Université de Mons
Faculté Polytechnique

Multilayer and deep matrix factorizations

Pierre DE HANDSCHUTTER

A thesis submitted in partial fulfilment of the requirements for the degree of *Docteur en Sciences de l'Ingénieur et Technologies*

Dissertation committee:

Prof. Nicolas GILLIS (supervisor)	Université de Mons
Prof. Xavier SIEBERT (co-supervisor)	Université de Mons
Prof. Arnaud VANDAELE (chair)	Université de Mons
Prof. Fabian LECRON	Université de Mons
Prof. Xiao FU	Oregon State University
Dr. Jérémy E. COHEN	Chercheur CNRS, CREATIS

Abstract

Matrix factorizations (MF) are standard techniques in linear algebra that approximate a data matrix as the product of two smaller matrices whose inner dimension is called the rank of the factorization. A general assumption in MFs is the nonnegativity of the input matrix which led to the development of nonnegative matrix factorization (NMF) models, where the factors are constrained to be nonnegative as well, motivated by several real-world applications. NMF allows one to express the columns of the input matrix, which generally represent data points, as linear combinations of a small number of latent features.

In the first part of this thesis, we introduce near-convex archetypal analysis (NCAA), a flexible extension of archetypal analysis, a well-known NMF variant, which has an interesting geometric interpretation and performs competitively with minimum-volume (minVol) NMF.

In the second part of this thesis, we study deep MF, that is, the extension of MF to several layers, inspired by the recent advances in deep learning. We conduct a thorough literature review of multilayer and deep MF, focusing on the models, the choice of the parameters, the applications, and the theoretical aspects. We also illustrate the abilities of deep MF to extract hierarchical features within complex data sets on three showcase examples, namely the extraction of facial features, hyperspectral unmixing (HU) and recommender systems.

We then introduce two new loss functions for deep MF together with a general optimization framework. We show their efficiency to tackle sparse and minVol deep MF on both synthetic and real-world data. These loss functions alleviate the drawbacks of the current approaches both in a theoretical and experimental point of view. We also design a new greedy initialization algorithm for deep MF and extend symmetric NMF to the deep case. We apply this latter successfully to the extraction of overlapping communities of symptoms in psychiatric networks, with promising clinical interpretation.

Finally, we sketch perspectives of future works, including the study of the identifiability of deep MF, the investigation of the connexions between deep MF and deep neural networks, and the exploration of new deep MF models.

Personal note and acknowledgements

University has three main missions: research, teaching and societal commitment. I tried to involve humbly in all three during this PhD. Undoubtedly, this PhD has grown me up from several angles: human, scientific,... But undoubtedly also, this PhD was all along a path of pain and doubt.
What I will remember the most are the numerous interactions with many UMONS staff members and students, both for research purposes and "societal" projects to which I paid a lot of attention. I already apologize for all the people whose I could have forgotten the names, but I want to thank:

- *My promoter Nicolas Gillis for his incredible support, for the team spirit that he permanently strives to spark, for his outstanding technical abilities and for his restless advice and encouragement,*
- *My co-promoter Xavier Siebert for his helpful advice and encouragement,*
- *All the other members of my jury for their insightful advice: Arnaud Vandaele, Fabian Lecron, Xiao Fu and Jérémy E. Cohen,*
- *All the members of the COLORAMAP (and beyond) research team: Nicolas Nadisic, François Moutier, Tim Marrinan, Maryam Abdolali, Le Thi Khanh Hien, Junjun Pan, Man Shun Ang "Andersen", Christophe Kervazo, Christos Kolomvakis, Olivier Vu Thanh, Valentin Leplat, Atharva Awari,*
- *All the members (former or current) of the "Informatique et Gestion" department of UMONS with whom I had great collaborations: Sarah Itani, Boris Edgar Ndjia Njike, Amine Larhmam, Mohammed Benjelloun, Philippe Fortemps, Sébastien Bette, Pierre Manneback but also Maxime Gobert, Guillaume Briffoteaux, Gwendolyn Lacroix, Daniel Tuyttens, Maxime Manderlier, Landelin Delcoucq, Saïd Mahmoudi, Sidi Mahmoudi, Olivier Debauche, Adriano Guttadauria, Laurence Wouters,*
- *Other people (former or current) from UMONS for the various inspiring discussions and collaborations: Christine Renotte, Philippe Dubois, Marc Labie, Cynthie Marchal, Céline Thillou, Céliane Jennebauffe, Alexandra Dache, Emilie Genin, Rosabelle Lekemo*

Atonfack, Wivine Blekic, Giovanni Briganti, Laurence Ris, Emerance Delacenserie, Anne-Emilie Declèves, Anne-Emmanuelle Bourgaux, Maxime Duménil, Nicolas Dupont, Valentine Fays, Julie Walaszczyk, Thierry Dutoit, Thomas Brihaye, Véronique Vitry, Véronique Van Renterghem, Erika Wauthia, Pietro Favaro, Francesco Lo Bue, Julien Lefevere,

- *People from outside UMONS who worked with me and/or inspired me: Luc Marchal, Nathanaël Ackerman, Marylise Ledouble, Joyce Ann Quinto, Carl Mörch,*
- *And many, many more...*

Of course, I thank my parents for their support along this PhD. I also thank my old good friend Bastien Lemahieu for the journey at UMONS; all the staff from the CETIC research center where I did my master internship, especially Paul Vanabelle and Mohamed Boukhebouze; the FNRS, who funded a part of my PhD; and finally the direction of Institut Saint-André Ramegnies-Chin, Sophie Liagre and Pascal Mol for their understanding, and the colleagues, especially Manon Delhaye, for the advice.
Finally, a special thanks (again) to Nicolas Gillis for proofreading the whole manuscript and to Arnaud Vandaele and Wivine Blekic for proofreading some chapters !

Contents

List of Figures

List of Tables

List of main acronyms

AA	Archetypal analysis
BCD	Block coordinate descent
DSM	Diagnostic and Statistical Manual of Mental Disorders
DSNMF	Deep symmetric nonnegative matrix factorization
FPGD	Fast projected gradient descent
GSP	Grouped sparse projection
GT	Ground truth
HC	Hierarchical clustering
HU	Hyperspectral unmixing
LM	Louvain method
LRMF	Low-rank matrix factorization
MF	Matrix factorization
minVolNMF	Minimum-volume nonnegative matrix factorization
MRSA	Mean-removed spectral angle
NCAA	Near-convex archetypal analysis
NMF	Nonnegative matrix factorization
NNLS	Nonnegative least squares
NSNMF	Non-smooth nonnegative matrix factorization
ONMF	Orthogonal nonnegative matrix factorization
PGD	Projected gradient descent
PTSD	Post-traumatic stress disorder
SNPA	Successive nonnegative projection algorithm
SODA	Successive orthogonal decomposition algorithm
SSNMF	Simplex-structured nonnegative matrix factorization
symNMF	Symmetric nonnegative matrix factorization

CHAPTER 1

INTRODUCTION AND OUTLINE

Modern life has seen an overwhelming explosion of data collection. To extract meaningful information among them, many computational models have emerged over the last decades, known as machine learning techniques. In many applications, a careful tradeoff should be considered between, on one side, accuracy and on the other side, interpretability of the model.

A particularly interesting class of models, intimately intricated with linear algebra, are matrix factorizations (MFs). MFs are unsupervised models in the sense that they do not require labeled training data to be built, contrary to supervised learning models such as deep neural networks, see for example [107] for an overview of the main machine learning models. An additional key advantage of MFs is that they are linear methods hence provide easy interpretation in many cases. Though linearity is often a simplifying hypothesis as most of real systems are essentially non-linear, it is generally an acceptable assumption to describe them.

In fact, MFs perform a dimensionality reduction, that is, they give a representation of the input data, which are generally high-dimensional, in a lower dimension. This dimension is generally small compared to that of the input data, hence these models are referred to as low-rank matrix factorizations (LRMF) [116]. In other words, LRMFs are able to identify a small number of significant features within the dataset and leave away the minor ones.

With the development of deep learning and the need to deal with more and more complex datasets, MF's followed the trend and scaled up. Therefore, over the last few years, extensions of MF's referred to as "Multilayer and deep matrix factorizations" have emerged and constitute the main topic of interest of this PhD thesis.

The remainder of this chapter aims to present the general context of this PhD, describe its main motivations and list the key results. In Section 1.1, we describe the mathematical setting of LRMFs; their geometric intuition is discussed in Section 1.2. Then, in Section 1.3, we present the main real-world applications of LRMFs with a special focus on those which were actually investigated during the PhD. Finally, Section 1.4 summarizes the main contributions of this PhD, including scientific publications, and gives a detailed outline of the remainder of this document.

1.1 Matrix factorizations, from low-rank to nonnegative

Let us assume that we have a set of n data points, each one represented by a vector in an m-dimensional space. We build a data matrix $X \in \mathbb{R}^{m\times n}$ such that each column of X corresponds to one of the n data points. The goal of a low-rank matrix factorization is to find two matrices $W \in \mathbb{R}^{m\times r}$ and $H \in \mathbb{R}^{r\times n}$ such that the product WH approximates X as well as possible, that is, $X \approx WH$. We define the inner dimension r as the rank of the factorization, which is in most cases fixed before solving the problem. As the factorizations are usually low-rank, to only capture a small number of features out of the data, r is generally small, that is, $r \ll \min(m,n)$.

Each column of the matrix W corresponds to an m-dimensional vector called a *basis vector*. Consequently, the approximation $X \approx WH$ consists in approaching each column of X, that is, each data point, as a linear combination of the r columns of W. In other words, we have for all $j = 1,\dots,n$

$$X(:,j) \approx \sum_{k=1}^{r} W(:,k)H(k,j), \tag{1.1}$$

where the coefficients $H(k,j)$'s are the weights of the linear combination of basis vectors that approximates the j-th data point $X(:,j)$.

To evaluate the quality of the factorization, that is, the approximation of X by WH, a loss function (also called objective function) needs to be defined to evaluate the "gap" between the original matrix X and its low-rank approximation WH. One of the most natural loss function, frequently used in the literature, is the squared Frobenius norm of the residual matrix, equal to the difference between X and WH. The "best" factor matrices W and H are those that minimize this loss function, also called the reconstruction error, which implies to solve an optimization problem. The LRMF problem with Frobenius norm can therefore be written as:

$$\min_{\substack{W\in\mathbb{R}^{m\times r}\\ H\in\mathbb{R}^{r\times n}}} \frac{1}{2}\|X - WH\|_F^2. \tag{1.2}$$

In practice, real-world data matrices often correspond to nonnegative measurements hence it makes sense to consider that $X \geq 0$, which means that each entry of X is nonnegative. It makes sense to approximate such a matrix as the product of two matrices W and H that are also element-wise nonnegative, that is $W \geq 0$ and $H \geq 0$. The problem of computing such matrices W and H has been studied a lot since it was introduced in the seminal papers of Paatero and Tapper [97] and later on, Lee and Seung [75], and is known as nonnegative matrix factorization (NMF). Hence, NMF with squared Frobenius norm consists in solving

$$\min_{\substack{W\in\mathbb{R}^{m\times r}\\ H\in\mathbb{R}^{r\times n}}} \frac{1}{2}\|X - WH\|_F^2 \quad \text{such that } W \geq 0 \text{ and } H \geq 0 \tag{1.3}$$

where X is assumed to be nonnegative as well.

Since the r basis vectors, that is, the columns of W, are constrained to be nonnegative, it is generally easy to give them a physical interpretation in real-world settings. Moreover, due to their nonnegativity, the entries of H can also be interpreted interestingly. More precisely, in Eq. (1.3), $H(k, j)$ reflects the proportion with which the k-th basis vector contributes to the approximation of the j-th data point.

Unfortunately, NMF has two main drawbacks [55]:

1. When the rank r is part of the input, solving Problem (1.3) is NP-hard in general, meaning that there does not exist any algorithm able to compute the NMF of a matrix in polynomial time with respect to m, n and r (unless $P = NP$). In practice, one usually chooses the value

of the rank r *a priori*, depending on the application, and then applies an optimization algorithm (typically running in $\mathcal{O}(mnr)$ operations) to solve NMF.

2. NMF does not have a unique solution in most cases. Indeed, let us suppose that we have computed a solution (W,H) of NMF such that $X = WH$. It is easy to see that, given any invertible matrix $Q \in \mathbb{R}^{r\times r}$, $(WQ, Q^{-1}H)$ is also a solution if $WQ \geq 0$ and $Q^{-1}H \geq 0$. To alleviate the non-uniqueness of the NMF solution, additional constraints are enforced on the factors W and H, such as sparsity, orthogonality and many more, or a regularization term is added to the loss function of Eq. (1.3), such as in minimum-volume NMF. We will discuss much more this concern in the next chapters.

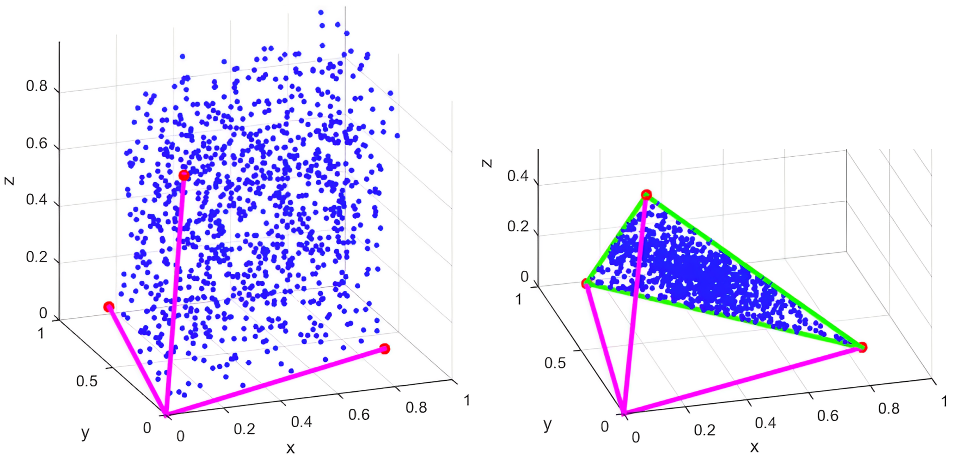
It turns out that NMF has raised a lot of interest among researchers over the last decades in terms of theoretical study, development of new models and algorithms and validation on real-world applications. We warmly encourage the reader to have a look at the comprehensive SIAM book by Nicolas Gillis [56] for more details on NMF.

1.2 Geometric interpretation of NMF

The nonnegativity of the NMF factors offers an interesting geometric interpretation. Let us recall that the conical hull of k data points $\boldsymbol{y}_1, \dots, \boldsymbol{y}_k$ is the set of all conical combinations of these points, that is, $\left\{z = \sum_{i=1}^{k} \alpha_i \boldsymbol{y}_i \text{ such that } \alpha_i \geq 0 \text{ for all } i\right\}$. Hence, regarding Problem (1.1), NMF aims to find r nonnegative basis vectors such that the n data points, that is, the n columns of X, lie in their conical hull, as shown on Fig. 1.1a for $m = 3$ and $r = 3$.

With the sole nonnegativity constraints on W and H, a scaling ambiguity appears in the NMF solution. Indeed, let us assume that (W,H) is a solution of Problem (1.3). Then, for any $k = 1, \dots, r$, multiplying the k-th column of W by some $\alpha_k > 0$ and the k-th row of H by $\frac{1}{\alpha_k}$ leads to the same reconstructed matrix. More precisely, for each k,

$$W(:,k)H(k,:) = (\alpha_k W(:,k))\left(\frac{H(k,:)}{\alpha_k}\right),$$



(a) NMF (b) SSNMF

Figure 1.1: Geometric interpretation of (a) NMF and (b) SSNMF with $m = 3$ and $r = 3$. The data points (blue dots) are (a) in the conical hull and (b) in the convex hull of the basis vectors (red dots). The boundary of this convex hull is the green triangle.

that is, there is a scaling degree of freedom for every rank-one factor of the decomposition. Hence, to alleviate this degree of freedom, one can, without loss of generality, normalize the columns of X by dividing each entry of X by the sum of the elements of its column. By doing so, the elements of each column of X sum to one, that is, $X^T e = e$ with e the vector of all ones of appropriate dimension. The columns of W can also be normalized that way, without loss of generality, that is, $W^T e = e$. Consequently, $e = X^T e = (WH)^T e = H^T W^T e = H^T e$, that is, the sum of the elements of each column of H is constrained to be equal to one. Let us recall that the convex hull of k data points $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k$ is the set of all convex combinations of these points, that is, $\left\{ z = \sum_{i=1}^{k} \alpha_i \boldsymbol{y}_i \text{ such that } \alpha_i \geq 0 \text{ for all } i \text{ and } \sum_{i=1}^{k} \alpha_i = 1 \right\}$. With this additional constraint on H (called column-stochasticity), the data points are approximated as convex combinations of the basis vectors, that is, as members of their convex hull.

Hence, the NMF problem with both the nonnegativity constraints and column-stochasticity of H, known as simplex-structured nonnegative matrix factorization (SSNMF), is stated as

$$\min_{W,H} \frac{1}{2}||X - WH||_F^2 \quad \text{s.t.} \quad W, H \geq 0, \quad \sum_{k=1}^{r} H(k,j) = 1 \quad \text{for all } j = 1, ..., n, \tag{1.4}$$

and its geometric interpretation is represented on Fig. 1.1b. The feasible set for each column of H is called, by definition, the $r-1$ standard simplex.

The column-stochasticity of H corresponds to a willingness of physical interpretation: the j-th column of H can be seen as the vector indicating the proportions, or abundances, of each basis vector in the j-th data point, which is meaningful in many applications; see next section.

1.3 Real-world applications of NMF

Given its linear formulation, NMF has been leveraged in a wide range of applications, such as recommender systems [84], community detection [120], topic modeling [7] and many more. Let us already detail two of them, the ones that have been used the most throughout this thesis:

- *Hyperspectral unmixing (HU)*: In this case, the data matrix X is a hyperspectral image, that is, a set of n pixels described by their reflectance (defined as the fraction of the incoming light reflected) in m wavelength bands, referred to as their spectral signature. The goal of NMF is to identify the spectral signature of r materials (also called *endmembers*) present in the image (the columns of W) and the proportions in which each material appears in each pixel, called the abundances (the columns of H), see Fig. 1.2. HU has been widely studied in the literature from the angle of NMF, see [15] and [47] among others.

  Let us describe a well-known hyperspectral image, frequently used as a benchmark for our experiments, namely the HYDICE Urban hyperspectral image [135]. It is an airborne image of the surroundings of a Walmart supermarket in Copperas Cove (Texas). The Urban image is made of 307×307 pixels whose reflectance is measured in 162 wavelength bands (after denoising). Depending on the versions, the number of materials commonly considered as being present in the Urban image varies between 4 and 6. In this last case, the materials are trees, grass, roof, dirt, metal and road, as depicted on Fig. 1.3 where we have

$$\underbrace{X(:,j)}_{\substack{\text{spectral signature}\\ \text{of } j\text{th pixel}}} \approx \sum_{k=1}^{r} \underbrace{W(:,k)}_{\substack{\text{spectral signature}\\ \text{of } k\text{th endmember}}} \underbrace{H(k,j)}_{\substack{\text{abundance of } k\text{th endmember}\\ \text{in } j\text{th pixel}}} .$$

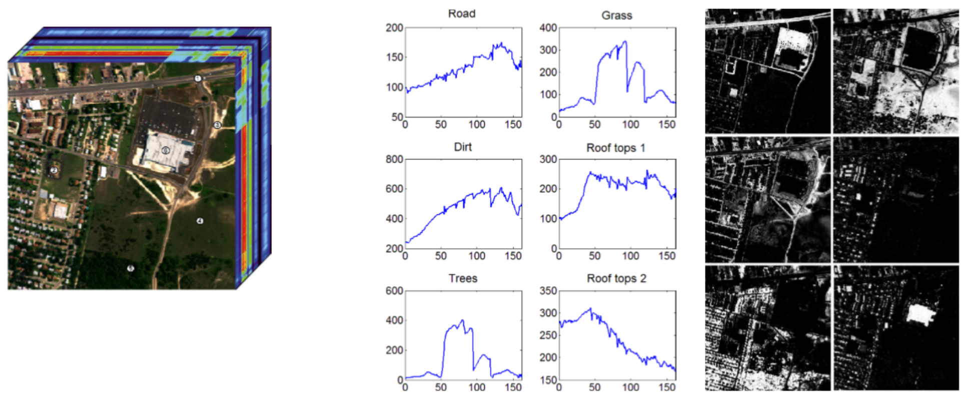


Figure 1.2: Illustration of NMF applied to hyperspectral unmixing. Figure taken from [55].

highlighted areas of pixels containing mostly a given material. When only 4 materials are considered, these are trees, grass, roof and road. On Fig. 1.4, we show the ground-truth spectral signatures of the end-members (according to [135]) in the cases where 4 and 6 materials are

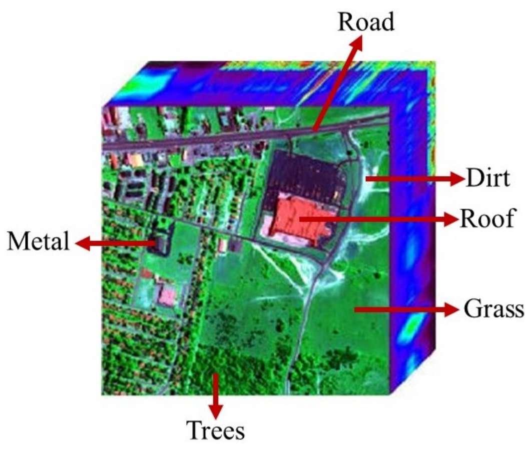
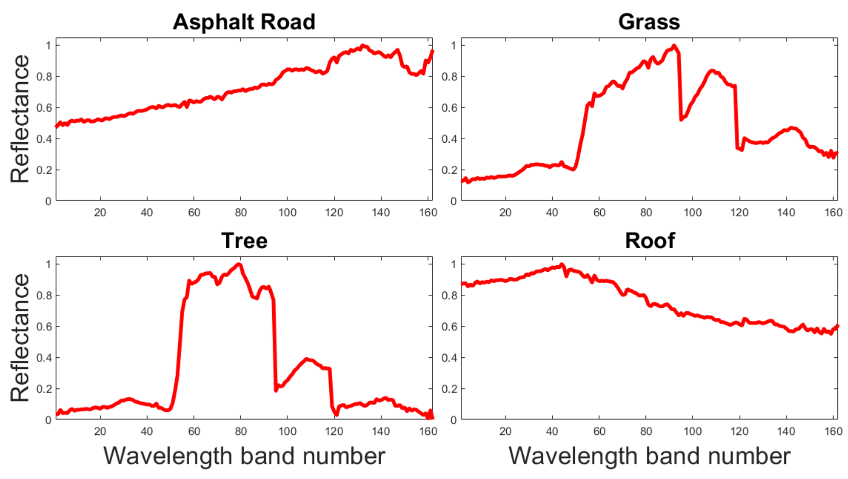
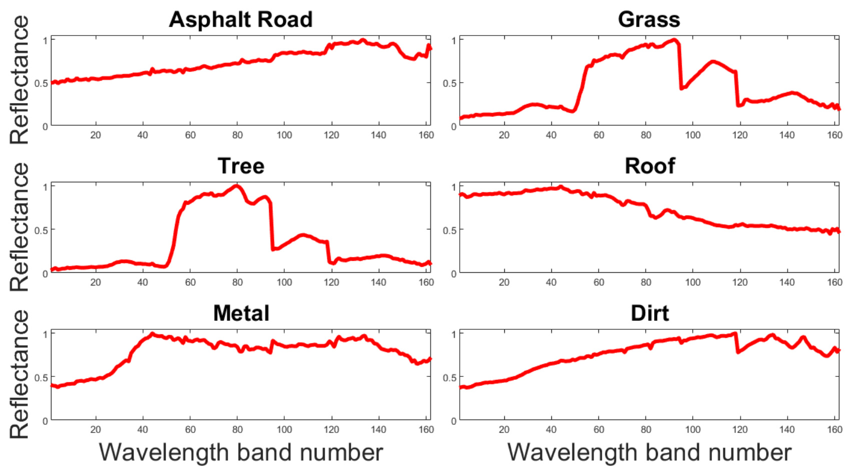


Figure 1.3: The Urban hyperspectral image and its 6 main constitutive materials.



(a)

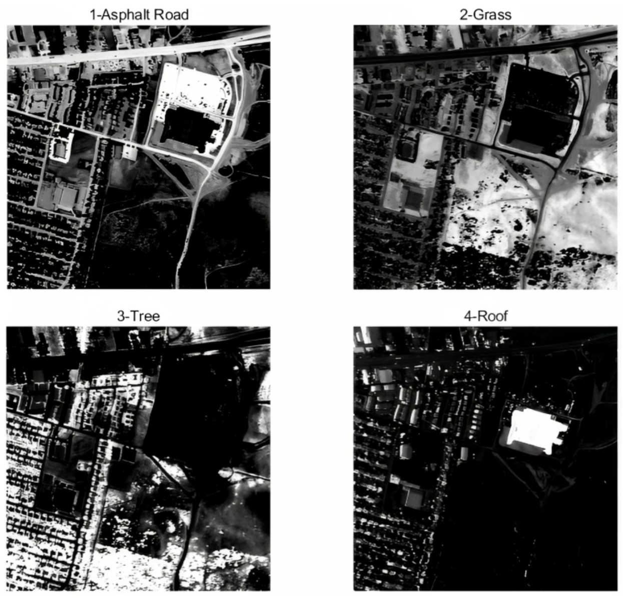


(b)

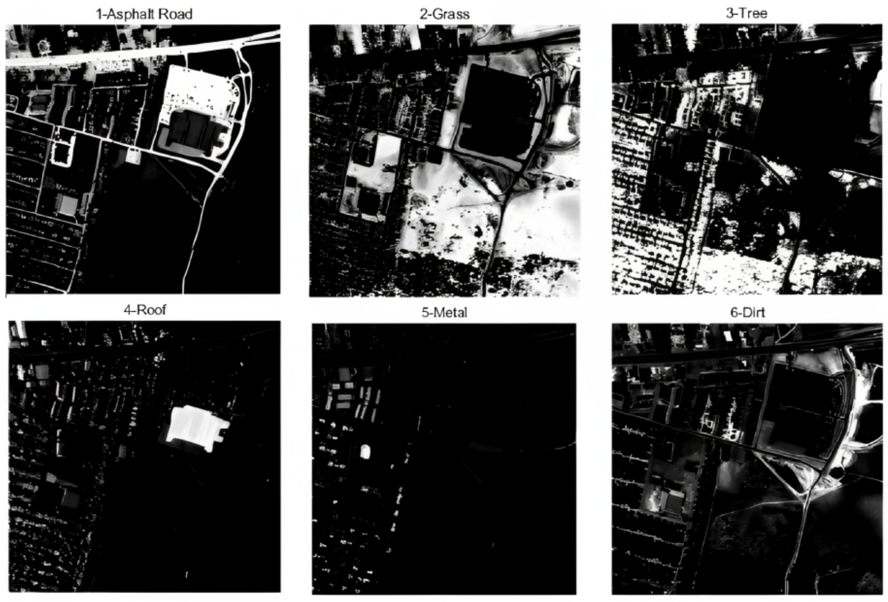
Figure 1.4: Ground-truth spectral signatures of the materials present in the Urban image for (a) 4 and (b) 6 materials considered.

considered. Similarly, Fig. 1.5 presents the corresponding abundance maps, that is, the proportions in which each material appears in each pixel (for a given material, a black pixel indicates the absence of this material in the pixel).

- *Extraction of facial features*: In this case, the data matrix X is a set of n faces, each one made of m grey-scale pixels. The goal of NMF is to



(a)



(b)

Figure 1.5: Ground-truth abundance maps of the Urban image for (a) 4 and (b) 6 materials considered.

identify *r* facial features such as mouths, noses, eyes,... (the columns of *W*) whose linear combinations allow to generate all the faces of the original dataset, according to some proportions (the columns of *H*), see Fig. 1.6.

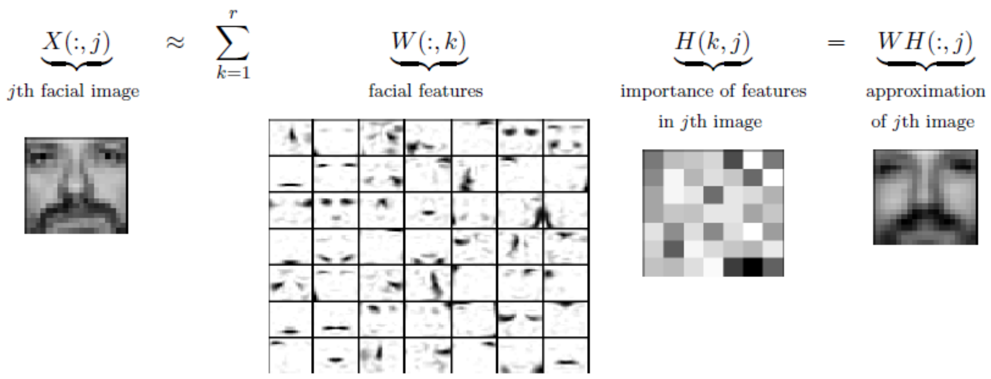

$\underbrace{X(:,j)}_{j\text{th facial image}} \approx \sum_{k=1}^{r} \underbrace{W(:,k)}_{\text{facial features}} \underbrace{H(k,j)}_{\text{importance of features in } j\text{th image}} = \underbrace{WH(:,j)}_{\text{approximation of } j\text{th image}}$

Figure 1.6: Illustration of NMF applied to the extraction of facial features. Figure taken from [55].

1.4 Contributions and outline of the thesis

This PhD investigated several directions of research, which define the organization of the remainder of this document. Let us first list the publications and preprints that we released during the PhD:

1. Pierre De Handschutter, Nicolas Gillis, Arnaud Vandaele, and Xavier Siebert. Near-convex archetypal analysis. *IEEE Signal Processing Letters*, 27:81–85, 2019 [39],

2. Pierre De Handschutter, Nicolas Gillis, and Xavier Siebert. A survey on deep matrix factorizations. *Computer Science Review*, 42:100423, 2021 [38],

3. Pierre De Handschutter and Nicolas Gillis. Deep orthogonal matrix factorization as a hierarchical clustering technique. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 1466–1470. IEEE, 2021 [35],

4. Pierre De Handschutter and Nicolas Gillis. A consistent and flexible framework for deep matrix factorizations. *Pattern Recognition*, 134:109102, 2023 [36],

5. Pierre De Handschutter, Nicolas Gillis, and Wivine Blekic. Deep symmetric matrix factorization. 2023. Online available at: `https://www.researchgate.net/publication/368693970_Deep_Symmetric_Matrix_Factorization` [37].

At the time of writing, a new paper was being written. Note that all the publications contain a link to the corresponding full MATLAB code reproducing the described experiments. Let us also mention that another publication not directly linked with the topic of this PhD was finalized and published during the time of the PhD:

1. Paul Vanabelle, Pierre De Handschutter, Riëm El Tahry, Mohammed Benjelloun, and Mohamed Boukhebouze. Epileptic seizure detection using EEG signals and extreme gradient boosting. *Journal of Biomedical Research*, 34(3):228, 2020 [118].

Let us now describe the organization of the following chapters in view of the scientific contributions:

- Chapter 2 presents key concepts related to matrix factorizations that have been extensively used all along the researches. This includes the description of the optimization framework usually adopted to solve NMF problems in Section 2.1 as well as a quick overview of the initialization techniques in Section 2.2. We also briefly present some important NMF variants in Section 2.3. Finally, in Section 2.4, we discuss aspects of the experimental set-up of NMF models.

- Chapter 3 describes a new NMF model called near-convex archetypal analysis (NCAA) [39]. This model is inspired by a well-known NMF variant, namely archetypal analysis (AA) [32]. Our new model NCAA is more flexible than AA and compares favourably to state-of-the-art algorithms on both synthetic data experiments and HU.

- Chapter 4 introduces the history and motivations behind deep matrix factorizations (deep MF's), the core topic of this thesis. This chapter follows closely the structure of our survey paper [38] and illustrates the recent advances in the deep MF field, in terms of theoretical results, variants of the main models and applications, among others. We also provide showcase examples of deep MF. Finally, we list some interesting challenges and perspectives of research; some of them have been tackled during this PhD.

- In Chapter 5, we discuss two new loss functions for deep MF and sketch a flexible and consistent optimization framework for constrained deep MF [36].
- Chapter 6 reviews the deep MF models and algorithms developed during this PhD. This includes:
  - The successive orthogonal decomposition algorithm (SODA) [35], which is mostly used as a greedy initialization technique for deep MF. In fact, SODA leverages the closed form of the orthogonal NMF of two points, see Section 6.1,
  - Sparse deep MF: we successfully apply the generic framework described in Chapter 5 to sparse deep MF, where a global sparsity measure of the factors is used, see Section 6.2,
  - Minimum-volume deep MF: we successfully apply the generic framework described in Chapter 5 to minimum-volume deep MF, see Section 6.3,
  - Deep symmetric NMF (DSNMF) [37]: we consider a symmetric data matrix X, that is, $X = X^T$, and scale up the standard symmetric NMF algorithm, see Section 6.4.
- In Chapter 7, we present an original application of deep MF to psychiatric networks. More precisely, DSNMF is applied to detect hierarchical overlapping communities of psychiatric symptoms. This work was carried out in collaboration with a post-doctoral psychologist, Wivine Blekic.
- Finally, in Chapter 8, we summarize the main contributions of this PhD and open potential directions of future research.

For readability, Chapters 4 to 7 are grouped together in a different part than Chapter 3.

A word about the writing style: Our goal in this manuscript is mainly to report on the main contributions of this PhD. We endeavour to be as pedagogical as possible and to go along with the reader through the reasoning. However, we assume that the reader has minimal prerequisites in linear algebra and is familiar with vector/matrix notations. Though most of the content of the publications and preprints aforementioned appears in this manuscript, some details are skipped on purpose to make reading easier. Also, some important concepts linked to NMF, such as identifiability, that is, the study of the uniqueness of the factors retrieved by a model, on which we did not focus during this PhD, are put under the rug to avoid useless scattering [1]. On the contrary, we emphasize the open questions and the take-home messages induced by every developed model through dedicated sections at the end of each chapter. Indeed, this seems to us to be the most important possible track of this document. Hence, we aim more at efficiency than at comprehensiveness.

[1] NMF identifiability specialists will maybe see a very unintentional pun in this sentence ☺

Chapter 2

Fundamentals

This chapter focuses on describing important concepts that are reused in most of the remaining chapters. More specifically, Section 2.1 sketches the general optimization framework for NMF, adopted in most of the papers. In Section 2.2, we briefly present several initialization strategies for the NMF factors. In Section 2.3, we describe the main NMF variants and especially focus on orthogonal NMF and minimum-volume NMF. Finally, Section 2.4 gives a brief word on how to assess the performance of any NMF model, and on the generation of synthetic data.

2.1 General optimization framework to solve NMF

Let us consider the following NMF optimization problem,

$$\min_{\substack{W\in\mathbb{R}^{m\times r}\\ H\in\mathbb{R}^{r\times n}}} \frac{1}{2}\|X-WH\|_F^2 \quad \text{s.t. } W\geq 0,\, H\geq 0. \tag{2.1}$$

Most works in the literature solve this NMF problem with a two-block coordinate descent (BCD) strategy. This means that Problem (2.1) is solved as follows. Let us assume that W and H have been initialized somehow (see

Section 2.2 for details). First, the sole matrix W is updated, keeping the matrix H unchanged, that is, the problem $\min_{W\geq 0}\frac{1}{2}\|X-WH^*\|_F^2$ is solved where H^* denotes the current (fixed) value of matrix H. Then, H is updated keeping W fixed, that is, the problem $\min_{H\geq 0}\frac{1}{2}\|X-W^*H\|_F^2$ is solved where W^* denotes the current (fixed) value of matrix W. This alternated procedure, that is, update W with H fixed and update H with W fixed is repeated until some stopping criterion is met. This stopping criterion may be a given number of iterations, a maximal time elapsed, a sufficient decrease of the loss function, a too small gap between two consecutive iterates of a given factor, etc.

This 2-BCD is illustrated by Algorithm 1. Given a nonnegative matrix X of m rows and n columns, and a factorization rank r, the factors W and H are first initialized somehow. Then, W and H are alternatively optimized during a certain number of iterations (the superscripts in Algorithm 1 indicate the iterations).

Algorithm 1 Two-block coordinate descent for solving NMF

```
Input: Nonnegative matrix $X \in \mathbb{R}_+^{m\times n}$, rank $r$.
Output: A rank-$r$ NMF of $X \approx WH$ with $W, H \geq 0$.
1: Generate some initial matrices $W^{(0)}$ and $H^{(0)}$.
2: for $t = 1, 2, \ldots$ do
3:     $W^{(t)} = \text{update\_}W(X, H^{(t-1)}, W^{(t-1)})$
4:     $H^{(t)} = \text{update\_}H(X, W^{(t)}, H^{(t-1)})$
5: end for
```

It turns out that, contrary to the overall NMF Problem (2.1), each subproblem with respect to either W or H, that is, lines 3 and 4, is convex and constitutes a so-called nonnegative least squares (NNLS) problem. Many techniques have been widely used to solve these subproblems, such as multiplicative updates, alternating least squares, alternating nonnegative least squares, hierarchical alternating least squares to cite a few, see [55] for more details.

Let us consider a more general problem than NNLS where the factor to update is not necessarily nonnegative but is enforced to belong to some set. For example, let us consider

$$\min_{W\in\mathcal{W}}\frac{1}{2}\|X-WH^*\|_F^2, \tag{2.2}$$

where $\mathscr{W}$ is the feasible set for W that encompasses the constraints applied to W. In the case of NMF, $\mathscr{W} = \mathbb{R}_+^{m\times r}$, that is, the set of matrices of m rows and n columns with nonnegative entries. A similar reasoning can of course be applied to H.

We will focus on a particular method to solve Problem (2.2), called fast projected gradient descent (FPGD) with restart, a well-known first-order optimization technique (that is, using only gradient information). Starting with the estimation $W^{(t-1)}$ of matrix W at iteration $t-1$, FPGD computes a new estimate $W^{(t)}$ such that $\|X - W^{(t)}H^*\|_F^2 \leq \|X - W^{(t-1)}H^*\|_F^2$, and $W^{(t)} \in \mathscr{W}$.

We will illustrate the working of FPGD on Problem (2.2) through Algorithm 2. Until some stopping criterion is met, the following steps are performed at each inner iteration b:

1. A descent step size s_b is chosen according to some appropriate strategy (line 3).

2. A projected gradient step is performed, that is, the current iterate is moved along the negative gradient direction and then projected on the feasible set (line 4).

3. An extrapolation step is performed through Nesterov acceleration [90] to induce faster convergence of the iterates (line 5).

4. A restart step is applied if the extrapolation did not lead to a decrease of the loss function (line 7). Indeed, despite its acceleration effect, the extrapolation does not guarantee a monotonic decrease of the loss function, contrary to the classical gradient descent.

Note than in Algorithm 2, we used subscripts for the "inner" loop indices and superscripts for the BCD loop indices (as in Algorithm 1). When NMF is considered, the projection operator is $\mathscr{P}_{\mathscr{W}}(x) = \max(0, x)$ where the *max* is taken element-wise. In fact, each column is projected on the nonnegative orthant, that is, the set of nonnegative vectors of appropiate dimension.

FPGD has the advantage to be easy to implement and very flexible. Indeed, the only stage to modify when various constraints are considered is the projection on the feasible set at line 4. Hence, we have often used FPGD to solve the subproblems in the models we developed.

Algorithm 2 Fast projected gradient descent to solve constrained least squares problem

```
Input: Nonnegative matrix $X \in \mathbb{R}_+^{m\times n}$; rank $r$; previous iterate $W^{(t-1)}$ of $W$;
    feasible set $\mathscr{W}$ for $W$; some given matrix $H^*$; parameter $\alpha_1 \in [0,1]$.
Output: An estimate $W^{(t)}$ of $W$ such that $\mathscr{L}(W^{(t)}) \leq \mathscr{L}(W^{(t-1)})$ where
        $\mathscr{L}(W) = \frac{1}{2}\|X - WH^*\|_F^2$.
 1: $W_{(0)} = W^{(t-1)}$; $Z_{(0)} = W^{(t-1)}$
 2: for $b = 1,\dots,B$ do
 3:     Choose somehow the descent step size $s_b$
 4:     $W_{(b)} = \mathscr{P}_{\mathscr{W}}\left(Z_{(b-1)} - s_b \nabla \mathscr{L}(Z_{(b-1)})\right)$
 5:     $Z_{(b)} = W_{(b)} + \beta_b\left(W_{(b)} - W_{(b-1)}\right)$ with $\beta_b = \frac{\alpha_b(1-\alpha_b)}{\alpha_b^2+\alpha_{b+1}}$ and
        $\alpha_{b+1} = \frac{1}{2}\left(\sqrt{\alpha_b^4 + 4\alpha_b^2} - \alpha_b^2\right)$
 6:     if $\mathscr{L}\left(W_{(b)}\right) > \mathscr{L}\left(W_{(b-1)}\right)$ then
 7:         $Z_{(b)} = W_{(b)}$, $\alpha_{b+1} = \alpha_1$
 8:     end if
 9: end for
10: $W^{(t)} = W_{(B)}$
```

2.2 Initialization of NMF

Initialization of the factors of NMF, that is, finding appropriate $W^{(0)}$ and $H^{(0)}$, can be performed through various techniques:

- *Random initialization*: The factors are initialized randomly. However, with this technique, nothing guarantees that the initial approximation $W^{(0)}H^{(0)}$ is close to X or even of the same order of magnitude. Hence, a scaling parameter α^* is also used such that $\alpha^* = \underset{\alpha}{\operatorname{argmin}}\frac{1}{2}\left\|X - \alpha W^{(0)}H^{(0)}\right\|_F^2 = \frac{\langle X, W^{(0)}H^{(0)}\rangle}{\|W^{(0)}H^{(0)}\|_F^2}$ where $\langle\cdot,\cdot\rangle$ indicates the scalar product of two matrices. The initial factors are then "scaled" according to the value of α^*: $W^{(0)} \leftarrow \sqrt{\alpha^*}W^{(0)}$ and $H^{(0)} \leftarrow \sqrt{\alpha^*}H^{(0)}$.

- *Successive nonnegative projection algorithm* (SNPA) [54]: SNPA is an algorithm originally used to solve (near) separable NMF. Separable NMF is a particular NMF variant where the columns of W constitute a subset of the columns of X. In other words, there exists a subset

$\mathcal{J} \subset \{1, \ldots, n\}$ of r indices such that $W = X(:, \mathcal{J})$. This also implies that there is an identity submatrix of size r (up to columns permutations) within H. Hence, separable NMF is such that each of the n data points can be expressed as a nonnegative linear (also called conical) combination of a subset of r data points. SNPA may be a useful way to initialize the factors of any NMF model; let us detail its working through Algorithm 3.

