W^k * A_W^k * \frac{YH^k}{W^kH^{k^T}H^k} \text{ and } H^{k+1} \leftarrow H^k * A_H^k * \frac{Y^TW^{k+1}}{H^{k+1}W^{k^T}W^k}.$$

where A_W^k and A_H^k are iteration-dependant trainable matrices. There are two advantages to this choice:

1. Setting A_W^k and A_H^k to all-one matrices recovers MU. Therefore, the unrolled algorithm is easy to initialize, and we can understand how it differs from MU numerically.
2. We can prove that when updating a single matrix, say W with fixed H, and with shared weights A_W^k across all iterations, the modified updates of NALMU can be obtained by a majorization minimization strategy using Jensen inequality, see *Nonnegative regressions: NNLS and NNKL*, minimizing a modified cost function

$$\|Y - WH^T\|_F^2 + \langle (W * A_W)H^T, Y \rangle,$$

where the data is compared to a masked NMF with factors $W * A_W$ and H. The trained parameters, therefore, act as weights emphasizing the reconstruction towards certain entries of W.

19.1.4 Training NALMU

The supervision loss for NALMU is defined as

$$\mathcal{L} = \sum_{i=1}^{p} \sum_{k=1}^{K} \nu_k \left(\ell_W(W_i^k(\theta), W_i^{gt}) + \ell_H(H_i^k(\theta), H_i^{gt}) \right)$$

where $W_i^k(\theta)$ and $H_i^k(\theta)$ ar the estimated factors from algorithm $\mathcal{A}$ after $k \leq K$ iterations. Functions ℓ_W and ℓ_H are user-defined loss functions for the factor matrices, typically ℓ_2 norms or application-specific metrics such as the Spectral Angular Distance (SAD) used in remote sensing. Parameters ν_k control how much the estimated factor after k iterations impacts the supervision loss; for instance, if only the final output of algorithm $\mathcal{A}$ should match the ground-truth, then $\nu_k = 0$ for any $k < K$. Setting nontrivial values for parameters ν_k avoids training issues such as vanishing gradients and is a common trick in the unrolling literature and more generally in deep learning.

The initial weights A_W and A_H can be set to one to start the learning phase from the MU algorithm. The number of truncated iterations K is, in general, set rather low compared to the maximum number of iterations used in MU. Typically, we choose $K = 25$. The training algorithm can be any classic optimizer in the deep learning community with default parameters. For instance, the AdamW optimizer with a learning rate of 10^{-5} for 1000 epochs.

An issue with unrolled NMF is the scaling ambiguity, which makes the training less consistent. A simple solution we propose is to normalize the columns of the data matrices Y_i when appropriate, and to normalize the initial guesses for the factors accordingly. As a rule of thumb, normalisation in regularized and unrolled LRA can be tricky and should be designed depending on the application at hand.

19.2 Toy example

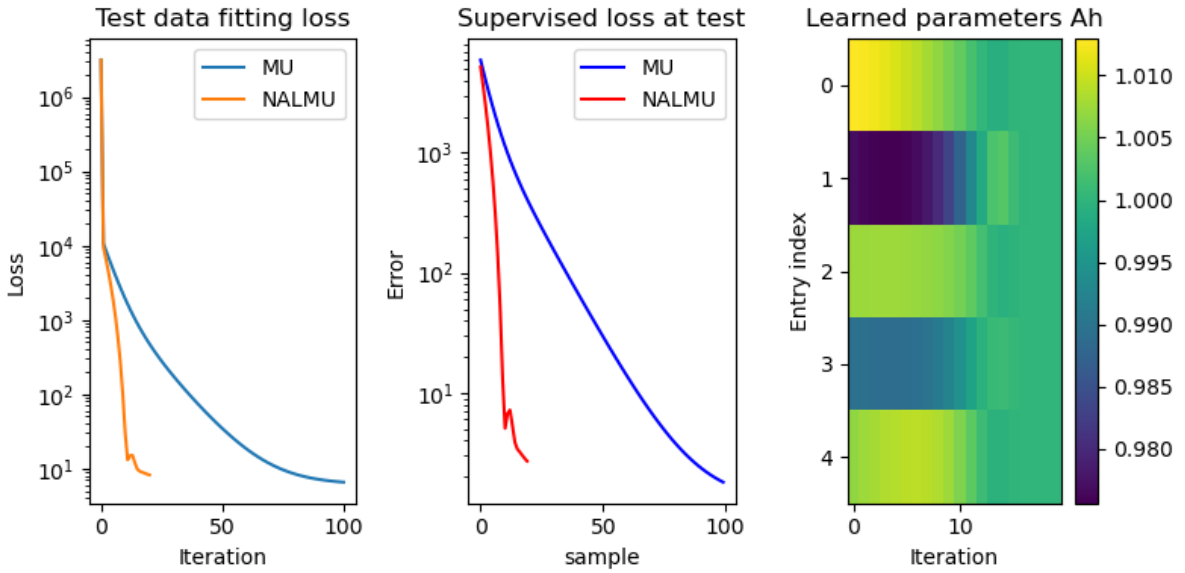
Providing a full example of unrolled NMF in this manuscript is rather challenging, as the training of NALMU typically requires large computing resources. Rather, a minimal working example is implemented below to train NNLS (i.e., NMF with a fixed factor matrix, here W) on synthetic images generated from a noisy mixture. The goal of this toy experiment is to showcase one key feature of unrolled algorithms: they often provide a good reconstruction in much fewer iterations than the baseline algorithm.

```
import torch
import matplotlib.pyplot as plt
# runs on CPU

# Hyperparameters
n = 5
m = 20
p = 500
sig = 0.1
K = 20
itermax = 1000
nu = torch.logspace(-7, 0, K)  # logspaced weights
# Play with the code: you can disable distillation
#nu = torch.zeros(K)
#nu[K-1] = 1.0

# define training data
torch.manual_seed(2)
mu = 50*torch.rand(n, 1)
H = torch.randn(n, p) + mu  # H are generated around a true value mu
W = torch.rand(m, n)
Y = W@H + sig*torch.randn(m, p)
```

(continues on next page)

(continued from previous page)

```
# define trainable parameters
Ah = torch.ones(n,K, requires_grad=True)
#Ah = torch.ones(n, requires_grad=True)  # for tied weights

# define loss function
def loss_fn(Y, W, H):
    return torch.norm(Y - W@H)**2

# define MU updates
def MU_update(W, Y, Xinit, itermax=1000, Xgt=None):
    X = Xinit.clone().detach()
    WtW = W.T@W
    WtY = W.T@Y
    loss = [loss_fn(Y, W, X)]
    sup_loss = []
    for i in range(itermax):
        X = X*WtY/(WtW@X)
        loss.append(loss_fn(Y, W, X))  # can be optimized using stored quantities
        if Xgt is not None:
            sup_loss.append(torch.norm(X - Xgt)**2)
    return X, loss, sup_loss

# define unrolled MU updates
def NALMU(W, Y, Xinit, Ah, itermax=25, Xgt=None):
    X = Xinit.clone().detach()
    loss = [loss_fn(Y, W, X)]
    WtW = W.T@W
    WtY = W.T@Y
    Xs = []
    sup_loss = []
    for i in range(itermax):
        XAh = (X.T*Ah[:,i]).T
        #XAh = (X.T*Ah).T  # for tied weights
        X = XAh*WtY/(WtW@X)
        Xs.append(X)
        loss.append(loss_fn(Y, W, X))  # can be optimized using stored quantities
        if Xgt is not None:
            supl = torch.norm(X - Xgt)**2
            sup_loss.append(supl.detach().numpy())
    return X, Xs, loss, sup_loss

# define optimizer
optimizer = torch.optim.Adam([Ah], lr=1e-2)   # using the Adam optimizer for␣
 ↪simplicity

# Training
for epoch in range(500):
    optimizer.zero_grad()
    Hinit = torch.ones(n, p)
    H_est, Hs, _, _ = NALMU(W, Y, Hinit, Ah, itermax=K)
    loss = sum([nu[i]*torch.norm(Hs[i] - H)**2 for i in range(K)])  # weighted loss
    loss.backward()
    optimizer.step()
    if epoch % 10 == 0:
        print(f'Epoch {epoch}, Loss: {loss.item()}')
```



A few lessons to learn from this toy experiment:

- The choice of the ν_k values greatly affect how far the weights A_W are from one. With the proposed choice (logarithmically spaced values in $[10^{-7}, 1]$), the weights in the last layers are barely updated. Try setting all ν_k to zero except the last one: the result is reversed, and the overall performance of NALMU decreases!
- Tied weights (when A_W does not depend on the iteration index) do not perform well in this example. This can be checked by changing the definition of A_W and updating the NALMU rule accordingly. In particular, NALMU with tied weights has difficulty reducing the NMF loss $\|Y - WH^T\|^2$ over iterations.
- Unrolling algorithms is tricky in practice, and one needs to toy with the various hyperparameters and design choices.

Part VI

Applications of rLRA

CHAPTER

TWENTY

APPLICATIONS OF RLRA: SUMMARY

Regularizations in LRA are necessarily driven by the properties of the low-rank factors one seeks to recover from the computation of LRA. Therefore, by understanding the applications of LRA in signal processing, one can derive both interesting fundamental problems related to LRA and ideas for designing regularizations and algorithms. Conversely, in some applications, such as medical imaging, theoretical guarantees on the quality of the reconstructed factors in rLRA are important because interpretation errors in these methods' outputs may have a significant impact. In an *optical biopsy application* I contributed to, rLRA outputs allow a surgeon to decide whether to remove brain cells during surgery. Removing too few may lead to cancer recurrence, but removing too much may harm critical brain functionalities.

In my work, I have focused on a few specific modalities to gain expert knowledge in these fields and making actually useful practical contributions based on rLRA. The first modality is *spectral imaging*, both in remote sensing and in microscopy, and the second is music information retrieval, in particular *automatic music transcription*. Around the end of my PhD, I also worked on chemometrics datasets that included fluorescence spectroscopy [Cohen *et al.*, 2016], chromatography, and nuclear magnetic resonance, which share similarities with spectral imaging in terms of signal processing.

20.1 Spectral imaging and rLRA

20.1.1 Basics of spectral representation of light

Note: The following paragraphs deal with radiative transfer, a scientific discipline in optics and physics that I am not very familiar with. In particular, I have not read many scientific articles on this topic and have instead used mediation materials, such as those found on Wikipedia. Here are some links for radiative transfer, color, and additive and substractive mixtures.

In everyday life, color helps distinguish different objects and materials in space. Color, in fact, carries a lot of information about our environment: one would probably never eat a light-blue apple because this unusual color is a sign that the apple may contain unwanted chemical components. Tree leaves are typically orange-ish when they dry and are about to fall, but typically green-ish in spring and summer. It is, therefore, a very natural and old idea to exploit color information in science to obtain information on an observable system.

From a mathematical and informatics point of view, defining color formally is, maybe surprisingly, a large and complex research domain. For simplicity, let us assume that colors can be represented in a three-dimensional space using the basis vectors red, blue, and green. Any two-dimensional m by n color image is, in this simplistic model, a tensor of size $m \times n \times 3$.

From a physical perspective, color is not a physical property of matter. It relates to human perception (and therefore varies for each individual) of a physical property called wavelength, which characterizes the distance an electromagnetic wave travels in a single cycle. More precisely, any wave can be decomposed in the spectral domain as the sum of sinusoidal waves with a fixed wavelength. This is exactly the Fourier transform ubiquitous in signal processing. Any light emitted or reflected by a source of interest has a specific spectrum that represents the amplitude of the individual additive sine

waves, and which contains information about the source, just like color. In fact, color, as described in the simplified RGB format, is a compressed representation of the full light spectrum, in which spectral coefficients are grouped into three blobs, respectively named red, green, and blue, see Fig. 20.1. This is mostly due to human perception: while human ears excel at identifying the Fourier coefficients (the amplitude of each additive sinusoidal wave) of air pressure wave, the human eye is much less sensitive; cones in the human eye that are sensitive to color naturally perform this dimensionality reduction.

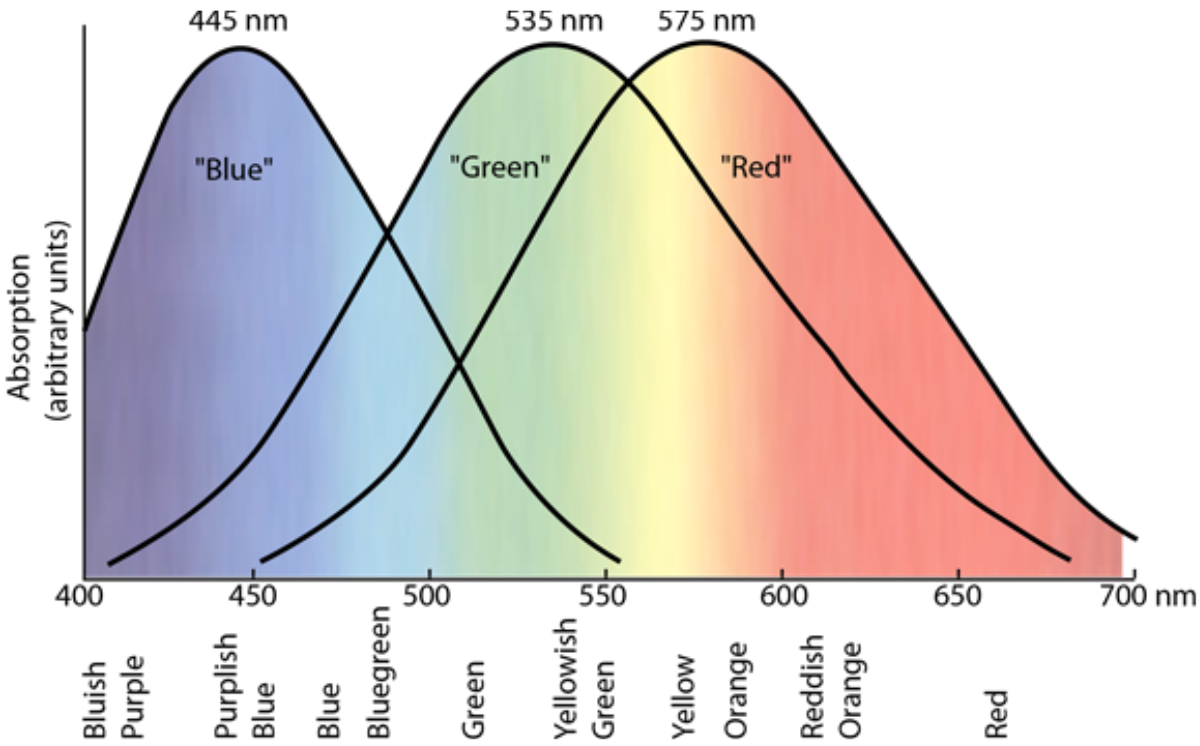


Fig. 20.1: [From hyperphysics.phy-astr.gsu.edu] *"There are three types of color-sensitive cones in the retina of the human eye, corresponding roughly to red, green, and blue sensitive detectors. Painstaking experiments have yielded response curves for three different kind of cones in the retina of the human eye. The "green" and "red" cones are mostly packed into the fovea centralis. By population, about 64% of the cones are red-sensitive, about 32% green sensitive, and about 2% are blue sensitive. The "blue" cones have the highest sensitivity and are mostly found outside the fovea. The shapes of the curves are obtained by measurement of the absorption by the cones, but the relative heights for the three types are set equal for lack of detailed data. There are fewer blue cones, but the blue sensitivity is comparable to the others, so there must be some boosting mechanism. In the final visual perception, the three types seem to be comparable, but the detailed process of achieving this is not known. When light strikes a cone, it interacts with a visual pigment which consists of a protein called opsin and a small molecule called a chromophore which in humans is a derivative of vitamin A. Three different kinds of opsins respond to short, medium and long wavelengths of light and lead to the three response curves shown above. For a person to see an object in color, at least two kinds of cones must be triggered, and the perceived color is based on the relative level of excitation of the different cones."*

This fact has a few interesting consequences. First, the human eye is blind to wavelengths outside the visible range, limiting our ability to detect spectral information in the infrared or ultraviolet. Second, two objects of the same color can have very different spectra. Therefore, the human eye is a poor spectral sensor. Hopefully, it is possible to accurately measure the light spectra of a single light flux using prisms that have the property to spread light beams spatially depending on the wavelength. In other words, prisms perform, in some way, a Fourier transform of an incoming light wave and output the individual sinusoidal light waves with pure wavelengths separated spatially. It is then possible to measure the spectrum by essentially taking a black-and-white picture of the prism output. Devices that measure light spectra, using prisms or

other separation means, are called spectrometers.

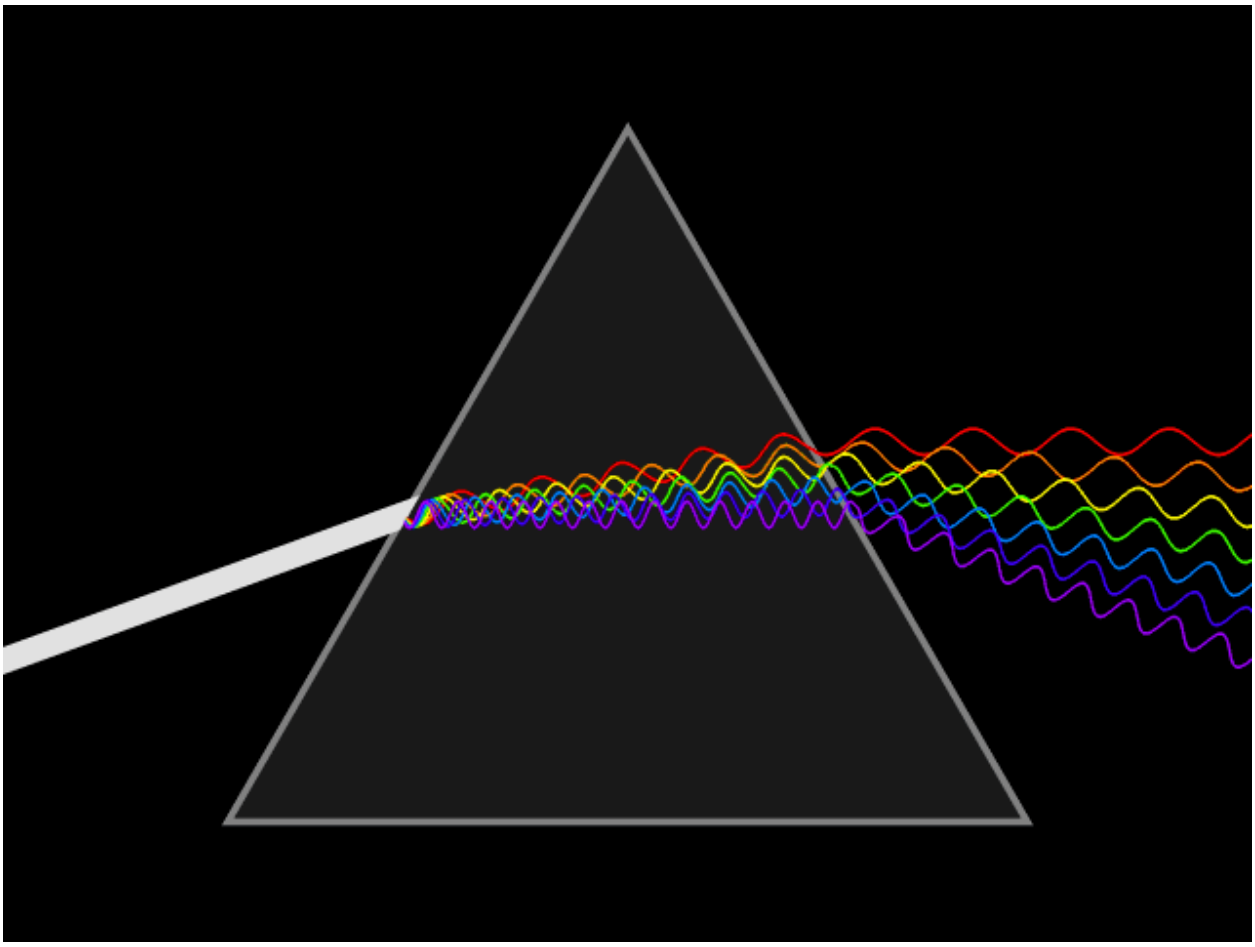


Fig. 20.2: [From Wikipedia, credits Lucas Vieira] An illustration of the dispersion property of a prism. Incomming white light is decomposed as a sum of monochromatic light beams with increasing wavelength.

20.1.2 Spectral mixing and unmixing

We are taught in high school that colors obey two sets of rules for mixing: additive and subtractive. Additive mixture works for light and is typically described in the RGB color space: red + blue light yields magenta, red + blue + green light yields white. This principle is ubiquitous in everyday life, since color screens rely on it to generate colors by combining red, blue, and green light from tiny light sources for each pixel. Subtractive synthesis applies, for instance, to paint, typically in CMY format: cyan + magenta yield blue, cyan + magenta + yellow yield black. The coexistence of these two models can be slightly confusing: when is a mixture additive or subtractive? Is it possible to design additive mixtures of paints and subtractive mixtures for light beams?

A way to explain additive and subtractive mixtures, which I find useful, is to relate them to the physical process underlying the mixture and to the full spectral description of color. An additive mixture is a matter of perception. Various light beams with different spectra hit the human eye and are seen as a single light source by the lens, which focuses the input light onto the cones, integrating the contributions of each light source. In the case of a spectrometer, an optical lens can be used to produce a similar additive acquisition. Addition synthesis is therefore not a physical modification of the wavelengths of a light wave, but rather the (spatial) superposition of different light waves. In contrast, a subtractive mixture is a direct modification of the spectrum of a light wave that removes a part of that spectrum. A key concept to understand a subtractive mixture is filtering: when a light wave hits an object, it is partially absorbed and partially reflected. The

intensity of reflected light depends on the wavelength. The reflected light has a different spectrum than the input light, filtered by the object's spectral response. What we usually call the color of an object is, in fact, the projection in RGB space of the resulting spectrum of "white" light after it has been reflected by the item. The absorption spectrum of an object is the spectral filter by which the incoming light is multiplied to obtain the reflected light spectrum. A subtractive mixture is simply the sequential filtering of a light source by several items, where spectral filters are combined multiplicatively. When mixing paints, the chemical compounds in each paint are intimately mixed, and the paint after mixture essentially filters light jointly for all paints.

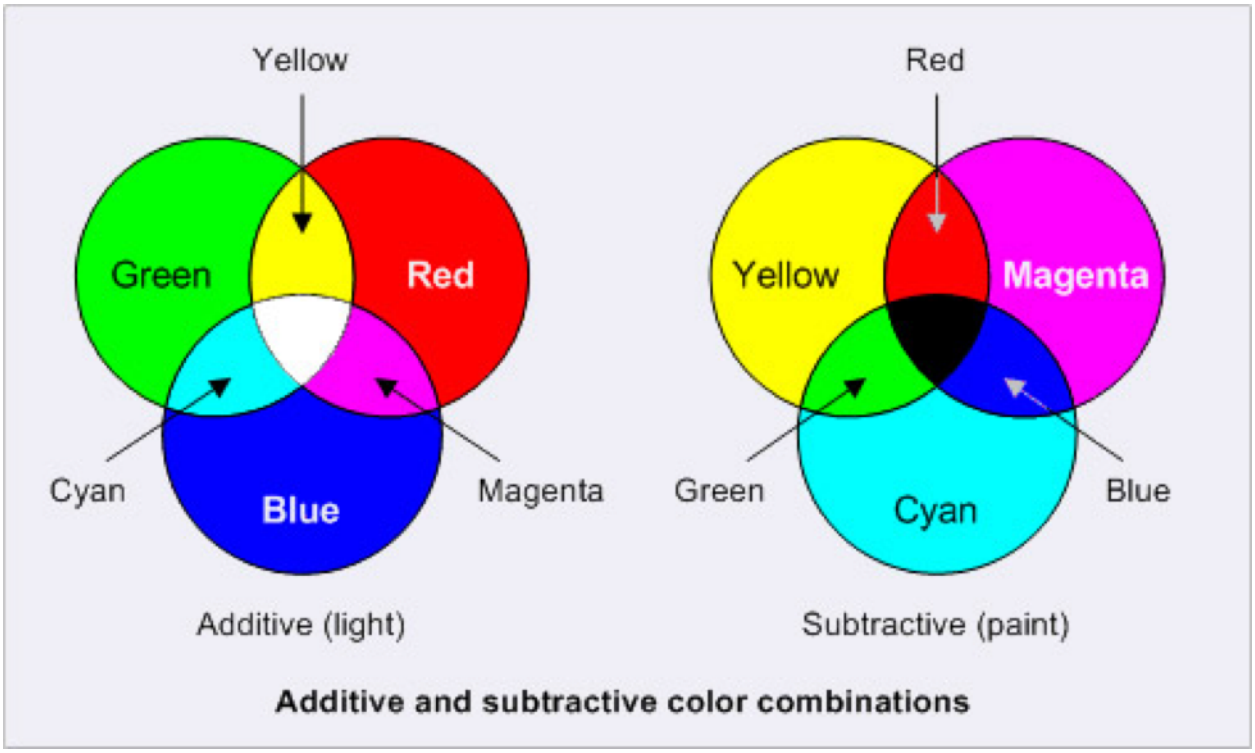


Fig. 20.3: [From intranet.mcad.edu] An illustration of additive and substractive mixtures. Additive mixture is essentially a property of light, and allows to generate a large range of pure wavelengths in the visible range. Substractive mixture is obtained by spectral filtering and is less expressive in general than RBG.

In scientific imaging, both additive and subtractive mixtures are usually encountered. The important point is that an additive mixture is usually well-suited to linear models, whereas a subtractive mixture leads to non-linear models. Two examples of additive mixture help illustrate this fact.

a. **Fluorescence Spectroscopy in chemometrics**. A mixture of several fluorophores is observed with a spectrometer. Each fluorophore emits a light wave with a specific spectrum. Since the fluorophores are loosely mixed within the sample and each emits a fluorescence signal individually, the measured spectrum is simply the sum of the fluorescence spectra of each component. The light emitted by the fluorophores depends on the wavelength of the light exciting the sample; using a laser excitation with a controlled wavelength leads to a series of spectra acquisitions, stored in a matrix (fluorescence excitation emission matrix) Y with n columns corresponding to excitation wavelengths and m rows containing the measured additive mixture of fluorescence spectra. Additive mixture in this context is also related to the Beer-Lambert law, and translates into an (approximate) low-rank NMF

$$Y \approx WH^T$$

where W is a matrix containing the spectra of each individual fluorophore in the chemical mixture, and H contains the amplitude of each fluorophore response to the excitation wavelength. Applying NMF to the data matrix Y, or nonnegative tensor factorization to several such measurement matrices, can in principle recover the individual fluorescence spectra, essentially performing spectral unmixing.

b. **The linear mixing model in remote sensing** makes the hypothesis that materials on an observed scene are spatially distributed and non-overlapping. The scene is cut into pixels by the camera, and each pixel may therefore contain several materials with proportions given by the portion of the pixel covered by each material. The spectral acquisition is then the additive mixture of the reflectance spectra (the ambient-light spectrum, essentially white, filtered by each material). For a single-pixel $Y[:,i]$ of the acquired spectral image Y with m spectral wavelengths (or spectral bands if spectra are acquired in a compressed spectral representation) and n pixels, the additive mixture of K materials simply translates into a linear model

$$Y[:,i] \approx \sum_{k=1}^{K} W[:,k]H^T[k,i];\ Y \approx WH^T$$

where W contains columnwise the spectra of each material in the scene (supposing they are consistent over the whole image), and H contains columnwise the proportions of the materials in each pixel, also called abundances. Extensions of the linear mixing model that account for multiple reflections typically involve products of the matrix W with itself; this is consistent with the subtractive mixture model, where the ambient light is filtered consecutively by several materials [Kervazo *et al.*, 2021].

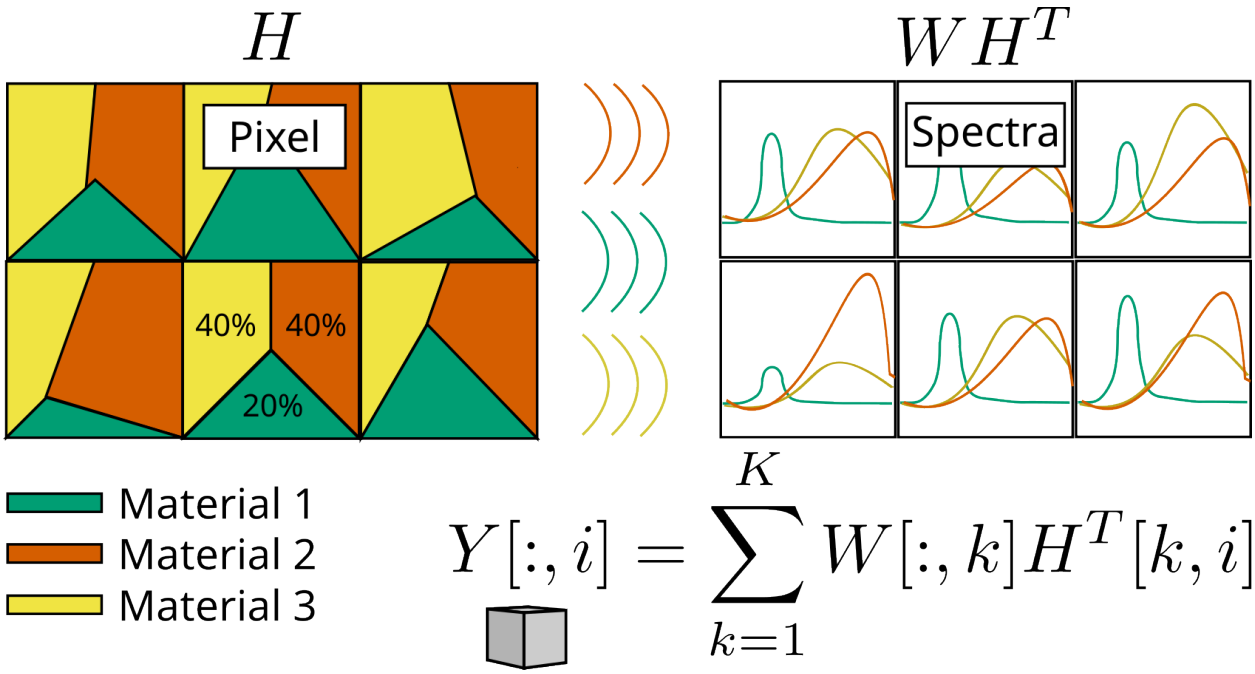


Fig. 20.4: An illustration of the linear mixing model. Each pixels (squares in the top left grid) have different abundances of each material. This translates into an additive mixture of the spectra contained in matrix W, weighted by these abundances pixelwise.

To conclude this introduction to spectral unmixing, we can now answer the question above: Is it possible to mix paints additively to produce white? The answer is nuanced. It is impossible to mix paint to produce pure white. Mixing paints intimately will result in a filtering effect that attenuates the ambient light spectrum across all wavelengths, leading to black. However, we can produce grey by juxtaposing paints on a surface and looking from afar. Spatial juxtaposition will result in additive filtering of the filtered white light, as in the linear mixing model. If three paints red, blue, and green are used in equal proportions, the resulting spectrum is the sum of blue, red, and green light but with reduced intensity, and the object will appear gray. Screens can produce white because they can emit red, blue, and green spectra at full intensity, which is not possible with reflectance spectra.

