Université catholique de Louvain
École polytechnique de Louvain
Département d'ingénierie mathématique

Nonnegative Matrix Factorization
Complexity, Algorithms and Applications

Nicolas Gillis

Thesis submitted in partial fulfillment
of the requirements for the degree of
Docteur en Sciences de l'Ingénieur

Dissertation committee:
Prof. Philippe Lefèvre (UCL, President)
Prof. François Glineur (UCL, Advisor)
Prof. Pierre-Antoine Absil (UCL)
Prof. Paul Van Dooren (UCL)
Prof. Samuel Fiorini (Université libre de Bruxelles, Belgium)
Prof. Didier Henrion (LAAS-CNRS, University of Toulouse, France)
Prof. Sabine Van Huffel (Katholieke Universiteit Leuven, Belgium)
Prof. Stephen Vavasis (University of Waterloo, Canada)

February 2011

Acknowledgments

I would like to thank my advisor, François Glineur. I have enjoyed working with him since my master's thesis, and he has constantly supported and encouraged me for my different projects.

I also thank Robert Plemmons who warmly greeted me at Wake Forest University during the Fall semester 2009. It was a pleasure working with him, Paùl Pauca and their colleagues.

I am grateful to my support committee, Didier Henrion and Paul Van Dooren, with whom I had the chance to discuss my problems, and benefit from their experience. I also thank the members of my defense committee Pierre-Antoine Absil, Samuel Fiorini, Sabine Van Huffel, Stephen Vavasis, and the chair Philippe Lefèvre.

I am also grateful to the other people who have helped me in various ways, and have interacted with this work: Michael Berry, Mathieu Van Vyve, Raphaël Jungers, Chia-Tche Chang, Quentin Rentmeesters, Santanu Dey, Laurence Wolsey, and others.

This experience could not have been what it has without the warm ambiance at CORE. I thank my office mates, Gauthier, Joachim, Salome, Grégory, Stéphane, Laurent, Alexis, Vladislav and Julie ; my colleagues Robert, Olivier, Rafael, Sylvette, Joël, Peter, Maia, Margherita, Olivier', Jean-Sébastien, Claudia, Mikel and many others. I also thank the precious help of the CORE administratives.

As a F.R.S.-FNRS research fellow, I thank the Fonds de la recherche scientifique (F.R.S.-FNRS) for their administrative and financial support.

Last but not least, I thank my family and my friends. I dedicate this thesis to my grandparents.

Abstract

Linear dimensionality reduction techniques such as principal component analysis are powerful tools for the analysis of high-dimensional data. In this thesis, we explore a closely related problem, namely nonnegative matrix factorization (NMF), a low-rank matrix approximation problem with nonnegativity constraints. More precisely, we seek to approximate a given nonnegative matrix with the product of two low-rank nonnegative matrices. These nonnegative factors can be interpreted in the same way as the data, e.g., as images (described by pixel intensities) or texts (represented by vectors of word counts), and lead to an additive and sparse representation. However, they render the problem much more difficult to solve (i.e., NP-hard).

A first goal of this thesis is to study theoretical issues related to NMF. In particular, we make connections with well-known problems in graph theory, combinatorial optimization and computational geometry. We also study computational complexity issues and show, using reductions from the maximum-edge biclique problem, NP-hardness of several low-rank matrix approximation problems, including the rank-one subproblems arising in NMF, a problem involving underapproximation constraints (NMU) and the unconstrained version of the factorization problem where some data is missing or unknown.

Our second goal is to improve existing techniques and develop new tools for the analysis of nonnegative data. We propose accelerated variants of several NMF algorithms based on a careful analysis of their computational cost. We also introduce a multilevel approach to speed up their initial convergence. Finally, a new greedy heuristic based on NMU is presented and used for the analysis of hyperspectral images, in which each pixel is measured along hundreds of wavelengths, which allows for example spectroscopy of satellite images.

Table of contents

Notation

Scalars, Vectors, Matrices

$\mathbb{R}$	set of real numbers
$\mathbb{R}_0$	set of nonzero real numbers
$\mathbb{R}_+$	set of nonnegative real numbers
$\mathbb{R}^n$	set of real column vectors of dimension n
$\mathbb{R}^{m\times n}$	set of real matrices of dimension m-by-n
$\mathbb{R}^n_+$	set of nonnegative real column vectors of dimension n
$\mathbb{R}^{m\times n}_+$	set of nonnegative real matrices of dimension m-by-n

Norms

$\|.\|_1$	ℓ_1-norm, $\|x\|_1 = \sum_{i=1}^n \|x_i\|$, $x \in \mathbb{R}^n$
$\|.\|_2$	vector ℓ_2-norm, $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$, $x \in \mathbb{R}^n$
	matrix ℓ_2-norm, $\|A\|_2 = \max_{x\in\mathbb{R}^n, \|x\|_2=1} \|Ax\|_2$, $A \in \mathbb{R}^{m\times n}$
$\|.\|_\infty$	vector ℓ_∞-norm, $\|x\|_\infty = \max_{1\le i\le n} \|x_i\|$, $x \in \mathbb{R}^n$
$\|.\|_0$	ℓ_0-'norm', $\|x\|_0 = \big\|\{i \mid x_i \neq 0\}\big\|$, $x \in \mathbb{R}^m$
$\|.\|_F$	Frobenius norm, $\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n A_{ij}^2}$, $A \in \mathbb{R}^{m\times n}$

Functions on Matrices

$\langle ., . \rangle$	scalar product, $\langle A, B \rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} A_{ij} B_{ij}, A, B \in \mathbb{R}^{m \times n}$
$\sigma_i(.)$	i^{th} singular values of a matrix, in non-decreasing order
rank(.)	rank of a matrix
$\text{rank}_+(.)$	nonnegative rank of a nonnegative matrix
$\text{rank}_+^*(.)$	restricted nonnegative rank of a nonnegative matrix
conv(.)	convex hull of the columns of a matrix
col(.)	column space of a matrix
$A_{i:}$ or $A(i, :)$	i^{th} row of A
$A_{:j}$ or $A(:, j)$	j^{th} column of A
A_{ij} or $A(i, j)$	entry at position (i, j) of A
$A(I, J)$	submatrix of A with row (resp. column) indices in I (resp. J)
$\circ$	component-wise multiplication, $(A \circ B)_{ij} = A_{ij} B_{ij}$
$\frac{[.]}{[.]}$	component-wise division, $\left(\frac{[A]}{[B]}\right)_{ij} = \frac{A_{ij}}{B_{ij}}$
$(.)^T$	transpose of a matrix, $(A^T)_{ij} = A_{ji}$

Miscellaneous

$\mathbf{1_m}$	vector of all ones of dimension m
$\mathbf{1_{m \times n}}$	matrix of all ones of dimension m-by-n
$\mathbf{0_{m \times n}}$	matrix of zeros of dimension m-by-n
I_n	identity matrix of dimension n
$\doteq$	is equal by definition
$a{:}b$	set $\{a, a+1, \ldots, b-1, b\}$ (for a and b integers with $a \leq b$)
∇f	It is the the gradient of the function f
$\nabla^2 f$	It is the hessian of the function f
$\lceil . \rceil$	$\lceil x \rceil$ is the smallest integer greater or equal to $x \in \mathbb{R}$
$\lfloor . \rfloor$	$\lfloor x \rfloor$ is the largest integer smaller or equal to $x \in \mathbb{R}$
$\setminus$	substraction of two sets, $R \setminus S$ is the set of elements in R not in S
$\lvert . \rvert$	cardinality of a set, $\lvert S \rvert$ is the number of elements in S
$\bar{.}$	complement of a set, for $S \subset R$, $\bar{S} = R \setminus S$
supp(.)	support (set of non-zero entries), $\text{supp}(x) = \{1 \leq i \leq n \mid x_i \neq 0\}, x \in \mathbb{R}^n$
$\overline{\text{supp}}(.)$	sparsity pattern (set of zero entries, complement of the support)

Acronyms

PCA	principal component analysis (p. 1)
SPCA	sparse principal component analysis (p. 1)
WLRA	weighted low-rank approximation (p. 1 and p. 200)
ICA	independent component analysis (p. 2)
NMF	nonnegative matrix factorization (p. 3)
LP	linear programming (p. 10)
QP	quadratic programming (p. 10)
SOCP	second order cone programming (p. 10)
SDP	semidefinite positive (p. 10)
GP	geometric programming (p. 10 and p. 143)
SVD	singular value decomposition (p. 13)
LRA	low-rank matrix approximation (p. 12)
LRA-1	rank-one matrix approximation (p. 13)
PSV	principal singular vectors (p. 14)
PC	principal component (p. 14)
Exact NMF	exact nonnegative matrix factorization (p. 22)
IS	intermediate simplex (p. 22)
RNR	restricted nonnegative rank (p. 26)
NPP	nested polytopes problem (p. 27)
EDM	Euclidean distance matrix (p. 48)
NNLS	nonnegative least squares (p. 64)
MU	multiplicative updates (p. 65)
ANLS	alternating nonnegative least squares (p. 68)
HALS	hierarchical alternating least squares (p. 70)
R1NF	rank-one nonnegative factorization (p. 110)
NF	nonnegative factorization (p. 127)
NMU	nonnegative matrix underapproximation (p. 135)
NMU-1	rank-one nonnegative matrix underapproximation (p. 139)
PCAMD	PCA with missing data (p. 199)
SFM	structure from motion (p. 203)
LRAMD	low-rank matrix approximation with missing data (p. 208)

Chapter 1

Introduction

Approximating a matrix with one of lower rank is a key problem in data analysis, and is widely used for linear dimensionality reduction. Typically, we are given a matrix $M \in \mathbb{R}^{m \times n}$, where each of the n columns represents an element of a dataset in an m-dimensional space. The aim is to approximate M with a low-rank matrix, which can be expressed as the product of two matrices $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{r \times n}$ where r is the rank of the approximation:

$$M \approx UV \quad \iff \quad M \approx \sum_{k=1}^{r} U_{:k} V_{k:} \quad \iff \quad M_{:j} \approx \sum_{k=1}^{r} U_{:k} V_{kj} \quad 1 \leq j \leq n,$$

i.e., M is approximated with the sum of r rank-one factors $U_{:k}V_{k:}$. Matrix factorization allows to represent the n columns of M in a smaller r-dimensional space defined by the columns of U, with coordinates given by the n columns of V, see Figure 1.1. In fact, each input column $M_{:j}$ for $1 \leq j \leq n$ is a linear combination of a set of r basis elements $U_{:k}$ $1 \leq k \leq r$ with corresponding weights V_{kj}. This linear dimensionality reduction technique is widely used in machine learning, and notably enables noise filtering, classification, visualization, and interpretation. There exists numerous variants emphasizing different objective functions measuring the quality of the approximation and different constraints on the factors, e.g.,

- ◇ principal component analysis (PCA) [95], without any constraints and using the Frobenius norm of the approximation error as the objective function;
- ◇ sparse PCA, adding sparsity constraints on PCA [46];
- ◇ weighted low-rank approximation (WLRA), attaching to each entry of the data matrix a different importance through the use of a weighting in

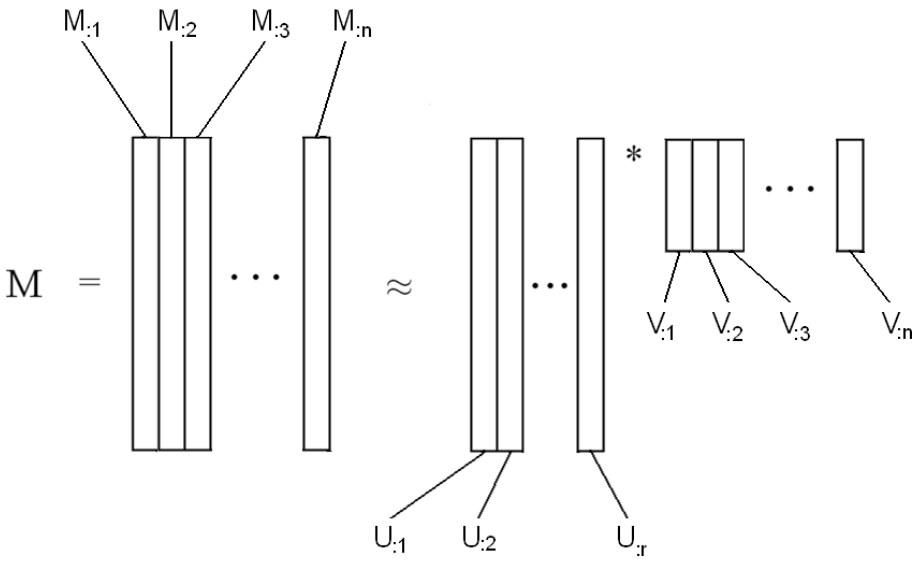

Figure 1.1: Linear dimensionality reduction through matrix factorization.

the objective function. For example, associating a zero weight to missing or unknown data is equivalent to PCA with missing data (see Chapter 8);

- ◇ independent component analysis (ICA) [38], assuming statistical independence of the sources (i.e., the columns of matrix U).

In the context of nonnegative data, i.e., when data matrix M is (componentwise) nonnegative which is denoted $M \geq 0$, one might consider nonnegativity of the input columns to be an important feature, and consequently require the basis elements to be also nonnegative, i.e., impose $U \geq 0$, so that they can be interpreted in the same way (e.g., these columns can correspond to images described by nonnegative pixel intensities or to texts represented by vectors of nonnegative word counts). Furthermore, one can impose nonnegativity of the weights with $V \geq 0$, leading to an essentially additive reconstruction of the input columns by the basis elements. This representation is then *parts-based*: basis elements $U_{:k}$'s will represent common parts of the columns of M. This low-rank approximation technique with additional nonnegativity constraints is referred to as *nonnegative matrix factorization* (NMF)[1] and is often formulated

[1]Strictly speaking NMF is an approximate problem and should preferably be referred to as approximate NMF, or nonnegative matrix approximation [52].

as follows

$$\min_{\substack{U \in \mathbb{R}^{m\times r} \\ V \in \mathbb{R}^{r\times n}}} ||M - UV||_F^2 \quad \text{such that} \quad U \geq 0,\ V \geq 0. \tag{NMF}$$

Because of these additional constraints,

NMF is an additive linear model for nonnegative data.

For example, let each column of M be a vectorized gray-level image of a face using (nonnegative) pixel intensities. The nonnegative matrix factorization of M will generate a matrix U whose columns are nonnegative basis elements of the original images which can then be interpreted as images as well. Moreover, since each original face is reconstructed through a weighted sum of these basis elements, the latter are common parts extracted from the original faces, such as eyes, noses and lips. Figure 1.2 illustrates this property of the NMF decomposition.

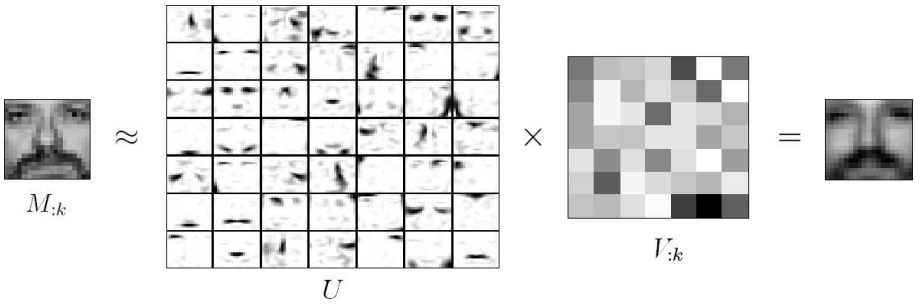

Figure 1.2: NMF applied to the CBCL Face Database #1, MIT Center For Biological and Computation Learning. It consists of the approximation of 2429 gray-level images of faces represented with 19×19 pixels (columns of M) using $r = 49$ basis elements (columns of U).

NMF was introduced in 1994 by Paatero and Tapper [124] and started to be extensively studied after the publication of an article by Lee and Seung [105] in 1999. It has been widely used in machine learning, e.g., for air emission control, music analysis, graph clustering, food quality and safety analysis, microarray data analysis, collaborative filtering; see [10, 50, 52, 89] and the references therein. In this thesis, we will be particularly interested in the following three applications

1. *Image processing.* NMF enables the extraction of the different constitutive parts of a set of images, e.g., the shared features of a set of faces as shown

on Figure 1.2. The decomposition by parts is useful for classification tasks such as face recognition, because it makes the dimensionality reduction more robust to some unfavorable conditions, such as occlusion [84] (e.g., if a person is wearing sunglasses, the other parts of the face are still visible, hence can be well represented with the part-based basis).

2. *Text mining.* NMF can be interpreted as a biclustering model: each rank-one factor of the decomposition will be sparse (because of the nonnegativity constraints, and because M is) and will typically correspond to a dense rectangular submatrix of M (a bicluster), enabling NMF to extract highly connected rows and columns of the matrix M. In text mining applications, NMF extracts closely related sets of texts and words [55].

3. *Hyperspectral data analysis.* Given a hyperspectral image (which is constituted of several images taken at different wavelengths), NMF is able to automatically detect the constitutive materials present in the scene being imaged, and classify the pixels accordingly [126].

An important feature of NMF is that its nonnegativity constraints typically induce *sparse factors*, i.e., factors with relatively many zero entries. Intuitively, decomposition into parts requires the basis elements to be sparse, cf. Figure 1.2. More formally, the reason for this behavior is that stationary points (U, V) of NMF will typically be located at the boundary of the feasible domain $\mathbb{R}_+^{m\times r} \times \mathbb{R}_+^{r\times n}$, hence will feature zero components. This can be explained with the first-order optimality conditions: because the set of stationary points of a problem of the type

$$\min_{x\in\mathbb{R}^n} \quad f(x) \text{ such that } x \geq 0,$$

is given by the following expression

$$S_f = \{x \in \mathbb{R}^n \mid x \geq 0, \nabla f(x) \geq 0 \text{ and } x_i[\nabla f(x)]_i = 0 \; 1 \leq i \leq n\},$$

some components of the solution can be expected to be equal to zero.

Sparsity of the factors is an important consideration in practice: in addition to reducing memory requirements to store the basis elements and their weights, sparsity improves interpretation of the factors, especially when dealing with classification/clustering problems, e.g., in text mining [137] and computational biology [66, 97]. By contrast, unconstrained low-rank approximations such as PCA do not naturally generate sparse factors (for that reason, low-rank approximations techniques with additional sparsity constraints have been recently introduced; this is referred to as sparse PCA or SPCA, see, e.g., [46] and the references therein).

Unfortunately the advantages of NMF (part-based representation and sparsity) over PCA come at a certain price. First, because of the additional nonnegativity constraints, the approximation error of the input data for a given

factorization rank r will always be higher for NMF than in the unconstrained case. Second, optimization problem (NMF) is more difficult to solve than its unconstrained counterpart: while PCA problems can be solved in polynomial time (e.g., using a singular value decomposition technique [80]), NMF problems belong to the class of $\mathcal{NP}$-hard problems, as recently shown by Vavasis [148]. However, it should also be pointed out that these drawbacks (higher error, $\mathcal{NP}$-hardness) are also present for competing techniques emphasizing sparsity, such as sparse PCA.

Thesis outline and related publications

The aim of this thesis is twofold:

(1) study theoretical issues related to NMF and try to improve understanding of some of its aspects; in particular, try to make connections with other problems, e.g., in graph theory and combinatorial optimization,

and

(2) develop new tools for the analysis of nonnegative data, or improve them; in particular, develop improved and new algorithms for NMF, and use them for applications.

The thesis is structured as follows.

Chapter 2. Preliminary

Some theoretical background needed throughout this thesis is briefly presented.

Chapter 3. Nonnegative Rank

The nonnegative rank of a nonnegative matrix is the minimum number of nonnegative rank-one factors needed to reconstruct it exactly. It is directly related to NMF, with the additional requirement that the factorization must be exact. In this chapter, we first review some of its complexity aspects, and link them with NMF. We then introduce and study a slightly different quantity called the restricted nonnegative rank. We prove its computation is equivalent to a problem in polyhedral combinatorics, which allows us to fully characterize the algorithmic complexity of its computation. This in turn sheds new light on the nonnegative rank problem, and in particular on its computational complexity. It also allows us to provide new improved lower bounds based on its geometric interpretation. We illustrate these results on slack matrices (arising from the

study of extended formulations in linear programming) and on linear Euclidean distance matrices.

[74] N. Gillis and F. Glineur, **On the Geometric Interpretation of the Nonnegative Rank**, CORE Discussion paper 2010/51, 2010.

Chapter 4. Algorithms for NMF

In this chapter, three well-known and widely used NMF algorithms are first presented, namely the multiplicative updates (MU), the hierarchical alternating least squares (HALS) and the alternating nonnegative least squares (ANLS). We then propose accelerated versions of MU and HALS, based on the analysis of the computational cost needed at each iteration. This approach can potentially be used for any first-order NMF algorithm, and is applied successfully on a projected gradient method. Our numerical experiments show that HALS and its accelerated version perform remarkably well, which is theoretically supported by an argument based on the properties of NMF and its solutions. We also embed NMF algorithms into the framework of multilevel methods, speeding up significant their convergence in situations where data admits a good approximate representation in a lower dimensional space through linear transformations preserving nonnegativity.

[71] N. Gillis and F. Glineur, **A Multilevel Approach for Nonnegative Matrix Factorization**, CORE Discussion paper 2010/47, 2010.

[72] N. Gillis and F. Glineur, **Accelerated Multiplicative Updates and Hierarchical ALS Algorithms for Nonnegative Matrix Factorization**, preprint, 2010.

Chapter 5. Nonnegative Factorization

Rank-one subproblems arising in NMF consists in approximating a not necessarily nonnegative matrix with a nonnegative rank-one matrix; this is referred to as rank-one nonnegative factorization (R1NF). R1NF is shown to be $\mathcal{NP}$-hard using a reduction from the maximum-edge biclique problem. Stationary points of R1NF are then linked to feasible solutions of the biclique problem, which allows us to design a new type of biclique finding algorithm based on the application of a block-coordinate descent scheme to R1NF. Also, the multiplicative updates for NMF are generalized to nonnegative factorization (NF) of any rank, which provides an explanation of the better performances of HALS over MU.

[69] N. Gillis and F. Glineur, **Nonnegative Factorization and The Maximum Edge Biclique Problem**, CORE Discussion paper 2008/64, 2008.

Chapter 6. Nonnegative Matrix Underapproximation

A novel approach for obtaining sparse solutions to NMF problems is introduced. It is based on a recursive approach and the use of additional underapproximation constraints. Our algorithm is based on the Lagrangian relaxation of the corresponding optimization problem, and its effectiveness to obtain sparse solutions is experimentally demonstrated. We also consider some convex formulations in the rank-one case, and give some theoretical evidences explaining why there might not exist 'good' convex reformulations of NMU and NMF when the factorization rank is larger than one.

[11] M.W. Berry, N. Gillis and F. Glineur, **Document Classification Using Nonnegative Matrix Factorization and Underapproximation**, Proc. of the IEEE Int. Symp. on Circuits and Systems (ISCAS), p. 2782–2785, 2009.

[75] N. Gillis and F. Glineur, **Using Underapproximations for Sparse Nonnegative Matrix Factorization**, Pattern Recognition, 43(4), 1676-1687, 2010.

Chapter 7. Hyperspectral Data Analysis using Underapproximation

The recursive approach described in Chapter 6 is used for the analysis of hyperspectral images. In particular, we focus on rank-one underapproximations which enables us to extract features in a recursive way, like PCA, but preserving nonnegativity. It allows recovering of the constitutive elements in hyperspectral data. Both ℓ_2-norm and ℓ_1-norm based minimization of the energy functional are considered. We experimentally show the efficiency of this new strategy on hyperspectral images associated with space object material identification, and on HYDICE and related remote sensing images.

[76] N. Gillis and R.J. Plemmons, **Dimensionality reduction, classification, and spectral mixture analysis using nonnegative underapproximation**, SPIE conference Volume 7695, paper 46, Orlando, 2010.

[77] N. Gillis and R.J. Plemmons, **Dimensionality reduction, classification, and spectral mixture analysis using nonnegative underapproximation**, Optical Engineering, 2011, in press.

Chapter 8. Weights and Missing Data

In some cases, it might be necessary to attach to each entry of the data matrix a weight corresponding to its relative importance. This problem is referred to as weighted low-rank approximation (WLRA). We prove that computing an optimal weighted low-rank approximation is $\mathcal{NP}$-hard, already when a rank-one approximation is sought. In fact, we show that it is hard to compute approximate solutions to the WLRA problem with some prescribed accuracy. Our proof is based on a reduction from the maximum-edge biclique problem, and applies to strictly positive weights as well as binary weights; the latter corresponding to low-rank matrix approximation with missing data.

Because the reduction uses nonnegative input matrices, these results also apply to NMF implying that weighted NMF and NMF with missing data are both $\mathcal{NP}$-hard to approximate in the rank-one case (while rank-one NMF can be solved in polynomial time).

[73] N. Gillis and F. Glineur, **Low-Rank Matrix Approximation with Weights or Missing Data is NP-hard**, CORE Discussion paper 2010/75, 2010.

Finally, a conclusion will summarize the contributions of the thesis, attempt to put them into perspective, and give some directions for further research.

Chapter 2

Preliminaries

This thesis revolves around three main topics: optimization, low-rank matrix approximation and computational complexity. In this chapter, we summarize some well-known theoretical results which will be the groundwork of this thesis.

2.1 Optimization

Optimization, also referred to as mathematical programming, is an important field of applied mathematics. Given a set of feasible solutions and an objective function (i.e., a quality measure), it looks for (one of) the best possible solution in the feasible domain. In general, optimization problems can be described as follows

$$\min_{x \in X \subseteq \mathbb{R}^n} \; f(x), \tag{P}$$

where $f : X \to \mathbb{R} : x \to f(x)$ is the objective function, and X is the (feasible) domain with

$$X = \{x \in \mathbb{R}^n \mid g_i(x) = 0 \; \forall \, i \in E \text{ and } g_i(x) \geq 0 \; \forall \, i \in I\},$$

where $g_i : X \to \mathbb{R} : x \to g_i(x)$ define an equality constraint for $i \in E$ or an inequality constraint for $i \in I$.

2.1.1 First-Order Optimality Conditions

Assuming functions f and g_i's are continuously differentiable (i.e., ∇f and ∇g_i's are well-defined and continuous on X), one can use this first-order information in order to derive necessary conditions for feasible solutions of (P) to be optimal. In fact, it is clear that if a solution x is globally optimal, it must also be locally

optimal. Intuitively, it means that the first-order approximation of the function around x

$$f(x + \delta x) \approx f(x) + \nabla f(x)^T \delta x,$$

must be larger than $f(x)$ in the domain, i.e., $\nabla f(x)^T \delta x \geq 0$ for any feasible direction δx, i.e., for any δx pointing inside the domain (including $\delta x \to 0$ which might be tangent to the domain). For example, if there is no constraint, i.e., $X = \mathbb{R}^n$, all directions are feasible and it reduces to $\nabla f(x) = 0$. In the general constrained case, these conditions are referred to as Karush-Khun-Tucker (KKT) optimality conditions. Defining the set of active constraints at point x as $A(x) = \{j \mid g_j(x) = 0\}$, they can be described as follows. Assuming that the gradients $\nabla g_i(x)$ $i \in A(x)$ of the active constraint at x are *linearly independent*, the first-order necessary conditions for a solution x to be locally optimal are given by

$$\begin{aligned} \nabla f(x) &= \sum_{i \in I \cup E} \mu_i \nabla g_i(x), \\ g_i(x) &= 0, \quad i \in E, \\ g_i(x) \geq 0,\ \mu_i \geq 0,\ \mu_i g_i(x) &= 0, \quad i \in I, \end{aligned} \tag{KKT}$$

where each μ_i is the Lagrangian multiplier (dual variable) associated with the constraint g_i. A point satisfying the KKT conditions will be referred to as a first-order stationary point, or a *stationary point* for short.