Algorithm 3 Successive Nonnegative Projection Algorithm (SNPA) in the separable case [54]

Input: Separable nonnegative matrix $X \in \mathbb{R}_+^{m \times n}$, rank r.
Output: Set of r indices $\mathcal{J}$, matrix $W \in \mathbb{R}^{m \times r}$ such that $W = X(:, \mathcal{J})$, matrix $\tilde{H} \in \mathbb{R}^{r \times n}$.

```
1: $R = X \quad ; \quad \mathcal{J}, W = \{\} \quad ; \quad t = 1$
2: while $R \neq 0$ and $t \leq r$ do
3:     $p = \arg\max_j \|R(:, j)\|_2$
4:     $\mathcal{J} = \mathcal{J} \cup \{p\}$
5:     $W = [W, X(:, p)]$   (that is, $W = X(:, \mathcal{J})$)
6:     $\tilde{H} = \arg\min_H \|X - WH\|_F^2$   s.t. $H \geq 0$,  $\sum_{k=1}^{t} H(k, j) \leq 1$   for all $j = 1, \ldots, n$
7:     $R = X - W\tilde{H}$
8:     $t = t + 1$
9: end while
```

Given an input data matrix X, SNPA first extracts the column with the largest ℓ_2-norm (line 3), which corresponds to the first basis vector, that is, column of W. Then, the columns of X are projected onto the convex hull of the columns of W (at this stage, there is only one) and the origin. More precisely, the matrix $W\tilde{H}$, where $\tilde{H}$ is the solution of the optimization problem at line 6, corresponds to the projection of the columns of X on this convex hull. Then, at line 7, a residual matrix R is defined as the difference between the input matrix X and the computed projection. This residual matrix can somehow be interpreted as the "portion" of X that still needs to be explained by additional basis vectors. The previous stages are repeated on the residual matrix of the previous iteration until r basis vectors are found.

Since SNPA is easy to use and gives good initial factors in practice, we have often used SNPA as an initialization technique for the models we developed.

- *Hierarchical clustering (HC):* The NMF factors can also be initialized through hierarchical clustering methods which aim at splitting the data in an increasing number of clusters becoming smaller. Let us focus on a technique based on rank-2 NMF [58] and originally inspired by HU to explain how HC can be leveraged for the initialization of NMF factors. At the first step, all the data points are gathered in a single cluster. Then, for some iterations, the method splits one cluster into two smaller clusters such that a particular loss function is decreased as much as possible. In fact, this loss function aims to approximate each cluster by a matrix of rank 1 corresponding to a single basis vector. The splitting procedure consists in an NMF of rank $r = 2$ over the points belonging to the initial cluster. Hence, $W^{(0)}$ can be found by applying this method until r clusters, whose centroids correspond to the initial basis vectors, are identified. The initial matrix $H^{(0)}$ can then be computed by solving the NNLS problem $\min_{H \geq 0} \| X - W^{(0)} H \|_F^2$ for which many algorithms exist such as active set and gradient descent methods. In the following, we will denote HC the specific aforementioned algorithm proposed in [58].

  To sum up, contrary to SNPA which extracts extreme data points, HC extracts more central points.

- *Other techniques*, such as singular value decomposition (SVD) based initialization [18].

Initialization of NMF factors still remains an open problem. In practice, multiple initializations can be run, and the one leading to the best performance is kept.

2.3 Main NMF variants

NMF has numerous variants depending on the additional constraints and/or regularizations added to the original model (1.3), or the assumptions on the input data matrix X. Let us focus on two particular variants, namely orthogonal NMF (ONMF) in Section 2.3.1 and minimum-volume NMF (minVolNMF) in Section 2.3.2, which are extensively referred to in the next chapters.

2.3.1 Orthogonal NMF

ONMF [40] is an NMF variant for which the rows of H are constrained to be orthogonal to each other, that is, $H(k,:)H(l,:)^T = 0$ for all $k \neq l$. Moreover, an additional normalization enforces each row of H to have its l_2 norm equal to 1, that is, $H(k,:)H(k,:)^T = 1$ for all $k = 1,\dots,r$. In other words, H is such that $H \geq 0$ and $HH^T = I_r$ where I_r is the identity matrix of dimension r. Together with the nonnegativity of H, orthogonality implies that each column of H has a single entry strictly greater than 0. Overall, the ONMF model can be written as:

$$\min_{\substack{W\in\mathbb{R}_+^{m\times r}\\ H\in\mathbb{R}_+^{r\times n}}} \|X - WH\|_F^2 \qquad \text{such that} \qquad HH^T = I_r. \tag{2.3}$$

ONMF is an interesting model in terms of interpretability: since H has a single non-zero entry per column, this implies that each data point is only associated to one basis vector (one column of W), and ONMF is equivalent to a hard clustering problem (in fact, ONMF is a weighted version of spherical k-means, see [99] for more explanations). In other words, ONMF imposes that each data point belongs to a single cluster, which is represented by a single basis vector corresponding to the cluster's centroid. Hence, the ONMF factors can be easily interpreted: the columns of W are cluster centroids, while the columns of H assign each data point to its closest centroid (up to a scaling factor).

2.3.2 Minimum-volume NMF

MinVolNMF is an NMF variant where the volume delimited by the basis vectors, that is, the columns of W, is minimized. More precisely, a regularization term is added to the NMF loss function such that the minVolNMF loss function is given by

$$\mathscr{L}(W,H) = \frac{1}{2}\Big(\|X - WH\|_F^2 + \lambda \text{vol}(W)\Big), \tag{2.4}$$

where λ is a regularization parameter and $\text{vol}(W)$ is a function that evaluates the volume of the convex hull of the columns of W. Several choices are possible for $\text{vol}(W)$, see [3]. A popular choice consists in taking $\text{vol}(W) = \log\det(W^TW+\delta I_r)$, where $\det(W^TW)$ is the determinant of the square matrix W^TW, δ a positive parameter and I_r the identity matrix of order r. The term δI_r is simply added to avoid the risk of computing $\log(0)$. The overall minVolNMF problem is formulated as:

$$\min_{W,H} \frac{1}{2}\Big(||X-WH||_F^2+\lambda\log\det(W^TW+\delta I_r)\Big)$$
$$\text{such that} \quad W,H\geq 0, \quad \sum_{k=1}^{r} H(k,j)\leq 1 \quad \text{for all } j=1,...,n. \tag{2.5}$$

where the constraints $\sum_{k=1}^{r} H(k,j) \leq 1$ for all $j = 1,\dots,n$, similarly to SSNMF (see Eq. (1.4)), leverage the scaling degrees of freedom. MinVolNMF is generally solved with a 2-BCD as in Algoritm 1.

The subproblem with respect to H is the same as in SSNMF, since H is not involved in the penalty term. However, the attentive reader may have noticed that the column-stochasticity of H has been relaxed into an inequality constraint. In fact, this is mainly application-driven: for example, if black pixels (that is, with the corresponding column of X being an all-zeros vector) are present in a hyperspectral image, the column-stochasticity of H would require the zero vector to be a basis vector. Hence, the "less or equal to one" constraint is more general and enables to handle these situations.

The subproblem with respect to W, that is,

$$\min_{W}\Big(\mathscr{L}(W)=\frac{1}{2}\Big(||X-WH||_F^2+\lambda\log\det(W^TW+\delta I_r)\Big)\Big) \quad \text{s.t.} \quad W\geq 0, \tag{2.6}$$

is more challenging but has been intensively discussed in the literature. In fact, Problem (2.6) can be solved with a majorization-minimization technique: the loss function $\frac{1}{2}\Big(||X-WH||_F^2+\lambda\log\det(W^TW+\delta I_r)\Big)$ is upper-bounded by a strongly convex function easier to minimize. More precisely, the iterate $W^{(t+1)}$ at iteration $t+1$ is found by observing [77, 53] that

$$\text{vol}(W) = \log\det(W^TW+\delta I_r) \leq f(W,W^{(t)}) = \text{Tr}(Z^{(t)}W^TW) - \log\det(Z^{(t)}) + K$$

where $\mathrm{Tr}(\cdot)$ denotes the trace, that is, the sum of the diagonal elements of a matrix, K is some constant and $Z^{(t)}$ is a fixed matrix such that $Z^{(t)} = (W^{(t)^T} W^{(t)} + \delta I_r)^{-1}$. Hence, the overall loss function $\mathscr{L}(W)$ can in turn be bounded above by

$$g(W, W^{(t)}) = \frac{1}{2}\Big(||X - WH^{(t)}||_F^2 + \lambda(\mathrm{Tr}(Z^{(t)} W^T W) - \log\det(Z^{(t)}) + K)\Big).$$

Getting rid of the terms constant with respect to W, this comes to minimizing the quadratic function

$$h(W) = \frac{1}{2}\langle W^T W, A^{(t)}\rangle - \langle C^{(t)}, W\rangle$$

with $A^{(t)} = H^{(t)} H^{(t)^T} + \lambda Z^{(t)}$, $C^{(t)} = XH^{(t)^T}$. The gradient is easy to compute (in fact, $\nabla h(W) = WA - C$). A summary of the main steps of FPGD applied to minVolNMF is presented in Algorithm 4.

Algorithm 4 Minimum-volume NMF with fast projected gradient descent

```
Input: Nonnegative matrix $X \in \mathbb{R}_+^{m \times n}$, rank $r$, parameter $\delta > 0$.
Output: Two matrices $W \in \mathbb{R}^{m \times r}$, $H \in \mathbb{R}^{r \times n}$.
1: Compute initial matrices $W^{(0)}$ and $H^{(0)}$, and the regularization parameter $\lambda$
2: for $t = 1, 2, \dots$ do
3:     $Z^{(t-1)} = (W^{(t-1)^T} W^{(t-1)} + \delta I_r)^{-1}$
4:     $A^{(t-1)} = H^{(t-1)} H^{(t-1)^T} + \lambda Z^{(t-1)}$; $C^{(t-1)} = XH^{(t-1)^T}$
5:     $W^{(t)}$=Update $W$ with FPGD on $\min_{W \geq 0}\big(\frac{1}{2}\langle W^T W, A^{(t-1)}\rangle - \langle C^{(t-1)}, W\rangle\big)$
6:     $H^{(t)}$=Update $H$ with FPGD on $\min_{H \geq 0} \frac{1}{2}\|X - W^{(t)} H\|_F^2$ such that
       $\sum_{k=1}^{r} H(k, j) \leq 1$ for all $j$
7: end for
```

The parameter δ is of minor importance and is generally fixed to a small positive value, for example $\delta = 0.1$. In addition, the regularization parameter λ balances the importance of the data fitting term, that is, the reconstruction error, on the one hand, and the volume regularization term on the other hand. A possible strategy for the choice of λ is to tune an initial guess after the initialization of the factors. More precisely, the regularization parameter is initialized to some value, for example $\tilde{\lambda} = 10^{-2}$. Then, given $W^{(0)}$ and $H^{(0)}$, the regularization parameter is adapted such that the ratio between the two contributions of the loss function is equal to $\tilde{\lambda}$, that is, $\tilde{\lambda} = \frac{\lambda \text{vol}(W)}{\|X-WH\|_F^2}$. Hence, the "final" regularization parameter λ is given by:

$$\lambda = \tilde{\lambda} \frac{\|X - WH\|_F^2}{\text{vol}(W)}. \tag{2.7}$$

Intuitively, minVolNMF aims at reducing the volume delimited by the basis vectors while at the same time keeping a low reconstruction error. Therefore, the basis vectors are likely to be close to the convex hull of the data points (due to the normalization of the columns of H) hence to be associated to a meaningful physical interpretation in real-world settings.

Note that other minVol approaches exist, such as Minimum Volume Simplex Analysis (MVSA) [80] and Minimum Volume Enclosing Simplex (MVES) [21] which are similar models minimizing the sole volume of the basis vectors and asking for the data points to lie inside their convex hull, after dimensionality reduction. These models were specifically developed for HU. Robust variant of both MVSA and MVES, respectively SISAL [14] and RMVES [2], allow the data points to lie (slightly) outside the convex hull of the endmembers. These variants relax the nonnegativity constraint on H (contrary to minVolNMF described above), which renders these models more robust to noise. We refer the reader to Bioucas Dias et al. comprehensive survey on HU methods [15] for more details.

2.4 Experimental set-up

When a new NMF model is developed, it is important to assess its performance on both synthetic data, that is, data for which the setting is highly controlled by the experimenter, and real-life applications. In Section 2.4.1, we explain how we generate synthetic data in most situations. More specific settings are described in the next chapters, when they are explicitly used. In Section 2.4.2, we present the evaluation metric the most frequently adopted along this thesis. Some other metrics will be introduced later.

2.4.1 Generation of synthetic data

Synthetic data allow to test the performance of a model by keeping a high level of control on the data distribution, the dimension, the noise...

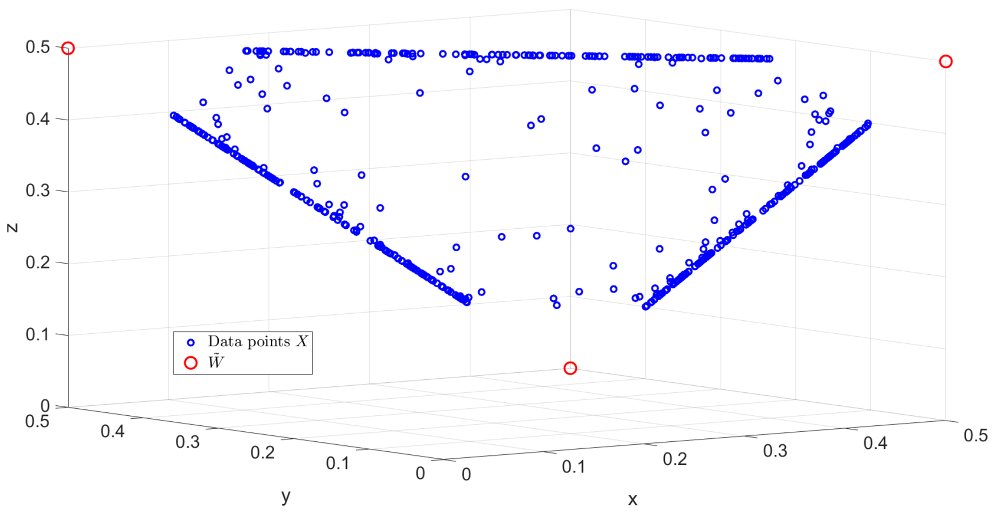


Figure 2.1: Illustration of synthetic data generated with a symmetric Dirichlet distribution in $m = 3$ dimensions, with $r = 3$.

In practice, we first create the "true" matrix of basis vectors $\tilde{W}$ and then generate the columns of the "true" matrix of proportions $\tilde{H}$ according to some probability distribution. In our experiments, unless specified otherwise, we generate the n columns of $\tilde{H}$ according to a symmetric Dirichlet distribution. This is a multivariate distribution depending on a vector of r parameters all equal to α, which produces r-dimensional samples whose entries are real numbers in $[0,1]$ summing to 1. If $\alpha = 1$, this merely degenerates to a uniform distribution over the $r-1$ standard simplex while if α is close to 0, the distribution will be highly sparse, with only a few non-zero entries per sample. Let us define the purity level p_k associated to the basis vector $\tilde{W}(:,k)$ as $p_k = \max_{1 \le j \le n} \tilde{H}(k,j)$ for any $k = 1,\dots,r$, that is, as the maximal "proportion" of the kth basis vector in any data point. In practice, it is also important to control the purity levels in experiments. Hence, if a column of $\tilde{H}$ sampled with the Dirichlet distribution has an entry exceeding the corresponding expected purity level, we resample it until it is fully compliant with the purity requirements. Once both $\tilde{W}$ and $\tilde{H}$ have been generated, the noiseless "true" data matrix is given by $\tilde{X} = \tilde{W}\tilde{H}$.

For example, Fig. 2.1 shows $n = 500$ points generated in $m = 3$ dimensions with $r = 3$, $\alpha = 0.05$, and all the p_k's equal to 0.8.

In general, noise is added to $\tilde{X}$ to test the robustness to noise of the model. When the data is generated as described, it is common to consider white additive Gaussian noise. Practically, we generate a matrix $N \in \mathbb{R}^{m\times n}$ whose elements are drawn from the normal distribution $\mathcal{N}(0,1)$ and define ν as the relative noise level. Then, we build the final data matrix X as

$$X = \tilde{X} + \nu N \frac{\|\tilde{X}\|_F}{\|N\|_F}. \tag{2.8}$$

This way, $\|X - \tilde{X}\|_F = \nu\|\tilde{X}\|_F$.

2.4.2 Evaluation metric

It is important to have proper evaluation metrics to assess the quality of a model and compare it to the state of the art.

Comparing the final values of the loss function is not always the most practical option since different models may have different loss functions which are not necessarily meaningfully comparable.

For synthetic data, as well as for many intensively used real-world datasets, the "true" expected factors $\tilde{W}$ and $\tilde{H}$ that generate the data are known, either by construction (for synthetic data) or by domain expertise (for real data). Hence, one can compare the pair of matrices W and H computed through NMF and the ground-truth factors $\tilde{W}$ and $\tilde{H}$. However, there is no guarantee that the basis vectors are retrieved in the same order as they appear in the ground truth. Therefore, the columns of W (and analogously, the rows of H) must be reordered according to an affectation algorithm between the (normalized) columns of W and $\tilde{W}$ (and similarly with the rows of H). This reordering is performed through the well-known Hungarian or Munkres combinatorial algorithm, see [89] for details.

Once the columns of W are properly reordered, we can compute the so-called mean-removed spectral angle (MRSA) between each pair of corresponding columns of W and $\tilde{W}$. The MRSA between two vectors x and y is defined as

$$\text{MRSA}(x,y) = \frac{100}{\pi}\text{arcos}\left(\frac{\langle x-\overline{x}, y-\overline{y}\rangle}{\|x-\overline{x}\|_2\|y-\overline{y}\|_2}\right) \in [0,100], \tag{2.9}$$

where $\langle\cdot,\cdot\rangle$ indicates the scalar product of two vectors and $\overline{x}$ is the mean of vector x, that is, $\overline{x} = \text{mean}(x)$. In fact, the MRSA between two vectors is the normalized angular distance between the centred version of these vectors.

In practice, the metric of interest is the average of the MRSA's over the r basis vectors, that is, $\frac{1}{r}\sum_{k=1}^{r} \text{MRSA}\big(W(:,k), \tilde{W}(:,k)\big)$. By convenience, in the following, we will denote MRSA this average quantity instead of the column-wise metric. The lower the MRSA is, the better the matching between the ground truth and the retrieved factors is, hence the better the underlying algorithm is.

Part I

Near-Convex Archetypal Analysis

CHAPTER 3

Near-Convex Archetypal Analysis

In this chapter, we introduce a model called *Near-convex archetypal analysis* (NCAA) [39]. NCAA is a flexible extension of archetypal analysis (AA), which is in turn a state-of-the-art variant of NMF.

In Section 3.1, we describe the AA model and emphasize its main limitations, which motivated the development of NCAA. Then, we introduce the mathematical formulation and geometrical interpretation of NCAA in Section 3.2. Section 3.3 discusses algorithmic aspects of NCAA, including its computational cost while Section 3.4 proposes some strategies for the choice of NCAA parameters. We end up with the final NCAA algorithm in Section 3.5. Section 3.6 and 3.7 discuss the performance of NCAA on synthetic data and hyperspectral unmixing respectively. Section 3.8 lists some perspectives of improvement, including drafts of alternative models. Finally, Section 3.9 summarizes the main points of the chapter.

3.1 The archetypal analysis framework

Archetypal analysis (AA) is a model closely related to SSNMF (see Section 1.2). However, in AA, an additional major constraint is considered compared to Eq. (1.4). Indeed, the basis vectors, also called in this case *archetypes*, are not only points such that all the columns of X are approximated as convex combinations of them but are also themselves enforced to be convex combinations of the data points. Fig. 3.1 sketches the geometry of AA for $m = 2$ and $r = 3$.

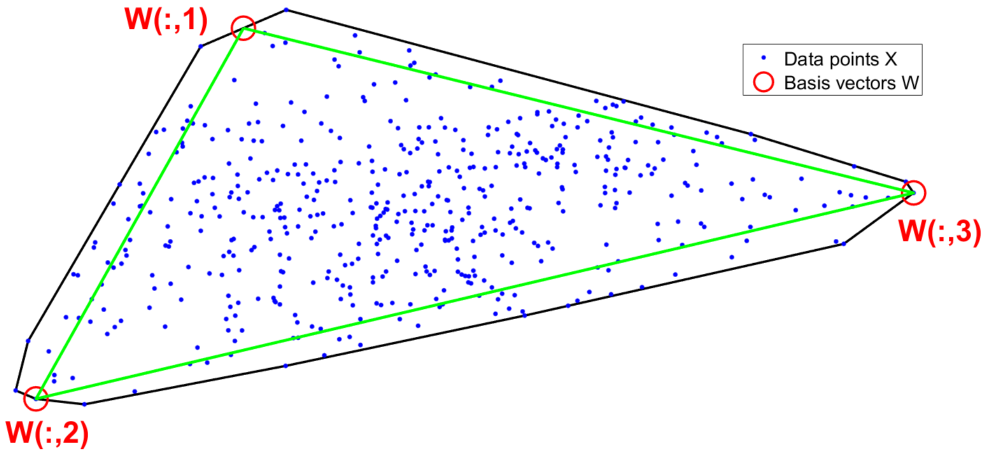


Figure 3.1: Illustration of archetypal analysis for $r = 3$: the data points are in blue, their convex hull is in black, the three basis vectors are in red and their convex hull is the green triangle. The data points outside the green triangle can not be perfectly represented as convex combinations of the red points.

More precisely, the basis vectors are given by $W = XA$ where A satisfies the same conditions as H (that is, nonnegativity and column-stochasticity) and the goal of AA is therefore to find two matrices $A \in \mathbb{R}^{n\times r}$ and $H \in \mathbb{R}^{r\times n}$ that solve the following constrained optimization problem:

$$\begin{aligned} &\min_{A,H} \frac{1}{2}||X - XAH||_F^2 \\ \text{such that } A, H \geq 0, \quad &\sum_{j=1}^{n} A(j,k) = 1 \quad \text{for all } k = 1, \ldots, r, \\ &\sum_{k=1}^{r} H(k,j) = 1 \quad \text{for all } j = 1, \ldots, n. \end{aligned} \tag{3.1}$$

Note that AA is closely related to convex NMF, introduced by [41], where the data matrix X is not necessarily nonnegative and the sum-to-one constraints are discarded (that is, the convex combinations are replaced by *conical* combinations). AA has raised a lot of interest in the NMF literature, see [12], [23] and [88] among others.

An obvious advantage of AA lies in its interpretability: the basis vectors are combinations of the data points. Moreover, as the basis vectors are in the convex hull of the columns of X, they are likely to be "not too far" from most of the data points hence to correspond to some physical reality. However, in its standard version, AA has two main drawbacks:

1. As the data points are themselves approximated by convex combinations of the archetypes, there is a risk that the reconstruction error is high. Indeed, on Fig. 3.1, some data points lie outside the convex hull of the columns of W hence can not be correctly approximated as convex combinations of the basis vectors through Model (3.1).

   In [66], AA and standard NMF are somehow combined through an objective function that makes a tradeoff between the reconstruction error and the distance between the archetypes and the convex hull of X. More precisely, this loss function is given by:

   $$\mathscr{L}(W,H) = \frac{1}{2}\|X - WH\|_F^2 + \lambda \sum_{k=1}^{r} D(W(:,k),X)$$
   $$\text{such that} \quad \sum_{k=1}^{r} H(k,j) = 1 \quad \text{for all } j = 1,\dots,n.$$

   where $D(W(:,k),X)$ is the minimal Euclidean distance between $W(:,k)$, that is, the kth archetype, and any point in the convex hull of X. In other words, this model can be seen as a relaxed version of AA, where the archetypes are not hardly constrained to be convex combinations of the columns of X, but the objective function tries to minimize their distance from the convex hull of X. Unfortunately, the factors involved in the optimization problem are $W \in \mathbb{R}^{m\times r}$ and $H \in \mathbb{R}^{r\times n}$, that is, the standard NMF variables. Consequently, though the model is close to AA, it does not allow to interpret how the archetypes are built from the data through a coefficient matrix A as in Problem (3.1).

   In [88], an interesting relaxation of AA handles the case where it is not possible to find a good solution with archetypes expressed as convex combinations of the data points. The idea is to allow the sum of the entries of each column of A to be slightly different from 1, more precisely between $1-\delta$ and $1+\delta$ for some $\delta > 0$. In other words, the relaxed model consists in solving the following optimization problem:

$$\min_{A,H} \frac{1}{2}||X - XAH||_F^2$$

$$\text{such that} \quad A, H \geq 0, \quad 1-\delta \leq \sum_{j=1}^{n} A(j,k) \leq 1+\delta \quad \text{for all } k = 1, \ldots, r,$$

$$\sum_{k=1}^{r} H(k,j) = 1 \quad \text{for all } j = 1, \ldots, n.$$

This model adds flexibility to AA but the parameter δ is fixed by hand, which is difficult to do in practice for real applications.

2. The number of parameters to optimize is higher in AA than in classical NMF. Indeed, there are $nr + nr$ parameters in AA while "only" $mr + nr$ in NMF, and in many applications, such as HU, $m < n$. Intuitively, it does not make sense to consider that all the n data points will contribute to all the r basis vectors, that is, A is likely to be sparse (in other words, A is likely to contain a lot of entries equal to 0). To tackle this concern, the support of AA (that is, the points whose the archetypes are effectively convex combinations of) is discussed in [11]. Instead of using the whole X as in Model (3.1), they propose to express the basis vectors as convex combinations of only the vertices of the convex hull of X. Unfortunately, computing the convex hull of X is itself $\mathcal{O}(n \log n)$ for $m = 2$ and worsens as the dimension increases.

NCAA aims to alleviate these two concerns by allowing more flexibility to build the archetypes and by reducing the computation cost.

3.2 Near-convex archetypal analysis model

The proposed NCAA model is stated as follows:

$$
\begin{aligned}
&\min_{\substack{A\in\mathbb{R}^{d\times r}\\ H\in\mathbb{R}^{r\times n}}} \frac{1}{2}\|X-YAH\|_F^2 \\
\text{s.t.}\quad & A(l,k)\geq -\epsilon \quad \text{for all } k,l, \quad \epsilon\geq 0, \quad \sum_{l=1}^{d} A(l,k)=1 \quad \text{for all } k=1,\dots,r, \\
& H\geq 0, \quad \sum_{k=1}^{r} H(k,j)=1 \quad \text{for all } j=1,\dots,n.
\end{aligned}
\tag{3.2}
$$

Given an input matrix X of n data points in dimension m and a matrix Y of d points in dimension m, the goal of NCAA is to minimize the squared Frobenius norm of the difference between the data matrix X and its approximation YAH.

The constraints on H are exactly the same as in AA and in SSNMF, that is, nonnegativity together with column-stochasticity. Indeed, the data points are expected to be convex combinations of the archetypes.

By contrast, the constraints on A can be seen as a relaxed form of those that hold in classical AA. Indeed, letting some entries of A be slightly negative ("slightly" meaning "up to $-\epsilon$", with $\epsilon\geq 0$) implies that the archetypes could lie out of the convex hull of the data points. More precisely, as ϵ increases, the archetypes $W=YA$ are allowed to lie further away from the convex hull of the columns of Y and if $\epsilon=0$, NCAA boils down to classical AA (if $Y=X$), see Lemma 3.1 afterwards. On the other hand, the column-stochasticity of A is preserved and guarantees that the archetypes are in the affine hull (that is, the set of linear combinations whose coefficients sum to one) of the columns of X. These constraints justify the naming "near-convex" of Model (3.2). More precisely, regarding the constraints that apply on A, we say that the columns of $W=YA$ are near-convex combinations of the columns of Y.

Finally, let us observe that the archetypes are not built upon the whole set of data points (that is, the matrix X) but only upon a subset of $d\ll n$ points corresponding to the columns of Y.

In summary, there are two main differences between NCAA and AA: the constraints on the entries of A and the use of a matrix Y narrower than X to build the archetypes.

At this stage, key questions arise from the model statement and will be discussed progressively:

- Which geometric interpretation can we give to NCAA ? See hereunder in this section.
- How to solve Problem (3.2) ? See Section 3.3.
- How to choose the parameters of NCAA, that is, the matrix Y and the parameter ϵ ? See Section 3.4.

Let us first focus on the geometrical interpretation of NCAA. In NCAA, the basis vectors are given by $W = YA$ where A satisfies the constraints $A(l,k) \geq -\epsilon$ for all k, l (with $\epsilon \geq 0$) and $\sum_{l=1}^{d} A(l,k) = 1$ for all $k = 1,\ldots,r$. Each archetype, that is, each column of W, can be written as:

$$W(:,k) = YA(:,k) = \sum_{l=1}^{d} Y(:,l)A(l,k) \quad \text{for all } k = 1,\ldots,r. \tag{3.3}$$

For simplicity, we will call $\boldsymbol{y}_l$ the l-th column of Y in the following (notice the bold writing, to distinguish vectors from scalars), that is, $\boldsymbol{y}_l = Y(:,l)$. We are interested in describing the feasible set of the archetypes, that is, the set of near-convex combinations of d points $\boldsymbol{y}_l$ $(l = 1,\ldots,d)$:

$$\mathscr{S} = \left\{ w \in \mathbb{R}^m : w = \sum_{l=1}^{d} a_l \boldsymbol{y}_l \text{ s.t. } a_l \geq -\epsilon \quad \text{for } l = 1,\ldots,d, \quad \epsilon \geq 0, \quad \sum_{l=1}^{d} a_l = 1 \right\}. \tag{3.4}$$

This set $\mathscr{S}$ has a nice geometrical meaning, as stated in the following lemma:

Lemma 3.1. *The set $\mathscr{S}$ of near-convex combinations of points $\boldsymbol{y}_l$ $(l = 1,\ldots,d)$, as defined in Eq.* (3.4)*, is equal to the set of convex combinations of points $\boldsymbol{z}_l$ such that*

$$\boldsymbol{z}_l = \boldsymbol{y}_l(1 + d\epsilon) - d\epsilon\overline{\boldsymbol{y}} \quad \textit{for } l = 1,\ldots,d \tag{3.5}$$

where $\overline{\boldsymbol{y}} = \frac{1}{d}\sum_{l=1}^{d} \boldsymbol{y}_l$ is the mean of the $\boldsymbol{y}_l$'s, that is,

$$\mathscr{S} = \left\{ w \in \mathbb{R}^m : w = \sum_{l=1}^{d} b_l \boldsymbol{z}_l \quad \text{s.t.} \quad b_l \geq 0, \quad \text{for } l = 1,\ldots,d, \quad \sum_{l=1}^{d} b_l = 1 \right\}. \tag{3.6}$$

Proof. Let us consider the set given by Eq. (3.6) where the $\boldsymbol{z}_l$'s are defined according to Eq. (3.5). We will show that this set is the same as the set defined by Eq. (3.4).

Each element w of the set of Eq. (3.6) can be written as

$$w = \sum_{l=1}^{d} b_l \boldsymbol{z}_l = \sum_{l=1}^{d} b_l(\boldsymbol{y}_l(1+d\epsilon) - d\epsilon\overline{\boldsymbol{y}}) \tag{3.7}$$

$$= \sum_{l=1}^{d} b_l(1+d\epsilon)\boldsymbol{y}_l - d\epsilon\overline{\boldsymbol{y}}\sum_{l=1}^{d} b_l. \tag{3.8}$$

Since, $\sum_{l=1}^{d} b_l = 1$, we have

$$\sum_{l=1}^{d} b_l \boldsymbol{z}_l = \sum_{l=1}^{d} b_l(1+d\epsilon)\boldsymbol{y}_l - d\epsilon\overline{\boldsymbol{y}} \tag{3.9}$$

$$= \sum_{l=1}^{d} b_l(1+d\epsilon)\boldsymbol{y}_l - d\epsilon\frac{1}{d}\sum_{l=1}^{d} \boldsymbol{y}_l \tag{3.10}$$

$$= \sum_{l=1}^{d} [b_l(1+d\epsilon) - \epsilon]\boldsymbol{y}_l. \tag{3.11}$$

Let us call $c_l = b_l(1+d\epsilon) - \epsilon$ for all l. As we know that $b_l \geq 0$ for all l and $1 + d\epsilon \geq 1$ if $\epsilon \geq 0$, it appears that $c_l \geq -\epsilon$. Moreover,

$$\sum_{l=1}^{d} c_l = \sum_{l=1}^{d} [b_l(1+d\epsilon) - \epsilon] \tag{3.12}$$

$$= \sum_{l=1}^{d} b_l + d\epsilon\sum_{l=1}^{d} b_l - d\epsilon = 1 \tag{3.13}$$

as $\sum_{l=1}^{d} b_l = 1$. Putting all together, we have $w = \sum_{l=1}^{d} b_l \boldsymbol{z}_l = \sum_{l=1}^{d} c_l \boldsymbol{y}_l$ with $c_l \geq -\epsilon$ for $l = 1, \ldots, d$ and $\sum_{l=1}^{d} c_l = 1$ which corresponds exactly to the definition of any vector of the set $\mathscr{S}$ defined in Eq. (3.4). □

We illustrate this interesting geometric observation for $\epsilon = 0.05$ through the example of Fig. 3.2. Given a set of $d = 6$ well-chosen points $\boldsymbol{y}_l$ with $l = 1, \ldots, 6$, we generate $n = 1000$ data points as near-convex combinations

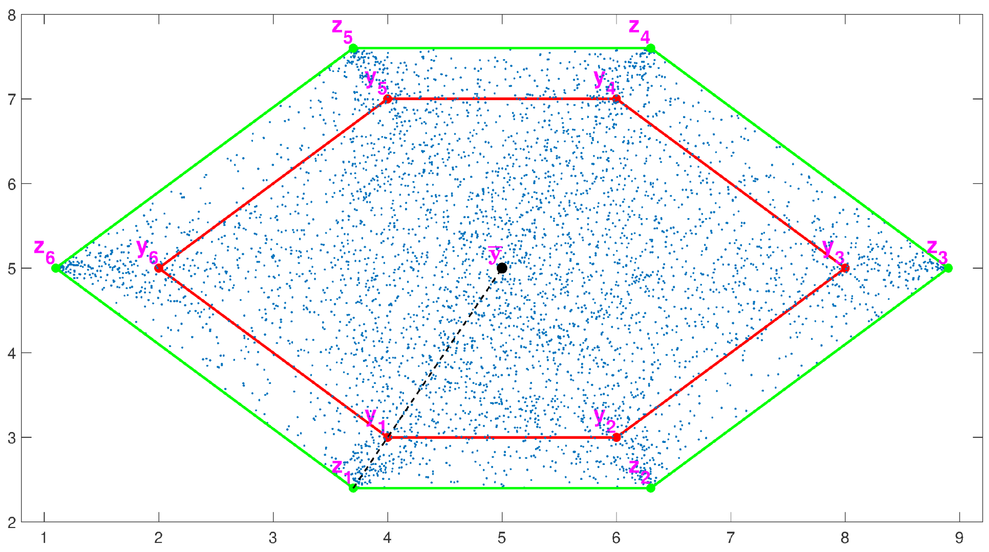


Figure 3.2: Illustration of Lemma 3.1 for $d = 6$, $n = 1000$ and $\epsilon = 0.05$: the data points (blue dots) are near-convex combinations of the points $\boldsymbol{y}_l$'s hence convex combinations of the points $\boldsymbol{z}_l$'s given by Eq. (3.5).

of $\boldsymbol{y}_l$'s and check that they lie within the convex hull of the $d = 6$ points $\boldsymbol{z}_l$'s given by Eq. (3.5). Let us briefly explain how the $n = 1000$ data points were generated in practice for this example. Given Eq. (3.4), we generate randomly a_1. For this purpose, we need to know the lower and upper bound of a_1. We know that $a_1 \geq -\epsilon$ but what is the upper bound ? As the a_l's must sum to one, $a_1 = 1-(a_2+a_3+a_4+a_5+a_6)$. The greatest value of a_1 is obtained when the other a_l's $(l = 2,...,6)$ are at their lowest level, that is, $-\epsilon$. Thus, the upper bound for a_1 is $1+5\epsilon$. Similarly, for a_2, the lower bound is $-\epsilon$ and the upper bound is deduced by the fact that $a_2 = 1-(a_1+a_3+a_4+a_5+a_6)$. As we suppose that a_1 has already been chosen, the upper bound of a_2 is $1+4\epsilon-a_1$.

To generalize, if we want to generate randomly near-convex combinations of d data points according to Eq. (3.4), we need to pick the coefficients a_l's (for $l = 1,\ldots,d$) uniformly in the following interval:

$$a_l \in [-\epsilon; 1+(d-l)\epsilon - \sum_{j=1}^{l-1} a_j] \quad \text{for all } l = 1,\ldots,d-1, \tag{3.14}$$

and $a_d = 1 - \sum_{j=1}^{d-1} a_j$. If we generate 1000 points, we create 1000 sets of coefficients $\{a_1, ..., a_d\}$ according to Eq. (3.14) and then shuffle each set to ensure a uniform distribution. Indeed, the bounds on the coefficients were built by supposing that they are picked sequentially (first a_1, then a_2,...) which implies that a_d is always tighter bounded than a_{d-1} which is in turn tighter bounded than a_{d-2} and so on. The shuffling allows to avoid this bias.

3.3 Two-block coordinate descent to solve NCAA

To solve Problem (3.2), we use a standard two-block coordinate descent where each subproblem is solved with a fast projected gradient descent (FPGD) method, exactly as described in Section 2.1. In other words, A and H are alternatively optimized similarly to Algorithm 1 and the subproblems with respect to either A or H are solved in a similar way as in Algorithm 2. In the following, we discuss the initialization of the factors, the step sizes, the computation of the gradients and the projections.

3.3.1 Initialization of the factors

The initialization of A and H is performed as follows. To initialize A, we use SNPA (see Algorithm 3). More precisely, SNPA extracts r out of d columns in Y that stand for the initial archetypes $W^{(0)}$. Of course, this first requires to set up properly the matrix Y, see Section 3.4. SNPA retrieves a set $\mathcal{T}$ of r indices such that $Y(:, \mathcal{T}) = YA^{(0)} = W^{(0)}$. The initial matrix $A^{(0)}$ is then set up such that the only non-zero entries are $A^{(0)}(\mathcal{T}(k), k) = 1$ for $k = 1, \ldots, r$.
To initialize H, the following optimization problem is solved:

$$\begin{aligned} H^{(0)} = \arg\min_H \frac{1}{2} \|X - YA^{(0)}H\|_F^2 \\ \text{such that} \quad H \geq 0, \quad \sum_{k=1}^{r} H(k,j) \leq 1 \quad \text{for all } j = 1, ..., n. \end{aligned} \tag{3.15}$$

As already mentioned in Section 2.3.2, this is a convex optimization problem which can be solved easily with FPGD, for example through the implementation described in Appendix A of [54].

3.3.2 Gradient step

Once both $A^{(0)}$ and $H^{(0)}$ are set up, the 2-BCD is applied. To solve the subproblem with respect to A, that is,

$$\begin{gathered}\min_A \left(\mathscr{L}(A) = \frac{1}{2}\|X - YAH\|^2\right) \quad \text{such that} \\ A(l,k) \geq -\epsilon \quad \text{for all } k,l, \quad \sum_{l=1}^{d} A(l,k) = 1 \quad \text{for all } k = 1,\ldots,r,\end{gathered} \tag{3.16}$$

with a FPGD method, the gradient of $\mathscr{L}$ with respect to A needs to be computed:

$$\nabla\mathscr{L}(A) = -Y^T(X - YAH)H^T. \tag{3.17}$$

The step size of gradient descent is chosen through a backtracking line search. More precisely, we initialize the step size to $\frac{1}{L}$ where L is the Lipschitz constant of the problem. For the quadratic loss function of Problem 3.16, it is well-known that the Lipschitz constant can be computed as the spectral norm of the Hessian matrix of the loss function, that is, $L = \|\nabla^2\mathscr{L}(A)\|_2 = \|Y^TY \otimes HH^T\|_2 = \|Y^TY\|_2\|HH^T\|_2$ where $\otimes$ is the Kronecker product. Moreover, it is also known [91] that $\frac{1}{L}$ is a step size that guarantees decrease of the loss function at the gradient step. Starting with an initial step of $\frac{1}{L}$, we apply a backtracking line search strategy in order to find a step size that decreases as much as possible the loss function. More precisely, at the beginning of each iteration of FPGD applied to Problem 3.16, the step size is multiplied by 3. If the loss function decreases with the current step size, we use it. If it is not the case, we keep dividing the step size by 2 until the loss function $\mathscr{L}(A)$ decreases (up to a termination condition). The re-increase of the step size at the beginning of each iteration avoids a vanishing step size.