20.1.3 Several applications of spectral unmixing

Spectral unmixing separates spectra from several acquisitions of additive mixtures with varying mixture conditions. A popular example of spectral unmixing in the signal processing community is found in remote sensing, and I have often motivated rather theoretical works with such images. Since my arrival at CREATIS in 2022, however, my work has focused on optical systems that enable optical biopsy. Optical biopsy is a non-invasive technique that enables the acquisition of spectral information from tissues (*e.g.*, brain cells) during live surgery without requiring tissue extraction. The idea is to design lightweight optical systems with high-end processing pipelines to acquire spectral images, and then use both spatial and spectral information to infer tissue nature (*e.g.*, whether brain cells are cancerous). Most of my work on this topic has been dedicated to the joint reconstruction and spectral unmixing for the *single-pixel spectral camera*, but I have also contributed to *a system working with RGB videos*.

20.2 Automatic music transcription

Music information retrieval is a collection of music-oriented machine learning tasks, ranging from tempo detection to automatic music composition. The tasks that compose music information retrieval are always evolving (see, for instance, the MIREX competition that lists old and new tasks yearly). Automatic Music Transcription (AMT) is a challenging task that aims at converting an audio recording into a MIDI file [Benetos *et al.*, 2018, Bertin, 2009, Bittner *et al.*, 2022, Smaragdis and Brown, 2003]. Polyphonic instruments are typically the most challenging to transcribe, but singing voice or wind/brass instruments also have inherent difficulties despite being monophonic, such as intonation. In what follows, we focus primarily on piano transcription.

Remark

MIDI (Musical Instrument Digital Interface) is a numeric standard for storing and exchanging symbolic music notation between musical instruments and computers. In the context of music notation, it contains for each played note its pitch (in the Western system from C0 to A7), its onset (the activation time with respect to the song start), its offset, and its velocity (nuance) in 8 bits. While MIDI representations are in many regards much more precise than usual musical notations found on music sheets, it is often insufficient to describe the full interaction of the artist with the instrument: velocity may be time-dependent, pitch can be bent or microtonal, and activation may not be instantaneous.

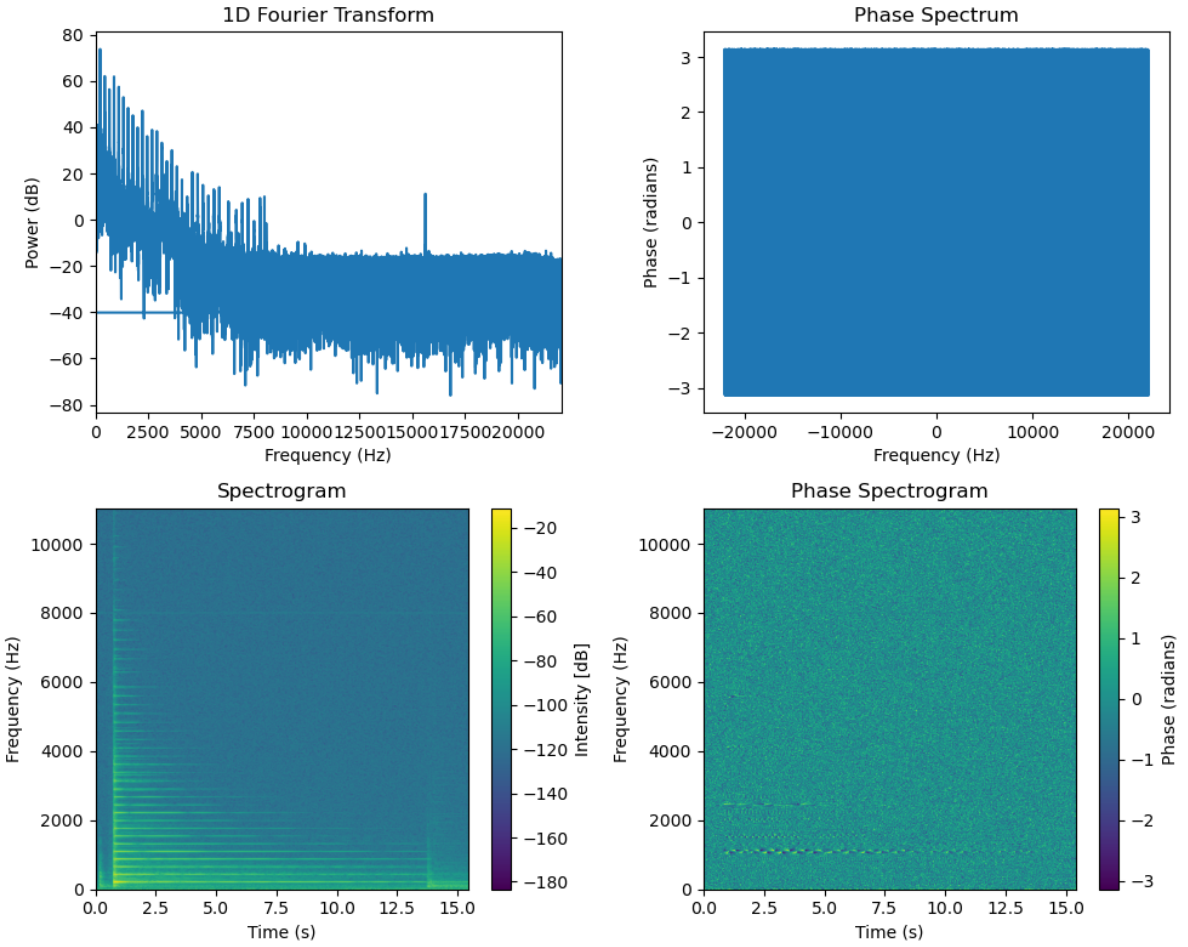
Among existing AMT algorithms, many rely on a time-frequency representation of the audio signal. The rationale is that for a single note played on the piano, a comb-shaped spectrum in the Fourier domain corresponds to the fundamental frequency (440Hz for A4) and all harmonics. These harmonics, and in particular their relative intensity and position, define the tone of the piano. Harmonic instruments are characterized by the existence of this comb-shaped spectrum, while inharmonic instruments, such as drums, do not produce comb-shaped spectra. This is easily observed on a single note recording, here extracted from the MAPS database [Emiya *et al.*, 2010].

```
from cmath import phase
import matplotlib.pyplot as plt
import soundfile as sf
import numpy as np
import scipy.signal as signal

A4_wav, sr = sf.read('../../tensorly_hdr/dataset/MAPS_ISOL_LG_F_S1_M57_ENSTDkAm.wav')
A4_wav = A4_wav / np.max(np.abs(A4_wav)) # audio normalization
A4_wav = A4_wav[:,0] # use only one channel if stereo

# Compute 1D Fourier transform
frequencies_1D = np.fft.fftfreq(len(A4_wav), d=1/sr)
fourier_1D = np.fft.fft(A4_wav)
```

(continues on next page)

(continued from previous page)

```
magnitude_1D = np.abs(fourier_1D)
phase_1D = np.angle(fourier_1D)

# Plot 1D Fourier transform
plt.figure(figsize=(10, 8))
plt.subplot(2, 2, 1)
plt.plot(frequencies_1D, 20*np.log10(magnitude_1D))
plt.title('1D Fourier Transform')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power (dB)')
plt.xlim(0, sr/2)
plt.subplot(2, 2, 2)
plt.plot(frequencies_1D, phase_1D)
plt.title('Phase Spectrum')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Phase (radians)')
#plt.xlim(0, sr/2)

# Compute spectrogram
frequencies, times, Sxx = signal.stft(A4_wav, fs=sr, nperseg=4096, nfft=4096,␣
↪noverlap=4096 - 882)
Sxx_dB = 20 * np.log10(np.abs(Sxx) + 1e-10)  # convert to dB scale
Sxx_abs = np.abs(Sxx)
Sphase = np.angle(Sxx)
#Sphase_smoothed = signal.medfilt(Sphase, kernel_size=(25, 5))
half_freqs = len(frequencies) // 2

# Plot power spectrogram (up to 10kHz)
plt.subplot(2, 2, 3)
#plt.imshow(np.abs(Sxx), aspect='auto')
# use correct extent to display time and frequency axes
plt.imshow(Sxx_dB[:half_freqs,:], aspect='auto', origin='lower', extent=[times.min(),␣
↪times.max(), frequencies.min(), frequencies[half_freqs]])
plt.title('Spectrogram')
plt.ylabel('Frequency (Hz)')
plt.xlabel('Time (s)')
plt.colorbar(label='Intensity [dB]')

# Phase spectrogram
plt.subplot(2, 2, 4)
plt.imshow(Sphase[:half_freqs,:], aspect='auto', origin='lower', extent=[times.min(),␣
↪times.max(), frequencies.min(), frequencies[half_freqs]])
plt.title('Phase Spectrogram')
plt.ylabel('Frequency (Hz)')
plt.xlabel('Time (s)')
plt.colorbar(label='Phase (radians)')
plt.tight_layout()
plt.show()
```



We show above the spectrum of the full audio recording (about 15s) in both magnitude and phase (which is arguably harder to interpret directly). Using a single spectrum to describe the frequency content of a long recording is often innadequate. Over the 15 seconds of recording, there are times when the note is not played or when the timbre changes. High frequencies tend to dissipate faster than low frequencies. Therefore, in audio signal processing, the spectrogram (magnitude and phase) that contains spectra for small time windows is incredibly useful. We see that the magnitude spectrogram of a single note contains rich information. The comb-shaped spectra fades over time, the hammer action is visible at around 0.5s, and some noise occurs at around 12s. The phase spectrogram works similarly and, while it is harder to read, it contains a lot of information. In particular, we can decipher the various partials of the comb. For AMT, we typically use only the magnitude spectrogram and discard the phase spectrogram.

An interesting property of the spectrogram of a single note is that it is well approximated by a rank-one matrix. This suggests using a low-rank model to analyse more complex signals and perform AMT. NMF for AMT, however, has several issues: the rank-one hypothesis is overly strong, the source mixture is non-linear, and there are too few guarantees of uniqueness. I addressed these issues partially in my work on weakly-supervised convolutive NMF for AMT; my contributions are detailed in *Automatic Music Transcription*.

20.3 Music structure estimation

We can also use spectrograms of full songs to detect similarities between bars. I will not detail this contribution in this manuscript; it was already detailed in length in Axel Marmoret's PhD manuscript [Marmoret, 2022]. There are also available tutorials in the BarMusComp toolbox link and the more recent Autosimilarity Segmentation toolbox link. The main methodological tool to enhance the similarity detection is the Nonnegative Tucker Factorization of the tensor spectrogram obtained by stacking spectrograms of each bar of a song [Smith and Goto, 2018]. See also *the related paragraph in the HDR summary*.

CHAPTER

TWENTYONE

AUTOMATIC MUSIC TRANSCRIPTION

Reference

[Wu *et al.*, 2022] H. Wu, A. Marmoret and J. E. Cohen, "Semi-Supervised Convolutive NMF for Automatic Music Transcription", SMC2022, pdf code data

21.1 Nonnegative matrix factorization for automatic music transcription

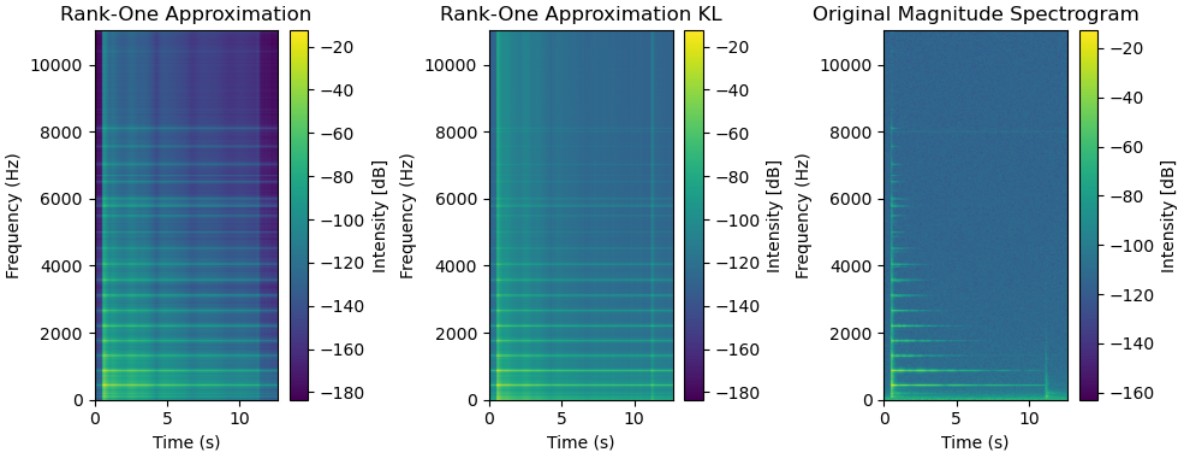
Spectrograms of piano recordings contain the spectral information of the notes played over time. An interesting property of the spectrogram of a single piano note recording is that it is typically well approximated by a rank-one matrix. For instance, on a recorded piano A4 (440Hz) from the dataset MAPS [Emiya *et al.*, 2010], the best rank-one approximation of the magnitude spectrogram looks similar to the magnitude spectrogram, in particular when using KL-divergence as a loss function, see *the end of the section on NNLS* for a discussion on the rank-one approximation closed-form algorithm.

```
from cmath import phase
import matplotlib.pyplot as plt
import soundfile as sf
import numpy as np
import scipy.signal as signal

A4_wav, sr = sf.read('../../tensorly_hdr/dataset/MAPS_ISOL_LG_M_S0_M69_ENSTDkAm.wav')
A4_wav = A4_wav / np.max(np.abs(A4_wav)) # audio normalization
A4_wav = A4_wav[:,0] # use only one channel if stereo

# Compute spectrogram
frequencies, times, Sxx = signal.stft(A4_wav, fs=sr, nperseg=4096, nfft=4096,␣
↪noverlap=4096 - 882)
half_freqs = len(frequencies) // 2
frequencies = frequencies[:half_freqs]
Sxx = Sxx[:half_freqs, :] # keep only lower half
Sphase = np.angle(Sxx)
Sxx_dB = 20 * np.log10(np.abs(Sxx) + 1e-10)  # convert to dB scale
Sxx_abs = np.abs(Sxx)

# Best rank-one approximation of magnitude spectrogram in the Frobenius sense
U,s,V = np.linalg.svd(Sxx_abs)
r1_approx = s[0]*np.outer(U[:,0],V[0,:])
# in dB
r1_approx_dB = 20 * np.log10(r1_approx + 1e-10)
```

(continues on next page)

(continued from previous page)

```
# Same in KL sense with NMF
ukl = np.sum(Sxx_abs, axis=1)/np.sqrt(np.sum(Sxx_abs))
vkl = np.sum(Sxx_abs, axis=0)/np.sqrt(np.sum(Sxx_abs))
r1_approx_kl = np.outer(ukl, vkl)

r1_approx_kl_dB = 20 * np.log10(r1_approx_kl + 1e-10)
```



Magnitude spectrograms are rather well approximated by rank-one matrices. This suggests using low-rank NMF to analyse recordings containing multiple notes. Assuming a linear additive mixture of time signals, however, does not yield an exact low-rank model of the magnitude spectrogram, as explained next.

21.1.1 Simplified model and low-rank spectrograms

Assume two notes are played simultaneously by the piano, with time signals $s_1(t)$ and $s_2(t)$. We focus on a single time window in the STFT. We assume linear mixing, and record the sum of the two signals $s(t) = s_1(t) + s_2(t)$. The Fourier transform of the summed signal $\mathcal{F}[s](\nu)$, by linearity of the Fourier transform, is exactly $\mathcal{F}[s_1](\nu) + \mathcal{F}[s_2](\nu)$.

As we saw in *Applications of rLRA: summary*, the phase of the spectrogram is not low-rank even for a single note, therefore we only want to compute the magnitude spectrogram $|\mathcal{F}[s](\nu)|$, where the modulus is taken elementwise. However if we assume that $|\mathcal{F}[s_i](\nu)|$ are rank-one matrices, we find that

$$|\mathcal{F}[s](\nu)|^2 = |\mathcal{F}[s_1](\nu) + \mathcal{F}[s_2](\nu)|^2 = |\mathcal{F}[s_1](\nu)|^2 + |\mathcal{F}[s_2](\nu)|^2 + 2\mathcal{R}e\left(\mathcal{F}[s_1]^*(\nu)\mathcal{F}[s_2](\nu)\right),$$

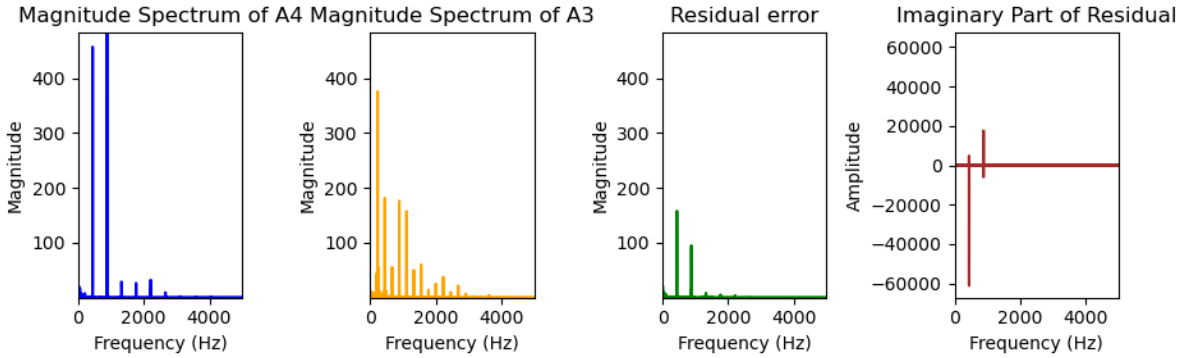
and by Cauchy-Schwartz, the spectrogram computed from the recording is rank-two if and only if $\mathcal{F}[s_1](\nu) = \lambda_\nu \mathcal{F}[s_2](\nu)$ for each ν and $\lambda_n u \geq 0$. This condition is equivalent to assuming that the two signals are in phase, *i.e.*, $\mathcal{I}m\left(\mathcal{F}[s_1]^*(\nu)\mathcal{F}[s_2](\nu)\right) = 0$.

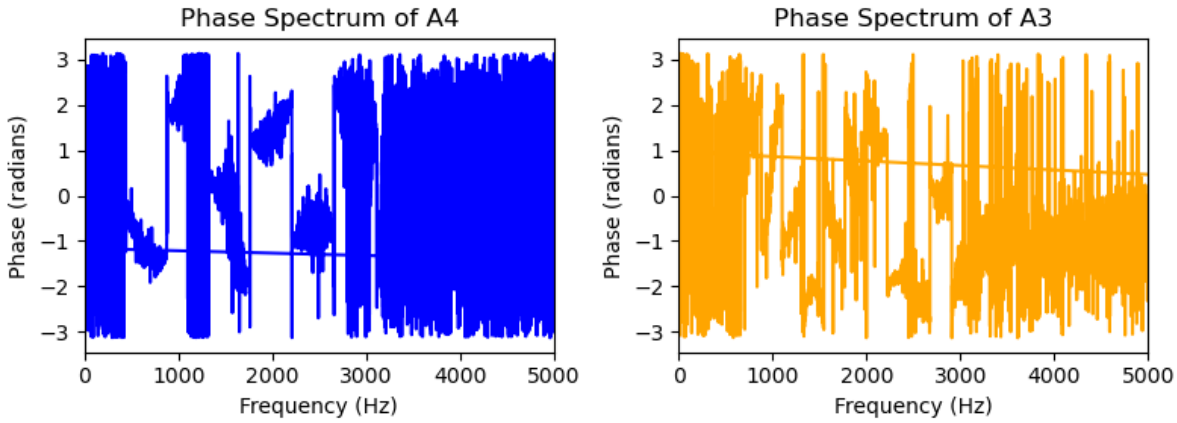
A particular case of the equality condition in Cauchy-Schwartz is obtained when the two frequency spectra have disjoint support, in which case λ_ν is null. In other words, if the notes $s_1(t)$ and $s_2(t)$ have no partials in common, then the spectrogram of their additive mixture in the temporal domain should remain low-rank, and one can hope to recover each rank-one spectrogram by performing a rank-two NMF. This assumption is related to time-frequency masking, a technique that has been used for decades in audio source separation and has remained in use at the start of the era of deep learning [Araki *et al.*, 2025].

The disjoint support hypothesis is useful, but incorrect in practice. For instance, two notes separated by an octave have similar spectra, but the colinearity condition will be significantly violated. Not only are the fundamental frequencies and partials different in amplitude, but the phase of each signal can also change. This can be observed on the MAPS dataset below with notes A4 and A3. Observe that the imaginary part of the cross product is, in particular, nonzero at locations

where the sum of the modulus is far from the modulus of the sum, and that this corresponds to partials of interest. This implies that methods based on low-rank approximations of spectrograms are particularly prone to octave errors.

```
from cmath import phase
from re import A
import matplotlib.pyplot as plt
import soundfile as sf
import numpy as np
import scipy.signal as signal

A3_wav, sr = sf.read('../../tensorly_hdr/dataset/MAPS_ISOL_LG_F_S1_M57_ENSTDkAm.wav')
A4_wav, sr = sf.read('../../tensorly_hdr/dataset/MAPS_ISOL_LG_M_S0_M69_ENSTDkAm.wav')
A4_wav = A4_wav / np.max(np.abs(A4_wav)) # audio normalization
A4_wav = A4_wav[:,0] # use only one channel if stereo
A3_wav = A3_wav / np.max(np.abs(A3_wav)) # audio normalization
A3_wav = A3_wav[:,0] # use only one channel if stereo
# Use short time window (rectangular window for simplicity)
t_1 = 2*sr
t_2 = 3*sr
A4_segment = A4_wav[t_1:t_2]
A3_segment = A3_wav[t_1:t_2]

# Compute 1D Fourier transform
frequencies_1D = np.fft.fftfreq(len(A4_segment), d=1/sr)
fourier_1D_A4 = np.fft.fft(A4_segment)
fourier_1D_A3 = np.fft.fft(A3_segment)
magnitude_1D_A4 = np.abs(fourier_1D_A4)
phase_1D_A4 = np.angle(fourier_1D_A4)
magnitude_1D_A3 = np.abs(fourier_1D_A3)
phase_1D_A3 = np.angle(fourier_1D_A3)

# Compute sum of magnitude and magnitude of sum
fourier_sum = fourier_1D_A4 + fourier_1D_A3
magnitude_sum = np.abs(fourier_sum)
fourier_magnitude_sum = magnitude_1D_A4 + magnitude_1D_A3
```





Note: AMT entered the MIREX competition as polyphonic transcription in 2024 (in fact, it was just piano, not guitar, organs, synthesizers, or others), even though it was already a research topic in the early 2000s. Single-instrument transcription remains an open problem, and there is much to do in a multi-instrument setting, particularly for joint source separation with AMT.

21.1.2 NMF for AMT: useful but limited

Remark

Spectrograms have a few hyperparameters that should be fixed, most importantly the time window size and type, and the amount of overlap for the windows. In the following, we use parameters from the literature [Cheng *et al.*, 2016]: Hann windows, 4096 samples per window, and 4096 - 882 overlapping samples. This yields a time resolution of about 20ms (the sample rate is usually 44kHz), with spectra computed over 93ms, and a frequency resolution of about 10.7Hz, which is a reasonable time-frequency resolution trade-off for piano transcription.

Given a spectrogram Y of an audio recording, based on the observation that individual notes have approximately rank-one spectrograms, and assuming both linear mixture and marginally overlapping support of the spectrograms, one may hope to recover each individual note spectrogram with NMF. In fact, if the supports of the spectrograms of each note do not overlap, NMF is separable (each row of the data spectrogram belongs to exactly one note spectrogram), and therefore, the unique NMF solution can be recovered by any reasonable NMF algorithm. On top of estimating each individual spectrogram, these spectrograms are factorized into a temporal activation, which indicates when each note is played and at what intensity, and a frequency spectrum containing the fundamental and partials characteristic of the note. A simple post-processing step thus involves identifying the fundamental frequency for each rank-one NMF component, and thresholding the time-activation [Vincent *et al.*, 2008]. Although more elaborate strategies could be designed, this pipeline is essentially what has been used in the AMT literature to post-process NMF factors.

We can try this on a simple recording from the MAPS dataset [Emiya *et al.*, 2010] with many isolated notes and few chords. The first example below is performed on a song with only isolated notes (first three bars). Notice how good the reconstruction is. There are six distinct notes played in the first six seconds of the recording. We used a rank seven NMF. One of the component models the hammer action; this can be seen by the spectral signature, which is not comb-shaped, and the time activation following each played note, except the high F, which has a softer attack.

```
# Load song
song, sr = sf.read('../../tensorly_hdr/dataset/MAPS_MUS-muss_1_ENSTDkAm.wav')
song = song / np.max(np.abs(song)) # audio normalization
```

(continues on next page)

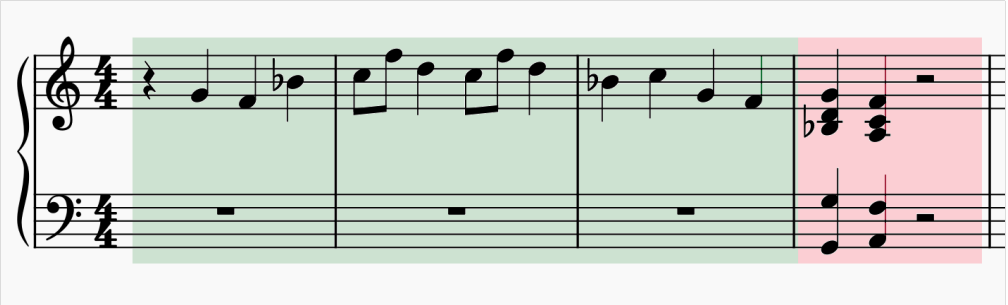
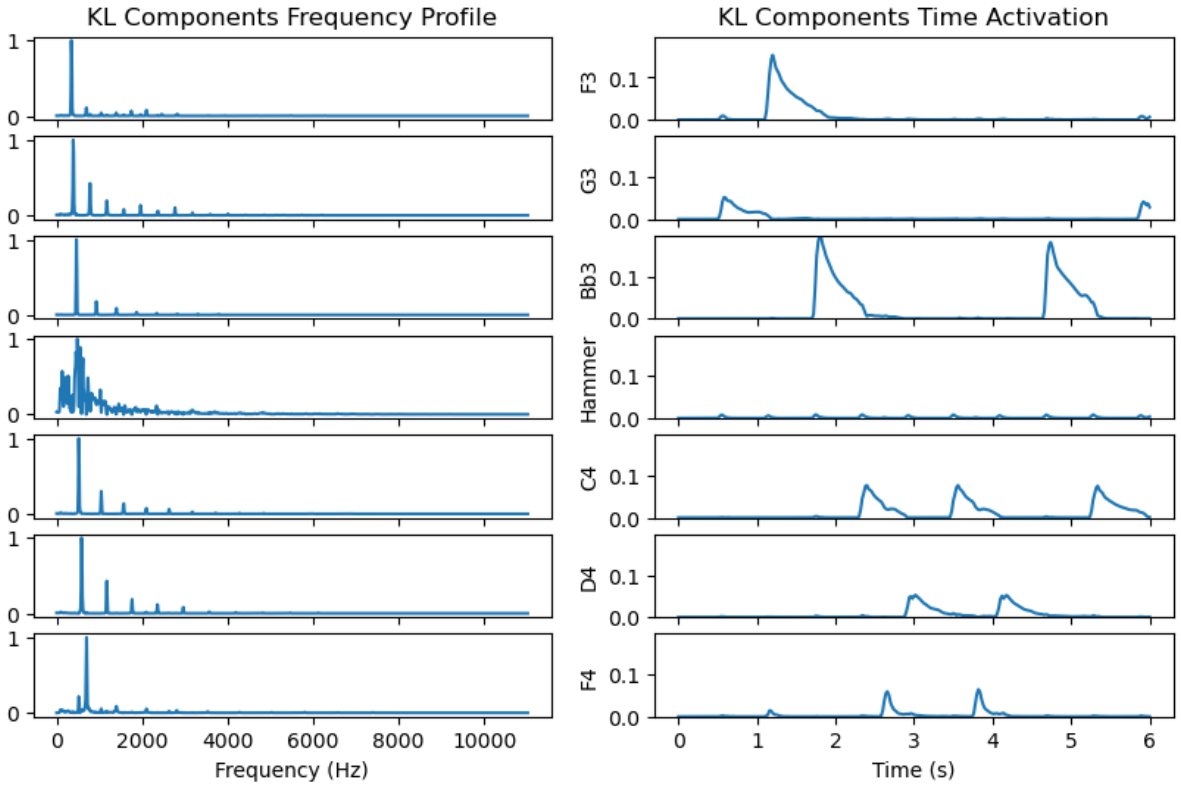
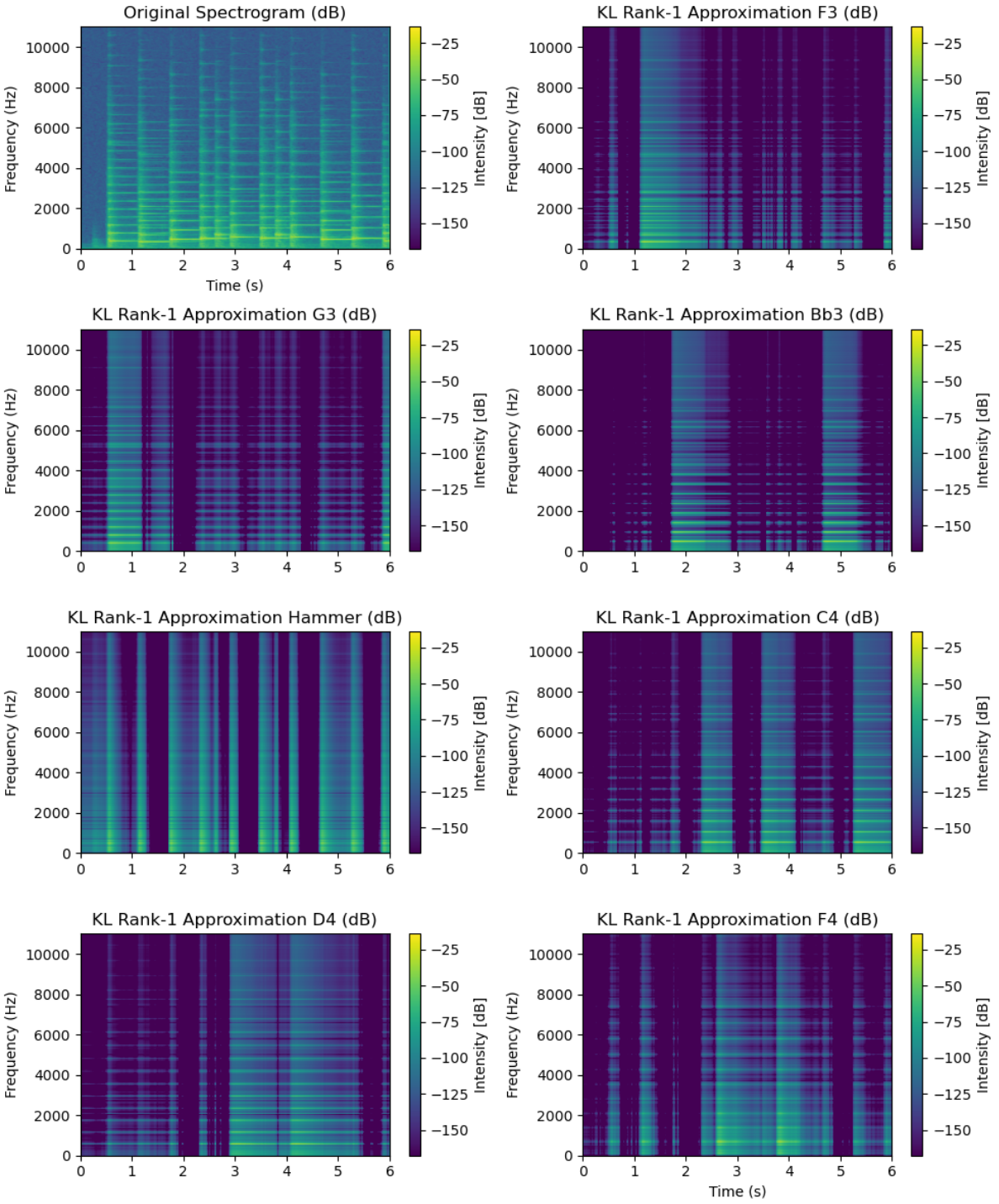


Fig. 21.1: Symbolic notation for the recorded audio sample. In green, the first six seconds with isolated notes, and in red, the second part, up to eight seconds, with chords.