2.1.2 Convexity

If the function f is convex and if the functions g_i are affine for $i \in E$ and concave for $i \in I$ (more generally, X is convex), problem (P) is *convex*. This notably implies that

- ◇ The set of optimal solutions is convex ;
- ◇ Any local minimum is also global ;
- ◇ The (KKT) conditions are sufficient to guarantee global optimality ;
- ◇ If there exists a strictly feasible solution (an interior point), i.e., a point x_s such that $g_i(x_s) = 0 \; \forall \, i \in E$ and $g_i(x_s) > 0 \; \forall \, i \in I$, the (KKT) conditions are *necessary and sufficient* (without assuming the linear independence of the gradients). This condition is referred to as *Slater's condition* and x_s is also called a Slater point.

In addition to (and in part because of) these nice properties, this class of problems can in general be solved efficiently, e.g., polynomial-time algorithms are available for linear programming (LP), quadratic programming (QP) recently extended to second-order cone programming (SOCP), semidefinite programming (SDP), geometric programming (GP), etc. However, it is important

to keep in mind that there are convex problems for which no polynomial-time algorithm are known, e.g., copositive programming which deals with the following feasible set of matrices

$$C = \{A \in \mathbb{S}^n \mid x^T Ax \geq 0 \text{ for all } x \in \mathbb{R}^n_+\},$$

called the copositive cone, can be used to model $\mathcal{NP}$-hard problems, see [60] and the references therein; it is therefore $\mathcal{NP}$-hard to solve copositive programs in general. In this case, it is even $\mathcal{NP}$-complete to check whether a point does not belong to the feasible set. In contrast, some non-convex problems are easy to solve. Many problems can be reformulated as efficiently solvable convex programs, or can be solved with dedicated algorithms; possibly after reformulation. In fact, it is well-known that

Formulation is crucial in mathematical programming.

We refer to [147] and references therein for a survey on these issues.

2.1.3 Block-Coordinate Descent Methods

There exist many techniques to locally improve a current solution x of an optimization problem (P), e.g., gradient descent, conjugate gradient, Newton and Quasi-Newton methods. Under some (often mild) additional conditions, they can guarantee convergence to a first-order stationary point.

Block-coordinate descent is another class of methods (also referred to as alternating variables) which work as follows:

1. Define k disjoint subset sets $F_i \subset N$ $1 \leq i \leq k$ whose union is equal to $N = \{1, 2, \ldots, n\}$, i.e.,

$$\cup_{1 \leq i \leq k} F_i = N = \{1, 2, \ldots, n\} \quad \text{and} \quad F_i \cap F_j = \emptyset \; \forall i \neq j.$$

2. Repeat until some stopping criterion is met

   For $i = 1, 2, \ldots, k$, fix variables x_j $j \in N \backslash F_i$ and minimize (P) in the remaining variables x_j $j \in F_i$.

In many cases, block-coordinate descent methods fail to converge rapidly when they get close to stationary points of (P), typically because of their zigzagging behavior (similar to what is frequently observed for gradient descent approaches, see, e.g., [12]). However, when the blocks of coordinates are rather large and can be optimized efficiently (possibly up to global optimality), they can be proved to be a powerful technique. At least, they often exhibit a relatively fast initial convergence to the neighborhood of a stationary point. If one requires a solution with high accuracy, one can then switch to a more sophisticated second-order scheme, such as Newton's method. Block-coordinate

descent methods have recently attracted much interest, notably because in general (1) they converge initially quite fast, (2) they are easy to implement, (3) they can deal with huge scale problems, and (4) in practice, it does not always make much sense to look for very accurate solutions (i.e., stationary points) because the underlying mathematical model is often approximate and built with noisy data.

It is sometimes possible to perform an exact block-coordinate descent, i.e., find a global minimum for (P) for each block of variables. In that particular case, we can ensure the following

Theorem 2.1. *[128], [12, p.268] The limit points of the iterates of an exact block-coordinate descent algorithm are stationary points provided that the following two conditions hold:*

1. *each block of variables is required to belong to a closed convex set,*
2. *the minimum computed at each iteration for a given block of variables is uniquely attained.*

Theorem 2.2. *[82] The limit points of the iterates of an exact* two-*block coordinate descent algorithm are stationary points provided that the following two conditions hold:*

1. *each block of variables is required to belong to a closed convex set,*
2. *the objective function is continuously differentiable.*

Hence exact two-block coordinate descent does not require the minimum of the subproblems to be uniquely attained to guarantee convergence to a stationary point.

2.2 Low-Rank Matrix Approximation

Given a real matrix $A \in \mathbb{R}^{m \times n}$, we would like to find its best approximation $B \approx A$ such that $\text{rank}(B) \leq r$ (possibly with additional constraints on B). Using the Frobenius norm of the difference between A and B as an objective function, this problem can be formulated as

$$\min_B ||A - B||_F^2, \quad \text{such that } \text{rank}(B) \leq r. \tag{LRA}$$

Since $\text{rank}(B) \leq r$, B can be decomposed as the product of two matrices $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{n \times r}$ with $B = UV^T$, and (LRA) can be equivalently written as

$$\min_{U,V} ||A - UV^T||_F^2. \tag{LRA}$$

The following two theorems provide a way to compute an optimal solution of (LRA).

Theorem 2.3 (Singular value decomposition (SVD), [80])**.** *If A is a real m-by-n matrix, then there exist orthogonal matrices*

$$U = [u_1\, u_2\, \ldots\, u_m] \in \mathbb{R}^{m\times m} \text{ and } V = [v_1\, v_2\, \ldots\, v_n] \in \mathbb{R}^{n\times n},$$

such that

$$U^T AV = \Sigma = diag(\sigma_1, \sigma_2, \ldots, \sigma_p) \in \mathbb{R}^{m\times n}, \quad p = \min(m, n),$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ are the singular values of A. This also implies $A = U\Sigma V^T$, $AV = U\Sigma$, $A^T U = V\Sigma^T$,

$$Av_i = \sigma_i u_i \quad \text{and} \quad A^T u_i = \sigma_i v_i \qquad \text{for all } 1 \leq i \leq p,$$

and $A = \sum_{i=1}^{p} \sigma_i u_i v_i^T$ (sum of p rank-one terms).

Any (σ, u, v) satisfying $Av = \sigma u$, $A^T u = \sigma v$, $||u||_2 = ||v||_2 = 1$, $\sigma \geq 0$ will be called a *singular triplet of A* (notice that $\sigma = u^T Av$).

Theorem 2.4 (Eckart-Young [62])**.** *Given (U, Σ, V) a singular value decomposition of A, and introducing the truncated sum of rank-one terms $A_r = \sum_{i=1}^{r} \sigma_i u_i v_i^T$, we have that A_r solves* (LRA) *and*

$$\min_{\text{rank}(B)\leq r} ||A - B||_F^2 = ||A - A_r||_F^2 = \sum_{i=r+1}^{\min(m,n)} \sigma_i^2.$$

2.2.1 Rank-One Approximation

In this thesis, we will be particularly interested by rank-one approximations, i.e., the case where approximation B can be written as an outer product σuv^T,

$$\min_{\sigma\in\mathbb{R}, u\in\mathbb{R}^m, v\in\mathbb{R}^n} ||A - \sigma uv^T||_F^2, \quad \text{such that } \sigma \geq 0, ||u||_2 = 1, ||v||_2 = 1, \tag{LRA-1}$$

and, by Theorems 2.3 and 2.4, an optimal solution is given by any singular triplet (σ, u, v) of A associated with the largest singular value $\sigma = \sigma_1(A)$. In the following, we derive some alternative formulations for (LRA-1). Let $A \neq 0$ (otherwise the problem is trivial) and let (σ, u, v) be an optimal solution of (LRA-1). Observing that

$$\begin{aligned} ||A - \sigma uv^T||_F^2 &= \left\langle A - \sigma uv^T, A - \sigma uv^T \right\rangle \\ &= \langle A, A\rangle - 2\sigma \left\langle A, uv^T \right\rangle + \sigma^2 \left\langle uv^T, uv^T \right\rangle \\ &= ||A||_F^2 - 2\sigma \left\langle A, uv^T \right\rangle + \sigma^2, \end{aligned}$$

the optimal solution for variable σ is given by $\sigma = \max(0, u^T Av)$. In fact, either $\langle A, uv^T \rangle = u^T Av$ is nonnegative and $\sigma = u^T Av$, or it is negative and $\sigma = 0$. Since $A \neq 0$, $u^T Av$ is strictly positive hence $\sigma > 0$ and $\sigma = u^T Av$. Therefore at optimality we must have

$$||A - \sigma uv^T||_F^2 = ||A||_F^2 - (u^T Av)^2,$$

and minimizing $||A - \sigma uv^T||_F^2$ is equivalent to maximizing $(u^T Av)^2$. Since $u^T Av \geq 0$, (LRA-1) is then equivalent to

$$\max_{u,v} \quad u^T Av, \quad \text{such that } ||u||_2 = 1, ||v||_2 = 1. \tag{PSV}$$

Optimal solutions (u, v) of (PSV) are pairs of principal singular vectors associated with the maximum singular value σ_1 of A. Given u, we have that the optimal v in (PSV) is given by

$$v = \frac{A^T u}{||A^T u||_2}.$$

One can then equivalently reformulate (PSV) as[1]

$$\max_{u} \quad \frac{u^T AA^T u}{||A^T u||_2} = ||A^T u||_2, \quad \text{such that } ||u||_2 \leq 1, \tag{PC}$$

with optimal value $||A^T||_2 = ||A||_2$ (by definition). We have then that, for $A \neq 0$,

$$(\sigma, u, v) \text{ is an optimal solution of (LRA-1)}$$

$$\Longleftrightarrow$$

$$(u, v) \text{ is an optimal solution of (PSV) and } \sigma = u^T Av$$

$$\Longleftrightarrow$$

$$u \text{ is optimal for (PC), } v = \frac{A^T u}{||A^T u||_2} \text{ and } \sigma = u^T Av = ||A||_2 = \sigma_1(A).$$

Stationary Points

One can derive the stationarity conditions for (LRA-1) and obtain

$$Av = \sigma u, A^T u = \sigma v, \sigma = u^T Av \geq 0, ||u||_2 = ||v||_2 = 1, \tag{2.1}$$

implying that (σ, u, v) is a singular triplet of A if and only if it is a stationary point of (LRA-1).

[1]The equality constraints $||u||_2 = 1$ has been relaxed without loss of generality to $||u||_2 \leq 1$ since the constraint will be active at optimality; otherwise it is trivial to construct a better solution multiplying u by $||u||_2^{-1}$.

Theorem 2.5. *The set of stationary points of* (LRA-1) *is given by the set of singular triplets of A. The set of optimal solutions is the set of singular triplets associated with the largest singular value(s). The other stationary points are saddle points.*

Proof. The first part of the proof is direct since singular triplets are defined as in Equation (2.1).

For the second part of the proof, assume $A \neq 0$ otherwise the result is trivial. Let (σ, u, v) be any stationary point of (LRA-1) with $\sigma < \sigma_1 = \sigma_1(A)$ and note (σ_1, u_1, v_1) an optimal solution of (LRA-1). Since $\sigma < \sigma_1$, u (resp. v) is orthogonal to u_1 (resp. v_1). In fact, they are associated with different eigenvalues of the symmetric matrix AA^T ($AA^T u = \sigma^2 u$ and $A^T Av = \sigma^2 v$, see Equation (2.1)). Then, for

$$u' = \sqrt{1-\epsilon^2}u + \epsilon u_1 \quad \text{and} \quad v' = \sqrt{1-\epsilon^2}v + \epsilon v_1,$$

where $-1 \leq \epsilon \leq 1$, we have $||u'||_2 = ||v'||_2 = 1$, and

$$u'^T Av' = (1-\epsilon^2)\sigma + \epsilon^2\sigma_1 = \sigma + \epsilon^2(\sigma_1 - \sigma) = \sigma' > \sigma.$$

Hence (σ, u, v) is a saddle point (because it is not locally optimal). □

The Power Method

A possible way to find singular triplets associated with a maximum singular value is to solve (LRA-1) or, equivalently, (PSV). Applying an exact two-block coordinate descent scheme (cf. Section 2.1.3) to (PSV) leads to the following updates

$$v \leftarrow \frac{A^T u}{||A^T u||_2}, \qquad u \leftarrow \frac{Av}{||Av||_2},$$

which amounts to performing

$$u \leftarrow AA^T u, \quad u \leftarrow \frac{u}{||u||_2}.$$

This is the *power method* applied to AA^T which is used to compute the eigenvector associated with the largest eigenvalue (in module). This algorithm is guaranteed to converge given that the initial vector u is not perpendicular to the principal eigenspace of AA^T (or equivalently, to the principal singular subspace of the columns of A).

2.2.2 Nonnegative Matrices

It is clear that if A is nonnegative, any solution (σ, u, v) of (LRA-1) can be improved by taking its absolute value $(\sigma, |u|, |v|)$ since $(A_{ij} - \sigma u_i v_j)^2 \geq (A_{ij} - \sigma|u_i||v_j|)^2$ because $A_{ij} \geq 0$. This is also a consequence of the Perron-Frobenius Theorem.

Theorem 2.6. *For any nonnegative real matrix A, there exists an optimal nonnegative solution $u \geq 0$ and $v \geq 0$ to* (LRA-1).

Remark that there might exist optimal solutions with negative entries, if the matrix M is reducible [9] and $\sigma_1(M) = \sigma_2(M)$. The simplest example is the 2-by-2 identity matrix $I_2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Two singular triplets of I_2 associated with the unique singular value (=1) are given by $\left(1, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right)$ and $\left(1, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}\right)$. Using the same construction as in the proof of Theorem 2.5, the following are optimal rank-one approximations

$$\begin{pmatrix} \lambda_1^2 & \lambda_1\lambda_2 \\ \lambda_1\lambda_2 & \lambda_2^2 \end{pmatrix}, \quad \text{for any } \lambda_1^2 + \lambda_2^2 = 1.$$

For example, with $\lambda_1 = -\lambda_2 = \frac{\sqrt{2}}{2}$, we obtain an optimal solution

$$\begin{pmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{pmatrix},$$

with negative entries.

2.3 Computational Complexity

In this section, we define some key concepts of computational complexity and explain intuitively the computational models we are going to work with. We refer the reader to [5, 16, 67] and references therein for detailed discussions on the subject.

2.3.1 The Turing Machine Model

The Turing machine model aims to model any algorithm or computation that can be carried out on a computer. Given a finite input (whose size is the number of bits needed to represent it), the machine can perform a finite number of operations in order to generate a finite output. The main purpose of computational complexity is to classify problems that are formulated on these machines in different categories. The aim is to predict whether it is possible to solve (or approximate) these problems efficiently (i.e., in polynomial time) on such a machine.

Complexity Classes: $\mathcal{NP}$, $\mathcal{P}$, $\mathcal{NP}$-Complete and $\mathcal{NP}$-Hard

$\mathcal{NP}$ refers to the class of *decision* problems, i.e., with a yes/no answer, for which there exists a certificate for the yes-answer that can be verified by a

polynomial-time algorithm[2] on a Turing machine.

The class $\mathcal{P} \subset \mathcal{NP}$ is the class of decision problems that can be decided by a polynomial-time algorithm (it requires the number of operations to be bounded by a polynomial in the size of the input).

A class of problems A can be reduced to a class B if there exists a function r (a reduction) that can be computed in polynomial time, and that transforms any problem of A into a problem of B in a polynomial number of operations in such a way that for any problem instance $a \in A$: a is a yes-instance of A if and only if $r(A)$ is a yes-instance of B. This result implies that any algorithm for B would solve A, i.e., that problems in B are at least as hard than those in A.

The class $\mathcal{NP}$-complete $\subset \mathcal{NP}$ is the class of the most difficult problems in $\mathcal{NP}$, i.e., problems such that any problem in $\mathcal{NP}$ can be reduced to them.

A well-known open question in computer science is: Is $\mathcal{P} = \mathcal{NP}$?, i.e., can any decision problem which can be verified in polynomial time be solved in polynomial time? It is in general believed that $\mathcal{P} \neq \mathcal{NP}$. Showing that a problem is $\mathcal{NP}$-complete then implies that it is probably not possible to find a polynomial-time algorithm to solve it (at least nobody has succeeded so far). In particular, this justifies the use of polynomial-time algorithms that find 'good' but non-optimal solutions (e.g., using heuristic approaches such as local search or convex relaxations; sometimes with guaranteed approximation properties).

Another class of problem is $\mathcal{NP}$-hard $\supset$ $\mathcal{NP}$-complete. It contains any problem (not necessarily a decision problem, e.g., an optimization problem) to which any problem in $\mathcal{NP}$ can be reduced.

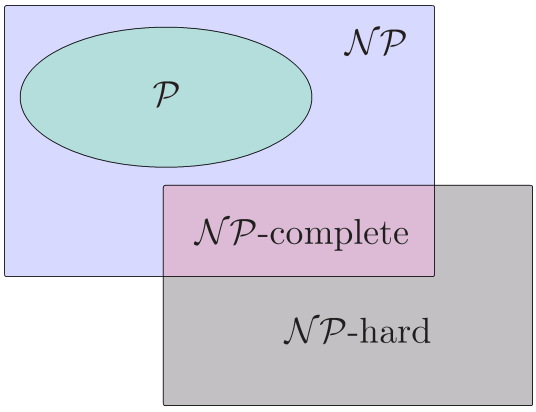

Figure 2.1: Illustration of the complexity classes if $\mathcal{P} \neq \mathcal{NP}$.

[2] $\mathcal{NP}$ stands for nondeterministic polynomial, which roots from another equivalent definition of $\mathcal{NP}$, namely the class of decision problems solvable in polynomial time by a nondeterministic machine.

Example 2.1. Let define the following class of problems, referred to as Quadratic Programming (QP)

> Instance: A is an m-by-n rational matrix, b is a rational m-dimensional vector, H is a rational symmetric n-by-n matrix , c is a rational n-dimensional vector and K is a rational number.
> Question: Is there an n-dimensional vector x such that $Ax \geq b$ and $x^T Hx + c^T x \leq K$?

QP has first been shown to be $\mathcal{NP}$-hard: Sahni [133] showed that partition problems can be reduced to QP, i.e., any instances of Partition can be reduced to an instance of QP (see also [67, p. 245, MP2]). Pardalos and Vavasis [125] showed that QP is $\mathcal{NP}$-hard even if only one eigenvalue of H is negative, by reduction from the clique problem.

However, it is not easy to determine whether QP is in $\mathcal{NP}$ or not, because some instances of QP could potentially have a solution x not polynomially bounded in the size of the input (e.g., irrational solutions) and, therefore, no polynomial-time algorithm could verify a certificate of a yes-instance. Vavasis [146] showed that QP actually belongs to $\mathcal{NP}$: he proved that there always exists a solution to QP which is rational, and polynomially bounded by the size of the input. This implies that there exists a polynomial-length certificate for any yes-instance of QP above and therefore that QP is in $\mathcal{NP}$, hence also $\mathcal{NP}$-complete.

2.3.2 Real Computations

In many problems one has to deal with real numbers, which cannot be represented by a finite number of digits (e.g., solutions to $x^2 = 2$ are irrational). Blum, Cucker, Shub and Smale [16] introduced a model to deal with real computations. It assumes that there exists a computer able to deal with infinite-precision real numbers. In this model, the size of the input is defined as the vector length of the input, in opposition with the Turing model which counts the number of bits needed to represent it. Within this theory, one can solve $\mathcal{NP}$-complete problems in the Turing machine model in polynomial time (using intricate constructions). One must then carefully specify the model used.

Example 2.2 ([15])**.** Most complexity results in convex optimization assume infinite-precision arithmetic. For example, let us consider linear programming (LP) defined as follows

$$\min_{x \in \mathbb{R}^n, x \geq 0} c^T x \quad \text{such that} \quad Ax \geq b,$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. The size of the input is $mn + m + n$ and LP can be solved in polynomial time to any given precision in the real

computation model. However, it is an open question to determine whether solving LP exactly belongs to the class of polynomially solvable problems in the real model. In fact, the number of iterations might depend on the size of the coefficients of matrix A and vectors b and c (which is not part of the input size). The open question is then: does LP admit a strongly polynomial-time algorithm?, i.e., an algorithm whose complexity is polynomial in mn, hence does not depend on the size of the coefficients of the input.

For rational input, LP can be solved in polynomial time in the Turing model because the optimal solution is rational, and polynomially bounded by the size of the input.

In this thesis, we will mostly focus on the Turing machine model.

Chapter 3

Nonnegative Rank

In this chapter, we seek exact nonnegative matrix factorizations of nonnegative matrices $M \in \mathbb{R}_+^{m\times n}$, i.e., we look for $U \in \mathbb{R}_+^{m\times k}$ and $V \in \mathbb{R}_+^{k\times n}$ such that $M = UV = \sum_{i=1}^{k} U_{:i}V_{i:}$, or equivalently such that $||M - UV||_F^2 = 0$. The pair (U, V) will be called a rank-k nonnegative factorization[1] of M. The minimum k such that there exists a rank-k nonnegative factorization of M, i.e., the minimum number of nonnegative rank-one factors needed to reconstruct M exactly, is called the nonnegative rank of M and denoted $\text{rank}_+(M)$. Clearly,

$$\text{rank}(M) \leq \text{rank}_+(M) \leq \min(m, n).$$

Determining the nonnegative rank and computing the corresponding nonnegative factorization has been studied relatively recently in linear algebra [8, 37]. In the literature, much more attention has been devoted to the approximate problem (i.e., NMF), which has been widely used as a data analysis technique, see Chapter 1. Nevertheless, there are not too many theoretical results about the nonnegative rank and better characterizations could help practitioners. For example, efficient computations of exact nonnegative factorizations could help to design new NMF algorithms using a two-step strategy [148]: first approximate M with a low-rank nonnegative matrix A (e.g., using the singular value decomposition[2]) and then compute a nonnegative factorization of A. Bounds for the nonnegative rank could also help select the factorization rank of the NMF, replacing the trial and error approach often used by practitioners. For example, in hyperspectral image analysis, the nonnegative rank corresponds to the number of materials present in the image and its computation could lead to

[1] Notice that matrices U and V in a rank-k nonnegative factorization are not required to have rank k.

[2] Even though an optimal low-rank approximation of a nonnegative matrix might not necessarily be nonnegative, it is often the case in practice [100].

more efficient algorithms detecting these constitutive elements, see Chapter 7.

The main goals of this chapter are to (1) better understand the computational complexity of determining the nonnegative rank using a geometric interpretation, and its implications for NMF, (2) make connections with other related problems (in particular in computational geometry and combinatorial optimization) and (3) provide new bounds for the nonnegative rank. It is organized as follows.

In Section 3.1, we review the complexity results about the nonnegative rank, and its relationship with NMF. In Section 3.2, we explain how the nonnegative rank is closely related to other problems; in particular the characterization of the size of extended formulations in combinatioral optimization. In Section 3.3, we introduce a new related quantity called *restricted nonnegative rank*. Generalizing a recent result of Vavasis [148] (see also [114]), we show that computing this quantity is equivalent to a problem in polyhedral combinatorics, and fully characterize its computational complexity. In Section 3.4, based on the geometric interpretation of the nonnegative rank and the relationship with the restricted nonnegative rank, we derive new improved lower bounds for the nonnegative rank. In Section 3.5, we apply our results to slack matrices and linear Euclidean distance matrices. We obtain counter-examples to two conjectures of Beasly and Laffey [6], namely we show that the nonnegative rank of linear Euclidean distance matrices is not necessarily equal to their dimension, and that the rank of a matrix is not always greater than the nonnegative rank of its square. Finally, in Section 3.6, using the relationship between the restricted nonnegative rank of a matrix and its transpose, we further improve the bounds provided in Section 3.5.

3.1 Computational Complexity

Vavasis studies in [148] the algorithmic complexity of the NMF optimization problem, i.e., (NMF), see Section 1. More specifically, he studies the following problem, which he calls *exact nonnegative matrix factorization*:

> (Exact NMF) Given a nonnegative matrix $M \geq 0$ of rank r, find, if possible, two nonnegative factors $U \in \mathbb{R}^{m \times r}_+$ and $V \in \mathbb{R}^{r \times n}_+$ such that $M = UV$.

Clearly, exact NMF is equivalent to asking whether $\text{rank}_+(M) = \text{rank}(M) = r$? and, if yes, to compute a rank-r nonnegative factorization of M. In order to prove $\mathcal{NP}$-hardness of exact NMF, Vavasis introduces the following problem, called intermediate simplex

> (IS) Given a bounded polyhedron
>
> $$P = \{x \in \mathbb{R}^{r-1} \mid 0 \leq f(x) = Cx + d\},$$

> with $(C\ d) \in \mathbb{R}^{m \times r}$ of rank r, and a set S of n points in P not contained in any hyperplane (i.e., conv(S) is full-dimensional), are there r points in P whose convex hull T contains S, i.e., is there a polytope T with r vertices such that $S \subseteq T \subseteq P$?

P is referred to as the outer simplex, conv(S) as the inner simplex, and T as the intermediate simplex.

Vavasis then proves the following result

Theorem 3.1 ([148]). *There exists a polynomial-time reduction from exact NMF to IS and vice-versa.*

He concludes by showing that IS is $\mathcal{NP}$-hard using a reduction from 3-SAT. His construction requires the rank r of matrix M to increase to obtain $\mathcal{NP}$-hardness (and therefore its dimensions m and n have to increase as well), i.e., he shows that there is no algorithm running in polynomial time in r solving the exact NMF problem, unless $\mathcal{P} = \mathcal{NP}$. NMF is therefore also $\mathcal{NP}$-hard, since any optimal solution to NMF can be used to answer the exact NMF problem (the answer being positive if and only if the optimal objective value of NMF is equal to zero).

However, if rank r of matrix M is fixed, no complexity results are known, and the following questions are still open (see also [148])

- ◇ What is the complexity of NMF when the rank of the matrix M is fixed?
- ◇ What is the complexity of NMF when the factorization rank is fixed?

In contrast, in the special cases when rank of matrix M is equal to 1 or 2, the exact NMF problem can always be answered in the affirmative:

1. When rank$(M) = 1$, it is obvious that for any nonnegative rank-one matrix $M \geq 0$ there are nonnegative factors $u \geq 0$ and $v \geq 0$ such that $M = uv^T$.