3.3.3 Projection step

A more challenging stage is the projection of the iterates on the feasible set, that is, line 4 of Algorithm 2 applied to Problem 3.16. Let us first notice that the constraints apply independently on each column of A, that is, we can perform column-wise projections. Let us call a any column of A satisfying the constraints, that is, **after** the projection on the feasible set and $\tilde{a}$ the output of the gradient step **before** the projection. Given $\tilde{a}$, we want to find the vector a as close as possible to $\tilde{a}$ satisfying the constraints, that is $a(l) \geq -\epsilon$ for $l = 1, \ldots, d$ and $\sum_{l=1}^{d} a(l) = 1$. This results in a new optimization problem:

$$\min_{a \in \mathbb{R}^d} \frac{1}{2} \|\tilde{a} - a\|^2 \quad \text{s.t.} \quad a(l) \geq -\epsilon \quad \text{for all } l = 1, \ldots, d, \quad \sum_{l=1}^{d} a(l) = 1. \tag{3.18}$$

The last constraint can also be rewritten as $e^T a = 1$ where e is a column vector of all ones. We can build the corresponding dual Lagrangian incorporating the equality constraint:

$$\max_{\mu} \min_{a \geq -\epsilon} \frac{1}{2} \|\tilde{a} - a\|^2 - \mu(1 - e^T a), \tag{3.19}$$

where μ is the Lagrangian multiplier and the notation $a \geq -\epsilon$ means that each entry of a is greater or equal to $-\epsilon$. If we suppose that the value μ^* of μ at optimality is known, the optimal solution of

$$\min_{a \geq -\epsilon} \frac{1}{2} \|\tilde{a} - a\|^2 - \mu^* (1 - e^T a) \tag{3.20}$$

can be easily computed and is given by

$$a^*(l) = \max(-\epsilon, \tilde{a}(l) - \mu^*) \quad \text{for all } l = 1, \ldots, d. \tag{3.21}$$

However, to find μ^*, we need to express the sum-to-one constraint, that is, solve the following equation, which is not trivial:

$$\sum_{l=1}^{d} \max(-\epsilon, \tilde{a}(l) - \mu^*) = 1. \tag{3.22}$$

First, let us sort the entries of $\tilde{a}$ in ascending order. Hence, in the remaining steps, we will assume that $\tilde{a}$ is sorted such that $\tilde{a}(1) \leq \tilde{a}(2) \leq \cdots \leq \tilde{a}(d)$. To solve Eq. (3.22), let us consider the function

$$S(\mu) = \sum_{l=1}^{d} \max(-\epsilon, \tilde{a}(l) - \mu), \tag{3.23}$$

sketched on Fig. 3.3. It can easily be shown that $S(\mu)$ is a monotonically decreasing piecewise linear function with a softer slope as μ increases. Indeed, let us observe that if μ has a very negative value, each $\tilde{a}(l) - \mu$ will be

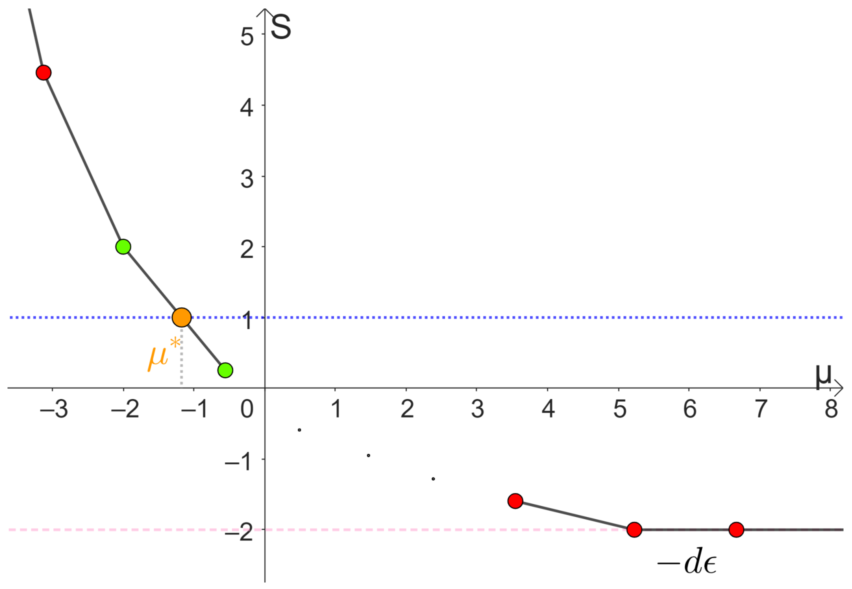


Figure 3.3: Illustration of the function $S(\mu)$ used in the projection of the factor A in NCAA.

greater than $-\epsilon$ and

$$S(\mu) = \sum_{l=1}^{d} \big(\tilde{a}(l) - \mu\big) = \sum_{l=1}^{d} \tilde{a}(l) - d\mu, \tag{3.24}$$

which is independent of ϵ. In fact, this happens for $\mu \le \mu_1 = \tilde{a}(1) + \epsilon$ (let us recall that $\tilde{a}$ has been sorted). On the other hand, if μ is very positive, each $\tilde{a}(l) - \mu$ will be negative and lower than $-\epsilon$, and $S(\mu) = -d\epsilon$. This happens for $\mu \ge \mu_d = \tilde{a}(d) + \epsilon$. Between these two extremes, a breakpoint, that is, a change in the slope, arises at $\mu_l = \tilde{a}(l) + \epsilon$ for any l. Indeed, the value of $S(\mu)$ within the interval $\big[\mu_l; \mu_{l+1}\big]$ is given by

$$S(\mu) = -l\epsilon + \sum_{i=l+1}^{d} (\tilde{a}(i) - \mu) \tag{3.25}$$

which is a linear decreasing function of slope $-(d-l)$. As l increases, μ_l increases and the slope of the corresponding linear piece becomes smaller in absolute value.

Therefore, given the two extreme points $(\mu_1, S(\mu_1))$ and $(\mu_d, S(\mu_d))$, we apply a dichotomy on the indices of $\tilde{a}$. If $S(\mu_1) < 1$, we solve

$$S(\mu^*) = \sum_{l=1}^{d} \tilde{a}(l) - d\mu^* = 1, \tag{3.26}$$

and μ^* is hence given by

$$\mu^* = \frac{\sum_{l=1}^{d} \tilde{a}(l) - 1}{d}. \tag{3.27}$$

In any other case, after fixing $l_{min} = 1$ and $l_{max} = d$, we compute the median index $l = \lfloor \frac{l_{min}+l_{max}}{2} \rfloor$, $\mu_l = \tilde{a}(l) + \epsilon$ and the corresponding value $S(\mu_l)$. Then, as in a classical dichotomy scheme, we compare $S(\mu_l)$ with 1. If $S(\mu_l) > 1$, we fix $l_{\min} = l$ while if $S(\mu_l) < 1$, we fix $l_{\max} = l$. The dichotomy procedure continues until either $S(\mu_l) = 1$ or $S(\mu_l) > 1 > S(\mu_{l+1})$ for some l. In this last case, the optimal value μ^* is given by $S^{-1}(1)$ on the interval $]\mu_l; \mu_{l+1}[$.

Once μ^* is found with this algorithm, the components of a^* can be completely computed through Eq. (3.21) and the projection is achieved for a given column of A.

The optimization of the second factor in NCAA, namely H, is performed the same way as its initialization (of course, by replacing $A^{(0)}$ and $H^{(0)}$ by the corresponding factors of the appropriate iteration), that is, by solving Problem (3.15) with the FPGD implementation of Appendix A of [54]. Let us mention that the dichotomy procedure described above could also be applied with $\epsilon = 0$ for the projection of H on its feasible set if the column-stochasticity constraint is not relaxed, that is, if we consider the original constraint $\sum_{k=1}^{r} H(k, j) = 1$ for all $j = 1, ..., n$.

3.3.4 Computational cost

Let us give a word about the computational cost of BCD with FPGD applied to NCAA. The optimization of both factors A and H through FPGD involves two main (from a computational point of view) stages : the computation of the gradient and the projection.

The computation of the gradient requires matrix multiplications (see Eq. (3.17) for the gradient with respect to A) whose computational cost is

$\mathcal{O}(mnd)$ operations where d is generally of the order of a small multiple of r.

The most costly stage in the projection of both A and H is to sort each column[2]. The matrix A has r columns of d elements while the matrix H has n columns of r elements, which requires respectively $\mathcal{O}(rd \log d)$ and $\mathcal{O}(nr \log r)$ operations for the sorting (performed with the "best" possible algorithm, that is, QuickSort).

Consequently, the cost of a single iteration of BCD is $\mathcal{O}(mnd)$ operations; the computation of the gradient is the most costly step. As long as d is chosen small enough (typically, as a small multiplicative factor of r), the computational cost of the algorithm remains linear in the dimensions of the input matrix (m and n) hence can be applied to large-scale data sets.

3.4 Possible strategies for NCAA parameters

The goal of this section is to discuss the choice of two important parameters of NCAA, namely the matrix Y in Section 3.4.1 and the number ϵ in Section 3.4.2.

3.4.1 Choice of Y

Let us recall that the matrix Y contains d columns, corresponding to the d points from which the archetypes are built (see Eq. (3.2)). We consider two main strategies for the design of Y:

1. SNPA: The matrix Y can be initialized by applying SNPA, that is, Algorithm 3, on the initial data matrix X. SNPA returns d extreme points of the dataset, which should roughly approximate the convex hull of X. The example on Fig 3.4 illustrates the use of SNPA for the design of Y in NCAA. More specifically, the $d = 6$ points of Y are computed with SNPA and according to Lemma 3.1, the $r = 3$ basis vectors lie within the near-convex hull of Y for some ϵ. However, a drawback of SNPA is that it can be sensitive to outliers.

2. HC: The matrix Y can also be initialized by applying HC, see Section 2.2. In this case, the d columns of Y correspond to points of

[2]In the worst case, the projection of H in [54] requires to sort the entries of each column of H, as for A.

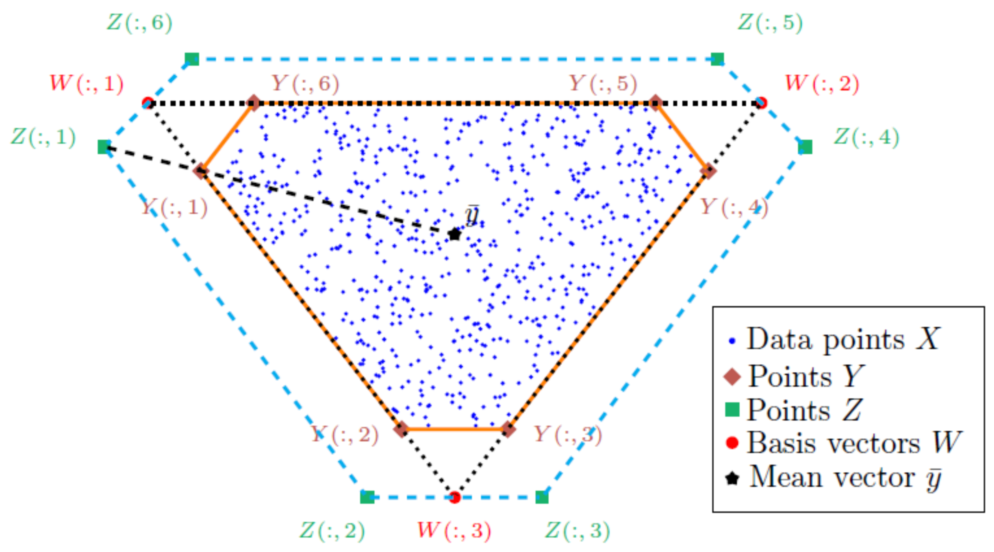


Figure 3.4: Geometric interpretation of NCAA for $r = 3$, $d = 2r = 6$, $m = 2$. According to Lemma 3.1, the estimated basis vectors are within the convex hull of the columns of Z, that is, the near-convex hull of the columns of Y. In this case, Y was set up with SNPA.

X that are rather central. As we want to allow the archetypes to lie outside the convex hull of the columns of X, SNPA might seem more adequate but experiments on real data showed that SNPA sometimes performs poorly (possibly due to noisy data points), and HC is therefore more adapted in these situations.

In addition to the method to build the points of Y, we also need to choose an appropriate value for d, the number of columns in Y. If d is too high, the computational load will increase and some points might be useless or redundant to build the archetypes. If d is too small, the archetypes could be not well spread and the model inefficient.

For SNPA, we keep adding points in Y as long as the ratio

$$\min_{\substack{H\geq 0\\ H^T e\leq e}} \frac{\|X - X(:,\mathscr{J})H\|_F}{\|X\|_F} \tag{3.28}$$

where $\mathscr{J}$ is the set of indices extracted by SNPA, is above a certain threshold (10^{-2}). In other words, SNPA keeps extracting points until the relative reconstruction error is sufficiently small.

For HC, although there exist some empirical guidelines for the ideal number of clusters, such as the CH index [19], such methods were not very successful when applied to our HC-based initialization. Hence, in our experiments, the value of d is fixed *a priori*, depending on the dataset. A more proper choice of d is let as a possible direction of future research. However, let us mention that in most NMF algorithms, the value of the rank r is already estimated either by trial and error or with experts insight hence, it does not seem to us that fixing d "by hand" is a major issue.

3.4.2 Choice of ϵ

The parameter ϵ intuitively indicates how far the archetypes are allowed to lie from the convex hull of X. If $\epsilon = 0$, NCAA boils down to AA (if $Y = X$) while if ϵ has a high positive value, the basis vectors might lie very far from the convex hull of the data points. In practice, choosing ϵ equal to a constant value is difficult as it is mostly data-dependent.

For these reasons, the parameter ϵ is tuned along the algorithm according to the following procedure. We first set a lower bound $\epsilon_{min} = 10^{-3}$ and an upper bound $\epsilon_{max} = 1$ for ϵ. Let us recall that the initial matrix of basis vectors $W^{(0)} = YA^{(0)}$ is created by applying SNPA on Y, for which the two techniques proposed in Section 3.4.1 provide points located within the convex hull of the columns of X, where the initial archetypes therefore also lie. We start by taking $\epsilon = \epsilon_{min}$ and run some iterations of NCAA with this value of ϵ. Then, we evaluate the objective function and update ϵ in consequence. Let us call:

- *init_obj*: the value of the loss function obtained with the initialization, that is, *init_obj* $= \frac{1}{2}\|X - YA^{(0)}H^{(0)}\|$,
- *curr_obj*: the value of the loss function at the end of the optimization process with the current value of ϵ,
- *prev_obj*: the value of the loss function at the end of the optimization process with the previous value of ϵ (at the first iteration *prev_obj* = *init_obj*).

The value of ϵ is updated according to Algorithm 5. If the reconstruction error with the current value of ϵ is sufficiently decreased compared to the previous error, ϵ is doubled (within the $\epsilon_{\max}$ upper bound). This situation, which is most likely to happen in the first iterations, means that the algorithm has not achieved to reach stable archetypes yet and it is worth allowing the archetypes to lie further away from the convex hull of the data points by increasing ϵ. Moreover, any lower value of ϵ would lead to a higher error, as the archetypes would be more constrained, that is why we update $\epsilon_{\min}$ to the current value of ϵ.

Algorithm 5 Global tuning of ϵ

```
1: if $\frac{|curr\_obj - prev\_obj|}{init\_obj} > \text{tol}$ then
2:     $\epsilon_{\min} = \epsilon$
3:     $\epsilon = \min(2\epsilon, \epsilon_{\max})$
4: else
5:     $\epsilon_{\max} = \epsilon$
6:     $\epsilon = \frac{\epsilon_{\min} + \epsilon_{\max}}{2}$
7: end if
```

On the other hand, if the error stagnates, it means that most of the data points can already be expressed as convex combinations of the archetypes with the current ϵ. Increasing ϵ would be useless, as it would only push the archetypes further away without a significant improvement of the loss function, apart from noisy points. Therefore, we update the upper bound such that $\epsilon_{\max} = \epsilon$. On the contrary, there might exist a smaller value of ϵ that does not increase the error by a lot but keeps the archetypes closer to the data cloud, hence we fix the next value of ϵ to $\frac{\epsilon_{\min}+\epsilon_{\max}}{2}$. The tuning stops when $\epsilon_{\max} - \epsilon_{\min} < 10^{-3}$ or when a maximum number of iterations is reached.

This procedure works well for "nice" data configurations, especially when the data exhibit some degree of symmetry as on Fig. 3.4, but quickly reaches its limits when the data is less structured. On Fig. 3.4, all the purity levels p_k's (see Section 2.4.1) are equal to 0.8. However, let us consider a situation where the purity levels are unbalanced between the basis vectors, such as the situation on Fig. 3.5, for $r = 3$. In this case, the p_k's are 0.9, 0.8 and 0.75 hence some true archetypes are closer to the data points than others. It is worth refining the NCAA model as follows: we modify the constraint $A(l,k) \geq -\epsilon$ for all k,l of Model (3.2) into $A(l,k) \geq -\epsilon_k$ where a different $\epsilon_k \geq 0$ is considered for each column of A. The other constraints remain unchanged.

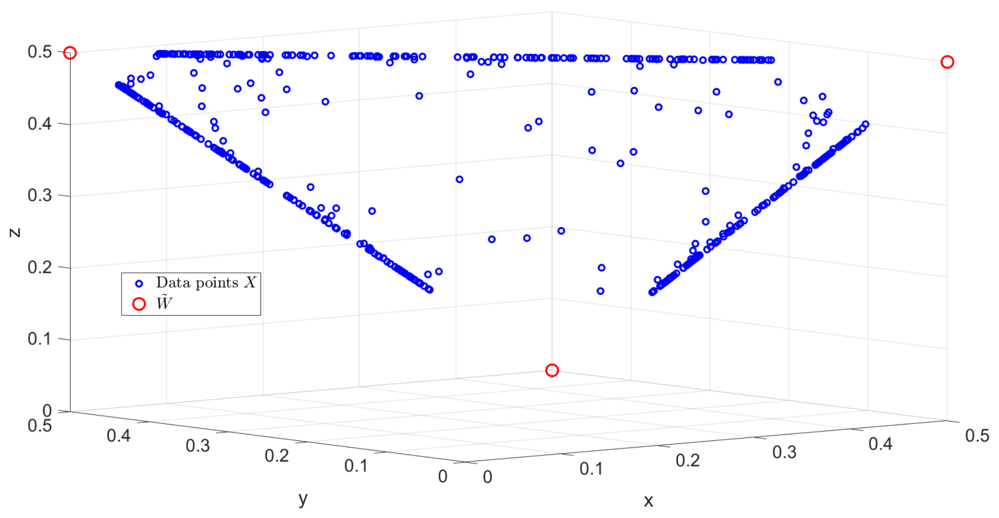


Figure 3.5: Illustration of a data set for which the purities are not identical across the $r = 3$ basis vectors.

The algorithm is modified as follows. The first stage of the algorithm involves a common ϵ for all the basis vectors and works the same way as previously described. Once the stopping criterion of this global tuning has been reached, a "fine-tuning" stage is performed for each archetype independently, as presented in Algorithm 6.

More precisely, starting from the absolute value of the minimum entry of the kth column of A, ϵ_k is progressively decreased by a factor α (fixed to 0.8 in the implementation) at each iteration until the reconstruction error becomes too high compared to the value of the error just after the global tuning. This technique aims at bringing the kth archetype as close as possible to the data cloud, keeping the others fixed, without degrading too much the loss function. For a given value of ϵ_k, a few iterations of alternated optimization of A and H are performed. Finally, as each column of A is tuned independently from each other, the matrix H needs to be updated once again at the end of the fine-tuning (line 17).

Algorithm 6 Fine tuning of ϵ_k's

```
Input: Matrices $A$ and $H$ resulting from the global tuning of $\epsilon$, $\text{err}^{(\epsilon)}$ the
    value of the objective function after the global tuning of $\epsilon$.
Output: Final matrices $A$ and $H$.
    final_$A = \{\}$
 1: for $k = 1 : r$ do
 2:     new_$A = A$
 3:     $\epsilon_{\text{curr},k} = -\min(A(:,k))$
 4:     while stop == false do
 5:         $\epsilon_{\text{curr},k} = \alpha\epsilon_{\text{curr},k}$
 6:         for $t = 1,2,\ldots$ do
 7:             Update new_$A(:,k)$ with FPGD on $\min \frac{1}{2}\|X - Y\text{new\_}AH\|_F^2$
                such that  new_$A(l,k) \geq -\epsilon_{\text{curr},k}$,  $\sum_{l=1}^{d} \text{new\_}A(l,k) = 1$
 8:             Update $H$ with FPGD on $\min_{H\geq 0} \frac{1}{2}\|X - Y\text{new\_}AH\|_F^2$
                such that  $\sum_{k=1}^{r} H(k,j) \leq 1$
 9:         end for
10:         if $\frac{\frac{1}{2}\|X - Y\text{new\_}AH\|_F^2}{\text{err}^{(\epsilon)}} > \text{tol}$ or $\epsilon_{\text{curr},k} < 10^{-10}$ then
11:             stop = true
12:         end if
13:     end while
14:     final_$A(:,k)$ = new_$A(:,k)$
15: end for
16: $A$ = final_$A$
17: Update $H$ with FPGD on $\min_{H\geq 0} \frac{1}{2}\|X - YAH\|_F^2$  s.t.  $\sum_{k=1}^{r} H(k,j) \leq 1$
    for all $j$
```

3.5 The NCAA algorithm

Now that we have detailed all the aspects of NCAA, let us summarize its whole working through Algorithm 7. After the initialization of both NCAA factors A and H, some iterations of alternated updates of A and H are performed through FPGD with ϵ equal to its lower bound. Then, the value of ϵ is updated depending on the evolution of the loss function and the alternated optimization scheme is repeated, always ending by an update of the value of ϵ. When a stopping criterion is fulfilled, a fine-tuning stage is applied for each archetype individually.

Algorithm 7 NCAA

```
Input: Nonnegative matrices $X \in \mathbb{R}_+^{m \times n}$ and $Y \in \mathbb{R}_+^{m \times d}$, rank $r$, bounds
    $0 < \epsilon_{\min} < \epsilon_{\max}$, tolerance tol.
Output: Matrices $A \in \mathbb{R}^{d \times r}$ and $H \in \mathbb{R}^{r \times n}$ that solve Problem (3.2).
 1: Compute initial matrices $A^{(0)}$ and $H^{(0)}$, $i = 0$, $\epsilon^{(1)} = \epsilon_{\min}$
 2: $\text{err}^{(0)} = \frac{1}{2}||X - YA^{(0)}H^{(0)}||_F^2$
 3: for $t = 1, 2, \dots$ do
 4:     for $u = 1, 2, \dots$ do
 5:         $i = i + 1$
 6:         $A^{(i)}$ = Update $A$ with FPGD on $\min \frac{1}{2}\|X - YAH^{(i-1)}\|_F^2$
            such that  $A(l,k) \geq -\epsilon$,  for all $k, l$;  $\sum_{l=1}^{d} A(l,k) = 1$  for all $k$
 7:         $H^{(i)}$ = Update $H$ with FPGD on $\min_{H \geq 0} \frac{1}{2}\|X - YA^{(i)}H\|_F^2$
            such that  $\sum_{k=1}^{r} H(k,j) \leq 1$  for all $j$
 8:     end for
 9:     $\text{err}^{(t)} = \frac{1}{2}||X - YAH||_F^2$
10:     if $\frac{|\text{err}^{(t)} - \text{err}^{(t-1)}|}{\text{err}^{(0)}} < \text{tol}$ then
11:         $\epsilon_{\max} = \epsilon^{(t)}$;  $\epsilon^{(t+1)} = \frac{\epsilon_{\min} + \epsilon_{\max}}{2}$
12:     else
13:         $\epsilon_{\min} = \epsilon^{(t)}$;  $\epsilon^{(t+1)} = \min(2\epsilon^{(t)}, \epsilon_{\max})$
14:     end if
15: end for
16: Apply the fine-tuning procedure of Algorithm 6
```

3.6 NCAA applied to synthetic data

Let us first illustrate the performance of NCAA on a simple synthetic dataset. Let us take the configuration presented on Fig. 3.5, that is, a data matrix X of $n = 500$ points in dimension $m = 3$ where the ground-truth matrix of basis vectors is given by:

$$\tilde{W} = \begin{pmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0.5 \end{pmatrix}, \tag{3.29}$$

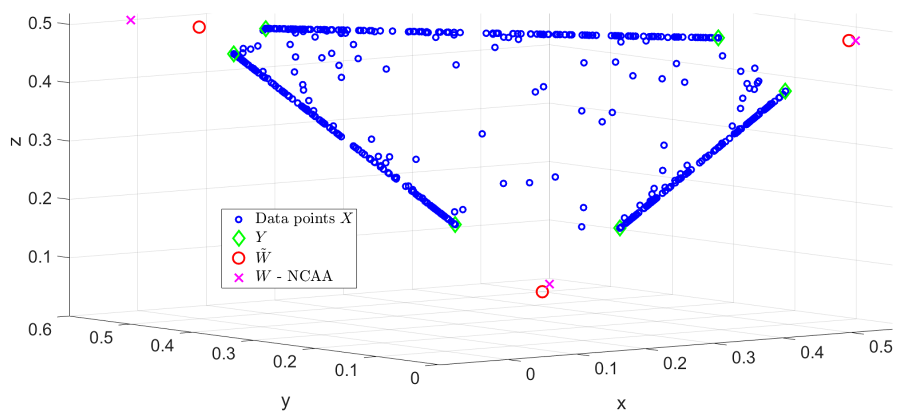
and $\tilde{H}$ is generated through a Dirichlet distribution, see Section 2.4.1 for more details. Moreover, the purities p_k's of the $r = 3$ archetypes are respectively 0.75, 0.8 and 0.9. We build the matrix Y with SNPA and use $d = 6$.

Unsurprisingly, the NCAA model applied with only the global tuning of ϵ does not provide satisfying results. More precisely, the relative error on the matrix of basis vectors $\frac{\|W-\tilde{W}\|_F}{\|\tilde{W}\|_F} = 6.39\%$. When the fine-tuning is applied column by column, the matrix W retrieved by NCAA is closer to $\tilde{W}$ and the relative error is only 3.35%. This difference is illustrated on Fig. 3.6.

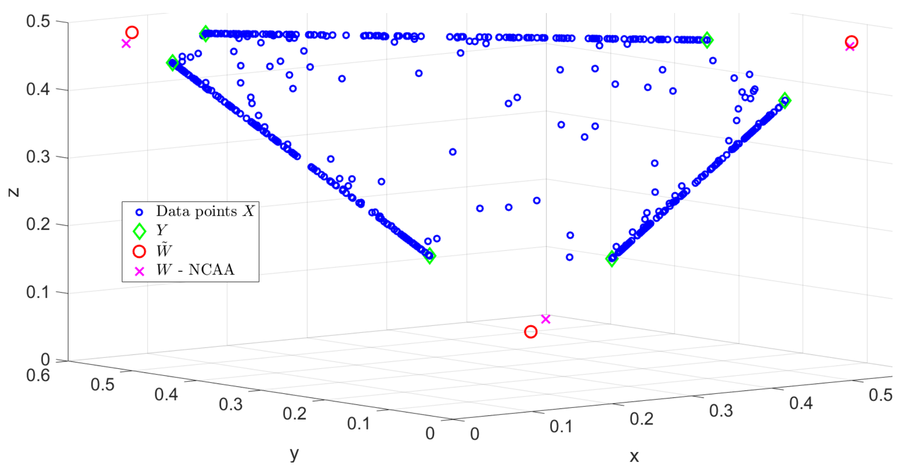
More systematic tests were also conducted. We generate $n = 1000$ data points in dimension $m = 10$ and compare our NCAA model with minVol-NMF with the "logdet" penalty term, that is, Eq. (2.5) for which the values 0.01 and 0.1 of the regularization parameter $\tilde{\lambda}$ are considered. For NCAA, Y is always set up with SNPA and $d = 10r$. However, we vary the rank, the (symmetric) purity level and the relative additive white Gaussian noise level ν (see Section 2.4.1) as follows:

- The rank r is set to either 3, 7, 12, 20 (note that the two last possibilities correspond to rank-deficient factorizations, that is , $m < r$),
- The purity level p, equal for all the r archetypes, is set to either 0.7, 0.8, 0.9, 1,
- The relative noise level ν is set to either 0, 0.01, 0.05, 0.1, 0.2.

Given the value of the rank, each entry of the ground-truth matrix $\tilde{W}$ is drawn uniformly within the interval $[0,1]$. Then, each column is normalized so that its entries sum to one. The columns of $\tilde{H}$ are drawn from a



(a)



(b)

Figure 3.6: NCAA applied to synthetic data with non-symmetric purities; $d = 6$ and Y obtained with SNPA. The $r = 3$ basis vectors retrieved by NCAA are closer to the ground truth when a fine-tuning is applied (Fig. (b)) than with the sole global tuning of ϵ (Fig. (a)).

Dirichlet distribution as previously described. For every possible combination of the variable parameters r, p and v, we perform 25 runs (with each time a new generation of the ground-truth factors $\tilde{W}$ and $\tilde{H}$) of each compared algorithm and compute the mean and the standard deviation of the corresponding MRSA over the 25 runs. These results are presented in Table 3.1 for some interesting configurations, to facilitate the analysis: we fix

$r = 7$, $p = 0.8$ and $\nu = 0$ and vary one parameter at a time. Moreover, we also consider SNPA as a compared algorithm. Let us recall that SNPA is originally dedicated to solve (near) separable NMF (see Section 2.2), hence it is interesting to compare NCAA and SNPA especially when the purity level is close to 1. In addition, we also display between parentheses for each configuration the number of times (over the 25 runs) that a given model is "the best", that is, produces the lowest MRSA. We observe the following:

- The variability of the settings generates in general high standard deviations. However, the ranking trend given by the average MRSA is confirmed by the distribution of the best runs.

- The MRSA of NCAA is in most cases lower than the one of minVolNMF, which indicates that there is less error between the estimated basis vectors and the expected ones in NCAA than in minVolNMF. Note that minVolNMF with the two values of λ give similar results. The baseline SNPA is only competitive in separable cases (that is, when $p = 1$). NCAA performs particularly well in the difficult scenarios, namely when $r > m$, in presence of heavy noise or in highly mixed situations ($p \ll 1$). As opposed to minVolNMF, NCAA uses the data points to construct the basis vectors hence is much more robust in these difficult scenarios.

In summary, NCAA performs very well on synthetic data and is more than competitive with state-of-the-art approaches such as minVolNMF and SNPA.

3.7 NCAA applied to hyperspectral unmixing

To evaluate our NCAA model on real-world data, we apply it to the Urban hyperspectral image (see Fig. 1.3 in Section 1.3) and compare NCAA to the logdet variant of minVolNMF, that is, the model described by Eq. (2.5), with the initial regularization parameter $\tilde{\lambda}$ fixed to 0.1. This value of $\tilde{\lambda}$ follows from the recommandations of [77]. We set the rank $r = 4$ for all models, that is, we aim to extract 4 materials in the Urban image.

The matrix Y of NCAA is set up with $d = 20$ points obtained with the hierarchical clustering technique, as described in Section 3.4. Other values of d have been tested but lead to poorer results.

(p, r, v)	NCAA	minVolNMF ($\lambda = 0.01$)	minVolNMF ($\lambda = 0.1$)	SNPA
(*0.7*, 7, 0)	**1.13** ± 2.61 (24)	7.42 ± 5.22 (0)	6.09 ± 5.30 (1)	15.04 ± 2.40 (0)
(*0.8*, 7, 0)	**0.37** ± 0.61 (24)	1.99 ± 2.27 (0)	1.70 ± 2.25 (1)	7.40 ± 1.20 (0)
(*0.9*, 7, 0)	**0.21** ± 0.07 (20)	0.45 ± 0.23 (0)	0.41 ± 0.23 (5)	3.13 ± 0.28 (0)
(*1*, 7, 0)	$2.12 \cdot 10^{-3} \pm 4.27 \cdot 10^{-3}$ (8)	$3.18 \cdot 10^{-3} \pm 6.56 \cdot 10^{-3}$ (0)	$3.17 \cdot 10^{-3} \pm 6.53 \cdot 10^{-3}$ (0)	$\mathbf{1.22} \cdot 10^{-5} \pm 1.40 \cdot 10^{-5}$ (17)
(0.8, *3*, 0)	1.88 ± 1.05 (10)	1.73 ± 0.90 (1)	**1.47** ± 0.95 (14)	7.16 ± 0.80 (0)
(0.8, *7*, 0)	**0.37** ± 0.61 (24)	1.99 ± 2.27 (0)	1.70 ± 2.25 (1)	7.40 ± 1.20 (0)
(0.8, *12*, 0)	**3.81** ± 3.97 (23)	5.70 ± 3.80 (0)	5.44 ± 3.80 (2)	10.08 ± 2.53 (0)
(0.8, *20*, 0)	**6.39** ± 2.41 (22)	7.20 ± 2.48 (0)	7.09 ± 2.47 (3)	10.45 ± 1.79 (0)
(0.8, 7, *0*)	**0.37** ± 0.61 (24)	1.99 ± 2.27 (0)	1.70 ± 2.25 (1)	7.40 ± 1.20 (0)
(0.8, 7, *0.01*)	2.31 ± 3.11 (7)	2.44 ± 3.07 (0)	**2.16** ± 3.03 (18)	7.85 ± 1.98 (0)
(0.8, 7, *0.05*)	6.32 ± 2.20 (6)	6.48 ± 2.50 (0)	**5.59** ± 2.39 (19)	10.35 ± 2.94 (0)
(0.8, 7, *0.1*)	**8.44** ± 1.73 (14)	11.02 ± 3.78 (0)	9.43 ± 3.56 (11)	12.18 ± 2.16 (0)
(0.8, 7, *0.2*)	**13.87** ± 3.30 (22)	23.23 ± 3.90 (0)	21.19 ± 4.12 (1)	18.79 ± 2.29 (2)

Table 3.1: Comparison of the performances of NCAA, minVolNMF and SNPA on synthetic data, with $n = 1000$, $m = 10$, $d = 10r$ in function of the purity level, rank and noise level respectively. The mean and standard deviation of the MRSA over 25 runs with randomly generated true factors is reported. For each configuration, the best average MRSA is highlighted in bold and the number of times each algorithm performs the best is between parentheses.

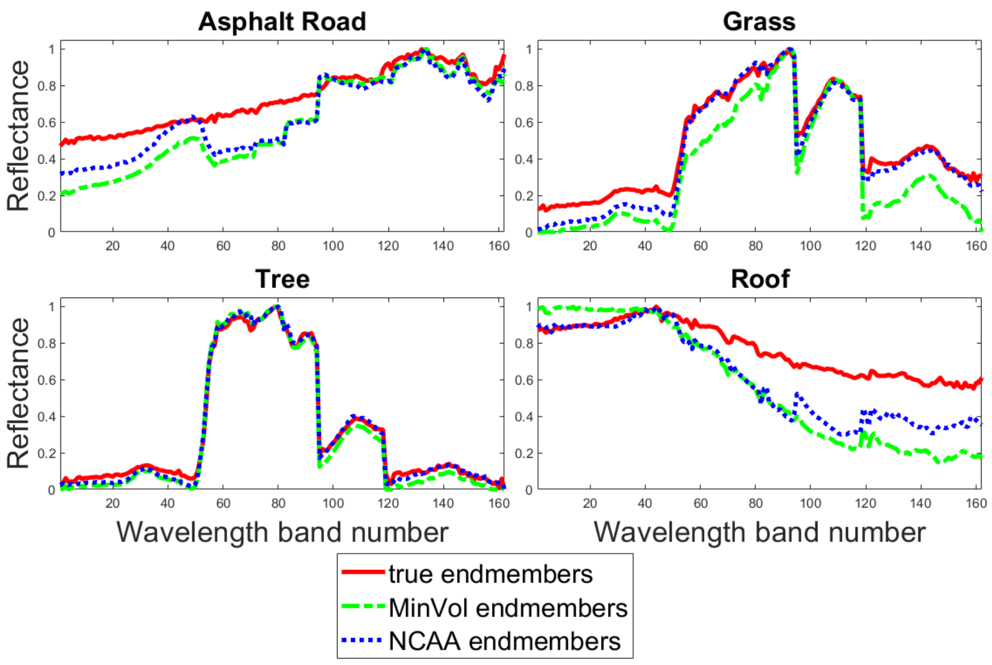
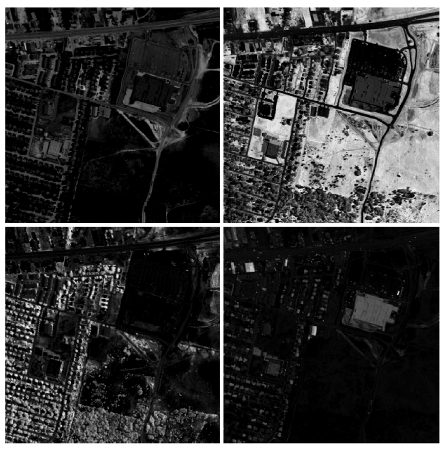


Figure 3.7: Comparison of the spectral signatures of the endmembers extracted by NCAA, minVolNMF and the ground truth in the Urban image with $r = 4$.

To quantify the performances of the models, we compute the MRSA (see Section 2.4) between the matrix of basis vectors W estimated by each model (let us recall that for NCAA, $W = YA$, see Eq. (3.3)) and the matrix of ground-truth basis vectors $\tilde{W}$ whose columns correspond to the spectral signatures of Fig. 1.4a.

On Fig. 3.7, we compare the three sets of spectral signatures, that is, the ground truth of [135], and the columns of the matrices W's obtained with on the one hand minVolNMF and on the other hand NCAA (with fine-tuning of ϵ_k's). It appears that the endmembers extracted by NCAA have spectral signatures very close to the ground truth. Similarly, the corresponding abundance maps presented on Fig. 3.8 show that NCAA is able to retrieve meaningful proportions of each endmember in the pixels of the initial image, with the same level of quality as minVolNMF.

In Table 3.2, we report the MRSA obtained on the Urban image with minVolNMF, NCAA with fine-tuning (NCAA-FT) but also NCAA without


(a)

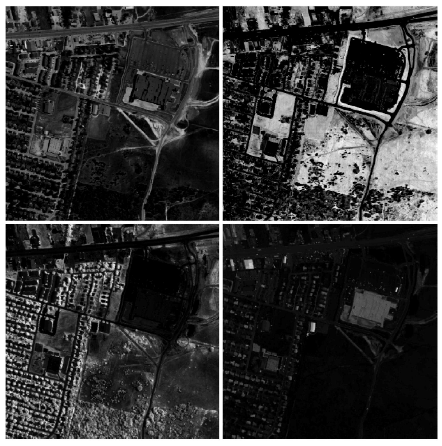

(b)

Figure 3.8: Abundance maps extracted by (a) NCAA and (b) minVolNMF in the Urban image with $r = 4$. From left to right, on top: road, grass; on bottom: tree, roof.

fine-tuning (NCAA-raw), as well as the corresponding runtime. For a fair comparison, we run 500 iterations[3] for all methods. Note that, as mentioned in Section 3.4.2, the tuning stages of ϵ have their own stopping criteria, which might be more appropriate outside the context of the comparison with other methods. The time performance is not a motivation of the proposed NCAA model and highly depends on the parameters of the tuning stages (for example the choice of the parameter α in Algorithm 6). However, NCAA is typically much slower than minVolNMF. On the other hand, the advantage of the fine-tuning in terms of performance seems obvious, despite the increase of computational cost. Indeed, NCAA-FT performs better than minVolNMF in terms of MRSA while NCAA-raw gives rather poor results although being faster. This is due to the fact that using a single ϵ is not sufficient for such a dataset since some endmembers can be further from the data cloud than others.

[3]By iteration, we mean one update of all the factors involved.

Algorithm	MRSA	Runtime (s)
minVolNMF	5.72	233.8
NCAA-FT	5.57	2875.5
NCAA-raw	10.0	1286.4

Table 3.2: MRSA and runtime of minVolNMF and NCAA for the unmixing of Urban image.

3.8 Perspectives of improvement

This section lists some perspectives of improvement for NCAA.

First, let us mention that our model has a drawback: the basis vectors $W = YA$ are not necessarily nonnegative. Indeed, although the matrix Y contains only nonnegative entries if a proper strategy such as SNPA or HC is used, A can contain slightly negative entries (up to $-\epsilon$). By consequence, nothing prevents the columns of W to contain negative entries. However, this drawback can be more or less alleviated by the tuning strategy of ϵ. Indeed, since we start with ϵ close to 0 and progressively increase it to allow the archetypes to lie further away from the convex hull of the data, a control of the nonnegativity of the basis vectors can be done at each incrementation of ϵ to allow an "early stopping" if needed (that is, if a basis vector is no longer nonnegative).

A key aspect that would be particularly interesting to study is the identifiability of NCAA, that is, the conditions under which the NCAA factors are unique, up to scaling and permutations. Many recent works have studied the identifiability of NMF [51, 52], and especially minVolNMF [82]. We obtained some preliminary results (not published) on very specific settings but since, more broadly, identifiability is not the core of this PhD, we did not investigate much these aspects.

We only compared NCAA with the minVolNMF formulation of Eq. (2.5). However, robust approaches, relaxing some constraints on the factors, exist (see Section 2.3.2) and may be interesting to compare with NCAA as well; this is a topic for further research.

Some variants of NCAA, adding a penalty term to the objective function instead of a constraint on the matrix A, could also be investigated. For example, it could be interesting to consider a model whose loss function is $\frac{1}{2}\|X - YAH\|_F^2 + \lambda \sum_{k=1}^{r} \max_{l}\big(-\min(0, A(l,k))\big)$. In other words, this penalty would take into account the sum of the most negative entry (if any) of every column of A. Of course, a key challenge is to use a proper strategy to set the value of the hyperparameter λ.

Let us mention a last interesting possibility of improvement, which is to include some degree of sparsity in the matrix A. Indeed, to enhance the interpretability of NCAA, we could enforce that each archetype is only combination of a small number of columns of Y.

3.9 Take-home messages

In this chapter, we introduced a new model called near-convex archetypal analysis (NCAA). Let us summarize the main aspects of this model:

- NCAA is a flexible variant of a well-known NMF model, archetypal analysis (AA), which allows the basis vectors to lie slightly outside the convex hull of the data points,
- We have developed an optimization framework based on two-block coordinate descent and fast projected gradient descent to solve NCAA,
- NCAA performs better than minimum-volume NMF on synthetic data, especially in "difficult" settings, for example, in the presence of heavy noise or in rank-deficient factorizations, as well as on the hyperspectral unmixing task,
- Several perspectives of improvement of NCAA have been identified, such as the introduction of sparsity in the model.

Part II

Deep Matrix Factorizations

CHAPTER 4

A GENTLE INTRODUCTION TO DEEP MATRIX FACTORIZATIONS

Deep matrix factorizations (deep MF) is the core topic of this PhD. In this chapter, we review the main aspects of deep MF, covered by our survey paper [38]. In Section 4.1, we describe the main motivations of deep MF compared to single-layer MF. Then, in Section 4.2, we present the main models in a historical perspective, as well as important variants. Section 4.3 briefly describes how to solve deep MF problems and choose the parameters. In Section 4.4, we review the main real-world applications of deep MF and especially illustrate its working on three showcase examples. We present the main theoretical results in Section 4.5, including the connections with neural networks, while in Section 4.6, we propose directions of future research; some of them are explored in the next chapters. Finally, Section 4.7 gathers the main take-home messages.

4.1 Motivations of deep MF

The interest of low-rank matrix approximations has already been highlighted in the previous chapters: it allows to extract relevant information from large data sets by expressing each data point as a linear combination of a few features. On the other hand, deep neural networks have been extensively studied over the last decades as deep learning gained success in many supervised classification tasks and even in generative models. Their main advantage lies in the ability to combine features in a highly non-linear way but the inner machinery of deep neural networks can be hard to grasp.

The main motivation of deep MF is to combine both interpretability, as in classical matrix factorizations, of which it is an extension, and the extraction of multiple hierarchical features, as in deep neural networks. The goal of deep MF is to decompose a data matrix $X \in \mathbb{R}^{m\times n}$ as

$$\begin{aligned} X &\approx W_1 H_1, \\ H_1 &\approx W_2 H_2, \\ &\vdots \\ H_{L-1} &\approx W_L H_L, \end{aligned} \tag{4.1}$$

where L is the number of layers, $W_l \in \mathbb{R}^{r_{l-1}\times r_l}$ with $r_0 = m$, $H_l \in \mathbb{R}_+{}^{r_l\times n}$ for $l = 1,\dots,L$.