(continued from previous page)

```
song = song[:,0] # use only one channel if stereo
# The song has six different single notes until 6s, then chords
# We cut the song after 6s (rectangular window for simplicity)
t_1 = 0*sr
t_2 = 6*sr
song = song[t_1:t_2]

# Compute spectrogram
frequencies, times, Sxx = signal.stft(song, fs=sr, nperseg=4096, nfft=4096,␣
 ↪noverlap=4096 - 882)
half_freqs = len(frequencies) // 2
frequencies = frequencies[:half_freqs]
Sxx = Sxx[:half_freqs, :] # keep only lower half
Sxx_dB = 20 * np.log10(np.abs(Sxx) + 1e-10)  # convert to dB scale
Sxx_abs = np.abs(Sxx)

# NMF with KL divergence (function taken from paper with Quyen, using alternating MU,␣
 ↪code from tensorly_hdr)
# Could also use spa or snpa on the rows
from tensorly_hdr.nmf_kl import Lee_Seung_KL
rank = 7
np.random.seed(0)  # for reproducibility
W_init = np.abs(np.random.randn(Sxx.shape[0], rank))
H_init = np.abs(np.random.randn(Sxx.shape[1], rank)).T
crit, W_kl, H_kl, toc, cnt = Lee_Seung_KL(Sxx_abs, W_init, H_init, NbIter=20,␣
 ↪verbose=False, print_it=20)
# Normalize W columnwise and put the norm in H
W_kl_norms = np.max(W_kl, axis=0)
W_kl = W_kl / W_kl_norms
H_kl = H_kl.T * W_kl_norms

# Permuting spectra in increasing fundamental frequency
from tensorly_hdr.image_utils import permute_spectra
perm = permute_spectra(W_kl)
W_kl = W_kl[:, perm]
H_kl = H_kl[:, perm]
```


KL Components Frequency Profile
KL Components Time Activation
F3
G3
Bb3
Hammer
C4
D4
F4
Frequency (Hz)
Time (s)



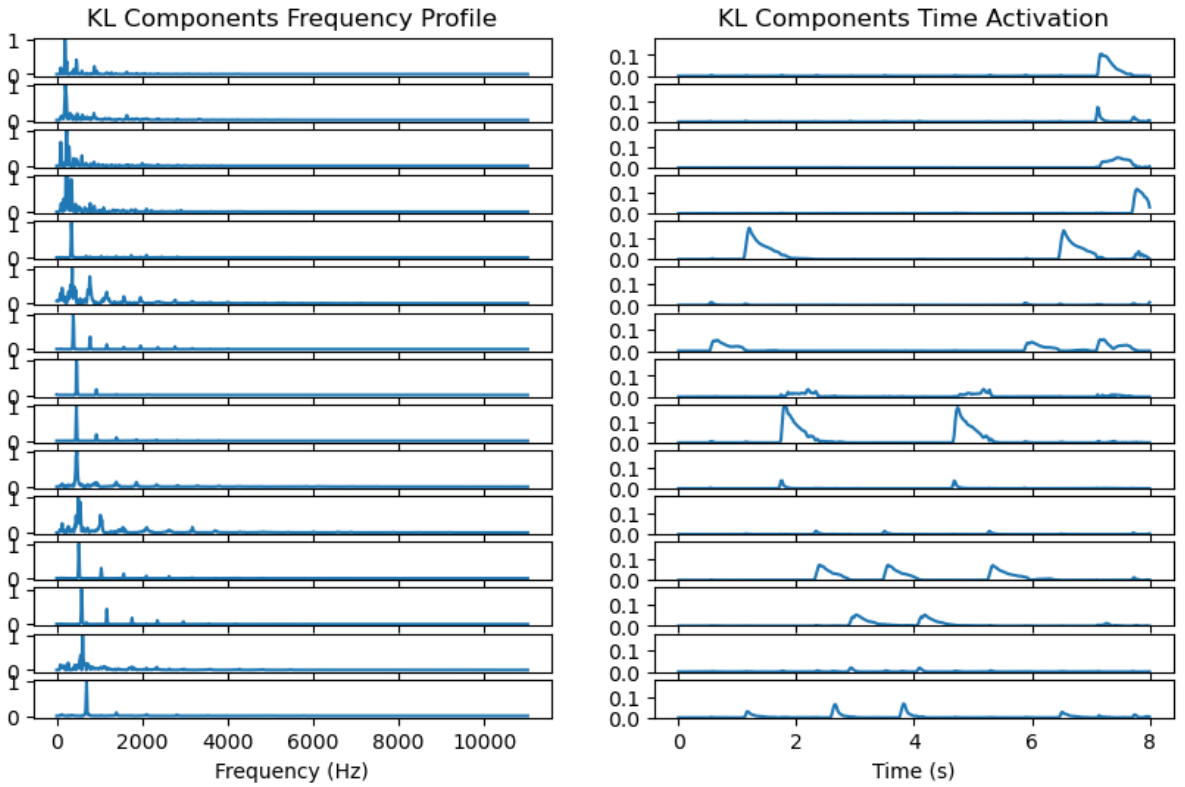
Remark

The components are ordered by increasing fundamental frequency. The notes played in the recording are, in order, G3, F3, Bb3, C4, F4, D4, C4, F4, D4, Bb3, C4, G3, F3. We can see that the components match that order.

While this is promising, we knew the correct number of notes in the recording. Also, the rank-one spectrograms are not

well located in time when analyzed on a logarithmic scale. For instance, the spectrogram of C4 has small but nonzero values when all the other notes are played.

Adding the next few seconds makes things much harder to analyse. The audio contains two chords, each with five notes, for a total of 14 distinct notes played. We see in the experiment below that NMF-KL does not identify all the notes. Instead, it introduces components to refine its approximation of the already identified notes.



This suggests a few problems with NMF:

- The rank-one hypothesis for isolated notes is not good enough. Therefore, the model tends to use several components to model a single note. This makes the choice of the rank harder, and the post-processing also more challenging.
- We have little guarantee that the output spectra are indeed interpretable as comb-like spectra with a clear fundamental frequency. We clearly see in the above example that the components with the smallest fundamental frequency are hardly interpretable. Their time activation is also far from the standard attack-hold-sustain-decay pattern. In other words, the uniqueness of NMF is unclear as soon as many notes overlap in time and frequency, which is usually the case with chords in tonal music.

Tentative improvements to blind NMF for AMT involve parameterizing the time activations to actually follow and attack-decay pattern [Cheng *et al.*, 2016]. We used a different route and utilized the so-called Convolutive NMF with supervision.

21.2 Convolutive pretrained NMF for improved piano transcriptions

In [Wu *et al.*, 2022], we proposed to address both issues: the modeling errors of spectrograms of single notes are reduced by introducing convolutive rank-one components, while the interpretability is ensured by pretraining the dictionary W on isolated notes recordings, thus switching from an unsupervised setup to a (weakly) supervised framework. Let us first discuss convolutive NMF.

21.2.1 Convolutive NMF, a better model for notes spectrograms

As mentioned in *Applications of rLRA: summary*, magnitude spectrograms of single piano notes are not exactly rank-one matrices. While it is rather accurate that a single activation vector can describe the dynamics of the note over time, the description of a single note as a fixed frequency spectrum is wrong. The spectrum near the attack contains transients that rapidly dissipate. It therefore may differ significantly from the spectrum measured when the note is sustained. Ideally, to obtain a linear model, we could split attack and decay/sustain spectra as proposed in [Cheng *et al.*, 2016]. We proposed to model each single note spectrogram $Y[f,t]$ as the convolution of a small time-frequency matrix with a time activation:

$$\hat{Y}[f,t] = \sum_{\tau=0}^{T-1} W[f,\tau]h[t-\tau], \tag{21.1}$$

where T is the size of the convolution window. This rank-one convolutive model can also be seen as a constrained rank T NMF if we consider the Toeplitz matrix $\tilde{H}$ obtained by stacking rows of h shifted by one:

$$\hat{Y} = W\tilde{H}.$$

Convolutive NMF is obtained by modeling a nonnegative matrix as the sum of rank-one convolutive terms as defined in (21.1), that now depend on the component index q. It was proposed originally in the context of speech separation [Smaragdis, 2006], and writes

$$\hat{Y} = \sum_{q=1}^{r}\sum_{\tau=0}^{T-1} W[f,\tau,q]H[q,t-\tau].$$

The figure below compares NMF and CNMF on a visual example.

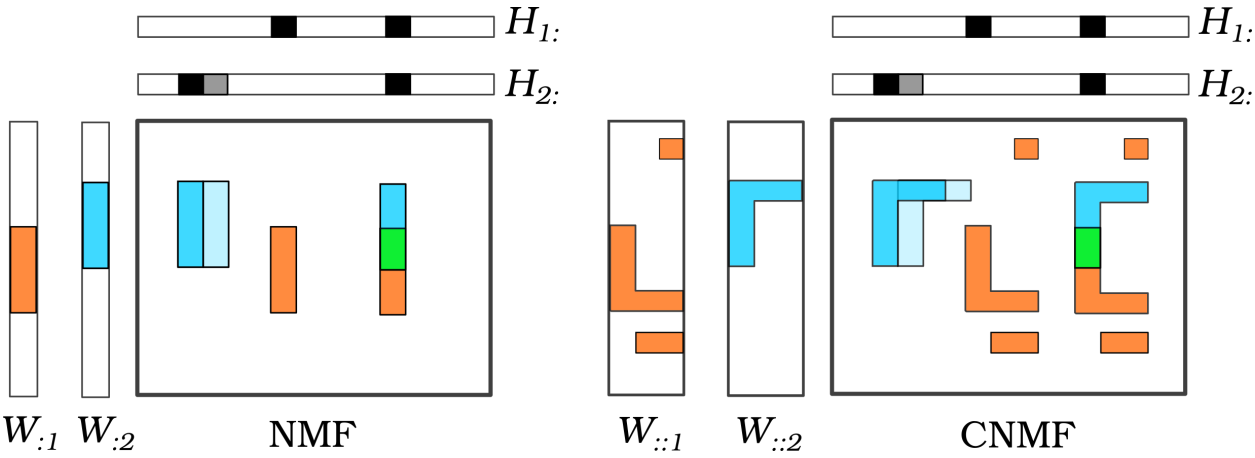


Fig. 21.2: Illustration of the CNMF model compared to NMF.

Computing CNMF is typically done in an alternating fashion. Here, we do not need the full update of the tensor W in CNMF, since we will only use *rank-one CNMF* during training. The update for H provided in the original CNMF paper is a heuristic, but it was later refined using majorization-minimization techniques [Fagot *et al.*, 2019] similar to what we

describe in *Nonnegative regressions: NNLS and NNKL*. In terms of optimization, multiplicative update rules for CNMF are obtained essentially with the same majorization techniques as introduced in *Nonnegative regressions: NNLS and NNKL*. Indeed, the MU algorithm relies on a majorization that removes the dependence of each variable on the others, and, in that respect, CNMF has the same structure as NMF. The updates for tensor $W^{(k)}$ and matrix $H^{(k)}$ and iteration k can be written in matrix format as follows (following the MM2 algorithm from [Fagot *et al.*, 2019]):

$$W^{(k+1)} = W^{(k)} * \frac{\frac{Y}{\hat{X}^{(k)}}\tilde{H}^{(k)T}}{\mathbf{1} \otimes \sum_t \tilde{H}^{(k)}[:,t]} \text{ and } H^{(k+1)}[:,t] = H^{(k)}[:,t] * \frac{W^{(k)} \times_{1,2} \frac{Y}{\hat{Y}}[:,t:t+T]}{\sum_{f,\tau} W^{(k)}[f,\tau,:]\mathbf{1}_{t+\tau \leq n}},$$

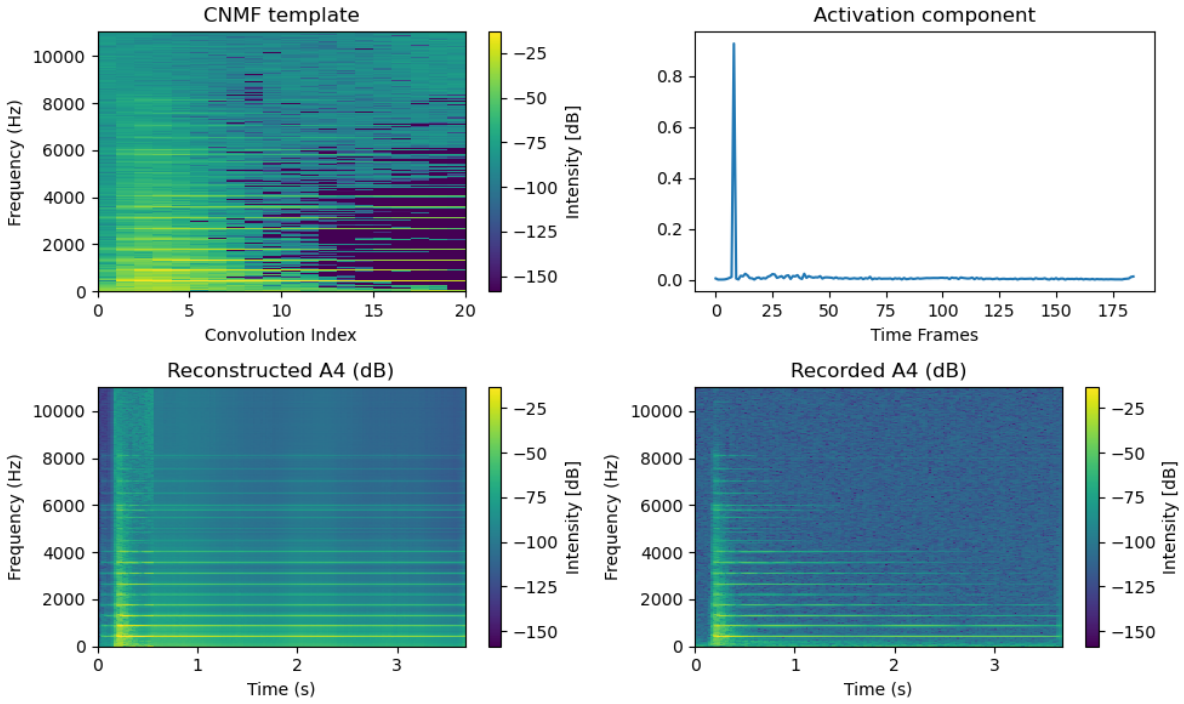
where $\hat{Y} \in \mathbb{R}^{m \times n}$ is the reconstructed dataset, matrix $\tilde{H}$ is computed from matrix H by shifting each row T times and concatenating the results, resulting in a block-Toeplitz matrix. Recall that $\times_{1,2}$ denotes the tensor contraction on modes 1 and 2, and here corresponds to the broadcasted inner product of slices of $\frac{Y}{\hat{Y}}$ with each slice of tensor W along the third mode. Zero-padding can be considered to handle the border of the input spectrogram.

Similarly to NMF, CNMF is a nonconvex problem. Personal observations lead me to believe that CNMF is significantly harder, with many non-trivial local minima, and therefore initialization is crucial.

21.2.2 A CNMF dictionary of pure notes with rank-one CNMF

Armed with the iterative algorithm we just described, we can decompose the spectrogram of a single note, say A4, with rank-one CNMF. The template tensor W, which has a single slice for rank-one CNMF and is therefore essentially a matrix, is initialized by copying the slice of the input spectrogram $Y[:, t_0 - 3 : t_0 - 3 + T]$ where t_0 is the index of the maximum column in ℓ_1 norm, and T is set to 10. This corresponds to selecting the most intense 0.2 seconds of the input spectrogram. This way, we ensure that the CNMF template is interpretable as a part of the spectrogram of A4. Random initialization may lead to a smaller loss function at convergence, but the results are harder to interpret. Separable CNMF has been studied in the literature [Degleris and Gillis, 2020].

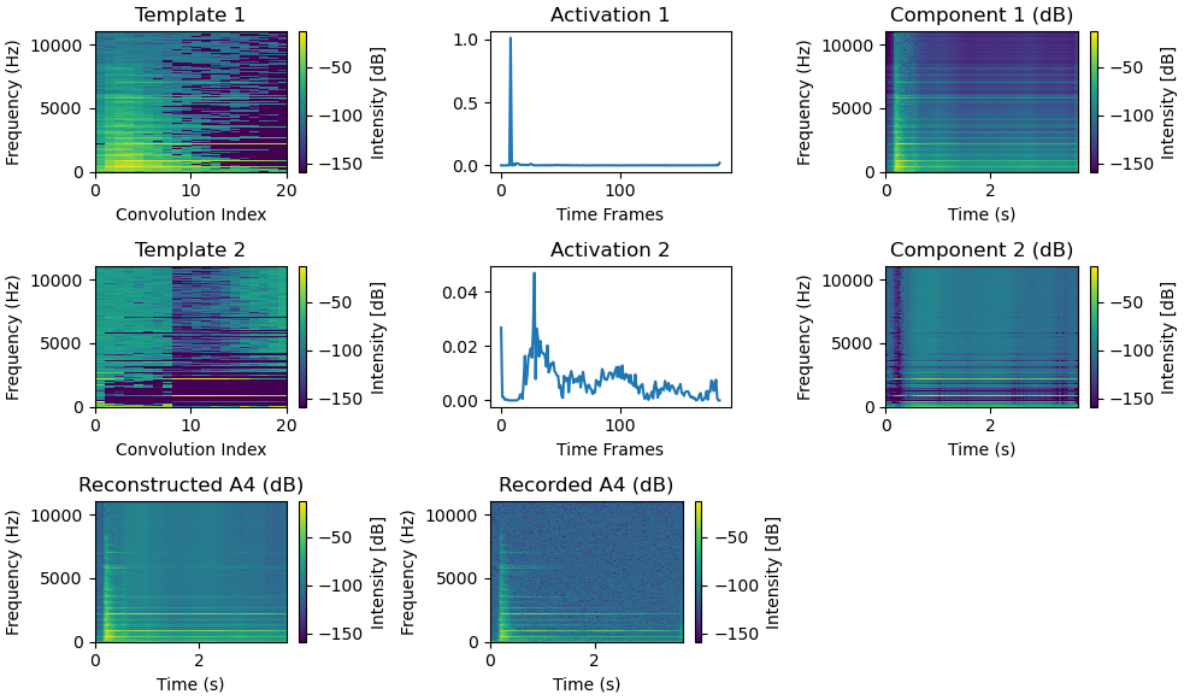
```
A4_wav, sr = sf.read('../../tensorly_hdr/dataset/MAPS_ISOL_LG_M_S0_M69_ENSTDkAm.wav')
A4_wav = A4_wav / np.max(np.abs(A4_wav)) # audio normalization
A4_wav = A4_wav[:,0] # use only one channel if stereo
# Work on the first few seconds to avoid learning the recording noise around 10s
A4_wav = A4_wav[sr//3:sr*4]
# Compute spectrogram up to 10kHz
frequencies, times, Sxx = signal.stft(A4_wav, fs=sr, nperseg=4096, nfft=4096,␣
 ↪noverlap=4096 - 882) #8192
# Cut for speed
half_freqs = len(frequencies) // 2
Sxx_dB = 20 * np.log10(np.abs(Sxx[:half_freqs,:]))  # convert to dB scale for display
Sxx_abs = np.abs(Sxx[:half_freqs,:])
# normalization
Sxx_abs = Sxx_abs / np.max(Sxx_abs)

# Testing rank-1 CNMF
T = 20
from tensorly_hdr.cnmf import convolutive_nmf
W_r1, H_r1, err_r1 = convolutive_nmf(
    Sxx_abs, rank=1, T=T, itmax=10, eps=1e-16, tol=0,
    print_it=1, n_iter_inner=10, init="separable", verbose=False
    )
W_cnmf=W_r1[:, :, 0]
h_cnmf = H_r1[0, :]
```



Observe that the time activation profile is sparser than with the best rank-one NMF approximation, which is expected since the template now contains short-time information. The template, however, has some issues. It captures correctly the attack spectrum with a comb-shaped spectrum and many harmonics, but the high-pitch harmonics should decrease in intensity. The content of the template after the attack above 4kHz is therefore not realistic.

This problem appears to be due to the rank-one model modeling echoes observed in the signal. To alleviate this issue, one can compute a rank-two CNMF with the second template initialized randomly. Observe how this second component captures noise and high-frequency decay spectra, effectively cleaning the first component. In particular, the second component does not model the attack. For simplicity and computational speed reasons, we will work with rank-one CNMF approximations of single notes, as done in regular NMF.

```
W_r2, H_r2, err_r2 = convolutive_nmf(
    Sxx_abs, rank=2, T=T, itmax=10, eps=1e-16, tol=0,
    print_it=1, n_iter_inner=10, init="separable", verbose=False
    )
```



21.2.3 Transcription example on MAPS

In this document, we simply provide a toy example of AMT with pretrained templates. Our goal is to transcribe the same audio recording as above with isolated notes and two chords. We know the notes utilized in the recording, and MAPS has all single notes recorded individually. Therefore, we can perform rank-one CNMF on each note recording, and then transcribe the recorded music.

```
import os
import re
import numpy as np
import soundfile as sf
from scipy import signal
from tensorly_hdr.cnmf import convolutive_nmf

# Directory with your .wav files
data_dir = '../../tensorly_hdr/dataset/notes_to_learn'
T = 10
rank = len(os.listdir(data_dir))  # 14

# Templates list
Wlist = []
names = []
display = ""

# Learning the templates
# perform the code below but with every file in ../../tensorly_hdr/dataset/notes_to_
↪learn
for filename in os.listdir(data_dir):
    midi_num = re.search(r'M(\d+)_', filename).group(1)
    display += f"Processing MIDI note: {midi_num} \n"
    names.append(midi_num)
```

(continues on next page)

(continued from previous page)

```
    filepath = os.path.join(data_dir, filename)
    A4_wav, sr = sf.read(filepath)
    A4_wav = A4_wav / np.max(np.abs(A4_wav)) # audio normalization
    A4_wav = A4_wav[:,0] # use only one channel if stereo
    # Work on the first few seconds to avoid learning the recording noise around 10s
    A4_wav = A4_wav[0:sr*4]

    # Compute spectrogram up to 10kHz
    frequencies, times, Sxx = signal.stft(A4_wav, fs=sr, nperseg=4096, nfft=4096,␣
↪noverlap=4096 - 882) #8192
    # Cutting last few time frames (40ms) because there are artifacts (probably from␣
↪the windowing above)
    Sxx = Sxx[:,:-2]
    times = times[:-2]
    half_freqs = len(frequencies) // 2
    Sxx_dB = 20 * np.log10(np.abs(Sxx[:half_freqs,:]))  # convert to dB scale
    Sxx_abs = np.abs(Sxx[:half_freqs,:])
    # normalization
    Sxx_abs = Sxx_abs / np.max(Sxx_abs)

    # rank-1 CNMF
    # Only init
    W_r1, H_r1, err_r1 = convolutive_nmf(Sxx_abs, rank=1, T=T, itmax=10, eps=1e-16,␣
↪tol=0, print_it=10, n_iter_inner=10, init="separable", verbose=False)
    Wlist.append(W_r1[:, :, 0])
    h_cnmf = H_r1[0, :]
    display += f"Reconstruction error: {err_r1[-1]} \n"
print(display)

# Post-process Wlist into a tensor
W = np.zeros((Wlist[0].shape[0], Wlist[0].shape[1], rank))
for r in range(rank):
    W[:, :, r] = Wlist[r]

# Permute by increasing MIDI name
perm = np.argsort(names)
names = np.sort(names)
W = W[:,:,perm]
names_music = ["G2", "A2", "F3", "G3", "A3", "Bb3", "C4", "D4", "F4", "G4", "Bb4", "C5
↪", "D5", "F5"]
```

```
Processing MIDI note: 45
Reconstruction error: 30.902170020850857
Processing MIDI note: 65
Reconstruction error: 13.281670437794645
Processing MIDI note: 57
Reconstruction error: 16.41841723313163
Processing MIDI note: 77
Reconstruction error: 12.29054938682003
Processing MIDI note: 62
Reconstruction error: 69.544849487878
Processing MIDI note: 58
Reconstruction error: 35.808382524387284
Processing MIDI note: 74
Reconstruction error: 28.388713018691245
Processing MIDI note: 70
```

(continues on next page)

(continued from previous page)

```
Reconstruction error: 10.952840654037638
Processing MIDI note: 72
Reconstruction error: 29.03727098089707
Processing MIDI note: 55
Reconstruction error: 19.88201846548771
Processing MIDI note: 43
Reconstruction error: 45.745351713545816
Processing MIDI note: 60
Reconstruction error: 21.024161144483852
Processing MIDI note: 67
Reconstruction error: 60.42459961908096
Processing MIDI note: 53
Reconstruction error: 39.660180152500374
```

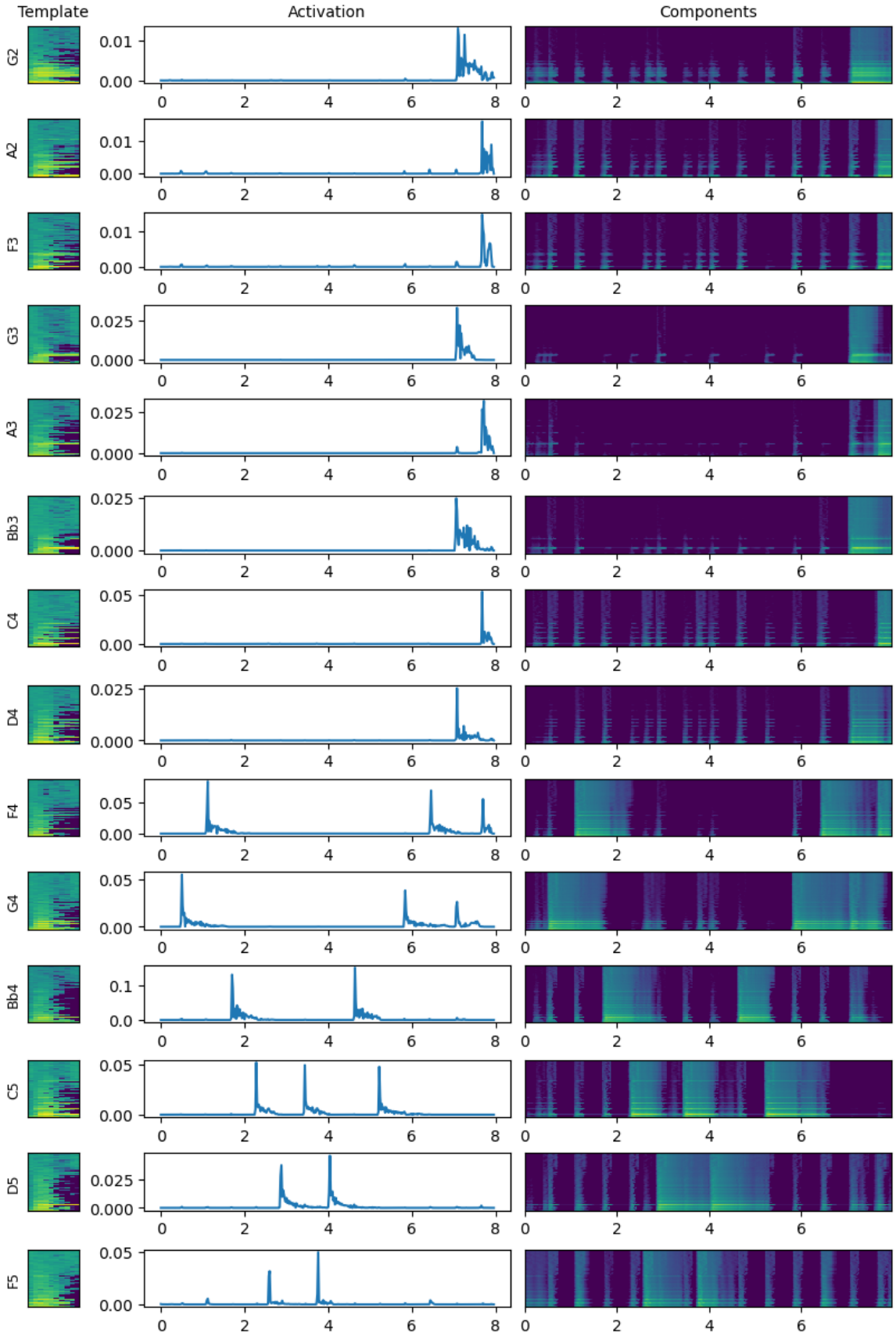
Now that the training is performed, we can transcribe the audio recording!

```
from tensorly_hdr.cnmf import convolutive_regression
H_tr, err = convolutive_regression(Sxx_abs, W, itmax=50, verbose=True, print_it=25)
```

```
Iteration initial, loss: 94752.47307634767
```

```
Iteration: 25, loss: 19.216221084630558
```

```
Iteration: 50, loss: 19.02185542594544
```


Template
Activation
Components
G2
A2
F3
G3
A3
Bb3
C4
D4
F4
G4
Bb4
C5
D5
F5

The song is now well transcribed. In particular, the last two chords have the right five notes, played simultaneously. The chord activations are still imperfect, but this is expected because of the non-linear mixing artifacts and the frequency similarity between octaves. We also see that all components slightly activate on each attack. This might be due to the hammer-action sound, which is the same across all notes and has not been taken into account separately.

The full model, weakly supervised on all piano notes, has performance close to that of fully supervised networks but is not robust to distribution shift (*e.g.*, changing the piano or the recording conditions) [Wu *et al.*, 2022]. Improving these aspects and the training procedure is part of my *research project*.

CHAPTER

TWENTYTWO

SINGLE-PIXEL HYPERSPECTRAL IMAGE RECONSTRUCTION AND UNMIXING

Reference

[Hariga *et al.*, 2024] S. Hariga, J. E. Cohen and N. Ducros, "Joint Reconstruction and Spectral Unmixing from Single-Pixel Acquisitions", EUPSICO 2024. hal

[Hariga *et al.*, 2025] S. Hariga, A. Repetti, N. Ducros, J. E. Cohen, "Approche plug-and-play pour la reconstruction des cartes d'abondance en imagerie hyperspectrale mono-pixel" GRETSI 2025 hal

[Ducros, 2024] N. Ducros, J. E. Cohen, L. Mahieu-Williame, "Freeform Hadamard imaging: Back to the roots of computational optics", under review, hal

22.1 Single-pixel hyperspectral imaging principle

Spectral cameras are invaluable tools for acquiring detailed spectral information. They function by spatially spreading an incoming light flux across its constituent wavelengths. This spectral-spatial (Fourier) transformation implies that, by design, it is difficult to acquire spectral images with high spatial and spectral resolution. This is particularly true if the imaging device must remain affordable and lightweight.