2. When nonnegative matrix M has rank 2, Thomas has shown [143] that exact NMF is also always possible (see also [37]). The fact that any rank-two nonnegative matrix can be exactly factorized as the product of two rank-two nonnegative matrices can be explained geometrically as follows: viewing columns of M as points in $\mathbb{R}^m$, the fact that M has rank 2 implies that the set of its columns belongs to a two-dimensional subspace. Furthermore, because these columns are nonnegative, they belong to a two-dimensional pointed cone, see Figure 3.1. Since such a cone is always spanned by two extreme vectors, this implies that all columns of M can be represented exactly as nonnegative linear combinations of

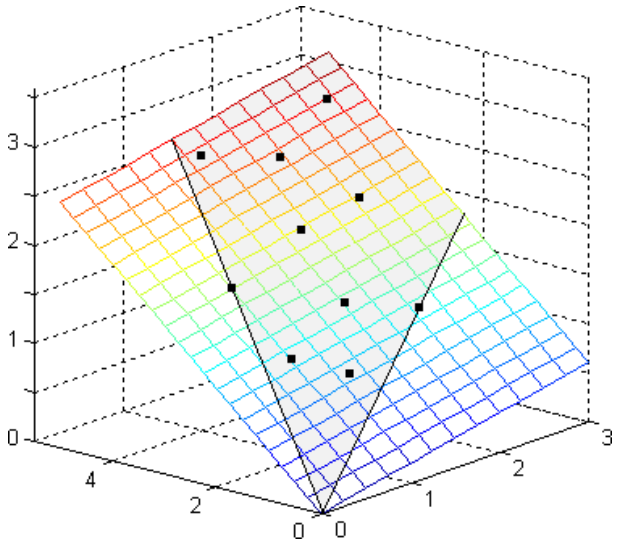

Figure 3.1: Illustration of exact NMF for a rank-two 3-by-10 matrix.

two nonnegative vectors, and therefore the exact NMF is always possible[3]. Moreover, these two extreme columns can easily be computed in polynomial time (using for example the fact that they define an angle of maximum amplitude among all pairs of columns).

Based on these observations, rank-one[4] NMF can be solved in polynomial time: the Perron-Frobenius theorem implies that the dominant left and right singular vectors of a nonnegative matrix M are nonnegative, while the Eckart-Young theorem states that the outer product of these dominant singular vectors is the best rank-one approximation of M; and these vectors can be computed in polynomial time, see Section 2.2. Also, when the optimal rank-two approximation of matrix M is unique and nonnegative[5], rank-two NMF can be solved in polynomial time. However, this optimal rank-two approximation is not always nonnegative; typically when M contains many relatively small entries (e.g., zeros, see Section 2.2.2). For example, the unique optimal rank-two

[3]The reason why this property no longer holds for higher values of the rank r is that a r-dimensional cone is not necessarily spanned by a set of r vectors when $r > 2$.

[4]Rank-one NMF refers to as the NMF optimization problem NMF with $r = 1$. Notice that, in this chapter, r is often used to refer to the rank of the nonnegative matrix M, which will be clear from the context.

[5]If the optimal rank-two approximations is non-unique (i.e., $\sigma_2(M) = \sigma_3(M) = \cdots = \sigma_i(M)$, $i \geq 3$), then it is non-trivial to check whether a nonnegative solution exists, see Corollary 3.5, so that the complexity is also unknown in this case.

approximation of the following matrix

$$M = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix},$$

is

$$\begin{pmatrix} 0.8924 & 1.1341 & 0.9403 \\ 0.2417 & 0.6986 & 1.1341 \\ -0.1938 & 0.2417 & 0.8924 \end{pmatrix}.$$

Hence the complexity of rank-two (NMF) is not known (more generally it is not known for any fixed factorization rank greater than 2, see also Chapter 5). Furthermore, to the best of our knowledge, the complexity of *the exact NMF problem and NMF of a matrix of any fixed rank greater than* 3 *are still unknown.*

3.2 Extended Formulations

The nonnegative rank is closely related to other areas of optimization, in particular to linear programming (LP) extended formulations in combinatorial optimization.

An extended formulation (or lifting) for a polytope $P \subseteq \mathbb{R}^n$ is a polyhedron $Q \subseteq \mathbb{R}^{n+p}$ such that

$$P \;=\; \text{proj}_x(Q) := \{x \in \mathbb{R}^n \,|\, \exists y \in \mathbb{R}^p \text{ s.t. } (x, y) \in Q\}.$$

Extended formulations whose size (number of constraints plus number of variables defining Q) is polynomial in n are called *compact* and are of great importance in integer programming. They allow to reduce significantly the size of certain linear programs arising in the context of integer programming and combinatorial optimization, and therefore provide a way to solve them efficiently, i.e., in polynomial time (see [39] for a survey). Yannakakis [152, Theorem 3] showed that the minimum size s of an extended formulation of a polytope[6]

$$P = \{x \in \mathbb{R}^n \,|\, Cx + d \geq 0, Ax = b\},$$

is of the same order[7] as the sum of its dimension n and the nonnegative rank of its slack matrix $S_M \geq 0$, where each column of the slack matrix is defined as

$$S_M(:, i) = Cv_i + d \geq 0, \quad i = 1, 2, \ldots, m, \tag{3.1}$$

[6]This can be generalized to polyhedra [39].

[7]One can actually show that the minimum number of facets of an extension of P is equal to the nonnegative rank of the slack matrix of P, see, e.g., 'Nonnegative rank of a matrix and projections onto a polyhedron' on `http://dirkolivertheis.wordpress.com/`.

and vectors v_i are the m vertices of the polytope P. Formally, we then have

$$s = \Theta(n + \text{rank}_+(S_M)).$$

In particular, any rank-k nonnegative factorization (U, V) of $S_M = UV$ provides the following extended formulation for P with size $\Theta(n + k)$

$$Q = \{(x, y) \in \mathbb{R}^{n+k} \mid Cx + d = Uy, Ax = b, y \geq 0\}. \tag{3.2}$$

In fact, $\text{proj}_x(Q) \subseteq P$ since $Uy \geq 0$ implies $Cx + d \geq 0$ for any $x \in \text{proj}_x(Q)$, and $P \subseteq \text{proj}_x(Q)$ since $Cv_i + d = UV(:, i)$ implies that $(v_i, V(:, i)) \in Q$ for all i and therefore each vertex v_i of P belongs to $\text{proj}_x(Q)$. Notice that the system $Cx + d = Uy$ contains at most $n + k$ linearly independent equalities. Intuitively, this extended formulation parametrizes the space of slacks of the original polytope with the convex cone $\{Uy \mid y \geq 0\}$.

It is therefore interesting to compute bounds for the nonnegative rank in order to estimate the size of these extended formulations. Recently, Goemans [79] used this result to show that the size of LP formulations of the permutahedron (polytope whose $n!$ vertices are permutations of $[1, 2, \ldots, n]$) is at least $\Omega(n \log(n))$ variables plus constraints (see Section 3.4).

We will see in Section 3.4.1 that the nonnegative rank is closely related to a problem in computational geometry consisting in finding a polytope with minimum number of vertices nested between two given polytopes. Therefore a better understanding of the properties of the nonnegative rank would presumably also allow to characterize better the solutions of this geometric problem. The nonnegative rank also has connections with other problems, e.g., in communication complexity theory [107, 152], probability [26], and graph theory (cf. Section 3.4).

3.3 Restricted Nonnegative Rank

In this section, we analyze the following quantity

Definition 3.1. The *restricted nonnegative rank* of a nonnegative matrix M is the minimum value of k such that there exists $U \in \mathbb{R}_+^{m \times k}$ and $V \in \mathbb{R}_+^{k \times n}$ with $M = UV$ and $\text{rank}(U) = \text{rank}(M)$, i.e., $\text{col}(U) = \text{col}(M)$. It is denoted $\text{rank}_+^*(M)$.

In particular, given a nonnegative matrix M, we are interested in computing its restricted nonnegative rank $\text{rank}_+^*(M)$ and a corresponding nonnegative factorization, i.e., solve

(RNR) Given a nonnegative matrix $M \in \mathbb{R}_+^{m\times n}$, find $k = \text{rank}_+^*(M)$ and compute $U \in \mathbb{R}_+^{m\times k}$ and $V \in \mathbb{R}_+^{k\times n}$ such that $M = UV$ and $\text{rank}(U) = \text{rank}(M) = r$.

Without the rank constraint on the matrix U, this problem reduces to the standard nonnegative rank problem. Motivation to study this restriction includes the following

1. The restricted nonnegative rank provides a new *upper* bound for the nonnegative rank, since $\text{rank}_+(M) \leq \text{rank}_+^*(M)$.

2. The restricted nonnegative rank can be characterized much more easily. In particular, its geometrical interpretation (Section 3.3.1) will lead to new improved *lower* bounds for the nonnegative rank (Sections 3.4 and 3.5).

RNR is a generalization of exact NMF. Noting $r = \text{rank}(M)$, recall that exact NMF asks whether $\text{rank}_+(M) = r$ and, if the answer is positive, to compute a rank-r nonnegative factorization of M. If $\text{rank}_+(M) = r$ then it is clear that $\text{rank}_+^*(M) = \text{rank}_+(M)$ since the rank of U in any rank-r nonnegative factorization (U, V) of M must be equal to r.

The $\mathcal{NP}$-hardness result of Vavasis therefore also implies $\mathcal{NP}$-hardness of RNR when the rank of matrix M is not fixed. However, in the case where the rank r of matrix M is fixed, no complexity results are known (except in the trivial cases $r \leq 2$, see Section 3.1). The situation for RNR is quite different: we are going to show that RNR can be solved in polynomial time when $r = 3$ and that it is $\mathcal{NP}$-hard for any fixed $r \geq 4$. In particular, this result implies that exact NMF can be solved in polynomial time for rank-three nonnegative matrices, see Corollary 3.4.

In order to do so, we first show equivalence of RNR with another problem in polyhedral combinatorics, closely related to intermediate simplex (Section 3.3.1), and then apply results from the computational geometry literature to conclude about its computational complexity for fixed rank (Section 3.3.2).

3.3.1 Equivalence with the Nested Polytopes Problem

Let consider the following problem called nested polytopes problem (NPP):

(NPP) Given a bounded polyhedron

$$P = \{x \in \mathbb{R}^{r-1} \mid 0 \leq f(x) = Cx + d\},$$

with $(C\ d) \in \mathbb{R}^{m\times r}$ of rank r, and a set S of n points in P not contained in any hyperplane (i.e., $\text{conv}(S)$ is full-dimensional), find the minimum number k of points in P whose convex hull T contains S such that $S \subseteq T \subseteq P$.

Polytope P is referred to as the outer polytope, and $\text{conv}(S)$ as the inner polytope; note that they are given by two distinct types of representations: facets for P, extreme points for $\text{conv}(S)$.

The intermediate simplex problem mentioned earlier is a particular case of NPP in which one asks whether k is equal to r (which is the minimum possible value), i.e., if there exists a simplex T (defined by r vertices in a $r-1$ dimensional space) contained in P and containing S.

We now prove equivalence between RNR and NPP. It is a generalization of Theorem 3.1. The reductions are valid in both Turing machine and real computations models (see Section 2.3).

Theorem 3.2. *There is a polynomial-time reduction from RNR to NPP and vice-versa.*

Proof. Let us construct a reduction of RNR to NPP. First we (1) delete the zero rows and columns of M and (2) normalize its columns such that M becomes column stochastic (columns are nonnegative and sum to one). One can easily check that it gives a polynomially equivalent RNR instance [31]. We then decompose M as the product of two rank-r matrices (using, e.g., reduction to row-echelon form)

$$M = AB \iff M_{:i} = \sum_{l=1}^{r} A_{:l} B_{li} \; \forall i,$$

where $r = \text{rank}(M)$, $A \in \mathbb{R}^{m \times r}$ and $B \in \mathbb{R}^{r \times n}$. We observe that one can assume without loss of generality that the columns of A and B sum to one. Indeed, since M is column stochastic, at least one column of A does not sum to zero (otherwise all columns of $AB = M$ would sum to zero). One can then update A and B in the following way so that their columns sum to one:

- ◇ For each column of A which sums to zero, add a column of A which does not sum to zero, and update B accordingly;
- ◇ Normalize the columns of A such that each of them sums to one, and update B accordingly;
- ◇ Observe that since the columns of A sum to one, M is column stochastic and $M = AB$, the columns of B must also sum to one.

In order to find a solution of RNR, we have to find $U \in \mathbb{R}_+^{m \times k}$ and $V \in \mathbb{R}_+^{k \times n}$ such that $M = UV$ and $\text{rank}(U) = r$. For the same reasons as for A and B, U and V can be assumed to be column stochastic without loss of generality. Moreover, since

$$M = UV = AB,$$

and $\text{rank}(M) = \text{rank}(A) = \text{rank}(U) = r$, the column spaces of M, A and U coincide; implying that the columns of U must be a linear combination of the columns of A. The columns of U must then belong to the following set

$$Q = \{u \in \mathbb{R}^m \mid u \in \text{col}(A),\ u \geq 0 \text{ and } \sum_{i=1}^{m} u_i = 1\}.$$

One can then reduce the search space to the $(r-1)$-dimensional polyhedron corresponding to the coefficients of all possible linear combinations of the columns of A generating stochastic columns. Defining

$$C(:,i) = A(:,i) - A(:,r) \quad 1 \leq i \leq r-1, \text{ and } d = A(:,r),$$

and introducing affine function $f : \mathbb{R}^{r-1} \to \mathbb{R}^m : x \to f(x) = Cx + d$, which is injective since C is full rank (because A is full rank), this polyhedron can be defined as

$$\begin{aligned} P &= \{x \in \mathbb{R}^{r-1} \mid A(:,1{:}r-1)x + \Big(1 - \sum_{i=1}^{r-1} x_i\Big) A(:,r) \geq 0\} \\ &= \{x \in \mathbb{R}^{r-1} \mid f(x) \geq 0\}. \end{aligned}$$

Note that $B(1{:}r-1,j) \in P\ \forall j$ since $M(:,j) = AB(:,j) = f(B(1{:}r-1,j)) \geq 0\ \forall j$.

Let us show that P is bounded: suppose P is unbounded, then

$$\begin{aligned} \exists\, x \in P, \exists\, y \neq 0 \in \mathbb{R}^{r-1}, \forall \alpha \geq 0 \quad : \quad & x + \alpha y \in P, \\ & C(x + \alpha y) + d = (Cx + d) + \alpha Cy \geq 0. \end{aligned}$$

Since $Cx + d \geq 0$, this implies that $Cy \geq 0$. Observe that columns of C sum to zero (since the columns of A sum to one) so that Cy sums to zero as well; moreover, C is full rank and y is nonzero implying that Cy is nonzero and therefore that Cy must contain at least one negative entry, a contradiction.

Notice that the set Q can be equivalently written as

$$Q = \{u \in \mathbb{R}^m \mid u = f(x),\ x \in P\}.$$

Noting $X = [x_1\, x_2\, \ldots x_k] \in \mathbb{R}^{r-1 \times k}$, $f(X) = [f(x_1)\, f(x_2)\, \ldots f(x_k)] = CX + [d\, d \ldots d]$, we finally have that the following statements are equivalent

(i) $\exists U \in \mathbb{R}^{m \times k}, V \in \mathbb{R}^{k \times n}$ column stochastic with $\text{rank}(U) = \text{rank}(M)$, and $M = UV$,

(ii) $\exists x_1, x_2, \ldots x_k \in P$ and $V \in \mathbb{R}^{k \times n}$ column stochastic such that

$$M = f(B(1{:}r-1,:)) = f(X)V = f(XV),$$

(iii) $\exists x_1, x_2, \dots x_k \in P$ and $V \in \mathbb{R}^{k\times n}$ column stochastic such that

$$B(1{:}r{-}1, :) = XV.$$

The equivalence between (i) and (ii) follows from the above derivations (i.e., $U = f(X)$ for some $x_1, x_2, \dots x_k \in P$); $f(X)V = f(XV)$ because V is column stochastic (so that $[d\,d\,\dots\,d]V = [d\,d\,\dots\,d]$), and the second equivalence between (ii) and (iii) is a consequence from the fact that f is an injection.

We have then reduced RNR to NPP: find the minimum number k of points x_i in P such that n given points (the columns of $B(1{:}r{-}1, :)$ constructed from the columns of M, which define the set S in the NPP instance) are contained in the convex hull of these points (since V is column stochastic). Because all steps in the above derivation are equivalences, we have actually also defined a reduction from NPP to RNR; to map a NPP instance to a RNR instance, we take

$$M(:, i) = f(s_i) = Cs_i + d \geq 0,\ s_i \in S \quad 1 \leq i \leq n,$$

and $\operatorname{rank}(M) = r$ because the n points s_i are not all contained in any hyperplane (they affinely span P). □

It is worth noting that M would be the slack matrix of P if S was the set of vertices of P (cf. Section 3.2). This will be useful later in Section 3.5.1.

Remark 3.1 (Uniqueness)**.** The reduction from RNR to NPP is not unique. In fact, two different factorizations of M correspond to different NPP instances. However, these instances are simply affine transformations of each other, hence equivalent.

Let us show this formally. Let $M = AB$ and $M = CD$ be two factorizations of M, with the columns of A, B, C and D summing to one. We then have two different NPP instances corresponding to M:

$$S_B = \{B(1{:}r-1, j), 1 \leq j \leq n\} \ \subseteq\ P_A = \{\ x \in \mathbb{R}^{r-1} \mid A\begin{pmatrix} x \\ 1 - \Sigma(x) \end{pmatrix} \geq 0\ \}, \tag{NPP$_1$}$$

where $\Sigma(y) = \sum_{i=1}^{n} y_i, y \in \mathbb{R}^n$, and

$$S_D = \{D(1{:}r-1, j), 1 \leq j \leq n\} \ \subseteq\ P_C = \{\ x \in \mathbb{R}^{r-1} \mid C\begin{pmatrix} x \\ 1 - \Sigma(x) \end{pmatrix} \geq 0\ \}. \tag{NPP$_2$}$$

Since $AB = CD$ and A, B, C, D are full rank, there exists an invertible matrix Q such that $A = CQ$ and $B = Q^{-1}D$ where the columns of Q and Q^{-1} must sum to one (because the columns of A, B, C, D do). Let us note

$$Q = \begin{pmatrix} q_1 & q_2 & \dots & q_r \\ 1 - \Sigma(q_1) & 1 - \Sigma(q_2) & \dots & 1 - \Sigma(q_r) \end{pmatrix}.$$

Then

$$A \begin{pmatrix} x \\ 1-\Sigma(x) \end{pmatrix} = AQ^{-1}Q \begin{pmatrix} x \\ 1-\Sigma(x) \end{pmatrix} = C \begin{pmatrix} \sum_i x_i(q_i - q_r) + q_r \\ 1 - \Sigma(\sum_i x_i(q_i - q_r) + q_r) \end{pmatrix}.$$

Hence

$$x \in P_A \iff y = \sum_i x_i(q_i - q_r) + q_r = Q'x + q_r \in P_C,$$

where $Q'(:, i) = q_i - q_r$ $1 \leq i \leq r-1$. Finally, $f(x) = Q'x + q_r$ is an affine transformation from (NPP_1) to (NPP_2).

3.3.2 Computational Complexity

Rank-Three Matrices

Using Theorem 3.2, RNR of a rank-three matrix can be reduced to a two-dimensional nested polytopes problem. Therefore, one has to find a convex polygon T with minimum number of vertices nested in between two given convex polygons $S \subseteq P$. This problem has been studied by Aggarwal et al. in [1], who proposed an algorithm running in $\mathcal{O}(p \log(k))$ operations[8], where p is the total number of vertices of the given polygons S and P, and k is the number of vertices of the minimal nested polygon T. If M is a m-by-n matrix then $p \leq m + n$ since S has n vertices, and the polygon P is defined by m inequalities so that it has at most m vertices. Moreover $k = \text{rank}_+^*(M) \leq \min(m, n)$ follows from the trivial solutions $T = S$ and $T = P$. Finally, we conclude that one can compute the restricted nonnegative rank of a rank-three m-by-n matrix in $O\big((m+n)\log(\min(m,n))\big)$ operations.

Theorem 3.3. *For* $\text{rank}(M) \leq 3$*, RNR can be solved in polynomial time.*

Proof. Cases $r = 1, 2$ are trivial since any rank-1 (resp. 2) nonnegative matrix can always be expressed as the sum of 1 (resp. 2) nonnegative factors, see Section 3.1.

Case $r = 3$ follows from Theorem 3.2 and the polynomial-time algorithm of Aggarwal et al. [1]. □

For the sake of completeness, we sketch the main ideas of the algorithm of Aggarwal et al. They first make the following observations: (1) any vertex of a solution T can be assumed to belong to the boundary of the polygon P (otherwise it can be projected back on P in order to generate a new solution containing the previous one), (2) any segment whose ends are on the boundary of P and tangent to S (i.e., S is on one side of the segment, and the segment touches S) defines a polygon with the boundary of P which must contain at

[8] Wang generalized the result for non-convex polygons [149]. Bhadury and Chandrasekaran propose an algorithm to compute all possible solutions [13].

least one vertex of any feasible solution T (otherwise the tangent point on S could not be contained in T), see, e.g., set Q on Figure 3.2 delimited by the segment $[p_1, p_2]$ and the boundary of P, and such that $T \cap Q \neq \emptyset$ for any feasible solution T.

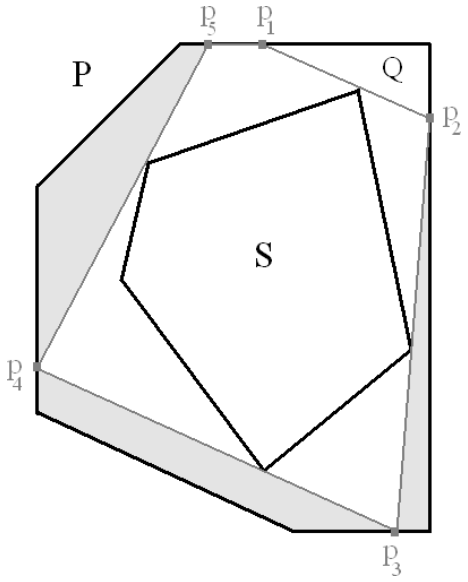

Figure 3.2: Illustration of the algorithm of Aggarwal et al. [1].

Starting from any point p_1 of the boundary of P, one can trace the tangent to S and hence obtain the next intersection p_2 with P. Point p_2 is chosen as the next vertex of a solution T, and the same procedure is applied (say k times) until the algorithm can reach the initial point without going through S, see Figure 3.2. This generates a feasible solution $T(p_1) = \text{conv}(\{p_1, p_2, \ldots, p_k\})$. Because of (1) and (2), this solution has at most one vertex more than an optimal one, i.e., $k \leq \text{rank}_+^*(M) + 1$ (since T determines with the boundary of P $k-1$ disjoint polygons tangent to S). Moreover, because of (1) and (2), there must exist a vertex of an optimal solution on the boundary of P between p_1 and p_2.

The point p_1 is then replaced by the so called 'contact change points' located on this part of the boundary of P while the corresponding solution $T(p_1)$ is updated using the procedure described above. The contact change points are: (a) the vertices of P between p_1 and p_2, and (b) the points for which one tangent point of $T(p_1)$ on S is changed when p_1 is replaced by them. This (finite) set of points provides a list of candidates where the number of vertices of the

solution T could potentially be reduced (i.e., where p_1 and p_k could coincide) by replacing p_1 by one of these points. It is then possible to check whether the current solution can be improved or not, and guarantee global optimality. In the example of Figure 3.2, moving p_1 on the (only) vertex of P between p_1 and p_2 generates an optimal solution of this RNR instance (since it reduces the original solution from 5 to 4 vertices).

Finally, Theorem 3.3 also allows to shed some light on the nonnegative rank and the nonnegative matrix factorization problems.

Corollary 3.4. *Exact NMF can be solved in polynomial time for nonnegative matrices of rank r smaller than 3.*

Proof. If $\text{rank}_+(M) = r$, then clearly $\text{rank}_+^*(M) = r$ since in any rank-r nonnegative factorization (U, V) of M, $\text{rank}(U) = r$. Therefore $\text{rank}_+(M) > r$ if and only if $\text{rank}_+^*(M) > r$, so that it suffices to compute $\text{rank}_+^*(M)$ to solve exact NMF (as mentioned in the introduction of this section). □

Corollary 3.5. *The optimal solution of* (NMF) *with $r = 3$ can be computed in polynomial time, given that*

(1) the best rank-three approximation A of M is unique and nonnegative, and

(2) its nonnegative rank $\text{rank}_+(A)$ *is smaller than three.*

Proof. The unique best rank-three approximation A of M can be computed in polynomial time, e.g., using the SVD, see Section 2.2.

If A is nonnegative and $\text{rank}_+(A) \leq 3$, a rank-3 nonnegative factorization of A can be computed with the algorithm of Aggarwal et al. (see Corollary 3.4). □

This result is similar as in the rank-two case, except that the nonnegative rank of a rank-two nonnegative matrix is always equal to two (cf. Section 3.1). Notice that if $\text{rank}(M) = 3$, then the assumptions of Theorem 3.5 reduce to having $\text{rank}_+(M) = 3$. In fact, if $\text{rank}_+(M) = 4$, we do not know how to solve the rank-three NMF problem in polynomial-time.

Remark 3.2. If the nonnegative rank of the best rank-three approximation A of M is larger than 3, then knowing A is useless for obtaining a rank-three NMF solution for M. In fact, even if we were able to compute an optimal rank-three NMF of A, it will most likely not be optimal for M.

Remark 3.3. If the best rank-three approximation of M is non-unique (i.e., $\sigma_i(M) = \sigma_{i-1}(M) = \cdots = \sigma_4(M) = \sigma_3(M)$, $i \geq 4$), the set of optimal solutions can be described from the singular triplets of M which can be computed in polynomial time (see Section 2.2). However, checking whether there exists an

optimal solution which is nonnegative and has nonnegative rank three seems to be non-trivial, and we do not know how to do it in general. This is why we need the uniqueness assumption.