Each matrix W_l ($l = 1,\dots,L$) can be interpreted as the feature matrix of layer l and each H_l can be interpreted as the representation matrix of layer l. In other words, successive factorizations of rank r_l ($1 \le l \le L$) are performed such that various recombinations of the features of the first layers would appear in the following ones, allowing numerous interpretations of the semantics hidden in the data set. While standard MFs decompose the data matrix in only two factors, deep MF is able to extract several layers of features in a hierarchical way, giving new insights in a broad range of applications. Overall, the data matrix X is approximated as

$$X \;\approx\; W_1 W_2 \cdots W_L H_L. \tag{4.2}$$

Without any constraint on the factors of deep MF, Eq. (4.2) merely degenerates into classical matrix factorization. Indeed, in this case, the product of the matrices W_l's could be replaced by a single equivalent (that is, without additional particular property) matrix whose rank is less than or equal to the minimum of the r_l's and the factorization is highly non-unique. One could simply replace any W_l by $W_l Q$ and W_{l+1} by $Q^{-1} W_{l+1}$ for any l and any invertible matrix $Q \in \mathbb{R}^{r_l\times r_l}$, and obtain another decomposition of X with the same approximation error but most likely a different interpretation. Therefore, constraints on the factors such as nonnegativity and sparsity, and/or regularizations should be considered, which results in various deep MF models. Most deep MF models assume the nonnegativity of several factors of the decomposition and therefore extend NMF ideas.

4.2 Deep MF models

In this section, we first present the evolution from the early multilayer models to the recent deep models in Section 4.2.1. Then, in Section 4.2.2, we describe the main variants, which are inspired by those of classical matrix factorizations.

4.2.1 A brief history of "deep" factorizations

The first model extending constrained low-rank matrix factorizations to several levels is multilayer NMF proposed by Cichocki et al. in 2006 [26, 27]. Based on the hierarchical factorizations of a nonnegative data matrix $X \in \mathbb{R}_+^{m\times n}$ as described in Eq. (4.1), multilayer NMF decomposes X in a sequential manner. At the first layer, an NMF of rank r_1 of X is computed through Algorithm 1 such that $X \approx W_1 H_1$. At the second layer, the matrix H_1 is factorized as $H_1 \approx W_2 H_2$, and so on until H_{L-1} is decomposed as $W_L H_L$; see Algorithm 8.

Algorithm 8 Early multilayer NMF [26]

```
Input: Nonnegative data matrix X, number of layers $L$, inner ranks $r_l$'s for
    $l = 1, \dots, L$.
Output: Matrices $W_1, \cdots, W_L$ and $H_1, \cdots, H_L$.
1: $H_0 = X$
2: for $l = 1, \dots, L$ do
3:     $(W_l, H_l)$ = Algorithm 1 $(H_{l-1}, r_l)$
4: end for
```

However, multilayer NMF does not investigate much the hierarchical power of deep schemes as the decomposition is purely sequential, that is, multilayer NMF is equivalent to a succession of single layer NMFs, with a single forward pass. More precisely, Algorithm 8 consists in sequentially minimizing the reconstruction errors $\| H_{l-1} - W_l H_l \|_F^2$ for all $l = 1, \dots, L$ with

$H_0 = X$, but it does not involve a global loss function. In other words, the error is minimized layer by layer, but there is no retroaction of the last layers on the first ones.

A key improvement was achieved by the papers of Trigeorgis et al., who introduced deep MF [114, 115]. The data matrix X still undergoes hierarchical factorizations as in Eq. (4.1), but the breakthrough lies in the iterative updates of the factors. As opposed to multilayer MF, deep MF not only propagates the information from the first, more abstract layer, to the last, more refined layer, but also propagates the information in the reverse direction. The following error function involving the factors of all layers is considered:

$$\mathscr{L}(W_1, W_2, \cdots, W_L; H_L) = \frac{1}{2}\|X - W_1 W_2 \cdots W_L H_L\|_F^2, \tag{4.3}$$

and a block-coordinate descent strategy is used to iteratively update all the factors. The deep MF algorithm [115] is described in Algorithm 9, and illustrated on Fig. 4.1c. In Algorithm 9, *arg reduce* means that the factor is updated through some algorithm that (typically) decreases the objective function, such as FPGD.

Algorithm 9 Deep MF [115]

```
Input: Data matrix X, number of layers $L$, inner ranks $r_l$'s for $l = 1, \ldots, L$.
Output: Matrices $W_1, \cdots, W_L$ and $H_1, \cdots, H_L$.
1: Compute initial matrices $W_l^{(0)}$ and $H_l^{(0)}$ for all $l$
2: for $k = 1, \ldots$ do
3:     for $l = 1, \ldots, L$ do
4:         $A_l^{(k)} = \prod_{j<l} W_j^{(k)}$
5:         $B_l^{(k)} = \begin{cases} H_L^{(k-1)} & \text{if } l = L \\ W_{l+1}^{(k-1)} H_{l+1}^{(k-1)} & \text{otherwise} \end{cases}$
6:         $W_l^{(k)} = \arg\operatorname*{reduce}_{W} \frac{1}{2}\|X - A_l^{(k)} W B_l^{(k)}\|_F^2$
7:         $H_l^{(k)} = \arg\operatorname*{reduce}_{H \geq 0} \frac{1}{2}\|X - A_l^{(k)} W_l^{(k)} H\|_F^2$
8:     end for
9: end for
```

Several comments can be formulated:

- First, the work of Trigeorgis et al. was inspired by semi-nonnegative matrix factorization (semi-NMF) [41], a variant of NMF where only one factor, typically H, must contain nonnegative entries while W is allowed to contain mixed-sign elements. Therefore, this model should rather be called deep semi-NMF as the W_l's are not directly constrained to be nonnegative, and the matrix X is not required to have nonnegative entries neither.

  In practice however, as most physical systems record nonnegative data, it often makes sense to impose nonnegativity of the basis vectors as well. In this case, one can easily modify the model by adding nonnegativity constraints on the W_l's through line 7 of Algorithm 9.

- The attentive reader may have noticed that Algorithm 9 does not correspond to applying a BCD method on Eq. (4.3) by optimizing the factors $(W_1, \cdots, W_L, H_L)$ alternatively. In fact, the nonnegative matrices H_l for $l = 1, \ldots, L-1$ are intermediate variables that do not appear in Eq. (4.3). However, one needs to remember the underlying decompositions of Eq. (4.1): as $H_l \approx W_{l+1} H_{l+1}$ for $l = 1, \ldots, L-1$ are constrained to be nonnegative, they have a dedicated update rule.

- Finally, the choice of the loss function itself is not obvious. Is a loss function of the type $D(X, W_1 W_2 \cdots W_L H_L)$, where $D(A, B)$ is a distance measure between two matrices A and B, a good choice, as in Eq. (4.3) ? Or would a loss function that balances the contribution of each layer, such as

$$D(X, W_1 H_1) + \lambda_1 D(H_1, W_2 H_2) + \cdots + \lambda_{L-1} D(H_{L-1}, W_L H_L),$$

  be more appropriate ? Moreover, most works in the deep MF literature have only considered the Frobenius norm. Alternatives such as the Kullback-Leibler and Itakura-Saito divergences, which have been shown to be particularly appropriate for specific applications in the case of standard NMF [49, 78], have not been investigated yet. More detailed discussions about deep MF loss functions are present in Chapter 5, where we introduce two new loss functions for deep MF.

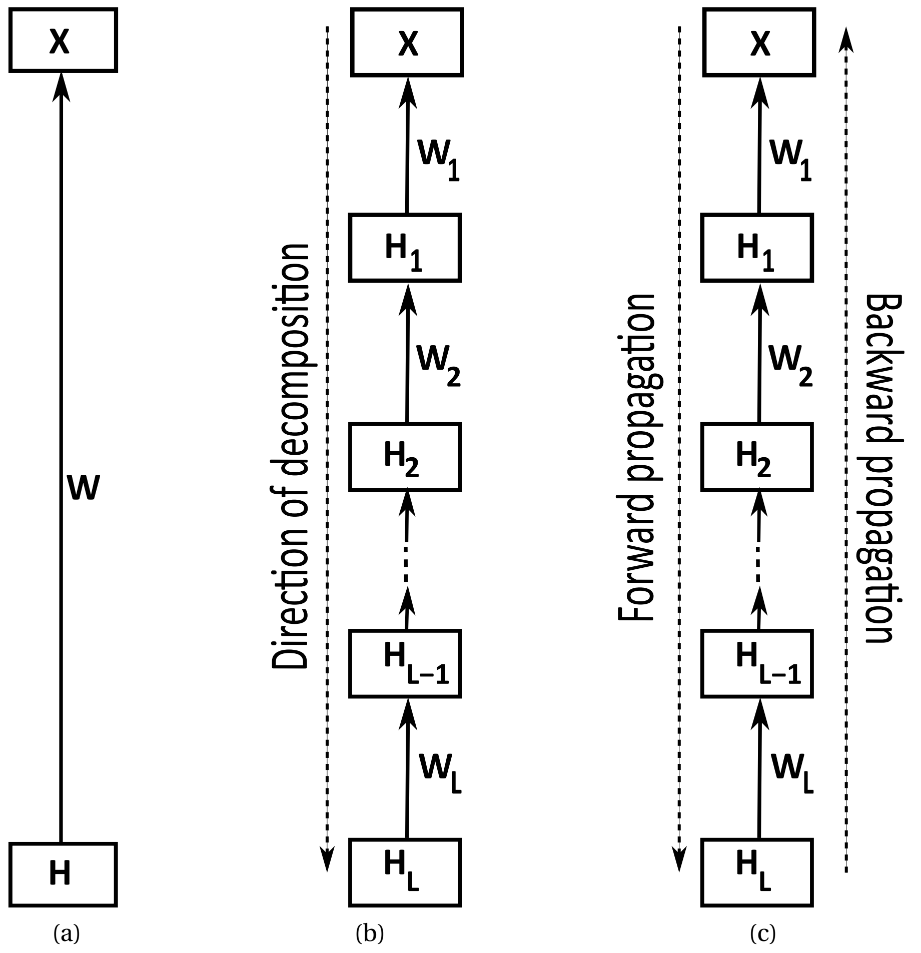


Figure 4.1: Comparison of (a) MF, (b) multilayer MF [26] and (c) deep MF [115]. An arrow means that a matrix multiplication is performed: $\boxed{H}\xrightarrow[W]{}\boxed{X}$ means that H is multiplied by W to approximate X.

To sum up, a comparison of one-layer matrix factorization, multilayer MF [26] and deep MF [115] is illustrated on Fig. 4.1. Multilayer MF on Fig. 4.1b and deep MF on Fig. 4.1c both perform several levels of decomposition but the key difference is the iterative nature of the update rules in deep MF, while the decomposition is only sequential in multilayer MF. Note that the *model* is the same for deep and multilayer MF, but the *algorithms* used to solve them are different.

4.2.2 Main deep MF variants

Beside the standard models presented in Section 4.2.1, some variants have been studied in the recent literature. These variants consist in adding constraints on the factors or a regularization term to the loss function. In this section, we briefly review some of these models. In many of them, non-negativity is assumed on the factors, and the variants are therefore closely related to NMF models.

4.2.2.1 Deep orthogonal NMF

The deep version of ONMF (see Section 2.3.1 for the single-layer model) was introduced in [85] and enriched in [101]. The decomposition is slightly different than in the multilayer and deep MF described above because rather than having the activations matrices H_l's successively decomposed, it factorizes the features matrices W_l's:

$$\begin{aligned} X &\approx W_1 H_1, \\ W_1 &\approx W_2 H_2, \\ &\vdots \\ W_{L-1} &\approx W_L H_L, \end{aligned} \tag{4.4}$$

leading to $X \approx W_L H_L \cdots H_1$, with each H_l constrained to be nonnegative and row-wise orthogonal, that is, $H_l \geq 0$ and $H_l H_l^T = I_{r_l}$ for all l. By doing so, for a given l, each column $H_l(:,k)$ has at most one non-zero entry, say the $j_k^{(l)}$th. Consequently, each layer of deep ONMF can be interpreted as a hard clustering: the r_{l-1} columns of W_{l-1} are spread among $r_l \leq r_{l-1}$ clusters. More precisely, at layer l, the kth column of W_{l-1} is assigned to the $j_k^{(l)}$th cluster whose centroid is $W_l(:, j_k^{(l)})$.

Applying the successive decompositions over the basis matrices W_l's rather than the activations matrices H_l's as in [115] seems more natural: it allows to directly interpret the basis vectors of a given layer as combinations of the basis vectors of the next layer. More insights on deep ONMF are given in Section 6.1.

ONMF admits a relaxation called approximately orthogonal NMF (AONMF) [79]. Instead of enforcing orthogonality constraints, a penalty term of the form $\sum\limits_{j<k} H(j,:)H(k,:)^T$ is added to the reconstruction error.

Similarly, deep AONMF adds a penalty to the objective that minimizes the inner products $H_l(j,:)H_l(k,:)^T$, for all $j \neq k$ at each layer l.

4.2.2.2 Deep sparse MF

A very common constraint considered in low-rank factorizations is the sparsity of some factors, that is, limiting the number of non-zero elements of W and/or H. Many papers have studied the case of one-layer sparse MF, see for example [63, 43, 68, 29] among others. The goal of sparse MF is to render the factors more interpretable. In particular, if each column of H contains only a few non-zero entries, each data point is associated to only a few basis vectors.

The extension of sparse NMF to the deep setting was proposed in [60]. An ℓ_1 norm penalty is considered either on each column of the matrices W_l's and/or on each column of the matrices H_l's. Similarly to shallow sparse MF, the goal of sparse deep MF is to obtain sparse and easily interpretable factors at each layer. Note that a normalization of the other factor should be used to avoid a pathological case where the entries of the factor for which sparsity is promoted tend to zero while those of the other factor tend to infinity, because of the scaling degree of freedom in such decompositions ($W_l H_l = (\alpha W_l)(H_l/\alpha)$ for any $\alpha > 0$).

Inspired by deep learning, dropout could also promote sparsity. Dropout [110] consists in randomly "dropping" some activations during the learning process to improve generalization. It has recently been employed for one layer NMF [61], and was shown to be equivalent to a deterministic low-rank regularizer [20]. It would be interesting to see to what extent dropout might regularize deep MF as well.

4.2.2.3 Deep archetypal analysis

Archetypal analysis (AA), largely described in Chapter 3, was also extended to a deep version. To the best of our knowledge, the first proposal of deep AA was made in [109] for the acoustic scene classification task. Given a data matrix X composed of n temporal frames characterized by an m-dimensional features vector, a discriminative representation is learnt through successive AA decompositions performed in a greedy forward way:

$$\begin{aligned} X &\approx XA_1H_1, \\ H_1 &\approx H_1A_2H_2, \\ &\vdots \\ H_{L-1} &\approx H_{L-1}A_LH_L. \end{aligned}$$

However, schemes including a "backpropagation" stage do not seem to have been tested yet.

4.2.2.4 Semi-supervised settings

While the models presented so far were all unsupervised, that is, do not require any labeled data, some deep MF models are able to cope with available prior information in a semi-supervised fashion, such as deep weakly-supervised semi-NMF [115]. To handle side information, a weighted graph is built at each layer, where the nodes are the data points and two nodes are connected by an edge if they share the same label. In the simplest case, the graph weights, denoted by the $n \times n$ symmetric matrix G_l for the l-th layer, are binary, that is, $G_l(i,j) = 1$ if $X(:,i)$ and $X(:,j)$ share the same label with respect to the features extracted at layer l. A smoothness regularization term is added to the loss function of Eq. (4.3) with the form:

$$\sum_{l=1}^{L} \lambda_l \sum_{\substack{j,k=1 \\ j\neq k}}^{n} \|H_l(:,j) - H_l(:,k)\|_2^2 G_l(j,k) = \sum_{l=1}^{L} \lambda_l \operatorname{Tr}(H_l L_l H_l^T) \tag{4.5}$$

where Tr(.) denotes the trace of a matrix, that is, the sum of its diagonal elements, and $L_l = D_l - G_l$ is the Laplacian matrix at layer l with D_l a diagonal matrix such that $D_l(j,j) = \sum_{k=1}^{n} G_l(j,k)$ for $j = 1,\ldots,n$. Intuitively, Eq. (4.5) enforces the hidden representations $H_l(:,j)$ and $H_l(:,k)$ of data points j and k that share the same label at layer l to be as close as possible.

4.3 Algorithms and parameters of deep MF

In this section, we briefly describe the initialization techniques as well as the algorithms that can be used to solve the subproblems with respect to either W_l or H_l at lines 6 and 7 of Algorithm 9. As these algorithms are standard optimization techniques, we refer the reader to previous surveys [28, 55] for more details and references on their applications in matrix approximations problems. We also discuss the choice of several parameters such as the number of layers L and the inner ranks r_l's.

4.3.1 Initializations

The most commonly used initialization of deep MF consists in applying a sequential decomposition of the data matrix X, as in multilayer MF (see Algorithm 8), but there is no guarantee about the quality of this initialization. For the initialization of the factors W_l and H_l of each layer l, several methods exist; see Section 2.2. The study of initialization techniques dedicated to deep MF is still an open direction of research.

4.3.2 Algorithms

Similarly to standard NMF, most algorithms for deep MF consist in alternatively updating each factor while keeping the others fixed as in Algorithm 9. The stopping criterion can be based either on a maximum number of iterations, a sufficient decrease of the loss function, or a sufficient modification of the factors between two consecutive iterations.

The subproblems with respect to either W_l or H_l for any l are typically solved using standard first-order optimization algorithms. Trigeorgis et al. [115] use a closed-form expression for the W_l's (let us recall that they are not constrained to be nonnegative) and a multiplicative update (MU) for the H_l's. MU is a well-known algorithm to solve NMF [76], and was also proposed to solve a sequential multilayer MF in [1]. Other techniques such as projected gradient descent (PGD) [83], possibly combined with an acceleration scheme, such as Nesterov's one [90] are widely used; see for example [60, 129]. Another standard optimization scheme is the alternating direction method of multipliers (ADMM) which consists in reformulating the problem by decoupling the variables, and minimizing the augmented

Lagrangian. It is standard in the LRMF literature [65], and it has also been used for constrained deep MF in [134].

4.3.3 Parameters

The choice of the parameters in deep MF models, in particular the number of layers and the inner ranks, mainly depends on the application. In most cases, the number of layers used for the results reported in the literature does not even exceed three. Moreover, the ranks tend to be chosen in decreasing order as the first layers of the model are expected to capture features with a larger variance, thus requiring a larger capacity to encode them, while the last layers capture attributes with a lower variance [115]. This observation is also derived from the analogy with autoencoders: the inner layer of an autoencoder is generally the one that contains fewer units as the goal is to obtain a compact representation of the input data; see Section 4.5.1 for more details. On the other hand, when the decomposition is performed on the features matrices such as in Eq. (4.4), the ranks should also be chosen in decreasing order: given $W_1 \in \mathbb{R}^{m \times r_1}$ with r_1 columns in dimension m, it only makes sense to approximate the columns of $W_1 \approx W_2 H_2$ as linear combinations of $r_2 \leq r_1$ columns of W_2 otherwise there exists an infinity of solutions unless rather strong constraints apply on the factors of the decomposition. Besides, how to choose the inner ranks in deep MF is still an open problem. In the following chapters, we stick to inner ranks decreasing along the layers, which seems to be the most natural choice in most applications. Therefore, there is no compromise over the choice of these parameters and their values mainly depend on the application at hand.

In some applications however, such as community detection, state-of-the-art hierarchical clustering algorithms, such as the Louvain method [16], can be applied on top of deep MF to provide an estimation of the number of levels of communities to extract, corresponding to the number of layers L in deep MF, and the corresponding ranks. This also provides an initialization of the factors at the same time. Unfortunately, this approach is not spread yet in the literature, but we used it to set up deep symmetric NMF in a later work, see Section 6.4 for more details.

4.4 Applications of deep MF

We now describe several applications for which deep MF has given interesting results. Let us first mention that since very diverse uses of the terminology "deep MF" have been employed in the literature, it is difficult to circumscribe the scope of deep MF applications. Besides, [6] mentions that some researchers call their model "deep MF" but actually use it for supervised tasks, or introduce a high degree of non-linearity inside it. This is for example the case of [45] that performs deep non-linear matrix completion, and [123] where the inner representations are obtained through a highly non-linear deep MF architecture. Therefore, in this part, we mainly focus on some important works in the same spirit as Trigeorgis [115], aiming to extract hierarchical features in a non-supervised fashion.

In Section 4.4.1, we present showcase examples illustrating the ability of deep MF to extract hierarchical features. Then, Section 4.4.2 lists several applications for which deep MF has been successfully used in the recent literature.

4.4.1 Three showcase examples

In this section, we present three showcase examples that we have designed to show the potentialities of deep MF. In particular, we illustrate the extraction of facial features in Section 4.4.1.1, hyperspectral unmixing (HU) in Section 4.4.1.2 and recommender systems in Section 4.4.1.3.

4.4.1.1 Extraction of facial features

One of the first applications for which deep MF has proven to be useful is the extraction of facial features, as described in the seminal paper of Trigeorgis et al. [115]. Given a set of n grey-scale facial images, each one described by m pixel values, deep MF extracts several layers of features, each one corresponding to a specific interpretation ranging from low-level features at the first layer to high-level features at the last layer.

Fig. 4.2 displays the features extracted at each layer by deep NMF (that is, deep MF with nonnegativity constraints on the factors W_l's, $l = 1, 2, \ldots, L$, and H_L) in the CBCL faces data set, originally used for standard NMF by Lee & Seung [75]. This dataset contains 2429 images of 19×19 pixels, and we apply deep NMF with $L = 3$ layers, $r_1 = 100$, $r_2 = 50$ and $r_3 = 25$. At the

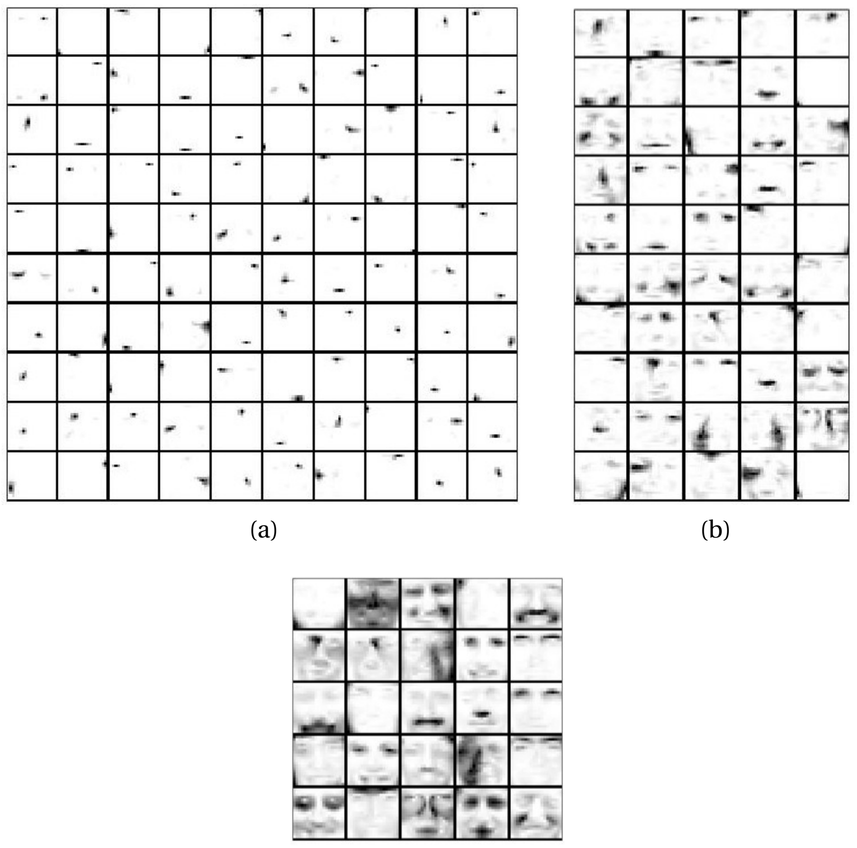

(a) (b) (c)

Figure 4.2: Application of deep NMF to the CBCL faces data set, with $L = 3$, $r_1 = 100$, $r_2 = 50$, and $r_3 = 25$. Each image contains the features extracted at: (a) first layer W_1, (b) second layer $W_1 W_2$, and (c) third layer $W_1 W_2 W_3$.

first layer, 100 low-level features, very specific and localized, are extracted. At the second layer, deep MF extracts 50 meaningful parts of the faces, such as eyebrows, noses, eyes, and mouths. Finally, the last layer exhibits faces made of several facial features, and is more closely related to the identities of the persons. Hence, along the layers, deep MF extracts higher-level features obtained by combining lower-level features from the previous layers.

In summary, deep MF is able to extract hierarchical facial features corresponding to the columns of $\prod_{i=1}^{l} W_i$ at the lth layer.

4.4.1.2 Hyperspectral unmixing

Applied to HU (see Section 1.3), deep ONMF (see Section 4.2.2.1) is able to extract the materials in a hierarchical manner, as illustrated on Fig. 4.3 which shows the abundance maps H_l's, representing the proportions of each material in the pixels. This solution was obtained by applying deep ONMF on the Urban image with $L = 4$, $r_1 = 7$, $r_2 = 6$, $r_3 = 4$, $r_4 = 2$. The first layer extracts seven materials, namely two types of grass, trees, road, dirt, metal and roof. At the next layers, the materials are successively merged by two within a single cluster. More precisely, at layer 2, the two clusters corresponding to road and metal, which have similar spectral signatures, are merged in a single cluster. At layer 3, the road/metal and dirt are merged to create a single cluster while the two kinds of grass are also merged in a single cluster. Finally, at layer 4, the road and roof are merged, while trees and grass are also merged in a cluster made of only vegetation. This example illustrates the ability of deep MF to extract materials in a hierarchical manner in hyperspectral images. Compared to shallow methods, deep MF brings an undeniable value in terms of interpretability.

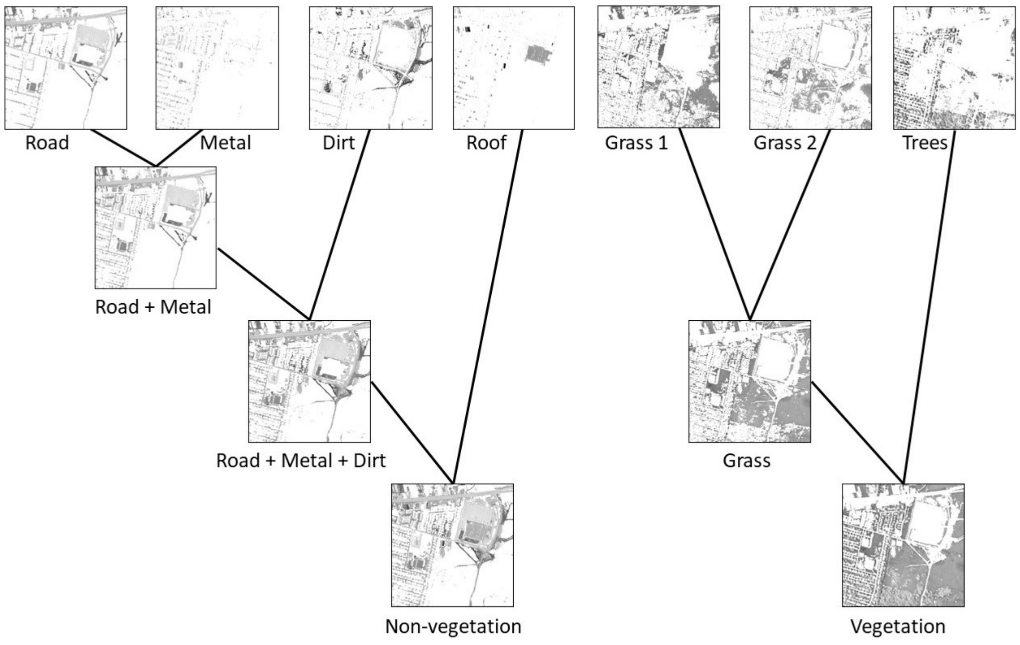


Figure 4.3: Abundance maps hierarchically extracted by deep ONMF on the Urban image. From top to bottom: first, second, third and fourth layer.

In addition, Fig. 4.4 provides a comparison between the extracted spectral signatures at the third ($r_3 = 4$) layer and the ground truth, which are very similar.

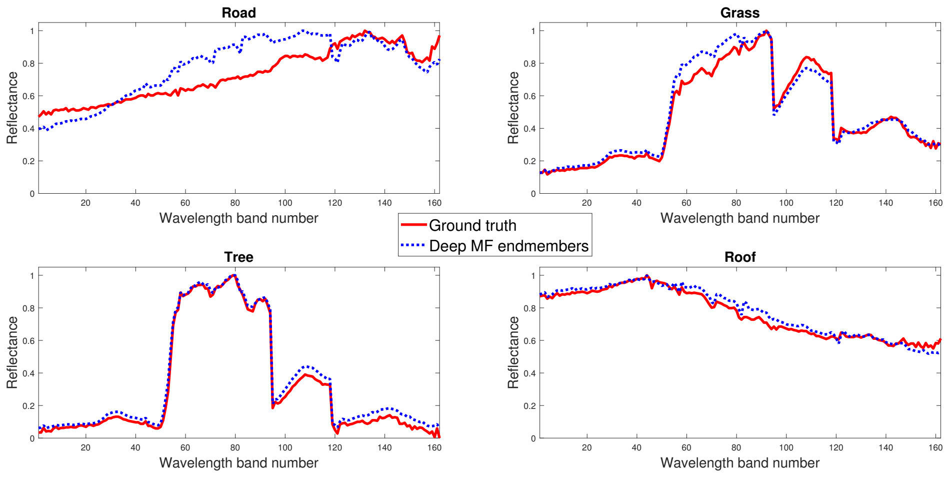


Figure 4.4: Comparison between the endmembers extracted by deep ONMF at the third ($r_3 = 4$) layer and the "ground-truth" endmembers for the Urban hyperspectral image.

4.4.1.3 Recommender systems

Recommender systems consist in predicting the ratings of users over unseen items based on historical ratings on seen items. In other words, given an incomplete rating matrix $X \in \mathbb{R}^{m\times n}$ of n users over m items (such as movies), the goal is to predict the missing entries.

A standard approach to perform this task is to factorize X as the product of two matrices $W \in \mathbb{R}^{m\times r}$ and $H \in \mathbb{R}^{r\times n}$ where W contains the ratings of r basis users over the m items and H represents the proportions in which each user behaves as the r basis users [69]. In the following, we describe how deep MF is able to extract hierarchical levels of basis users in a recommender system on a simple example.

Let us consider the following synthetic matrix $X \in \mathbb{R}^{9\times 15}$ such that $X(i, j)$ stands for the rating of user j over movie i, and is between 1 (highly dislike) and 10 (highly like):

$$X = \begin{pmatrix} 7 & 8 & 10 & 8 & 9 & 4 & 1 & 2 & 3 & 2 & 4 & 1 & 3 & 5 & 2 \\ 8 & 8 & 7 & 8 & 10 & 5 & 1 & 2 & 6 & 1 & 4 & 2 & 2 & 1 & 2 \\ 9 & 9 & 9 & 9 & 10 & 4 & 1 & 4 & 2 & 1 & 4 & 3 & 1 & 1 & 1 \\ 3 & 1 & 2 & 1 & 3 & 8 & 8 & 7 & 8 & 10 & 3 & 4 & 1 & 2 & 2 \\ 2 & 1 & 3 & 2 & 3 & 9 & 10 & 9 & 9 & 8 & 2 & 2 & 2 & 2 & 2 \\ 1 & 2 & 1 & 1 & 2 & 8 & 8 & 9 & 9 & 8 & 3 & 2 & 3 & 3 & 1 \\ 4 & 1 & 1 & 2 & 2 & 2 & 5 & 1 & 1 & 5 & 9 & 7 & 8 & 9 & 7 \\ 2 & 2 & 2 & 1 & 2 & 2 & 6 & 2 & 1 & 3 & 8 & 8 & 8 & 8 & 8 \\ 1 & 2 & 2 & 1 & 3 & 2 & 4 & 1 & 1 & 4 & 6 & 9 & 8 & 10 & 7 \end{pmatrix}.$$

Let us suppose, for example, that the first three movies (first three rows of X) are horror movies, the next three are comedies, and the last three are biopics. We observe that the first five users mostly enjoy horror movies, the next five ones comedies, and the last five biopics. Note that we do not consider missing data, as our goal through this example is to interpret the deep MF decomposition rather than predicting missing entries in itself.

We apply deep ONMF on X with $L = 2$, $r_1 = 4$, $r_2 = 3$, that is, by computing the following decomposition:

$$\begin{aligned} X &\approx W_1 H_1, \quad H_1 H_1^T = I_4, \quad (W_1, H_1) \geq 0, \\ W_1 &\approx W_2 H_2, \quad H_2 H_2^T = I_3, \quad (W_2, H_2) \geq 0. \end{aligned}$$

To render the interpretation of the factors easier, we relax the constraints by only imposing that $H_l H_l^T$ is diagonal, which allows each row of H_l's to have a norm different from 1. In counterpart, we normalize W_l's such that all the elements are between 1 and 10. This allows an easier comparison between the features extracted at each layer.

Let us interpret such a decomposition, layer by layer. At the first layer, we have $X \approx W_1 H_1$, where the matrices W_1 and H_1 are as follows (the values are rounded to one digit of accuracy):

$$W_1 = \begin{pmatrix} 9.1 & 1.7 & 3.4 & 3.8 \\ 8.9 & 1.1 & 4.9 & 2.7 \\ 10.0 & 1.1 & 3.7 & 2.5 \\ 2.2 & 10.0 & 8.6 & 3.0 \\ 2.4 & 10.0 & 10.0 & 2.5 \\ 1.5 & 8.9 & 9.6 & 3.0 \\ 2.2 & 5.5 & 1.5 & 10.0 \\ 2.0 & 5.0 & 1.8 & 9.9 \\ 2.0 & 4.4 & 1.5 & 10.0 \end{pmatrix}, \quad H_1 = \begin{pmatrix} 0.88 & 0 & 0 & 0 \\ 0.87 & 0 & 0 & 0 \\ 0.92 & 0 & 0 & 0 \\ 0.87 & 0 & 0 & 0 \\ 1.05 & 0 & 0 & 0 \\ 0 & 0 & 0.92 & 0 \\ 0 & 0.92 & 0 & 0 \\ 0 & 0 & 0.85 & 0 \\ 0 & 0 & 0.92 & 0 \\ 0 & 0.88 & 0 & 0 \\ 0 & 0 & 0 & 0.82 \\ 0 & 0 & 0 & 0.80 \\ 0 & 0 & 0 & 0.79 \\ 0 & 0 & 0 & 0.89 \\ 0 & 0 & 0 & 0.71 \end{pmatrix}^T .$$

The columns of W_1 are themselves combinations of those of W_2, as $W_1 \approx W_2 H_2$ with

$$H_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1.02 & 0.98 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

At the second layer, we have $X \approx W_2 \hat{H}_2$, with $\hat{H}_2 = H_2 H_1$. As $r_2 = 3$, we expect that each column of W_2 corresponds to the profile of a basis user liking only one category of movies, which is indeed the case as we obtain

$$W_2 = \begin{pmatrix} 9.1 & 2.7 & 3.8 \\ 8.9 & 3.3 & 2.7 \\ 10.0 & 2.6 & 2.5 \\ 2.2 & 9.1 & 3.0 \\ 2.4 & 10.0 & 2.5 \\ 1.5 & 9.3 & 3.0 \\ 2.2 & 3.2 & 10.0 \\ 2.0 & 3.1 & 9.9 \\ 2.0 & 2.7 & 10.0 \end{pmatrix}, \quad \hat{H}_2^T = \begin{pmatrix} 0.88 & 0 & 0 \\ 0.87 & 0 & 0 \\ 0.92 & 0 & 0 \\ 0.87 & 0 & 0 \\ 1.05 & 0 & 0 \\ 0 & 0.91 & 0 \\ 0 & 0.94 & 0 \\ 0 & 0.84 & 0 \\ 0 & 0.91 & 0 \\ 0 & 0.90 & 0 \\ 0 & 0 & 0.82 \\ 0 & 0 & 0.80 \\ 0 & 0 & 0.79 \\ 0 & 0 & 0.89 \\ 0 & 0 & 0.71 \end{pmatrix}.$$

The first column of W_2 corresponds to a basis user liking horror movies, the second comedies, and the third biopics. The first and fourth columns of W_1 are identical to the first and third columns of W_2 respectively while the second and third column of W_1 bring more refined information. Indeed, while the second column of W_2 only exhibits strong ratings for items 4 to 6 and low ones for the other items, the second and third columns of W_1 correspond to two different patterns. They both have high ratings over items 4 to 6, as for the second column of W_2, but the other ratings are different: the second column of W_1 has intermediate ratings for biopics (items 7 to 9) but very poor ones for horror movies (items 1 to 3), and conversely for the third column. This level of granularity is not caught by the second layer, which grasps the three main patterns that appear at first sight when looking at X. This justifies the benefit of using two layers of factorizations. To be more precise, deep MF first extracts 4 refined basis users and then gathers two of them at the second layer to model the more global structure of the data.

Let us mention that single layer ONMF applied separately with both values of the ranks recovers similar factors but, of course, does not exhibit any hierarchical relation between basis users.

4.4.2 Other applications in the literature

In this section, we briefly review the main applications of deep MF across the recent literature.

4.4.2.1 Recommender systems

We have already shown that deep MF extracts hierarchical information within recommender systems in Section 4.4.1.3. Several models based on deep MF have been proposed in the recent literature.

Mongia et al. [87] use a PGD to tackle deep MF with missing entries in the data matrix X. They refer to their model as deep latent factor model, and use it to infer missing entries while extracting several layers of explanatory factors, with a similar interpretation as in our showcase example.

Deep MF was also used in the context of recommender systems with implicit feedback, when the ratings are not given as a numerical value but as a binary feedback (such as "like" or "dislike") [128]. For each user and each item, a vector containing both a representation of this implicit feedback and side information is provided. Then, deep MF is applied separately on the users and items to derive meaningful representations $H_L^{(u_i)}$'s and $H_L^{(v_j)}$'s for all users u_i's and items v_j's respectively, where the inner dimension r_L is the same for both factorizations. The rating r_{ij} of user i over item j is predicted as $r_{ij} = H_L^{(u_i)^T} H_L^{(v_j)} + G_{u_i} + G_{v_j}$ where G_{u_i} and G_{v_j} describe the specific influence of user and item, respectively. Using deep MF decreases the error between the predicted and actual ratings compared to standard MF models on several benchmarks.

4.4.2.2 Multi-view clustering

Another application explored by deep MF researchers is multi-view clustering. It consists in clustering items for which the information is given through several views; for example, images described both by their pixels and textual tags, see [126] for a survey.

In [132] and [31], the data matrix $X^{(v)}$ of each of the V views is deeply factorized and the last hidden representations $H_L^{(v)}$'s are constrained to be the same for all views. This leads to the following objective function to minimize:

$$\mathscr{L} = \sum_{v=1}^{V} \|X^{(v)} - W_1^{(v)} \cdots W_L^{(v)} H_L\|_F^2$$

where $X^{(v)}$ is the data matrix associated to the v-th view, $W_l^{(v)}$ is the features matrix at layer l for the v-th view and H_L is the shared inner representation matrix at the last layer, common to all the views.

Cui et al. [31] incorporate weights, which are also learned, in the loss function mentioned above. A semi-supervised variant is considered by Xu et al. [124] with a graph Laplacian penalty aiming to both minimize the gap between the inner representation H_L of instances sharing the same label and maximize the gap between the inner representation H_L of instances belonging to different classes.

4.4.2.3 Community detection

Community detection consists in identifying communities, that is, subsets of nodes that are highly connected, inside a graph. While NMF is able to extract overlapping communities [125], deep MF allows to interpret the dynamics along which the nodes are progressively grouped. More precisely, taking as input of deep MF an adjacency matrix, that is, a symmetric binary matrix A such that $A(i, j) = 1$ if the nodes i and j are connected and 0 otherwise, leads to the extraction of the membership coefficients of all nodes to r_l communities $H_l \in \mathbb{R}^{r_l \times n}$ at layer l with $r_L \leq r_{L-1} \leq \cdots \leq r_0 = n$ [127]. Deep MF allows to extract communities at different scales; smaller communities at the first layers are merged together in larger communities at the deeper layers. We refer the reader to Section 6.4 for more details on the application of deep NMF to community detection.

4.4.2.4 Hyperspectral unmixing

As illustrated in Section 4.4.1.2, deep MF can be used meaningfully for HU, extracting several layers of materials.

The early sequential multilayer NMF of Cichocki et al. [26] was used, together with sparsity regularization, by Rajabi and Ghassemian [102]. Though the endmembers are estimated accurately, no additional insight is given on the interpretability power of the model. Later, Tong et al. [113] used the deep model of Trigeorgis et al. [115] and show that it is efficient for the extraction of endmembers, though the interpretation of the successive inner representations is not emphasized neither. A similar approach takes into account an additional regularization in [46]: on the one hand a sparsity constraint is considered on the abundance matrix H_L while on the other hand, a spatial regularization is applied through the total variation minimization (TVM). In a nutshell, TVM [105] is a well-known regularization which consists in computing the differences between the abundances of each pair of adjacent pixels and minimizing their sum to reduce the noise and get a smooth abundance map.

4.4.2.5 Audio processing

The audio source separation problem consists in extracting the frequential spectra of the sources contained in a sound recording as well as their respective activations over time. NMF is efficient to solve this problem, when the matrix X is a time-frequency representation of the input data, for example the spectrogram obtained with a short-time Fourier transform (STFT); see [48] and the references therein.

Sharma et al. [108] used deep MF for speech recognition: given a matrix X of n frames, each one corresponding to a sequence of successive words, described by an m-dimensional vector of cepstral coefficients [34], a deep MF alternating sparse (l odd) and dense (l even) layers factorizes X. The authors empirically notice that this framework leads to more discriminative features, and the features used for the classification of the frames are the inner representations H_l's corresponding to sparse layers.

Thakur et al. [112] used deep AA (see Section 4.2.2.3) to extract sources based on the spectrograms of bioacoustics signals, with the features learnt at the first layers corresponding to archetypes on the convex hull of the data while deeper features being more in the center of the data.

4.5 Theoretical considerations

In this section, we present some theoretical results of deep MF. More precisely, Section 4.5.1 describes the interesting analogy that exists between deep MF and deep neural networks while Section 4.5.2 reviews recent results from the literature.