Although the idea of compressive acquisitions in computational optics precedes the development of compressive sensing [Nelson and Fredman, 1970], Duarte and co-authors proposed the single-pixel camera for Single-Pixel Imaging (SPI), which uses spatial multiplexing to feed focalized masked images into a high-resolution point spectrometer [Duarte and Baraniuk, 2012, Duarte *et al.*, 2008]. In the setup available at CREATIS, the compressed light flux is the input of a spectrometer. The hyperspectral cube containing both the spatial information (images) and the spectral information (spectra) is then reconstructed by solving an inverse problem, following a long tradition of reconstruction imaging techniques in the computational imaging community [Beneti Martins *et al.*, 2023].

Let us formalize the problem. Let $Y \in \mathbb{R}_+^{m\times p}$ be the acquired matrix, with m the number of measurements or patterns, and p the number of spectral bands measured by the spectrometer. Each acquisition, say the k-th, is obtained by summing the pixels of a masked image $a_k *_{1,2} X_t$ where $a_k \in \{0,1\}^{n_1\times n_2}$ is a binary mask and $X_t \in \mathbb{R}_+^{n_1\times n_2\times p}$ is the unknown hypespectral cube to reconstruct, with $n_1 \times n_2$ pixels. The elementwise product $*_{1,2}$ indicates that the mask is applied on each slice $X_t[:,:,i]$ of the hypercube. It can be convenient to vectorize the spatial dimension and consider the vector patterns $A[k,:] \in \{0,1\}^{n_1 n_2}$ as well as the unknown matrix $X \in \mathbb{R}_+^{n_1 n_2\times p}$. The noiseless acquisition is therefore modeled linearly as $Y = AX$.

The noise model for single-pixel imagers is standardized as a Poisson-Gaussian mixture, see the EMVA Standard 1288. For the sake of simplicity, and since Gaussian noise is typically weak in these acquisitions, we consider the noise to be pure Poisson. Poisson noise arises because the spectrometer counts photons hitting its electronic sensors, and counting errors/fluctuations are well modeled by the Poisson distribution. The model then writes

$$Y \sim \mathcal{P}(\alpha AX)$$

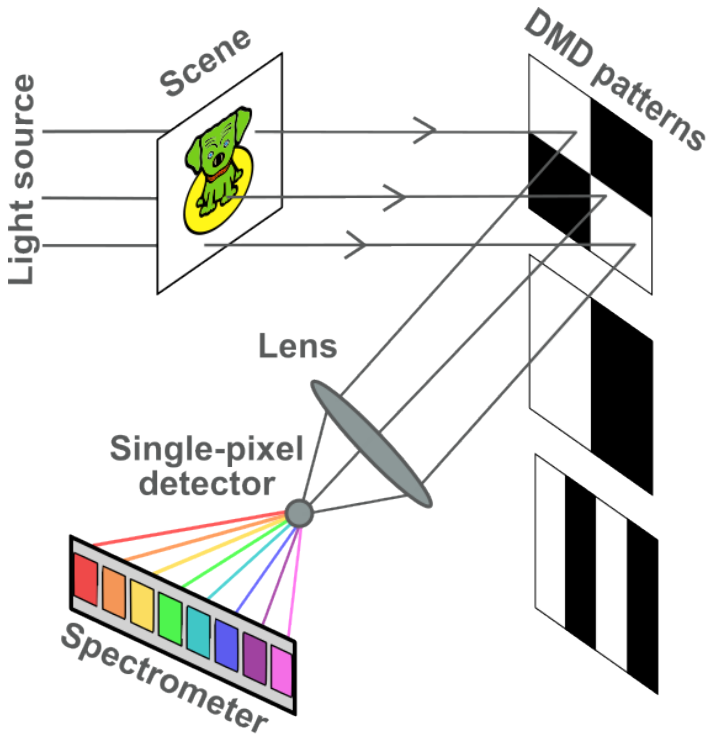


Fig. 22.1: Representation of the single-pixel camera [credits: Séréna Hariga].

where α is the average photon count that can be interpreted as the signal strength. The signal to noise ratio can be shown to be proportional to $\sqrt{\alpha}$: the higher α the less noisy the acquisition. We consider that the hypercube X has normalized intensities in $[0, 1]$.

The reconstruction of the hypercube X, knowing the measurements Y and the acquisition matrix A, is an inverse problem. It is ill-posed in the sense that infinitely many measurements are required to obtain a perfect reconstruction, because of Poisson noise. Furthermore, if the number of patterns is strictly smaller than the number of pixels $n := n_1 n_2$, the linear system $Y = AX$ has infinitely many solutions. For both these reasons, regularization is essential in single-pixel image reconstruction. The *classical approach* is to reconstruct each slice of the hypercube separately, wavelength by wavelength, using state-of-the-art image reconstruction algorithms and priors. In the PhD thesis of Serena Hariga, co-supervised with Nicolas Ducros, we have studied the joint reconstruction and unmixing of the hypercube using spectral regularization [Hariga *et al.*, 2024], as well as spectral unmixing directly from the measurements with spatial priors [Hariga *et al.*, 2025].

22.2 Reconstruction wavelength by wavelength (existing work)

22.2.1 Hadamard patterns

Before considering a baseline reconstruction method, it is necessary to discuss more about the choice of the acquisition matrix A. There are both practical and theoretical constraints that guide the choice of A.

- Practically, the patterns are applied to the image by leveraging a micromirror matrix (DMD). The image is focused on the DMD, which has small mirrors corresponding to pixels that can point in two different directions. All the mirrors that point towards the lens that targets the spectrometer reflect light from the observed object and contribute to the measured spectrum. The mirrors pointing in the opposite direction effectively block the incoming light. Therefore, we may assume that the observation matrix is binary. Contrary to other imaging modalities, such as

magnetic resonance imaging, the patterns can be changed at will for each acquisition.

- Theoretically, if the patterns could be negative in $[-1, 1]$, the "optimal" acquisition patterns are known to be Hadamard matrices [Nelson and Fredman, 1970]. Optimality here is meant in the sense of statistical minimal linear reconstruction error, *i.e.*, minimal estimation variance. In particular, Hadamard patterns are shown to outperform raster scan, which observes each pixel individually and therefore does not perform spatial multiplexing. The theoretical gain of Hadamard matrices versus raster scan is known as the Felgett advantage.

Why are Hadamard patterns "optimal"

In 1970, Nelson and Fredman introduced the concept of Hadamard spectrocopy [Nelson and Fredman, 1970]. In their seminal work, they prove that Hadamard patterns are an "optimal" choice for the acquisition basis. Let us clarify in what sense this optimality is meant, and under which assumptions.

Nelson and Fredman are concerned with linear reconstructions of acquisitions $y = Ax + e$, where matrix A is either the identity matrix (raster scan) or a complete Hadamard basis. The noise e is supposed to be additive, i.i.d. with finite variance σ, but the distribution of e is not necessarily Gaussian.

Linear reconstructions are of the form $\hat{x} = By$ with matrix B defined as the pseudo-inverse of matrix A. This choice for the reconstruction corresponds to the least squares estimation of the unknown x; it is the MLE under Gaussian noise. If the noise is not Gaussian, its statistical meaning is more ambiguous.

The quantity that Nelson and Fredman consider for measuring the quality of the reconstruction is the Root Mean Square Error (RMSE), defined as

$$\mathbb{E}\left[\|x - \hat{x}\|^2\right].$$

RMSE is exactly the average estimation error in ℓ_2 norm, and therefore is a reasonable error metric for Gaussian noise. Because matrix B is the pseudo-inverse of the acquisition operator A, we see that the residual $x - \hat{x}$ amounts to exactly Be, a centered noise with covariance $B^T B$ ($\hat{x}$ is an unbiaised estimate). The RMSE is thus easily derived as

$$\text{RMSE}(B) = \sigma\sqrt{\text{Tr}\,(BB^T)},$$

or for the i-th measurement only,

$$\text{RMSE}_i(B) = \mathbb{E}\left[\left(x[i] - \hat{x}[i]\right)^2\right] = \sigma\sqrt{\sum_j B[i,j]^2}.$$

The Felgett advantage is obtained when comparing the RMSE of Hadamard patterns and raster scan. The ratio of the RMSE_i values, denoted by F_i, is simply

$$F_i = \frac{\text{RMSE}_i(B)}{\text{RMSE}_i(I_n)} = \frac{\sigma\sqrt{\sum_j B[i,j]^2}}{\sigma} = \|B[i,:]\|_2.$$

For raster scan the rows of matrix B have unit ℓ_2 norm, while for Hadamard patterns the rows of $B = \frac{1}{n}A^T$ have constant norm, $\sqrt{n}$, which leads to

$$F_i =: F = \sqrt{n}.$$

A reasonable question is whether there exist better choices than Hadamard matrices for maximizing this ratio. Interestingly, Nelson and Fredman show that the ratio F is bounded by $\sqrt{n}$. Indeed, by the Cauchy-Schwarz inequality,

$$1 = \sum_j B[i,j]A[i,j] \leq \underbrace{\|B[i,:]\|_2}_{F_i^{-1}} \|A[i,:]\|_2.$$

Therefore, it holds that $F_i \leq \|A[i,:]\|_2 \leq \sqrt{n}$ where the last inequality assumes that the measurement operator A has all values between -1 and 1. Hadamard matrices are thus MSE-optimal.

The optimality of the Hadamard patterns, however, is meant under many unrealistic assumptions:

- The measurement operator A has values in $[-1, 1]$. The bound on the magnitude is reasonable for SPI, but a DMD can only implement nonnegative entries. Neslon and Fredman discussed this issue and introduced binary S-matrices, which have a Felgett advantage of $\frac{n+1}{2\sqrt{n}}$, therefore quasi-optimal.
- The noise is additive. In SPI, the noise model is Poisson-Gaussian, so additivity is not a valid assumption.
- The reconstruction is linear. Most state-of-the-art reconstruction algorithms employ more sophisticated reconstruction strategies.
- The error is measured with the RMSE, which is not a robust or statistically meaningful error metric for Poisson-Gaussian noise.

There are, therefore, ample research avenues for improving upon the (nonnegative) Hadamard patterns design. Revisiting the acquisition patterns, MSE gains for Poisson-Gaussian noise and statistical estimators is one of my *research perspectives*.

A practical solution for implementing the Hadamard patterns on the DMD is to acquire the positive and negative parts sequentially and concatenate the measurements into a single vector. If the image has n pixels and n Hadamard patterns to be acquired, the number of measurements is then $m := 2n - 1$ (one pattern is full of zeros and ignored), and the data vector y has size m. Doing so preserves the Poisson distribution. Another usual post-processing method computes the difference of the positive and negative acquisitions, resulting in a simulated Hadamard acquisition, but the noise distribution is modified. Formally, we define

$$A = \begin{bmatrix} [H]^+ \\ [H]^- \end{bmatrix}$$

where H here denotes a Hadamard matrix, and $[x]^- \geq 0$ is the negative part of a vector or matrix x. Matrix $[H]^-$ contains a row of zeros that is usually removed.

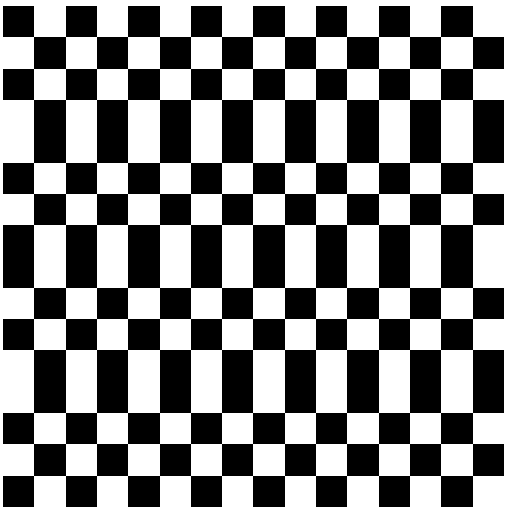


Fig. 22.2: Examples of Hadamard patterns stored in the rows of matrix H. Ones are shown in white, and minus ones (zeros in practice) in black. The patterns are orthogonal and have the same number of black and white pixels, except the first two (all white, all black).

In short, we suppose that measurements Y are stored in a matrix of size $m \times p$, and assume the model $Y \sim \mathcal{P}(\alpha AX)$ with A of size $m \times n$ a matrix with entries in $\{0, 1\}$.

Hadamard matrices have both a simple form for the pseudo-inverse and a cheap matrix-vector product [Magoarou and Gribonval, 2015]. It is natural to question whether one can apply, and invert, the split operator A at a similar cost, and we show here that both operations can be performed efficiently when A is the split Hadamard matrix.

First, the product $A^T y$ can be obtained using the fast Hadamard transform. Indeed, $[H]^{\pm} = \frac{1}{2}[H \pm 1]$. Second, the pseudo-inverse $A^\dagger$ admits a closed-form expression, see section 3.5.4 in [Harwit and Sloane, 1979].

$$A^\dagger = \frac{2}{n}\left(I_n - \frac{1}{n+1}1_{n\times n}\right)A^T$$

where $1_{n,n}$ is a matrix of ones, and its contraction is implemented as a summation. This is obtained by noting that

$$A^T A = \frac{n}{2}\left(I_n + 1_{n\times n}\right)$$

and that $\left(I_n - \frac{1}{n+1}1_{n\times n}\right)\left(I_n + 1_{n\times n}\right) = I_n$.

Inversion of the positive Hadamard matrix

Most matrices obtained from the Hadamard matrix admit fast inversion algorithms. The methodology, exemplified on matrix $[H]^+$, is as follows.

Matrix $[H]^+$ is left-invertible, and by noticing that

$$\frac{1}{n}H^T[H]^+ = \frac{1}{2}I + \frac{1}{2}e_1 1^T,\ e_1 = [1, 0, \dots, 0]^T$$

we get that for any vector $x \in \mathbb{R}^n$,

$$\frac{1}{n}H^T[H]^+ x = \left[x[1] + \sum_{j>1}\frac{x[j]}{2}, \frac{x[2]}{2}, \dots, \frac{x[n]}{2}\right]^T,$$

and vector x is recovered as

$$x = \frac{1}{n}\left(2I_n - e_1 1^T\right)H^T[H]^+ x.$$

We see that the pseudo-inverse of $[H]^+$ is computed as

$$\frac{1}{n}\left(2I_n - e_1 1^T\right)H^T = \frac{1}{n}\left(2H^T - e_1 e_1^T\right)$$

which can be efficiently contracted with any vector x given a fast Hadamard transform algorithm.

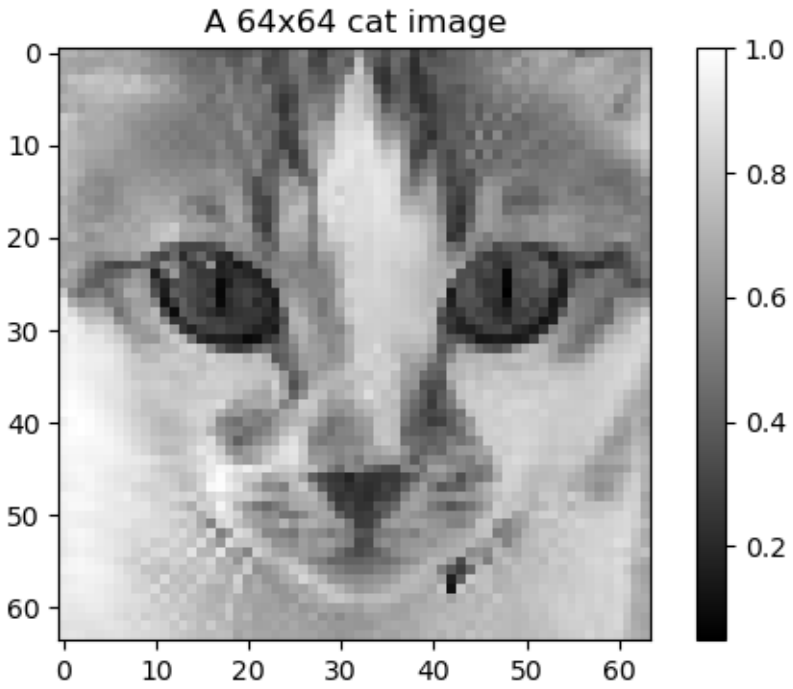
We can easily simulate the acquisition with the help of the Python package spyrit, developed at CREATIS [Abascal *et al.*, 2025].

```
import torch, torchvision
from spyrit.core.meas import HadamSplit2d
from spyrit.core.noise import Poisson
import matplotlib.pyplot as plt
import tensorly as tl
tl.set_backend("pytorch")

# Cat obtained from https://www.kaggle.com/datasets/mahmudulhaqueshawon/cat-image␣
↪(small)
x = torch.sum(torchvision.io.read_image("../../tensorly_hdr/dataset/cat.jpg"), axis=0)
```

(continues on next page)

(continued from previous page)

```
x = x/torch.max(x)  # normalize to [0,1], this is realistic

plt.figure(figsize=(6,4))
plt.imshow(x, cmap='gray')
plt.colorbar()
plt.title("A 64x64 cat image")
plt.show()
```



We define the measurement operator and the noise model, then use fast transforms to efficiently simulate the acquisition. Note that while the mathematical discussion in this section is focused on 1D vectors x, SPI involves 2D monochromatic images (or 3D hyperspectral cubes when spectra are considered). The Hadamard and split Hadamard operators can be defined in 2D using separability.

We can also visualize the measurements as coefficients in the Hadamard basis, which can be seen as a wavelet transform. This is more meaningful here to display the coefficients corresponding to the split acquisitions, $[H]^+$ and $[H]^-$.

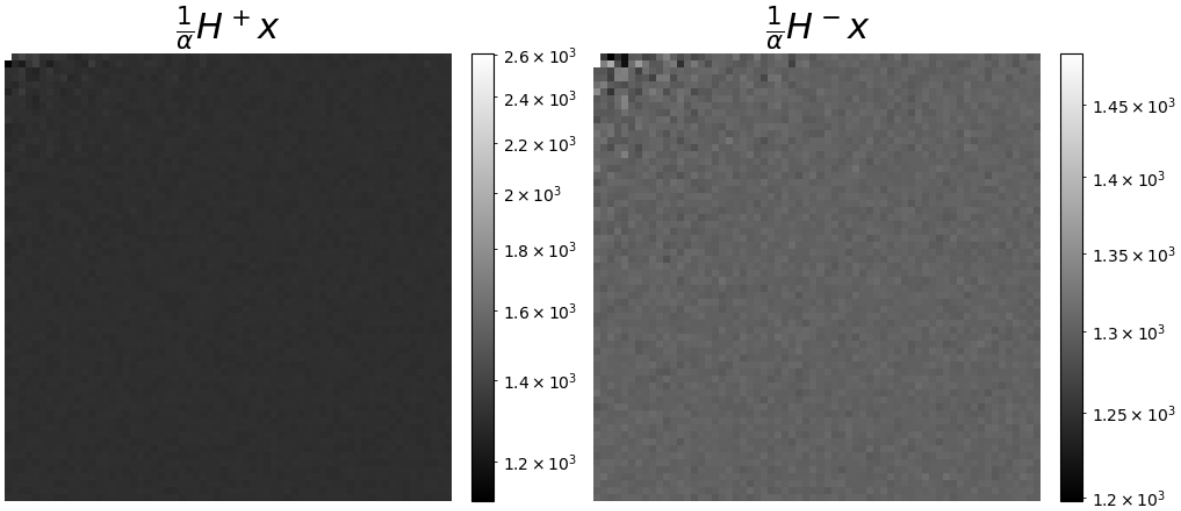
```
from matplotlib.colors import LogNorm

alpha=1e2
meas_op = HadamSplit2d(64, noise_model=Poisson(alpha))  # this creates the split␣
↪measurement operator A, and defines the model parameters (noise level)
y = meas_op.forward(x)  # This sampled y~P(\alpha Ax)
y = y/alpha  # rescale to get back the right intensity scale
print(f"We can check that the shape of y, {y.shape[0]}, is 64^2*2=8192, as expected␣
↪for a split Hadamard of size 64x64, where the row of zero in A is not removed")

y_pos = y[0::2]
y_neg = y[1::2]  # Note: y[1] is zero

f, axs = plt.subplots(1, 2, figsize=(12, 5))
# increase fontsize of the title
```

(continues on next page)

(continued from previous page)

```
axs[0].set_title(r"$\frac{1}{\alpha}H^+x$", fontsize=22)
# In logscale to better see small values
im = axs[0].imshow(y_pos.reshape(64, 64), cmap="gray", norm=LogNorm())
# add a colorbar, that fits the size of the image, and with fixed range
plt.colorbar(im, ax=axs[0], fraction=0.046, pad=0.04)
axs[1].set_title(r"$\frac{1}{\alpha}H^-x$", fontsize=22)
im = axs[1].imshow(y_neg.reshape(64, 64), cmap="gray", norm=LogNorm())
plt.colorbar(im, ax=axs[1], fraction=0.046, pad=0.04)
# remove axis for the both plots
axs[0].axis('off')
axs[1].axis('off')
plt.show()
```

```
We can check that the shape of y, 8192, is 64^2*2=8192, as expected for a split␣
↪Hadamard of size 64x64, where the row of zero in A is not removed
```



22.2.2 Reconstruction

As long as the spectral dimension is ignored and reconstruction, that is, recovering the true image x from the measurements y, is computed wavelength by wavelength, there are a variety of well-known available methods that are listed thereafter.

Pseudo-inverse reconstruction

The simplest reconstruction algorithm performs a least squares reconstruction,

$$\hat{X} = \frac{1}{\alpha} A^{\dagger} Y = \underset{X \in \mathbb{R}^{n \times p}}{\operatorname{argmin}} \| \frac{1}{\alpha} Y - AX \|_F^2.$$

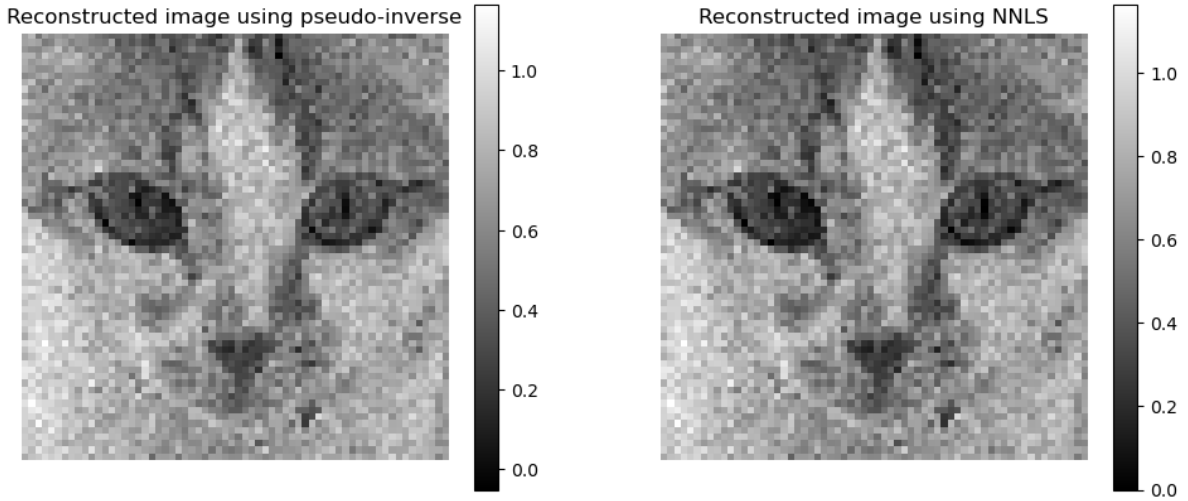
This reconstruction has several advantages.

- It is unbiased.
- It is reasonably cheap to perform as long as the observation matrix A is not too large. Recall that the pseudo-inverse of matrix A is known in closed form, and its contraction with a vector or matrix leverages fast Hadamard transforms.

In practice, the spyrit package that we rely on does not yet implement these fast transforms for the split Hadamard measurements without further processing, so the pseudo-inverse computation is quite slow. The solvers I use are not compatible with fast Hadamard transforms and are also quite slow. Since the image is nonnegative, we can also use a NNLS solver such as HALS.

```
# Reconstruction with pseudo-inverse
x_rec = torch.linalg.lstsq(meas_op.A, y).solution.reshape(64, 64)
print(f"The pseudo-inverse reconstruction has {torch.sum(x_rec<0)} negative entries")
# Bonus: we can use a nnls solver, also very slow without fast operators
from tensorly.solvers.nnls import hals_nnls
Aty = meas_op.adjoint(y)[:,None]
AtA = meas_op.A.T@meas_op.A
x_rec_nnls = hals_nnls(Aty, AtA, n_iter_max=20, tol=0).reshape(64, 64)
```

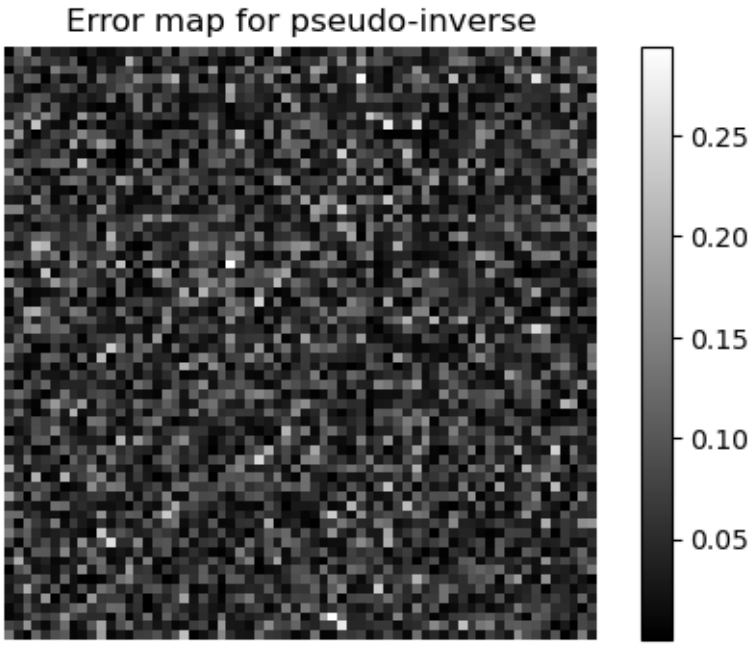
```
The pseudo-inverse reconstruction has 6 negative entries
```



Note that the pseudo-inverse reconstruction gives essentially the same result as the NNLS solver. The two reconstructions are not the same, however, if we did not have to split the acquisitions and could acquire Hx directly, then they would be the same. Indeed, H is an invertible matrix and therefore $x^* = H^T y$ minimizes the loss $\|y - Hx\|_2^2$. The same solution would also minimize any positive loss $g(x)$ equal to zero at x^*, which includes the MLE discussed next. This observation suggests that the choice of reconstruction method may not significantly affect the results, and that the pseudo-inverse is a good compromise among speed, simplicity, and performance.

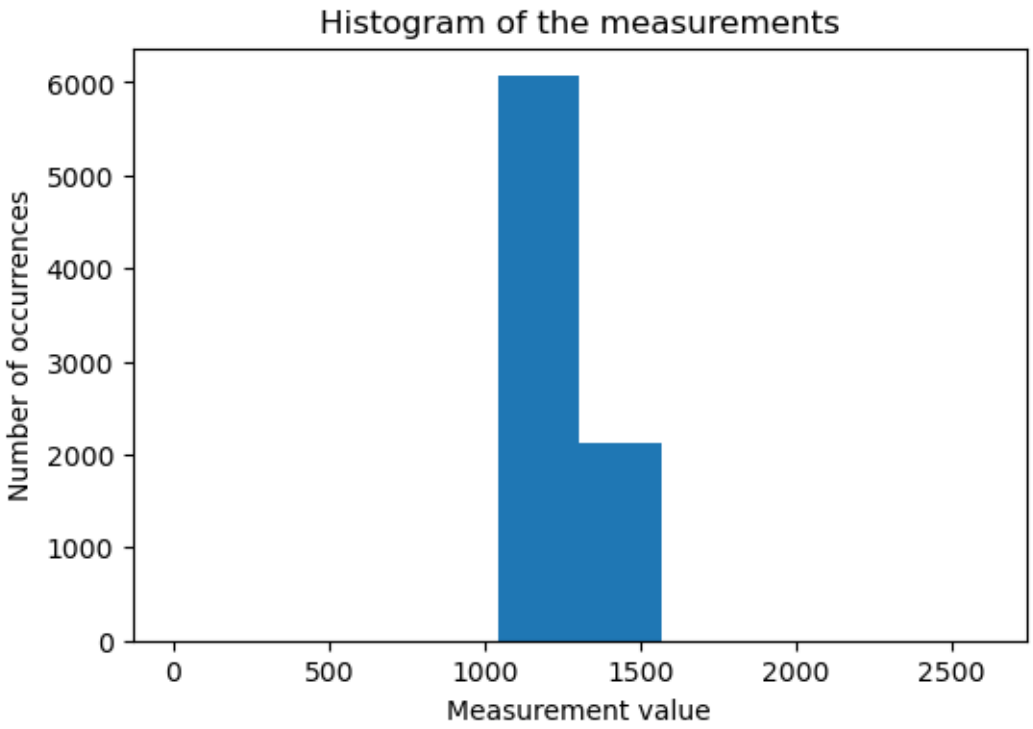
We can also show the error map for the pseudo-inverse reconstruction, to observe that the residual is not evidently spatially correlated, despite the misfit between the noise model and the reconstruction loss.

```
plt.figure(figsize=(6,4))
plt.imshow(torch.abs(x - x_rec), cmap='gray')
plt.colorbar()
plt.axis('off')
plt.title("Error map for pseudo-inverse")
plt.show()
```



The good performance of the least squares estimate for this Poisson noise problem can be explained by the fact that entries of αAx are large and concentrated around the same value $\frac{1}{2}\alpha \sum_{i \leq n} x[i]$. Indeed, except for the first row of A and the row of zeros, all rows have exactly half their values set to 1, and the other half to zero. The measurements are therefore essentially bootstraped mean evaluations of the image. This has two consequences:

- The empirical measurements y_i are large enough to obtain a faithful approximation of the (elementwise) Poisson distributions by Gaussian distributions of mean $\alpha A[i,:]x$ and variance $\alpha A[i,:]x$.
- The variance is essentially the same for all measurements, except $y[0]$ (corresponding to the row of ones) and $y[1]$ (corresponding to the row of zeros, and that could be ignored). We denote $\sigma = \frac{1}{2}\alpha \sum_{i \leq n} x[i]$ the empirical variance estimation. If we consider the model $y \sim \mathcal{N}(\alpha Ax, \sigma)$, which according to the previous point, should correctly approximate the Poisson distribution, then the pseudo-inverse is a reasonable approximation of the MLE of the image. For Gaussian problems, it is known that the residuals are uncorrelated with the image, this can be observed numerically on the previous example. Below we plot the histogram of the values in the measurement y, to check that they are indeed concentrated around the same numerical value.



Maximum Likelihood reconstruction

The MLE for Poisson noise is given by the minimizer of the KL-divergence

$$\hat{X} = \underset{X \geq 0}{\operatorname{argmin}} \, \mathcal{D}_{\mathrm{KL}} \left(\frac{1}{\alpha} Y, AX \right),$$

which can be approximately computed using, *e.g.*, the *MU algorithm*. Below is an example of MLE reconstruction.