Notice that, in the special case $\sigma_2(M) > \sigma_3(M) = \sigma_4(M) > \sigma_5(M)$,

$$A(\lambda) = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \sigma_3 u(\lambda) v(\lambda)^T,$$

where (σ_i, u_i, v_i) are the singular triplets associated with M, $u(\lambda) = \lambda u_3 + \sqrt{1-\lambda^2} u_4$, $v(\lambda) = \lambda v_3 + \sqrt{1-\lambda^2} v_4$, and $-1 \leq \lambda \leq 1$, is the set of optimal rank-three approximations of M (see Theorem 2.5). The aim is to find λ such that $A(\lambda) \geq 0$ and $\text{rank}_+(A(\lambda)) = 3$. Using the change of variable $\lambda = \cos(\theta)$, $\theta \in [0, 2\pi]$, checking whether there exists λ such that $A(\lambda) \geq 0$ is equivalent to checking if there exists θ such that

$$\cos^2(\theta) u_3 v_3^T + \cos(\theta)\sin(\theta)(u_3 v_4^T + u_4 v_3^T) + \sin^2(\theta) u_4 v_4^T \geq \frac{\sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T}{-\sigma_3}.$$

We then have mn inequalities of the type

$$f(\theta) = a\cos^2(\theta) + b\cos(\theta)\sin(\theta) + c\sin^2(\theta) + d \geq 0.$$

Multiplying d by $\cos^2(\theta) + \sin^2(\theta) (= 1)$, and dividing the inequalities by $\cos^2(\theta)$ (for $\cos(\theta) \neq 0$), we observe that $\frac{f(\theta)}{\cos^2(\theta)}$ is a quadratic function in $\tan(\theta)$. We can then easily compute its roots, hence an union of intervals for the admissible values of θ. We finally have to find θ in the intersection of these intervals satisfying $\text{rank}_+(A(\theta)) = 3$. For a given θ, checking whether $\text{rank}_+(A(\theta)) = 3$ can be done in polynomial time (cf. Theorem 3.4). However, we do not know how to compute a solution θ in polynomial time (we actually don't even know if there always exists a solution that is polynomially bounded in the size of the input matrix M, see also [148]).

Higher Rank Matrices

For a rank-four matrix, RNR reduces to a three-dimensional problem of finding a polytope T with the minimum number of vertices nested between two other polytopes $S \subseteq P$. This problem has been studied by Das et al. [42, 44] and has been shown to be $\mathcal{NP}$-hard when minimizing the number of facets of T (the reduction is from *planar-3SAT*). From this result, one can deduce using a duality argument[9] that minimizing the number of vertices of T is $\mathcal{NP}$-hard as well [36, 43].

Theorem 3.6. *For* $\text{rank}(M) \geq 4$*, RNR is* $\mathcal{NP}$*-hard.*

[9] Taking the polar of the three nested polytopes exchanges the roles of the inner and outer polytopes, and transforms facet descriptions into vertex descriptions, so that the description of the inner and outer polytopes is unchanged but the intermediate polytope is now described by its facets. This will be proved in Section 3.6.2.

Proof. This is a consequence of Theorem 3.2 and the $\mathcal{NP}$-hardness results of Das et al. [42–44]. □

Note however that several approximation algorithms have been proposed in the literature. For example, Mitchell and Suri [121] approximate $\text{rank}_+^*(M)$ in case $\text{rank}(M) = 4$ within a $\mathcal{O}(\log(p))$ factor, where p is the total number of vertices of the given polygons S and P. Clarkson [36] proposes a randomized algorithm finding a polytope T with at most $r_+^* \mathcal{O}(5d \ln(r_+^*))$ vertices and running in $\mathcal{O}({r_+^*}^2 p^{1+\delta})$ expected time (with $r_+^* = \text{rank}_+^*(M)$, $d = \text{rank}(M) - 1$ and δ is any fixed value > 0).

3.3.3 Some Properties

In this section, we derive some useful properties of the restricted nonnegative rank.

Example 3.1. Construct M using the following NPP instance: P is the three dimensional cube $P = \{x \in \mathbb{R}^3 \mid 0 \leq x_i \leq 1,\ 1 \leq i \leq 3\}$, with 6 facets and S is the set of its 8 vertices $S = \{x \in \mathbb{R}^3 \mid x_i \in \{0, 1\},\ 1 \leq i \leq 3\}$. By construction, the convex hull of S is equal to P and the unique and optimal solution to this NPP instance is $T = P = \text{conv}(S)$ with 8 vertices. By Theorem 3.2, the corresponding matrix M of the RNR instance

$$M = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

has restricted nonnegative rank equal to 8 (note that its rank is 4 and its nonnegative rank is 6, see Section 3.4).

It is well-known that for a matrix $M \in \mathbb{R}_+^{m \times n}$, we have $\text{rank}_+(M) \leq \min(m, n)$; surprisingly, this does not hold for the restricted nonnegative rank.

Lemma 3.7. *For $M \in \mathbb{R}_+^{m \times n}$,*

$$\text{rank}_+^*(M) \leq n \quad \textit{but} \quad \text{rank}_+^*(M) \not\leq m.$$

Proof. The first inequality is trivial since $M = MI$ (I being the identity matrix). Example 3.1 provides an example when $\text{rank}_+^*(M) = 8$ for a 6-by-8 matrix M. □

Lemma 3.7 implies that in general $\text{rank}_+^*(M) \neq \text{rank}_+^*(M^T)$, unlike the rank and nonnegative rank [37]. Note however that when $\text{rank}(M) \leq 3$, the corresponding NPP instance is two-dimensional or lower, and we have

$$\text{rank}_+^*(M) \leq \min(m, n).$$

In fact, the number of vertices of the outer polygon P in the NPP instance is smaller or equal to its number of facets m in the two-dimensional case or lower, and that the solution $T = P$ is always feasible.

Lemma 3.8. *Let $A \in \mathbb{R}_+^{m\times n}$ and $B \in \mathbb{R}_+^{m\times r}$, then*

$$\text{rank}_+^*([A\,B]) \leq \text{rank}_+^*(A) + \text{rank}_+^*(B).$$

Proof. Let (U_a, V_a) and (U_b, V_b) be solutions of RNR for A and B respectively, then

$$[A\,B] = [U_a\,U_b] \begin{bmatrix} V_a & 0 \\ 0 & V_b \end{bmatrix},$$

and $\text{rank}([U_a\,U_b]) = \text{rank}([A\,B])$ since $\text{col}(U_a) = \text{col}(A)$ and $\text{col}(U_b) = \text{col}(B)$ by definition. □

Lemma 3.9. *Let $M \in \mathbb{R}_+^{m\times n}$ with $\text{rank}(M) = r$ and $\text{rank}_+(M) = r_+$, $U \in \mathbb{R}_+^{m\times r_+}$ and $V \in \mathbb{R}_+^{r_+\times n}$ with $M = UV$. Then*

$$r_+ < \text{rank}_+^*(M) \quad \Rightarrow \quad r - 1 \leq \text{rank}(U) \leq r_+ \;\; \textit{and} \;\; r \leq \text{rank}(V) \leq r_+ - 1.$$

Moreover, if M is symmetric,

$$r_+ < \text{rank}_+^*(M) \quad \Rightarrow \quad r-1 \leq \text{rank}(U) \leq r_+-1 \;\; \textit{and} \;\; r-1 \leq \text{rank}(V) \leq r_+-1.$$

Proof. Clearly,

$$r \leq \text{rank}(U) \leq r_+ \quad \text{and} \quad r \leq \text{rank}(V) \leq r_+.$$

If $\text{rank}(U) = r$, we would have $\text{rank}_+^*(M) = r_+$ which is a contradiction, and $\text{rank}(V) = r_+$ would imply that V has a right pseudo-inverse $V^\dagger$, so that we could write $U = MV^\dagger$ and then $r \leq \text{rank}(U) \leq \min(r, \text{rank}(V^\dagger)) \leq r$, a contradiction for the same reason.

In case M is symmetric, to show that $\text{rank}(U) < r_+$ and $\text{rank}(V) > r$, we use symmetry and observe that $UV = M = M^T = V^T U^T$. □

Corollary 3.10. *Given a nonnegative matrix M,*

$$\text{rank}_+^*(M) \leq \text{rank}(M) + 1 \quad \Rightarrow \quad \text{rank}_+(M) = \text{rank}_+^*(M).$$

If M is symmetric,

$$\text{rank}_+^*(M) \leq \text{rank}(M) + 2 \quad \Rightarrow \quad \text{rank}_+(M) = \text{rank}_+^*(M).$$

Proof. Let $r = \operatorname{rank}(M)$, $r_+ = \operatorname{rank}_+(M)$, and $U \in \mathbb{R}_+^{m \times r_+}$ and $V \in \mathbb{R}_+^{r_+ \times n}$ such that $M = UV$. If $r_+ < \operatorname{rank}_+^*(M)$, by Lemma 3.9, we have

$$r < \operatorname{rank}(U) \leq r_+ < \operatorname{rank}_+^*(M),$$

which is a contradiction if $\operatorname{rank}_+^*(M) \leq r + 1$. If M is symmetric, we have $\operatorname{rank}(U) < r_+$ and the above equation is a contradiction if $\operatorname{rank}_+^*(M) \leq r + 2$. □

For example, this implies that to find a symmetric rank-three nonnegative matrix with $\operatorname{rank}_+(M) < \operatorname{rank}_+^*(M)$, we need $\operatorname{rank}_+^*(M) > \operatorname{rank}(M) + 2 = 5$ and therefore have to consider matrices of size at least 6-by-6 with $\operatorname{rank}_+^*(M) = 6$.

Example 3.2. Let us consider the following matrix M and the rank-5 nonnegative factorization (U, V),

$$M = \begin{pmatrix} 0 & 1 & 4 & 9 & 16 & 25 \\ 1 & 0 & 1 & 4 & 9 & 16 \\ 4 & 1 & 0 & 1 & 4 & 9 \\ 9 & 4 & 1 & 0 & 1 & 4 \\ 16 & 9 & 4 & 1 & 0 & 1 \\ 25 & 16 & 9 & 4 & 1 & 0 \end{pmatrix} = UV,$$

with

$$U = \begin{pmatrix} 5 & 0 & 4 & 0 & 1 \\ 3 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 4 & 1 \\ 0 & 1 & 0 & 4 & 1 \\ 0 & 3 & 1 & 1 & 0 \\ 0 & 5 & 4 & 0 & 1 \end{pmatrix}, \text{ and } V = \begin{pmatrix} 0 & 0 & 0 & 1 & 3 & 5 \\ 5 & 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

One can check that $\operatorname{rank}(M) = 3$, and, using the algorithm of Aggarwal et al. [1], the restricted nonnegative rank can be computed[10] and is equal to 6. Using the above decomposition, it is clear that $\operatorname{rank}_+(M) \leq 5 < \operatorname{rank}_+^*(M) = 6$. By Lemma 3.9, for any $\operatorname{rank}_+(M)$-nonnegative factorization (U, V) of M, we then must have $3 < \operatorname{rank}(U) = 4 < \operatorname{rank}_+(M)$ implying that $\operatorname{rank}_+(M) = 5$.

As we have already seen with Lemma 3.7, the restricted nonnegative rank does not share all the nice properties of the rank and the nonnegative rank functions [37]. The next two lemmas exploit Example 3.2 further to show different behavior between nonnegative rank and restricted nonnegative rank.

[10]The problem is actually trivial because each vertex of the inner polygon S is located on a different edge of the polygon P, so that they define with the boundary of P 6 disjoint polygons tangent to S. This implies that $\operatorname{rank}_+^*(M) = 6$, cf. Section 3.3.2. This matrix is actually a linear Euclidean distance matrix which will be analyzed later in Section 3.5.2.

Lemma 3.11. *Let $A \in \mathbb{R}_+^{m\times n}$ and $B \in \mathbb{R}_+^{m\times r}$, then*

$$\operatorname{rank}_+^*(A+B) \not\leq \operatorname{rank}_+^*(A) + \operatorname{rank}_+^*(B).$$

Proof. Take M, U and V from Example 3.2 and construct $A = U(:, 1{:}3)V(1{:}3, :)$ with $\operatorname{rank}_+^*(A) = 3$ (since $\operatorname{rank}(A) = 3$), $B = U(:, 4{:}5)V(4{:}5, :)$ with $\operatorname{rank}_+^*(B) = 2$ (trivial) and $\operatorname{rank}_+^*(A+B) = 6$ since $A + B = M$. □

Lemma 3.12. *Let $A \in \mathbb{R}_+^{m\times n}$ and $B \in \mathbb{R}_+^{m\times r}$, then*

$$\operatorname{rank}_+^*([A\,B]) \not\geq \operatorname{rank}_+^*(A).$$

where $[A\,B] \in \mathbb{R}_+^{m\times(n+r)}$ denotes the concatenation of the columns of A and B.

Proof. Let us take M, U and V from Example 3.2, and construct $A = M$ and $B = U(:, 1)$ with $\operatorname{rank}_+^*([A\,B]) \leq 5$ since $\operatorname{rank}([A\,B]) = 4$ (this can be checked easily) and $[A\,B] = U[V\,e_1]$ with $\operatorname{rank}(U) = 4$ (where e_i denotes the i^{th} column of the identity matrix of appropriate dimension). □

Lemma 3.13. *Let $B \in \mathbb{R}_+^{m\times r}$ and $C \in \mathbb{R}_+^{r\times n}$, then*

$$\operatorname{rank}_+^*(BC) \not\leq \min(\operatorname{rank}_+^*(B), \operatorname{rank}_+^*(C)).$$

Proof. See Example 3.2 where $\operatorname{rank}_+^*(U) \leq 5$ by Lemma 3.7 and $\operatorname{rank}_+^*(M) = 6$. □

3.4 Lower Bounds for the Nonnegative Rank

In this section, we provide new lower bounds for the nonnegative rank based on the restricted nonnegative rank. Recall that the restricted nonnegative rank already provides an upper bound for the nonnegative rank since for an m-by-n nonnegative matrix M,

$$0 \leq \operatorname{rank}(M) \leq \operatorname{rank}_+(M) \leq \operatorname{rank}_+^*(M) \leq n.$$

Notice that this bound can in general only be computed in polynomial time in the case $\operatorname{rank}(M) = 3$ (unless $\mathcal{P} = \mathcal{NP}$, see Theorems 3.3 and 3.6).

As mentioned in the introduction, it might also be interesting to compute lower bounds on the nonnegative rank. Some work has already been done in this direction, including the following

1. Let $M \in \mathbb{R}_+^{m\times n}$ be any weighted biadjacency matrix of a bipartite graph $G = (V_1 \cup V_2, E \subseteq V_1 \times V_2)$ with $M(i,j) > 0 \iff (V_1(i), V_2(j)) \in E$. A biclique of G is a complete bipartite subgraph (it corresponds to a positive rectangular submatrix of M). One can easily check that each rank-one

factor $(U_{:k}, V_{k:})$ of any rank-k nonnegative factorization (U, V) of M can be interpreted as a biclique of M (i.e., as a positive rectangular submatrix) since $M = \sum_{i=1}^{k} U_{:i}V_{i:}$. Moreover, these bicliques $(U_{:k}, V_{k:})$ must cover G completely since $M = UV$. The minimum number of bicliques needed to cover G is then a lower bound for the nonnegative rank. It is called the biclique partition number and denoted $b(G)$, see [150] and the references therein. Its computation is $\mathcal{NP}$-complete [123] and is directly related to the minimum biclique cover problem (MBC).
Consider for example the matrix M from Example 3.1. The largest biclique of the graph G generated by M has 4 edges[11]. Since G has 24 edges, we have $b(G) \geq \frac{24}{4} = 6$ and therefore $6 \leq \text{rank}_+(M) \leq \min(m, n) = 6$.

A crown graph G is a bipartite graph with $|V_1| = |V_2| = n$ and $E = \{(V_1(i), V_2(j)) \,|\, i \neq j\}$ (it can be viewed as a biclique where the horizontal edges have been removed). de Caen, Gregory and Pullman [49] showed that

$$b(G) = \min_k \left\{ k \mid n \leq \binom{k}{\lfloor k/2 \rfloor} \right\} = \Theta(\log n).$$

Beasley and Laffey [6] studied linear Euclidean distance matrices defined as $M(i, j) = (a_i - a_j)^2$ for $1 \leq i, j \leq n$, $a_i \in \mathbb{R}$, $a_i \neq a_j$ $i \neq j$. They proved that such matrices have rank three and that

$$\min_k \left\{ k \mid n \leq \binom{k}{\lfloor k/2 \rfloor} \right\} \leq r_+ \quad \text{which means} \quad n \leq \binom{r_+}{\lfloor r_+/2 \rfloor}, \tag{3.3}$$

where $r_+ = \text{rank}_+(M)$. In fact, such matrices are biadjacency matrices of crown graphs (only the diagonal entries are equal to zero).

2. Goemans makes [79] the following observation: the product UV of two nonnegative matrices $U \in \mathbb{R}_+^{m \times k}$ and $V \in \mathbb{R}_+^{k \times n}$ generates a matrix M with at most 2^k columns (resp. 2^k rows) with different sparsity patterns. In fact, the columns (resp. rows) of M are additive linear combinations of the k columns of U (resp. rows of V) and therefore no more than 2^k sparsity patterns can be generated from these columns (resp. rows). Letting s_p be the maximum between the number of columns and rows of $M \in \mathbb{R}_+^{m \times n}$ having a different sparsity pattern, we then have

$$\text{rank}_+(M) \geq \log_2(s_p).$$

In particular, if all the columns and rows of M have a different sparsity pattern, then

$$\text{rank}_+(M) \geq \log_2(\max(m, n)). \tag{3.4}$$

[11] This can be computed explicitly, e.g., with a brute force approach. Note however that finding the biclique with the maximum number of edges is a combinatorial $\mathcal{NP}$-hard optimization problem [127]. It is closely related to a variant of the approximate nonnegative factorization problem, see Chapter 5.

Goemans then uses this result to show that any extended formulation of the permutahedron in dimension n must have $\Omega(n\log(n))$ variables and constraints. In fact,

- ◇ The minimal size s is of the order of the nonnnegative rank of its slack matrix plus n (cf. Section 3.2).
- ◇ The slack matrix has $n!$ columns (corresponding to each vertex of the polytope) with different sparsity patterns (cf. Equation (3.1)).

This implies that

$$s = \Theta(\text{rank}_+(S_M) + n) \geq \Theta(\log(n!)) = \Theta(n\log(n)).$$

In this section, we provide some theoretical results linking the restricted nonnegative rank with the nonnegative rank, which allow us to improve and generalize the above results in Section 3.5 for both slack and linear Euclidean distance matrices.

3.4.1 Geometric Interpretation of a Nonnegative Factorization as a Nested Polytopes Problem

In the following, we lay the groundwork for the main results of this chapter, introducing essential notations and observations that will be extensively used in this section. We rely on the geometric interpretation of the nonnegative rank, see also [31, 59, 148] where related results are presented. The main observation is that any rank-k nonnegative factorization (U, V) of a nonnegative matrix M can be interpreted as the solution with k vertices of a nested polytopes problem in which the inner polytope has dimension $\text{rank}(M) - 1$ and the outer polytope has dimension $\text{rank}(U) - 1$.

Without loss of generality, let $M \in \mathbb{R}_+^{m\times n}$, $U \in \mathbb{R}_+^{m\times k}$ and $V \in \mathbb{R}_+^{k\times n}$ be column stochastic with $M = UV$ (cf. proof of Theorem 3.2, the columns of M are convex combination of the columns of U). If the column space of U does not coincide with the column space of M, i.e., $r_u = \text{rank}(U) > \text{rank}(M) = r$, it means that the columns of U belong to a higher dimensional affine subspace containing the columns of M (otherwise, see Theorem 3.2).

Let us factorize $U = AB$ where $A \in \mathbb{R}^{m\times r_u}$ and $B \in \mathbb{R}^{r_u\times r_+}$ are full rank and their columns sum to one. As in Theorem 3.2, we can construct the polytope of the coefficients of the linear combinations of the columns of A that generate stochastic vectors. It is defined as

$$P_u = \{x \in \mathbb{R}^{r_u-1} \mid f_u(x) = A(:, 1{:}r_u{-}1)x + \Big(1 - \sum_{i=1}^{r_u-1} x_i\Big)A(:, r_u) \geq 0\}.$$

Since $\text{col}(M) \subseteq \text{col}(U)$, there exists $B' \in \mathbb{R}^{r_u \times n}$ whose columns must sum to one such that $M = AB'$. Since $\text{rank}(M) = r$ and A is full rank, we must have $\text{rank}(B') = r$. By construction, the columns of $B_u = B(1{:}r_u{-}1, :)$ (corresponding to the columns of U) and $B_m = B'(1{:}r_u{-}1, :)$ (corresponding to the columns of M) belong to P_u. Note that since $\text{rank}(B') = r$, the columns of B_m live in a lower $(r-1)$-dimensional polytope

$$P_m = \{x \in \mathbb{R}^{r_u - 1} \mid f_u(x) \geq 0, f_u(x) \in \text{col}(M)\} \quad \subseteq \quad P_u.$$

Polytope P_m contains the points in P_u generating vectors in the column space of M.

Moreover

$$M = AB' = UV = ABV,$$

implying that (since A is full rank)

$$B' = BV \quad \text{and} \quad B_m = B_u V.$$

Finally, the columns of B_m are contained in the convex hull of the columns of B_u, inside P_u, i.e.,

$$\text{conv}(B_m) \subseteq \text{conv}(B_u) \subseteq P_u.$$

Defining the polytope T as the convex hull of the columns of B_u, and the set of points S as the columns of B_m, we can then interpret the nonnegative factorization (U, V) of M as follows. The $(r_u - 1)$-dimensional polytope T with k vertices (corresponding to the columns of U) is nested between an inner $(r-1)$-dimensional polytope $\text{conv}(S)$ (where each point in S corresponds to a column of M) and a outer $(r_u - 1)$-dimensional polytope P_u.

Let us use the matrix M and its nonnegative factorization (U, V) of Example 3.2 as an illustration: $\text{rank}(M) = 3$ so that P_m is a two-dimensional polytope and contains the set of points S, while $\text{rank}(U) = 4$ and defines a three-dimensional polytope T containing S, see Figure 3.3.

3.4.2 Upper Bound for the Restricted Nonnegative Rank

From the geometric interpretation introduced in the previous paragraph, we can now give the main result of this section. The idea is the following: using notations of Section 3.4.1, we know that

- ◇ The polytope T (whose vertices correspond to the columns of U) contains the (lower dimensional) set of points S (corresponding to the columns of M), and that
- ◇ The polytope S is contained in P_m (which corresponds to the set of stochastic vectors in the column space of M).

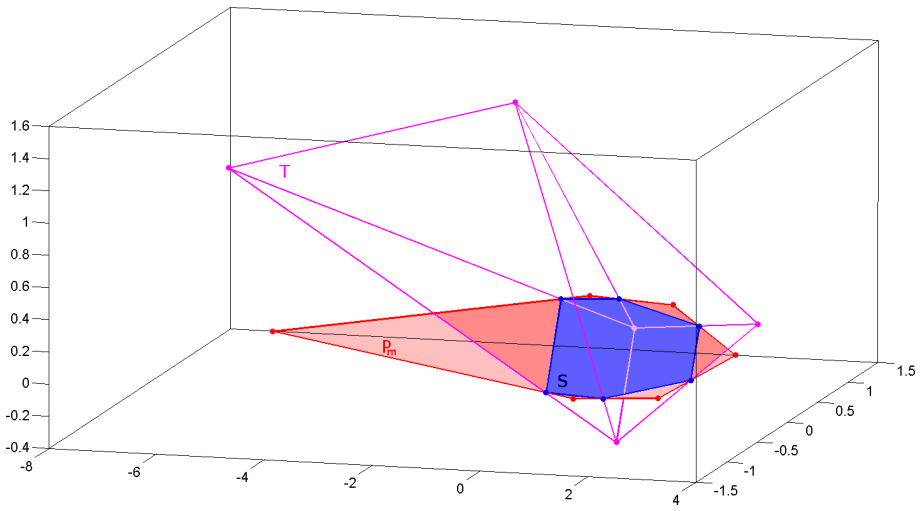

Figure 3.3: Illustration of the solution from Example 3.2 as a nested polytopes problem, with $\text{rank}(M) = 3 < \text{rank}(U) = 4 < \text{rank}_+(M) = 5 < \text{rank}_+^*(M) = 6 = n$.

Therefore, the intersection between T and P_m must also contain S, i.e., the intersection $T \cap P_m$ defines a polytope which is contained in the column space of M, and contains S. Hence its vertices provide a feasible solution to the RNR problem, and an upper bound for the restricted nonnegative rank can then be computed.

In other words, any nonnegative factorization (U, V) of a nonnegative matrix M can be used to construct a feasible solution to the restricted nonnegative rank problem. One has 'simply' to compute the intersection of the polytope generated by the (normalized) columns of U with the column space of M (which can obviously increase the number of vertices).

Theorem 3.14. *Using notations of Section 3.4.1, we have*

$$\text{rank}_+^*(M) \leq \#\,\text{vertices}(T \cap P_m), \tag{3.5}$$

where $\#\,\text{vertices}(Q)$ *denotes the number of vertices of* Q.

Proof. Let $x_1, x_2, \ldots, x_v$ be the v vertices of $T \cap P_m$ and note $X = [x_1\, x_2\, \ldots x_v]$ which has rank at most r (since it is contained in the $(r-1)$-dimensional polyhedron P_m). By construction,

$$B_m(:, j) \;\in\; T \cap P_m = \text{conv}(X) \quad 1 \leq j \leq n.$$

Therefore, there must exist a matrix $V^* \in \mathbb{R}^{v \times n}$ column stochastic such that

$$B_m = XV^*,$$

implying that

$$M = f_u(B_m) = f_u(XV^*) = f_u(X)V^* = U^*V^*,$$

where $U^* = f_u(X) \in \mathbb{R}^{m \times v}$ is nonnegative since $x_i \in P_m \subseteq P_u \, \forall i$, and U^* has rank r since $M = U^*V^*$ implies that its rank is at least r and $U^* = f_u(X)$ that it is at most r. The pair (U^*, V^*) is then a feasible solution of the corresponding RNR problem for M and therefore $\text{rank}_+^*(M) \leq v = \#\,\text{vertices}(T \cap P_m)$. □

3.4.3 Lower Bound for the Nonnegative Rank based on the Restricted Nonnegative Rank

We can now obtain a lower bound for the nonnegative rank based on the restricted nonnegative rank. Indeed, if we consider an upper bound on the quantity $\#\,\text{vertices}(T \cap P_m)$ that increases with the nonnegative rank (i.e., the number of vertices of T), we can reinterpret Theorem 3.14 as providing a lower bound on the nonnegative rank. For that purpose, define the quantity $\text{faces}(n, d, k)$ to be the maximal number of k-faces of a polytope with n vertices in dimension d.