4.5.1 Analogy with neural networks

Several connections can be made between deep MF and deep learning. However, we restrict ourselves as much as possible to models aiming at extracting features from data in an unsupervised and interpretable way. Works embedding MF ideas in a neural network architecture, such as [106, 130, 67, 96] are interesting but are outside the scope of deep MF in itself.

Deep artificial neural networks [74] have been known for several decades as one of the best classification paradigms. On Fig. 4.5, we have represented a standard neural network made of a succession of P fully-connected layers[4]. Each layer k, $k = 0, \ldots, P-1$, is made of s_k units. Let us consider a data matrix $X \in \mathbb{R}^{m \times n}$ of n points in dimension $m = s_0$ and a binary label matrix $Y \in \mathbb{R}^{c \times n}$ indicating the membership of each data point $X(:,j)$ to each of the $c = s_{P-1}$ classes, that is, $Y(i,j) = 1$ if $X(:,j)$ belongs to the i-th class and 0 otherwise. Given $X(:,j)$ for any j as input, the network produces as output a c-dimensional vector $\hat{Y}(:,j)$. Calling

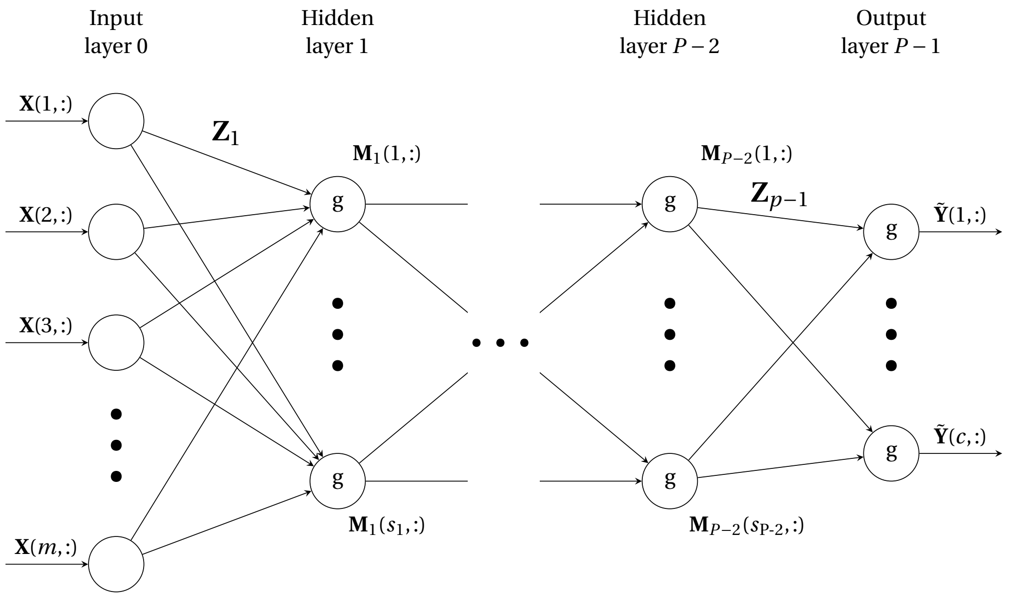


Figure 4.5: Illustration of an artificial neural network.

[4]We use an unusual naming of the parameters to avoid the confusion with the notation introduced for deep MF.

$Z_k \in \mathbb{R}^{s_{k-1} \times s_k}$, $k = 1, \ldots, P-1$, the weights matrix between layer $k-1$ and layer k, the first layer computes a vector $M_1(:,j) = g(Z_1 X(:,j))$ where g is an activation function applied element-wise. Then, any layer k, $k = 2, \ldots, P-1$, computes $M_k(:,j) = g(Z_k M_{k-1}(:,j))$, with $M_{P-1}(:,j) = \hat{Y}(:,j)$. The goal of the neural network is to classify the data points $X(:,j)$'s at best, that is, optimize the Z_k's such that the prediction $\hat{Y}(:,j)$ is as close as possible to the true label vector $Y(:,j)$ for all j. Overall, considering all the data points, the prediction matrix is given by $\hat{Y} = g(Z_{P-1} g(Z_{P-2} \cdots g(Z_1 X)))$.

Autoencoders [92] are particular neural networks where the output matrix does not correspond to a membership matrix (containing class labels) but is identical to the input, that is, $Y = X$. Assuming that the number of layers P is odd, the purpose of an autoencoder is to extract a compressed representation M_Q of the input data at the central layer $Q = \frac{P-1}{2}$ through the encoder, and approximate as well as possible the initial data back after the decoder layers. Fig. 4.6a provides an illustration when the encoder and decoder are symmetric, that is, $s_k = s_{P-1-k}$ for all $k = 0, \ldots, P-1$ and $Z_k = Z_{P-k}^T$ for all $k = 1, \ldots, P-1$. This leads to the approximation

$$X \approx \tilde{Y} = g(Z_1^T g(Z_2^T \cdots g(Z_Q^T M_Q))).$$

Let us number the layers in the reverse sense, that is, let us consider $l = P-1-k$ and $r_l = s_{P-1-l}$ for $l = 0, \ldots, P-1$. Let us also denote $W_l = Z_{P-l} = Z_l^T$ and $H_l = M_{P-1-l}$ for $l = 1, \ldots, Q$, with $L = Q$ such that the decoder performs the following decomposition:

$$X \approx \tilde{Y} = g(W_1 g(W_2 \cdots g(W_L H_L))). \tag{4.6}$$

When the activation function g is the identity, Eq. (4.6) becomes

$$X \approx \tilde{Y} = W_1 \cdots W_L H_L, \tag{4.7}$$

which corresponds to a so-called linear neural network. The decomposition performed by Eq. (4.7) is the same as deep MF but deep MF usually requires additional constraints, such as the nonnegativity of some factors, to render the solution meaningful (see Section 4.1 for more details). In other words, the basis matrices W_l's of deep MF are analogous to the weight matrices of the decoder part of an autoencoder while the abundances matrices H_l's are analogous to the output of the decoder's layers.

A widely used activation function g is the rectified linear unit, that is, $g(x) = ReLu(x) = \max(x, 0)$. With this setting, each inner representation matrix $H_{l-1} = g(W_l H_l)$ for $l = 2, \dots, L$ is imposed to be nonnegative, as in the original deep MF of [115]. Though such a network is very similar to deep MF, as shown on Fig. 4.6b, the two models are not exactly equivalent since the representation matrix H_L of an autoencoder is learnt in a supervised way and is given by $H_L = g(W_L^T g(W_{L-1}^T \dots g(W_1^T X))))$. In fact, autoencoders are mainly used in semi-supervised settings, for example to pre-train networks for classification tasks, while deep MF mines unknown hierarchical features hidden in the data set.

Note that deep MF and deep neural networks have rather different goals. For example, deep neural networks are known to satisfy the universal approximation theorem [33], which states that any continuous function can be approximated, up to some arbitrary precision, with a one-layer network of a finite (but unknown) number of neurons. This comes from the use of non-linear activation functions. Since deep MF's are linear by essence, they do not offer the same representation abilities but are efficient at extracting levels of features in an interpretable way, as already described.

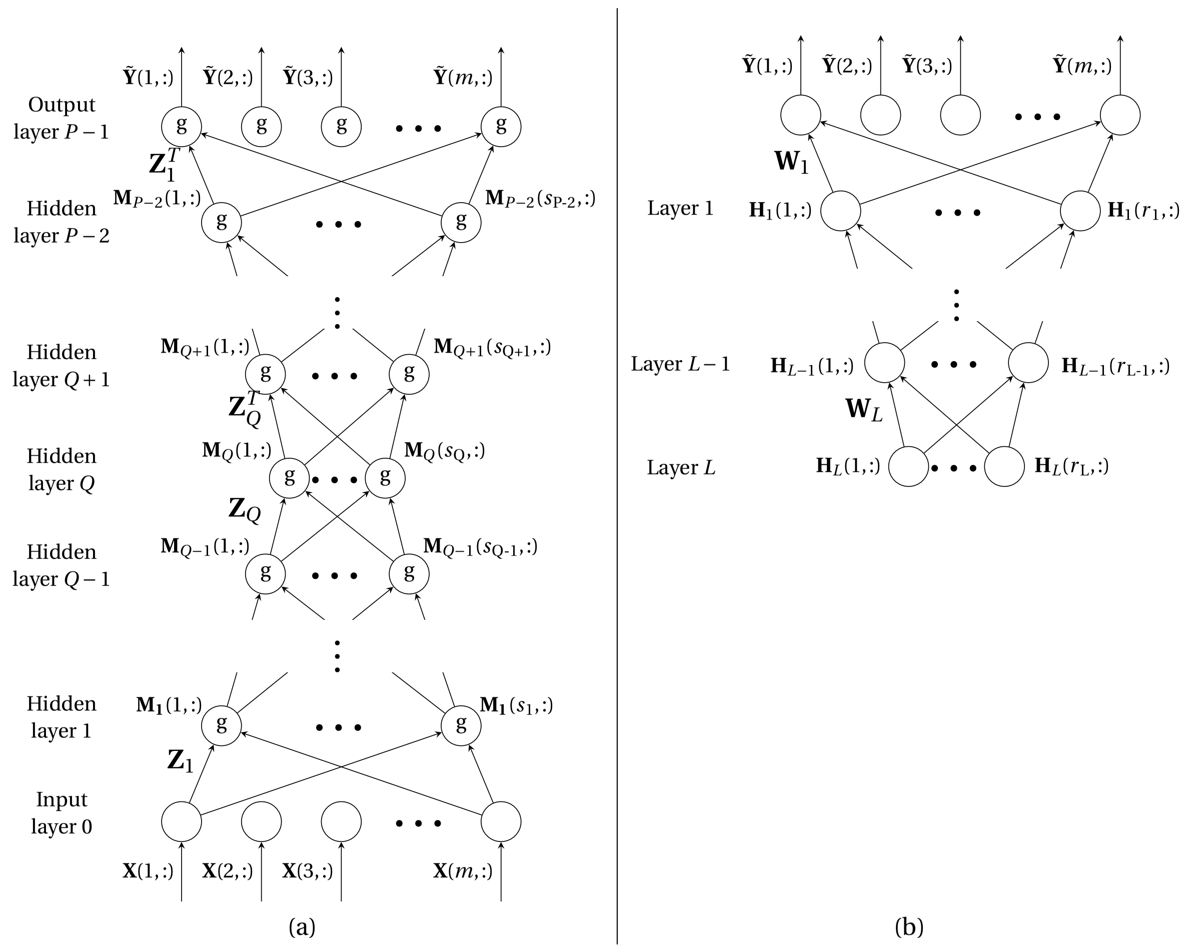
Similarly, deep AA (see Section 4.2.2.3) is closely related to neural networks. An archetypal regularization based on an autoencoder was proposed by van Dijk et al. in [117]. The inner latent representation H is learnt through a deep encoder performing a non-linear transformation of the input data and the addition of Gaussian noise to H enforces the basis vectors to be close to the data at the decoding layer. More precisely, the noise pushes the columns of H outside the unit simplex, which in turn enforces the columns of W to shrink in order to maintain a low reconstruction error.

4.5.2 Various theoretical results

Although numerous models, algorithms and applications have been developed for deep MF, theoretical studies remain scarce, apart from insights from the deep learning community working on linear networks.

4.5.2.1 Convergence issues

The effect of the number of layers on the convergence of first-order optimization methods applied to the minimization of the loss function defined by Eq. (4.3) has not been much studied to the best of our knowledge, but recent results have been obtained on deep linear networks (see Section 4.5.1). Some of the theoretical results presented in the following are



Output layer $P-1$
Hidden layer $P-2$
Hidden layer $Q+1$
Hidden layer Q
Hidden layer $Q-1$
Hidden layer 1
Input layer 0
$\tilde{\mathbf{Y}}(1,:)$ $\tilde{\mathbf{Y}}(2,:)$ $\tilde{\mathbf{Y}}(3,:)$ $\tilde{\mathbf{Y}}(m,:)$
$\mathbf{Z}_1^T$
$\mathbf{M}_{P-2}(1,:)$ $\mathbf{M}_{P-2}(s_{P-2},:)$
$\mathbf{M}_{Q+1}(1,:)$ $\mathbf{M}_{Q+1}(s_{Q+1},:)$
$\mathbf{Z}_Q^T$
$\mathbf{M}_Q(1,:)$ $\mathbf{M}_Q(s_Q,:)$
$\mathbf{Z}_Q$
$\mathbf{M}_{Q-1}(1,:)$ $\mathbf{M}_{Q-1}(s_{Q-1},:)$
$\mathbf{M}_1(1,:)$ $\mathbf{M}_1(s_1,:)$
$\mathbf{Z}_1$
$\mathbf{X}(1,:)$ $\mathbf{X}(2,:)$ $\mathbf{X}(3,:)$ $\mathbf{X}(m,:)$
g
(a)
Layer 1
Layer $L-1$
Layer L
$\mathbf{W}_1$
$\mathbf{H}_1(1,:)$ $\mathbf{H}_1(r_1,:)$
$\mathbf{H}_{L-1}(1,:)$ $\mathbf{H}_{L-1}(r_{L-1},:)$
$\mathbf{W}_L$
$\mathbf{H}_L(1,:)$ $\mathbf{H}_L(r_L,:)$
(b)

Figure 4.6: Illustration of the similarity between (a) deep autoencoders and (b) deep MF.

not directly related to the deep MF models described so far but rather concern supervised learning. However, we strongly believe that these insights coming from the deep linear networks community might be helpful to better understand deep MF and possibly open directions of future research. We refer the interested reader to the recent survey [111] for more details.

When the thinnest layer of a deep linear network is either the input or the output one, Laurent et al. [72] showed that deep linear networks with arbitrary convex differentiable loss function produce local minima that are all global. In addition, when the input data is whitened (that is, the covariance matrix is the identity) and a proper initialization of all layers is chosen, Arora et al. [4] proved the linear convergence of gradient descent to a global minimum on such a network. This generalizes the results of [10] in which linear residual networks, where the weights of each layer are initialized to be the identity matrix and the inner ranks $r_0 = m, \dots, r_L$ are the same, are considered. Indeed, in [4], the network architecture is more general and softer restrictions on the initialization are required.

When the loss function is the squared error between Y and $\hat{Y}$, Arora et al. [5] provide an interesting result: if the weight matrices W_l's are updated with gradient descent and if the initialization ${W_{l+1}^{(0)}}^T W_{l+1}^{(0)} = W_l^{(0)} {W_l^{(0)}}^T$ holds for all l, there exists an equivalent update rule for the end-to-end matrix $W = W_1 \cdots W_L$ which can be seen as an acceleration of the gradient descent update as long as the learning step is sufficiently small. As the depth L grows, the effect is intensified which shows that over-parametrization, that is, considering several hidden layers, might accelerate the optimization process.

In the same spirit, when the network is restricted to be such that each hidden layer contains the same number r of units, but without considering specific assumptions on the input data nor the initialization, Du et al. [42] proved the linear convergence of gradient descent to a global optimum if r is sufficiently large.

4.5.2.2 Low-rank structure

Arora et al. demonstrate in [6] some advantages of unconstrained deep MF compared to standard shallow MF [59] in terms of regularization properties. Indeed, deep MF fosters an implicit tendency towards low-rank solutions. The original problem considered is matrix completion, that is, impute missing entries in a given matrix $X \in \mathbb{R}^{m \times n}$. When the number of known entries is sufficiently large, factorizations of any depth admit solutions that tend to minimize the nuclear norm of the end-to-end matrix $W = W_1 \cdots W_L$, that is, the sum of the singular values of W. However, when there are fewer observed entries, the approximation tends to have a lower effective rank at the expense of a higher nuclear norm, especially when the depth increases. More interestingly, the evolution of the singular values of W obtained with gradient flow, that is, gradient descent with infinitesimally small step size, reveals that the solutions tend to have a few large singular values and many small ones, with a gap that intensifies with the depth of the factorization. This can be seen as an implicit regularization promoting low-rank solutions.

In summary, the recent literature gives evidence of the advantages of using a deep factorization, in terms of both speed of convergence of gradient descent to a global minimum and low-rank-*ness* of the factors. However, the settings described do not assume specific constraints on the factors of the decomposition, unlike most deep MF models do.

4.6 Perspectives of future research

In this section, we list some interesting perspectives of future research related to deep MF, whose some have already been mentioned earlier. Let us also emphasize that some of these directions have been investigated later on during this PhD and are discussed in the next chapters:

- **Choice of the parameters**: The choice of the parameters, namely the inner ranks r_l's and the number of layers L, has not been discussed much as it is mostly application dependent (see Section 4.3.3). This concern is also mentioned as an important open problem by Chen et al. in their survey paper on deep MF [22]. Establishing proper guidelines to choose these parameters is a crucial issue. Let us note that in Section 6.4, we propose a way to set up the number of layers and inner ranks based on the Louvain method [16] that is likely to be more efficient than "handcrafted" choice. More broadly, the initialization of deep MF models is an open problem. In Section 6.1, we propose a greedy initialization of deep MF based on orthogonal factorizations.

- **Identifiability**: Identifiability of deep MF has not been investigated much. This consists in stating the conditions under which a factorization is unique up to trivial permutations and scalings within the factors. Contrary to the flourishing standard NMF literature, only a few works have investigated the identifiability of deep MF, namely [86] where the identifiability conditions are derived for a very particular setting and are hard to check in practice, and [133] for sparse multilayer MF, considering a rather specific definition of a sparse matrix. Establishing more general conditions for deep MF to be unique could be particularly meaningful in some applications. Hence, diving into the theoretical aspects of the uniqueness of deep MF is a promising direction of research.

- **Loss function**: Very few works have carefully investigated the choice of the loss function, see the discussion in Section 4.2.1. Besides, this influences the way the algorithms are designed and how the quality of the extracted features is assessed. In Chapter 5, we discuss more in details this issue and propose two new consistent loss functions for deep MF.

- **Links between deep MF and deep learning**: Though there exist obvious connections between deep neural networks and deep MF as described in Section 4.5.1, they do not seem to have been fully exploited yet. It is not clear whether it is possible to integrate advanced deep learning concepts, such as convolutions or dropout inside deep MF.

- **Applications**: Deep MF has not been applied yet to several important applications such as topic modeling, where it seems obvious that hierarchical structures appear. For example, for NMF, given a word-by-document matrix X where the entry $X(i,j)$ is the number of times that the word i appears in the document j, NMF allows to automatically extract topics as the columns of the basis matrix W, while H indicates which document discusses which topic [75]. In this context, deep NMF would be able to extract hierarchies of topics, from coarser to finer topics. For example, the first layers would extract general topics while the deeper layers would refine these topics in sub-topics. A recent work of Wang & Zhang [122] claims to apply deep NMF for topic modeling but the deepness only results from a non-linear transformation of the input data to obtain a first guess of the matrix of basis vectors, while the factorization in itself remains shallow.

Moreover, in the applications described in Section 4.4.2, the interpretation of the features obtained at each layer is not always clear. This is also an important research issue. In particular, the original data points are usually clustered by applying k-means on the last representation matrix H_L. However, more robust techniques taking into account the information of the previous layers have not been spread yet, to the best of our knowledge.

Let us also emphasize the lack of ground-truth baselines for deep MF: since deep MF is an unsupervised model in its essence, it is hard to guess precisely which features are expected at each layer and how to compare their quality in a fair way. A proper quantitative assessment of deep MF outputs is still to be investigated.

4.7 Take-home messages

In this chapter, we introduced deep matrix factorizations. Let us summarize the main points:

- Deep MF is at the intersection of low-rank matrix approximations and deep learning, and combines the ability to extract hierarchical features and high interpretability. Many models have been developed, mostly as extensions of standard MF models to several layers.
- Deep factorizations are used in an increasing number of applications, from recommender systems and hyperspectral unmixing to multi-view clustering and community detection.
- Deep MF has close links with neural networks, especially autoencoders, but the theoretical insights remain weak.
- Several perspectives of further research have been identified, such as the study of possible alternative loss functions and a better understanding of the contributions of deep MF in various applications.

CHAPTER 5

NEW LOSS FUNCTIONS FOR DEEP MATRIX FACTORIZATIONS

In this chapter, we discuss the choice of the loss function for deep matrix factorization (deep MF). More precisely, we first highlight the drawbacks of the mainstream deep MF loss function and propose two new loss functions in Section 5.1. Then, we introduce a general framework to solve constrained deep MF in Section 5.2. Experiments on synthetic data with this framework are carried out in Section 5.3 with a simple deep MF model. More thorough experiments with both more advanced deep MF models and real-world datasets are presented in Chapter 6. Finally, we summarize the key points in Section 5.4.

5.1 New loss functions for deep MF

In Chapter 4, we presented two milestone models, namely multilayer NMF [26] on the one hand and deep MF [115] on the other hand. In both cases, an input data matrix X is factorized across L layers such that the basis matrices are decomposed[5] as:

$$\begin{aligned} X &\approx W_1 H_1, \\ W_1 &\approx W_2 H_2, \\ &\vdots \\ W_{L-1} &\approx W_L H_L. \end{aligned} \tag{5.1}$$

The multilayer MF model optimizes one layer at a time, that is, minimizes $\|W_{l-1} - W_l H_l\|_F^2$, with $W_0 = X$ successively for each layer $l = 1, \dots, L$. On the other hand, the deep MF model is rather different. The initialization of the factors is performed through multilayer MF but is followed by iterative updates of the factors. For this purpose, a global loss function was proposed by Trigeorgis et al. [115], namely

$$\mathscr{L}_0(H_1, H_2, \cdots, H_L; W_L) = \frac{1}{2}\|X - W_L H_L \cdots H_2 H_1\|_F^2. \tag{5.2}$$

This loss function was reused by most of the papers in the deep MF literature. According to Trigeorgis et al. [115], the update of the factors is performed through Algorithm 10, which is simply Algorithm 9 described in Chapter 4 adapted to a decomposition of the features matrices W_l's.

To understand more clearly how Algorithm 10 works, let us consider the simple case where $L = 3$. In Table 5.1, we report the loss function minimized at each stage of the algorithm, regarding the updates of H_l's and W_l's, with the updated factor in bold, the others being fixed.

Layer l	Update of H_l	Update of W_l
1	$\|X - W_2 H_2 \mathbf{H_1}\|_F^2$	$\|X - \mathbf{W_1} H_1\|_F^2$
2	$\|X - W_3 H_3 \mathbf{H_2} H_1\|_F^2$	$\|X - \mathbf{W_2} H_2 H_1\|_F^2$
3	$\|X - W_3 \mathbf{H_3} H_2 H_1\|_F^2$	$\|X - \mathbf{W_3} H_3 H_2 H_1\|_F^2$

Table 5.1: Loss functions minimized at each step of Algorithm 10 for $L = 3$.

Table 5.1 shows that Algorithm 10 minimizes three different loss functions depending on which factor matrix is updated. More precisely, only

[5]From now, we consider the decompositions of the features matrices W_l's as described in Section 4.2.2.1 rather than the decompositions of H_l's as mentioned in Section 4.2.1. Indeed, this facilitates the interpretation in the applications.

the updates of the last layer ($l = 3$) and the one of H_2 are performed according to the "global" loss function claimed in Eq. (5.2). For $L > 3$, the situation would be even worse, hence Algorithm 10 does not appear to be coherent since different loss functions are minimized along the factors updates. Consequently, convergence guarantees cannot be derived for such an optimization framework. We illustrate this "failure" through numerical examples in the next chapter.

Algorithm 10 Deep MF

```
Input: Data matrix X, number of layers $L$, inner ranks $r_l$'s
Output: Matrices $W_1, \cdots, W_L$ and $H_1, \cdots, H_L$
1: Compute initial matrices $W_l^{(0)}$ and $H_l^{(0)}$ for all $l$
2: for $k = 1, \dots$ do
3:     for $l = 1, \dots, L$ do
4:         $A_l^{(k)} = \begin{cases} W_L^{(k-1)} & \text{if } l = L \\ W_{l+1}^{(k-1)} H_{l+1}^{(k-1)} & \text{otherwise} \end{cases}$
5:         $B_l^{(k)} = H_{l-1}^{(k)} \cdots H_1^{(k)}$
6:         $H_l^{(k)} = \underset{H \geq 0}{\arg\text{reduce}} \frac{1}{2} \| X - A_l^{(k)} H B_l^{(k)} \|_F^2$
7:         $W_l^{(k)} = \underset{W}{\arg\text{reduce}} \frac{1}{2} \| X - W H_l^{(k)} B_l^{(k)} \|_F^2$
8:     end for
9: end for
```

Let us make the following remark. Assuming that the ranks are decreasing, that is, $r_{l+1} \leq r_l$ for all l (this is the most reasonable setting; see Section 4.3.3), any solution $(H_1, \cdots, H_L; W_L)$ can be transformed into a degenerate solution $(H_1^{(*)}, \cdots, H_L^{(*)}; W_L^{(*)})$ with the same value of the loss function in Eq. (5.2), and with the following form

$$H_l^{(*)} = \begin{pmatrix} I_{r_L} & 0_{r_L \times (r_{l-1} - r_L)} \\ 0_{(r_l - r_L) \times r_L} & 0_{(r_l - r_L) \times (r_{l-1} - r_L)} \end{pmatrix} \text{ for } l = 2, \dots, L,$$

where I_{r_L} is the identity matrix of dimension r_L, and $0_{x \times y}$ is the x-by-y zero matrix, $H_1^{(*)} = H_L \cdots H_1$ and $W_L^{(*)} = W_L$. The reason is that $W_L H_L \cdots H_1$ has rank at most $\min_l r_l = r_L$, and with the choice above, we have

$$W_1^{(*)} = W_L^{(*)} H_L^{(*)} \cdots H_2^{(*)} = [W_L \, 0_{m \times (r_1 - r_L)}] \in \mathbb{R}^{m \times r_1}$$

such that $\operatorname{rank}(W_1^{(*)}) \leq r_L$. This means that deep MF would simply reduce to an overparametrized LRMF.

Most of the recent works only rely on the loss given by Eq. (5.2) and follow Algorithm 10. However, for the reasons mentioned above, such a choice is not consistent across layers. In fact, the loss function of Eq. (5.2) only minimizes the error at the last layer of factorization but does not control the accuracy of the factorizations of the previous layers, which yet may be crucial regarding the applications of deep MF. To alleviate this issue, we propose two new loss functions that are consistent, that is, are guaranteed to decrease after each factor update and reflect the balance between the errors due to each layer of decomposition.

Let us note that, as for NMF (see the discussion in Section 1.2), due to the scaling degree of freedom, normalization of one factor per layer is required to avoid that one of the factors tends to 0 while the other tends to infinity and hence "compromises" the remaining layers of decomposition.

5.1.1 Layer-centric loss function

The first loss function proposed consists of a weighted sum of the errors caused by each layer of decomposition, that is, by the layer-wise factorizations:

$$\mathscr{L}_1(H_1, H_2, \cdots, H_L; W_1, W_2, \cdots, W_L) = \frac{1}{2}\Big(\|X - W_1 H_1\|_F^2 + \lambda_1 \|W_1 - W_2 H_2\|_F^2 + \cdots + \lambda_{L-1}\|W_{L-1} - W_L H_L\|_F^2\Big). \quad (5.3)$$

This loss function is quite intuitive, with each term corresponding to a layer-wise error as the factorizations unfold. In fact, this can be seen as a globalization of the model of Cichocki et al. [26]: instead of trying to minimize the errors of each layer-wise factorization sequentially, all of them are aggregated within a global weighted loss function.

As l increases and the ranks decrease, the computational cost to evaluate each term in Eq. (5.3) decreases. More precisely, the l-th term requires $m r_l r_{l-1}$ elementary operations to be computed hence the computational cost of evaluating Eq. (5.3) is $\mathcal{O}(Lmnr_1)$.

5.1.2 Data-centric loss function

The second loss function considers the errors between X and its successive approximations of ranks r_l's:

$$\mathcal{L}_2(H_1, H_2, \cdots, H_L; W_1, W_2, \cdots, W_L) = \frac{1}{2}\Big(\|X - W_1 H_1\|_F^2 + \mu_1 \|X - W_2 H_2 H_1\|_F^2 \\ + \cdots + \mu_{L-1} \|X - W_L H_L \cdots H_2 H_1\|_F^2 \Big). \quad (5.4)$$

This loss function is data-centric in the sense that it evaluates the errors between the data matrix X and its successive low-rank approximations. An advantage of this loss function is that the parameters μ_l's ($l = 1, \ldots, L-1$) are easier to tune since all the terms are likely to have a similar order of magnitude. However, since the successive approximations involve an increasing number of matrix multiplications, this loss function and the associated update rules are slightly more computationally expensive. Indeed, the most costly term is the last one, which requires $m(r_L r_{L-1} + \cdots + r_2 r_1 + r_1 n)$ operations to be computed, which may become high if the number of layers is high and the ranks do not decrease rapidly.

5.2 General framework for constrained deep MF

A general algorithm to minimize the two proposed loss functions of Eq. (5.3) and Eq. (5.4) under general constraints on W_l's and H_l's is given in Algorithm 11. We denote $W_{\to l}$ the set of matrices $\{W_1, \cdots, W_{l-1}\}$ for any $l = 1, \ldots, L$ and $W_{l \to}$ the set of matrices $\{W_{l+1}, \cdots, W_L\}$, and similarly for the H_l's. We also call $\mathcal{W}_l$ and $\mathcal{H}_l$ the feasible set for respectively W_l and H_l for any l. This feasible set encompasses all the constraints applied to the concerned factor. Algorithm 11 consists in a BCD over the factors of each layer. The subproblems in one factor matrix (in bold at lines 4 and 5) can be solved by various well-known techniques. In particular, when the feasible set is convex, the corresponding subproblem is convex. This general framework is also very flexible and we illustrate its use in Chapter 6, considering usual constraints on the factors of each layer.

One possibility to implement the updates of the factors in Algorithm 11, that is, to solve the *arg reduce* subproblems, is FPGD, already described in Algorithm 2. We choose $s_b = \frac{1}{L}$ as the step size, with L the Lipschitz constant, except for the updates of H_l's for the second loss function $\mathcal{L}_2$. Indeed, in this case, the Lipschitz constant is quite costly to compute hence we simply compute the step size through a backtracking line search.

Algorithm 11 Framework to solve deep MF with general constraints and consistent global loss function

```
Input: Data matrix X, number of layers $L$, inner ranks $r_l$'s, feasible sets $\mathscr{W}_l$
    and $\mathscr{H}_l$ for all $l$, a global loss function $\mathscr{L}$ such as $\mathscr{L}_1$ in Eq. (5.3) or $\mathscr{L}_2$
    in Eq. (5.4).
Output: Matrices $W_1, \cdots, W_L$ and $H_1, \cdots, H_L$.
1: Compute initial matrices $W_l^{(0)}$ and $H_l^{(0)}$ for all $l$ through a sequential
   decomposition of $X$ (such as multilayer MF)
2: for $k = 1, \ldots$ do
3:   for $l = 1, \ldots, L$ do
4:     $H_l^{(k)} = \underset{H \in \mathscr{H}_l}{\arg\text{reduce}}\, \mathscr{L}\left(W_{\rightarrow l}^{(k)}, W_l^{(k-1)}, W_{l\rightarrow}^{(k-1)}; H_{\rightarrow l}^{(k)}, \boldsymbol{H}, H_{l\rightarrow}^{(k-1)}\right)$
5:     $W_l^{(k)} = \underset{W \in \mathscr{W}_l}{\arg\text{reduce}}\, \mathscr{L}\left(W_{\rightarrow l}^{(k)}, \boldsymbol{W}, W_{l\rightarrow}^{(k-1)}; H_{\rightarrow l}^{(k)}, H_l^{(k)}, H_{l\rightarrow}^{(k-1)}\right)$
6:   end for
7: end for
```

Let us compute the gradients with respect to W_l's and H_l's ($l = 1, \ldots, L$) of the loss functions of Eq. (5.3) and Eq. (5.4), by introducing $W_0 = X$, $\lambda_0 = \mu_0 = 1$. For $\mathscr{L}_1$, we have:

$$\frac{\partial \mathscr{L}_1}{\partial W_l} = \lambda_{l-1}(W_l H_l - W_{l-1}) H_l^T + \delta_l \lambda_l (W_l - W_{l+1} H_{l+1}) \tag{5.5}$$

where $\delta_l = 0$ if $l = L$ and 1 otherwise,

$$\frac{\partial \mathscr{L}_1}{\partial H_l} = \lambda_{l-1} W_l^T (W_l H_l - W_{l-1}). \tag{5.6}$$

For $\mathscr{L}_2$, let us call for all l, $D_l = H_{l-1} \cdots H_1$, $\tilde{H}_l = H_l \cdots H_1 = H_l D_l$ and $C_l^{(k)} = W_k H_k \cdots H_{l+1}$ for all $k \geq l$ (for $k = l$, $C_l^{(l)} = W_l$). Then, the gradients are given by:

$$\frac{\partial \mathscr{L}_2}{\partial W_l} = \mu_{l-1}(W_l \tilde{H}_l - X) \tilde{H}_l^T, \tag{5.7}$$

$$\frac{\partial \mathscr{L}_2}{\partial H_l} = \sum_{k=l}^{L} \mu_{k-1} C_l^{(k)^T} (C_l^{(k)} H_l D_l - X) D_l^T. \tag{5.8}$$

5.3 Experiments on synthetic data

In this section, we apply our two new loss functions described in Section 5.1 on synthetic data, together with the optimization framework presented in Section 5.2, and compare them to the existing algorithms. More precisely, we compare the following five methods:

- Single-layer NMF of ranks r_l, $l = 2, \ldots, L$.
- The sequential multilayer MF of [26], see Algorithm 8, dubbed MMF. Note that at the first layer, the solution of MMF corresponds to the one of single-layer NMF, by construction.
- The deep MF of [115], see Algorithm 10, dubbed Tri-DMF. Although the factor updates of the original paper are performed with multiplicative updates (MU), we decided to solve the subproblems with FPGD, as our framework also solves the subproblems with FPGD.
- Deep MF with the layer-centric loss function of Eq. (5.3), dubbed LC-DMF.
- Deep MF with the data-centric loss function of Eq. (5.4), dubbed DC-DMF.

Hence, for all these methods, the factor updates are performed through a BCD whose subproblems are solved with FPGD.

Due to the difficulty to handle deep MF settings even for synthetic data, we consider a 2-layers network (that is, $L = 2$) in dimension $m = 3$. The ranks are set to $r_1 = 6$, $r_2 = 3$, and the "ground-truth" basis matrices W_1^* and W_2^* are fixed to

$$W_1^* = \begin{pmatrix} 0.1 & 0.1 & 0.4 & 0.4 & 0.5 & 0.5 \\ 0.4 & 0.5 & 0.1 & 0.5 & 0.1 & 0.4 \\ 0.5 & 0.4 & 0.5 & 0.1 & 0.4 & 0.1 \end{pmatrix} = W_2^* H_2^*,$$

where $W_2^* = \begin{pmatrix} 1/2 & 0 & 1/2 \\ 0 & 1/2 & 1/2 \\ 1/2 & 1/2 & 0 \end{pmatrix}$ and $H_2^* = \begin{pmatrix} 0.2 & 0 & 0.8 & 0 & 0.8 & 0.2 \\ 0.8 & 0.8 & 0.2 & 0.2 & 0 & 0 \\ 0 & 0.2 & 0 & 0.8 & 0.2 & 0.8 \end{pmatrix}$.

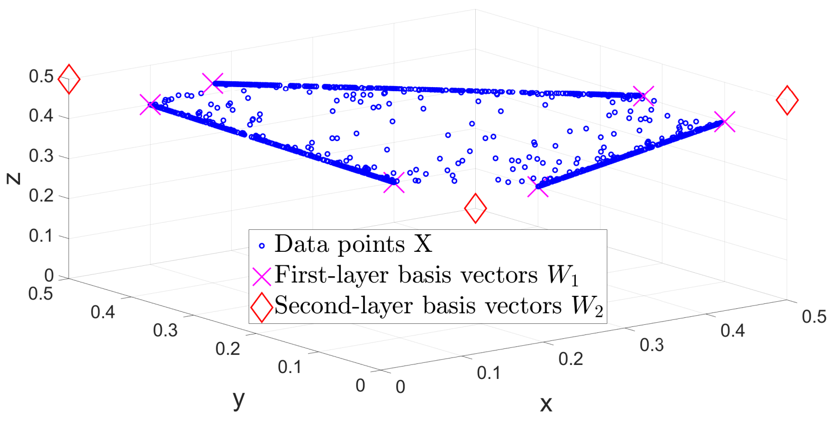


Figure 5.1: Setting of the synthetic data considered for the experiments, in the noiseless case.

Each column of the matrix H_1^* is generated according to a Dirichlet distribution of parameter $\alpha = 0.05$. The data matrix X, made of $n = 1000$ points, is therefore generated as $X = X^* + N$ where $X^* = W_1^* H_1^*$ and N is additive Gaussian noise, see Section 2.4.1. In the following, we consider 10 levels of the relative noise level: $\nu = 10^{-2}$, $2.51\ 10^{-2}$, $6.31\ 10^{-2}$, $9.49\ 10^{-2}$, $1.267\ 10^{-1}$, $1.585\ 10^{-1}$, $2.384\ 10^{-1}$, $3.182\ 10^{-1}$, $3.981\ 10^{-1}$, 1. These noise levels correspond to 6 values logarithmically spaced [6] between 10^{-2} and 1, and additional levels in the areas of fast evolution of the performances. An example of a data set generated in that way in the noiseless case ($\nu = 0$) is presented on Fig. 5.1, with the ground-truth basis vectors at both layers.

For all methods, the initial factors $W_l^{(0)}$ and $H_l^{(0)}$ are obtained by projecting onto the feasible set the output of SNPA [54] (see Algorithm 3 in Section 2.2) applied to $W_{l-1}^{(0)}$, with $W_0^{(0)} = W_0 = X$.

To compute the parameters λ_l's, $l = 1, ..., L-1$ in the first loss function, that is, Eq. (5.3), we proceed by always considering the first layer of decomposition as a baseline. More precisely, based on an initial guess $\tilde{\lambda}_l$, we set $\lambda_l = \tilde{\lambda}_l \frac{err_1^{(0)}}{err_{l+1}^{(0)}}$, where $err_k^{(0)}$ denotes the k-th layer error $\| W_{k-1}^{(0)} - W_k^{(0)} H_k^{(0)} \|_F^2$ after the initialization. By doing so, the ratio between the $(l+1)$-th and the first term of Eq. (5.3) is approximately equal to an arbitrary value $\tilde{\lambda}_l$, fixed by the user. In practice, we used $\tilde{\lambda}_l = 10$ for all $l = 1, \dots, L-1$. For the pa-

[6] In Matlab, this is done through *logspace(-2,0,6)*.

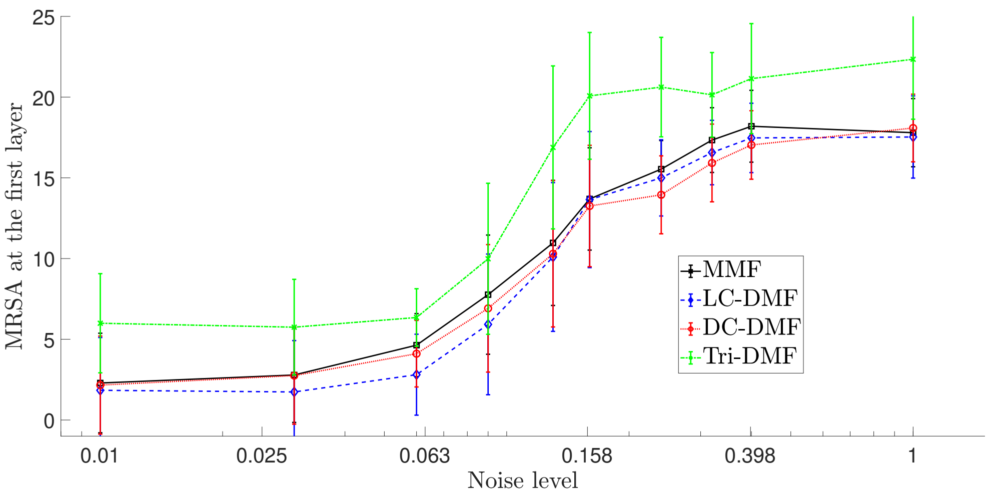


Figure 5.2: Comparison of the MRSA obtained at the first layer with non-negative MMF, LC-DMF, DC-DMF, and Tri-DMF on synthetic data in function of the noise level.

rameters μ_l's of the data-centric loss function, that is, Eq. (5.4), we fix $\mu_l = 1$ for all l since all the reconstruction error terms are expected to be of the same order of magnitude.

Since the global loss functions of the compared methods are not meaningfully comparable to each other due to, on the one hand, the absence of global loss function for multilayer MF and, on the other hand, the difference of magnitude between the terms of layer-centric and data-centric loss functions, we report the average MRSA (see Section 2.4.2) between the corresponding expected and computed basis vectors, that is, the columns of W_l^* and W_l respectively, at the first and second layer, on Fig. 5.2 and Fig. 5.3 respectively.

We also report the number of times each method performs the best over the 25 experiments for each noise level (ranging from 1, lowest to 10, highest), at the first and second layer in Table 5.2 and 5.3 respectively. Note that at the second layer, we only focus on the "deep" methods (single-layer NMF is discarded).

For both layers, the approaches using our new loss functions, that is, LC-DMF and DC-DMF, tend to produce the lowest MRSA. On the contrary, especially at the second layer, MMF produces a higher MRSA than DC-DMF and LC-DMF. This confirms that a weighted loss function is more efficient

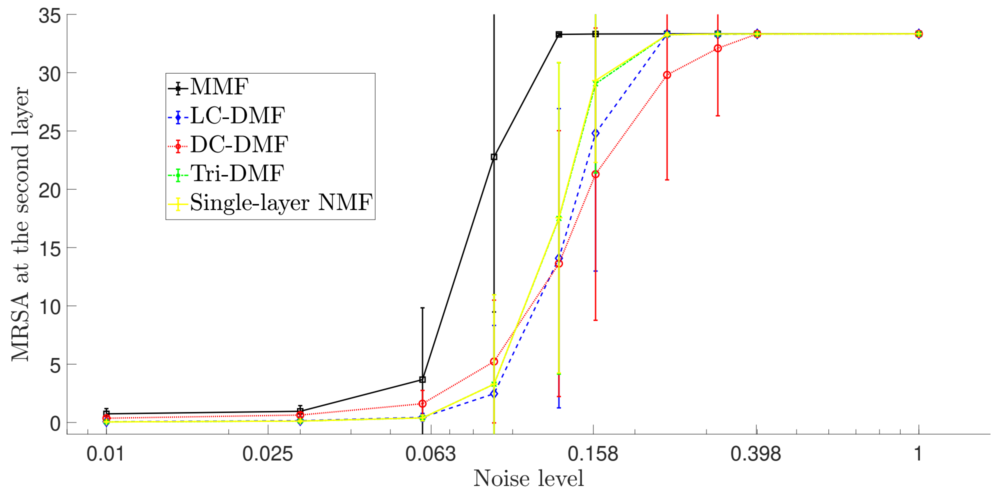


Figure 5.3: Comparison of the MRSA obtained at the second layer with non-negative MMF, LC-DMF, DC-DMF, Tri-DMF and MF on synthetic data in function of the noise level.

than the purely sequential approach of Cichocki et al. On the other side, the approach of Trigeorgis et al. completely fails to retrieve the correct basis vectors at the first layer. In practice, we empirically observe that there are always one or two predicted basis vectors located inside the convex hull of the others. At the second layer, Tri-DMF is not always the best method although the corresponding loss function is designed to minimize the reconstruction error at this last layer. Finally, single-layer NMF also performs slightly worse than our models at the second layer and, of course, does not

	Noise levels									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
MMF	0	0	0	2	6	8	4	3	2	7
Tri-DMF	0	0	0	0	0	0	1	0	0	1
LC-DMF	20	25	23	18	12	11	5	8	8	12
DC-DMF	5	0	2	5	7	6	15	14	15	5

Table 5.2: Number of "winning" runs (over 25) of each method in function of the noise level at the first layer.