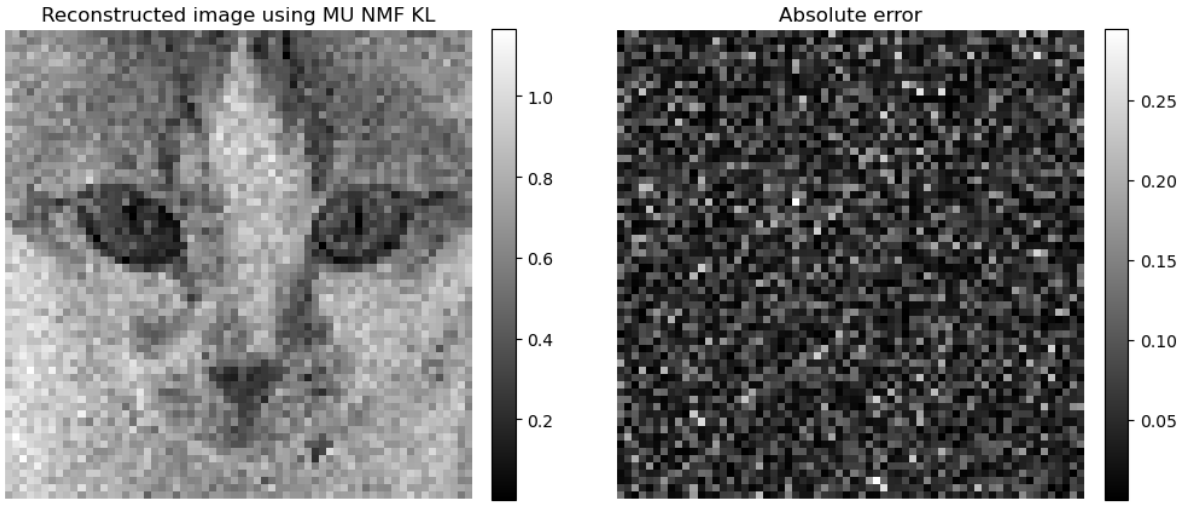
```
# Reco avec MU
from tensorly_hdr.nmf_kl import Lee_Seung_KL_regression
# remove the second element in y and the row of zeros to avoid division by zero
yr = torch.cat([y[0:1], y[2:]])
Ar = torch.cat([meas_op.A[0:1,:], meas_op.A[2:,:]], dim=0)
crit, x_mu, time, _ = Lee_Seung_KL_regression(yr, Ar, Hini=x_rec_nnls.reshape(64**2),␣
↪epsilon=1e-8, NbIter=60, tol=1e-6, verbose=True, print_it=20)
```

```
------Lee_Sung_KL running------
Loss at initialization: 20.84039878845215
Loss at iteration 1: 20.840301513671875
```

```
Loss at iteration 21: 20.84000587463379
```

```
Loss at iteration 41: 20.839679718017578
```

```
Loss at iteration 60: 20.839391708374023
```



Arguably, in this context, the complication induced by using the Poisson MLE instead of the pseudo-inverse may not be worth the gain in estimation performance. However, the MLE can be easily generalized to measurements that are not Hadamard-based, such as a raster scan that acquires each pixel sequentially. We therefore work with this estimator in the following, with least squares initialization.

Both the least squares and the MLE reconstruction are computed in parallel across wavelengths. Therefore, it is expected that incoherent reconstruction artifacts will be found in the reconstructed spectra. The goal of the works presented subsequently is to use known spectra or spectral priors to improve spectral reconstruction.

22.2.3 Undersampled reconstruction

The acquisition of the data matrix Y is performed in parallel along the wavelength dimension, but sequentially along the spatial dimension: Hadamard patterns are loaded onto the DMD one after the other. Each acquisition lasts for a predetermined time Δt; the higher Δt, the more signal is acquired. Therefore, to reduce the total acquisition time, one may either reduce Δt, which increases the noise level for all acquisitions, or acquire only a subset of the Hadamard patterns. This second choice allows for the acquisition of the most important patterns with a good signal-to-noise ratio. However, the reconstruction using the left pseudo-inverse is no longer available.

Remark

It is possible to interpret the Hadamard transform as a binary equivalent of the Fourier transform and to associate with each pattern a notion of spatial frequency. In this context, the low-frequency patterns are typically crucial for reconstructing the acquired image, while the high-frequency coefficients can help reconstruct finer details but are also more susceptible to noise.

One may use the right pseudo-inverse instead of the left one, which is known to amount to solving a problem of the form

$$\underset{Y=AX}{\operatorname{argmin}} \|X\|_F^2.$$

Prior information about the image is typically incorporated into the model to ensure identifiability. Classical priors include nonnegativity, sparsity in wavelet bases and sparsity in finite differences, also called Total Variation.

Denoting $g(X)$ the negative log-prior, reconstruction in the Maximum *A Posteriori* (MAP) sense is then performed by minimizing

$$\mathcal{D}_{\mathrm{KL}}\left(Y, AX\right) + \lambda g(X),$$

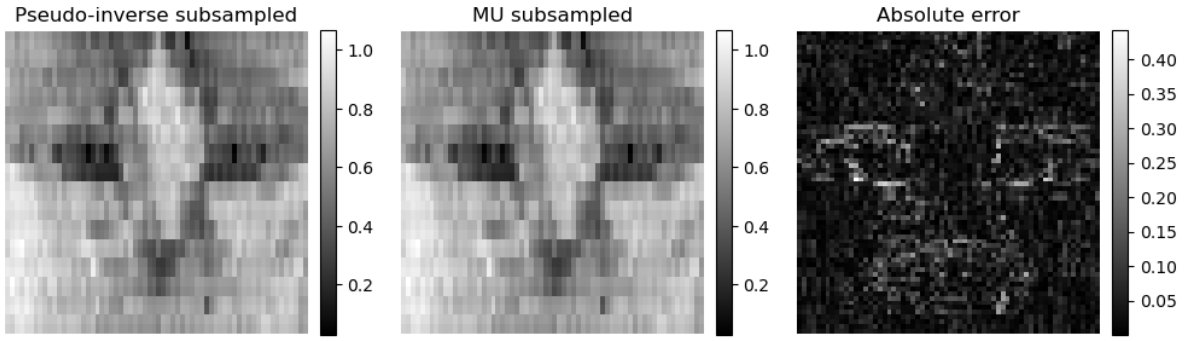
while the regularized least squares estimator is obtained by minimizing

$$\|Y - AX\|_F^2 + \lambda g(X).$$

We can illustrate the reconstruction by right pseudo-inverse and the effect of the nonnegativity prior with an NNLS solver, in which case $g(X)$ is the characteristic function of the nonnegative orthant.

```
# NNLS with subsampling
yr = torch.cat([y[0:1], y[2:]])
Ar = torch.cat([meas_op.A[0:1,:], meas_op.A[2:,:]], dim=0)
# Removing the second half of measurements
yr = yr[:len(Aty)//2]
Ar = Ar[:len(AtA)//2, :]
# Right pseudo-inverse
x_rec_ls_sub = torch.linalg.pinv(Ar)@yr
x_rec_ls_sub = x_rec_ls_sub.reshape(64,64)
# MU initialized with the pseudo-inverse
crit, x_mu_sub, time, _ = Lee_Seung_KL_regression(yr, Ar, Hini=torch.abs(x_rec_ls_sub.
↪reshape(64**2)), epsilon=1e-8, NbIter=40, tol=1e-6, verbose=True, print_it=20)
x_mu_sub = x_mu_sub.reshape(64,64)
```

```
------Lee_Sung_KL running------
Loss at initialization: 4.685630798339844
Loss at iteration 1: 4.685628890991211
Loss at iteration 21: 4.68560791015625
Loss at iteration 40: 4.685576438903809
```



Again, in this setup, nonnegativity alone is essentially useless, since the subsampled pseudo-inverse reconstruction is already nonnegative. This means that the reconstruction error for both the regularized least squares and the MAP is null at $\hat{X} = A^\dagger y$; both reconstructions may provide the exact same estimate. Moreover, we see on the error map that the original image is poorly estimated around the edges. This observation is consistent with the absence of high-frequency content. The image details cannot be reconstructed solely from the measurements.

22.3 Unmixing and reconstruction with known spectra and nonsmooth priors

22.3.1 Joint reconstruction and unmixing principle

As discussed in the introduction of this section, when considering hyperspectral images, it is reasonable to assume that the spectrum at each pixel is a linear combination of so-called endmembers, which are the spectra of pure materials present

in the scene. When there are r different endmembers and r is smaller than the number of pixels n and the number of spectral bands p, a simple but powerful model to represent the hyperspectral image is NMF,

$$X = UV^T,$$

where the nonnegative matrix $V \in \mathbb{R}_+^{p\times r}$ contains the endmembers columnwise, and the nonnegative matrix $U \in \mathbb{R}_+^{n\times r}$ contains the abundance maps, that is, the proportion of each material at each pixel. Because the illumination of the scene is not homogeneous spatially, and spectra may vary depending on other physical parameters such as the geometry of the scene, we refrain from assuming that the abundances sum to one pixelwise, but we assume that they are bounded by one elementwise.

Remark

The notations U and V are used to avoid confusion with the Hadamard matrix H and the observation matrix A.

In preliminary works, we have assumed that the endmembers V are known. The acquisition model then becomes

$$Y \sim \mathcal{P}\left(\alpha AUV^T\right), \tag{22.1}$$

and our goal is to design a state-of-the-art reconstruction algorithm that estimates the abundances U, therefore performing jointly the reconstruction of the hypercube and its spectral unmixing.

22.3.2 Joint reconstruction and unmixing in one step

In [Hariga *et al.*, 2024], a simple strategy to jointly reconstruct and unmix a single-pixel image was proposed, computing the MLE from Equation (22.1) with an alternating optimization algorithm, in this case alternating MU. The update rule for the spectra V is the same as detailled in *Nonnegative regressions: NNLS and NNKL*, and for U it can be obtained simply by 1) flattening the variables, 2) computing the classical MU rule and 3) folding the output into a matrix (see also [Févotte and Idier, 2011]). This yields after simplification

$$U^{(k+1)} = U^{(k)} * \frac{A^T \frac{Y}{AUV^T} V}{A^T 1_{m,p} V}.$$

The code for the algorithm is slightly too complex to show here and is available in the `tensorly_hdr` package. A ℓ_1 regularization is added on both factors, which adds a constant term in the denominator of the update and helps control the dynamics of the factors, see *Implicit regularization in rLRA*.

Since the optimization problem

$$\underset{U\geq 0, V\geq 0}{\operatorname{argmin}} \; \mathcal{D}_{\mathrm{KL}}\left(Y, AUV^T\right) + \lambda\left(\|U\|_1 + \|V\|_1\right)$$

is jointly nonconvex, initialization is important. A viable optimization for simple unmixing problems is to first perform the reconstruction, for instance, solving a NNLS problem, then use a pure-pixel selection algorithm such as SNPA [Gillis, 2014] to initialize V.

This procedure is illustrated below on, first, a synthetic exemple, and then on a real dataset acquired in CREATIS.

Synthetic experiment

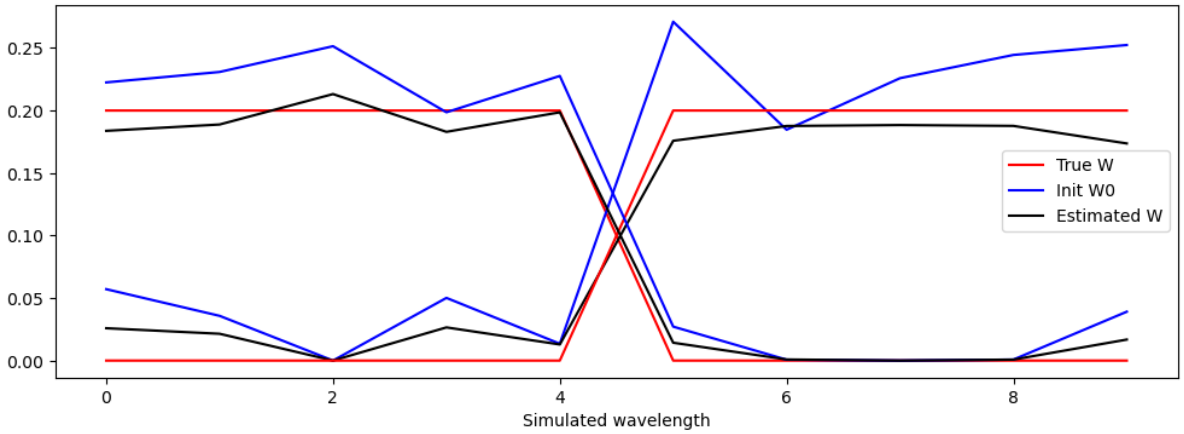
A dataset is generated in which two spectra have disjoint supports and lie on the boundary of the nonnegative orthant. This way, the NMF decomposition may be unique. Moreover, the abundances are designed to include pixels that are almost pure. No noise is added. We can observe that the reconstruction is good, but not perfect; this is mostly due to the non-uniqueness of the NMF model (which would require sparse abundances).

```
# Separation on synthetic dataset (Eusipco + HCERES)
import numpy as np
from tensorly_hdr.nmf_kl import MU_SinglePixel
from tensorly_hdr.sep_nmf import snpa

W = torch.tensor([[1,1,1,1,1,0,0,0,0,0],[0,0,0,0,0,1,1,1,1,1]], dtype=torch.float32).T
W = W/torch.sum(W, dim=0, keepdim=True)  # normalize columns
n = 32
At = torch.rand(2, n**2)
At = At/tl.sum(At,axis=0)
At[0,0]=0.99
At[1,1]=0.99
At[0,1]=0.01
At[1,0]=0.01
# Measurement operator is Ar, noiseless
Xtrue = W@At
alpha=1e2
meas_op = HadamSplit2d(n, noise_model=Poisson(alpha))  # this creates the split␣
 ↪measurement operator A, and defines the model parameters (noise level)
Ar = torch.cat([meas_op.A[0:1,:], meas_op.A[2:,:]], dim=0) # TODO: adapt to use␣
 ↪forward with wavelengths
Y = torch.poisson(alpha*Xtrue@Ar.T)
#Y = meas_op.forward(Xtrue)  # This sampled y~P(\alpha Ax)
Y = Y/alpha  # rescale to get back the right intensity scale

# Init pinv+spa
#X_rec = torch.linalg.lstsq(Ar, Y.T).solution.T
X_rec = hals_nnls(Ar.T@Y.T, Ar.T@Ar, n_iter_max=10).T
_, W0, A0 = snpa(X_rec, 2)

# Reconstruction with NMF
W_est, A_est, crit = MU_SinglePixel(Y, Ar, tl.abs(A0), tl.abs(W0), lmbd=0, maxA=None,␣
 ↪niter=300, n_iter_inner=20, eps=1e-8, verbose=True, print_it=100)
# Normalization of W_est and A_est
sum_W_est = torch.sum(W_est, dim=0, keepdim=True)
W_est = W_est/sum_W_est
A_est = A_est*sum_W_est.T
```

```
Iteration 0, Cost: 97.12303161621094
```

```
Iteration 100, Cost: 95.68405151367188
```

```
Iteration 200, Cost: 95.01849365234375
```



Real data reconstruction and unmixing

An acquisition was performed in the laboratory, with a cat image layered with two color filters, red and green. There is a thin spatial overlap between the two filters, which induces subtractive mixing; therefore, the hyperspectral image will be rank-two plus nonlinear mixing terms plus noise. However, the abundance maps are mostly disjoint, and the spectral bands where the green and red filters are intense also mostly do not overlap. Therefore, the separation problem is rather simple and amenable to separable NMF algorithms.

We first load the data and the metadata and process them to permute the acquisitions correctly. Then we remove the dark current, which is essentially the mean of the Gaussian noise in the Poisson-Gaussian mixture. We chose to ignore the Gaussian noise and assume that the only source of measurement variation is the Poisson distribution. However, Gaussian noise is not zero-centered, and the bias must be corrected. It can be estimated from the measurement acquired with the first row of the negative patterns H^-, filled with zeros. The data is then clipped and normalized to lie in the interval $[0, 1]$.

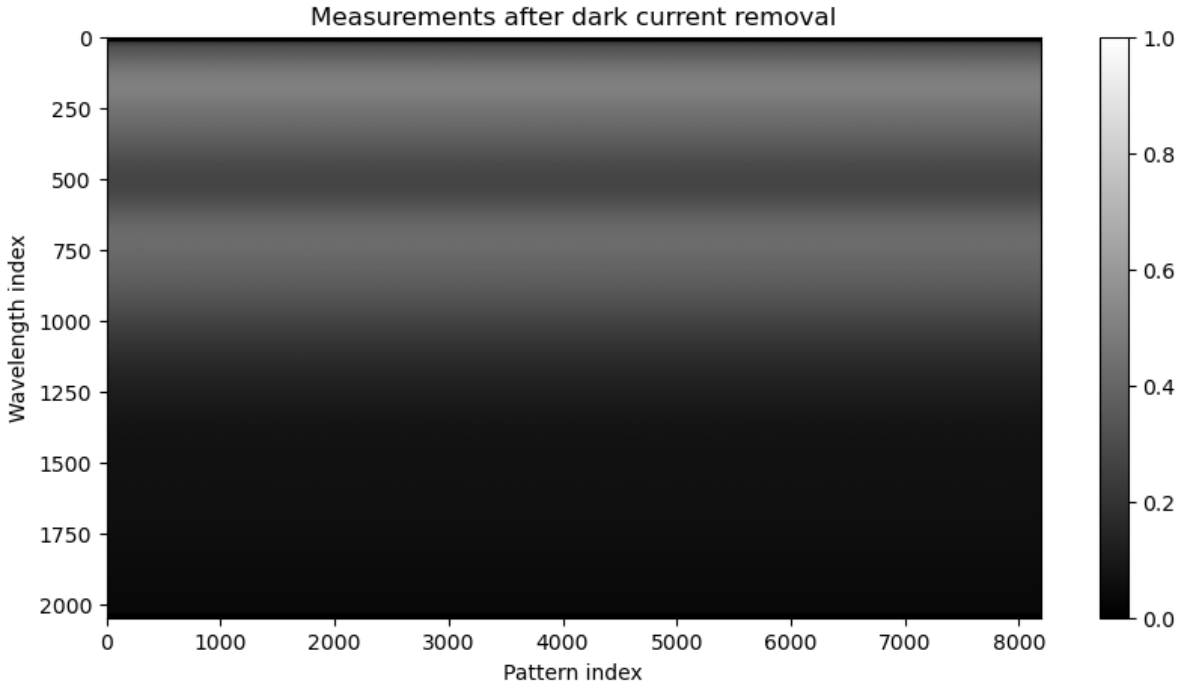
```
# Loading
import json
import ast

# Fetching the spectral measurements
data = np.load("../../tensorly_hdr/dataset/obj_Cat_bicolor_thin_overlap_source_white_
 ↪LED_Walsh_im_64x64_ti_9ms_zoom_x1_spectraldata.npz", allow_pickle=True)
Ymeas = data["spectral_data"]

# Fetching metadata
file = open("../../tensorly_hdr/dataset/obj_Cat_bicolor_thin_overlap_source_white_LED_
 ↪Walsh_im_64x64_ti_9ms_zoom_x1_metadata.json", "r")
json_metadata = json.load(file)[4]
file.close()

# replace "np.int32(" with an empty string and ")" with an empty string
tmp = json_metadata["patterns"]
tmp = tmp.replace("np.int32(", "").replace(")", "")
patterns = ast.literal_eval(tmp)  # the list (of list of) of pattern indices␣
 ↪(evaluation because stored as text)
wavelengths = ast.literal_eval(json_metadata["wavelengths"])

# Permutation of measurements, that are acquired in an experiment-specific order
from spyrit.misc import sampling as samp
img_size = 64
acq_size = img_size
```

(continues on next page)

(continued from previous page)

```
Ord_acq = (-np.array(patterns)[::2] // 2).reshape((acq_size, acq_size))
Ord_rec = torch.ones(img_size, img_size)
Perm_rec = samp.Permutation_Matrix(Ord_rec)
Perm_acq = samp.Permutation_Matrix(Ord_acq).T
Ymeas = samp.reorder(Ymeas, Perm_acq, Perm_rec)


# Post-processing of the measurements
Y = torch.tensor(Ymeas, dtype=torch.float32).T
del Ymeas

# Unbiased by removing the dark current, estimated as the minimum value of marginals␣
 ↪of Y (better?)
dc = tl.sum(Y[:,1])/Y.shape[0] # average of dc over all wavelengths
print(f"Estimated dark current to remove: {dc}")
Y = Y - dc
Y = tl.clip(Y, 0, tl.max(Y))
# Normalization to [0,1]
Y = Y / tl.max(Y)

# Showing the measurements after dark current removal
plt.figure(figsize=(10,5))
plt.imshow(Y.cpu().numpy(), aspect='auto', cmap='gray')
plt.title("Measurements after dark current removal")
plt.xlabel("Pattern index")
plt.ylabel("Wavelength index")
plt.colorbar()
plt.show()
```

```
Estimated dark current to remove: 688.939453125
```



We then follow the reconstruction procedure described earlier. We remove some elements from the data matrix: the measurement with the row of zeros, which, apart from the dark current estimation, is uninformative, and some spectral bands that acquired no signal; here, the first 16 bands. The remaining spectral bands are binned into groups of size eight to reduce the dimension of the problem due to runtime issues. We first show here the pseudo-inverse reconstruction.

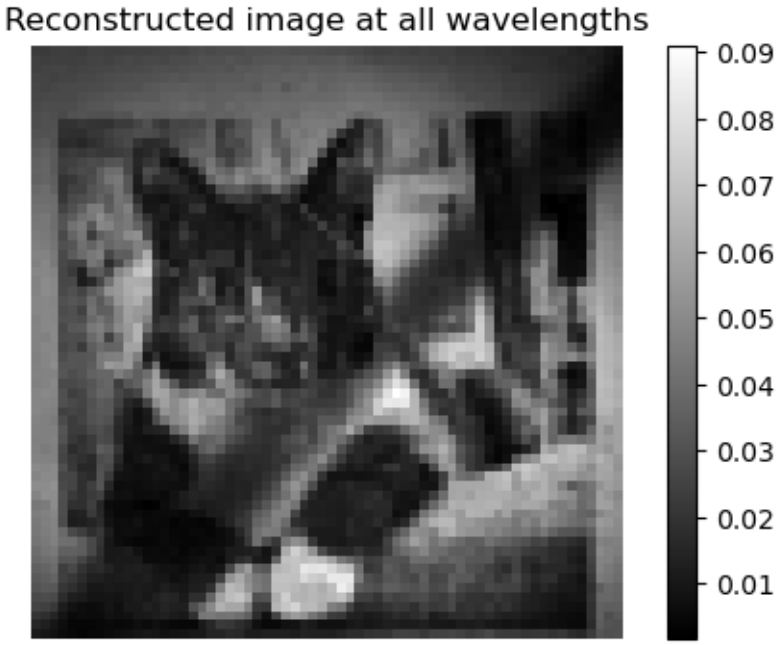
```
# Reconstruction with pseudo-inverse
from spyrit.core.meas import HadamSplit2d
import spyrit.misc.sampling as samp

# Patterns are acquired in order Acq, compared to order nat
# Patterns are stored in order Rec in spyrit
acq_size = img_size
meas_op = HadamSplit2d(img_size)
A = meas_op.A  # noiseless operator
Anz = torch.cat([A[0:1,:], A[2:,:]], dim=0)
print(Anz.shape)

# Remove row of zero and remove 16 first bands that are null
nbremove = 16
Ynz = torch.cat([Y[nbremove:,0:1], Y[nbremove:,2:]], dim=1)
wavelengths_nz = wavelengths[nbremove:]
print(f"Measurements shape after removing zero rows and first 16 bands: {Ynz.shape}")
# five bands binning
bin_size = 8
Ynz_binned = tl.zeros((Ynz.shape[0]//bin_size, Ynz.shape[1]))
wavelengths_binned = []
for i in range(0, Ynz.shape[0], bin_size):
    if i+bin_size <= Ynz.shape[0]:
        Ynz_binned[i//bin_size,:] = tl.mean(Ynz[i:i+bin_size,:], axis=0)
        wavelengths_binned.append(np.mean(wavelengths_nz[i:i+bin_size]))
    else:
        Ynz_binned[i//bin_size,:] = tl.mean(Ynz[i:,:], axis=0)
        wavelengths_binned.append(np.mean(wavelengths_nz[i:]))
Ynz = Ynz_binned
wavelengths_nz = wavelengths_binned
print(f"Binned measurements shape: {Ynz.shape}")

# define custom forward and adjoint forward functions
def forward(x):
    # x@Anz.T or Anz@x
    # x is of shape (B, N**2)  # batch first
    # reshape to (B, N, N)
    temp = x.reshape((x.shape[0], img_size, img_size))  # Whyyyyyyyy >????
    temp = meas_op.forward(temp).T  # shape (M, B)
    # Removing the zero row of A changes the forward A@X
    return torch.cat([temp[0:1,:], temp[2:,:]], dim=0)
def adjoint(y):
    # Also implemented as a contraction in spyrit
    return y@Anz

# Pseudo-inverse reconstruction, the NNLS is quite slow here
X_rec = torch.linalg.lstsq(Anz, Ynz.T).solution.T

# Show cat reconstructed with all wavelengths
plt.figure(figsize=(6,4))
plt.imshow(np.rot90(torch.sum(X_rec, dim=0).reshape(64,64), 2), cmap='gray')
plt.title("Reconstructed image at all wavelengths")
```

(continues on next page)

(continued from previous page)

```
plt.colorbar()
plt.axis('off')

plt.show()
```

```
torch.Size([8191, 4096])
Measurements shape after removing zero rows and first 16 bands: torch.Size([2032,␣
↪8191])
Binned measurements shape: torch.Size([254, 8191])
```



We now perform the pure-pixel selection and reconstruct/unmix jointly with an alternating MU algorithm. Observe how the abundance maps are well estimated with both methods, but the spectra are slightly smoother with the alternating MU algorithm. The third component (and in fact, most other components if the rank is increased) models the border of the overlap zone between the filters, which is intense because at the center of the image, and is not well modeled by a linear combination of the two filter spectra.

```
# Init pinv+spa
rank = 3
Kset, W0, A0 = snpa(X_rec, rank, verbose=True)

# Reconstruction with NMF
from tensorly_hdr.nmf_kl import MU_SinglePixel_fast
lmbd = 1e-4
W_est, A_est, crit = MU_SinglePixel_fast(Ynz, forward, adjoint, tl.abs(A0), tl.
↪abs(W0), lmbd=lmbd, maxA=1, niter=60, n_iter_inner=20, eps=1e-8, verbose=True,␣
↪print_it=20)  # regularization just for implicit scaling
print("Computation done")
```

```
0 [0, 0, 0]
1 [1688, 0, 0]
2 [1688, 2547, 0]
```

(continues on next page)

(continued from previous page)

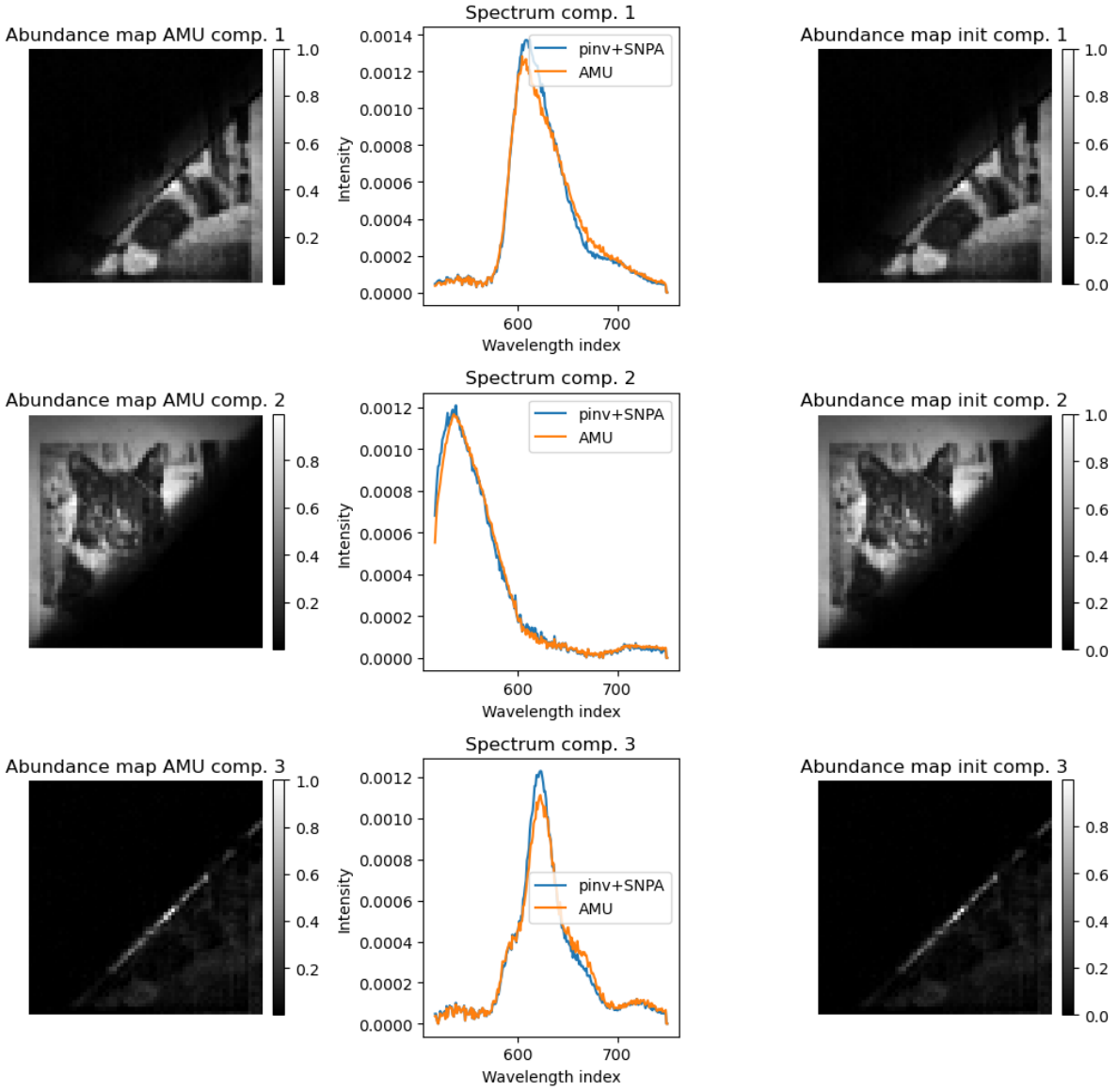
```
Returning [1688, 2547, 1816] as estimated pure pixel indices
```

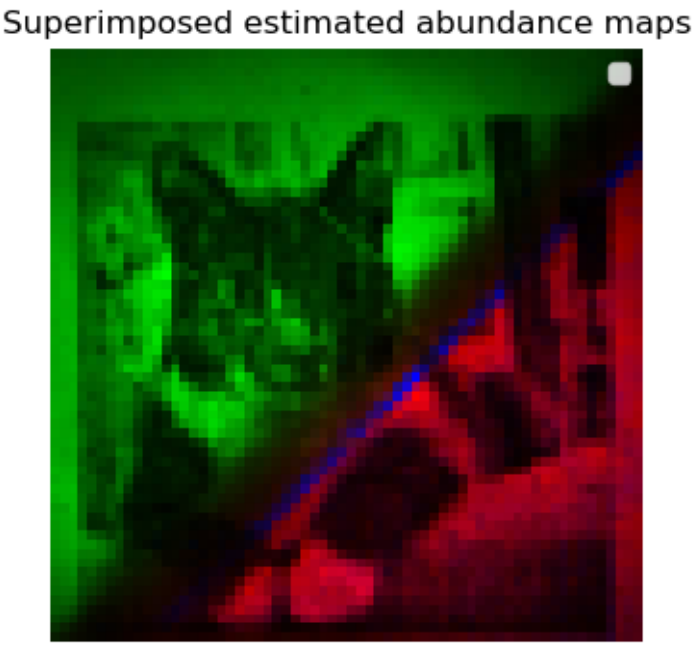
```
Iteration 0, Cost: 3.2183761596679688
```

```
Iteration 20, Cost: 3.169161558151245
```

```
Iteration 40, Cost: 3.1316823959350586
```

```
Computation done
```




Superimposed estimated abundance maps

Part VII

Perspectives and open problems

CHAPTER

TWENTYTHREE

NUMERICAL OPTIMIZATION FOR KL-BASED REGULARIZED INVERSE PROBLEMS

The primary focus of my research over the past decade has been on rLRA. rLRA is a discipline at the intersection of inverse problems, statistics, numerical optimization, and machine learning, with diverse applications such as music information retrieval and spectral imaging that require specialized expertise. While I have always enjoyed embracing the diversity of the mathematical tools required to make contributions to rLRA, in the coming years, I want to focus especially on numerical optimization, with potential contributions beyond rLRA. I enjoy the idea that, in numerical optimization, there is often a clear problem-solving objective, such as cost minimization or speed maximization, that allows us to compare methods. I also enjoy the mathematical framework of both smooth and discrete optimization problems and algorithms. Numerical optimization provides an opinionated view of the world, as many tasks can be expressed in this framework. On the mathematical side, theoretical results and proofs are both intuitive and rigorously enunciated.