Theorem 3.15. *The restricted nonnegative rank of a nonnegative matrix M with $r = \text{rank}(M)$ and $r_+ = \text{rank}_+(M)$ can be bounded above by*

$$\text{rank}_+^*(M) \leq \max_{r \leq r_u \leq r_+} \text{faces}(r_+, r_u - 1, r_u - r). \tag{3.6}$$

Proof. Let (U, V) be a rank-r_+ nonnegative factorization of M with $\text{rank}(U) = r_u$. Using notations of Section 3.4.1 and the result of Theorem 3.14, $\text{rank}_+^*(M)$ is bounded above by the the number of vertices of $T \cap P_m$. Defining $Q_m = \{x \in \mathbb{R}^{r_u - 1} \mid f_u(x) \in \text{col}(M)\}$, we have $P_m = Q_m \cap P_u$ and since $T \subseteq P_u$,

$$P_m \cap T = Q_m \cap P_u \cap T = Q_m \cap T.$$

Since Q_m is $(r-1)$-dimensional, the number of vertices of $T \cap Q_m$ is bounded above by the number of $(r_u - r)$-faces of T (in a $(r_u - 1)$-dimensional space, $(r_u - r)$-faces are defined by $r - 1$ equalities), we then have

$$\text{rank}_+^*(M) \leq \#\,\text{vertices}(T \cap P_m) = \#\,\text{vertices}(T \cap Q_m) \leq \text{faces}(r_+, r_u - 1, r_u - r).$$

Notice that for $r_u = r$, $\text{faces}(r_+, r-1, 0) = r_+$ which gives $r_+ = \text{rank}_+^*(M)$ as expected. Finally, taking the maximum over all possible values of $r \leq r_u \leq r_+$ gives the above bound (3.6). □

We introduce for easier reference a function ϕ corresponding to the upper bound in Theorem 3.15, i.e.,

$$\phi(r, r_+) = \max_{r \le r_u \le r_+} \text{faces}(r_+, r_u - 1, r_u - r).$$

Clearly, when r is fixed, ϕ is an increasing function of its second argument r_+, since $\text{faces}(n, d, k)$ increases with n. Therefore inequality $\text{rank}_+^*(M) \le \phi(r, r_+)$ from Theorem 3.15 implicitly provides a lower bound on the nonnegative rank r_+ that depends on both rank r and restricted nonnegative rank $\text{rank}_+^*(M)$.

Explicit values for function ϕ can be computed using a tight bound for $\text{faces}(n, d, k)$ attained by cyclic polytopes [157, p.257, Corollary 8.28]

$$\text{faces}(n, d, k-1) = \sum_{i=0}^{\frac{d}{2}} {}^* \left(\binom{d-i}{k-i} + \binom{i}{k-d+i} \right) \binom{n-d-1+i}{i},$$

where $\sum^*$ denotes a sum where only half of the last term is taken for $i = \frac{d}{2}$ if d is even, and the whole last term is taken for $i = \lfloor \frac{d}{2} \rfloor = \frac{d-1}{2}$ if d is odd. Alternatively, simpler versions of the bound can be worked out in the following way:

Theorem 3.16. *The upper bound $\phi(r, r_+)$ on the restricted nonnegative rank of a nonnegative matrix M with $r = \text{rank}(M)$ and $r_+ = \text{rank}_+(M)$ satisfies*

$$\begin{aligned} \phi(r, r_+) &= \max_{r \le r_u \le r_+} \text{faces}(r_+, r_u - 1, r_u - r) \\ &\le \max_{r \le r_u \le r_+} \binom{r_+}{r_u - r + 1} \le \binom{r_+}{\lfloor r_+/2 \rfloor} \le 2^{r_+} \sqrt{\frac{2}{\pi r_+}} \le 2^{r_+} . \end{aligned}$$

Proof. The first inequality follows from the fact that $\text{faces}(n, d, k-1) \le \binom{n}{k}$, since any set of k distinct vertices defines at most one $(k-1)$-face. The second follows from the maximality of central binomial coefficients. The third is a standard upper bound on central binomial coefficients, and the fourth is an even cruder upper bound. □

We will see in Section 4 that some of these weaker bounds correspond to existing results from the literature.

When matrix M is symmetric, the bound can be slightly strengthened, leading to a different function ϕ':

Corollary 3.17. *Given a symmetric matrix M with $r_+ = \text{rank}_+(M)$, $r = \text{rank}(M)$ and $r_+ \ge r + 1$, we have*

$$\text{rank}_+^*(M) \le \max_{r \le r_u \le r_+ - 1} \text{faces}(r_+, r_u - 1, r_u - r) = \phi'(r, r_+) \le \phi(r, r_+).$$

Proof. We have seen in Lemma 3.9 that for symmetric matrices $r_u = r_+$ implies $\text{rank}_+^*(M) = r_+$. Therefore, in case $r_+ \ge r + 1$, one can strengthen the result of Theorem 3.15 and only consider the range $r \le r_u \le r_+ - 1$. □

Improvements in the rank-three case

It is possible to improve the above bound by finding better upper bounds for $\#\,\text{vertices}(T \cap P_m)$ in Equation (3.5). For example, since two-dimensional polytopes (i.e., polygons) have the same number of vertices (0-faces) and edges (1-faces), we have for $\text{rank}(M) = 3$ that

$$\#\,\text{vertices}(T \cap P_m) \quad = \quad \#\text{edges}(T \cap P_m).$$

Using the same argument as in Theorem 3.15, the number of edges of $T \cap P_m$ is bounded above by the number of $(r_u - r + 1)$-faces of T (defined by $r - 2$ equalities) leading to

Corollary 3.18. *The restricted nonnegative rank of a rank-three nonnegative matrix M with $r_+ = \text{rank}_+(M)$ can be bounded above with*

$$\text{rank}_+^*(M) \leq \max_{3 \leq r_u \leq r_+} \min_{i=0,1} \text{faces}(r_+, r_u - 1, r_u - 3 + i) \leq \phi(3, r_+). \qquad (3.7)$$

The minimum taken between 0 and 1 simply accounts for the two possible cases, i.e., the bound based on $\#\,\text{vertices}(T \cap P_m)$ with $i = 0$ as in Theorem 3.15, or based on $\#\text{edges}(T \cap P_m)$ with $i = 1$. A similar bound holds in the symmetric case.

3.5 Applications: Slack and Linear Euclidean Distance Matrices

So far, we have not provided explicit lower bounds for the nonnegative rank. As we have seen, inequalities (3.6) and (3.7) can be interpreted as implicit lower bounds on the nonnegative rank r_+, but have the drawback of depending on the restricted nonnegative rank, which cannot be computed in polynomial time unless the rank of the matrix is smaller than 3 or $\mathcal{P} = \mathcal{NP}$, see Theorems 3.3 and 3.6.

Nevertheless, we provide in this Section explicit lower bounds for the nonnegative rank of slack matrices (Section 3.5.1) and linear Euclidean distance matrices (Section 3.5.2), cf. introduction of Section 3.4. These bounds are derived by showing that the restricted nonnegative rank of such matrices is maximum, i.e., it is equal to the number of columns of these matrices (cf. Lemma 3.7).

3.5.1 Slack Matrices

Let us start with a simple observation: it is easy to construct a $m \times n$ matrix of rank $r < \min(m, n)$ with maximum restricted nonnegative rank n:

1. Take any $(r-1)$-dimensional polytope P with $n \geq r+1$ vertices.
2. Construct a NPP instance with $S = \text{vertices}(P)$.
3. Compute the corresponding matrix M in the equivalent RNR instance.

Clearly, the unique solution for NPP is $T = P = \text{conv}(S)$ and therefore the matrix M in the corresponding RNR instance must satisfy: $\text{rank}_+^*(M) = \#\,\text{vertices}(T) = n$; see Example 3.1 for an illustration with the three-dimensional cube.

Remark 3.4. The matrices constructed as described above also satisfy

$$\text{rank}(M) < \text{rank}_+(M).$$

Otherwise $\text{rank}_+^*(M) = \text{rank}_+(M) = \text{rank}(M) < \min(m,n)$ which is a contradiction. This is interesting because it is nontrivial to construct matrices with $\text{rank}(M) < \text{rank}_+(M)$ [114]. In fact, it is easy to check that generating randomly two nonnegative matrices U and V of dimensions $m \times r$ and $r \times n$ respectively, and constructing $M = UV$ will generate a matrix M of rank r with probability one.

In the context of compact formulations, the aim is to express a polytope Q with fewer constraints by using some additional variables, i.e., find a *lifting* of polynomial size. A possible way to do that is to compute a nonnegative factorization of the slack matrix S_M of Q [152] (see Equation (3.1)). The next theorem states that the restricted nonnegative rank of any slack matrix $S_M \in \mathbb{R}_+^{f \times v}$ is maximum (f is the number of facets of Q, v its number of vertices), i.e., $\text{rank}_+^*(S_M) = v$. This is directly related to the above observation: the slack matrix of a polytope Q corresponds to a NPP instance where Q is the outer polytope and its vertices are the points defining the inner polytope. Notice that the restricted nonnegative rank used as an upper bound for the nonnegative rank is useless in this case.

Theorem 3.19. *Let $Q = \{x \in \mathbb{R}^q \,|\, Fx \geq h, Ex = g\}$ be a p-dimensional polytope with v vertices, $v > 1$, and let $S_M(Q)$ be its slack matrix, then $\text{rank}_+^*(S_M(Q)) = v$.*

Proof. In order to prove this result, we first construct a bijective transformation L between Q and a full-dimensional polytope $P \subseteq \mathbb{R}^p$. The vertices of P can then be easily constructed from the vertices of Q, which allows to show that P and Q share the same slack matrix. Finally, using the result of Theorem 3.2, we show that the slack matrix of P has maximum restricted nonnegative rank.

Since Q is a p-dimensional polytope, there exists a polytope $P \subseteq \mathbb{R}^p$ and a bijective affine transformation

$$L : Q \to P : x \to L(x) = Ax + b$$

with

$$L^{-1} : P \to Q : y \to L^{-1}(y) = A^\dagger y - A^\dagger b,$$

such that $P = L(Q)$ and $Q = L^{-1}(P)$ (where $A \in \mathbb{R}^{p\times q}$ has full rank, $A^\dagger \in \mathbb{R}^{q\times p}$ is its right inverse and $b \in \mathbb{R}^p$).
By construction,

$$\begin{aligned} P &= \{y \in \mathbb{R}^p \,|\, y = L(x), x \in Q\} = \{y \in \mathbb{R}^p \,|\, L^{-1}(y) \in Q\}, \\ &= \{y \in \mathbb{R}^p \,|\, FL^{-1}(y) \geq h, EL^{-1}(y) = g\}, \\ &= \{y \in \mathbb{R}^p \,|\, FA^\dagger y \geq h + FA^\dagger b\}, \end{aligned}$$

since the equalities $EL^{-1}(y) = g$ must be satisfied for all $y \in \mathbb{R}^p$ since P is full-dimensional.

Noting $C = FA^\dagger$ and $d = h + FA^\dagger b$, we have $P = \{y \in \mathbb{R}^q \,|\, Cy \geq d\}$. Finally, we observe that

1. Noting v_i's the v vertices of Q, we have that $L(v_i)$'s define the v vertices of P. This can easily be checked since L is bijective ($\forall y \in P, \exists! x \in Q$ s.t. $y = L(x)$ and vice versa).

2. P can be taken as the outer polytope of a NPP instance, i.e., P is bounded and $(C\ d)$ is full rank. P is bounded since Q is. C is full rank because P has at least one vertex ($v > 1$). If $(C\ d)$ was not full rank, then $\exists z \in \mathbb{R}^p$ such that $d = Cz$, implying that $z \in P$. Since P has at least two vertices ($v > 1$), $\exists y \in P$ with $y \neq z$, and one can check that $y + \alpha(y - z) \in P$ $\forall \alpha \geq 0$. This is a contradiction because P is bounded.

3. The slack matrix of P is equal to the slack matrix of Q:

$$\begin{aligned} S_M(P) &= CL(V) - [d \ldots d] = FA^\dagger L(V) - [h + FA^\dagger b \ldots h + FA^\dagger b] \\ &= F(A^\dagger L(V) - [A^\dagger b \ldots A^\dagger b]) - [h \ldots h] \\ &= FL^{-1}(L(V)) - [h \ldots h] = FV - [h \ldots h] \\ &= S_M(Q), \end{aligned}$$

   where $V = [v_1\, v_2 \ldots v_v]$ is the matrix whose columns are the vertices of Q, and $L(V) = [L(v_1)\, L(v_2) \ldots L(v_v)]$ is the matrix whose columns are the vertices of P.

4. The NPP instance with P as the outer polytope and its v vertices $L(v_i)$'s as the set of points S defining the inner polytope has a unique and optimal solution $T = P = \text{conv}(S)$ with v vertices. The matrix M in the RNR instance corresponding to this NPP instance is given by the slack matrix $S_M(P)$ of P implying that its restricted nonnegative rank is equal to v (cf. Theorem 3.2).

We can conclude that $\text{rank}_+^*(S_M(Q)) = v$. □

We can now derive a lower bound on the nonnegative rank of a slack matrix and on the size of an extended formulation, by combining Theorem 3.15 (cf. Equation (3.6)), Theorem 3.16, Theorem 3.19 and the result of Yannakakis [152].

Corollary 3.20. *Let P be a polytope with v vertices and let $S_M \in \mathbb{R}_+^{f\times v}$ be its slack matrix of rank r (i.e., P has dimension $r-1$), then*

$$v \leq \phi(r, r_+) = \phi_r(r_+) \leq \max_{r\leq r_u\leq r_+} \binom{r_+}{r_u - r + 1} \leq \binom{r_+}{\lfloor r_+/2 \rfloor} \leq 2^{r_+}, \quad (3.8)$$

where $r_+ = \text{rank}_+(S_M)$. Therefore, the minimum size s of any extended formulation of P follows

$$s = \Theta(r_+ + n) \geq \Theta(\phi_r^{-1}(v)) \geq \Theta(\log_2(v)),$$

where $\phi_r^{-1}(\cdot)$ is the inverse of the nondecreasing function $\phi_r(\cdot) = \phi(r, \cdot)$.

The last bound 2^{r_+} from Equation (3.8) is the one of Goemans [79, Theorem 1] (see introduction of Section 3.4), and therefore Corollary 3.20 provides us with an improved lower bound, even though it is still in $\Omega(\log_2(v))$. It is actually not possible to provide a bound with a faster growth (i.e., without making additional hypothesis on the polytope P) since Goemans showed that the size of any LP formulation of the permutahedron (with $v = n!$ vertices) must be in $\mathcal{O}(n\log(n))$, this implies that the nonnegative rank of its slack matrix is in $\Theta(n\log(n))$. This result also implies that the gap between the nonnegative rank and the restricted nonnegative rank can be arbitrarily large.

3.5.2 Linear Euclidean Distance Matrices

Linear Euclidean distance matrices (linear EDM's) are defined by

$$M(i,j) = (a_i - a_j)^2, \quad 1 \leq i, j \leq n, \text{ for some } a \in \mathbb{R}^n. \quad (3.9)$$

In this section we assume $a_i \neq a_j$ $i \neq j$, so that these matrices have rank three. Linear EDM's were used in [6] to show that the nonnegative rank of a matrix with fixed rank (rank 3 in this case) can be made as large as desired (while increasing the size of the matrix), implying that an upper bound for the nonnegative rank of a matrix based only on the rank cannot exist.

We refer the reader to [102] and the references therein for detailed discussions about Euclidean distance matrices, and related applications.

Restricted Nonnegative Rank of Linear Euclidean Distance Matrices

We first show that the restricted nonnegative rank of linear EDM's is maximum, i.e., it is equal to their dimension n.

Definition 3.2. The columns of a matrix M have disjoint[12] sparsity patterns if and only if

$$\overline{\text{supp}}(M_{:i}) \not\subseteq \overline{\text{supp}}(M_{:j}), \quad \forall i \neq j,$$

where $\overline{\text{supp}}(M_{:i}) = \{k | M(k,i) = 0\}$ is the sparsity pattern of the i^{th} column of M (i.e., the complement of its support).

Theorem 3.21. *Let M be a rank-three nonnegative square matrix of dimension n whose columns have disjoint sparsity patterns, then*

$$\text{rank}_+^*(M) = n.$$

In particular, linear EDM's have this property.

Proof. Let P, S and T be the polygons defined in the two-dimensional NPP instance corresponding to the RNR instance of M (cf. Theorem 3.2). Aggarwal et al. [1] observe that if two points in S are on different edges of P, they define a polygon with the boundary of P (see each dark regions in Figure 3.4) which

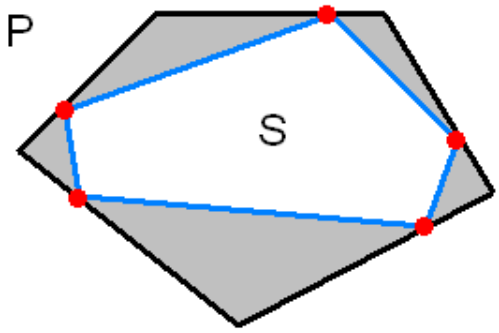

Figure 3.4: Illustration of the restricted nonnegative rank of a linear EDM of dimension 5. The solution T must contain a point in each dark region.

must contain a point of the solution T. Otherwise these two points could not be contained in T (see also Section 3.3.2). Therefore if each point of S is on a different edge of the boundary of P, any solution T to NPP must have at least $|S| = n$ vertices since S defines n disjoint polygons with the boundary of P.

[12]This definition is a little abusive since disjoint should refer to sets with an empty intersection.

Finally, two points x_1 and x_2 in S are on different edges of the boundary of the polytope $P = \{ x \in \mathbb{R}^2 \,|\, Cx+d \geq 0\}$ if and only if (Cx_1+d) and (Cx_2+d) have disjoint sparsity patterns or, equivalently, if and only if the two corresponding columns of M (which are precisely equal to $Cx_1 + d$ and $Cx_2 + d$) in the RNR instance have disjoint sparsity patterns. Indeed, for two vertices a and b to be located on different edges, one needs one inequality that is active at a and inactive at b and another inequality that is active at b and inactive at a. This is equivalent to requiring the sparsity patterns of the corresponding columns of the matrix M to be disjoint. □

Remark 3.5. This result does not hold for higher rank matrices. For example, the matrix

$$M = \begin{pmatrix} 0 & 1 & 4 & 9 & 16 & 25 \\ 2 & 0 & 1 & 4 & 9 & 16 \\ 8 & 1 & 0 & 1 & 4 & 9 \\ 13 & 4 & 1 & 0 & 1 & 4 \\ 17 & 9 & 4 & 1 & 0 & 1 \\ 25 & 16 & 9 & 4 & 1 & 0 \end{pmatrix} = UV,$$

with

$$U = \begin{pmatrix} 0 & 0 & 4 & 5 & 1 \\ 1 & 0 & 1 & 3 & 0 \\ 4 & 0 & 0 & 1 & 1 \\ 4 & 1 & 0 & 0 & 1 \\ 1 & 3 & 1 & 0 & 0 \\ 0 & 5 & 4 & 0 & 1 \end{pmatrix}, V = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 1 \\ 5 & 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 3 & 5 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix},$$

has $\operatorname{rank}(M) = 4$ and $\operatorname{rank}^*_+(M) \leq 5$ since $\operatorname{rank}(U) = 4$. Therefore we cannot conclude that higher dimensional Euclidean distance matrices have maximal restricted nonnegative rank.

Nonnegative Rank of Linear Euclidean Distance Matrices

Since linear EDM's are rank-three symmetric matrices, one can combine the results of Theorem 3.21 with Corollary 3.18 (cf. Equation (3.7)) and Corollary 3.17 in order to obtain lower bounds for the nonnegative rank of linear EDM's.

Corollary 3.22. *For any linear Euclidean distance matrix M, we have*

$$\begin{aligned} \operatorname{rank}^*_+(M) = n &\leq \max_{3 \leq r_u \leq r_+ - 1} \min_{i=0,1} \operatorname{faces}(r_+, r_u - 1, r_u - r + i) \\ &\leq \max_{3 \leq r_u \leq r_+ - 1} \operatorname{faces}(r_+, r_u - 1, r_u - r) = \phi'(r, r_+) \\ &\leq \binom{r_+}{\lfloor r_+/2 \rfloor} \\ &\leq 2^{r_+}. \end{aligned}$$

We observe that our results (first two inequalities above, from Theorem 3.15 and Corollary 3.18) strengthen the bounds from Equations (3.3) (Beasley and Laffey [6]) and (3.4) (Goemans [79]). Figure 3.5 displays the growth of the different bounds, and Table 3.1 compares the lower bounds on the nonnegative rank for small values of n. For example, for a linear EDM to be guaran-

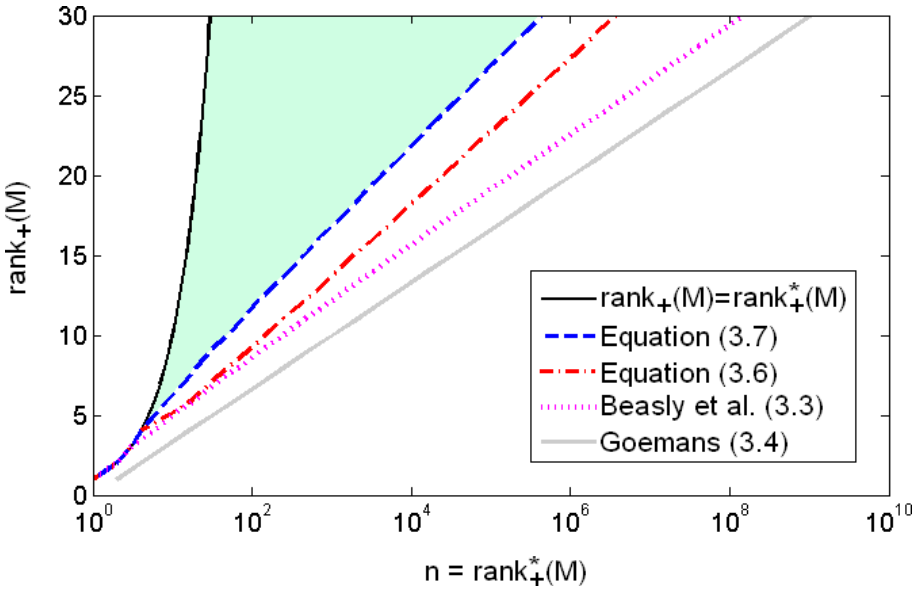

Figure 3.5: Comparison of the different bounds for symmetric n-by-n matrices, with $\text{rank}_+^*(M) = n$.

dimension n	4	5	6	7	8	9	10
Equation (3.7)	4	5	5	6	6	6	7
Equation (3.6)	4	5	5	5	5	5	6
Beasly and Laffey (3.3)	4	4	4	5	5	5	5
Goemans (3.4)	3	3	3	3	4	4	4

Table 3.1: Comparison of the lower bounds for the nonnegative rank of linear EDM's.

teed to have nonnegative rank 10, the bounds requires respectively $n = 50$ (3.7), $n = 150$ (3.6), $n = 252$ (3.3) and $n = 1024$ (3.4). This is a significant improvement, even though all the bounds are still of the same order with $r_+ = \Omega(\log(n))$.

Is it possible to further improve these bounds? Beasley and Laffey [6] conjectured that the nonnegative rank of linear EDM's is maximum, i.e., it is equal to their dimension. Lin and Chu [114, Theorem 3.1] first claimed to have proved that this equality always holds. However, Chu [30] has recently reported an error in the proof[13], and showed instead that the nonnegative rank of linear EDM's of dimension n is generically n (i.e., using a randomly generated vector a to construct the linear EDM M, we have that $\operatorname{rank}(M) = 3$ and $\operatorname{rank}_+(M) = n$ with probability one). Indeed, not all linear EDM's have maximum nonnegative rank because of the following example.

Example 3.3. Taking $M \in \mathbb{R}_+^{6\times 6}$ with

$$M(i,j) = (i-j)^2, \quad 1 \le i,j \le 6,$$

gives $\operatorname{rank}_+(M) = 5$. In fact,

$$\begin{aligned} M &= \begin{pmatrix} 0 & 1 & 4 & 9 & 16 & 25 \\ 1 & 0 & 1 & 4 & 9 & 16 \\ 4 & 1 & 0 & 1 & 4 & 9 \\ 9 & 4 & 1 & 0 & 1 & 4 \\ 16 & 9 & 4 & 1 & 0 & 1 \\ 25 & 16 & 9 & 4 & 1 & 0 \end{pmatrix} \\ &= \begin{pmatrix} 5 & 0 & 4 & 0 & 1 \\ 3 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 4 & 1 \\ 0 & 1 & 0 & 4 & 1 \\ 0 & 3 & 1 & 1 & 0 \\ 0 & 5 & 4 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 1 & 3 & 5 \\ 5 & 3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix}, \\ &= \begin{pmatrix} 5 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 3 & 0 & 1 & 0 \\ 0 & 5 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 1 & 3 & 5 \\ 5 & 3 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 4 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 4 & 1 & 0 & 0 & 1 & 4 \end{pmatrix}, \end{aligned} \tag{3.10}$$

so that $\operatorname{rank}_+(M) \le 5$, and $\operatorname{rank}_+(M) \ge 5$ is guaranteed by Equation (3.7), see Table 3.1 with $\operatorname{rank}_+^*(M) = n = 6$ (or by Lemma 3.9, see Example 3.2).

Example 3.3 proves that linear EDM's do not necessarily have a nonnegative rank equal to their dimension. In fact, we can even show that

[13] In their proof, they actually show that the restricted nonnegative rank is maximum (not the nonnegative rank), see Theorem 3.21. In fact, they only consider the case when the vertices of the solution T (corresponding to the columns of U) belong to the low-dimensional affine subspace defined by S (corresponding to the column of M) in the NPP instance.

Theorem 3.23. *Linear EDM's of the following form*

$$M_n(i,j) = (i-j)^2 \quad 1 \le i,j \le n,$$

satisfy

$$\text{rank}_+(M_n) \le 2 + \left\lceil \frac{n}{2} \right\rceil,$$

where $\lceil x \rceil$ *is the smallest integer greater or equal to* x.

Proof. Let first assume that n is even and define

$$U = \left(\begin{array}{cc|c} n-1 & 0 & \\ n-3 & 0 & \\ \vdots & \vdots & I_{n/2} \\ 3 & 0 & \\ 1 & 0 & \\ \hline 0 & 1 & \\ 0 & 3 & \\ \vdots & \vdots & P_{n/2} \\ 0 & n-3 & \\ 0 & n-1 & \end{array}\right), V = \left(\begin{array}{cc|c} 0 & n-1 & \\ 0 & n-3 & \\ \vdots & \vdots & M_{n/2} \\ 0 & 3 & \\ 0 & 1 & \\ \hline 1 & 0 & \\ 3 & 0 & \\ \vdots & \vdots & P_{n/2}M_{n/2} \\ n-3 & 0 & \\ n-1 & 0 & \end{array}\right)^T,$$

where I_m is the identity matrix of dimension m and P_m is the permutation matrix with $P_m(i,j) = I_m(i, m-j+1)\ \forall i,j$; see Equation (3.10) for an example when $n = 6$. One can check that

$$M_n = UV = \begin{pmatrix} M_{n/2} & A + P_{n/2}M_{n/2} \\ A^T + P_{n/2}M_{n/2} & M_{n/2} \end{pmatrix},$$

with

$$A = \begin{pmatrix} n-1 \\ n-3 \\ \vdots \\ 3 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 3 \\ \vdots \\ n-3 \\ n-1 \end{pmatrix}^T.$$

If n is odd, we simply observe that $\text{rank}_+(M_n) \le \text{rank}_+(M_{n+1}) \le 2 + \frac{n+1}{2} = 2 + \lceil \frac{n}{2} \rceil$, since M_n is a submatrix of M_{n+1} [37]. □

Remark 3.6. In the construction of Theorem 3.23, we have $\text{rank}(V) = 4$ and the factorization can then be interpreted as a nested polytopes problem (corresponding to $M^T = V^T U^T$) in which the outer polytope has (only) dimension 3. Therefore, there is still some room for improvement and $\text{rank}_+(M_n)$ is probably (much?) smaller.