	Noise levels									
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
MMF	0	0	1	0	0	0	0	0	1	0
Tri-DMF	17	13	11	12	10	5	8	14	17	7
LC-DMF	8	12	10	10	7	8	2	1	4	8
DC-DMF	0	0	3	3	8	12	15	10	3	10

Table 5.3: Number of "winning" runs (over 25) of each method in function of the noise level at the second layer.

allow to automatically bind the features of consecutive layers, which is undoubtedly the main added value of deep approaches.

When the noise level is low, LC-DMF performs better while when the noise increases, DC-DMF seems more efficient.

5.4 Take-home messages

In this chapter, we introduced two new loss functions for deep MF. Let us summarize the main points:

- The mainstream loss function for deep MF suffers from several drawbacks: it does not allow us to obtain convergence guarantees and does not reflect well the layers contributions.
- We introduce a data-centric loss function and a layer-centric loss function which involve the contributions of each layer.
- These loss functions can be leveraged in a generic optimization framework to solve constrained deep MF.
- Preliminary results on synthetic data with basic constraints on the factors tend to show that the proposed loss functions are well suited for deep MF.

CHAPTER 6

NEW MODELS FOR DEEP MATRIX FACTORIZATIONS

In this chapter, we present various deep MF models that we developed along this PhD. More precisely, Section 6.1 aims at describing a greedy initialization technique for deep MF based on successive orthogonal factorizations [35]. In Sections 6.2 and 6.3, we apply the framework described in Chapter 5 to sparse and minVol deep MF respectively, with intensive experiments on both synthetic data and real-world applications [36]. In Section 6.4, we describe deep symmetric NMF and apply it to community detection [37]. We discuss perspectives of future research in Section 6.5. Finally, Section 6.6 summarizes the main results of this chapter.

6.1 Successive orthogonal decomposition algorithm

The initialization of deep MF factors is a problem that has not raised much attention in the literature so far. Usually, the factors $W_l^{(0)}$'s and $H_l^{(0)}$'s are initialized layer by layer. In this section, we propose a new initialization for deep MF, based on deep ONMF (see Section 4.2.2.1). First, we briefly recall the model of deep ONMF and explain why it can be seen as a hierarchical clustering technique in Section 6.1.1. Then, we present the successive orthogonal decomposition algorithm (SODA) in Section 6.1.2. Results on both synthetic data and HU are presented in Section 6.1.3.

6.1.1 Deep orthogonal NMF as a hierarchical clustering technique

The extension of orthogonal NMF (ONMF) to several layers was already presented in Section 4.2.2.1 and consists in the following decomposition:

$$\begin{aligned}
X &\approx W_1 H_1 \qquad && H_1 \geq 0, \quad H_1 H_1^T = I_{r_1}, \\
W_1 &\approx W_2 H_2 \qquad && H_2 \geq 0, \quad H_2 H_2^T = I_{r_2}, \\
& && \vdots \\
W_{L-1} &\approx W_L H_L \qquad && H_L \geq 0, \quad H_L H_L^T = I_{r_L}.
\end{aligned} \tag{6.1}$$

Each representation matrix H_l is both nonnegative and row-wise orthogonal. Since two nonnegative and orthogonal vectors have disjoint supports, this implies that each column of each H_l has at most a single non-zero entry. Hence, at each layer, deep ONMF associates each data point to a single basis vector and performs a hard clustering of the data points.

In deep ONMF, the ranks r_l's decrease as the factorization unfolds, that is, $r_l > r_{l+1}$ for all l, otherwise the factorizations are trivial. Under this assumption, deep ONMF can be interpreted as a hierarchical clustering (HC) model, as it successively merges the data points by aggregating clusters, which is referred to as bottom-up HC or agglomerative clustering [17]. More precisely, the first layer splits the columns of X into r_1 clusters (the columns of W_1), then the second layer splits the centroids W_1 into r_2 clusters (the columns of W_2), and so on. In particular, when the ranks are such that $r_l = r_{l-1} - 1$ for all l, each layer merges two clusters of the previous layer into a single new cluster while keeping the others unchanged.

Note that other HC techniques are based on NMF ideas. This is for example the case in [71, 58] where rank-2 NMFs are applied sequentially to split clusters in two. In contrast to deep ONMF, these techniques are top-down which means that they start by assigning all the data points to a single cluster and progressively split them in several clusters, hence leading to a different interpretation of the semantics of the data than deep ONMF.

6.1.2 Description of the SODA algorithm

In this section, we first show that there exists a closed-form solution of ONMF when $r = n-1$ in Section 6.1.2.1. Then, we leverage this observation to design a greedy initialization of deep MF in Section 6.1.2.2.

6.1.2.1 Closed-form solution of ONMF with $r = n-1$

SODA is a greedy initialization strategy for deep MF that leverages the fact that there exists an optimal closed-form solution to the ONMF problem (see Eq. (2.3) in Section 2.3.1) when $r = n-1$. Let us consider such a factorization applied to a data matrix $X \in \mathbb{R}^{m\times n}$. Because the model assigns n data points to $n-1$ clusters, only two data points need to be merged within the same cluster, say $X(:,i)$ and $X(:,j)$. Assuming we know i and j, minimizing the reconstruction error requires to find the scalars h_i and h_j and the new centroid $w \in \mathbb{R}^m$ such that

$$e(i,j) = \frac{1}{2}\left(\|X(:,i) - h_i w\|_2^2 + \|X(:,j) - h_j w\|_2^2\right) \tag{6.2}$$

is minimized, with $h_i^2 + h_j^2 = 1$ and $(h_i, h_j) \geq 0$.

Using the first-order optimality conditions, it is easy to show that, at optimality,

$$w = h_i X(:,i) + h_j X(:,j)$$

and

$$h_k = \frac{X(:,k)^T w}{\|w\|_2^2}$$

for $k = i, j$. Combining these expressions and the fact that $h_i^2 + h_j^2 = 1$, and writing everything in terms of $\alpha = h_i$, we obtain

$$\frac{\alpha}{\sqrt{1-\alpha^2}} = \frac{\alpha\|X(:,i)\|_2^2 + \sqrt{1-\alpha^2}X(:,i)^T X(:,j)}{\alpha X(:,i)^T X(:,j) + \sqrt{1-\alpha^2}\|X(:,j)\|_2^2}.$$

Let us assume without loss of generality that $\|X(:,i)\|_2 \geq \|X(:,j)\|_2$. In the case of equality, $\alpha = \frac{1}{\sqrt{2}}$, otherwise $\alpha = \sqrt{\frac{L+\sqrt{L}}{2L}}$ where $L = 4K^2 + 1$ and $K = \frac{X(:,i)^T X(:,j)}{\|X(:,i)\|_2^2 - \|X(:,j)\|_2^2}$.

Finally, the optimal solution of ONMF with $r = n-1$ will merge the data points $X(:,i)$ and $X(:,j)$ for which $e(i,j)$ takes the smallest value. After permutation of the data points, let us assume without loss of generality that $i = n-1$ and $j = n$. Then, the optimal ONMF has the form

$$X \approx \left(X(:,1:n-2) \;\; w\right)\begin{pmatrix} I_{n-2} & 0 & 0 \\ 0 & h_i & h_j \end{pmatrix}.$$

Algorithm 12 summarizes the ONMF of two data points. Note that, given the indices (i,j) of the points to "merge", computing α and w requires $\mathcal{O}(m)$ operations. Hence ONMF$(n-1)$, that is, orthogonal NMF with rank equal to $n-1$, requires $\mathcal{O}(mn^2)$ operations to test all pairs (i,j) and pick the one that leads to the lowest error.

Algorithm 12 Optimal rank-one ONMF of two points

```
Input: Two data points $X(:,i), X(:,j) \in \mathbb{R}^m$.
Output: Basis vector $w \in \mathbb{R}^m$, coefficient $\alpha$, error $e(i,j)$.
 1: perm= 0
 2: if $\|X(:,i)\| < \|X(:,j)\|$ then
 3:     $(X(:,i), X(:,j)) \leftarrow (X(:,j), X(:,i))$; perm = 1
 4: end if
 5: if $\|X(:,i)\| == \|X(:,j)\|$ then
 6:     $\alpha = \frac{1}{\sqrt{2}}$
 7: else
 8:     $K = \frac{X(:,i)^T X(:,j)}{\|X(:,i)\|^2 - \|X(:,j)\|^2}$; $L = 4K^2 + 1$; $\alpha = \sqrt{\frac{L+\sqrt{L}}{2L}}$
 9: end if
10: $w = \alpha X(:,i) + \sqrt{1-\alpha^2} X(:,j)$
11: $e(i,j) = \frac{1}{2}(\|X(:,i) - \alpha w\|^2 + \|X(:,j) - \sqrt{1-\alpha^2} w\|^2)$
12: if perm == 1 then
13:     $\alpha \leftarrow \sqrt{1-\alpha^2}$
14: end if
```

6.1.2.2 A greedy initialization of deep MF

We now exploit Algorithm 12 to initialize deep MF. We call this initialization SODA.

Starting from the trivial decomposition $X = W_1 H_1$, with $W_1 = X$ and $H_1 = I_n$, that is, the identity matrix of size n, the two data points $W_1(:,i)$ and $W_1(:,j)$ that lead to the smallest value of $e(i,j)$ (see Eq. (6.2)) are merged at layer 2 according to the closed-form solution described in the previous section. Then, similarly, at each layer, two basis vectors are merged in a new one according to Algorithm 12. When the current number of basis vectors is equal to the value of some desired inner rank r_l of the deep MF model, they are used to initialize the corresponding W_l. The working of SODA is summarized by Algorithm 13.

Algorithm 13 Successive orthogonal decomposition algorithm (SODA)

```
Input: Nonnegative matrix $X \in \mathbb{R}^{m\times n}$, number of layers $L$, inner ranks $r_l$'s
    for $l = 1,\dots,L$ .
Output: Initialization of $W_l$'s, $l = 1,...,L$.
 1: $U_1 = X$, $V_1 = I$, $l = 1$
 2: for $i = 1,...,n - r_L + 1$ do
 3:     $best\_err = \infty$
 4:     $d_i = n - i + 1$
 5:     if $d_i = r_l$ for some $l$ then
 6:         $W_l = U_i$
 7:     end if
 8:     for $j = 1,...,d_i$ do
 9:         for $k = j + 1,...,d_i$ do
10:             $[w, \alpha, err]$ = Algorithm 12($U_i(:,j)$, $U_i(:,k)$)
11:             if $err < best\_err$ then
12:                 $best\_err = err$, $best\_w = w$, $best\_alpha = \alpha$, $best\_ind = [j,k]$
13:             end if
14:         end for
15:     end for
16:     $U_{i+1} = U_i$
17:     $U_{i+1}(:, best\_ind) = [\,]$; $U_{i+1} = [U_{i+1}\ best\_w]$ % Replace the two data
        points by the new basis vector
18:     $V_{i+1} = \begin{pmatrix} I_{d_i-2} & 0 & 0 \\ 0 & \alpha & \sqrt{1-\alpha^2} \end{pmatrix}$
19: end for
```

Let us analyse the computational cost of SODA. First, the reconstruction errors $e(i,j)$'s of all pairs of data points i and j are computed, which requires $\frac{n(n-1)}{2}$ times $\mathcal{O}(m)$ operations. SODA needs $n - r_L$ steps to be able to construct W_L; each of them requires to compute the reconstruction error of Eq. (6.2) between the remaining basis vectors of the previous layer and the new one, which takes $\mathcal{O}(nm)$ operations. Moreover, assuming an efficient sorting strategy of the array of errors $e(i,j)$'s, finding the couple of indices which generates the smallest $e(i,j)$ is $\mathcal{O}(n^2 \log(n^2))$. Hence, SODA requires in total $\tilde{\mathcal{O}}(n^2 m)$ operations (where $\tilde{}$ indicates that we removed logarithmic terms), which is not practical for large data sets. However, for applications such as hyperspectral unmixing, this greedy initialization can be used as well, up to a slight adaptation. We first compute an ONMF of rank $r_1' \geq r_1$ with any standard algorithm faster than SODA, and then "unfold" the remaining r_1' clusters through SODA. In practice, we recommend to use r_1' as a small multiple of r_1, see the next section.

6.1.3 Experiments

In this section, we apply SODA initialization on both synthetic data in Section 6.1.3.1 and hyperspectral unmixing (HU) in Section 6.1.3.2.

6.1.3.1 Synthetic data

In this section, we evaluate the performance of several initialization techniques for deep ONMF on synthetic data. We compare the following initializations for the features matrices W_l's of all layers (given that $W_0 = X$):

- Random initialization (RAND): any W_l, $l = 1, \ldots, L$ is set up by randomly picking $r_l < r_{l-1}$ columns of W_{l-1}.
- Successive nonnegative projection algorithm (SNPA): each W_l is obtained by applying SNPA [54] on W_{l-1}.
- Our proposed greedy algorithm, SODA, that is, Algorithm 13.
- RAND+SODA: Following the remark on the computational cost of SODA at the end of Section 6.1.2.2, we randomly choose $r_1' \ll n$ points, and then apply SODA on this subset of r_1' points.

In all cases, after the initialization, deep ONMF is solved through an alternating block coordinate descent, inspired by Algorithm 10[7]. The subproblems with respect to W_l's are easy to solve with FPGD, since the only constraint is nonnegativity. On the other hand, to optimize a factor H_l for a given l, we need to solve

$$\min_{\substack{H_l \geq 0 \\ H_l H_l^T = I_{r_l}}} \frac{1}{2} \|X - W_L H_L \cdots H_l \cdots H_1\|_F^2. \tag{6.3}$$

Let us call $D = W_L H_L \cdots H_{l+1}$ ($D = W_L$ if $l = L$) and $F = H_{l-1} \cdots H_1$. The matrix H_l is updated by extending the multiplicative updates proposed for ONMF by [24], that is

$$H_l \leftarrow H_l \circ \frac{D^T X F^T}{H_l F X^T D H_l}. \tag{6.4}$$

To generate the data, we restrict ourselves to a $m = 3$-dimensional setting, to facilitate the assessment of the models. We take $r_1 = 16$ and $r_2 = 4$ and generate the ground-truth basis matrices W_1^* and W_2^* whose columns have unit ℓ_1 norm in such a way that the 16 basis vectors of the first layer are clustered in 4 groups around the 4 basis vectors of the second layer. As shown on Fig. 6.1, each column of W_2^* is the average of 4 columns of W_1^*.

We then pick $n = 1000$ points such that each data point is equal to one of the columns of W_1^*, up to a scaling factor. Finally, additive white Gaussian noise is added to the data points, the same way as described in Section 2.4.1, with 4 different values of the relative noise level ν: $10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}$.

To assess the quality of the different initializations, 10 data sets are generated for each noise level and we report the mean and standard deviation of the clustering accuracy (ACC) over these 10 runs at both layers. Given K estimated clusters C_k's and K ground-truth clusters C_k^*'s, the ACC is defined as

$$ACC(C, C^*) = \frac{1}{n} \max_{P \in [1 \cdots K]} \sum_{k=1}^{K} |C_k \cap C_{P(k)}^*| \tag{6.5}$$

where P is any permutation of $\{1, 2, \ldots, K\}$. In this case, at a given layer l, the ground-truth clusters C_k^*'s are the columns of W_l^* and the estimated clusters C_k's are the columns of the final matrix W_l. The assignment of a

[7]Note that, at the time of doing this research, we had not worked on the loss functions of Chapter 5 yet hence still used the loss function of [115].

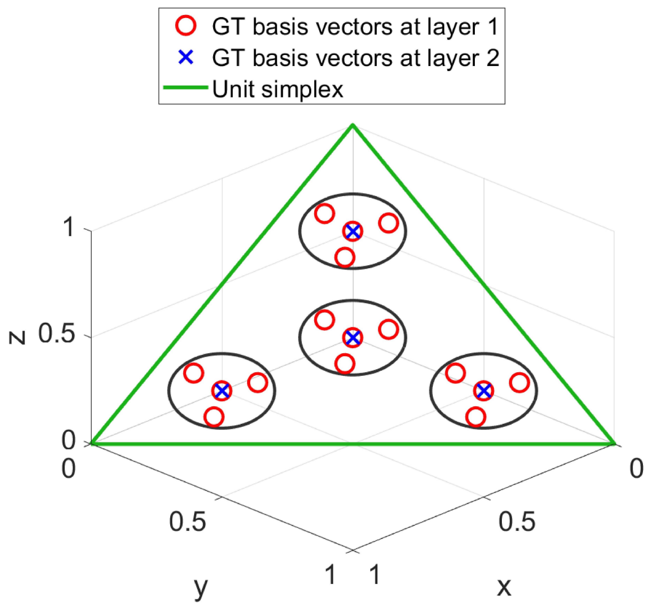


Figure 6.1: Illustration of the synthetic data set used to assess SODA initialization.

data point to a cluster is easy since deep ONMF provides factors H_l's with a single non-zero entry per column. At any layer l, the cluster to which a given data point belongs to is simply the index of the row containing this non-zero entry in the corresponding column of H_l.

Table 6.1 reports for all the configurations the final relative reconstruction error $\frac{\|X-W_2H_2H_1\|_F}{\|X\|_F}$, denoted *rE*, and the accuracy at the first and second layers, denoted *ACC 1* and *ACC 2*, respectively. In all the experiments, deep ONMF is applied with the modalities described above (that is, $L = 2$, $r_1 = 16$, $r_2 = 4$); only the initialization varies. The metrics reported in Table 6.1 are the ones computed with the final factors of the whole deep ONMF. Note that for the hybrid initialization RAND+SODA, $r_1' = 100$.

Clearly, SODA outperforms RAND and SNPA in terms of clustering accuracy. When the noise is small, it always manages to reach a perfect clustering at both layers, contrary to the two other methods. Of course, this is at the expense of a larger computational cost, from $\mathcal{O}(mnr)$ for RAND and SNPA, to $\mathcal{O}(mn^2)$ for SODA. However, the hybrid variant RAND+SODA performs almost as well as SODA at a reduced cost, showing that using the greedy procedure further in the decomposition is also worthwhile.

	RAND			SNPA			SODA			RAND+SODA		
ν	ACC 1	ACC 2	rE (%)	ACC 1	ACC 2	rE (%)	ACC 1	ACC 2	rE (%)	ACC 1	ACC 2	rE (%)
10^{-4}	0.54 ± 0.14	0.74 ± 0.21	9.26 ± 6.23	0.21 ± 0.03	0.69 ± 0.17	14.84 ± 3.91	**1**	**1**	7.49	**1**	**1**	7.50
10^{-3}	0.49 ± 0.17	0.66 ± 0.19	9.41 ± 6.19	0.18 ± 0.02	0.67 ± 0.14	15.49 ± 4.24	**1**	**1**	7.49	**1**	**1**	7.50
10^{-2}	0.48 ± 0.17	0.76 ± 0.16	10.31 ± 6.51	0.42 ± 0.09	0.72 ± 0.16	10.82 ± 4.21	**0.97**	**0.99**	7.51	**0.97**	**0.99**	7.52
10^{-1}	0.40 ± 0.07	0.70 ± 0.17	15.46 ± 4.57	0.38 ± 0.07	0.68 ± 0.09	14.27 ± 3.20	$\mathbf{0.69 \pm 0.01}$	$\mathbf{0.92 \pm 0.01}$	9.75	0.57 ± 0.07	$\mathbf{0.92 \pm 0.01}$	10.00 ± 0.67

Table 6.1: Comparison of the clustering accuracies at layer 1 (*ACC 1*) and 2 (*ACC 2*) and final relative error (*rE*) of deep ONMF applied to synthetic data with several initialization strategies, in function of the noise level ν. The average and standard deviation (if above 0.01) over 10 data sets are reported. The best method in terms of accuracy is highlighted in bold for each configuration.

Note that the accuracy of all algorithms is always a bit higher for the second layer since there are fewer and better separated clusters. The reason SNPA underperforms is because some clusters are contained in the conical hull of the others, while SNPA is designed to identify only the extreme rays of the cone generated by the columns of X.

6.1.3.2 Hyperspectral unmixing

We have already shown in Section 4.4.1.2 that the hierarchical extraction of materials within a hyperspectral image could be performed through deep MF.

The abundance maps extracted by deep ONMF, with $L = 5$, $r_l = 7 - l$ for all $l = 1, \dots, 5$ are represented on Fig. 6.2.

To initialize the factors, we first apply ONMF with $r_1 = 6$ with HC initialization, and then apply SODA to initialize the other layers, contrary to Section 4.4.1.2 where SNPA initialization was applied at *all* layers. However, it turns out that the abundance maps are qualitatively similar in both cases. For conciseness, we gathered the representations of layers 2 and 3 as well as those of layers 4 and 5 in a single level as distinct clusters were merged at these layers.

These results show that a hybrid initialization of deep MF combining a standard method such as HC on top of SODA is efficient to initialize the factors.

6.2 Sparse deep matrix factorizations

In this section, we apply the framework of Chapter 5 to sparse deep MF. Our goal is to show experimentally that the new loss functions introduced in Chapter 5 are efficient to tackle deep MF models with various constraints, such as sparsity in this case. As a reminder, sparse matrix factorization consists in enforcing some factor(s) of the decomposition to be sparse in order to foster the interpretability. As it will be highlighted in the experiments, sparse deep MF allows to extract several levels of sparse features in a dataset, for example, for the extraction of facial features. In this case, only a small number of pixels are activated in each feature of each layer, see Section 6.2.2 for more details.

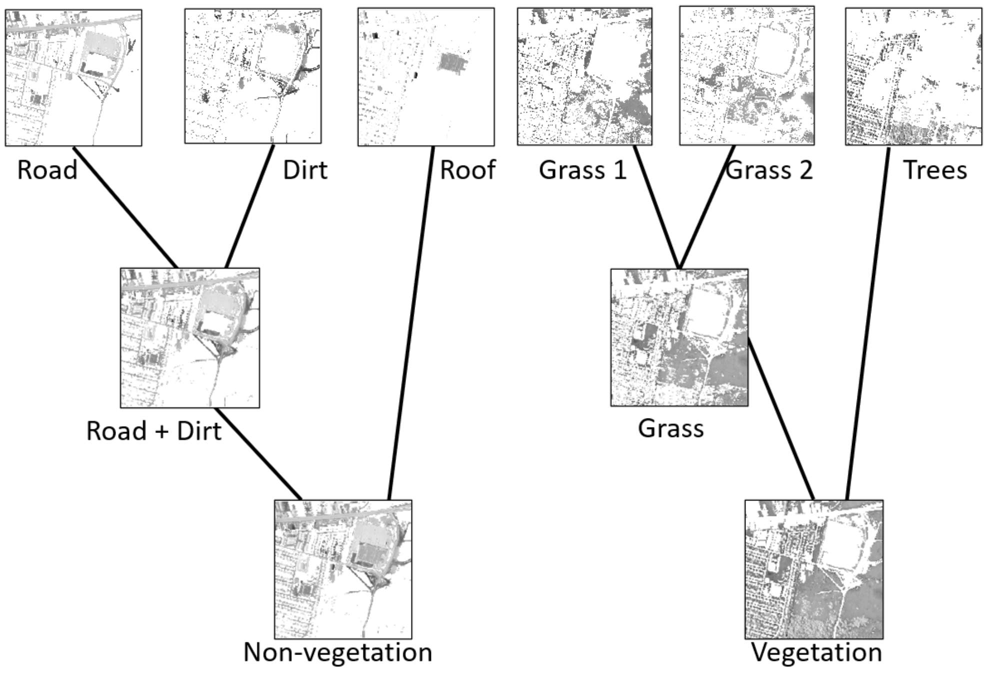


Figure 6.2: Deep ONMF applied to the Urban HI with SODA-based initialization.

Numerous ways of tackling sparsity in MFs have been proposed in the literature, including targeting a row-wise (or column-wise) l_1 norm for some factors [63], adding an l_1 and/or l_2 norm penalty [68] to the loss function, see Section 4.2.2.2 for more references.

Recently, an efficient and fast method, referred to as grouped sparse projection (GSP) [94], was developed. An advantage of this approach is that it allows to avoid the tuning of many parameters, unlike most sparse models. In the context of deep MF, where the number of factors to update grows linearly with the number of layers, it is more than welcome to limit the number of hyperparameters coming from the additional constraints or regularizations.

Given any matrix $A \in \mathbb{R}^{t\times u}$, GSP aims to solve

$$\min_{B\in\mathbb{R}^{t\times u}} \|B-A\|_F^2 \quad \text{such that} \quad \frac{1}{u}\sum_{i=1}^{u} \text{sp}(B(:,i)) \geq s \tag{6.6}$$

where s is the target average sparsity level and

$$\mathrm{sp}(B(:,i)) = \frac{\sqrt{t} - \frac{\|B(:,i)\|_1}{\|B(:,i)\|_2}}{\sqrt{t}-1} \tag{6.7}$$

for any i is known as the Hoyer sparsity of the corresponding column of B.

We refer the reader to Algorithm 1 of [94] for the details of the implementation of the projection onto the feasible set required by Eq. (6.6). Therefore, it suffices to consider this implementation at the projection step in the FPGD (see Algorithm 2 in Chapter 2) used to solve the corresponding subproblem in Algorithm 11.

Hence, for a given factor, GSP aims to reach a target average sparsity of the whole matrix instead of each row/column separately, which confers much more flexibility to the sparsity pattern compared to standard approaches, while reducing drastically the number of parameters involved. Let us remark that since the feasible set of GSP is not convex, the convergence to a stationary point of the underlying framework is not theoretically guaranteed.

We apply sparse deep MF, where the sparsity is defined as described above, to both synthetic data, in Section 6.2.1, and the extraction of facial features in Section 6.2.2. The same five models as described in Section 5.3 are compared in all the experiments, that is, MMF, Tri-DMF, LC-DMF, DC-DMF and single-layer NMF.

6.2.1 Experiments on synthetic data

The experimental setting considered in this section is exactly the same as the one of Section 5.3.

Due to the structure of the ground-truth factors, we only consider sparsity on the factors of the second layer, fixing the target grouped sparsity of W_2 and H_2 to the average Hoyer sparsity of W_2^* and H_2^* respectively.

Fig. 6.3 and 6.4 display the MRSA (mean and standard deviation over 25 runs) of MMF, LC-DMF, DC-DMF, Tri-DMF and single-layer NMF (for the second layer only) in function of the noise level for the first and second layers, respectively. At both layers, LC-DMF produces the lowest MRSA. As already highlighted in Section 5.3, MMF and Tri-DMF show their weaknesses compared to our approaches. These results on synthetic data tend to show that LC-DMF is the most efficient method to recover the ground-truth factors at both layers.

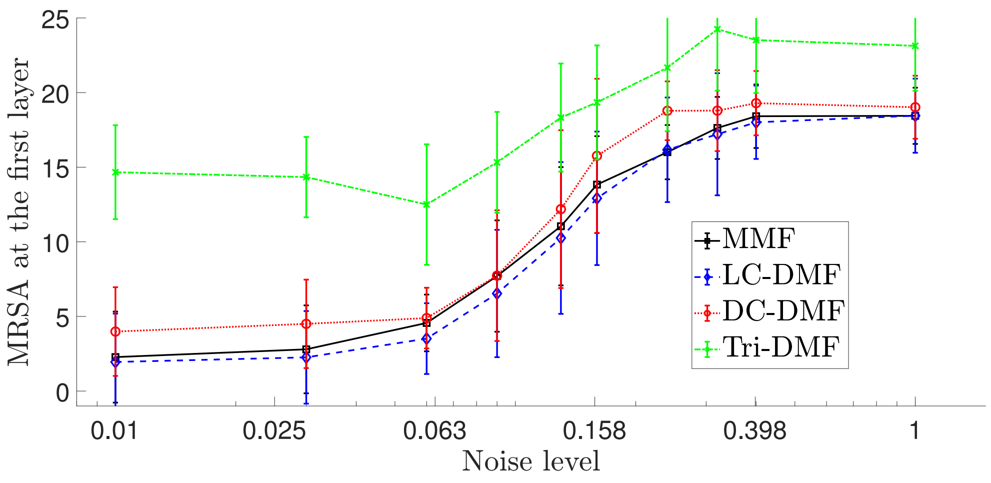


Figure 6.3: Comparison of the MRSA obtained at the first layer with sparse MMF, LC-DMF, DC-DMF and Tri-DMF on synthetic data in function of the noise level.

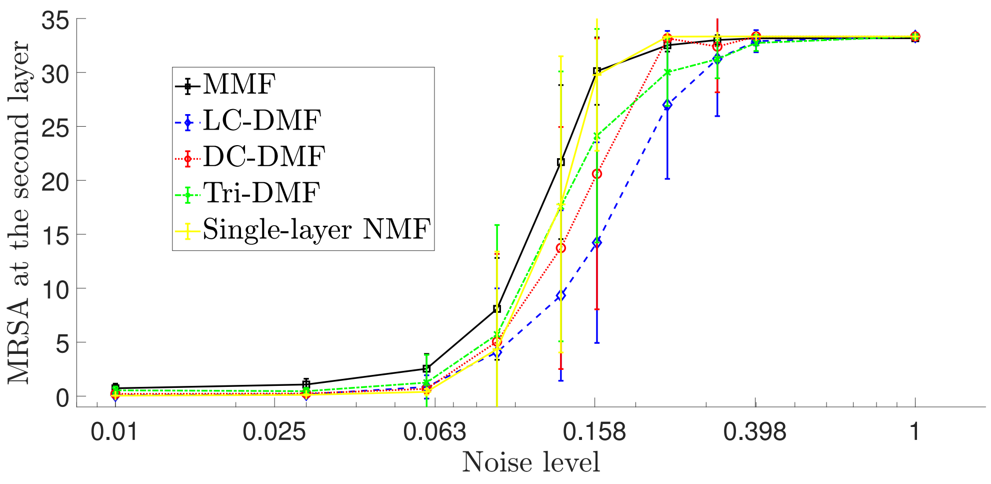


Figure 6.4: Comparison of the MRSA obtained at the second layer with sparse MMF, LC-DMF, DC-DMF, Tri-DMF and NMF on synthetic data in function of the noise level.

6.2.2 Extraction of facial features

We have already shown in Section 4.4.1.1 that deep MF is efficient to extract facial features hierarchically. The CBCL dataset[8] is made of 2429 grey-scale images of 19×19 pixels representing faces of different people exhibiting various expressions.

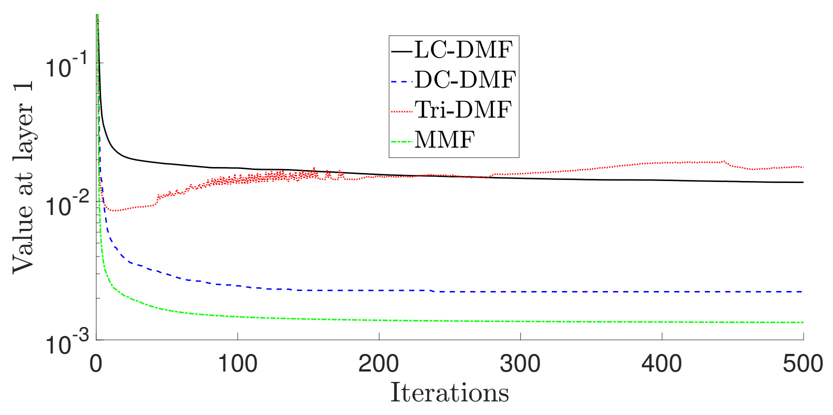
For this application, we set $L = 3$, $r_1 = 100$, $r_2 = 49$ and $r_3 = 25$, and perform the initialisation of the factors with SNPA. We run MMF, LC-DMF, DC-DMF and Tri-DMF with a grouped sparsity level s (see Eq. (6.6)) of 70% for W_1, 80% for W_2 and 85% for W_3 (the factors H_l's are not constrained to be sparse)[9]. The features extracted at all layers by sparse MMF are displayed on Fig. 6.7. It shows the hierarchical decomposition of the data set: the larger facial features extracted at the first levels are made of smaller features extracted at the deeper levels, such as eyes, mouths, and eyebrows[10]. Sparsity allows the features, especially at the last layer, to contain only a few activated pixels. The features extracted by the other methods, that is, LC-DMF, DC-DMF, Tri-DMF and single-layer NMF are displayed on Fig. 6.8, 6.9, 6.10 and 6.11, respectively.

To compare quantitatively these methods, we plot the evolution of several loss functions. More precisely, Fig. 6.5 shows the evolution of the relative layer-centric errors $\frac{\|W_{l-1}-W_l H_l\|_F^2}{\|W_{l-1}^{(0)}\|_F^2}$ for $l = 1,2,3$, with $W_0 = X$, for all methods, along 500 iterations. Similarly, Fig. 6.6 shows the relative data-centric errors $\frac{\|X-W_l H_l \cdots H_1\|_F^2}{\|X\|_F^2}$ for $l = 1,2,3$. Note that the first layer errors are both equal to $\frac{\|X-W_1 H_1\|_F^2}{\|X\|_F^2}$ and appear in all global loss functions except Tri-DMF. Also, looking at the evolution of data-centric errors does not make much sense for MMF, due to the way the decompositions are performed (the data matrix is "discarded" after the optimization of the factors of the first layer).

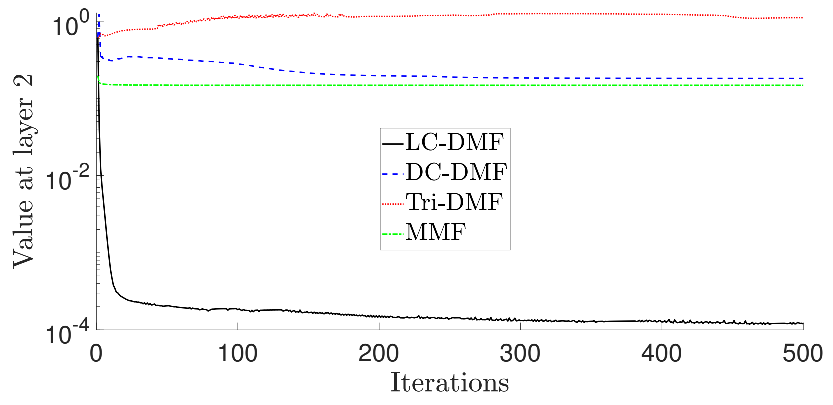
[8]`http://www.ai.mit.edu/courses/6.899/lectures/faces.tar.gz`

[9]In comparison, the average Hoyer sparsity of W_1, W_2 and W_3 obtained with MMF without sparsity constraints are respectively 71.97%, 79.46% and 76.64%.

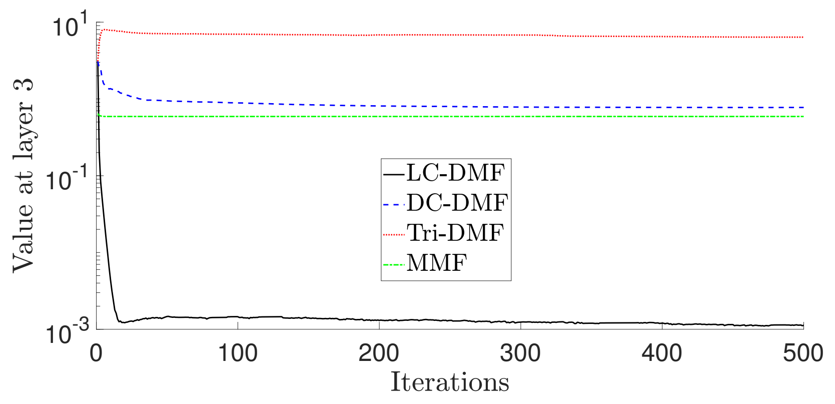
[10]The reader may be surprised to see an interpretation rather different than the one of Fig. 4.2 in Section 4.4.1.1. However, let us emphasize that the decompositions are also very different: on Fig. 4.2, the *abundances* matrices H_l's were successively decomposed while on Fig. 6.7, these are the *features* matrices W_l's which are decomposed. Hence, this leads to a reverse interpretation, with features going from the more specific to the more general ones on Fig. 6.7.



(a)

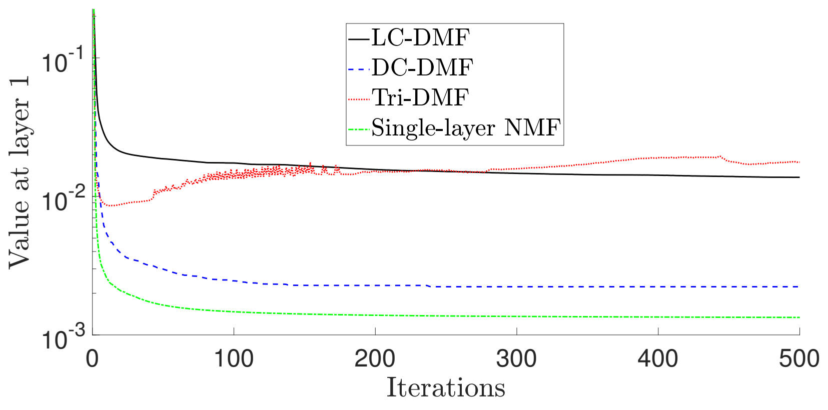


(b)

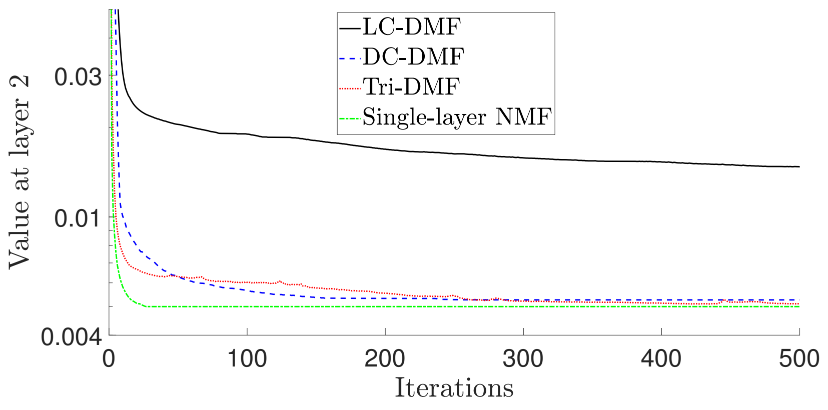


(c)

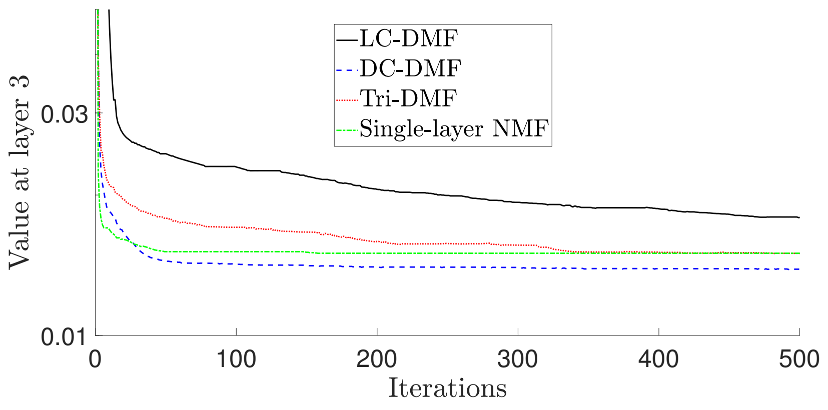
Figure 6.5: Layer-centric errors on the CBCL faces data set, with $L = 3$, $r_1 = 100$, $r_2 = 49$ and $r_3 = 25$ at the (a) first, (b) second, and (c) third layer.



(a)



(b)



(c)

Figure 6.6: Data-centric errors on the CBCL faces data set, with $L = 3$, $r_1 = 100$, $r_2 = 49$ and $r_3 = 25$ at the (a) first, (b) second, and (c) third layer.

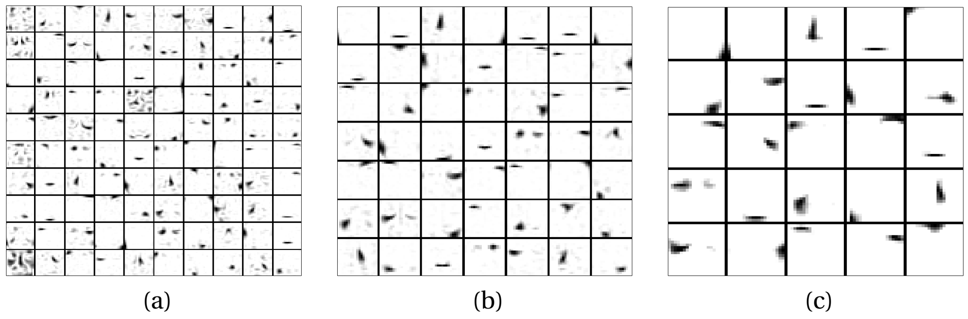


Figure 6.7: Features extracted by MMF in the CBCL faces data set, with $L = 3$, $r_1 = 100$, $r_2 = 49$, and $r_3 = 25$. Each image contains the features extracted at: (a) first layer W_1, (b) second layer W_2, and (c) third layer W_3.