A family of problems that I want to study in particular, related to nonnegativity of the parameters and data, is KL-divergence-based estimation, such as *NN-KL*. KL-divergence is a rich loss function to study because it lacks Lipschitz-smoothness at zero (its gradient tends to infinity), but it is also asymptotically flat towards infinity. This makes the design of global strategies, such as gradient descent with a fixed stepsize or global majorization-minimization, rather difficult. In what follows, I explain why the KL-divergence arises in inverse problems, what the hypotheses and challenges of my research project are, and its position in the state-of-the-art. I then detail the various contributions I have in mind for the coming years with identified collaborators.

Other, smaller-scale aspects of my research project are detailed in *a separate section*. This section contains redundant information on the basic mathematical tools described in the rest of the HDR manuscript to ensure that it can be read independently, to some degree.

23.1 Scientific context

KL-divergence, a fundamental measure of similarity between probability distributions in machine learning, naturally arises from *Poisson maximum likelihood models* [Fessler, 1995], but also provides a robust measure of discrepancies compared to the Euclidean norm. This robustness is particularly relevant for audio spectrograms that exhibit large dynamic ranges and for low signal-to-noise ratio counting processes, which could be considered the future of many existing computational imaging devices, offering reduced acquisition time. KL-divergence between two vectors y (the data) and Ax (the unknown) in $\mathbb{R}^n_+$ is defined as

$$\mathcal{D}_{\mathrm{KL}}\left(y, Ax\right) = \sum_{i=1}^{n} y[i] \log\left(\frac{y[i]}{Ax[i]}\right) + Ax[i] - y[i].$$

Inverse problems aim at estimating physically meaningful parameters (images, audio representations, sources) from incomplete and noisy data. In this research project, I focus on two problems: Nonnegative KL regression problems (NN-KL) and Nonnegative matrix and tensor KL factorization problems (NMF-KL). See the *dedicated chapter on NN-KL* and Fig. 23.1. NN-KL reconstructs or denoises images in many computational imaging applications such as tomography, Compton camera, cryoEM, and *single-pixel imagers* [Green, 1990, Hariga *et al.*, 2025]. NMF-KL is a linear dimensionality

reduction technique similar to PCA, where nonnegativity ensures an interpretable part-based representation [Hien and Gillis, 2021, Kervazo *et al.*, 2024, Modrzyk *et al.*, 2025]. NMF-KL is also a blind extension of NN-KL and has been used extensively to analyse spectrograms found in music information retrieval tasks, in particular to perform *Automatic Music Transcription* that translates music recordings into MIDI files (numerical music sheets) [Wu *et al.*, 2022]. NN-KL and NMF-KL problems are generally ill-posed: there are too few measurements or too much noise to uniquely and precisely recover the unknowns, which compromises interpretability. Regularization is thus essential. It can be introduced via handcrafted priors (*e.g.*, sparsity, smoothness), via pre-trained priors using Plug-and-Play (PnP) denoisers, or *Unrolled algorithms* that exploit training datasets [Hurault *et al.*, 2023, Kervazo *et al.*, 2024, Le Roux *et al.*, 2015].

23.1.1 Hypotheses and challenges

I hypothesize that it is possible to design algorithms that are both fast and provably convergent for KL-based regularized inverse problems. Purely from a smooth numerical optimization perspective, the main difficulties are

1. The lack of Lipschitz smoothness at zero of the KL-divergence, which causes instabilities with sparse data or low counts when using first-order methods. This explains why most existing approaches rely on second-order information, typically through separable quadratic surrogates derived from Hessian approximations.
2. The asymptotic linearity of the KL-divergence, encountered when the term $Ax[i]$ dominates the cost, and in particular when the data has many zeros. The objective flatness makes any gradient-based algorithm hard to use: the cost is not strongly convex, and the Hessian has near-zero singular values. Both these issues are known to imply slower convergence.
3. The presence of nonnegativity constraints, as well as other regularizations, possibly based on deep learning. Our challenge is twofold. First, to optimize the design of algorithms for NN-KL and NMF-KL, striking the right balance between accuracy and per-iteration cost. I plan to rely on local (rather than global) approximations of the cost function and on tools from linear programming. Local majorants are expected to work well because they do not need to diverge at zero, but their asymptotic flatness, in particular for sparse data, must be handled carefully. Second, to extend these strategies to include data-driven regularizations, which fall within variable metric forward–backward methods and remain difficult to implement for both handcrafted and data-driven priors [Chouzenoux *et al.*, 2016]. Beyond algorithm design, I will unify contributions scattered across numerical optimization, source separation, and computational imaging, producing a benchmark and a toolbox for KL-based inverse problems. The methodology will be validated on two applications where I have strong expertise: *SPI* and *AMT* [Kervazo *et al.*, 2024, Wu *et al.*, 2022]

23.1.2 Positioning with respect to the state of the art

MU, or ML-EM in computational imaging, remains the historical baseline for KL-based problems. MU is simple to implement and uses second-order information via preconditioning. It is a baseline that can be slow, unstable for sparse data, and difficult to adapt with regularizations. However, it is not so easily beaten, especially in the context of NMF-KL [Hien and Gillis, 2021]. MU has a wide number of variants, some of which have been developed in the computational imaging community and involve block-coordinate updates [Erdogan and Fessler, 1999, Fessler, 1995, Green, 1990], while others come from the signal processing community and focus, for instance, on all-at-once updates for wider classes of loss functions [Marmin *et al.*, 2023] or high-order tensor decompositions [Hood and Schein, 2026]. Most algorithms for NN-KL and NMF-KL, including MU, fall within the majorization-minimization framework, where the cost is globally majorized by a simpler function, often separable with respect to each parameter, and the majorant is then minimized efficiently. I will use local approximations that do not fall within the majorization-minimization framework.

Classical constrained formulations lack a general framework for KL-based problems. MU can be applied, after non-trivial modifications, for a limited set of priors such as ℓ_p norms and TV regularization. Data-driven approaches (Plug-and-Play, Unrolled NMF) are promising for achieving state-of-the-art performance while ensuring model interpretability, yet their KL-based theoretical grounding is weak. In particular, PnP algorithms for KL-divergence rely on a loose Bregman divergence, Burg's entropy, resulting in poor convergence speed [Bauschke *et al.*, 2017, Hurault *et al.*, 2023]. Burg's entropy is used in proximal mirror descent to build a global majorant of the cost, but it can be observed on the following

simple synthetic example that MU is typically much faster than Burg's entropy. Damien Lesens has also shown, in a personnal communication, that the majorant of MU is optimal within the set of separable majorants and is therefore always better than the majorant relying on Burg's entropy. The reason why Burg's entropy is used despite its loose majoration of the KL-divergence is that it is unclear how to relate the majorant leading to MU or other existing faster algorithms with proximal mirror gradient descent, and therefore obtain a clean setup for non-Euclidean data-driven algorithms. I hypothesize that such links can be obtained, and that other frameworks than mirror gradient descent may be used for convergent data-driven non-Euclidean algorithms.

Remark

The topic of linking MU and mirror descent is under active research within the TOMORADIO team.

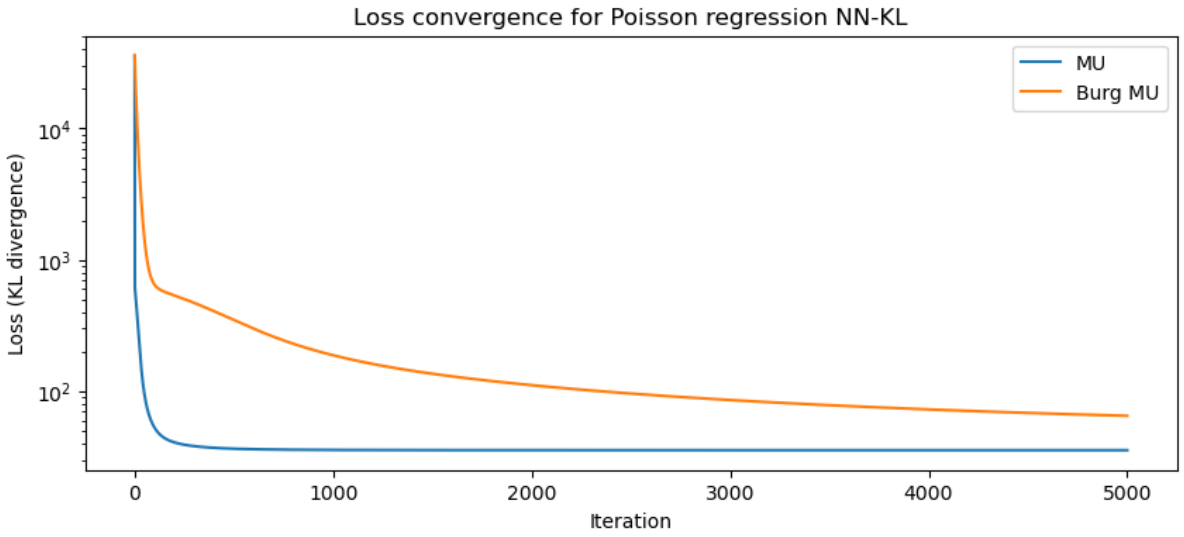
```
from matplotlib.pylab import f
import numpy as np

# Let's compare MU and Bregman mirror descent on a simple Poisson regression problem

# Hyperparameters
m, n = 100, 20  # number of samples and features
alpha = 100 # Poisson signal level

hgt = np.maximum(np.random.randn(n),0)
W = np.random.rand(m,n)
y = np.random.poisson(alpha * W @ hgt)

def loss(h):  # KL divergence
    return np.sum(W@h-y) + np.sum(y * np.log((y+1e-10)/(W @ h)))

def MU(y, W, h, itermax=1000, epsilon=1e-10):
    Wtsum = W.T @ np.ones_like(y)  # suboptimal but ok
    hout = np.copy(h)
    loss_val = [loss(hout)]
    for i in range(itermax):
        hout = np.maximum(epsilon, hout * (W.T @ (y / (W @ hout ))) / (Wtsum))
        loss_val.append(loss(hout))
    return hout, loss_val

def BurgMU(y, W, h, itermax=1000, epsilon=1e-10):  # NoLIPS by Bauschke et al
    Wtsum = W.T @ np.ones_like(y)
    lamb = 1/2/np.sum(y)
    hout = np.copy(h)
    loss_val = [loss(hout)]
    for i in range(itermax):
        hout = np.maximum(epsilon, hout / ( 1 + lamb* hout * ( Wtsum - W.T @ (y / (W␣
 ↪@ hout )))))
        loss_val.append(loss(hout))
    return hout, loss_val


hmu, loss_mu = MU(y, W, alpha*np.ones(n), itermax=5000)
hburg, loss_burg = BurgMU(y, W, alpha*np.ones(n), itermax=5000)


# Printing the final estimation error
print(f"Final loss MU: {loss_mu[-1]}, Burg MU: {loss_burg[-1]}")
```

(continues on next page)

(continued from previous page)

```
print(f"Estimation error MU: {np.linalg.norm(hmu - alpha*hgt)/np.linalg.
↪norm(alpha*hgt)}, Burg MU: {np.linalg.norm(hburg - alpha*hgt)/np.linalg.
↪norm(alpha*hgt)}")
```

```
Final loss MU: 35.52134837213764, Burg MU: 64.91138311001241
Estimation error MU: 0.07230955322797404, Burg MU: 0.20385714656018994
```



Finally, existing algorithms are designed for small, dense matrices, whereas audio applications also lead to sparse, large matrices. In summary, no existing method simultaneously achieves speed, robustness to data sparsity, and theoretical guarantees, leaving a clear gap that I aim to fill.

23.2 Methodology and scientific coverage

The research project is divided into four complementary subtasks to achieve the global objective of conceiving faster regularized algorithms for KL-based inverse problems, see Fig. 23.1. It is grounded in the numerical optimization community, but with strong interactions with the machine learning and signal processing communities.

23.2.1 1. NMF toolbox and benchmark

As of early 2026, it remains far from straightforward to navigate the world of NMF algorithms for researchers and end users alike. On the side of models, there are several loss functions to choose from depending on the data type and noise distribution, and various regularizations can be considered. Even for a bare-bones model such as NMF-KL, the literature does not clearly specify which algorithm to use in which context. Choices to be made have several layers of complexity: should the algorithm alternate over the two parameter matrices, or update both simultaneously? Should one use a majorization-minimization algorithm such as MU, a primal-dual algorithm, mirror descent with Burg entropy, or a second-order-inspired algorithm such as *the proposed mSOM*? Are extrapolation and momentum needed? The actual performance gap between these methods depends heavily on the application at hand. In particular, in the context of NMF-KL for document mining or audio processing, data sparsity and data size can be critical, and the implementation of algorithms (sparsity handling, efficient caching of operations) matters.

I believe that performing meaningful research on NMF-KL algorithms requires first taking a step back and analyzing precisely the current state-of-the-art. In particular, I plan to build a benchmark for NMF models, starting with NMF-

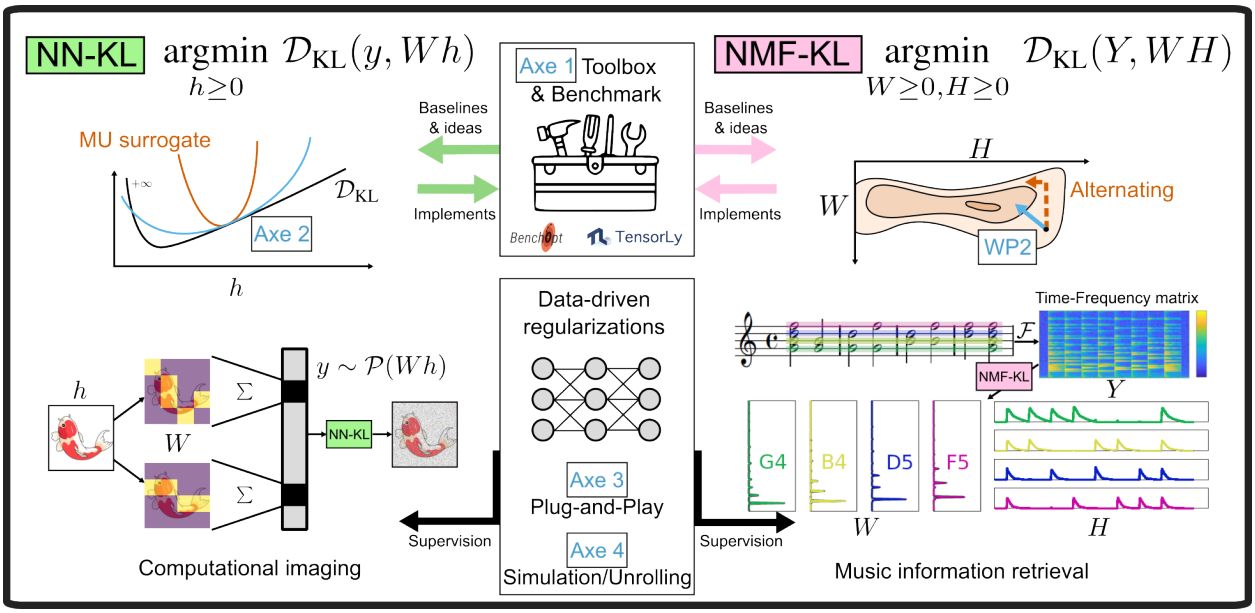


Fig. 23.1: Schematic organisation of my research project, its connections with numerical optimization and inverse problem in spectral imaging and music information retrieval.

KL, that includes as many algorithms from the literature as possible, and datasets publicly available from applications in document mining, music information retrieval, hyperspectral imaging in remote sensing and microscopy, computer vision, and chemometrics. The backbone of the benchmark was already developed in 2022 in collaboration with Cassio Fraga-Dantas, relying on Benchopt [Moreau *et al.*, 2022], but needs to be enhanced with algorithms and a dataset.

Alongside the development of this benchmark, the implementation of existing algorithms will be made available in a dedicated NMF toolbox in Python, along with other useful tools such as separable NMF algorithms and *Borgen plots* discussed later in the perspectives, and an integration of efficient implementations in a lower-level language of important routines such as *AS and HALS*. Available Python packages for NMF include scikit-learn, which implements only multiplicative updates for NMF-KL, and Nimfa, a toolbox no longer maintained that focuses on model diversity (including many Bayesian and graph-regularized variants of NMF) rather than on efficient algorithms for a few targeted models. The proposed NMF toolbox may rely on tensorly to be backend-agnostic (allowing lower-level operations to be performed transparently by NumPy, PyTorch, JAX, etc.). The architecture of the package is not yet decided, and it may be built on top of only PyTorch for simplicity and support for auto-differentiation, or in Julia for better integration of efficient sparse-oriented routines such as *sparse NNLS*.

Collaborators: The benchmark is already under construction by Damien Lesens (ENS Lyon, LIP) as part of his PhD thesis, co-supervised with Bora Uçar (CNRS, LIP). It will be completed by me and all supervised students working on problems related to the KL-divergence. Collaborators from Telecom Paris (Christophe Kervazo, Mathieu Fontaine, Roland Badeau) and IMT Atlantique (Quyen Pham) are also expected to participate. If this benchmark finds echo in the NMF community, other research groups may join the initiative, such as the group of Nicolas Gillis in Mons that lacks such a unified software platform.

23.2.2 2. New numerical optimization tools for faster NN-KL and NMF-KL

NN-KL is a rather old optimization problem, and both researchers and users (in particular those in computational optics) have proposed numerous ideas for designing fast and accurate solvers. One key scientific lever has been the design of global majorants of the KL-divergence, in direct relationship with the EM framework. In this context, it may seem unrealistic to assert that my collaborators and myself will propose faster algorithms for NN-KL. NMF-KL has been studied less than NMF, since NMF is far less commonly used than linear regression, but the same observation could be made.

This research project is built on the idea that some important research directions in optimization with the KL-divergence have not been explored. First, there are relatively few methods that do not rely on global MM. While MM is useful for designing convergent algorithms, the geometry of the KL-divergence, which diverges at zero yet is asymptotically linear, makes it difficult to design majorants that are both global and tight. Popular majorants such as those used in *MU* are also separable. Recent works on regularized NN-KL and NMF-KL based on mirror gradient descent use Burg's entropy to construct a majorant, which is worse than the MU majorant. One promising research direction is to resort to **local separable majorants using second-order information**. In an ongoing work with Mai Quyen Pham and Thierry Chonavel [Pham *et al.*, 2025], we show that we can design local majorants of the cost by building diagonal majorants of the Hessian matrix iteratively, of the form

$$M(u, x) = \text{Diag}\left(\frac{H(x)u}{u}\right),$$

where $H(x)$ is the Hessian matrix at point x and u is a vector that represents a diagonal matrix from this class. For any positive u, it can be shown that $M(u, x) - H(x)$ is symmetric and definite positive. In other words, we can build a local majorant of the cost with curvature given by the matrix $M(u, x)$. Diagonal approximations of the Hessian matrix allow decoupling of the parameters in the minimization of the majorant, which is important for scalability. Our work opens many perspectives, as we study a single class of local majorants, and pick a particular item in that class that can be obtained in closed form. We plan to find even better local majorants by solving small-scale optimization problems. We also plan to extend this strategy to all-at-once algorithms and explore similar ideas within the framework of mirror descent.

Second, KL-divergence has a linear term $\sum_{i,j} A[i, j]x[j] - \sum_i y[i]$, that links NN-KL with linear programming. More precisely, the KKT conditions show that, at optimality, this sum must cancel out. This allows us to replace the linear term in the KL-divergence minimization problem with a linearly constrained problem

$$\underset{1^T y = 1^T A x}{\text{argmin}} \; -\sum_i y[i] \log((Ax)[i]).$$

I hypothesize that it can be used to build efficient algorithms for sparse NN-KL and sparse NMF-KL problems. Indeed, if most entries of the data vector y are null, the problem is similar to an interior-point method for linear programming with a logarithmic barrier function. It is therefore tempting to associate NN-KL for sparse data with a linear program that could be used to initialize other algorithms. It is also interesting to explore the use of the Sinkhorn algorithm to enforce this scaling within any alternating algorithm for NMF-KL, since these marginal constraints are, in general, not satisfied during the iterations. Finally, second-order methods will be considered since the Hessian matrix should be highly structured, and linear constraints can be incorporated. Most algorithms on the market simply ignore the possibility that the data matrix and/or the parameters can be extremely large and extremely sparse. While most algorithms can be easily adapted to handle sparse data, some are not so easy to adapt, such as primal dual algorithms, where the dual variables are generally dense.

Collaborators: The design of local majorants is the purpose of a long-term collaboration with Mai Quyen Pham (IMT Atlantique). We plan to continue working on this topic together by recruiting a PhD student. The study of the links between KL-divergence optimization and linear programming is the topic of Damien Lesens's PhD thesis, co-supervised with Bora Uçar. The topic of optimization for KL-divergence and more generally Poisson-distributed data is currently under scrutiny by several research groups, including Voichita Maxim in my team at CREATIS, Lucas Calatroni (I3S, Nice and University of Genoa, Italy), Emilie Chouzenoux (Inria Saclay), Marceylo Pereyra (Heriott-Watt University, Edinburgh), Nicolas Gillis (UMONS, Belgium), and Shota Takahashi (University of Tokyo). This opens the way for new international collaborations. Research on nonnegative KL-divergence problems may also generalize beyond the realm of NMF to quadratic programming [Sha *et al.*, 2007] and any nonnegative einsum factorization [Hood and Schein, 2026].

23.2.3 3. KL-based inverse problems for computational imaging

Computational imaging has a long history of handling Poisson noise in inverse problems, probably more so than any other field within the scope of signal processing. Historically, Poisson noise was handled by a variance-stabilizing transformation (Poisson noise variance depends on the signal). The Anscombe transform is such a transformation of the data entrywise so that the noise distribution becomes Gaussian with a uniform variance. Variance-stabilizing transformations have the advantage of being fast at inference, since the forward and inverse transformations are cheap, and algorithms solving inverse problems with Gaussian noise are generally significantly faster than their KL-divergence counterparts. The Anscombe transform and other similar techniques, however, suffer in the presence of low-count data. KL-divergence is the loss function that appears when computing the MLE for inverse problems with Poisson noise. While it does not introduce statistical errors like variance-stabilizing transformations, it leads to significantly more expensive algorithms.

In modern signal processing and machine learning algorithms for inverse problems, we build on classical estimation algorithms, such as NN-KL solvers, using prior information provided by deep models. The general idea is that a neural network can be trained on denoising problems to learn the distribution of clean data, which in turn allows the construction of efficient priors for other inverse problems. The Plug-and-play (PnP) framework [Venkatakrishnan *et al.*, 2013] simply plugs a pretrained network in place of a proximal (projection) operator in iterative proximal gradient algorithms, while other methods like RED [Liu *et al.*, 2021] use differentiable architectures plugged in the cost function.

It is important to acknowledge that, while these data-driven techniques significantly improve the estimation performance of algorithms in inverse problems, they inherit the ups and downs of classical methods from which they are built. I believe that, while fellow researchers have focused on improving the data-driven aspect of data-driven algorithms for Poisson inverse problems based, for instance, on generalized Tweedie's formula [Efron, 2011], **the interplay between deep regularizers, classical methods including variance stabilizing transformations, and specific computational imaging applications is under-explored**. There is probably no need for data-driven KL-divergence-based algorithms for high-count data when the Anscombe transform can be applied without issue. In contrast, there is a clear need for a more detailed analysis of when to use heavy computational tools, such as PnP, with complex NN-KL solvers. Nevertheless, part of this project is also to develop such data-driven Poisson algorithms; this is ongoing work within Serena Hariga's PhD thesis [Hariga *et al.*, 2024, Hariga *et al.*, 2025].

We will target our contributions towards the computational imaging modalities developed at CREATIS, in particular *spectral SPI*. SPI is interesting because the Poisson noise level depends on the acquisition technique and on the output wavelength. The energy of incoming photons is split between spectral bands, and some spectral bands receive significantly fewer photons than others. This kind of data is perfect for analyzing the effect of signal noise on estimation performance. Moreover, the spectral data require unmixing, which can be performed efficiently with NN-KL or NMF-KL in the blind case, inside the reconstruction problem. In the blind case, the problem of estimating sources and abundances in SPI can be formulated as

$$\underset{U\geq 0,\, V\geq 0}{\operatorname{argmin}} \; \mathcal{D}_{\mathrm{KL}}\left(Y, HUV^T\right) + g_1(U) + g_2(V)$$

where H is the acquisition matrix (such as Hadamard patterns), and g_i are deep or classical regularizations required to ensure interpretability of the results and proper image reconstruction. A particularity of SPI is that the noise can be modeled as a mixture of Poisson and Gaussian noise, for which the exact MLE is not known in closed form and is often approximated using variance-stabilizing transformations [Anscombe, 1948, Fryzlewicz and Nason, 2004, Makitalo and Foi, 2011, Makitalo and Foi, 2011, Bar-Lev and Enis, 1988]. Other interesting modalities include the spectral CT scanner, compton camera, and cryo-EM [Singer, 2018].

Algorithms developed in the second research direction will also be good candidates for extensions to the data-driven framework. Some authors focus on Burg's entropy for building data-driven algorithms with Poisson noise [Hurault *et al.*, 2023], but better results should be achieved by considering better majorants of the KL-divergence, at the cost of more difficult convergence proofs.

Collaborators: SPI is the topic of an ongoing, fruitful collaboration with Nicolas Ducros (INSA Lyon, CREATIS). The PhD thesis of Serena Harriga, co-supervised with Prof. Ducros, which started in 2023, focuses on primal-dual data-driven algorithms for SPI. At the national level, joint spectral unmixing and inverse reconstruction in the context of Poisson noise is one of the topics discussed in an ANR project under construction with HORIBA, a spectrometer manufacturer, and in

collaboration with the DyNaChem team of the LASIRE (Lille). The Poisson-Gaussian noise and equivalences are a point of common interest with Valentin Debarnot, recently recruited to the team.

23.2.4 4. Automatic transcription using NMF-KL, neural architectures, and unsupervised learning

NMF has a 20-year history of use in music information retrieval. One of its first use cases, which has remained an application of choice for NMF, is Automatic Music Transcription (AMT) [Smaragdis and Brown, 2003]. NMF has shown promising performance for AMT, in particular when KL-divergence (and other $beta$-divergences) are used as loss functions. It has been outperformed by deep networks trained end-to-end for almost a decade [Hawthorne *et al.*, 2018].

Deep learning for AMT leverages training samples. The raw comparison with NMF, which has been used as an unsupervised model in AMT, is unfair. However, it is not clear at all how NMF could benefit from training samples. I have identified two promising routes.

First, one may use the *unrolling* framework to train parts of NMF for AMT. With Christophe Kervazo, we have recently proposed unrolled MU updates, with application to spectral unmixing. Our preliminary experiments on unrolled NMF for AMT show that this application is quite challenging because audio data size, sparsity, and dynamics pose challenges for current approaches. It should be possible to define a suitable unrolled state-of-the-art algorithm, a training procedure, and compare symmetric unrolling (both W and H) to older approaches that unroll H as a function of W [Le Roux *et al.*, 2015].

Second, training deep networks for AMT, and in fact for most music information retrieval tasks, is costly in terms of the amount of training data required. While training data are available for some instruments, such as the piano, thanks to acoustic instruments with activation sensors (Yamaha Disklavier), for rare instruments, it is unlikely that such a database can be acquired in the coming decades. Therefore, there is still room for unsupervised learning algorithms for AMT. NMF models have seen several refinements in recent years regarding identifiability (minimum volume, separability, and, of course, better NMF-KL algorithms) that can be helpful in their fruitful application to a source separation problem like AMT. I will also go further by leveraging both recent NMF developments and Differential Digital Signal Processing, a state-of-the-art framework for unsupervised learning in deep learning for audio. DDSP essentially trains encoder networks to play software synthesizers and reproduce unlabeled instrument recordings, given the knowledge of pitch class and sound amplitude [Engel *et al.*, 2020]. NMF could be trained alongside a DDSP model to serve as a generalized pitch-detection algorithm.

Research on AMT is, by nature, translational, and therefore it should be a priority to produce open-source software with a user-friendly interface and a Python backend that can be used to transcribe multi-instrument professional recordings, provided that the research is sufficiently fruitful to achieve satisfactory performance.

Collaborators: I started working on AMT with Nancy Bertin in Rennes, who is currently on leave from CNRS, and with Axel Marmoret, my former PhD student. Axel is still working on AMT, but because I want him to be as independent as possible, I refrain from collaborating with him on this topic. Since 2023, we have started to collaborate with Christophe Kervazo and Mathieu Fontaine (Telecom Paris) on *unrolled NMF* for AMT. We are currently seeking motivated PhD candidates and funding.

CHAPTER

TWENTYFOUR

OTHER PERSPECTIVES

The topics discussed in this section are scientific questions that I want to address, which are not directly related to the KL-divergence and Poisson noise.

24.1 Single-pixel imaging and compressive acquisition

SPI has already been discussed in *Single-pixel hyperspectral image reconstruction and unmixing*. While the reconstruction algorithm is critical, to improve the overall algorithmic and hardware pipeline, the acquisition system is maybe even more important. The acquisition is modeled as a linear system $y = Hx$, where the matrix H contains, in each row, the pattern of ones and zeros representing the micro-mirror positions at each acquisition. There are two relatively open questions regarding the choice of these patterns.

As discussed in *Single-pixel hyperspectral image reconstruction and unmixing*, the acquisition matrix coefficients have to belong to the interval $[0, 1]$. The statistical optimality of Hadamard patterns holds for additive noise and entries in $[-1, 1]$. There are other acquisition matrices, such as S matrices [Harwit and Sloane, 1979], that are shown to be optimal up to a factor 2. We are working on a research paper summarizing these ideas [Ducros *et al.*, 2025]. Ongoing research in the team is also concerned with learning the acquisition patterns from a training dataset, or from the color camera in a multimodal acquisition setup. I will not explore this direction.