This example also demonstrates that, in some cases, the structure of small size nonnegative factorizations (in this case, the one from Example 3.3) can be generalized to larger size nonnegative factorization problems. This might open new ways to computing large nonnegative factorizations.

In Example 3.3, the nonnegative rank is smaller than the restricted nonnegative rank because there exists a higher dimensional polytope with only 5 vertices whose convex hull encloses the 6 vertices defined by the columns of M. Nested polytopes instance corresponding to the RNR instance with M given by Example 3.3 and the two above solutions are illustrated on Figures 3.3 and 3.6 respectively (note that they are transposed to each other, but correspond to different solutions of the NPP instance), see Section 3.4.1. Notice that the second solution (Figure 3.6) completely includes the outer polytope P; therefore, the nonnegative rank of any nonnegative matrix with the same column space as the matrix M will be at most 5.

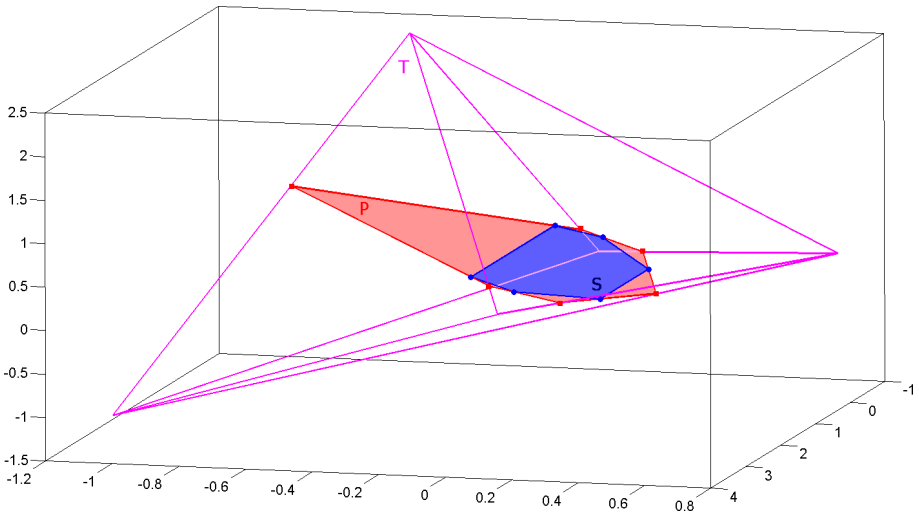

Figure 3.6: Illustration of the solution from Equation (3.10) as a nested polytopes problem, based on a linear EDM with $\text{rank}(M) = 3 < \text{rank}(U) = 4 < \text{rank}_+(M) = 5 < \text{rank}^*_+(M) = 6 = n$.

Remark 3.7. The solutions of the above nonnegative rank problems have been computed with a standard nonnegative matrix factorization algorithm, namely the hierarchical alternating least squares algorithm (HALS), see Chapter 4. In general, an optimal solution (with $||M - UV||_F$ close to machine accuracy) is found after 10 to 100 restarts of this algorithm on these small problems.

3.5.3 The Nonnegative Rank of a Product

Beasley and Laffey [6] proved that for $A = BC$ with A, B and $C \geq 0$

$$\operatorname{rank}_+(A) \leq \operatorname{rank}(B)\operatorname{rank}(C).$$

In particular, $\operatorname{rank}_+(A^2) \leq \operatorname{rank}(A)^2$. They also conjectured that for a nonnegative $n \times n$ matrix A,

$$\operatorname{rank}_+(A^2) \leq \operatorname{rank}(A),$$

which we prove to be false with the following counterexample (based on a circulant matrix)

$$A = \begin{pmatrix} 0 & 1 & a & 1+a & 1+a & a & 1 & 0 \\ 0 & 0 & 1 & a & 1+a & 1+a & a & 1 \\ 1 & 0 & 0 & 1 & a & 1+a & 1+a & a \\ a & 1 & 0 & 0 & 1 & a & 1+a & 1+a \\ 1+a & a & 1 & 0 & 0 & 1 & a & 1+a \\ 1+a & 1+a & a & 1 & 0 & 0 & 1 & a \\ a & 1+a & 1+a & a & 1 & 0 & 0 & 1 \\ 1 & a & 1+a & 1+a & a & 1 & 0 & 0 \end{pmatrix}, \quad (3.11)$$

where $a = 1+\sqrt{2}$. In fact, one can check that $\operatorname{rank}(A) = 3$ and $\operatorname{rank}_+(A^2) = 4$: indeed, $\operatorname{rank}_+^*(A^2) = 4$ can be computed with the algorithm of Aggarwal et al. [1] (see Figure 3.7 for an illustration) and, by Corollary 3.10, $\operatorname{rank}_+(A^2) = \operatorname{rank}_+^*(A^2)$ since $\operatorname{rank}_+^*(A^2) \leq \operatorname{rank}(A^2) + 1 = 4$.

Remark 3.8. The matrix A from Equation (3.11) is the slack matrix of a regular octagon with sides of length $\sqrt{2}$. By Theorem 3.19, we have $\operatorname{rank}_+^*(A) = 8$. Notice also that A has rank 3 and its columns have disjoint sparsity patterns so that $\operatorname{rank}_+^*(A) = 8$ is implied by Theorem 3.21 as well. What is the nonnegative rank of A? Defining

$$R = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix},$$

we have that $B = AR$ is symmetric, has rank 3 and only has zeros on its diagonal. By Theorem 3.21, $\operatorname{rank}_+^*(B) = 8$. Using Table 3.1, we have $\operatorname{rank}_+(B) \geq 6$. Moreover

$$\operatorname{rank}_+(AR) \leq \min(\operatorname{rank}_+(A), \operatorname{rank}_+(R)),$$

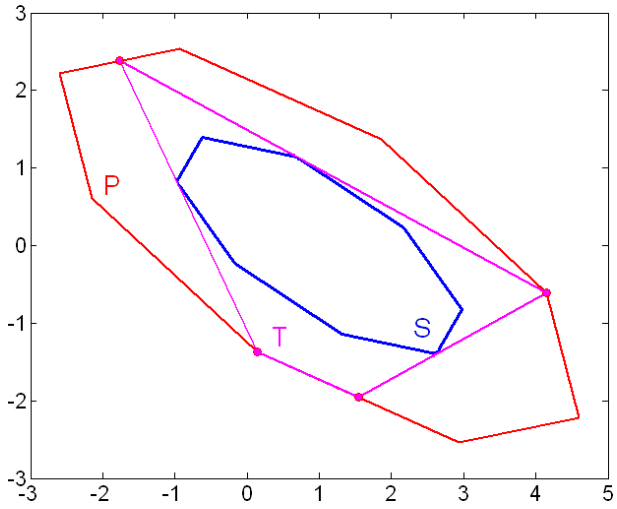

Figure 3.7: Illustration of a NPP instance corresponding to A^2 and an optimal solution T, cf. Equation (3.11).

implying that $6 \leq \text{rank}_+(B) \leq \text{rank}_+(A)$. Finally, $\text{rank}_+(A) = 6$ because

$$A = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & a \\ a & 0 & 0 & 0 & 1 & a+1 \\ 1 & 1 & 0 & 0 & 0 & a \\ 0 & a-1 & 1 & 0 & 0 & 1 \\ 0 & 1 & a & 0 & 1 & 0 \\ 1 & 0 & a+1 & 0 & a & 0 \\ 0 & 0 & a & 1 & 1 & 0 \\ 0 & 0 & 1 & a-1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & a & a \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & a & a & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}, \tag{3.12}$$

with $\text{rank}(U) = 4$ and $\text{rank}(V) = 5$ (U is the 8-by-6 matrices above, V the 6-by-8). Figure 3.8 displays the corresponding nested polytopes problem, see Section 3.4.1.

It is interesting to observe that, from this nonnegative factorization, one can obtain an extended formulation (lifting) Q of the regular octagon $P = \{x \in R^2 \mid Cx \leq d\}$, defined as $Q = \{(x, y) \in \mathbb{R}^2 \times \mathbb{R}^6 \mid Cx + Uy = d, y \geq 0\}$, with

$$C = \begin{pmatrix} 1 & \sqrt{2}/2 & 0 & -\sqrt{2}/2 & -1 & -\sqrt{2}/2 & 0 & \sqrt{2}/2 \\ 0 & \sqrt{2}/2 & 1 & \sqrt{2}/2 & 0 & -\sqrt{2}/2 & -1 & -\sqrt{2}/2 \end{pmatrix}^T,$$

and $d(i) = 1 + \frac{\sqrt{2}}{2}$ $\forall i$, see Equation (3.2). Since the system of equalities $Cx + Uy = d$ only defines 4 linearly independent equalities ($\text{rank}([C\ U]) = 4$), the

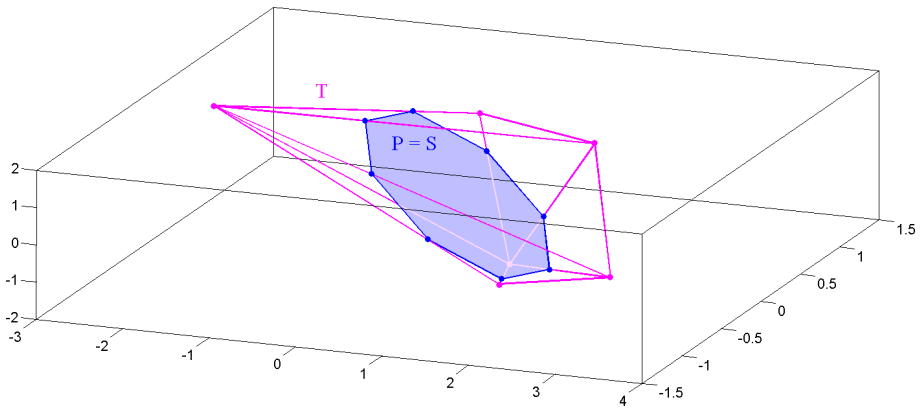

Figure 3.8: Illustration of a nested polytopes instance corresponding to A and an optimal solution, cf. Equations (3.11) and (3.12).

description of Q can then be simplified and expressed with 4 variables and 6 inequality constraints.

This extended formulation is actually a particular case of a construction proposed by Ben-Tal and Nemirovski [7] (see also Glineur [78], and [96]) to find an extended formulation of size $\mathcal{O}(k)$ for the regular 2^k-gon in two dimensions.

3.6 Improvements using the Matrix Transpose

In the previous sections, we have only considered the restricted nonnegative rank of $M \in \mathbb{R}_+^{m\times n}$. However, since $\operatorname{rank}_+(M) = \operatorname{rank}_+(M^T)$, the restricted nonnegative rank of M^T can also be used to generate lower and upper bounds for the nonnegative rank, exactly as for $\operatorname{rank}_+^*(M)$ with

$$\operatorname{rank}_+(M) \leq \operatorname{rank}_+^*(M^T) \leq m,$$

and

$$\operatorname{rank}_+^*(M^T) \leq \#\operatorname{vertices}(\operatorname{conv}(V^T) \cap \operatorname{col}(M^T)),$$

for any nonnegative factorization[14] (V^T, U^T) of $M^T = V^T U^T$. In particular, since $\operatorname{conv}(V^T)$ has the same number of vertices as $\operatorname{conv}(U)$, we have

$$\max\left(\operatorname{rank}_+^*(M), \operatorname{rank}_+^*(M^T)\right) \leq \phi(r, r_+).$$

In this section, we show how to go further in this direction in order to generate improved lower bounds for the nonnegative rank, using the relationship

[14] Without loss of generality, M, U and V are assumed to be column stochastic.

between the NPP instances corresponding to M and M^T. The main ingredient is the polar transformation.

3.6.1 Polar Transformation

The polar transformation of a set $C \in \mathbb{R}^n$ is defined as

$$C^0 = \{\ y \in \mathbb{R}^n \mid y^T x \leq 1\ \forall x \in C\}.$$

Based on this definition, the polar transformation of a bounded polytope P containing the origin in its interior with

$$P = \{\ x \in \mathbb{R}^n \mid Ax \leq \mathbf{1}_m\ \}, \tag{3.13}$$

where $A \in \mathbb{R}^{m \times n}$, is the set

$$P^0 = \text{conv}(A^T) = \text{conv}\left(\{\ A(i,:)^T \in\ \mathbb{R}^n, 1 \leq i \leq m\ \}\right),$$

i.e., it is the convex hull of the m rows of A, see Figure 3.9 for an illustration.

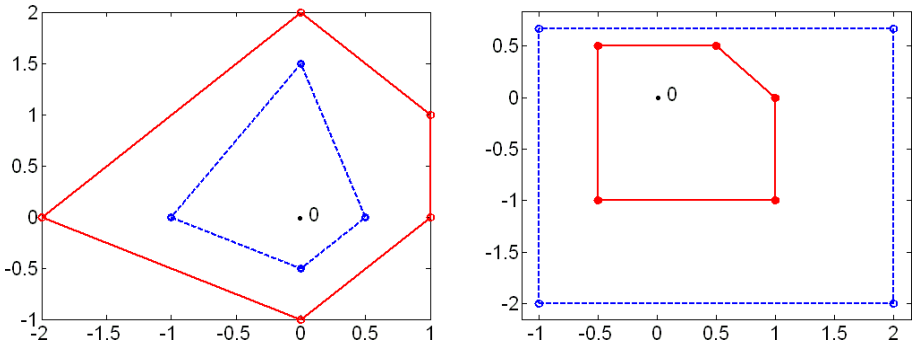

Figure 3.9: Example of the polar transformation: (left) polytopes P (continuous) and $S \subseteq P$ (dashed), and (right) polytopes P^0 (continuous) and $S^0 \supseteq P^0$ (dashed).

We now recall two well-known results, see, e.g., [157].

Lemma 3.24. *Let P be a bounded polytope containing the origin in its interior, then P^0 is a polytope containing the origin in its interior. Moreover, P has m facets if and only if P^0 has m vertices, and $(P^0)^0 = P$.*

Lemma 3.25. *Let P and S be two bounded polytopes containing the origin in their interior with $S \subseteq P$, then $P^0 \subseteq S^0$.*

3.6.2 Relationship between NPP instances of M and M^T

We now analyze the relationship between the NPP instances corresponding to M and M^T. Let us define the following problem,

> (NPP^f) Given a bounded polyhedron
>
> $$P = \{x \in \mathbb{R}^{r-1} \mid 0 \leq f(x) = Cx + d\},$$
>
> with $(C\ d) \in \mathbb{R}^{m \times r}$ of rank r, and a set S of n points in P not contained in any hyperplane (i.e., conv(S) is full-dimensional), find the minimum number k of facets defining a polytope T contained in P and containing S, i.e., $S \subseteq T \subseteq P$.

This is exactly the same problem as NPP where the number of facets of T is minimized instead of its number of vertices, as in [42,44], see Section 3.3.2. Let us introduce the following notation

- $\diamond$ NPP(M) is the NPP instance corresponding to the RNR instance of M, see Theorem 3.2.
- $\diamond$ NPP$^f(M)$ is the NPP^f instance where the polytope P and the set of points S are the ones from NPP(M).

We are going to show that computing the restricted nonnegative rank of M^T is polynomially equivalent to NPP$^f(M)$, i.e., NPP(M^T) is polynomially equivalent to NPP$^f(M)$. In other words, *minimizing the number of facets instead of the number of vertices in the NPP instance corresponding to M amounts to computing the restricted nonnegative rank of M^T.*

Theorem 3.26. *NPP(M^T) is polynomially equivalent to NPP^f(M).*

Proof. For rank$(M) \leq 1$, the problem is trivial: we have a zero-dimensional problem, and the solution of NPP(M^T) (resp. NPP$^f(M)$) has only one vertex (resp. facet).

Let us then assume $r = \text{rank}(M) \geq 2$, and note P (resp. S) the outer (resp. inner) simplex of NPP$^f(M)$ (same as in NPP(M)). The interior of P is non-empty since P contains the full-dimensional set S with $n \geq r$ points in $\mathbb{R}^{r-1}$. Without loss of generality, we may then assume that $0 \in \text{int}(P)$ (otherwise perform a translation, e.g., using the center of gravity of the set S, and generate an equivalent NPP instance, cf. Remark 3.1), and P is bounded (cf. Theorem 3.2). Therefore, P can be equivalently written as

$$P = \{\ x \in \mathbb{R}^{r-1} \mid Cx \leq \mathbf{1}_m\ \},$$

where $C \in \mathbb{R}^{m \times r-1}$ is full-rank. By construction, $M(:,i) = \mathbf{1}_m - Cv_i$ where the v_i are the n points in S, hence

$$S = \{v_i = C^\dagger(\mathbf{1}_m - M(:,i)) \in \mathbb{R}^{r-1}\ 1 \leq i \leq n\ \},$$

where $C^\dagger$ is the left inverse of C. Notice that this implies that $M(:,i) = \mathbf{1}_m - Cv_i = \mathbf{1}_m - CC^\dagger(\mathbf{1}_m - M(:,i)),\ 1 \le i \le n$ or equivalently

$$M(i,:) = \mathbf{1}_n^T - C(i,:)C^\dagger(\mathbf{1}_{m\times n} - M),\ 1 \le i \le m.$$

Let T be a solution for $\mathrm{NPP}^f(M)$. By Lemma 3.25, $S \subseteq T \subseteq P$ if and only if $P^0 \subseteq T^0 \subseteq (\mathrm{conv}(S))^0$. By Lemma 3.24, minimizing the number of facets of T in $\mathrm{NPP}^f(M)$ is then equivalent to minimizing the number of vertices of T^0 in an NPP instance with $(\mathrm{conv}(S))^0$ as the outer simplex, and P^0 as the inner simplex. By definition, the vertices of P^0 are given by

$$\{\ C(i,:)^T,\ i = 1,\dots,m\ \},\ \text{and}$$

$$(\mathrm{conv}(S))^0 = \{\ x \in \mathbb{R}^{r-1} \mid (C^\dagger(\mathbf{1}_{m\times n} - M))^T x \le \mathbf{1}_n\ \}.$$

Therefore the matrix N corresponding to this NPP instance is given by

$$\begin{aligned} N(:,i) &= \mathbf{1}_n - (C^\dagger(\mathbf{1}_{m\times n} - M))^T C(i,:)^T \\ &= \left(\mathbf{1}_n^T - C(i,:)C^\dagger(\mathbf{1}_{m\times n} - M)\right)^T = M(i,:)^T, \end{aligned}$$

implying $N = M^T$, which completes the proof. □

Corollary 3.27. *Using notations of Section 3.4.1, we have*

$$\mathrm{rank}_+^*(M^T) \le \#\,\mathrm{facets}(T \cap P_m), \tag{3.14}$$

where $\#\,\mathrm{facets}(Q)$ *denotes the number of facets of* Q.

Proof. By Theorem 3.14, $T \cap P_m$ is a feasible solution for $\mathrm{NPP}(M)$. Clearly, it also defines a feasible solution for $\mathrm{NPP}^f(M)$. Finally, $\mathrm{NPP}^f(M)$ is equivalent to $\mathrm{NPP}(M^T)$ (Theorem 3.14) which is equivalent to solving RNR of M^T (Theorem 3.2). □

3.6.3 Improved Lower Bound for the Nonnegative Rank

Corollary 3.27 together with Theorem 3.2 implies that for any nonnegative factorization (U,V) of M, we have

$$1 \le \min\Big(\frac{\#\,\mathrm{vertices}(T \cap P_m)}{\mathrm{rank}_+^*(M)}, \frac{\#\,\mathrm{facets}(T \cap P_m)}{\mathrm{rank}_+^*(M^T)}\Big),$$

where T represents the convex hull of the columns of U and P_m the column space of M, see Section 3.4.1.

Corollary 3.28. *Let* M *be a nonnegative matrix with* $r = \mathrm{rank}(M)$ *and* $r_+ = \mathrm{rank}_+(M)$, *then*

$$1 \le \max_{r \le r_u \le r_+} \min\Big(\frac{\mathrm{faces}(r_+, r_u - 1, r_u - r)}{\mathrm{rank}_+^*(M)}, \frac{\mathrm{faces}(r_+, r_u - 1, r_u - 2)}{\mathrm{rank}_+^*(M^T)}\Big).$$

Proof. For any rank-r_+ nonnegative factorization (U, V) of M, $T \cap P_m$ is a feasible solution for both NPP(M) and NPP$^f(M)$. Its number of vertices (resp. facets) is then a lower bound for $\text{rank}_+^*(M)$ (resp. $\text{rank}_+^*(M^T)$). Since T is a polytope with r_+ vertices in dimension (r_u-1), the number of vertices of $T \cap P_m$ can be bounded with $\text{faces}(r_+, r_u-1, r_u-r)$ (see Theorem 3.15), and its number of facets by $\text{faces}(r_+, r_u-1, r_u-2)$ because any facet of T intersected with P_m can potentially generate a facet of $T \cap P_m$, and the facets of a polytope in dimension $(r_u - 1)$ are $(r_u - 2)$-faces. □

Corollary 3.29. *If M is symmetric or rank-three, then*

$$\begin{aligned} \text{rank}_+^*(M) &= \text{rank}_+^*(M^T) \\ &\leq \max_{r \leq r_u \leq r_+} \min\Big(\text{faces}(r_+, r_u-1, r_u-r), \text{faces}(r_+, r_u-1, r_u-2)\Big). \end{aligned}$$

Proof. We have that $\text{rank}_+^*(M) = \text{rank}_+^*(M^T)$ for rank-three (cf. Section 3.3.3) and symmetric matrices (trivial), while Corollary 3.28 gives the result. □

This is a generalization of Equation (3.7) from Corollary 3.18, which only considered rank-three matrices (see Figure 3.5 and Table 3.1 for a comparison of the different bounds).

Concluding Remarks

In this chapter, we have first reviewed the known complexity results for the nonnegative rank computation, and linked them with NMF. We have then introduced a new quantity called the restricted nonnegative rank, whose computation amounts to solving a problem in computational geometry consisting of finding a polytope nested between two given polytopes. This allowed us to fully characterize its computational complexity (see Table 3.2). This geometric interpretation and the relationship between the nonnegative rank and the restricted nonnegative rank also let us derive new improved lower bounds for the nonnegative rank, in particular for slack matrices and linear Euclidean distance matrices. This also allowed us to provide counterexamples to two conjectures concerning the nonnegative rank.

We conclude the chapter with the following conjecture:

Conjecture 3.1. Computing the nonnegative rank and the corresponding nonnegative factorization of a nonnegative matrix is $\mathcal{NP}$-hard when the rank of the matrix is fixed and greater or equal to 4 (or even possibly 3).

In fact, we have shown that computing a nonnegative factorization amounts to solving a nested polytopes problem in which the outer polytope might live

in a higher dimensional space. Moreover, this space is not known a priori (we just know that it contains the columns of the matrix to be factorized, cf. Section 3.4.1). Therefore, it seems plausible to assume that this problem is at least as difficult than the restricted nonnegative rank computation problem in which the outer polytope lives in the same low-dimensional space and is known. Moreover, even in the rank-three case, even though the inner polytope has dimension two, the outer polytope might have any dimension (up to the dimensions of the matrix; see, e.g., Figures 3.3 and 3.6); therefore, it seems that the nonnegative rank computation might also be $\mathcal{NP}$-hard if the rank of the matrix is three. Notice that, when $\text{rank}_+^*(M) \leq 5$, Equation (3.6) implies $\text{rank}_+(M) = \text{rank}_+^*(M)$ so that the nonnegative rank can be computed in polynomial time in this particular case.

Table 3.2 recapitulates the complexity results for computing the restricted nonnegative rank and the nonnegative rank of a nonnegative matrix M.

$r = \text{rank}(M)$	$r_+^* = \text{rank}_+^*(M)$	$r_+ = \text{rank}_+(M)$
r not fixed	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard [148]
$r \geq 4$ fixed	$\mathcal{NP}$-hard (Theorem 3.6)	$\mathcal{NP}$-hard?
$r = 3$	polynomial (Theorem 3.3)	polynomial if $r_+^* \leq 5$ otherwise $\mathcal{NP}$-hard?
$r \leq 2$	trivial ($= r$)	trivial ($= r$) [143]

Table 3.2: Complexity of restricted nonnegative rank and nonnegative rank computations.

Chapter 4

Algorithms for Nonnegative Matrix Factorization

NMF is typically formulated as a nonlinear optimization problem with an objective function measuring the quality of the low-rank approximation. In this thesis, we consider the sum of squared errors, i.e., the (squared) Frobenius norm of the difference between the *nonnegative* matrix M and the approximation UV:

$$\min_{\substack{U \in \mathbb{R}^{m \times r} \\ V \in \mathbb{R}^{r \times n}}} ||M - UV||_F^2 \quad \text{such that} \quad U \geq 0,\ V \geq 0. \tag{NMF}$$

One of the main challenges of NMF is to design fast and efficient algorithms generating nonnegative factors (U, V) that minimize the objective function. In fact, on the one hand, practitioners need to compute rapidly good factorizations for large-scale problems (e.g., in text mining or image processing); on the other hand, as explained in Chapter 3, NMF is a $\mathcal{NP}$-hard problem and we cannot expect to find a globally optimal solution in a reasonable computational time. Hence practical algorithms aim instead at finding locally optimal solutions. More precisely, only convergence to stationary points of NMF is in general guaranteed. Recall that stationary points are points satisfying the necessary first-order optimality conditions, also called Karush-Kuhn-Tucker optimality conditions, see Section 2.1. For NMF, they reduce to

$$\begin{array}{lll} U \geq 0, & \nabla_U ||M - UV||_F^2 \geq 0, & U \circ \nabla_U ||M - UV||_F^2 = 0, \\ V \geq 0, & \nabla_V ||M - UV||_F^2 \geq 0, & V \circ \nabla_V ||M - UV||_F^2 = 0. \end{array} \tag{KKT-NMF}$$

where

$$\nabla_U ||M - UV||_F^2 = -2(M - UV)V^T, \quad \nabla_V ||M - UV||_F^2 = -2U^T(M - UV).$$

Most NMF algorithms are iterative and use the fact that NMF reduces to a convex nonnegative least squares problem (NNLS) when U or V is fixed (see Section 4.1.2). Actually, it seems that nearly all algorithms proposed for NMF adhere to the following general framework

(0) Select initial matrices (U, V). Then repeat the following two steps:

(a) Fix V: find a new $U \geq 0$ such that $||M - UV||_F^2$ is reduced.

(b) Fix U: find a new $V \geq 0$ such that $||M - UV||_F^2$ is reduced.

More precisely, at each iteration, one of the two factors is fixed and the other is updated in such a way that the objective function is reduced, which amounts to a two-block coordinate descent method. Notice that the role of matrices U and V is perfectly symmetric: if one transposes input matrix M, the new matrix M^T has to be approximated by a product $V^T U^T$, so that any formula designed to update for the first factor in this product directly translates into an update for the second factor in the original problem. Formally, if the update performed in step (a) is described by $U \leftarrow \text{update}(M, U, V)$, then an algorithm preserving symmetry will update the factor in step (b) according to $V \leftarrow \text{update}(M^T, V^T, U^T)^T$.