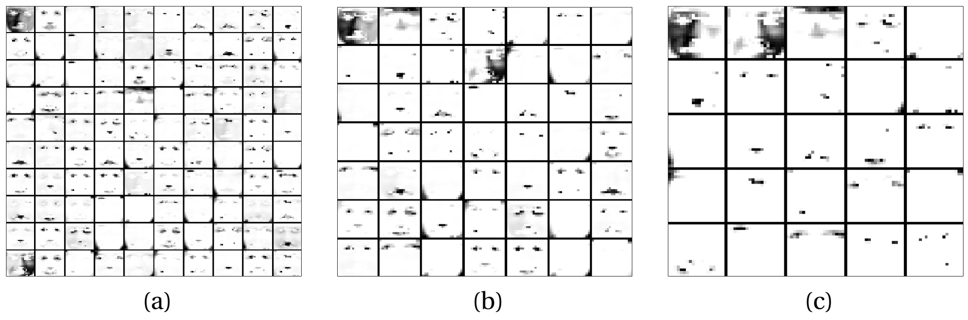


Figure 6.8: Features extracted by LC-DMF in the CBCL faces data set, with $L = 3$, $r_1 = 100$, $r_2 = 49$, and $r_3 = 25$. Each image contains the features extracted at: (a) first layer W_1, (b) second layer W_2, and (c) third layer W_3.

This comparison clearly confirms the advantage of DC-DMF and LC-DMF over MMF and Tri-DMF:

- At the second and third layers, DC-DMF produces the lowest data-centric errors while LC-DMF produces the lowest layer-centric errors. Note however that DC-DMF has slightly larger errors than single-layer NMF at the first two layers, while it has a lower error at the third layer. This is expected since DC-DMF optimizes all layers simultaneously while single-layer NMF performs independent NMFs at all "layers".

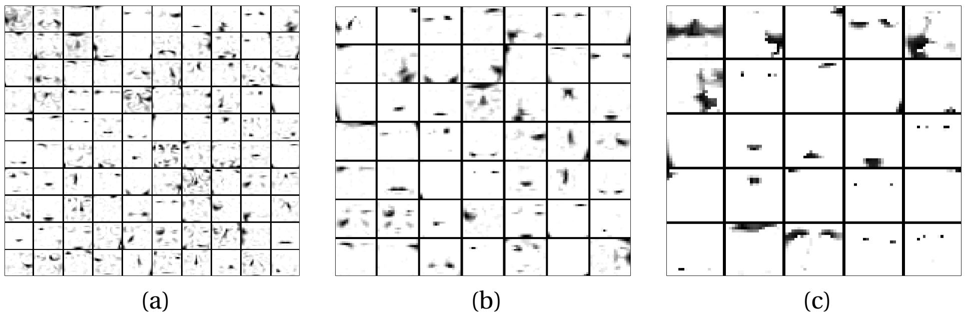


Figure 6.9: Features extracted by DC-DMF in the CBCL faces data set, with $L = 3$, $r_1 = 100$, $r_2 = 49$, and $r_3 = 25$. Each image contains the features extracted at: (a) first layer W_1, (b) second layer W_2, and (c) third layer W_3.

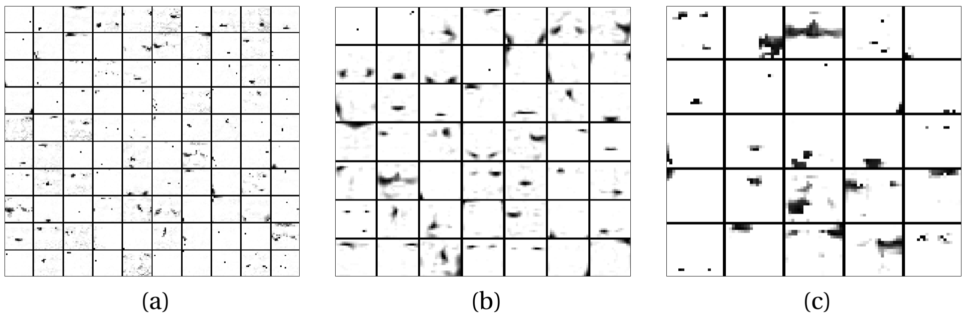


Figure 6.10: Features extracted by Tri-DMF in the CBCL faces data set, with $L = 3$, $r_1 = 100$, $r_2 = 49$, and $r_3 = 25$. Each image contains the features extracted at: (a) first layer W_1, (b) second layer W_2, and (c) third layer W_3.

- MMF has much higher relative errors than LC-DMF at the second and third layers (MMF is above 10^{-1} while LC-DMF is below or about 10^{-3}). This comes from the sequential optimization procedure of MMF. More precisely, the factors of the first layer, W_1 and H_1, are first extracted, then those of the second layer, and so on, without any possibility of "backpropagation".

- The error of Tri-DMF at the first layer oscillates and does not converge, which is an expected consequence of the different loss functions minimized at each layer; see the discussion of Section 5.1 in Chapter 5 for more details.

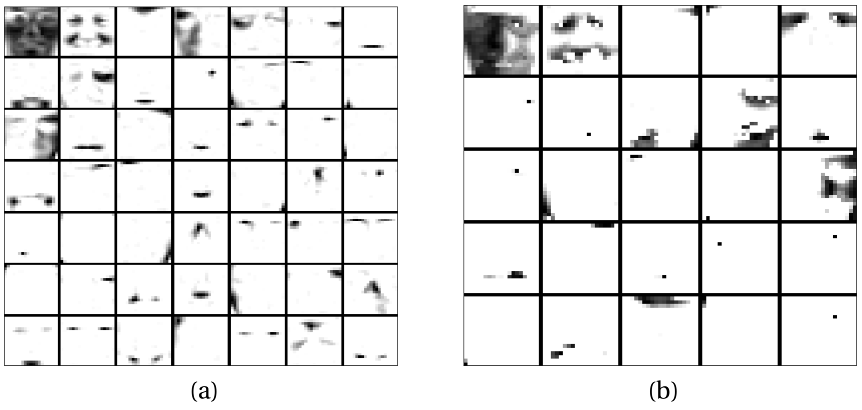


Figure 6.11: Features extracted by single-layer NMF in the CBCL faces data set, with (a) $r = 49$, (b) $r = 25$.

6.3 Minimum-volume deep matrix factorizations

In this section, we apply the framework of Chapter 5 to minimum-volume deep MF, the extension of minVolNMF (see Section 2.3.2 in Chapter 2) to several layers. In other words, the basis vectors of all layers are constrained to be as close as possible to the data points, through the minimization of the volume of the columns of each W_l for $l = 1,\dots,L$. As for minVolNMF, this aims at increasing the interpretability of the factors.

To alleviate the degree of freedom inherent to NMF (see Section 1.2), a usual additional constraint is to impose that all the columns of H sum to 1. However, in the minVol approach, it turns out that imposing the column-stochasticity of W instead of H, that is, $W^T e = e$, where e is the vector of all ones of appropriate dimension, leads to better results in many applications. When the elements of each column of W sum to one, W is better conditioned and normalizing the columns of W better balances their importance in the volume regularization term, see the discussion in Section 4.3.3 of [56] for more details.

To build minVol deep MF, we simply extend the approach of [53] by incorporating a volume contribution at every layer. More precisely, we add the following quantity[11] to the reconstruction error of each layer for both

[11] See Section 2.3.2 for a more detailed explanation of the minVol regularization.

loss functions given by Eq. (5.3) and Eq. (5.4):

$$\kappa_l \log\det(W_l^T W_l + \delta I_{r_l}) \tag{6.8}$$

while imposing column-stochasticity on every W_l for $l = 1, \dots, L$.

The volume contribution implies an additional term in the gradients of both $\mathscr{L}_1$ and $\mathscr{L}_2$ with respect to W_l's. More precisely, the term $\kappa_l W_l Z$, multiplied either by λ_{l-1} or μ_{l-1} is added to Eq. (5.5) and Eq. (5.7) respectively, with $Z = (W_l^{(*)^T} W_l^{(*)} + \delta I_{r_l})^{-1}$ where $W_l^{(*)}$ denotes the last iteration of W_l such that Z is constant during a given update of W_l.

A potential drawback of such an approach is the use of many regularization parameters, both for the weights of the terms of the loss functions and the volume penalties at each layer. Indeed, in addition to the $L-1$ parameters λ_l's of Eq. (5.3) or μ_l's of Eq. (5.4), the user has to fix the values of the L parameters κ_l's involved in the volume regularization. In practice, the κ_l for a given layer l is set as follows. Given the initial error $err_l^{(0)} = \|W_{l-1}^{(0)} - W_l^{(0)} H_l^{(0)}\|_F^2$ (for the layer-centric loss function) or $err_l^{(0)} = \|X - W_l^{(0)} H_l^{(0)} \cdots H_1^{(0)}\|_F^2$ (for the data-centric loss function) and a first guess $\tilde{\kappa}_l$, the final value κ_l is given by $\kappa_l = \tilde{\kappa}_l \frac{err_l^{(0)}}{|\log\det(W_l^{(0)^T} W_l^{(0)} + \delta I_{r_l})|}$, such that both error terms are of the same order of magnitude for a given layer.

We now apply minVol deep MF to the hyperspectral unmixing of the Urban image.

We consider a 3-layers network, with $r_1 = 6$, $r_2 = 4$ and $r_3 = 2$. To initialize the basis vectors of all layers, we use hierarchical clustering (HC) [58], see Section 2.2 for more details. Instead of simply initializing the factors of LC-DMF, DC-DMF and Tri-DMF with HC, we refine the initialization obtained with HC by performing a few iterations of BCD before moving to the initialization of the next layer. Moreover, after several trials, it turned out that appropriate values for the minVol hyperparameters $\tilde{\kappa}_l$'s are 10^{-2} for all l. We also run single-layer NMF for $r = 4$ and $r = 2$ (at the first layer, it would coincide with MMF) to evaluate the efficiency of deep approaches compared to a shallow one.

On Fig. 6.12 and 6.13, we plot the spectral signatures of the materials extracted by the different methods at the first and second layer respectively, that is, the columns of W_1 and W_2, and the ground truth from [135], which is only available for $r = 6$ and $r = 4$ to the best of our knowledge. The MRSA's

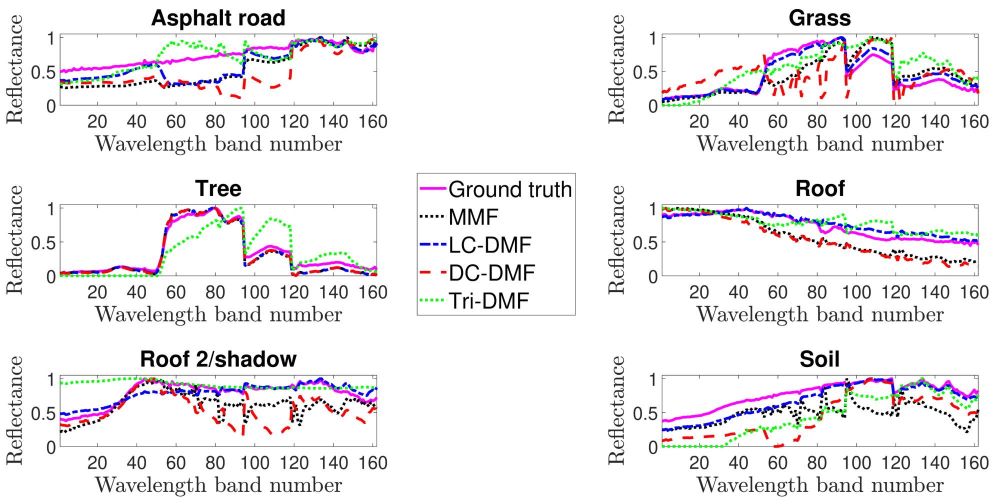


Figure 6.12: Endmembers extracted by MMF, LC-DMF, DC-DMF and Tri-DMF in the Urban image at the first layer ($r_1 = 6$), and the ground truth.

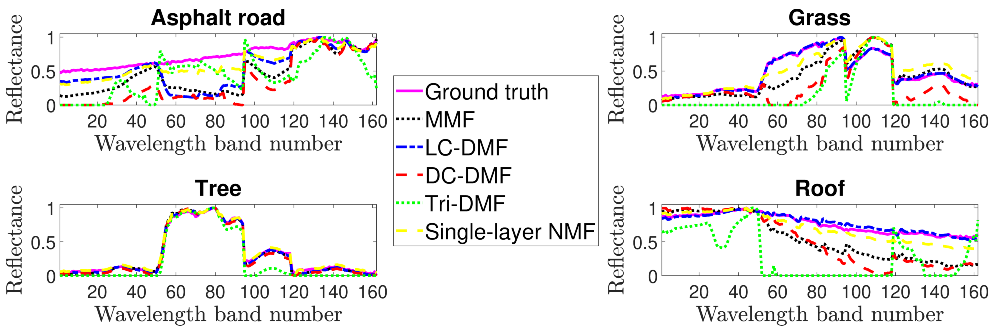


Figure 6.13: Endmembers extracted by MMF, LC-DMF, DC-DMF, Tri-DMF and single-layer NMF in the Urban image at the second layer ($r_2 = 4$), and the ground truth.

at both layers are presented in Table 6.2 with the best result at each layer (that is, the lowest MRSA) in bold.

At both layers, LC-DMF clearly outperforms the other deep methods, including Tri-DMF and MMF. Especially, Tri-DMF performs the worse. Moreover, at the second layer, LC-DMF achieves a MRSA very close to the one of single-layer NMF, which was run independently for both rank values. Therefore, LC-DMF seems again to be the best loss function to minimize when tackling deep MF.

Method	First layer ($r_1 = 6$)	Second layer ($r_2 = 4$)
MMF	16.98	12.35
LC-DMF	**9.48**	8.42
DC-DMF	22.95	14.15
Tri-DMF	26.07	20.07
Single-layer NMF	16.98	**7.74**

Table 6.2: MRSA at the first and second layer of the compared methods on the Urban hyperspectral image, with in bold the best value of each column.

The corresponding abundance maps at all layers are available in Appendix of [35] for all the compared methods. However, since the hierarchy of materials is similar to what was presented in Sections 4.4.1.2 and 6.1.3.2, we do not display them here, in order to keep the reading pleasant. Globally, the hierarchy of materials extracted by LC-DMF is rather sparse, with each material obtained by a combination of only a few materials of the previous layer. On the opposite, the decomposition of Tri-DMF is hardly interpretable since all materials contribute to those of the next layer with important proportions.

6.4 Deep symmetric matrix factorizations

In this section, we introduce a variant of deep MF where the input matrix is symmetric and nonnegative, dubbed deep symmetric nonnnegative matrix factorization (DSNMF).

Symmetric NMF (symNMF) [70] requires the data matrix X to be nonnegative and symmetric, that is, $X = X^T$ with $m = n$, and $H = W^T$. Hence, X is approximated by WW^T. SymNMF is appropriate when the entries of X represent the similarities between different items, for example a word co-occurrence matrix in topic modeling [64], or when X is the adjacency matrix of an undirected graph. In this context, symNMF extracts communities of nodes, possibly overlapping, such that the nodes of a given community have more connections (that is, edges) with each other than with nodes belonging to other communities.

The goal of DSNMF is to leverage L levels of factorization to give at each layer l a nonnegative symmetric approximation of rank r_l of the original matrix $X \in \mathbb{R}^{n\times n}$. More precisely, at the first layer, X is approximated by $W_1 W_1^T$ where $W_1 \in \mathbb{R}_+^{n\times r_1}$, as in symNMF. Let X be the symmetric adjacency matrix of a graph. In this case, each column of W_1 can be interpreted as a community, with $W_1(i,k)$ being the indicator of node i to belong to community k. In fact, $X \approx \sum_{k=1}^{r_1} W_1(:,k)W_1(:,k)^T$ means that X is approximated as the sum of r_1 communities which are rank-one nonnegative adjacency matrices. At the second layer, the matrix W_1 is factorized as $W_1 \approx W_2 H_2$ with $W_2 \in \mathbb{R}_+^{n\times r_2}$ and $H_2 \in \mathbb{R}_+^{r_2\times r_1}$, and $r_2 < r_1$. This gives a new symmetric approximation of X, namely $X \approx W_2(H_2 H_2^T)W_2^T$. At the second layer, each column of W_2 indicates to what extent the n data points belong to one of the r_2 communities. The square inner matrix $H_2 H_2^T \in \mathbb{R}^{r_2\times r_2}$ indicates how strongly the r_2 communities interact with each other. In fact, the second layer of DSNMF is a particular case of symmetric nonnegative tri-factorization, namely $X \approx WSW^T$ where $S = H_2 H_2^T$ [121]. As the factorization unfolds, the r_l columns of the matrices W_l's identify fewer communities (as the ranks of the factorization are decreasing) and the inner square matrices $H_l \cdots H_2 H_2^T \cdots H_l^T$ indicate how the numerous small communities of the first layers progressively gather in fewer larger communities at the last layers.

Let us briefly illustrate the idea behind DSNMF with $L = 2$ on the synthetic graph of Fig. 6.14, made of 14 nodes. DSNMF applied with $r_1 = 4$ and $r_2 = 2$ splits the nodes in four communities at the first layer, namely containing the nodes $\{1,2,3,4\}$, $\{4,5,6,7\}$, $\{8,9,10,11\}$ and $\{11,12,13,14\}$. They correspond to the sets of nodes surrounded by a solid line. Note that nodes 4 and 11 belong with the same proportion to two communities. Then, at the second layer, only two communities remain (dashed circles), obtained by merging respectively the first two and the last two communities of the first layer.

In Section 6.4.1, we present an algorithm to solve DSNMF while in Section 6.4.2, we perform experiments on synthetic data. The experiments on real data are specifically discussed in Chapter 7.

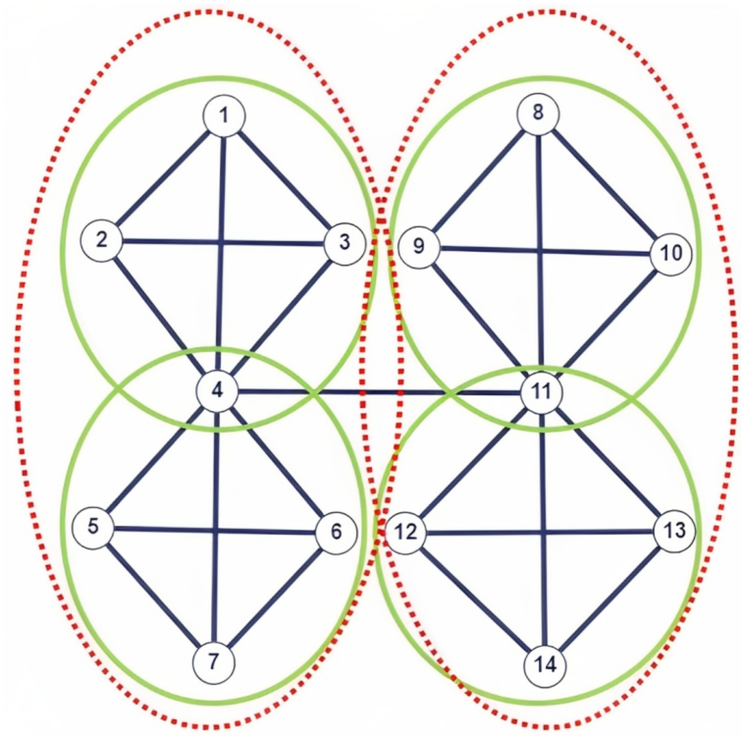


Figure 6.14: Simple graph to illustrate the working of DSNMF, with two levels of communities.

6.4.1 An algorithm for DSNMF

Given a nonnegative symmetric matrix $X \in \mathbb{R}_{+}^{n\times n}$, standard symNMF consists in solving the following optimization problem:

$$\min_{W\in\mathbb{R}_{+}^{n\times r}} \frac{1}{2}\|X - WW^T\|_F^2.$$

However, to avoid dealing with a fourth-order objective function in W, an alternative formulation was proposed in [70]:

$$\min_{\substack{W\in\mathbb{R}_{+}^{n\times r}\\ H\in\mathbb{R}_{+}^{r\times n}}} \frac{1}{2}\left(\|X - WH\|_F^2 + \mu\|W - H^T\|_F^2\right). \tag{6.9}$$

For μ sufficiently large, it has been shown that $W = H^T$ holds for the critical points of Eq. (6.9) [81]. Extending Eq. (6.9) to L layers requires to define an appropriate loss function. Inspired by those proposed in Chapter 5, we minimize

$$\begin{aligned}\mathscr{L}_{DSNMF} = \frac{1}{2}\Big(&\|X - W_1 H_1\|_F^2 + \mu_1 \|W_1 - H_1^T\|_F^2 + \\ \lambda_1\big(&\|W_1 - W_2 H_2\|_F^2 + \mu_2 \|W_2 - (H_2 H_1)^T\|_F^2\big) + \cdots + \\ \lambda_{L-1}\big(&\|W_{L-1} - W_L H_L\|_F^2 + \mu_L \|W_L - (H_L H_{L-1} \cdots H_1)^T\|_F^2\big)\Big).\end{aligned} \tag{6.10}$$

This layer-centric loss function performs a weighted sum of layer-wise errors, that is, $err(l) = \|W_{l-1} - W_l H_l\|_F^2 + \mu_l \|W_l - (H_l \cdots H_1)^T\|_F^2$ for $l = 1, \ldots, L$, with $W_0 = X$ and $\lambda_0 = 1$. Each layer-wise error is in turn the sum of two contributions. The first one, namely $err_1(l) = \|W_{l-1} - W_l H_l\|_F^2$, is the reconstruction error at layer l, that is, the error between W_{l-1} and its approximation $W_l H_l$ of rank r_l. The second term, namely $err_2(l) = \mu_l \|W_l - (H_l \cdots H_1)^T\|_F^2$, ensures the symmetry of the factorization, using the same trick as in Eq. (6.9).

To minimize $\mathscr{L}_{DSNMF}$ defined in Eq. (6.10), we use the BCD method presented in Algorithm 11 in Section 5.2, with the blocks of variables W_l's and H_l's. Again, the subproblems with respect to a given factor matrix are solved with FPGD. For this purpose, let us express the gradients of $\mathscr{L}$ with respect to each set of variables. First, let us call for all l, $D_l = H_{l-1} \cdots H_1$, $\tilde{H}_l = H_l \cdots H_1 = H_l D_l$, $C_l^{(k)} = H_k \cdots H_{l+1}$ for $k \geq l$ (for $k = l$, $C_l^{(l)}$ is the identity matrix of appropriate dimension). We have:

$$\frac{\partial \mathscr{L}}{\partial W_l} = \lambda_{l-1}\big((W_l H_l - W_{l-1}) H_l^T + \mu_l (W_l - \tilde{H}_l^T)\big) + \delta_l \lambda_l (W_l - W_{l+1} H_{l+1})$$

with $\delta_l = 0$ if $l = L$ and $\delta_l = 1$ otherwise, and

$$\frac{\partial \mathscr{L}}{\partial H_l} = \lambda_{l-1} W_l^T (W_l H_l - W_{l-1}) + \sum_{k=l}^{L} \lambda_{k-1} \mu_k (D_l (\tilde{H}_k^T - W_k) C_l^{(k)})^T.$$

A crucial aspect of deep MF models is the choice of the hyperparameters, namely the number of layers L, and the factorization ranks r_l's. In the following, we suggest two ways of initializing DSNMF:

- When the depth L and the ranks r_l's are given by the user, the initial factors $W_l^{(0)}$'s and $H_l^{(0)}$'s are initialized with a sequential multilayer approach, as in [27]. This is the strategy that we chose for experiments on synthetic data since it provides control on the network architecture.

- When no prior information on the network is provided, we resort to the well-known Louvain method (LM) [16]. LM is a widely used algorithm that extracts communities of nodes in a graph by trying to increase the so-called network modularity at each iteration. In a nutshell, the modularity is a scalar value that measures the density of links inside communities as compared to links between communities. LM starts with each node representing its own community and then tries to move nodes from one community to another. After each iteration t, LM provides a split of the graph into r_t disjoint communities, that is, with each node belonging to a single community. In other words, LM extracts a bottom-up hierarchy of communities inside a graph, similarly to what the deep MF models (in the broad sense) do, except that for these lasts, nodes can simultaneously belong to several communities, with some proportions. Hence, in the absence of values provided by the user, we set the number of layers L of DSNMF to be equal to the number of iterations of LM and the ranks r_l's are taken as the number r_t's of communities successively extracted by LM. The initial matrices $W_l^{(0)}$'s are built for all l such that each column corresponds to a community extracted by LM at iteration l.

6.4.2 Experiments on synthetic data

In this section, we apply DSNMF to synthetic data. DSNMF can be interpreted as a hierarchical fuzzy clustering approach hence no clear ground truth is available. More precisely, the features extracted at each factorization layer are not known in advance, which renders the quantitative assessment of the model challenging, even on synthetic data.

In the following experiments, we compare three algorithms, namely:

- DSNMF; to set up the parameters λ_l's for $l = 1, ..., L-1$ and μ_l's for $l = 1, ..., L$ in the loss function of Eq. (6.10), we balance the importance of each layer and proceed as follows. The λ_l's are chosen such that the initial contributions of all layers to the loss function are the same, that is, $\lambda_l = \frac{err^{(0)}(1)}{err^{(0)}(l+1)}$. Similarly, the μ_l's are such that for all l, $err_1^{(0)}(l) = err_2^{(0)}(l)$, that is $\mu_l = \frac{\|W_{l-1}^{(0)} - W_l^{(0)} H_l^{(0)}\|_F^2}{\|W_l^{(0)} - (H_l^{(0)} \cdots H_1^{(0)})^T\|_F^2}$.

- Multilayer symmetric NMF (MSNMF); this is the symmetric version of the sequential multilayer factorization of Cichocki et al. [26]. In other words, the symmetric factorizations are successively performed independently layer by layer.

- Spectral clustering (SpecClust) applied at each layer; this is a well-known clustering algorithm [119] which applies k-means to the n rows of the matrix whose columns are the first (that is, smallest) k eigenvectors of the graph Laplacian matrix (see Section 4.2.2.4). Hence, SpecClust performs a "hard" community detection (that is, generates disjoint communities), contrary to DSNMF and MSNMF.

We build our dataset in a similar way as the toy example of Fig. 6.14. More precisely, we set $L = 2$, $r_1 = 4$ and $r_2 = 2$. The noiseless graph consists of two disjoint sub-graphs of the same size, themselves composed of two cliques, that is, subsets of vertices that are all connected to each other, of the same size that have n^* nodes in common, which is the same for both sub-graphs. For $n = 14$ and $n^* = 1$, this is exactly the situation represented on Fig. 6.14 if the edge between nodes 4 and 11 is removed. We add symmetric white Gaussian noise to the noiseless adjacency matrix $\tilde{X}$ the same way as described in Section 2.4.1 of Chapter 2. For different relative noise levels v and for several combinations of n and n^*, we run the three methods described above with 25 different randomly generated noise matrices N. Table 6.3 reports the average and standard deviation of the MRSA between the columns of the ground truth and of the computed factors W_l's over these 25 runs, at both layers.

The interpretation of the results is not easy since no method clearly outperforms the others.

At the first layer, DSNMF and MSNMF perform comparably, and outperform spectral clustering, especially in more challenging settings, when the clusters are more overlapping (that is, when n^* is larger). It was expected that MSNMF performs well on the first layer since it optimizes the first layer independently of the next ones. It is reassuring to see that DSNMF performs comparably: although it has to take into account the decomposition at the second layer, the error at the first layer is similar to that of MSNMF. On the other hand, spectral clustering only extracts disjoint communities, which is a limitation since the communities at the first layer are considerably overlapping.

$(n,\ n^*, v)$	DSNMF	MSNMF	SpecClust
(14, 1, 0.01)	**0.10** ± 0.01	0.11 ± 0.02	9.54
(14, 1, 0.05)	**0.53** ± 0.08	0.54 ± 0.08	9.54
(14, 1, 0.1)	**1.06** ± 0.16	1.07 ± 0.17	9.54
(14, 1, 0.5)	5.95 ± 0.71	**5.65** ± 0.73	9.54
(100, 10, 0.01)	**0.05** ± 0.00	**0.05** ± 0.00	11.17
(100, 10, 0.05)	0.27 ± 0.01	**0.23** ± 0.01	11.17
(100, 10, 0.1)	0.55 ± 0.02	**0.46** ± 0.02	11.17
(100, 10, 0.5)	3.78 ± 0.15	**2.52** ± 0.12	11.33 ± 0.16
(100, 30, 0.01)	0.07 ± 0.00	**0.06** ± 0.00	18.31
(100, 30, 0.05)	0.34 ± 0.01	**0.31** ± 0.01	18.31
(100, 30, 0.1)	0.72 ± 0.03	**0.62** ± 0.03	20.19 ± 5.09
(100, 30, 0.5)	6.37 ± 0.20	**3.40** ± 0.19	18.31

(a)

$(n,\ n^*, v)$	DSNMF	MSNMF	SpecClust
(14, 1, 0.01)	2.39 ± 7.72	5.09 ± 7.26	**0**
(14, 1, 0.05)	2.42 ± 4.90	5.94 ± 6.30	**0**
(14, 1, 0.1)	5.41 ± 9.24	19.98 ± 11.84	**0**
(14, 1, 0.5)	21.54 ± 12.47	32.05 ± 12.25	**14.49** ± 17.04
(100, 10, 0.01)	0.05 ± 0.00	8.02 ± 5.14	**0**
(100, 10, 0.05)	0.29 ± 0.05	1.75 ± 1.18	**0**
(100, 10, 0.1)	0.56 ± 0.08	1.64 ± 1.10	**0**
(100, 10, 0.5)	3.85 ± 0.85	23.85 ± 7.47	**0**
(100, 30, 0.01)	0.04 ± 0.00	19.78 ± 2.45	**0**
(100, 30, 0.05)	0.19 ± 0.01	20.72 ± 2.40	**0**
(100, 30, 0.1)	0.39 ± 0.02	20.88 ± 2.00	**0**
(100, 30, 0.5)	2.10 ± 0.16	21.64 ± 2.02	**0**

(b)

Table 6.3: Comparison of the MRSA (average and standard deviation) of DSNMF, MSNMF and SpecClust on synthetic data over 25 runs in function of the noise level v and the configuration of the network at the (a) first ($r_1 = 4$) and (b) second ($r_2 = 2$) layer. The best average MRSA achieved for each configuration is highlighted in bold.

At the second layer, MSNMF completely fails, showing the limitations of a purely sequential multilayer approach. This behaviour is likely to worsen when the number of layers increases. Spectral clustering nearly performs a perfect clustering at the second layer, which is expected since the two corresponding communities are disjoint (in the noiseless case). In fact, since SpecClust performs the last clustering stage through k-means, it generates disjoint communities, that is, binary degrees of membership, unlike NMF-based models. Hence, for reasonable levels of noise, since there are only two communities, DSNMF is likely to retrieve the ground-truth factors perfectly, which explains why the MRSA is equal to 0. It is also important to keep in mind that SpecClust always works on the original data without the need to keep a balance between several layers, unlike DSNMF. Since it is by definition a single-layer method, SpecClust is not able to interpret the links between successive levels of communities, as opposed to DSNMF.

In summary, DSNMF is the only method able to balance both layers for the various tested configurations.

6.5 Perspectives of improvement

This section lists some perspectives of improvement related to the deep MF models proposed in this chapter.

First, concerning the SODA initialization (see Section 6.1), it would be interesting to test such an initialization on other applications, possibly combined with a cheaper method on top of it. Also, a thorough study of the robustness to noise of SODA would be interesting.

Concerning the application of the new loss functions described in Chapter 5 to various deep MF models, especially sparse deep MF in Section 6.2 and minVol deep MF in Section 6.3, an important direction of research is to find clever ways of choosing and tuning the regularization parameters in the loss functions. Another perspective is to embed deep MF in a more powerful optimization framework such as TITAN [73] which has proven to be particularly efficient to tackle non-smooth non-convex problems, hence would be appropriate to solve grouped sparse deep MF.

Finally, concerning DSNMF (Section 6.4), it would be interesting to experiment other initialization strategies for real data, especially when the number of layers of factorization is unknown. In our experiments, we used the Louvain method which extracts a hierarchy of disjoint communities but has the drawback to assign each node to a single community at each step. Another important axis of research according to us would be to define proper metrics to assess the quality of a hierarchical fuzzy clustering, whose deep (symmetric) MF is an example.

6.6 Take-home messages

In this chapter, we introduced various deep MF models. Let us summarize the main points:

- The successive orthogonal decomposition algorithm (SODA) is an interesting initialization algorithm for deep MF that can easily be hybridated with cheaper techniques when applied to real applications.
- The two new loss functions developed in Chapter 5 are particularly appropriate to solve various deep MF variants and defeat both multilayer MF and Trigeorgis deep MF on synthetic data and real applications. In particular, we recommend to use the layer-centric loss function defined in Eq. (5.3), that is, LC-DMF, to tackle deep MF as it performs the best on average in our numerical experiments.
- We introduced deep symmetric NMF, able to extract hierarchies of overlapping communities within networks. With the Louvain method of community extraction as initialization, it allows to automatically fix the number of layers and inner ranks of the factorization without user involvement (this will be illustrated in the next chapter).

CHAPTER 7

Application of deep matrix factorizations to psychiatric networks

In this chapter, we apply deep MF, especially DSNMF (see Section 6.4 in Chapter 6) to psychiatric networks, an emerging application, which has never raised interest among the NMF community before, to the best of our knowledge. First, in Section 7.1, we give a brief overview of community detection in psychiatric networks. Then, in Section 7.2, we apply DSNMF to the extraction of hierarchical communities within psychiatric networks and emphasize the clinical interpretation of the obtained results. We discuss perspectives of future research in Section 7.3. Finally, Section 7.4 summarizes the key messages of this chapter.

7.1 A short introduction to psychiatric networks

Psychiatric networks have become an important area of research in recent years to investigate the interconnections between psychiatric symptoms and their underlying mechanisms. One key aspect is the identification of communities of symptoms that are more strongly connected to each other than to symptoms in other communities. As a consequence, a novel theoretical framework of mental health disorders has recently been developed among psychiatric researchers : the network approach [104]. The network approach views mental disorders as complex systems of interconnected symptoms rather than discrete categories.

Let us consider a data matrix $Y \in \mathbb{R}^{m \times n}$ containing the ratings of m subjects on n symptoms (or items) on an ordinal scale. Given Y, the symmetric input matrix $X \in \mathbb{R}^{n \times n}$ is made of the partial correlations between the n symptoms, that is, $X(i,j) = -\frac{K(i,j)}{\sqrt{K(i,i)K(j,j)}}$ where $K = \Sigma^{-1}$ is the precision matrix, defined as the inverse of the covariance matrix of the columns of Y [44]. Partial correlation between two variables is measured while controlling for the effects of the other variables. In other words, it measures the degree of association between two variables while holding all other variables constant. Such a model, that is, a network in which the adjacency matrix is made of the partial correlations between the items, is referred to as a pairwise Markov random field (PRMF). To sparsify the adjacency matrix of such network, it is common to first apply LASSO regularization on the precision matrix K, together with the minimization of the extended Bayesian information criterion (EBIC) to find the best LASSO regularization parameter [62].

Considering the graph whose adjacency matrix is X and where each node corresponds to a symptom, it is interesting to identify its communities of nodes. This would allow us, for example, to evaluate the set of symptoms that will be somehow impacted by an action (such as a medication) on a particular symptom. Most works in this recent field only extract one level of disjoint communities, which does not allow to finely grasp all the possible interactions in the network. Currently, two of the most popular community detection algorithms in psychiatric networks are:

- The spinglass algorithm [103]: This is a method based on the physical concept of a spin glass, a material with randomly oriented magnetic spins. In the context of community detection, the spinglass algorithm models the network as a system of interacting "spins" (nodes) with a certain energy function that depends on the spin configuration (the community assignment). The algorithm seeks to find the configuration that minimizes the energy function, which corresponds to the optimal community structure. The communities extracted are disjoint and the number of communities corresponds to the number of spin states in the optimal configuration of energy.

- The walktrap algorithm [100]: This is a method that leverages random walks over the nodes of the graph. More precisely, the algorithm starts by simulating random walks on the network and then computes pairwise similarities between nodes based on the number of common walks they share. This similarity matrix is then used to gather similar nodes in the same community. In fact, the Walktrap algorithm can be seen as a bottom-up clustering algorithm which starts by considering each node as a cluster, merges two clusters in one at each iteration and ends up with a single cluster containing all the nodes. Overall, the community assignment output to the user is the one that maximizes the modularity, which is a scalar value that measures the density of links inside communities as compared to links between communities. Hence, Walktrap provides disjoint communities of nodes and the exact number of communities output depends on the iteration for which the modularity is maximal.

While Spinglass and Walktrap have demonstrated their utility in community detection within psychiatric networks, these methods are only able to extract a single level of disjoint communities. Note that Christensen & al. [25] recently performed an exhaustive comparison of community detection algorithms, including Walktrap, Spinglass and LM, and network estimation methods in psychometric networks. They already briefly mention that the Louvain method has the advantage to provide a hierarchical structure of communities. However, their experiments assume that the ground-truth communities (and their number) are known in advance.

7.2 Application of DSNMF to psychiatric networks

In this section, we apply DSNMF to several psychiatric networks. First, in Section 7.2.1, we focus on post-traumatic stress disorder (PTSD). Then, in Section 7.2.2, we study the Connor-Davidson resilience scale [30].

7.2.1 Post-traumatic stress disorder

Post-traumatic stress disorder (PTSD) is a mental health condition that can develop in individuals who have experienced or witnessed a traumatic event, such as military combat, sexual assault, or natural disaster. PTSD is characterized by symptoms such as intrusive thoughts, avoidance behaviors, hyperarousal, and negative changes in mood and cognition. These symptoms can have a significant impact on an individual's daily life.

In the following, we consider a dataset [95] of 359 women suffering from PTSD, evaluated through the PTSD Symptom Scale-Self Report (PSS-SR) [50]. The PSS-SR is an ordinal scale assessing 17 symptoms of PTSD. It is built on the fourth version of the Diagnostic and Statistical Manual of Mental Disorders (DSM-IV) [8], a standard classification of mental disorders used by mental health professionals. Note that the current version of this manual is the fifth one and is denoted DSM-5; see below for some details. The scale assesses the frequency of some behaviours considered as characteristic of the pathology. This scale was expected to have three communities representing symptoms of arousal (for example, *being jumpy*), avoidance (for example, *avoiding reminders of the trauma*) and re-experiencing (for example, *having bad dreams about the trauma*).

The network of symptoms is built in R with the EBIC graphical LASSO regularization and is represented on Fig. 7.1. Note that negative partial correlations were set to 0 in order to apply DSNMF.

We apply DSNMF on this network with the LM initialization. LM extracts 2 layers of respectively 4 and 3 communities hence we perform DSNMF with $L = 2$, $r_1 = 4$, $r_2 = 3$. Fig. 7.1 displays the extracted communities at the first and second layers in green (solid line) and red (dashed line), respectively. For convenience, we only plot the main communities to which each node belongs to. More precisely, at layer l, for a given node, we first assign it to the community for which it has the largest degree of membership (that is, largest value in the corresponding row of W_l). Then, we sequentially assign it to more communities as follows: we assign it to the next community with the largest degree of membership only if this degree is at least 60% of the degree of the latest community assigned to this node. Once this condition is no longer satisfied, the assignment stops.

At the first layer, the 4 extracted communities are:

- $\{1,3,4,13,14\}$, which represents symptoms of avoidance and physical

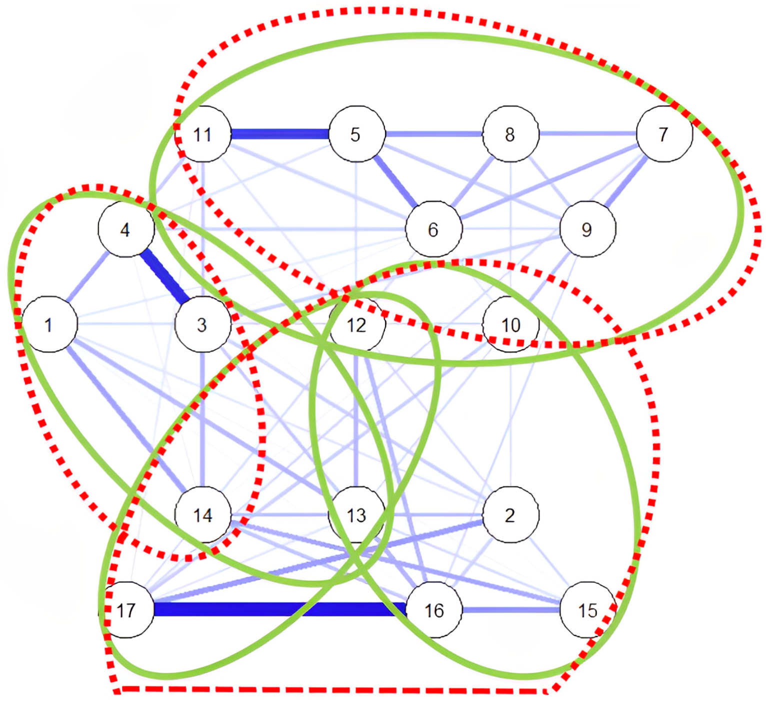


Figure 7.1: Communities extracted at the first and second layer by DSNMF on a PTSD dataset, in green (solid line) and red (dashed line), respectively. The thickness of the blue edges between two nodes is proportional to the corresponding (nonnegative) entry of the adjacency matrix.

reaction caused by the lack of avoidance of trauma reminders,

- $\{5,6,7,8,9,10,11,12\}$, which is clinically the most homogeneous community. These symptoms represent a negative mood and correspond to the community added in the new version of the scale (that is, DSM-5),
- $\{2,10,12,13,15,16\}$ and $\{12,13,14,17\}$ gather re-experiencing symptoms, attempts to avoid such re-experiences and physical reactions (arousal) when it fails.

Let us remark that node 10 and 14 belong to two communities and nodes 12 and 13 belong to three communities.

At the second layer, the 3 extracted communities are $\{1,3,4,14\}$, $\{5,6,7,8,9,10,11\}$ and $\{2,10,12,13,14,15,16,17\}$. Node 10 and 14 again belong to two communities. Roughly speaking, the second layer merges the two last communities of the first layer, keeping the two others mostly unchanged. This clinically makes sense: the third community of the first layer focuses on re-experiencing symptoms that appear during the sleep while the fourth one rather concerns avoidance of re-experiencing. Since these two communities are intimately related, they are merged together at the second layer.

The analysis of the different communities shows that the algorithm did not extract the different sub-scales of the PSS-SR, but rather extracted communities representing behaviours that are commonly presented together among patients. In other words, DSNMF did not extract the different symptoms of avoidance, re-experiencing and arousal distinctly, but behaviours that can be observed together. This could have strong clinical implication for disorders presenting complex interactions between multiple features, such as PTSD but also suicide behaviours.