Once the patterns have been chosen, another question of importance is how to subsample the acquisition patterns to acquire $y_S = H[S, :]x$, where S is the index set of acquired patterns. The compressive sensing literature informs how to choose these patterns, primarily by improving reconstruction with sparsity-based priors. Due to Poisson noise and the use of other reconstruction methods, in particular those based on deep priors, practical observations lead to a different policy for choosing the samples S. A better understanding of the impact of acquisition patterns and acquisition subsampling on reconstruction performances is therefore needed. This topic is under scrutiny with Nicolas Ducros in collaboration with Laurent Jacques (UCLouvain), Marc Georges (ULiège), and Clément Thomas (PhD student ULiège).

24.1.1 Spectral unmixing dataset acquisition

A problem that plagues signal processing research in hyperspectral imaging is the lack of reliable ground truth datasets. In remote sensing, where hyperspectral images often span several kilometers, it is simply not possible to go on the ground at the exact time the image was acquired and measure the spectra of each macroscopic element. Even if this were possible, the acquisition parameters, such as the incidence angle or illumination, impact the remotely acquired images. In hyperspectral microscopy, however, finer control over the scene may be possible. In Serena Hariga's PhD thesis, we tried to print monochromatic shapes, hoping that the printer's inks would be combinations of three primary colors, resulting in the predictable mixing of three components distributed spatially. However, the printer does not rely solely on *additive mixing*, and the printed image is therefore not rank three.

With Nicolas Ducros, we plan to design another acquisition protocol for hyperspectral microscopic images with controlled ground truth. One possible direction is to use several lamps to illuminate a black and white scene and rely on the additive

spatial mixing of the two lights. The challenge would be to control precisely the spatial distribution of each light in the scene.

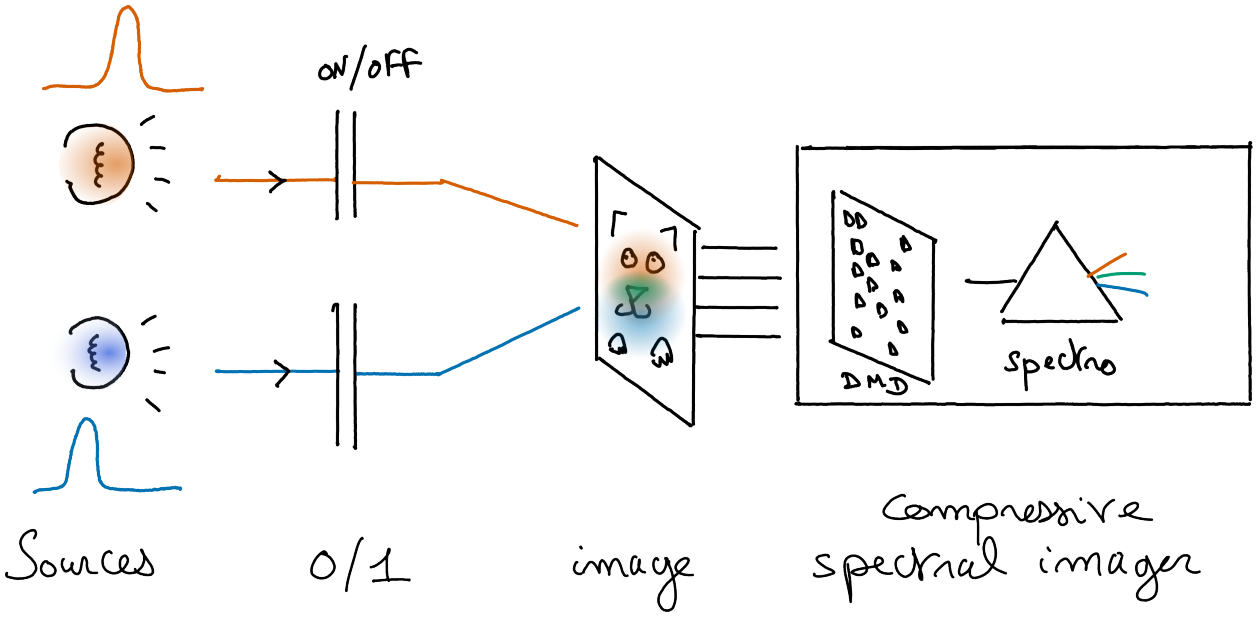


Fig. 24.1: Schematic illustration of the acquisition of a dataset for spectral unmixing in microscopy. Two lamps with different spectra are used to illuminate the same scene (a black and white image), and their spatial distribution partially overlap. This overlap produces a spatially varying additive mixture of the two spectra. The scene is acquired with any spectral imager such as the single-pixel spectral camera available at CREATIS. It is possible to obtain the ground-truth abundances for each component by shutting either one of the lamp. For instance, shutting down the red lamp means that we can acquire the abundances for the blue spectrum only. The challenge is to control rather accurately the spatial distribution of the illumination for each lamp, and to mitigate nonlinearities and experimental variability.

24.2 Inverse problems and separable NMF

Once the single-pixel data has been acquired, one may want to perform the reconstruction and the spectral unmixing jointly. This is the research project of Serena Hariga, currently finishing her PhD. We are designing Primal-Dual algorithms with data-driven priors that yield complicated algorithms even when spectral unmixing is not blind (the matrix W containing the spectra is known). In the future, we aim to develop a stable algorithm for reconstructing and performing blind spectral unmixing. One important missing step is a good initialization algorithm.

A powerful way to initialize NMF in other contexts is by performing *separable NMF*. Separable NMF searches for the spectra directly in the data matrix. However, in the context of inverse problems, this data matrix is not available. Ignoring the Poisson nature of the noise, the inverse separable NMF model may be formalized as the following optimization problem:

$$\underset{X\in\mathbb{R}^{n\times p},\, V\in\mathbb{R}_+^{p\times r},\, \#S=r}{\operatorname{argmin}} \|Y - HX\|_F^2 + \|X - X[:,S]V^T\|_F^2 + g(X)$$

where Y is the spectral measurement matrix, H the acquisition matrix, S the indices of the pure pixels in the reconstructed matrix X and $g(X)$ is some regularization, typically data-driven. The issue with this simple formulation is that it is unclear how to derive a fast algorithm to compute the solutions. The strength of separable NMF lies in the fact that algorithms such as SNPA [Gillis, 2014] that solve it are computationally cheap.

There are several ways to solve this issue. First, in the case where the acquisition matrix H is orthogonal, one may simply run separable NMF on H^TY. But as soon as the patterns are subsampled, this strategy may not perform well. Our goal is

therefore to improve the reconstruction pipeline compared to a two-step strategy, in which reconstruction and separable NMF are performed sequentially. Another possible direction is to use other variational formulations of separable NMF recently proposed [Pan *et al.*, 2025]. It may be possible to design a fast algorithm for inverse separable NMF when the regularization g is ignored or is simple enough.

The core problem behind inverse separable NMF is that performing spectral unmixing in the measurement domain of an inverse problem is inherently difficult. Indeed, spectral unmixing relies heavily on the geometry of the data: the true spectra form a cone containing the data, which can often be observed directly by plotting the data. The forward operator of an inverse problem mixes all the data points, and the geometry of the problem is consequently heavily modified. Whether it is possible and useful to perform spectral unmixing on the measurement coefficients using a geometric algorithm is an open question.

24.3 Borgen plots revisited

Many theoretical results on the identifiability of rLRA models exist, allowing a priori selection of the model and constraints adapted to a given practical signal processing problem. While these results are very useful, in most practical cases, they cannot be applied, as some of the quantities required to prove uniqueness are unknown or difficult to estimate. Another important line of work concerns a posteriori methods to check if the solutions to an inverse problem or rLRA optimization algorithm are unique. I believe this line of work has been largely ignored by the community despite its obvious usefulness for practitioners.

There is a small but deep literature on numerical algorithms for the visualization of solutions to NMF in small ranks (less than four) from the chemometrics community [Borgen and Kowalski, 1985], see [Neymeyr and Sawall, 2018] for a mathematical survey. Borgen and Kowalski proposed what is now sometimes called Borgen plots; see Fig. 24.2 below for an illustration with the rank-three case. The set of solutions to NMF is parameterized explicitly with $r-1$ parameters as

$$Y = UTT^{-1}V^T,$$

where T is a $r \times r$ invertible matrix. The trick to Borgen plots is to note that scaling and permutation ambiguities can be leveraged to restrict the parameterization of the matrix T to that of its first row, and that its first column may be set to 1. Therefore, the set of feasible solutions is parameterized in the rank three case by only two parameters α_1 and α_2, and exhaustive search for all equivalent solutions is reasonably doable. This was proposed rather recently in 2011 by A. Golshan, H. Abdollahi, and M. Maede [Golshan *et al.*, 2011], using covering triangles. It is possible to show other properties on the set of feasible NMF solutions that fasten this process. For instance, the set of solutions is a convex polytope. A MATLAB toolbox exists that provides efficient numerical algorithms for computing Borgen plots [Sawall and Neymeyr, 2014] Facpack website.

Borgen plots are an amazing tool for visualisation of NMF solutions a posteriori. However, there are several problems with them. Most importantly, they have been designed for very small ranks. Visualization becomes more difficult at higher ranks, and as rank and dimensions increase, the numerical strategies based on exhaustive search become intractable. The cost of verifying the validity of a matrix parameterized by the vector α also involves solving a small matrix inversion problem, which can be costly in higher dimensions. Consequently, there is room for the development of methods that scale to higher dimensions [Sawall *et al.*, 2026]. Another issue is that the current design of Borgen plots may return infeasible solutions, and that some solutions may be ignored in the visualization if the data is reducible (in which case the Perron-Frobenius theorem does not apply), or if the set of solutions has several disconnected convex subsets (because the geometric construction of the feasible set works locally).

Beyond NMF, I want to extend numerical representations of solution sets to other inverse problems. One possible direction is to work in the null-space of the Jacobian matrix, as proposed by Nathanaël Munier, Emmanuel Soubies, and Pierre Weiss [Munier *et al.*, 2025].

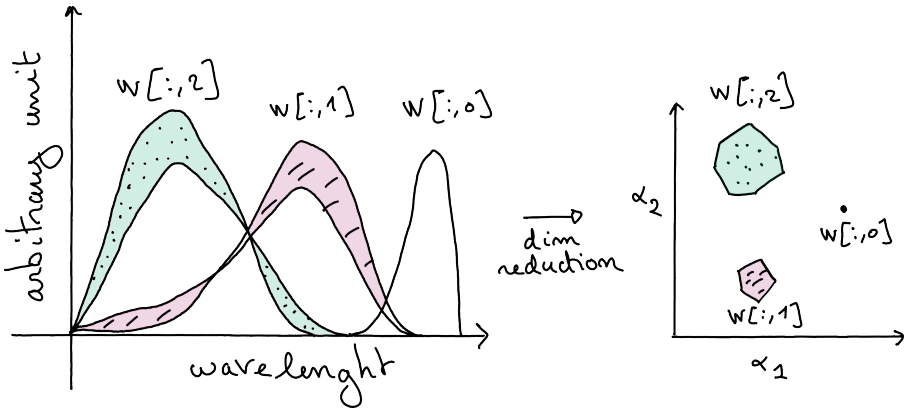


Fig. 24.2: A schematic illustration of the information provided by Borgen plots. On the left, several solutions of NMF are illustrated in the source domain. Several examples of the three components estimated by a rank three NMF are superposed. Borgen plots reduce the dimensionality of the set of solutions to a 2D space parameterized by α_1 and α_2 thanks to SVD and the clever use of symmetries in the problem. The set of solutions are now represented by three polytopes. In this examples, the first component $W[:, 0]$ can be uniquely recovered from the data, but the other two components are not identifiable.

BIBLIOGRAPHY

[ABP+25] J. Abascal, T. Baudier, R. Phan, A. Repetti, and N. Ducros. SPyRiT 3.0: an open source package for single-pixel imaging based on deep learning. *Optics Express*, 33(13):27988–28005, June 2025. doi:10.1364/OE.559227.

[ABG24] Maryam Abdolali, Giovanni Barbarino, and Nicolas Gillis. Dual Simplex Volume Maximization for Simplex-Structured Matrix Factorization. March 2024. arXiv:2403.20197.

[AG21] Maryam Abdolali and Nicolas Gillis. Simplex-Structured Matrix Factorization: Sparsity-based Identifiability and Provably Correct Algorithms. *SIAM Journal on Mathematics of Data Science*, 3(2):593–623, January 2021. arXiv:2007.11446, doi:10.1137/20M1354982.

[ADK11] E. Acar, D. M. Dunlavy, and T. G. Kolda. A scalable optimization approach for fitting canonical tensor decompositions. *Journal of Chemometrics*, 25(2):67–86, 2011.

[ADKMorup11] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup. Scalable tensor factorizations for incomplete data. *Chemom. Intel. Lab. Syst.*, 2011.

[AY09] E. Acar and B. Yener. Unsupervised multiway data analysis: a literature survey. *IEEE transactions on knowledge and data engineering*, 21(1):6–20, 2009.

[ALSCA17] Evrim Acar, Yuri Levin-Schwartz, Vince D Calhoun, and Tülay Adali. Acmtf for fusion of multi-modal neuroimaging data and identification of biomarkers. In *Signal Processing Conference (EUSIPCO), 2017 25th European*, 643–647. IEEE, 2017.

[ARS+13] Evrim Acar, Morten Arendt Rasmussen, Francesco Savorani, Tormod Næs, and Rasmus Bro. Understanding data fusion within the framework of coupled matrix and tensor factorizations. *Chemometrics and Intelligent Laboratory Systems*, 129:53–63, November 2013. doi:10.1016/j.chemolab.2013.06.006.

[AHK15] A. Anandkumar, D. Hsu, and M. Janzamin S. Kakade. When are overcomplete topic models identifiable? uniqueness of tensor Tucker decompositions with structured sparsity. *Jour. Machine Learning Research*, 16:2643–2694, dec 2015. URL: http://jmlr.org/papers/v16/anandkumar15a.html.

[ABB+99] Edward Anderson, Zhaojun Bai, Christian Bischof, L Susan Blackford, James Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, Alan McKenney, and others. *LAPACK users' guide*. SIAM, 1999.

[AECG19] A. Ang, J. E. Cohen, and N. Gillis. Accelerating approximate nonnegative canonical polyadic decomposition using extrapolation. In *XXVIIe Colloque GRETSI*. 2019.

[ACGH20] A.M.S. Ang, J.E. Cohen, N. Gillis, and L.T.K. Hien. Accelerating block coordinate descent for nonnegative tensor factorization. *arXiv preprint arXiv:2001.04321*, 2020.

[Ans48] F. J. Anscombe. The Transformation of Poisson, Binomial and Negative-Binomial Data. *Biometrika*, 35(3/4):246–254, 1948. doi:10.2307/2332343.

[AIHaebUmbach+25] Shoko Araki, Nobutaka Ito, Reinhold Haeb-Umbach, Gordon Wichern, Zhong-Qiu Wang, and Yuki Mitsufuji. 30+ Years of Source Separation Research: Achievements and Future Challenges. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Hyderabad, India, January 2025. doi:10.48550/arXiv.2501.11837.

[AB09] Hedy Attouch and Jérôme Bolte. On the convergence of the proximal algorithm for nonsmooth functions involving analytic features. *Mathematical Programming*, 116(1-2):5–16, January 2009. doi:10.1007/s10107-007-0133-5.

[AP16] Hedy Attouch and Juan Peypouquet. The Rate of Convergence of Nesterov's Accelerated Forward-Backward Method is Actually Faster Than \$1/k\textasciicircum 2\$. *SIAM Journal on Optimization*, 26(3):1824–1834, January 2016. doi:10.1137/15M1046095.

[BSLB08] Davide Ballabio, Thomas Skov, Riccardo Leardi, and Rasmus Bro. Classification of GC-MS measurements of wines by combining data dimension reduction and variable selection techniques. *Journal of Chemometrics*, 22(8):457–463, 2008.

[BK25] Grey Ballard and Tamara G. Kolda. *Tensor Decompositions for Data Science*. Cambridge University Press, Cambridge, 2025. doi:10.1017/9781009471664.

[BBT17] Heinz H. Bauschke, Jérôme Bolte, and Marc Teboulle. A Descent Lemma Beyond Lipschitz Gradient Continuity: First-Order Methods Revisited and Applications. *Mathematics of Operations Research*, 42(2):330–348, May 2017. doi:10.1287/moor.2016.0817.

[Bec17] Amir Beck. *First-order methods in optimization*. SIAM, 2017.

[BAbedMeraimCM97] A. Belouchrani, K. Abed-Meraim, J.-F. Cardoso, and E. Moulines. A blind source separation technique using second-order statistics. *IEEE Transactions on Signal Processing*, 45(2):434–444, 1997. doi:10.1109/78.554307.

[BMMahieuWilliameBD23] Guilherme Beneti Martins, Laurent Mahieu-Williame, Thomas Baudier, and Nicolas Ducros. OpenSpyrit: an ecosystem for open single-pixel hyperspectral imaging. *Optics Express*, 31(10):15599, May 2023. doi:10.1364/OE.483937.

[BDDE18] E. Benetos, S. Dixon, Z. Duan, and S. Ewert. Automatic music transcription: an overview. *IEEE Signal Processing Magazine*, 36(1):20–30, 2018.

[Ber09] Nancy Bertin. *Les factorisations en matrices non-négatives : approches contraintes et probabilistes, application à la transcription automatique de musique polyphonique*. PhD thesis, Télécom ParisTech, 2009. URL: http://www.theses.fr/2009ENST0051.

[Ber99] Dimitri P Bertsekas. *Nonlinear programming*. Athena scientific Belmont, 1999.

[BHH+22] Paolo Bientinesi, David Ham, Furong Huang, Paul H. J. Kelly, P. (Saday) Sadayappan, and Edward Stow. Tensor Computations: Applications and Optimization (Dagstuhl Seminar 22101). *Dagstuhl Reports*, 12(3):1–14, 2022. doi:10.4230/DagRep.12.3.1.

[BDPD+12] J.M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot. Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(2):354–379, 2012.

[BBR+22] Rachel M. Bittner, Juan José Bosch, David Rubinstein, Gabriel Meseguer-Brocal, and Sebastian Ewert. A Lightweight Instrument-Agnostic Model for Polyphonic Note Transcription and Multipitch Estimation. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 781–785. May 2022. doi:10.1109/ICASSP43922.2022.9746549.

[Boa93] JW Boardman. Spectral angle mapping: a rapid measure of spectral similarity. *AVIRIS. Delivered by Ingenta*, 1993.

[BST14] J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, 2014.

[BK85] Odd S. Borgen and Bruce R. Kowalski. An extension of the multivariate component-resolution method to three components. *Analytica Chimica Acta*, 174:1–26, January 1985. doi:10.1016/S0003-2670(00)84361-5.

[BUC25] Ricardo Borsoi, Konstantin Usevich, and Marianne Clausel. Low-Rank Tensor Decompositions for the Theory of Neural Networks. December 2025. doi:10.48550/arXiv.2508.18408.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[Bro96] R. Bro. Multiway calibration, multilinear pls. *Jour. Chemometrics*, 10:47–61, 1996.

[BDJ97] R. Bro and S. De Jong. A fast non-negativity-constrained least squares algorithm. *Journal of chemometrics*, 11(5):393–401, 1997.

[Bud25] Stanislav Budzinskiy. When big data actually are low-rank, or entrywise approximation of certain function-generated matrices. *SIAM Journal on Mathematics of Data Science*, 7(3):1098–1122, September 2025. doi:10.1137/24M1687133.

[BM03] Samuel Burer and Renato D.C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, February 2003. doi:10.1007/s10107-002-0352-8.

[Byr81] Charles L Byrne. *Applied Iterative Methods*. 1981.

[BejarDokmanicV22] Benjamín Béjar, Ivan Dokmanić, and René Vidal. The Fastest \ell \textsubscript 1,ınfty $\ell 1,\infty$ Prox in the West. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3858–3869, July 2022. doi:10.1109/TPAMI.2021.3059301.

[CFCC16] Rodrigo Cabral Farias, Jeremy Emile Cohen, and Pierre Comon. Exploring Multimodal Data Fusion Through Joint Decompositions with Flexible Couplings. *IEEE Transactions on Signal Processing*, 64(18):4830–4844, September 2016. doi:10.1109/TSP.2016.2576425.

[CCMahieuWilliame+23] Charly Caredda, Jérémy E. Cohen, Laurent Mahieu-Williame, Raphäel Sablong, Michäel Sdika, Jacques Guyotat, and Bruno Montcel. Separable spectral unmixing based on the learning of periodic absorbance changes: Application to functional brain mapping using RGB imaging. *Translational Biophotonics: Diagnostics and Therapeutics III*, pages 126271D, June 2023. doi:10.1117/12.2670506.

[CPK80] J Douglas Carroll, Sandra Pruzansky, and Joseph B Kruskal. CANDELINC: a general approach to multidimensional analysis of many-way arrays with linear constraints on parameters. *Psychometrika*, 45(1):3–24, 1980.

[CC70] J. Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319, September 1970. doi:10.1007/BF02310791.

[CSCA25] Christos Chatzis, Carla Schenker, Jérémy E. Cohen, and Evrim Acar. dCMF: Learning Interpretable Evolving Patterns from Temporal Multiway Data. In *2025 33rd European Signal Processing Conference (EUSIPCO)*, 2122–2126. September 2025. doi:10.23919/EUSIPCO63237.2025.11226087.

[CMBD16] T. Cheng, M. Mauch, E. Benetos, and S. Dixon. An attack/decay model for piano transcription. In *ISMIR 2016-17st International Society for Music Information Retrieval*. 2016.

[CJPT15] Emilie Chouzenoux, Anna Jezierska, Jean-Christophe Pesquet, and Hugues Talbot. A Convex Approach for Image Restoration with Exact Poisson–Gaussian Likelihood. *SIAM Journal on Imaging Sciences*, 8(4):2662–2682, January 2015. doi:10.1137/15M1014395.

[CPR14] Emilie Chouzenoux, Jean-Christophe Pesquet, and Audrey Repetti. Variable Metric Forward–Backward Algorithm for Minimizing the Sum of a Differentiable Function and a Convex Function. *Journal of Optimization Theory and Applications*, 162(1):107–132, July 2014. doi:10.1007/s10957-013-0465-7.

[CPR16] Emilie Chouzenoux, Jean-Christophe Pesquet, and Audrey Repetti. A block coordinate variable metric forward–backward algorithm. *Journal of Global Optimization*, 66(3):457–485, November 2016. doi:10.1007/s10898-016-0405-9.

[Cic11] Andrzej Cichocki. Tensor Decompositions: New Concepts in Brain Data Analysis ? *Journal of the Society of Instrument and Control Engineers*, 50(7):507–516, 2011.

[CDC24] Joris Claude, Raphaël Dumas, and Jérémy E. Cohen. Constrained Tensor Decomposition Reveals Lever Arms Synergies in the Musculoskeletal System. In *2024 32nd European Signal Processing Conference (EUSIPCO)*, 1322–1326. Lyon, France, August 2024. IEEE. doi:10.23919/EUSIPCO63174.2024.10715450.

[CCFC15] J. E. Cohen, R. Cabral-Farias, and P. Comon. Fast decomposition of large nonnegative tensors. *IEEE Signal Processing Letters*, 22(7):862–866, jul 2015. URL: hal-01069069.

[CG19] J. E. Cohen and N. Gillis. Identifiability of complete dictionary learning. *SIAM Journal on Mathematics of Data Science*, 1(3):518–536, 2019.

[Coh22] Jeremy E. Cohen. Dictionary-Based Low-Rank Approximations and the Mixed Sparse Coding Problem. *Frontiers in Applied Mathematics and Statistics*, 8:801650, February 2022. doi:10.3389/fams.2022.801650.

[CB18] Jeremy E. Cohen and Rasmus Bro. Nonnegative PARAFAC2: A Flexible Coupling Approach. In Yannick Deville, Sharon Gannot, Russell Mason, Mark D. Plumbley, and Dominic Ward, editors, *Latent Variable Analysis and Signal Separation*, volume 10891, pages 89–98. Springer International Publishing, Cham, 2018. doi:10.1007/978-3-319-93764-9_9.

[CCG17] Jeremy E. Cohen, Pierre Comon, and Nicolas Gillis. Some Theory on Non-negative Tucker Decomposition. In Petr Tichavský, Massoud Babaie-Zadeh, Olivier J.J. Michel, and Nadège Thirion-Moreau, editors, *Latent Variable Analysis and Signal Separation*, volume 10169, pages 152–161. Springer International Publishing, Cham, 2017. doi:10.1007/978-3-319-53547-0_15.

[CG18a] Jeremy E. Cohen and Nicolas Gillis. Spectral Unmixing With Multiple Dictionaries. *IEEE Geoscience and Remote Sensing Letters*, 15(2):187–191, February 2018. doi:10.1109/LGRS.2017.2779477.

[CL25] Jeremy E. Cohen and Valentin Leplat. Efficient Algorithms for Regularized Nonnegative Scale-Invariant Low-Rank Approximation Models. *SIAM Journal on Mathematics of Data Science*, pages 468–494, June 2025. doi:10.1137/24M1657948.

[CCFR18] Jeremy Emile Cohen, Rodrigo Cabral Farias, and Bertrand Rivet. Curve Registered Coupled Low Rank Factorization. In Yannick Deville, Sharon Gannot, Russell Mason, Mark D. Plumbley, and Dominic Ward, editors, *Latent Variable Analysis and Signal Separation*, volume 10891, pages 36–45. Springer International Publishing, Cham, 2018. doi:10.1007/978-3-319-93764-9_4.

[CCL16] Jeremy Emile Cohen, Pierre Comon, and Xavier Luciani. Correcting inner filter effects, a non multilinear tensor decomposition method. *Chemometrics and Intelligent Laboratory Systems*, 150:29–40, January 2016. doi:10.1016/j.chemolab.2015.11.002.

[CFC16] Jeremy Emile Cohen, Rodrigo Cabral Farias, and Pierre Comon. Joint tensor compression for coupled canonical polyadic decompositions. In *2016 24th European Signal Processing Conference (EUSIPCO)*, 2285–2289. Budapest, Hungary, August 2016. IEEE. doi:10.1109/EUSIPCO.2016.7760656.

[CBC23] Jérémy Cohen, Rasmus Bro, and Pierre Comon. Tensor Decompositions: Principles and Application to Food Sciences. In *Source Separation in Physical-Chemical Sensing*, chapter 6, pages 255–323. John Wiley & Sons, Ltd, 2023. doi:10.1002/9781119137252.ch6.

[Coh20] Jérémy E Cohen. Computing the proximal operator of the 1 induced matrix norm. 2020. arXiv:2005.06804.

[CG18b] Jérémy E. Cohen and Nicolas Gillis. Dictionary-based Tensor Canonical Polyadic Decomposition. *IEEE Transactions on Signal Processing*, 66(7):1876–1889, April 2018. arXiv:1704.00541, doi:10.1109/TSP.2017.2777393.

[CLDA09] Pierre Comon, Xavier Luciani, and André LF De Almeida. Tensor decompositions, alternating least squares and other tales. *Journal of Chemometrics*, 23(7-8):393–405, 2009.

[Cur75] L. J. Curtis. Simple formula for the distortions in a Gaussian representation of a Poisson distribution. *Am. J. Physics*, 1975.

[DVGN23] Alexandra Dache, Arnaud Vandaele, Nicolas Gillis, and Nicolas Nadisic. Exact and Heuristic Methods for Simultaneous Sparse Coding. In *2023 31st European Signal Processing Conference (EUSIPCO)*, 1753–1757. Helsinki, Finland, September 2023. IEEE. doi:10.23919/EUSIPCO58844.2023.10289845.

[Dan19] Cássio Fraga Dantas. *Accelerating Sparse Inverse Problems Using Structured Approximations*. These de Doctorat, Rennes 1, November 2019.

[DLDMV04] L. De Lathauwer, B. De Moor, and J. Vandewalle. Computation of the canonical decomposition by means of a simultaneous generalized schur decomposition. *SIAM journal on Matrix Analysis and Applications*, 26(2):295–327, 2004.

[DLV04] Lieven De Lathauwer and Joos Vandewalle. Dimensionality reduction in higher-order signal processing and rank-($R_1, R_2,…, R_n$) reduction in multilinear algebra. *Linear Algebra and its Applications*, 391:31–55, 2004.

[DG20] A. Degleris and N. Gillis. A provably correct and robust algorithm for convolutive nonnegative matrix factorization. *IEEE Transactions on Signal Processing*, 68:2499–2512, 2020.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data Via the \emph EM Algorithm. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 39(1):1–22, September 1977. doi:10.1111/j.2517-6161.1977.tb01600.x.

[DB12] Marco F Duarte and Richard G Baraniuk. Kronecker compressive sensing. *IEEE Transactions on Image Processing*, 21(2):494–504, 2012.

[DDT+08] Marco F. Duarte, Mark A. Davenport, Dharmpal Takhar, Jason N. Laska, Ting Sun, Kevin F. Kelly, and Richard G. Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2):83–91, March 2008. doi:10.1109/MSP.2007.914730.

[Duc24] Nicolas Ducros. An Introduction to Single-Pixel Imaging. In *Unconventional Optical Imaging for Biology*, pages 229–255. April 2024. doi:10.1002/9781394283996.ch8.

[DCMahieuWilliame25] Nicolas Ducros, Jérémy E. Cohen, and Laurent Mahieu-Williame. Freeform Hadamard imaging: Back to the roots of computational optics. October 2025. submitted.

[EY36] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[Efr11] Bradley Efron. Tweedie's Formula and Selection Bias. *Journal of the American Statistical Association*, 106(496):1602–1614, December 2011. doi:10.1198/jasa.2011.tm11181.

[EBDB10a] Valentin Emiya, Nancy Bertin, Bertrand David, and Roland Badeau. Maps-a piano database for multipitch estimation and automatic transcription of music. 2010.

[EBDB10b] Valentin Emiya, Nancy Bertin, Bertrand David, and Roland Badeau. MAPS-A piano database for multipitch estimation and automatic transcription of music. 2010.

[EHGR20] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. DDSP: Differentiable Digital Signal Processing. January 2020.

[EF99] H. Erdogan and J. A. Fessler. Ordered subsets algorithms for transmission tomography. *Physics in Medicine & Biology*, 44(11):2835, November 1999. doi:10.1088/0031-9155/44/11/311.

[FWFS19] Dylan Fagot, Herwig Wendt, Cedric Févotte, and Paris Smaragdis. Majorization-minimization Algorithms for Convolutive NMF with the Beta-divergence. In *2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8202–8206. Brighton, United Kingdom, May 2019. IEEE. doi:10.1109/ICASSP.2019.8683837.

[FH94] J.A. Fessler and A.O. Hero. Space-alternating generalized expectation-maximization algorithm. *IEEE Transactions on Signal Processing*, 42(10):2664–2677, October 1994. doi:10.1109/78.324732.

[Fes95] Jeffrey A Fessler. Penalized Maximum-Likelihood Image Reconstruction Using Space-Alternating Generalized EM Algorithms. *IEEE Transactions on Image Processing*, 1995.

[FR13] S. Foucart and H. Rauhut. *A mathematical introduction to compressive sensing*. Applied and Numeric Harmonic Analysis, Birkhauser/Springer, 2013.

[FN04] Piotr Fryzlewicz and Guy P Nason. A Haar-Fisz Algorithm for Poisson Intensity Estimation. *Journal of Computational and Graphical Statistics*, 13(3):621–638, September 2004. doi:10.1198/106186004X2697.

[FevotteI11] C. Févotte and J. Idier. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural computation*, 23(9):2421–2456, 2011.