This update can be carried out in many different ways: the most natural possibility is to compute an optimal solution for the NNLS subproblem, which leads to a class of algorithms called alternating nonnegative least squares (ANLS), see Section 4.1.2. However, because this computation is relatively costly, and since an optimal solution for the NNLS problem corresponding to one factor is not strictly necessary before the other factor is updated, several algorithms compute only an approximate solution of the NNLS subproblem, sometimes very roughly, but with a cheaper computational cost, leading to an inexact two-block coordinate descent scheme. Any nonlinear optimization method can be used for this purpose, such as the multiplicative updates (cf. Section 4.1.1), projected gradient methods [113], Newton-like methods [34,51], and block-coordinate descent with various number of blocks including hierarchical alternating least squares with r blocks (cf. Section 4.1.3); see also [10,32,52,89] and the references therein. Notice that it can also be profitable to combine different algorithms, and come up with a hybrid approach. For example, the ALS algorithm updates alternatively matrices U and V by solving the corresponding unconstrained least squares problems and then projecting the solutions onto the nonnegative orthant. Even though ALS is not guaranteed to converge and typically fail to obtain good solutions for NMF problems (especially if M is a dense matrix), it is relatively cheap and leads to a fast initial decrease of the error so that it can be successfully used as a preprocessing step for other techniques, see, e.g., [32, 68] and the references therein.

In this chapter, we present in Section 4.1 three well-known and widely used algorithms for NMF, namely the alternating nonnegative least squares (ANLS), the multiplicative updates (MU), and the hierarchical alternating least squares (HALS). We then propose in Section 4.2 a simple modification of MU and HALS speeding up significantly their convergence, based on the analysis of their computational cost. This approach can potentially be used for any first-order NMF algorithm, and is applied successfully to the projected gradient method of Lin [113]. An explanation for the good performances of HALS is also given. In Section 4.3, NMF algorithms are embedded into the framework of multilevel methods in order to accelerate their convergence in some specific situations.

4.1 Three Existing Algorithms: MU, ANLS and HALS

4.1.1 Multiplicative Updates (MU)

In their seminal paper [105], Lee and Seung propose multiplicative update rules that aim at minimizing the Frobenius norm between a nonnegative matrix M and its approximation UV:

$$U \leftarrow U \circ \frac{[MV^T]}{[UVV^T]}, \qquad V \leftarrow V \circ \frac{[U^TM]}{[U^TUV]}. \tag{MU}$$

Theorem 4.1 ([47,106]). *For $M, U, V \geq 0$, the Frobenius norm $||M - UV||_F$ is nonincreasing under each of the multiplicative update rules* (MU).

This technique was actually originally proposed by Daube-Witherspoon and Muehllehner [47] to solve nonnegative least squares problems. The popularity of this algorithm came along with the popularity of NMF.

MU can be interpreted as a rescaled gradient descent scheme [106] (i.e, Quasi-Newton or variable metric method with an approximate diagonal Hessian matrix)[1]. In fact, one can observe that MU is equivalent to

$$U \leftarrow U - S_U \circ \nabla_U ||M - UV||_F^2, \quad \text{and} \quad V \leftarrow V - S_V \circ \nabla_V ||M - UV||_F^2,$$

where $S_U = \frac{[U]}{[2UVV^T]}$ and $S_V = \frac{[V]}{[2U^TUV]}$.

The algorithm based on the alternated application of these rules (see Algorithm 1) is not guaranteed to converge to a first-order stationary point (see, e.g., [89, Section 3.1] and references therein), but a slight modification proposed

[1] In standard notation, we have that $\min_x f(x)$ is solved iteratively using $x \leftarrow x - D_x \nabla f(x)$, where D_x is a diagonal matrix approximating the inverse of the Hessian matrix $\nabla^2 f(x)$.

Algorithm 1 Multiplicative Updates

Require: Data matrix $M \in \mathbb{R}_+^{m\times n}$ and initial iterates $(U,V) \in \mathbb{R}_+^{m\times r} \times \mathbb{R}_+^{r\times n}$.

1: **while** stopping criterion not satisfied **do**

2: $\quad U \leftarrow U \circ \frac{[MV^T]}{[U(VV^T)]}$;

3: $\quad V \leftarrow V \circ \frac{[U^TM]}{[(U^TU)V]}$;

4: **end while**

in [112] achieves this property (roughly speaking, MU is recast as a rescaled gradient descent method as above and the step length is modified accordingly).

We propose another simple approach to overcome this problem by replacing the above updates by the following:

Theorem 4.2. *For any constant $\epsilon > 0$, $M \geq 0$ and any*[2] *$(U,V) \geq \epsilon$, $||M - UV||_F$ is nonincreasing under*

$$U \leftarrow \max\Big(\epsilon, U \circ \frac{[MV^T]}{[UVV^T]}\Big), \qquad V \leftarrow \max\Big(\epsilon, V \circ \frac{[U^TM]}{[U^TUV]}\Big), \tag{4.1}$$

where the max *is taken component-wise. Moreover, every limit point of the corresponding (alternated) algorithm is a stationary point of the following optimization problem*

$$\min_{U\geq\epsilon, V\geq\epsilon} ||M - UV||_F^2. \tag{4.2}$$

Proof. See Section 5.4 where the more general Theorem 5.11 is proved. □

We might wonder how this modification of the feasible domain influences the objective function value, i.e., what is the increase in the objective function value of (4.2) with respect to (NMF)? Let $(U,V) \geq 0$ be any feasible point of (NMF) and define $(U_\epsilon, V_\epsilon) \geq \epsilon$ as follows

$$U_\epsilon(i,j) = \begin{cases} U_{ij} & \text{if } U_{ij} \geq \epsilon \\ \epsilon & \text{if } U_{ij} < \epsilon \end{cases}, \ 1 \leq i \leq m, 1 \leq j \leq r,$$

and similarly for V_ϵ. We then have

$$\begin{aligned} ||M - U_\epsilon V_\epsilon||_F &= ||M - U_\epsilon V_\epsilon - UV + UV||_F \\ &\leq ||M - UV||_F + ||U_\epsilon V_\epsilon - UV||_F. \end{aligned}$$

[2] $(U,V) \geq \epsilon$ means that U and V are component-wise larger than ϵ.

By construction $0 \leq UV \leq U_\epsilon V_\epsilon \leq (U+\epsilon)(V+\epsilon)$ so that

$$\begin{aligned} ||U_\epsilon V_\epsilon - UV||_F &\leq ||(U+\epsilon)(V+\epsilon) - UV||_F \\ &\leq ||\epsilon U \mathbf{1}_{r\times n} + \epsilon \mathbf{1}_{m\times r} V + \epsilon^2 \mathbf{1}_{m\times r}\mathbf{1}_{r\times m}||_F \\ &\leq \epsilon ||U||_F ||\mathbf{1}_{r\times n}||_F + \epsilon ||\mathbf{1}_{m\times r}||_F ||V||_F + \epsilon^2 ||\mathbf{1}_{m\times r}||_F ||\mathbf{1}_{r\times n}||_F \\ &= \epsilon\sqrt{rn}||U||_F + \epsilon\sqrt{rm}||V||_F + \epsilon^2 r\sqrt{mn}. \end{aligned} \tag{4.3}$$

Hence $||M - U_\epsilon V_\epsilon||_F \leq ||M-UV||_F + \mathcal{O}(\epsilon)$.

If $(U,V) \geq 0$ is a stationary point of NMF, then $||UV||_F \leq ||M||_F$, see Theorem 4.4. Let us assume without loss of generality $||U_{:k}||_2 = ||V_{k:}||_2$. By nonnegativity,

$$||U_{:k}||_2^2 = ||U_{:k}||_2 ||V_{k:}||_2 = ||U_{:k}V_{k:}||_F \leq ||UV||_F \leq ||M||_F,$$

implying $||U||_F \leq \sqrt{r}\max_p ||U_{:p}||_2 \leq \sqrt{r}||M||_F^{1/2}$, which also holds for V by symmetry. Plugging it in Equation (4.3), we obtain

$$||M - U_\epsilon V_\epsilon||_F \leq ||M-UV||_F + \epsilon r(\sqrt{n}+\sqrt{m})||M||_F^{1/2} + \epsilon^2 r\sqrt{mn}.$$

Therefore, in terms of relative error $\frac{||M-UV||_F}{||M|_F}$, to have an increase of the optimal relative objective function value of (4.2) compared to the one of (NMF) to be at most δ, one has to choose ϵ such that

$$\frac{\epsilon r(\sqrt{n}+\sqrt{m})}{||M||_F^{1/2}} + \frac{\epsilon^2 r\sqrt{mn}}{||M||_F} \leq \delta.$$

For example, for the cbcl database with $M \in [0,1]^{m\times n}$, $m = 361$, $n = 2429$ and $r = 49$, we need $\epsilon \leq 7\ 10^{-6}$ to obtain an increase of the relative error of at most 0.1% (i.e., $\delta = 10^{-3}$); and taking $\epsilon = 10^{-16}$ gives an increase in the relative error of at most $1.5\ 10^{-12}$% (i.e., $\delta = 1.5\ 10^{-14}$), which is negligeable.

We might also wonder how a stationary point of (4.2) (which can be obtained as a limit point of the updates from Equation (4.1), see Theorem 4.2) can be used to approximate a stationary point of (NMF). Let us then now denote (U_ϵ, V_ϵ) a stationary point of (4.2) and let us focus on U_ϵ (the same reasoning applies for V_ϵ by symmetry). Noting $f(U_\epsilon, V_\epsilon) = ||M - U_\epsilon V_\epsilon||_F^2$ and $\nabla_U f(U_\epsilon, V_\epsilon) = -2(M - U_\epsilon V_\epsilon)V_\epsilon^T$, we then have, for all i,j, either

- $U_\epsilon(i,j) = \epsilon$ and $(\nabla_U f(U_\epsilon, V_\epsilon))_{ij} \geq 0$, or
- $U_\epsilon(i,j) > \epsilon$ and $(\nabla_U f(U_\epsilon, V_\epsilon))_{ij} = 0$.

Defining

$$U(i,j) = \begin{cases} 0 & \text{if } U_\epsilon(i,j) = \epsilon \\ U_\epsilon(i,j) & \text{if } U_{ij} > \epsilon \end{cases}, \ 1 \leq i \leq m, 1 \leq j \leq r,$$

and V similarly using V_ϵ, we would like to see how close (U, V) is from stationarity. A natural question is then: how do $\nabla_U f(U_\epsilon, V_\epsilon)$ and $\nabla_U f(U, V)$ differ? Using the same trick as for the objective function value, we obtain

$$\begin{aligned} e &= \frac{1}{2}\max_{ij} |\nabla_U f(U_\epsilon, V_\epsilon) - \nabla_U f(U, V)|_{ij} \\ &= \max_{ij} |((M - U_\epsilon V_\epsilon)V_\epsilon^T) - ((M - UV)V^T)|_{ij} \\ &\leq \max_{ij} |MV_\epsilon^T - MV^T|_{ij} + \max_{ij} |U_\epsilon V_\epsilon V_\epsilon^T - UVV^T|_{ij} \\ &\leq \max_{ij} |M(V + \epsilon \mathbf{1}_{r\times n})^T - MV^T|_{ij} \\ &\quad + \max_{ij} |(U + \epsilon \mathbf{1}_{m\times r})(V + \epsilon \mathbf{1}_{r\times n})(V + \epsilon \mathbf{1}_{r\times n})^T - UVV^T|_{ij} \\ &= \mathcal{O}(\epsilon). \end{aligned}$$

Hence, for all i, j, we have either

- $U_{ij} = 0$ and $(\nabla_U f(U, V))_{ij} \geq -\mathcal{O}(\epsilon)$, or
- $U_{ij} > 0$ and $|\nabla_U f(U, V)|_{ij} \leq \mathcal{O}(\epsilon)$.

Therefore, even though (U, V) is not a stationary point of (NMF), it is close to satisfying the required conditions (KKT-NMF) for ϵ is sufficiently small.

Stopping Criterion

There are many different approaches for the stopping criterion of NMF algorithms using different indicators of the convergence of the iterates, including the evolution of the objective function, the norm of the projected gradient [113] (directly related to the stationarity conditions), and the norm of the difference between two consecutive iterates (see Section 4.2). These criteria are typically combined with either a maximum number of iterations or a time limit to ensure termination.

4.1.2 Alternating Nonnegative Least Squares (ANLS)

As mentioned in the introduction, although NMF is a nonconvex and difficult problem, it is convex separately in each of the two factors U and V, i.e., finding the optimal factor U corresponding to a fixed factor V reduces to a convex optimization problem, and vice-versa. More precisely, this convex problem corresponds to a nonnegative least squares (NNLS) problem, i.e., a least squares problem with nonnegativity constraints. Notice that the NNLS subproblem $\min_{U\geq 0} ||M - UV||_F^2$ (and symmetrically for V) can be decomposed into m

independent NNLS subproblems in r variables corresponding to each row of U since

$$||M - UV||_F^2 = \sum_{i=1}^{m} ||M_{i:} - U_{i:}V||_F^2,$$

which makes this problem easier to solve. The so-called alternating nonnegative least squares (ANLS) algorithm for NMF minimizes the cost function alternatively over factors U and V so that a stationary point of NMF is obtained in the limit, see Theorem 2.2 in Section 2.1.3. A frequent strategy to solve

Algorithm 2 Alternating Nonnegative Least Squares

Require: Data matrix $M \in \mathbb{R}_+^{m \times n}$ and initial iterate $V \in \mathbb{R}_+^{r \times n}$.

1: **while** stopping criterion not satisfied **do**
2: $\quad U \leftarrow \text{argmin}_{U \geq 0} ||M - UV||_F$;
3: $\quad V \leftarrow \text{argmin}_{V \geq 0} ||M - UV||_F$.
4: **end while**

the NNLS subproblems is to use active set methods [104] (see Appendix A) for which efficient implementations[3] are described in [98, 99, 145], and shown to typically outperform other tested variants of NNLS methods on synthetic, image and text datasets. We refer the reader to [27] for a survey about NNLS methods.

Remark 4.1. There actually exists other partitions of the variables that preserve convexity of the alternating minimization subproblems: since the cost function can be rewritten as $||M - \sum_{i=1}^{r} U_{:i}V_{i:}||_F$, it is clearly convex as long as variables do not include simultaneously an element of a column of U and an element of the corresponding row of V (i.e., U_{ki} and V_{il} for the same index i). Therefore, given a subset of indices $K \subseteq R = \{1, 2, \ldots, r\}$, NMF is clearly convex for both the following subset of variables

$$P_K = \left\{ U_{:i} \;\middle|\; i \in K \right\} \cup \left\{ V_{j:} \;\middle|\; j \in R \setminus K \right\}$$

and its complement

$$Q_K = \left\{ U_{:i} \;\middle|\; i \in R \setminus K \right\} \cup \left\{ V_{j:} \;\middle|\; j \in K \right\}.$$

Convexity is lost as soon as one column of U ($U_{:i}$) and the corresponding row of V ($V_{i:}$) are optimized simultaneously, so that the corresponding minimization subproblem can no longer be solved up to global optimality. In fact, such subproblems will be studied in Chapter 5, and shown to be $\mathcal{NP}$-hard.

[3] Available at `http://www.cc.gatech.edu/~hpark/`.

4.1.3 Hierarchical Alternating Least Squares (HALS)

In ANLS, variables are partitioned at each iteration such that each subproblem is convex. However, the resolution of these convex NNLS subproblems is non-trivial and relatively expensive (see Appendix A). If we optimize instead one single variable at a time, we get a simple univariate quadratic problem which admits a closed-form solution

$$U_{ik}^* = \text{argmin}_{U_{ik}\geq 0} ||M - UV||_F^2 = \max\left(0, \frac{M_{i:}V_{k:}^T - \sum_{l\neq k} U_{il}V_{l:}V_{k:}^T}{V_{k:}V_{k:}^T}\right).$$

Moreover, since the optimal value for a given entry of U (resp. V) does not depend of the other entries of the same column (resp. row), one can optimize alternatively whole columns of U (resp. rows of V), with

$$\begin{aligned} U_{:k}^* &= \text{argmin}_{U_{:k}\geq 0} ||M - UV||_F^2 \\ &= \text{argmin}_{U_{:k}\geq 0} ||R_k - U_{:k}V_{k:}||_F^2 = \max\left(0, \frac{R_k V_{k:}^T}{||V_{k:}||_2^2}\right), \end{aligned} \tag{4.4}$$

and

$$\begin{aligned} V_{k:}^* &= \text{argmin}_{V_{k:}\geq 0} ||M - UV||_F^2 \\ &= \text{argmin}_{V_{k:}\geq 0} ||R_k - U_{:k}V_{k:}||_F^2 = \max\left(0, \frac{U_{:k}^T R_k}{||U_{:k}||_2^2}\right), \end{aligned} \tag{4.5}$$

where $R_k \doteq M - \sum_{i\neq k} U_{:i}V_{i:}$ is the k^{th} residual matrix. Notice that, in practice, one should not compute explicitly the residual matrices R_k, but instead observe that

$$R_k V_{k:}^T = MV_{k:}^T - \sum_{i\neq k} U_{:i}(V_{i:}V_{k:}^T) \text{ and } U_{:k}^T R_k = U_{:k}^T M - \sum_{i\neq k}(U_{:k}^T U_{:i})V_{i:},$$

see Algorithm 3. In fact, computing the residual matrices would be more expensive, especially for sparse matrices since R_k can in principle be dense.

This method was first proposed by Cichocki et al. [33, 35] and later independently by several other authors [70, 89, 110], and is herein referred to as hierarchical alternating least squares (HALS)[4]. It amounts to solving each NNLS subproblem with a single round of an exact-coordinate descent method with r blocks (for which any cyclic order on the columns of U and the rows of V is admissible).

[4] In [89], HALS is referred to as rank-one residue iteration (RRI), and in [110] as FastNMF.

Algorithm 3 Hierarchical Alternating Least Squares

Require: Data matrix $M \in \mathbb{R}_+^{m\times n}$ and initial iterates $(U,V) \in \mathbb{R}_+^{m\times r} \times \mathbb{R}_+^{r\times n}$.

$\alpha^* = \text{argmin}_\alpha ||M - \alpha UV||_F = \frac{\langle M,UV\rangle}{\langle UV,UV\rangle} = \frac{\langle MV^T,U\rangle}{\langle U^TU,V^TV\rangle}$; $U \leftarrow \alpha^* U$;
while stopping criterion not satisfied **do**
    Compute $A = MV^T$ and $B = VV^T$.
    for $k = 1 \ : \ r$ **do**
        $U_{:k} \leftarrow \max\left(0, \frac{A_{:k} - \sum_{l=1,l\neq k}^r U_{:l}B_{lk}}{B_{kk}}\right)$;
    end for
    Compute $C = U^TM$ and $D = U^TU$.
    for $k = 1 \ : \ r$ **do**
        $V_{k:} \leftarrow \max\left(0, \frac{C_{k:} - \sum_{l=1,l\neq k}^r D_{kl}V_{l:}}{D_{kk}}\right)$;
    end for
end while

Convergence of HALS

A potential issue with HALS is that, in the course of the optimization process, one of the vectors $U_{:k}$ (or $V_{k:}$) and the corresponding rank-one factor $U_{:k}V_{k:}$ may become equal to zero (this happens for example if one of the residuals $R_k = M - \sum_{i\neq k} U_{:i}V_{i:}$ is nonpositive). This then leads to numerical instabilities (the next update is not well-defined) and a rank-deficient approximation (i.e., with a rank lower than r). A possible way to overcome this problem is to replace the zero lower bounds on $U_{:k}$ and $V_{k:}$ by a small positive constant $\epsilon \lll 1$ (as for the MU), and consider the following subproblems

$$U_{:k}^* = \text{argmin}_{U_{:k}\geq\epsilon}||R_k - U_{:k}V_{k:}||_F^2 \quad \text{and} \quad V_{k:}^* = \text{argmin}_{V_{k:}\geq\epsilon}||R_k - U_{:k}V_{k:}||_F^2,$$

which lead to the modified closed-form update rules:

$$U_{:k}^* = \max\left(\epsilon, \frac{R_kV^T}{||V_{k:}||_2^2}\right) \quad \text{and} \quad V_{k:}^* = \max\left(\epsilon, \frac{U^TR_k}{||U_{:k}||_2^2}\right). \tag{4.6}$$

This idea was already suggested in [35] in order to avoid numerical instabilities. In fact, this variant of the algorithm is now well-defined in all situations because it guarantees $U_{:k} > 0$ and $V_{k:} > 0$ at each iteration. Furthermore, one can now easily prove that it converges to a stationary point of (4.2).

Theorem 4.3. *For every constant $\epsilon > 0$, the limit points of the block-coordinate descent algorithm* (4.6) *initialized with positive matrices and applied to the optimization problem* (4.2) *are stationary points.*

Proof. Following Theorem 2.1, we need the following two conditions to hold:

- ◇ each block of variables is required to belong to a closed convex set,
- ◇ the minimum computed at each iteration for a given block of variables is uniquely attained.

The first condition is clearly satisfied here, since $U_{:k}$ and $V_{k:}$ belong respectively to $([\epsilon, +\infty[)^m$ and $([\epsilon, +\infty[)^n$, which are closed convex sets. The second condition holds because subproblems can be shown to be strictly convex, so that their optimal value is uniquely attained by the solutions provided by rules (4.6). Strict convexity is due to the fact that the objective function of these problems are sums of quadratic terms, each involving a single variable and having a strictly positive coefficient (given by the corresponding diagonal element of matrix $VV^T > 0$). □

Scaling of Initial Iterates

Another practical issue is related to the choice of the initial matrices (U, V). In particular, HALS is sensitive to the scaling of the initial matrices (unlike MU and ANLS). For example, if the initial matrices U and V are chosen such that $UV \ggg M$, optimal columns of U and optimal rows of V computed by formulas (4.4) and (4.5) at the first step will most likely be equal to zero. If the initial matrices (U, V) are properly scaled [89, p.85], i.e., by ensuring that

$$1 \;=\; \text{argmin}_\alpha \, ||M - \alpha UV||_F = \frac{\langle M, UV\rangle}{\langle UV, UV\rangle}, \tag{4.7}$$

this behavior is in general avoided (see step 1 of Algorithm 3). Notice that since

$$\frac{\langle M, UV\rangle}{\langle UV, UV\rangle} = \frac{\langle MV^T, U\rangle}{\langle U^TU, VV^T\rangle},$$

and the matrices MV^T and VV^T have to be computed anyway (see step 3 of Algorithm 3), the additional cost for the initial scaling is rather low with $\mathcal{O}(mr^2)$ operations for the computation of U^TU.

Obviously, any stationary point is scaled ; the next Theorem is an extension of a result of Ho et al. [89].

Theorem 4.4. *The following statements are equivalent*

(1) (U, V) *is scaled;*

(2) UV *is on the boundary of* $\mathcal{B}\Big(\frac{M}{2}, \frac{1}{2}||M||_F\Big)$*, the ball centered at* $\frac{M}{2}$ *of radius* $\frac{1}{2}||M||_F$*;*

(3) $||M - UV||_F^2 = ||M||_F^2 - ||UV||_F^2$ *(implying* $||M||_F^2 \geq ||UV||_F^2$*).*

Proof. A solution (U, V) is scaled if and only if $\langle M, UV\rangle = \langle UV, UV\rangle$, see Equation (4.7), which is equivalent to

$$\begin{aligned}\langle UV - M, UV\rangle &= 0\\ \langle UV - M, UV\rangle + \left\langle \frac{M}{2}, \frac{M}{2}\right\rangle &= \left\langle \frac{M}{2}, \frac{M}{2}\right\rangle\\ \left\langle \frac{M}{2} - UV, \frac{M}{2} - UV\right\rangle &= \left\langle \frac{M}{2}, \frac{M}{2}\right\rangle,\end{aligned}$$

so that (1) and (2) are equivalent. For the equivalence of (1) and (3), we have

$$\begin{aligned}\langle M - UV, M - UV\rangle &= ||M||_F^2 - 2\langle M, UV\rangle + ||UV||_F^2\\ &= ||M||_F^2 - ||UV||_F^2 - 2\Big(\langle M, UV\rangle - \langle UV, UV\rangle\Big).\\ &= ||M||_F^2 - ||UV||_F^2\end{aligned}$$

if and only if $\langle M, UV\rangle = \langle UV, UV\rangle$. □

Theorem 4.4 can be used as follows: when one computes the error of the current solution, one can scale it without further computational cost. In fact,

$$\begin{aligned}||M - UV||_F^2 &= \langle M - UV, M - UV\rangle\\ &= ||M||_F^2 - 2\langle M, UV\rangle + ||UV||_F^2.\end{aligned} \tag{4.8}$$

Note that the third term of (4.8) can be computed in $O(\max(m, n)r^2)$ operations since

$$\begin{aligned}||UV||_F^2 &= \sum_{ij}\Big(\sum_k U_{ik}^2 V_{kj}^2\Big) + 2\sum_{ij}\Big(\sum_{k\neq l} U_{ik}V_{kj}U_{il}V_{lj}\Big)\\ &= \sum_k\Big(\sum_i U_{ik}^2\Big)\Big(\sum_j V_{kj}^2\Big) + 2\sum_{k\neq l}\Big(\sum_i U_{ik}U_{il}\Big)\Big(\sum_j V_{kj}V_{lj}\Big)\\ &= \sum_{ij}|(U^TU)\circ(VV^T)|_{ij}.\end{aligned}$$

This is especially interesting for sparse matrices since $\langle M, UV\rangle = \langle MV^T, U\rangle = \langle U^TM, V\rangle$, hence UV (which could be dense) does not need to be computed explicitly.

4.2 Accelerated MU and HALS Algorithms

In this section, we first analyze the computational cost needed to update the factors U in MU and HALS (since the V factor update is perfectly symmetric

with $M^T = V^T U^T$), then make several simple observations leading to the design of more efficient versions of these algorithms. This improvement can potentially be applied to any first-order NMF iterative algorithm, which is illustrated on a projected gradient method [113], but we focus here on MU (because it is by far the most popular NMF algorithm) and HALS (because it is very efficient in practice). Finally we experimentally demonstrate the improvements in speed of convergence on several image and text datasets, with a comparison with the state-of-the-art ANLS algorithm of Kim and Park [99]. We also give an explanation of the remarkable performances of HALS in Section 4.2.2.

4.2.1 Computational Cost

In order to make the analysis valid for both dense and sparse matrices, let us introduce the parameter K denoting the number of nonzero entries in M if M is sparse, $K = mn$ otherwise. We assume that NMF achieves compression, which is often a requirement in practice. This means that storing U and V must be cheaper than storing M: roughly speaking, the number of nonzero entries in M must be larger than the number of entries in U and V, i.e., $K \geq r(m+n)$.

Algorithms 4 and 5 give an estimate of the number of floating point operations (flops) of each matrix product computation needed to update U in MU and HALS respectively[5]. One can check that the proposed organization of the different matrix computations (and, in particular, the ordering of the matrix products) minimizes to the total computational cost (for example, starting the computation of the MU denominator UVV^T with the product UV is clearly worse than with VV^T).