Finally, it is to be noted that the communities extracted by DSNMF represent well the criticisms raised by the DSM-IV assessment of PTSD. The joint presentation of symptoms representing a negative mood and cognition led the research community to modify the assessment of PTSD in the DSM-5 [9]. Indeed, it was shown that certain items represented a different category of symptoms that the one originally conceptualized, and the formulation of certain items led to inconsistencies [98]. The analysis of the node shared by the most communities strengthens this interpretation. That is, node 12 has been largely questioned, re-written in the DSM-5 and finally added to the new community of symptoms of negative mood and cognition.

7.2.2 Resilience scale

The Connor-Davidson Resilience Scale (CD-RISC) is a widely used psychological assessment tool designed to measure an individual's resilience, or his ability to adapt and cope with stress and adversity [30]. The scale consists of 25 items, each of which measures a different aspect of resilience, such as personal competence, positive acceptance of change, and spiritual influences. Each item is rated on a 5-point scale (0 = not true at all to 4 = true nearly all of the time), yielding a total score between 0 and 100, with higher scores reflecting a higher level of "resilient behaviours".

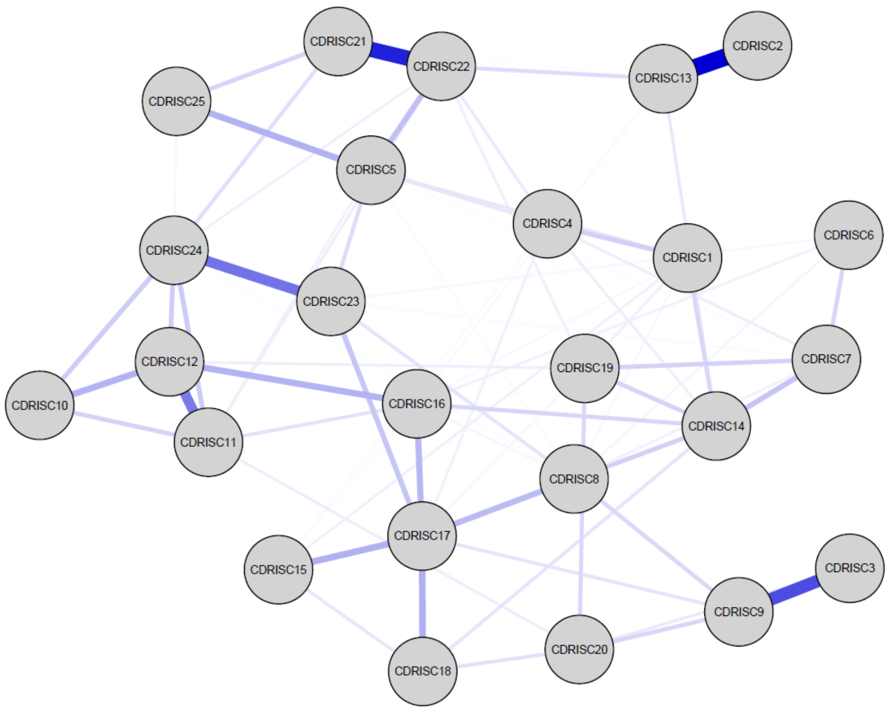


Figure 7.2: Regularized partial correlation resilience network. Blue edges represent positive connections; the thicker the connection is, the stronger it is.

Given a dataset of 408 individuals evaluated on the CD-RISC scale, we build a network as described in Section 7.1. Then, we apply DSNMF on this network with LM initialization. Both Spinglass and Walktrap algorithms are also applied as a comparison. We also compare our results with LM since on the one hand, it is a well known community detection technique and on the other hand, it serves as initialization for DSNMF. The network is represented on Fig 7.2. All the edges have positive weights representing regularized partial correlations between variables. In the following, we describe the communities extracted by the different algorithms, along with their clinical interpretation. A summary of the different communities and corresponding items is presented in Table 7.1.

Communities	Spinglass	Walktrap	LM Layer 1	LM Layer 2	DSNMF Layer 1	DSNMF Layer 2
1	1,4,6,7,8,14,15, 16,17,18,19	3,9,20	1,4,6,7, 8,14,19	1,4,6,7,8,14, 15,17,18,19	1,4,7,8,15, 16,18,19	1,4,6,7,8,14, 15, 16,17,18,19
2	10,11,12,23,24	5,10,11,12,21, 22,23,24,25	2,13	2,13	2,13	2,13
3	5,21,22,25	2,13	3,9,20	3,9,20	3,9,20	3,9,20
4	2,3,9,13,20	1,4,6,7,8,14,15, 16,17,18,19	5,25	5,21,22,23,24,25	5,21	5,21,22,23,24,25
5			10,11,12,16	10,11,12,16	10,11,12,16,24	10,11,12,16,24
6			15,17,18		1,6,7,14,15,17	
7			21,22		22,25	
8			23,24		23	

Table 7.1: Communities extracted by the different algorithms in the resilience network.

7.2.2.1 Spinglass

The Spinglass algorithm extracts 4 communities. Community 1 is close to the factor "Tolerance to negative affects" derived from factor analysis of the French CD-RISC. Compared to this factor, item 15 (*Prefer to take the lead in problem solving*) replaces item 23 (*I like challenges*). Four out of five items of the community 2 are present among the second factor of the French CD-RISC, labelled "tenacity". Community 3 is identical to the third factor, that refers to forward thinking behaviours/self-confidence toward the future. Finally, community 4 is the most heterogeneous, gathering items relative to two distinct themes: faith and support seeking behaviours.

The communities extracted by Spinglass seem mirroring to some extent the classic factor structure, but are less heterogeneous than the latter. For example, the exclusion of item 23 from the first community improves the semantic coherence of the community. The same observation can be made for the second community, which is "sparser" than what factor analysis reveals (that is, the exclusion of item 13 and 15 that were present in the second factor of the French CD-RISC and which are semantically different from the rest). However, the extraction of only four communities leads to some remaining heterogeneity, as demonstrated by the fourth community.

7.2.2.2 Walktrap

The Walktrap algorithm also isolates 4 communities. Community 1 represents the reliance on faith or spiritual thinking. Community 2 gathers tenacity and forward-thinking behaviours. Community 3 represents social support seeking behaviours. Community 4 is identical to the community 1 of the Spinglass and refers to tolerance to negative affects.

Multiple similarities exist between these communities and those extracted by Spinglass. Two main differences arise. On the one hand, community 1 and 3 split community 4 of Spinglass, which refers to two semantically distinct constructs. On the other hand, community 2 of Walktrap is more heterogeneous than what Spinglass provides, gathering items from two different communities identified in Spinglass that describe two different themes (tenacity and forward-thinking behaviours).

7.2.2.3 Louvain Method

At the first layer, LM extracts 8 communities. Community 1 is more restricted than the corresponding communities extracted by Spinglass (community 1) or Walktrap (community 4). Indeed, items 15, 16, 17 and 18 were not included in this first community. Community 2 gathers social support seeking behaviours and community 3 represents the reliance on faith or spiritual thinking. Community 4 represents confidence and pride given by past adversities. Community 5 refers to tenacity as a personality trait. Community 6 is more heterogeneous and represents decision-making behaviours when facing adversities (items 15 and 18) as well as believe in one's abilities when confronted to a difficult event (items 17). Community 7 represents sense of control and purpose of one's life, and community 8 reflects a goal-oriented mindset, item 23 being *I like challenges*, and item 24 being *I work to attain my goals*.

At the second layer, LM extracts 5 communities. Community 1 combines community 1 and 6 from the first layer. Communities 2, 3 and 5 are unchanged, and community 4 gathers multiple communities from the first layer (4, 7, and 8). All items present in the second factor "self-confidence" of the French CD-RISC are included in this community, in addition with items 23 and 24, which reflect confidence in one's abilities to overcome future obstacles.

Interestingly, LM provides a different solution than what has been highlighted by both psychometric exploration of the French version of the CD-RISC and Spinglass or Walktrap algorithms. The first layer performs a precise decomposition of specific behaviors, such as within community 4. This leads to an increased number of easily interpretable communities. Only community 6 is somewhat more heterogeneous. At the second layer, smaller communities are gathered together, showing some similarities with Spinglass and Walktrap (for example, the two communities representing reliance on faith or spiritual thinking and social support seeking behaviours).

7.2.2.4 DSNMF

At the first layer, 8 communities are identified. Out of the 8 items of community 1, 7 are present in the factor "tolerance to negative affects" derived from factor analysis of the French CDR. Compared to this factor analysis, items retained here are less heterogeneous, with the exclusion for example of the item 23 *I like challenges*, which is somewhat outside the scope. Community 2 indicates social support seeking behaviours and community 3 represents the reliance on faith or spiritual thinking. Community 4 represents confidence and purpose given by past adversities. Three items of the community 5 were gathered in the second factor of the French CDR "tenacity". Once again, the items retained here are less heterogeneous, with the exclusion for example of the item 13 *I know where to turn for help*, which is somewhat outside the scope of tenacity. In community 6, 3 items overlap with the first community. There is a nuance in this community, which tends to go beyond the general ability to tolerate negative affect, but targets one's personal competence to do so (6: *I can use humour to face a situation*, 14: *Under pressure I can think clearly*, 17: *I think of myself as strong person*). Community 7 represents the feeling of control of one's life, and finally community 8 is isolating the item 23 *I like challenges.* Let us remark that nodes 1, 7, 15 and 16 belong to two communities.

At the second layer, 5 communities are extracted. Community 1 gathers communities 1 and 6 from the first layer, which gives a more general view of tenacity. Communities 2 and 3 remain unchanged. Community 4 is the same as LM's fourth community (at layer 2) and represents self-confidence in one's abilities to overcome future obstacles. Community 5 is a community close to the French CDR factor "tenacity". Once again, the items are less heterogeneous, with the exclusion for example of the item 13 *I know where to turn for help.*

The results derived from the four methods presented above allow us to highlight some advantages of DSNMF compared to existing methods.

First, both Spinglass and Walktrap tend to produce large communities, thematically heterogeneous. The increased number of smaller communities extracted by both LM and DSNMF therefore represents an advantage, as it improves the clinical interpretability by decreasing the heterogeneity within communities.

Second, LM and DSNMF incorporate a layered analysis of the community structure. That is, the first layer captures finer-grained communities within the network, while the second layer encompasses slightly larger communities that bring together multiple communities from the first layer. This layering allows for a more in-depth exploration of the different facets or dimensions of the disorder.

Third, compared to LM, DSNMF allows each symptom to belong to more than one community at a time. In other words, the communities are not necessarily disjoint but instead, some nodes, that is, symptoms, belong to several communities at a given layer, with some proportions. This highlights the fact that symptoms can share features across different communities, illustrating the intricate nature of mental health disorders.

7.3 Perspectives of improvement

Since this chapter was mostly application-oriented, the perspectives are mainly related to the clinical challenges to such a network approach. In particular, the choice of the scale on which the network is based, for a given disorder, can be discussed.

Of course, the DSNMF algorithm itself could be improved along several aspects. For example, one could add some regularization in the model, enforcing specific properties for the extracted communities. One could also consider a more complex model, involving three factors per layer, which is referred to as tri-symmetric NMF [131]. This would allow to identify more clearly how the communities interact with each other at all layers.

7.4 Take-home messages

In this chapter, we applied DSNMF to psychiatry networks. Let us summarize the main points:

- A recent trend in psychiatry is to build networks of symptoms and try to identify meaningful communities.
- DSNMF with LM initialization provides a hierarchical structure of overlapping communities, with valuable clinical interpretation.
- The communities extracted by DSNMF are less heterogeneous than those extracted with classical community detection algorithms (Walktrap, Spinglass), allowing a finer understanding of the symptoms structure.

CHAPTER 8

CONCLUSION

In this chapter, we first summarize the main contributions of this PhD in Section 8.1 and then list some possible perspectives of future research in Section 8.2.

8.1 Summary of the main contributions

Let us summarize the main contributions of this PhD, paper by paper:

- **Near-convex archetypal analysis [39]; Chapter 3**: Inspired by archetypal analysis (AA), a well-known NMF variant, we developed near-convex archetypal analysis (NCAA), a more flexible variant of AA. Indeed, while AA constraints the basis vectors to lie within the convex hull of the data points, NCAA allows them to lie a bit outside this convex hull, in a minimum-volume fashion. The experiments on both synthetic data and hyperspectral unmixing (HU) of the Urban image tend to support NCAA: it performs favourably compared to state-of-the-art minVol NMF, while at the same time being associated with an interesting geometric interpretation.

- **A survey on deep matrix factorizations [35]; Chapter 4**: In this survey, we covered the main aspects of deep matrix factorizations, the extension of standard MF to several layers. Starting from an historical perspective, we reviewed the main models and variants, incorporating various constraints or regularizations. We also discussed the choice of the parameters and the main applications of deep MF. More specifically, we provided three "homemade" showcase examples, namely the extraction of facial features, hyperspectral unmixing and recommender systems, to clearly demonstrate the benefits of deep factorizations. We also tried to make insightful connections between deep MF and neural networks. Finally, we listed some key challenges related to deep MF, such as the study of the identifiability of the models and a thorough investigation of the links between deep MF and deep learning. By doing so, we hope to pave the way for meaningful future researches in the field.

- **Deep orthogonal matrix factorization as a hierarchical clustering technique [35]; Chapter 6, Section 6.1**: Based on the closed-form solution of the orthogonal NMF of two data points, we developed a greedy initialization for deep MF that merges the data points two by two to create clusters. This approach was shown to be efficient for synthetic data, especially when one cluster is within the convex hull of the others. However, this initialization is untractable on large-scale problems such as HU, unless it is combined with a cheaper technique on top of it. Hence, generalizing SODA is still an open problem.

- **A consistent and flexible framework for deep matrix factorizations [36]; Chapters 5 and 6, Sections 6.2 and 6.3**: Based on the observation that the loss function optimized by most of the deep MF researchers is not consistent, we proposed two new loss functions which are weighted sums of layer-wise contributions. Together with a generic framework based on block-coordinate descent and fast projected gradient descent, we introduced two new deep MF variants, namely grouped sparse deep MF and minVol deep MF, that leverage these new loss functions. We empirically showed evidence that our loss functions are the only ones that guarantee good retrieval of the ground-truth factors in the case of both synthetic data and real-world applications, for which the layer-centric loss function appears to be the most appropriate. On the other hand, the mainstream multilayer MF of [26] and deep MF of [115] showed their limitations, with either high final errors or even non-converging errors.

- **Deep symmetric matrix factorization [37]; Chapters 6, Section 6.4 and 7**: We introduced deep symmetric NMF, the extension of symmetric NMF to several layers, in order to extract hierarchies of communities within networks. Such a model can benefit from "clever" initialization strategies, such as the Louvain method [16], to automatically determine the number of layers and inner ranks. In particular, we showed promising results on psychiatric networks, a breakthrough application in computational psychiatry, which had never been investigated before by NMF practitioners, to the best of our knowledge. At the time of writing the present manuscript, we were working on a new paper on this topic.

8.2 Perspectives for future works

Miscellaneous perspectives of improvement have already been evoked along the chapters (see the corresponding sections) but in this section, we mainly focus on other ideas, not yet mentioned, which also include some projects which were initiated but did not come to their end or met hurdles. Let us also emphasize that deep MF will constitute a core topic of the ERC consolidator grant *eLinoR* of Prof. Nicolas Gillis, see [57] for more details.

8.2.1 Tri minVol NMF

Tri minVolNMF would be an extension of tri-NMF where the volume of one factor is also minimized. Let us first state the standard tri-NMF model. Given $X \in \mathbb{R}_+^{m\times n}$, tri-NMF aims at finding the matrices $W \in \mathbb{R}_+^{m\times r_1}$, $S \in \mathbb{R}_+^{r_1\times r_2}$ and $H \in \mathbb{R}_+^{r_2\times n}$ such that

$$X \approx WSH.$$

Tri-NMF is particularly appropriate for several applications such as recommender systems, where the columns of the matrix W represent r_1 communities of users, the rows of the matrix H represent r_2 communities of items and the inner matrix S indicates how these two sets of communities interact with each other.

Tri-NMF can be seen as a two-layer factorization but without additional constraints on the factors, the solution is highly non-unique. A possible way to alleviate this problem is to minimize the volume of the inner matrix S, that is, solve:

$$\min_{W,H,S \geqslant 0} \det(S^T S) \quad \text{s.t.} \quad \|X - WSH\| \leq \epsilon \tag{8.1}$$

for any ϵ sufficiently small.

The model described by Eq. (8.1) suffers from scaling ambiguities, as for minVol NMF (see Section 2.3.2). Since there is one more factor than in NMF, two normalizations should be applied. This could either be column-stochasticity of both W and H or "double-stochasticity" of S, that is, both $Se = e$ and $S^T e = e$ with e the vector of all ones of appropriate dimension. This latter set of "double-stochastic" matrices is referred to as the Birkhoff polytope.

Preliminary tests with both normalizations were not conclusive, and the interpretation of such a model remains quite elusive, even for synthetic data. However, we believe this would still be an interesting direction of future research, which would possibly build (theoretical) bridges between single-layer NMF and multilayer/deep MF.

8.2.2 Seizure detection with NMF

Automatic seizure detection is an important challenge in biomedical machine learning, see for example [118]. During summer 2020, we welcomed a Bachelor student to work on this topic from the angle of NMF. More precisely, the input data consist in a set of EEG recordings containing several channels corresponding to electrodes spread on the surface of the head. To start with, we only focused on one channel (located in the occipital area of the brain) and computed the corresponding spectrogram through a short-time Fourier transform. Applying NMF on this spectrogram with $r = 2$ allowed to extract the frequential spectrum of the background on the one hand and of the seizure on the other hand. For example, working on the absence seizure, a particular class of seizures encountered among children, we obtained the frequential spectrum represented on Fig. 8.1 which exhibits a peak at 3Hz, as expected for such seizures. However, a lot of possibilities of improvement exist:

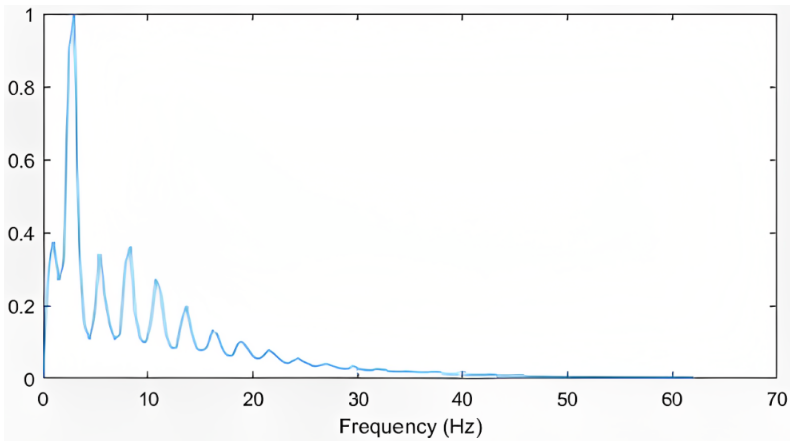


Figure 8.1: Illustration of the frequency pattern of an absence seizure obtained with NMF.

- The data set used for the experiments, namely the TUH EEG seizure detection dataset [93] still undergoes major improvements in terms of artefact removal and, more broadly, denoising, which requires to apply the models on more recent versions of the dataset.
- We only focused on one channel at a time to extract the frequential patterns of seizures. However, since most seizures impact several areas of the brain, it would be interesting to combine multi-channel information in the model, either through some post-processing after the factorization stage or through a tensorial approach.
- A basic NMF model was applied, without specific constraints. Incorporating sparsity in the columns of either W (the frequential spectra) or H (the activations of the frequential patterns over time) would be valuable. Moreover, applying convolutional NMF [13] could also be promising. In a nutshell, it is an extension of NMF which allows to extract patterns that repeat (over time) within the signal.

8.2.3 Sparse archetypal analysis

A potential drawback of (near-convex) archetypal analysis (see Chapter 3) is the choice of the support points, that is the points from which the basis vectors are built. Instead of using the whole data matrix X as the support or choose a matrix Y (see Eq. (3.2)) through some heuristics, one could introduce more flexibility by asking for the matrix A in Eq. (3.1) to be sparse.

The considered problem would therefore be, for example,

$$\min_{A,H\geq 0} \frac{1}{2}||X - YAH||_F^2 \qquad \text{such that A is sparse} \tag{8.2}$$

for some given matrices X and Y (Y can be equal to X as in AA). One possibility is to add a penalty term on the ℓ_1-norm of A to the objective function, that is, $\lambda\|A\|_1$. Due to its structure, that is, the fact that A is the "inner" matrix (between Y and H), this problem may require to be tackled differently than standard sparse NMF. Developing efficient algorithms to solve Problem (8.2) could be valuable for both sparse AA and by extension, sparse deep MF.

BIBLIOGRAPHY

[1] Jong-Hoon Ahn, Seungjin Choi, and Jong-Hoon Oh. A multiplicative up-propagation algorithm. In *Proceedings of the Twenty-first International Conference on Machine Learning*, page 3, 2004.

[2] ArulMurugan Ambikapathi, Tsung-Han Chan, Wing-Kin Ma, and Chong-Yung Chi. A robust minimum volume enclosing simplex algorithm for hyperspectral unmixing. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1202–1205. IEEE, 2010.

[3] Andersen Man Shun Ang and Nicolas Gillis. Algorithms and comparisons of nonnegative matrix factorizations with volume regularization for hyperspectral unmixing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(12):4843–4853, 2019.

[4] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. *arXiv preprint arXiv:1810.02281*, 2018.

[5] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning*, pages 244–253. PMLR, 2018.

[6] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32, 2019.

[7] Sanjeev Arora, Rong Ge, and Ankur Moitra. Learning topic models–going beyond SVD. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 1–10. IEEE, 2012.

[8] American Psychiatric Association. *Diagnostic and Statistical Manual of Mental Disorders: DSM-IV-TR (4th ed., text rev.)*, volume 4. American Psychiatric Association Washington, DC, 2000.

[9] American Psychiatric Association. *Diagnostic and Statistical Manual of Mental Disorders: DSM-5*, volume 5. American Psychiatric Association Washington, DC, 2013.

[10] Peter Bartlett, Dave Helmbold, and Philip Long. Gradient descent with identity initialization efficiently learns positive definite linear transformations by deep residual networks. In *International Conference on Machine Learning*, pages 521–530. PMLR, 2018.

[11] Christian Bauckhage. A note on archetypal analysis and the approximation of convex hulls. *arXiv preprint arXiv:1410.0642*, 2014.

[12] Christian Bauckhage and Christian Thurau. Making archetypal analysis practical. In *Pattern Recognition*, pages 272–281, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[13] Sven Behnke. Discovering hierarchical speech features using convolutional non-negative matrix factorization. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 4, pages 2758–2763. IEEE, 2003.

[14] José M Bioucas-Dias. A variable splitting augmented lagrangian approach to linear spectral unmixing. In *2009 First Workshop on Hyperspectral Image and Signal Processing: Evolution in remote sensing*, pages 1–4. IEEE, 2009.

[15] José M Bioucas-Dias, Antonio Plaza, Nicolas Dobigeon, Mario Parente, Qian Du, Paul Gader, and Jocelyn 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.

[16] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

[17] Athman Bouguettaya, Qi Yu, Xumin Liu, Xiangmin Zhou, and Andy Song. Efficient agglomerative hierarchical clustering. *Expert Systems with Applications*, 42(5):2785–2797, 2015.

[18] Christos Boutsidis and Efstratios Gallopoulos. SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350–1362, 2008.

[19] Tadeusz Caliński and Jerzy Harabasz. A dendrite method for cluster analysis. *Communications in Statistics - Theory and Methods*, 3(1):1–27, 1974.

[20] Jacopo Cavazza, Pietro Morerio, Benjamin Haeffele, Connor Lane, Vittorio Murino, and Rene Vidal. Dropout as a low-rank regularizer for matrix factorization. In *International Conference on Artificial Intelligence and Statistics*, pages 435–444. PMLR, 2018.

[21] Tsung-Han Chan, Chong-Yung Chi, Yu-Min Huang, and Wing-Kin Ma. A convex analysis-based minimum-volume enclosing simplex algorithm for hyperspectral unmixing. *IEEE Transactions on Signal Processing*, 57(11):4418–4432, 2009.

[22] Wen-Sheng Chen, Qianwen Zeng, and Binbin Pan. A survey of deep nonnegative matrix factorization. *Neurocomputing*, 491:305–320, 2022.

[23] Yuansi Chen, Julien Mairal, and Zaid Harchaoui. Fast and robust archetypal analysis for representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1478–1485, 2014.

[24] Seungjin Choi. Algorithms for orthogonal nonnegative matrix factorization. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1828–1832. IEEE, 2008.

[25] Alexander P Christensen, Luis Eduardo Garrido, Kiero Guerra-Peña, and Hudson Golino. Comparing community detection algorithms in psychometric networks: A monte carlo simulation. *Behavior Research Methods*, pages 1–21, 2023.

[26] Andrzej Cichocki and Rafał Zdunek. Multilayer nonnegative matrix factorisation. *Electronics Letters*, 42:947–948, 2006.

[27] Andrzej Cichocki and Rafal Zdunek. Multilayer nonnegative matrix factorization using projected gradient approaches. *International Journal of Neural Systems*, 17(06):431–446, 2007.

[28] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[29] Jeremy E Cohen and Nicolas Gillis. Nonnegative low-rank sparse component analysis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8226–8230. IEEE, 2019.

[30] Kathryn M Connor and Jonathan RT Davidson. Development of a new resilience scale: The Connor-Davidson resilience scale (CD-RISC). *Depression and Anxiety*, 18(2):76–82, 2003.

[31] Beilei Cui, Hong Yu, Tiantian Zhang, and Siwen Li. Self-weighted multi-view clustering with deep matrix factorization. In *Asian Conference on Machine Learning*, pages 567–582. PMLR, 2019.

[32] Adele Cutler and Leo Breiman. Archetypal analysis. *Technometrics*, 36(4):338–347, 1994.

[33] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[34] Steven Davis and Paul Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980.

[35] Pierre De Handschutter and Nicolas Gillis. Deep orthogonal matrix factorization as a hierarchical clustering technique. In *2021 29th European Signal Processing Conference (EUSIPCO)*, pages 1466–1470. IEEE, 2021.

[36] Pierre De Handschutter and Nicolas Gillis. A consistent and flexible framework for deep matrix factorizations. *Pattern Recognition*, 134:109102, 2023.

[37] Pierre De Handschutter, Nicolas Gillis, and Wivine Blekic. Deep symmetric matrix factorization. 2023. Online available at: `https://www.researchgate.net/publication/368693970_Deep_Symmetric_Matrix_Factorization`.

[38] Pierre De Handschutter, Nicolas Gillis, and Xavier Siebert. A survey on deep matrix factorizations. *Computer Science Review*, 42:100423, 2021.

[39] Pierre De Handschutter, Nicolas Gillis, Arnaud Vandaele, and Xavier Siebert. Near-convex archetypal analysis. *IEEE Signal Processing Letters*, 27:81–85, 2019.

[40] Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 126–135, 2006.

[41] Chris H.Q. Ding, Tao Li, and Michael I. Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):45–55, 2008.

[42] Simon Du and Wei Hu. Width provably matters in optimization for deep linear neural networks. In *International Conference on Machine Learning*, pages 1655–1664. PMLR, 2019.

[43] Julian Eggert and Edgar Korner. Sparse coding and NMF. In *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, volume 4, pages 2529–2533. IEEE, 2004.

[44] Sacha Epskamp and Eiko I Fried. A tutorial on regularized partial correlation networks. *Psychological Methods*, 23(4):617, 2018.

[45] Jicong Fan and Jieyu Cheng. Matrix completion by deep matrix factorization. *Neural Networks*, 98:34–41, 2018.

[46] Xin-Ru Feng, Heng-Chao Li, Jun Li, Qian Du, Antonio Plaza, and William J Emery. Hyperspectral unmixing using sparsity-constrained deep nonnegative matrix factorization with total variation. *IEEE Transactions on Geoscience and Remote Sensing*, 56(10):6245–6257, 2018.

[47] Xin-Ru Feng, Heng-Chao Li, Rui Wang, Qian Du, Xiuping Jia, and Antonio J. Plaza. Hyperspectral unmixing based on nonnegative matrix factorization: A comprehensive review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2022.

[48] Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu. Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis. *Neural Computation*, 21(3):793–830, 2009.

[49] Cédric Févotte and Jérôme Idier. Algorithms for nonnegative matrix factorization with the β-divergence. *Neural Computation*, 23(9):2421–2456, 2011.

[50] Edna B Foa, David S Riggs, Constance V Dancu, and Barbara O Rothbaum. Reliability and validity of a brief instrument for assessing post-traumatic stress disorder. *Journal of Traumatic Stress*, 6(4):459–473, 1993.

[51] Xiao Fu, Kejun Huang, and Nicholas D. Sidiropoulos. On identifiability of nonnegative matrix factorization. *IEEE Signal Processing Letters*, 25(3):328–332, 2018.

[52] Xiao Fu, Kejun Huang, Nicholas D. Sidiropoulos, and Wing-Kin Ma. Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications. *IEEE Signal Processing Magazine*, 36(2):59–80, 2019.

[53] Xiao Fu, Kejun Huang, Bo Yang, Wing-Kin Ma, and Nicholas D. Sidiropoulos. Robust volume minimization-based matrix factorization for remote sensing and document clustering. *IEEE Transactions on Signal Processing*, 64(23):6254–6268, 2016.

[54] Nicolas Gillis. Successive nonnegative projection algorithm for robust nonnegative blind source separation. *SIAM Journal on Imaging Sciences*, 7(2):1420–1450, 2014.

[55] Nicolas Gillis. The why and how of nonnegative matrix factorization. In *Regularization, Optimization, Kernels, and Support Vector Machines*, Machine Learning and Pattern Recognition, chapter 12, pages 257–291. Chapman & Hall/CRC, 2014.

[56] Nicolas Gillis. *Nonnegative Matrix Factorization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2020.

[57] Nicolas Gillis. Description of eLinoR project. `https://sites.google.com/site/nicolasgillis/projects/erc-cons`, 2023. [Online; accessed 13-Apr-2023].

[58] Nicolas Gillis, Da Kuang, and Haesun Park. Hierarchical clustering of hyperspectral images using rank-two nonnegative matrix factorization. *IEEE Transactions on Geoscience and Remote Sensing*, 53(4):2066–2078, 2014.

[59] Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. Implicit regularization in matrix factorization. *Advances in Neural Information Processing Systems*, 30, 2017.

[60] Zhenxing Guo and Shihua Zhang. Sparse deep nonnegative matrix factorization. *Big Data Mining and Analytics*, 3(1):13–28, 2019.

[61] Zhicheng He, Jie Liu, Caihua Liu, Yuan Wang, Airu Yin, and Yalou Huang. Dropout non-negative matrix factorization. *Knowledge and Information Systems*, 60:781–806, 2019.

[62] David Hevey. Network analysis: a brief overview and tutorial. *Health Psychology and Behavioral Medicine*, 6(1):301–328, 2018.

[63] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *The Journal of Machine Learning Research*, 5(9), 2004.

[64] Kejun Huang, Xiao Fu, and Nikolaos D Sidiropoulos. Anchor-free correlated topic modeling: Identifiability and algorithm. *Advances in Neural Information Processing Systems*, 29, 2016.

[65] 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, 2016.

[66] Hamid Javadi and Andrea Montanari. Nonnegative matrix factorization via archetypal analysis. *Journal of the American Statistical Association*, 115(530):896–907, 2020.

[67] Tae Gyoon Kang, Kisoo Kwon, Jong Won Shin, and Nam Soo Kim. NMF-based target source separation using deep neural network. *IEEE Signal Processing Letters*, 22(2):229–233, 2014.

[68] Jingu Kim and Haesun Park. Sparse nonnegative matrix factorization for clustering. Technical report, Georgia Institute of Technology, 2008.

[69] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[70] Da Kuang, Chris Ding, and Haesun Park. Symmetric nonnegative matrix factorization for graph clustering. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 106–117. SIAM, 2012.

[71] Da Kuang and Haesun Park. Fast rank-2 nonnegative matrix factorization for hierarchical document clustering. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 739–747, 2013.

[72] Thomas Laurent and James Brecht. Deep linear networks with arbitrary loss: All local minima are global. In *International Conference on Machine Learning*, pages 2902–2907. PMLR, 2018.

[73] Hien Le Thi Khanh, Duy Nhat Phan, and Nicolas Gillis. An inertial block majorization minimization framework for nonsmooth nonconvex optimization. *The Journal of Machine Learning Research*, 24:1–41, 2023.

[74] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[75] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[76] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13, 2000.

[77] Valentin Leplat, Andersen MS Ang, and Nicolas Gillis. Minimum-volume rank-deficient nonnegative matrix factorizations. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3402–3406. IEEE, 2019.

[78] Valentin Leplat, Nicolas Gillis, and Andersen MS Ang. Blind audio source separation with minimum-volume beta-divergence NMF. *IEEE Transactions on Signal Processing*, 68:3400–3410, 2020.

[79] Bo Li, Guoxu Zhou, and Andrzej Cichocki. Two efficient algorithms for approximately orthogonal nonnegative matrix factorization. *IEEE Signal Processing Letters*, 22(7):843–846, 2014.

[80] Jun Li and José M Bioucas-Dias. Minimum volume simplex analysis: A fast algorithm to unmix hyperspectral data. In *IGARSS 2008-2008 IEEE International Geoscience and Remote Sensing Symposium*, volume 3, pages III–250. IEEE, 2008.

[81] Xiao Li, Zhihui Zhu, Qiuwei Li, and Kai Liu. A provable splitting approach for symmetric nonnegative matrix factorization. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[82] Chia-Hsiang Lin, Wing-Kin Ma, Wei-Chiang Li, Chong-Yung Chi, and ArulMurugan Ambikapathi. Identifiability of the simplex volume minimization criterion for blind hyperspectral unmixing: The no-pure-pixel case. *IEEE Transactions on Geoscience and Remote Sensing*, 53(10):5530–5546, 2015.

[83] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.

[84] Xin Luo, Mengchu Zhou, Yunni Xia, and Qingsheng Zhu. An efficient non-negative matrix-factorization-based approach to collaborative

filtering for recommender systems. *IEEE Transactions on Industrial Informatics*, 10(2):1273–1284, 2014.

[85] Bensheng Lyu, Kan Xie, and Weijun Sun. A deep orthogonal non-negative matrix factorization method for learning attribute representations. In *Neural Information Processing: 24th International Conference, ICONIP 2017, Guangzhou, China, November 14–18, 2017, Proceedings, Part VI 24*, pages 443–452. Springer, 2017.

[86] François Malgouyres and Joseph Landsberg. On the identifiability and stable recovery of deep/multi-layer structured matrix factorization. In *2016 IEEE Information Theory Workshop (ITW)*, pages 315–319. IEEE, 2016.

[87] Aanchal Mongia, Neha Jhamb, Emilie Chouzenoux, and Angshul Majumdar. Deep latent factor model for collaborative filtering. *Signal Processing*, 169:107366, 2020.

[88] Morten Mørup and Lars Kai Hansen. Archetypal analysis for machine learning and data mining. *Neurocomputing*, 80:54–63, 2012.

[89] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[90] Yurii Nesterov. A method of solving a convex programming problem with convergence rate O (1/kˆ 2). In *Doklady Akademii Nauk*, volume 269, pages 543–547. Russian Academy of Sciences, 1983.

[91] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2003.

[92] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[93] Iyad Obeid and Joseph Picone. The Temple University Hospital EEG Data Corpus. *Frontiers in Neuroscience*, 10:196, 2016.

[94] Riyasat Ohib, Nicolas Gillis, Niccolò Dalmasso, Sameena Shah, Vamsi K Potluru, and Sergey Plis. Explicit group sparse projection with applications to deep learning and NMF. *Transactions on Machine Learning Research*, 2019.

[95] National Institute on Drug Abuse. National Drug Abuse Treatment Clinical Trials Network (CTN-0015): Women's Treatment for Trauma and Substance Use Disorders. `https://datashare.nida.nih.gov/study/nida-ctn-0015`, 2014. Accessed on May 8, 2023.

[96] Savas Ozkan, Berk Kaya, and Gozde Bozdagi Akar. Endnet: Sparse autoencoder network for endmember extraction and hyperspectral unmixing. *IEEE Transactions on Geoscience and Remote Sensing*, 57(1):482–496, 2018.

[97] Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[98] Anushka Pai, Alina M Suris, and Carol S North. Posttraumatic stress disorder in the DSM-5: Controversy, change, and conceptual considerations. *Behavioral Sciences*, 7(1):7, 2017.

[99] Filippo Pompili, Nicolas Gillis, P-A Absil, and François Glineur. Two algorithms for orthogonal nonnegative matrix factorization with application to clustering. *Neurocomputing*, 141:15–25, 2014.

[100] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS 2005: 20th International Symposium, Istanbul, Turkey, October 26-28, 2005. Proceedings 20*, pages 284–293. Springer, 2005.

[101] Yuning Qiu, Guoxu Zhou, and Kan Xie. Deep approximately orthogonal nonnegative matrix factorization for clustering. *arXiv preprint arXiv:1711.07437*, 2017.

[102] Roozbeh Rajabi and Hassan Ghassemian. Spectral unmixing of hyperspectral imagery using multilayer NMF. *IEEE Geoscience and Remote Sensing Letters*, 12(1):38–42, 2014.

[103] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1):016110, 2006.

[104] Donald J Robinaugh, Ria HA Hoekstra, Emma R Toner, and Denny Borsboom. The network approach to psychopathology: a review of

the literature 2008–2018 and an agenda for future research. *Psychological medicine*, 50(3):353–366, 2020.

[105] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.

[106] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 6655–6659. IEEE, 2013.

[107] Iqbal H. Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2(3):160, 2021.

[108] Pulkit Sharma, Vinayak Abrol, and Anil Kumar Sao. Deep-sparse-representation-based features for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(11):2162–2175, 2017.

[109] Pulkit Sharma, Vinayak Abrol, and Anshul Thakur. ASe: Acoustic Scene Embedding using deep archetypal analysis and GMM. In *Interspeech*, pages 3299–3303, 2018.

[110] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[111] Ruoyu Sun, Dawei Li, Shiyu Liang, Tian Ding, and Rayadurgam Srikant. The global landscape of neural networks: An overview. *IEEE Signal Processing Magazine*, 37(5):95–108, 2020.

[112] Anshul Thakur, Vinayak Abrol, Pulkit Sharma, and Padmanabhan Rajan. Deep convex representations: Feature representations for bioacoustics classification. In *Interspeech*, pages 2127–2131, 2018.

[113] Lei Tong, Jing Yu, Chuangbai Xiao, and Bin Qian. Hyperspectral unmixing via deep matrix factorization. *International*

Journal of Wavelets, Multiresolution and Information Processing, 15(06):1750058, 2017.

[114] George Trigeorgis, Konstantinos Bousmalis, Stefanos Zafeiriou, and Björn W Schuller. A deep semi-NMF model for learning hidden representations. In *International Conference on Machine Learning*, pages 1692–1700. PMLR, 2014.

[115] George Trigeorgis, Konstantinos Bousmalis, Stefanos Zafeiriou, and Björn W Schuller. A deep matrix factorization method for learning attribute representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(3):417–429, 2016.

[116] Madeleine Udell, Corinne Horn, Reza Zadeh, Stephen Boyd, et al. Generalized low rank models. *Foundations and Trends® in Machine Learning*, 9(1):1–118, 2016.

[117] David van Dijk, Daniel B Burkhardt, Matthew Amodio, Alexander Tong, Guy Wolf, and Smita Krishnaswamy. Finding archetypal spaces using neural networks. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2634–2643. IEEE, 2019.

[118] Paul Vanabelle, Pierre De Handschutter, Riëm El Tahry, Mohammed Benjelloun, and Mohamed Boukhebouze. Epileptic seizure detection using EEG signals and extreme gradient boosting. *Journal of Biomedical Research*, 34(3):228, 2020.

[119] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, 2007.

[120] Fei Wang, Tao Li, Xin Wang, Shenghuo Zhu, and Chris Ding. Community discovery using nonnegative matrix factorization. *Data Mining and Knowledge Discovery*, 22:493–521, 2011.

[121] Hua Wang, Heng Huang, and Chris Ding. Simultaneous clustering of multi-type relational data via symmetric nonnegative matrix tri-factorization. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 279–284, 2011.

[122] JianYu Wang and Xiao-Lei Zhang. Deep NMF topic modeling. *Neurocomputing*, 515:157–173, 2023.

[123] Qi Wang, Mengying Sun, Liang Zhan, Paul Thompson, Shuiwang Ji, and Jiayu Zhou. Multi-modality disease modeling via collective deep matrix factorization. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1155–1164, 2017.

[124] Cai Xu, Ziyu Guan, Wei Zhao, Yunfei Niu, Quan Wang, and Zhiheng Wang. Deep multi-view concept learning. In *International Joint Conference on Artificial Intelligence*, pages 2898–2904. Stockholm, 2018.

[125] Jaewon Yang and Jure Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, pages 587–596, 2013.

[126] Yan Yang and Hao Wang. Multi-view clustering: A survey. *Big Data Mining and Analytics*, 1(2):83–107, 2018.

[127] Fanghua Ye, Chuan Chen, and Zibin Zheng. Deep autoencoder-like nonnegative matrix factorization for community detection. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1393–1402, 2018.

[128] Baolin Yi, Xiaoxuan Shen, Hai Liu, Zhaoli Zhang, Wei Zhang, Sannyuya Liu, and Naixue Xiong. Deep matrix factorization with implicit feedback embedding for recommendation system. *IEEE Transactions on Industrial Informatics*, 15(8):4591–4601, 2019.

[129] Jinshi Yu, Guoxu Zhou, Andrzej Cichocki, and Shengli Xie. Learning the hierarchical parts of objects by deep non-smooth nonnegative matrix factorization. *IEEE Access*, 6:58096–58105, 2018.

[130] Yu Zhang, Ekapol Chuangsuwanich, and James Glass. Extracting deep neural network bottleneck features using low-rank matrix factorization. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 185–189. IEEE, 2014.

[131] Yu Zhang and Dit-Yan Yeung. Overlapping community detection via bounded nonnegative matrix tri-factorization. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 606–614, 2012.

[132] Handong Zhao, Zhengming Ding, and Yun Fu. Multi-view clustering via deep matrix factorization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[133] Léon Zheng, Elisa Riccietti, and Rémi Gribonval. Hierarchical identifiability in multi-layer sparse matrix factorization. *arXiv preprint arXiv:2110.01230*, 2021.

[134] Yijia Zhou and Lijun Xu. A deep structure-enforced nonnegative matrix factorization for data representation. In *Pattern Recognition and Computer Vision: First Chinese Conference, PRCV 2018, Guangzhou, China, November 23-26, 2018, Proceedings, Part III*, pages 340–350. Springer, 2018.

[135] Feiyun Zhu. Spectral unmixing datasets with ground truths. *arXiv preprint arXiv:1708.05125*, 2017.