[GASG18] Daniel G. A. Smith and Johnnie Gray. Opt_einsum - A Python package for optimizing contraction order for einsum-like expressions. *Journal of Open Source Software*, 3(26):753, June 2018. doi:10.21105/joss.00753.

[GTC05] Pando Georgiev, Fabian Theis, and Andrzej Cichocki. Sparse component analysis and blind source separation of underdetermined mixtures. *IEEE transactions on neural networks*, 16(4):992–996, 2005.

[GLoveHMorup25] Kazu Ghalamkari, Jesper Løve Hinrich, and Morten Mørup. Non-negative Tensor Low-rank Decompositions Through the Lens of Information Geometry. In *AAAI 25 Workshop on Connecting Low-rank Representations in AI,*. Philadelphia, 2025. UCI Machine Learning Repository. doi:10.24432/C5501T.

[Gil14] N. Gillis. Successive nonnegative projection algorithm for robust nonnegative blind source separation. *SIAM Journal on Imaging Sciences*, 7(2):1420–1450, 2014.

[GG12] N. Gillis and F. Glineur. Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization. *Neural computation*, 24(4):1085–1105, 2012.

[GV14] N. Gillis and S. A. Vavasis. Fast and Robust Recursive Algorithmsfor Separable Nonnegative Matrix Factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):698–714, 2014.

[Gil20] Nicolas Gillis. *Nonnegative Matrix Factorization*. Data Science. Society for Industrial and Applied Mathematics, January 2020. doi:10.1137/1.9781611976410.

[GG14] Nicolas Gillis and François Glineur. A continuous characterization of the maximum-edge biclique problem. *Journal of Global Optimization*, 58(3):439–464, March 2014. doi:10.1007/s10898-013-0053-2.

[GAM11] Azadeh Golshan, Hamid Abdollahi, and Marcel Maeder. Resolution of Rotational Ambiguity for Three-Component Systems. *Analytical Chemistry*, 83(3):836–841, February 2011. doi:10.1021/ac102429q.

[GVL89] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. The John Hopkins University Press, 1989.

[Gre90] Peter J. Green. On Use of the Em Algorithm for Penalized Likelihood Estimation. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 52(3):443–452, July 1990. doi:10.1111/j.2517-6161.1990.tb01798.x.

[GJB15] Rémi Gribonval, Rodolphe Jenatton, and Francis Bach. Sparse and spurious: dictionary learning with noise and outliers. August 2015. arXiv:1407.5155.

[GS00] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear Gauss Seidel method under convex constraints. *Operations Research Letters*, 26:127–136, 2000.

[Hac12] W. Hackbusch. *Tensor Spaces and Numerical Tensor Calculus*. Series in Computational Mathematics. Springer, Berlin, Heidelberg, 2012. ISBN 978-3-642-28026-9.

[HCD24] Séréna Hariga, Jérémy E Cohen, and Nicolas Ducros. Joint Reconstruction and Spectral Unmixing from Single-Pixel Acquisitions. In *EUSIPCO 2024 - 24th European Signal Processing Conference*. Lyon, France, August 2024.

[HDRC25] Séréna Hariga, Nicolas Ducros, Audrey Repetti, and Jérémy Cohen. Approche plug-and-play pour la reconstruction des cartes d'abondance en imagerie hyperspectrale mono-pixel. In *GRETSI*. Strasbourg, August 2025.

[Har70] R. A. Harshman. Foundations of the PARAFAC procedure: models and conditions for an "explanatory" multimodal factor analysis. *UCLA working papers in phonetics*, 1970.

[HHL03] R. A. Harshman, S. Hong, and M. E. Lundy. Shifted factor analysis—Part I: Models and properties. *Journal of chemometrics*, 17(7):363–378, 2003.

[Har72] Richard A Harshman. PARAFAC2: mathematical and technical notes. *UCLA working papers in phonetics*, 22(3044):122215, 1972.

[HS79] Martin Harwit and Neil J. A. Sloane. *Hadamard Transform Optics*. Academic Press, 1979. ISBN 978-0-12-330050-8 0-12-330050-9.

[HES+18] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck. Onsets and frames: dual-objective piano transcription. *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018.

[HG21] Le Thi Khanh Hien and Nicolas Gillis. Algorithms for Nonnegative Matrix Factorization with the Kullback-Leibler Divergence. *Journal of Scientific Computing*, 87(3):93, June 2021. doi:10.1007/s10915-021-01504-0.

[HLG25] Le Thi Khanh Hien, Valentin Leplat, and Nicolas Gillis. Block Majorization Minimization with Extrapolation and Application to \(\textbraceleft \beta \textbraceright \)-NMF. *SIAM Journal on Mathematics of Data Science*, pages 1292–1314, September 2025. doi:10.1137/24M1660188.

[HPG23] Le Thi Khanh Hien, Duy Nhat Phan, and Nicolas Gillis. An Inertial Block Majorization Minimization Framework for Nonsmooth Nonconvex Optimization. *Journal of Machine Learning Research*, 24(18):1–41, 2023.

[HL13] Christopher J. Hillar and Lek-Heng Lim. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, 2013.

[Hit27] Frank L. Hitchcock. The Expression of a Tensor or a Polyadic as a Sum of Products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927. doi:10.1002/sapm192761164.

[HVD08] Ngoc-Diep Ho and Paul Van Dooren. Non-negative matrix factorization with fixed row and column sums. *Linear Algebra and its Applications*, 429(5):1020–1025, September 2008. doi:10.1016/j.laa.2007.02.026.

[HWRL17] Mingyi Hong, Xiangfeng Wang, Meisam Razaviyayn, and Zhi-Quan Luo. Iteration complexity analysis of block coordinate descent methods. *Mathematical Programming*, 163(1):85–114, May 2017. doi:10.1007/s10107-016-1057-8.

[HS26] John Hood and Aaron Schein. Near-Universal Multiplicative Updates for Nonnegative Einsum Factorization. February 2026. doi:10.48550/arXiv.2602.02759.

[Hot92] Harold Hotelling. Relations between two sets of variates. In *Breakthroughs in statistics: methodology and distribution*, pages 162–190. Springer, 1992.

[HSW+21] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*. October 2021.

[HSL16a] K. Huang, N. D. Sidiropoulos, and A. P. Liavas. A flexible and efficient algorithmic framework for constrained matrix and tensor factorization. *IEEE Transactions on Signal Processing*, 64(19):5052–5065, 2016.

[HS17] Kejun Huang and Nicholas D. Sidiropoulos. Kullback-Leibler principal component for tensors is not NP-hard. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*, 693–697. October 2017. doi:10.1109/ACSSC.2017.8335432.

[HSL16b] Kejun Huang, Nicholas D. Sidiropoulos, and Athanasios P. Liavas. A Flexible and Efficient Algorithmic Framework for Constrained Matrix and Tensor Factorization. *IEEE Transactions on Signal Processing*, 64(19):5052–5065, October 2016. arXiv:1506.04209, doi:10.1109/TSP.2016.2576427.

[HKLP23] Samuel Hurault, Ulugbek Kamilov, Arthur Leclaire, and Nicolas Papadakis. Convergent Bregman Plug-and-Play Image Restoration for Poisson Inverse Problems. *Advances in Neural Information Processing Systems*, 36:27251–27280, December 2023.

[KNS16] Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear Convergence of Gradient and Proximal-Gradient Methods Under the Polyak-\L\textbraceleft \textbraceright ojasiewicz Condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Cham: Springer International Publishing, 2016.

[KUccar16] Oguz Kaya and Bora Uçar. High Performance Parallel Algorithms for the Tucker Decomposition of Sparse Tensors. In *2016 45th International Conference on Parallel Processing (ICPP)*, 103–112. August 2016. doi:10.1109/ICPP.2016.19.

[KCC24] Christophe Kervazo, Abdelkhalak Chetoui, and Jérémy E. Cohen. Deep unrolling of the multiplicative updates algorithm for blind source separation, with application to hyperspectral unmixing. In *EUSIPCO 2024 - 24th European Signal Processing Conference*. Lyon, France, August 2024.

[KC25] Christophe Kervazo and Jérémy Cohen. Mises à jour multiplicatives dépliées pour la Factorisation en Matrices Non-négatives appliquée au démélange spectral. In *GRETSI*. Strasbourg, 2025.

[KGD21] Christophe Kervazo, Nicolas Gillis, and Nicolas Dobigeon. Provably Robust Blind Source Separation of Linear-Quadratic Near-Separable Mixtures. *SIAM Journal on Imaging Sciences*, 14(4):1848–1889, January 2021. doi:10.1137/20M1382878.

[KTBB99] Henk AL Kiers, Jos MF Ten Berge, and Rasmus Bro. PARAFAC2-Part I. A direct fitting algorithm for the PARAFAC2 model. *Journal of Chemometrics*, 13(3-4):275–294, 1999.

[KB09] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, sep 2009.

[KLK+20] Jean Kossaifi, Zachary C Lipton, Arinbjorn Kolbeinsson, Aran Khanna, and Tommaso Furlanello. Tensor Regression Networks. *Journal of Machine Learning Research (JMLR)*, 21:1–21, 2020.

[KLB+93] F.A. Kruse, A.B. Lefkoff, J.W. Boardman, K.B. Heidebrecht, A.T. Shapiro, P.J. Barloon, and A.F.H. Goetz. The spectral image processing system (SIPS)—interactive visualization and analysis of imaging spectrometer data. *Remote Sensing of Environment*, 44(2-3):145–163, May 1993. doi:10.1016/0034-4257(93)90013-N.

[Kru77] Joseph B Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear algebra and its applications*, 18(2):95–138, 1977.

[Kuh55] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97, 1955.

[KKS21] Frederik Kunstner, Raunak Kumar, and Mark Schmidt. Homeomorphic-Invariance of EM: Non-Asymptotic Convergence in KL Divergence for Exponential Families via Mirror Descent. In *Conference on Artificial Intelligence and Statistics*. PMLR, 2021. doi:10.48550/arXiv.2011.01170.

[LJ18] D. Lahat and C. Jutten. A new link between joint blind source separation using second order statistics and the canonical polyadic decomposition. In *14th Int. Conf. on Latent Variable Analysis and Signal Separation (LVA-ICA)*. Univ. of Surrey, Guildford, UK, jul# 2-6 2018. Springer.

[LMV00] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Ana. Appl.*, 21(4):1253–1278, apr 2000.

[LH74] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems*. Prentice Hall, 1974.

[LRHW15] Jonathan Le Roux, John R. Hershey, and Felix Weninger. Deep NMF for speech separation. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. South Brisbane, Queensland, Australia, April 2015. IEEE. doi:10.1109/icassp.2015.7177933.

[LS99] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[LGI21] Valentin Leplat, Nicolas Gillis, and Jérôme Idier. Multiplicative Updates for NMF with \$\beta\$-Divergences under Disjoint Equality Constraints. *SIAM Journal on Matrix Analysis and Applications*, 42(2):730–752, January 2021. doi:10.1137/20M1377278.

[LCU26] Damien Lesens, Jéremy E. Cohen, and Bora Uçar. Algorithms for symmetric Birkhoff-Von Neumann decomposition of symmetric doubly stochastic matrices. *SIAM Journal on Matrix Analysis and Applications*, 2026. doi:10.48550/arXiv.2010.05984.

[LCUccar24] Damien Lesens, Jérémy E. Cohen, and Bora Uçar. Orthogonal Matching Pursuit-Based Algorithms for the Birkhoff-Von Neumann Decomposition. In *2024 32nd European Signal Processing Conference (EUSIPCO)*, 2457–2461. Lyon, France, August 2024. IEEE. doi:10.23919/EUSIPCO63174.2024.10715087.

[LBP+15] Jiajia Li, Casey Battaglino, Ioakeim Perros, Jimeng Sun, and Richard Vuduc. An input-adaptive and in-place approach to dense tensor-times-matrix multiply. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 1–12. Austin Texas, November 2015. ACM. doi:10.1145/2807591.2807671.

[LZFZ22] Geyu Liang, Gavin Zhang, Salar Fattahi, and Richard Y. Zhang. Simple Alternating Minimization Provably Solves Complete Dictionary Learning. October 2022. arXiv:2210.12816, doi:10.48550/arXiv.2210.12816.

[LAWK21] Jiaming Liu, Salman Asif, Brendt Wohlberg, and Ulugbek Kamilov. Recovery Analysis for Plug-and-Play Priors using the Restricted Eigenvalue Condition. In *Advances in Neural Information Processing Systems*, volume 34, 5921–5933. Curran Associates, Inc., 2021.

[LL18] Eric F. Lock and Gen Li. Supervised multiway factorization. *Electronic Journal of Statistics*, January 2018. doi:10.1214/18-EJS1421.

[LMPD22] Antonio Lorente Mur, Françoise Peyrin, and Nicolas Ducros. Deep expectation-maximization for single-pixel image reconstruction with signal-dependent noise. *IEEE Transactions on Computational Imaging*, August 2022. doi:10.1109/TCI.2022.3200841.

[Luc74] L. B. Lucy. An iterative technique for the rectification of observed distributions. *Astronomical Journal*, 79(6):745–754, 1974.

[MG15] L. Le Magoarou and R. Gribonval. Chasing butterflies: In search of efficient dictionaries. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3287–3291. April 2015. doi:10.1109/ICASSP.2015.7178579.

[MBP11] J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. *IEEE transactions on pattern analysis and machine intelligence*, 34(4):791–804, 2011.

[MF11a] Markku Makitalo and Alessandro Foi. A Closed-Form Approximation of the Exact Unbiased Inverse of the Anscombe Variance-Stabilizing Transformation. *IEEE Transactions on Image Processing*, 20(9):2697–2698, September 2011. doi:10.1109/TIP.2011.2121085.

[MF11b] Markku Makitalo and Alessandro Foi. Optimal Inversion of the Anscombe Transformation in Low-Count Poisson Image Denoising. *IEEE Transactions on Image Processing*, 20(1):99–109, January 2011. doi:10.1109/TIP.2010.2056693.

[MGFevotte23] Arthur Marmin, José Henrique de Morais Goulart, and Cédric Févotte. Majorization-Minimization for Sparse Nonnegative Matrix Factorization With the \$\beta\$-Divergence. *IEEE Transactions on Signal Processing*, 71:1435–1447, January 2023. doi:10.1109/TSP.2023.3266939.

[MHenriquedMGoulartFevotte23] Arthur Marmin, José Henrique de Morais Goulart, and Cédric Févotte. Joint majorization-Minimization for nonnegative matrix factorization with the β-divergence. *Signal Processing*, 209:109048, August 2023. doi:10.1016/j.sigpro.2023.109048.

[MCBB20] A. Marmoret, J. E. Cohen, N. Bertin, and F. Bimbot. Uncovering audio patterns in music with nonnegative Tucker decomposition for structural segmentation. In *ISMIR 2020-21st International Society for Music Information Retrieval*. 2020.

[Mar22] Axel Marmoret. *Unsupervised Machine Learning Paradigms for the Representation of Music Similarity and Structure*. Theses, Université de Rennes, December 2022.

[MVL+22] Axel Marmoret, Florian Voorwinden, Valentin Leplat, Jérémy E. Cohen, and Frédéric Bimbot. Nonnegative Tucker Decomposition with Beta-divergence for Music Structure Analysis of Audio Signals. In *GRETSI*. Grenoble, 2022. doi:10.48550/arXiv.2110.14434.

[MYDS20] D. Mitchell, N. Ye, and H. De Sterck. Nesterov acceleration of alternating least squares for canonical tensor decomposition: momentum step size selection and restart mechanisms. *Numerical Linear Algebra with Applications*, 27(4):e2297, 2020.

[MEBM25] Thibaut Modrzyk, Ane Etxebeste, Élie Bretin, and Voichita Maxim. A convergent Plug-and-Play Majorization-Minimization algorithm for Poisson inverse problems. 2025. hal-04968919v1.

[Moh19] Martin J. Mohlenkamp. The Dynamics of Swamps in the Canonical Tensor Approximation Problem. *SIAM Journal on Applied Dynamical Systems*, 18(3):1293–1333, January 2019. doi:10.1137/18M1181389.

[MMG+22] Thomas Moreau, Mathurin Massias, Alexandre Gramfort, Pierre Ablin, Pierre-Antoine Bannier, Benjamin Charlier, Mathieu Dagréou, Ghislain Durif, Cassio F Dantas, Quentin Klopfenstein, Johan Larsson, En Lai, Tanguy Lefort, Benoit Malézieux, Badr Moufad, Binh T Nguyen, Alain Rakotomamonjy, Zaccharie Ramzi, Joseph Salmon, and Samuel Vaiter. Benchopt: Reproducible, efficient and collaborative optimization benchmarks. In *NeurIPS*. 2022.

[MSW25] Nathanaël Munier, Emmanuel Soubies, and Pierre Weiss. Jackpot: Approximating Uncertainty Domains with Adversarial Manifolds. *Journal of Machine Learning Research*, 26(251):1–41, 2025.

[MorupHA08] Morten Mørup, Lars Kai Hansen, and Sidse M Arnfred. Algorithms for sparse nonnegative Tucker decompositions. *Neural computation*, 20(8):2112–2131, 2008.

[NCVG22] Nicolas Nadisic, Jeremy E Cohen, Arnaud Vandaele, and Nicolas Gillis. Matrix-wise $\ell 0$-constrained sparse nonnegative least squares. *Machine Learning*, 2022. doi:10.1007/s10994-022-06260-2.

[NVCG21] Nicolas Nadisic, Arnaud Vandaele, Jeremy E. Cohen, and Nicolas Gillis. Sparse Separable Nonnegative Matrix Factorization. In *Machine Learning and Knowledge Discovery in Databases*, volume 12457, pages 335–350. Springer International Publishing, 2021. doi:10.1007/978-3-030-67658-2_20.

[NVGC20] Nicolas Nadisic, Arnaud Vandaele, Nicolas Gillis, and Jeremy E. Cohen. Exact Sparse Nonnegative Least Squares. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 5395–5399. Barcelona, Spain, May 2020. IEEE. doi:10.1109/ICASSP40776.2020.9053295.

[NES22] Rami Nasser, Yonina C. Eldar, and Roded Sharan. Deep Unfolding for Non-Negative Matrix Factorization with Application to Mutational Signature Analysis. *Journal of Computational Biology: A Journal of Computational Molecular Cell Biology*, 29(1):45–55, January 2022. doi:10.1089/cmb.2021.0438.

[NF70] E. D. Nelson and M. L. Fredman. Hadamard Spectroscopy. *Journal of the Optical Society of America*, 60(12):1664, December 1970. doi:10.1364/JOSA.60.001664.

[Nes83] Y. Nesterov. A method of solving a convex programming problem with convergence rate o (1/k2). In *Soviet Mathematics Doklady*, volume 27, 372–376. 1983.

[NS18] Klaus Neymeyr and Mathias Sawall. On the Set of Solutions of the Nonnegative Matrix Factorization Problem. *SIAM Journal on Matrix Analysis and Applications*, 39(2):1049–1069, January 2018. doi:10.1137/17M1118439.

[NSID17] T.T. Nguyen, C. Soussen, J. Idier, and E-H. Djermoune. An optimized version of non-negative OMP. In *XXVIe Colloque GRETSI*. 2017.

[Paa97] P. Paatero. A weighted non-negative least squares algorithm for three-way 'parafac'factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 38(2):223–242, 1997.

[PLNG25] Junjun Pan, Valentin Leplat, Michael Ng, and Nicolas Gillis. A Provably-Correct and Robust Convex Model for Smooth Separable NMF. November 2025. doi:10.48550/arXiv.2511.07109.

[PSB13] E. E. Papalexakis, N. D. Sidiropoulos, and R. Bro. From k-means to higher-way co-clustering: multilinear decomposition with sparse latent factors. *IEEE transactions on signal processing*, 61(2):493–506, 2013.

[PFS16] Evangelos E. Papalexakis, Christos Faloutsos, and Nicholas D. Sidiropoulos. Tensors for data mining and data fusion: models, applications, and scalable algorithms. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):16, 2016.

[PRK93] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In *1993 Conference Record of The 27th Asilomar Conference on Signals, Systems and Computers*, 40–44 vol.1. 1993. doi:10.1109/ACSSC.1993.342465.

[PCC25] Mai-Quyen Pham, Jérémy Cohen, and Thierry Chonavel. A Second-Order Majorant Algorithm for Nonnegative Matrix Factorization. June 2025. Under Review. arXiv:2303.17992.

[PS16] T. Pock and S. Sabach. Inertial proximal alternating linearized minimization (iPALM) for nonconvex and nonsmooth problems. *SIAM Journal on Imaging Sciences*, 9(4):1756–1787, 2016.

[Pow73] M. J. D. Powell. On search directions for minimization algorithms. *Mathematical Programming*, 4(1):193–201, December 1973. doi:10.1007/BF01584660.

[PKLB22] Christos Psarras, Lars Karlsson, Jiajia Li, and Paolo Bientinesi. The landscape of software for tensor computations. June 2022.

[RCH08] Myriam Rajih, Pierre Comon, and Richard A. Harshman. Enhanced Line Search: A Novel Method to Accelerate PARAFAC. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1128–1147, January 2008. doi:10.1137/06065577.

[RHL13] Meisam Razaviyayn, Mingyi Hong, and Zhi-Quan Luo. A Unified Convergence Analysis of Block Successive Minimization Methods for Nonsmooth Optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.

[Ric72] William Hadley Richardson. Bayesian-Based Iterative Method of Image Restoration*. *JOSA*, 62(1):55–59, January 1972. doi:10.1364/JOSA.62.000055.

[RC16] Bertrand Rivet and Jeremy E. Cohen. Modeling time warping in tensor decomposition. In *2016 IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*, 1–5. Rio de Janerio, Brazil, July 2016. IEEE. doi:10.1109/SAM.2016.7569733.

[RDGuerinDugue+15] Bertrand Rivet, Marc Duda, Anne Guérin-Dugué, Christian Jutten, and Pierre Comon. Multimodal approach to estimate the ocular movements during EEG recordings: A coupled tensor factorization method. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 6983–6986. August 2015. doi:10.1109/EMBC.2015.7319999.

[Roa23] Marie Roald. MatCoupLy: Learning coupled matrix factorizations with Python. *SoftwareX*, 21:101292, February 2023. doi:10.1016/j.softx.2022.101292.

[RSC+22] Marie Roald, Carla Schenker, Vince D. Calhoun, Tülay Adalı, Rasmus Bro, Jeremy E. Cohen, and Evrim Acar. An AO-ADMM approach to constraining PARAFAC2 on all modes. *SIAM Journal on Mathematics of Data Science*, 4(3):1191–1222, September 2022. arXiv:2110.01278, doi:10.1137/21M1450033.

[RSCA21] Marie Roald, Carla Schenker, Jeremy E. Cohen, and Evrim Acar. PARAFAC2 AO-ADMM: Constraints in all modes. February 2021. arXiv:2102.02087.

[SBG25] Subhayan Saha, Giovanni Barbarino, and Nicolas Gillis. Identifiability of Nonnegative Tucker Decompositions – Part I: Theory. May 2025. doi:10.48550/arXiv.2505.12713.

[SAA+22] Mathias Sawall, Tomass Andersons, Hamid Abdollahi, Somaiyeh Khodadadi Karimvand, Bahram Hemmateenejad, and Klaus Neymeyr. Calculation of lower and upper band boundaries for the feasible solutions of rank-deficient multivariate curve resolution problems. *Chemometrics and Intelligent Laboratory Systems*, 226:104577, July 2022. doi:10.1016/j.chemolab.2022.104577.

[SAW+26] Mathias Sawall, Tomass Andersons, Chunhong Wei, Christoph Kubis, and Klaus Neymeyr. Sampling-based computation of the sets of feasible solutions and feasible bands for noisy data. *Chemom. Intell. Lab. Syst. 268, 105565*, 2026. doi:10.1016/j.chemolab.2025.105565.

[SN14] Mathias Sawall and Klaus Neymeyr. A fast polygon inflation algorithm to compute the area of feasible solutions for three-component systems. II: Theoretical foundation, inverse polygon inflation, and FAC-PACK implementation. *Journal of Chemometrics*, 28(8):633–644, 2014. doi:10.1002/cem.2612.

[SCA21a] Carla Schenker, Jeremy E. Cohen, and Evrim Acar. A Flexible Optimization Framework for Regularized Matrix-Tensor Factorizations With Linear Couplings. *IEEE Journal of Selected Topics in Signal Processing*, 15(3):506–521, April 2021. doi:10.1109/JSTSP.2020.3045848.

[SCA21b] Carla Schenker, Jeremy E. Cohen, and Evrim Acar. An Optimization Framework for Regularized Linearly Coupled Matrix-Tensor Factorization. In *2020 28th European Signal Processing Conference (EUSIPCO)*, 985–989. Amsterdam, Netherlands, January 2021. IEEE. doi:10.23919/Eusipco47968.2020.9287459.

[Sch75] Laurent Schwartz. *Les Tenseurs - Suivi de Torseurs Sur Un Espace Affine*. Hermann, 1975.

[SSvD+23] Jacob Seifert, Yifeng Shao, Rens van Dam, Dorian Bouchet, Tristan van Leeuwen, and Allard P. Mosk. Maximum-likelihood estimation in ptychography in the presence of Poisson-Gaussian noise statistics. *Optics Letters*, 48(22):6027, November 2023. doi:10.1364/OL.502344.

[SLSL07] Fei Sha, Yuanqing Lin, Lawrence K. Saul, and Daniel D. Lee. Multiplicative Updates for Nonnegative Quadratic Programming. *Neural Computation*, 19(8):2004–2031, August 2007. doi:10.1162/neco.2007.19.8.2004.

[SV82] Lawrence A. Shepp and Yehuda Vardi. Maximum-likelihood-reconstruction-emission-tomography. *IEEE Transactions on Medical Imaging*, 1(2):113–122, 1982.

[SBG00] N. D. Sidiropoulos, R. Bro, and G. B. Giannakis. Parallel factor analysis in sensor array processing. *IEEE Trans. Sig. Proc.*, 48(8):2377–2388, aug 2000.

[Sin18] Amit Singer. Mathematics for cryo-electron microscopy. March 2018. doi:10.48550/arXiv.1803.06714.

[SK67] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, May 1967. doi:10.2140/pjm.1967.21.343.

[SB03] P. Smaragdis and J. C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 177–180. IEEE, 2003.

[Sma06] Paris Smaragdis. Convolutive speech bases and their application to supervised speech separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1):1–12, 2006.

[SG18] Jordan B. L. Smith and Masataka Goto. Nonnegative Tensor Factorization for Source Separation of Loops in Audio. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 171–175. April 2018. doi:10.1109/ICASSP.2018.8461876.

[SK15] Shaden Smith and George Karypis. Tensor-matrix products with a compressed sparse tensor. In *Proceedings of the 5th Workshop on Irregular Applications: Architectures and Algorithms*, 1–7. Austin Texas, November 2015. ACM. doi:10.1145/2833179.2833183.

[SDL15] M. Sorensen and L. De Lathauwer. Coupled canonical polyadic decompositions and (coupled) decompositions in multilinear rank-(lr,n, lr,n, 1) terms — part i: uniqueness. *SIAM J. Matrix Anal. Appl.*, 36(2):496–522, may 2015.

[SBS14] Pablo Sprechmann, Alex M. Bronstein, and Guillermo Sapiro. Supervised non-euclidean sparse NMF via bilevel optimization with applications to speech enhancement. In *2014 4th Joint Workshop on Hands-free Speech Communication and Microphone Arrays (HSCMA)*, 11–15. Villers-les-Nancy, France, May 2014. IEEE. doi:10.1109/HSCMA.2014.6843241.

[SS07] A. Stegeman and N. Sidiropoulos. On Kruskal's uniqueness condition for the CP decomposition. *Linear Algebra and its Applications*, 420(2-3):540–552, January 2007.

[SBP17] Ying Sun, Prabhu Babu, and Daniel P. Palomar. Majorization-Minimization Algorithms in Signal Processing, Communications, and Machine Learning. *IEEE Transactions on Signal Processing*, 65(3):794–816, February 2017. doi:10.1109/TSP.2016.2601299.

[TH14] Norikazu Takahashi and Ryota Hibi. Global convergence of modified multiplicative updates for nonnegative matrix factorization. *Computational Optimization and Applications*, 57(2):417–440, March 2014. doi:10.1007/s10589-013-9593-0.

[TTI25] Shota Takahashi, Mirai Tanaka, and Shiro Ikeda. Majorization-Minimization Bregman Proximal Gradient Algorithms for NMF with the Kullback–Leibler Divergence. *Journal of Optimization Theory and Applications*, 208(1):14, September 2025. doi:10.1007/s10957-025-02833-y.

[TG26] Olivier Vu Thanh and Nicolas Gillis. Maximum-Volume Nonnegative Matrix Factorization. February 2026. doi:10.48550/arXiv.2602.04795.

[Tib13] Ryan J. Tibshirani. The lasso problem and uniqueness. *Electronic Journal of Statistics*, 7(none):1456–1490, January 2013. doi:10.1214/13-EJS815.

[TK02] Marieke E Timmerman and Henk AL Kiers. Three-way component analysis with smoothness constraints. *Computational statistics & data analysis*, 40(3):447–470, 2002.

[Tuc66] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[UT19] Madeleine Udell and Alex Townsend. Why Are Big Data Matrices Approximately Low Rank? *SIAM Journal on Mathematics of Data Science*, 1(1):144–160, January 2019. doi:10.1137/18M1183480.

[Use25] Konstantin Usevich. Approche algébrique pour la décomposition tensorielle ParaTuck-2. In *GRETSI*. 2025.

[Van17] N. Vannieuwenhoven. Condition numbers for the tensor rank decomposition. *Linear Algebra and its Applications*, 535:35–86, 2017.

[Vav10] S. A. Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2010.

[VBW13] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg. Plug-and-play priors for model based reconstruction. In *2013 IEEE Global Conference on Signal and Information Processing*, 945–948. IEEE, 2013.

[VVDL25] Charlotte Vermeylen, Nico Vervliet, and Lieven De Lathauwer. Reducing swamp behavior for the canonical polyadic decomposition problem by rank-1 freezing. *Numerical Algorithms*, June 2025. doi:10.1007/s11075-025-02111-y.

[VBB08] Emmanuel Vincent, Nancy Bertin, and Roland Badeau. Harmonic and inharmonic Nonnegative Matrix Factorization for Polyphonic Pitch transcription. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, 109–112. Las Vegas, NV, March 2008. IEEE. doi:10.1109/ICASSP.2008.4517558.

[WMC22] Haoran Wu, Axel Marmoret, and Jérémy E. Cohen. Semi-Supervised Convolutive NMF for Automatic Piano Transcription. April 2022. arXiv:2202.04989.

[XY13] Yangyang Xu and Wotao Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on imaging sciences*, 6(3):1758–1789, 2013.

[ZRG23] Léon Zheng, Elisa Riccietti, and Rémi Gribonval. Efficient Identification of Butterfly Sparse Matrix Factorizations. *SIAM Journal on Mathematics of Data Science*, 5(1):22–49, March 2023. doi:10.1137/22M1488727.

[BarLevE88] Shaul K. Bar-Lev and Peter Enis. On the classical choice of variance stabilizing transformations and an application for a Poisson variate. *Biometrika*, 75(4):803–804, December 1988. doi:10.1093/biomet/75.4.803.

[ODonoghueC15] Brendan O'Donoghue and Emmanuel Candes. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3):715–732, 2015.

[vanOosten14] C. F. van Oosten. The EM-algorithm for Poisson data. Bachelor Thesis, Mathematical Institute of Leiden University, 2014.