Algorithm 4 MU update for U

1: $A = MV^T$; $\rightarrow 2Kr$ flops
2: $B = VV^T$; $\rightarrow 2nr^2$ flops
3: $C = UB$; $\rightarrow 2mr^2$ flops
4: $U \leftarrow U \circ \frac{[A]}{[C]}$; $\rightarrow 2mr$ flops
% Total: $r(2K + 2nr + 2mr + 2m)$ flops

MU and HALS have almost exactly the same computational cost (the difference being mr flops). It is particularly interesting to observe that

1. The first two steps 1. and 2. in both algorithms are identical and do not depend on the matrix U;

2. The first step (i.e., computing MV^T) is the most expensive one, using the assumption $K \geq r(m+n)$.

[5]The product of a m-by-n matrix with a n-by-r matrix requires $2mnr$ flops.

Algorithm 5 HALS update for U

1: $A = MV^T$; $\rightarrow 2Kr$ flops
2: $B = VV^T$; $\rightarrow 2nr^2$ flops
3: **for** $i = 1, 2, \ldots, r$ **do**
4: $\quad C_{:k} = \sum_{l=1}^{p-1} U_{:l}B_{lk} + \sum_{l=p+1}^{r} U_{:l}B_{lk}$; $\rightarrow 2m(r-1)$ flops, r times
5: $\quad U_{:k} \leftarrow \max\left(0, \frac{A_{:k}-C_{:k}}{B_{kk}}\right)$; $\rightarrow 3m$ flops, r times
6: **end for**
% Total: $r(2K + 2nr + 2mr + m)$ flops

This time consuming step should be performed sparingly, i.e., we should take full advantage of having computed the relatively expensive MV^T and VV^T matrix products. This can be done by updating U several times before the next update of V, i.e., perform steps 3. and 4. in MU (resp. steps 3. to 6. in HALS) several times.

The original MU and HALS algorithms do not take advantage of this fact, and update alternatively matrices U and V only once at each iteration. The question is: how many times should we update U?, i.e., how many inner iterations of MU and HALS should we perform? This is the topic of the next section.

4.2.2 Stopping Criterion for the Inner Iterations

Let us first focus on the MU algorithm. Based on the flops count, it is possible to estimate how much more expensive the first update of U is with respect to the next ones (for V fixed), which is given by the following factor ρ_U (the corresponding value for V will be denoted by ρ_V)

$$\rho_U = \frac{2Kr + 2nr^2 + 2mr^2 + 2mr}{2mr^2 + 2mr} = 1 + \frac{K + nr}{mr + m}. \quad \left(\rho_V = 1 + \frac{K + mr}{nr + n}\right).$$

Values of ρ_U and ρ_V for several datasets are given in Section 4.2.3, see Tables 4.1 and 4.2.

Notice that for $K \geq r(m + n)$, we have $\rho_U \geq 2$ so that the first update of U is at least twice as expensive as the next ones. For a dense matrix, K is equal to mn and we actually have that $\rho_U = 1 + \frac{n(m+r)}{m(r+1)} \geq 1 + \frac{n}{r+1}$, which is typically quite large since n is often much larger than r. For example, U could be updated about $1 + \rho_U = \left(2 + \frac{n}{r+1}\right)$ times for the same computational cost as two independent updates of U in the original MU.

Fixed Number of Inner Iterations

A natural and simple choice is then to update U and V a fixed number of times, depending on the values of ρ_U and ρ_V. Let us introduce a parameter $\alpha \geq 0$ such that U is updated $(1 + \alpha\rho_U)$ times before the next update of V, and V is updated $(1 + \alpha\rho_V)$ times before the next update of U. Let us also denote the corresponding algorithm MU_α (MU_0 reduces to the original MU). We then have that performing the $(1 + \alpha\rho_U)$ updates of U in MU_α has approximately the same computational cost as updating $(1 + \alpha)$ times U in MU_0.

Note however that when the number of rows m and columns n of matrix M are not of the same order of magnitude, for example when $n \gg m$, we have $\rho_U \gg \rho_V$. Hence, on the one hand, matrix U has significantly less entries than V ($mr \ll nr$), and the corresponding NNLS subproblem features a much smaller number of variables ; on the other hand, $\rho_U \gg \rho_V$ and many more updates of U are performed. In other words, many more iterations are performed on the simpler problem, which does not seem to be reasonable. For example, for the CBCL face database (cf. Section 4.2.3) with $m = 361$, $n = 2429$ and $r = 20$, we have $\rho_V \approx 18$ and $\rho_U \approx 123$, and these hundred updates of U are not necessary to obtain an iterate close to an optimal solution of the corresponding NNLS subproblem. Therefore, we propose to add the following stopping criterion. Noting $U^{(l)}$ the iterate after l updates of U (while V is being kept fixed), we stop iterations as soon as

$$||U^{(l+1)} - U^{(l)}||_F \leq \delta||U^{(1)} - U^{(0)}||_F, \tag{4.9}$$

i.e., as soon as the improvement compared to the first update is negligible. Based on numerical experiments (cf. Section 4.2.3), it seems that $\delta = 0.01$ gives good results, and *this value will be used for all the tests of this section.*

Algorithm 6 displays the pseudocode for the accelerated MU and HALS algorithms.

In order to find an appropriate parameter α, we have performed some preliminary tests on image and text datasets. First, let us denote $e(t)$ the Frobenius norm of the error $||M - UV||_F$ achieved by an algorithm within time t, and define

$$E(t) = \frac{e(t) - e_{\min}}{e(0) - e_{\min}}, \tag{4.10}$$

where $e(0)$ is the error of the initial iterate, and $e_{\min}$ is the smallest error achieved among all algorithms across all initializations. The quantity $E(t)$ therefore is a normalized measure of the improvement of the objective function as a function of the initial gap with respect to time; and we have $0 \leq E(t) \leq 1$ for monotonically decreasing algorithms (such as MU and HALS). The advantage of $E(t)$ over $e(t)$ is that one can meaningfully take the average over several

Algorithm 6 Accelerated MU and HALS

Require: Data matrix $M \in \mathbb{R}_+^{m \times n}$ and initial iterates $(U, V) \in \mathbb{R}_+^{m \times r} \times \mathbb{R}_+^{r \times n}$.

1: **while** stopping criterion not satisfied **do**
2: Compute $A = MV^T$ and $B = VV^T$; $U^{(0)} = U$;
3: **for** $l = 1 : 1 + \alpha\rho_U$ **do**
4: Compute $U^{(l)}$ using either MU or HALS (cf. Algorithms 4 and 5);
5: **if** $||U^{(l)} - U^{(l-1)}||_F \leq 0.01||U^{(1)} - U^{(0)}||_F$ **then**
6: break;
7: **end if**
8: **end for**
9: $U \leftarrow U^{(l)}$;
10: Update V from U and M using a symmetrically adapted version of steps 2-9;
11: **end while**

runs involving different initializations and datasets, and display the average behavior of a given algorithm.

Figure 4.1 displays the average of this function $E(t)$ for dense (on the left) and sparse (on the right) matrices using the datasets described in Section 4.2.3 for five values of α (0, 0.5, 1, 2 and 4). We observe that the original MU algorithm ($\alpha = 0$) converges significantly less rapidly than all the other tested variants (especially in the dense case). The best value for the parameter α is around one.

Figure 4.2 displays the same computational experiments for HALS[6]. As for MU, HALS with α around one performs better than the original HALS. For sparse matrices, the improvement is harder to discern (but still present); an explanation for that observation will be given in Section 4.2.4.

Dynamical Choice of Number of Inner Iterations

Another possibility to decide when to switch from updating U to V and vice versa would be to use an appropriate criterion. For example, it is possible to use the norm of the projected gradient as proposed by Lin [113], or the norm of the difference between two iterates $||U^{(l+1)} - U^{(l)}||_F$ as presented in the above section but without any a priori fixed maximal number of inner iterations. However, when trying several variants based solely on these criteria, it turned out that none would consistently give better results than the simple approach

[6]Because HALS involves a loop over the columns of U and rows of V, we observed that an update of HALS is noticeably slower than an update of MU when using MATLAB® (especially for $r \gg 1$) despite the quasi-equivalent theoretical computational cost. Therefore, to obtain fair results, we adjusted ρ_U and ρ_V measuring directly the ratio between the time spent for the first update and the next one using *cputime* function of MATLAB®.

outlined in the previous section.

To illustrate this, we have modified Lin's projected gradient algorithm (PG) [113] by fixing the number of inner iterations along with the stopping criterion defined in Equation (4.9) as for MU and HALS, using[7] different values for the parameter α. Figure 4.3 displays the computational results, and demonstrates that this variant works significantly better than the original PG algorithm (as available from [113]). A value of α around 0.5 gives the best results.

4.2.3 Numerical Experiments

In this section, we compare the following algorithms

1. (**MU**) The multiplicative updates algorithm of Lee and Seung (Section 4.1.1).
2. (**A-MU**) The accelerated MU with fixed number of inner iterations using $\alpha = 1$ (Section 4.2.2).
3. (**HALS**) The HALS algorithm of Cichocki et al. (Section 4.1.3).
4. (**A-HALS**) The accelerated HALS with fixed number of inner iterations using $\alpha = 1$ (Section 4.2.2).
5. (**PG**) The projected gradient method of Lin [113].
6. (**A-PG**) The modified projected gradient method of Lin [113] using $\alpha = 0.5$ (Section 4.2.2).
7. (**ANLS**) The alternating nonnegative least squares algorithm of Kim and Park [99] (Section 4.1.3).

All tests were run with MATLAB® 7.1 (R14), on a 3GHz Intel® Core™2 Dual CPU PC. We present numerical results on images datasets (dense matrices, Section 4.2.3) and on text datasets (sparse matrices, Section 4.2.3). The code of the algorithms is available on

`http://www.core.ucl.ac.be/~ngillis/papers/Acc_MU_HALS_PG.zip`

(run file *RunME.m* for a nice example with the CBCL face dataset).

[7] Lin's algorithm [113] also requires the computation of VV^T and MV^T since the gradient is given by $\nabla_W ||M-UV||_F^2 = 2UVV^T - 2MV^T$, so that our approach can be easily extended.

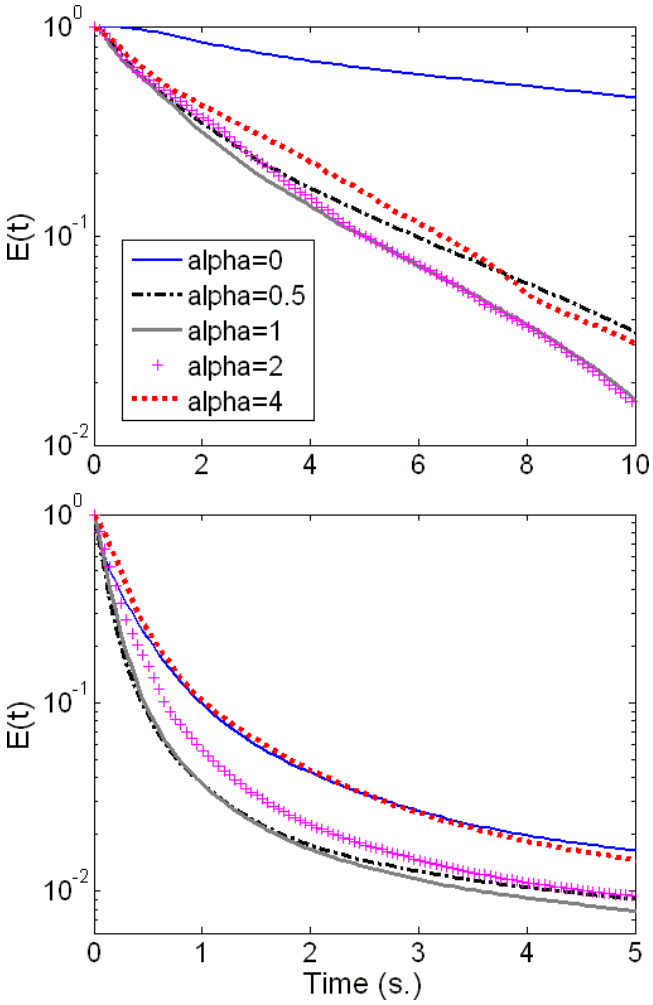

Figure 4.1: Average of functions $E(t)$ for MU using different values of α: (top) dense matrices, (bottom) sparse matrices. It is the average over 4 image datasets and 6 text datasets, using two different values for the rank for each dataset and 10 random initializations, see Section 4.2.3.

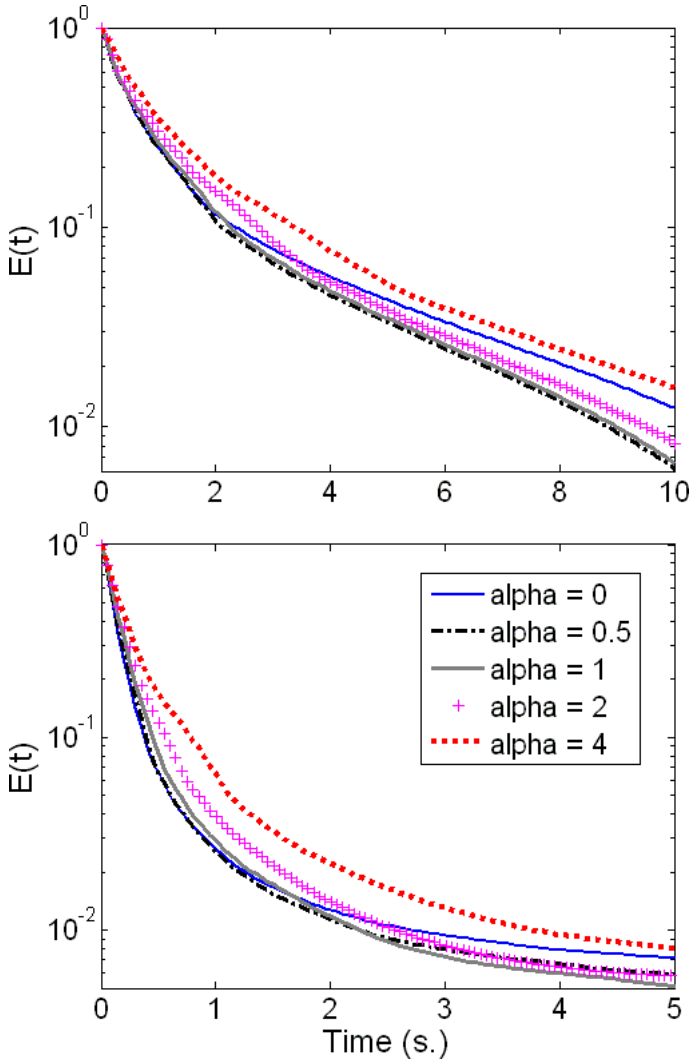

Figure 4.2: Average of functions $E(t)$ for HALS using different values of α: (top) dense matrices, (bottom) sparse matrices. Same settings as Figure 4.1.

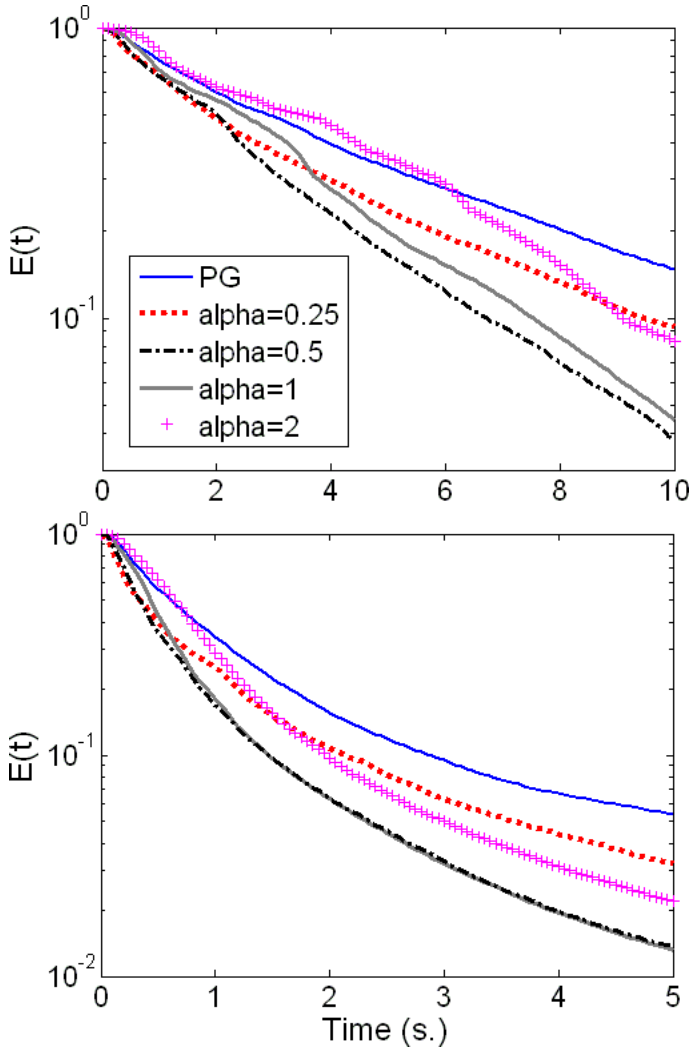

Figure 4.3: Average of functions $E(t)$ for the projected gradient algorithm of Lin [113], and its modification using a fixed number of inner iterations. Same settings as Figure 4.1.

Dense Matrices - Images Datasets

Table 4.1 summarizes characteristics for the different datasets.

Table 4.1: Image datasets.

Data	# pixels	m	n	$\lfloor \rho_U \rfloor$	$\lfloor \rho_V \rfloor$
ORL[1]	112×92	10304	400	13, 7	358, 195
Umist[2]	112×92	10304	575	19, 10	351, 188
CBCL[3]	19×19	361	2429	85, 47	12, 7
Frey[2]	28×20	560	1965	67, 36	19, 10

$\lfloor x \rfloor$ denotes the largest integer smaller than x.
[1] `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`
[2] `http://www.cs.toronto.edu/~roweis/data.html`
[3] `http://cbcl.mit.edu/cbcl/software-datasets/FaceData2.html`

For each dataset, we use two different values for the rank ($r = 30, 60$) and initialize the algorithms with the same 10 random factors (U, V) (using i.i.d. uniform random variables on $[0, 1]$)[8]. In order to assess the performance of the different algorithms, we display individually for each dataset the average over all runs of the function $E(t)$ defined in Equation (4.10), see Figures 4.4 and 4.5.

First, these results confirm what was already observed by previous work: PG performs better than MU [113], ANLS performs better than MU and PG [99] and HALS perform the best [89]. Second, they confirm that the accelerated algorithms indeed are more efficient: A-MU (resp. A-PG) clearly outperforms MU (resp. PG) in all cases, while A-HALS is much more efficient for the first two databases (ORL and Umist), and behaves slightly better as HALS for the other two (CBCL and Frey). It is interesting to notice that A-MU performs better than A-PG, and only slightly worse than ANLS, often converging as fast during the first iterations. Finally, A-HALS is, and sometimes by far, the best algorithm for all tested databases.

Sparse Matrices - Text Datasets

Table 4.2 summarizes characteristics for the different datasets.

[8] Generating initial matrices (U, V) randomly typically leads to a very large initial error $e(0)$. This implies that $E(t)$ will get very small after one step of any algorithm. To avoid this large initial decrease, we have applied one step of MU on (U, V) to obtain reasonable initial estimates.

Table 4.2: Text mining datasets [156] (sparsity is given in %: $100 * \#\text{zeros}/(mn)$).

Data	m	n	#nonzero	sparsity	r	$\lfloor \rho_U \rfloor$	$\lfloor \rho_V \rfloor$
classic	7094	41681	223839	99.92	4, 8	12, 9	2, 1
sports	8580	14870	1091723	99.14	7, 14	18, 11	10, 6
reviews	4069	18483	758635	98.99	5, 10	35, 22	8, 4
hitech	2301	10080	331373	98.57	6, 12	25, 16	5, 4
ohscal	11162	11465	674365	99.47	10, 20	7, 4	7, 4
la1	3204	31472	484024	99.52	6, 12	31, 21	3, 2

We used exactly the same settings as for the image datasets (cf. Section 4.2.3); except for the factorization rank r which was set to to the value proposed in [156] (first entry in the sixth column of Table 4.2) and twice this value (second entry). For the comparison, we used the same settings as for the dense matrices. Figures 4.6, 4.7 and 4.8 display for each dataset the evolution of the average of functions $E(t)$ over all runs.

Again the accelerated algorithms are much more efficient. In particular, A-MU and A-PG now converge initially faster than ANLS, and often obtain better or similar final solutions. A-MU, HALS and A-HALS have the fastest initial convergence rate, and HALS and A-HALS generate the best solutions in most cases. Notice that A-HALS does not always perform significantly better than HALS (it does for the classic, sports and la1 datasets), the reason being that HALS already performs remarkably well. An explanation for that behavior is given in the next section.

4.2.4 Why does HALS perform (so) well?

It is well-known that (block) coordinate-descent methods typically fail to converge rapidly because of their zig-zagging behavior, similar to what is frequently observed for gradient descent approaches, see, e.g., [12]. However, for the NNLS subproblems arising in NMF, we have observed in the previous section that this approach (i.e., HALS) is quite efficient (see also [33,69,89,110]). In this section, we offer a theoretical explanation for that fact. This is based on two simple observations.

First, it is well-known that NMF solutions are typically part-based: this is the main reason why NMF has become so popular as a data analysis technique [105], see Chapter 1 and Figure 1.2 for an illustration (the reason being the nonnegativity constraints on both the basis elements and the weights leading to an additive reconstruction of the input data). Therefore the supports (the set of nonzero entries) of the columns of U (resp. rows of V) typically share few

elements. In other words, these supports are almost disjoint implying that the matrix product U^TU (resp. VV^T) has large entries on its diagonal, and zeros or small entries nearly everywhere else.

Second, the NNLS problem $\min_{U\geq 0} ||M-UV||_F^2$ can be decomposed into m independent NNLS subproblems corresponding to each row of U since

$$||M-UV||_F^2 = \sum_{i=1}^{m} ||M_{i:}-U_{i:}V||_F^2.$$

Each of these NNLS subproblems for a given row $U_{i:}$ has the following form

$$\min_{U_{i:}\geq 0} ||M_{i:}-U_{i:}V||_F^2 = U_{i:}(VV^T)U_{i:}^T - 2U_{i:}VM_{i:}^T + M_{i:}M_{i:}^T, \quad 1\leq i\leq m. \quad (4.11)$$

The quadratic term $U_{i:}(VV^T)U_{i:}^T$ coupling the variables $U_{i:}$ together depends on the (Hessian) matrix VV^T. In particular, if VV^T is diagonal, Problem (4.11) can be decoupled in r NNLS problems in one variable $\min_{U_{ik}\geq 0} ||M_{i:}-U_{ik}V_{k:}||_F^2$ for which an exact-coordinate descent method would generate an optimal solution in one step (i.e., after the update of each variable). We therefore have the following result.

Theorem 4.5. *Let $M\in\mathbb{R}_+^{m\times n}$ and $V\in\mathbb{R}_+^{m\times r}$. If the supports of the rows of V are disjoint, i.e., if VV^T is a diagonal matrix, then an optimal solution U^* to the NNLS problem*

$$\min_{U\geq 0} ||M-UV||_F^2, \quad \text{(NNLS)}$$

can be obtained by performing one iteration of the exact-coordinate descent method from any initial matrix, i.e., by a single HALS update.

More generally, if matrix VV^T is close to being diagonal, Problem (4.11) typically features large coefficients for the quadratic terms (i.e., U_{ik}^2) in comparison to the bilinear terms (i.e., $U_{ik}U_{ip}$ $k\neq p$). Intuitively, this implies that the interaction between variables is low and therefore optimizing one variable at a time is still a relatively efficient procedure.

Combining the above two observations, we conclude that performing few iterations of HALS on the NNLS subproblems arising in NMF allows the algorithm to get close to an optimum solution. This is especially true for sparse matrices M since the factors (U,V) will be even sparser, which gives an explanation for the similar performances of HALS and A-HALS for sparse matrices.

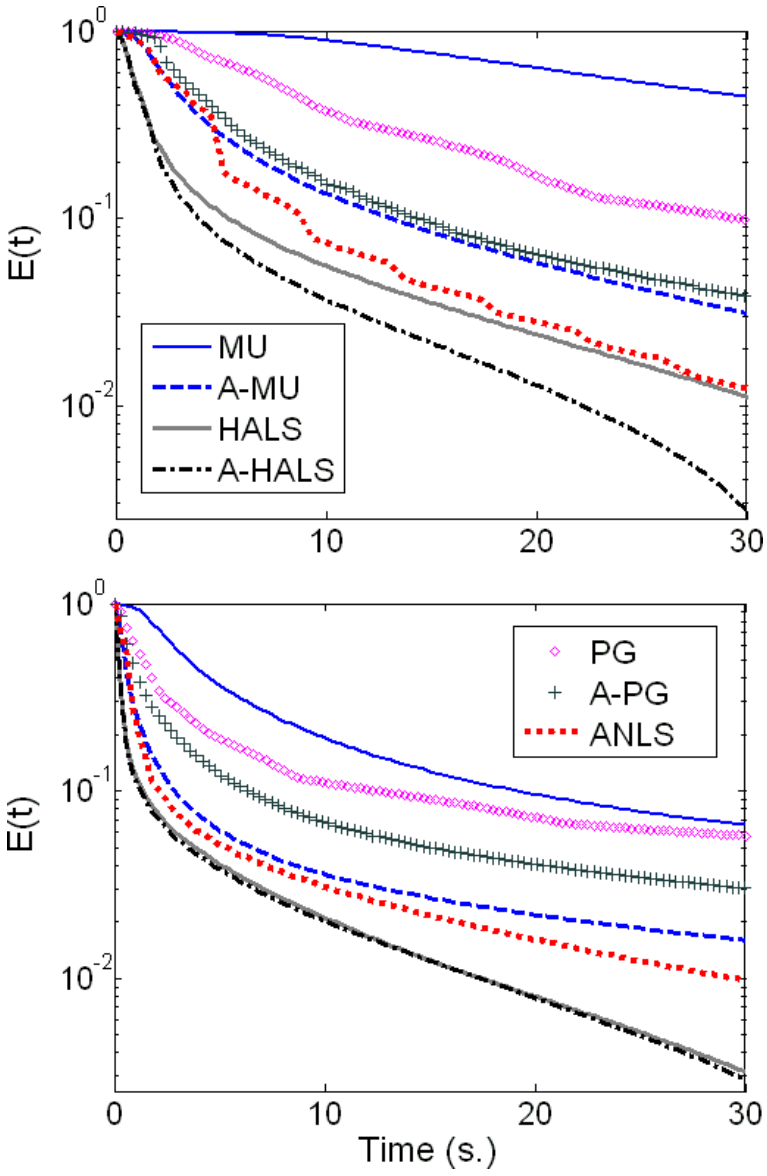

Figure 4.4: Average of functions $E(t)$ for different image datasets: ORL (top) and Frey (bottom).

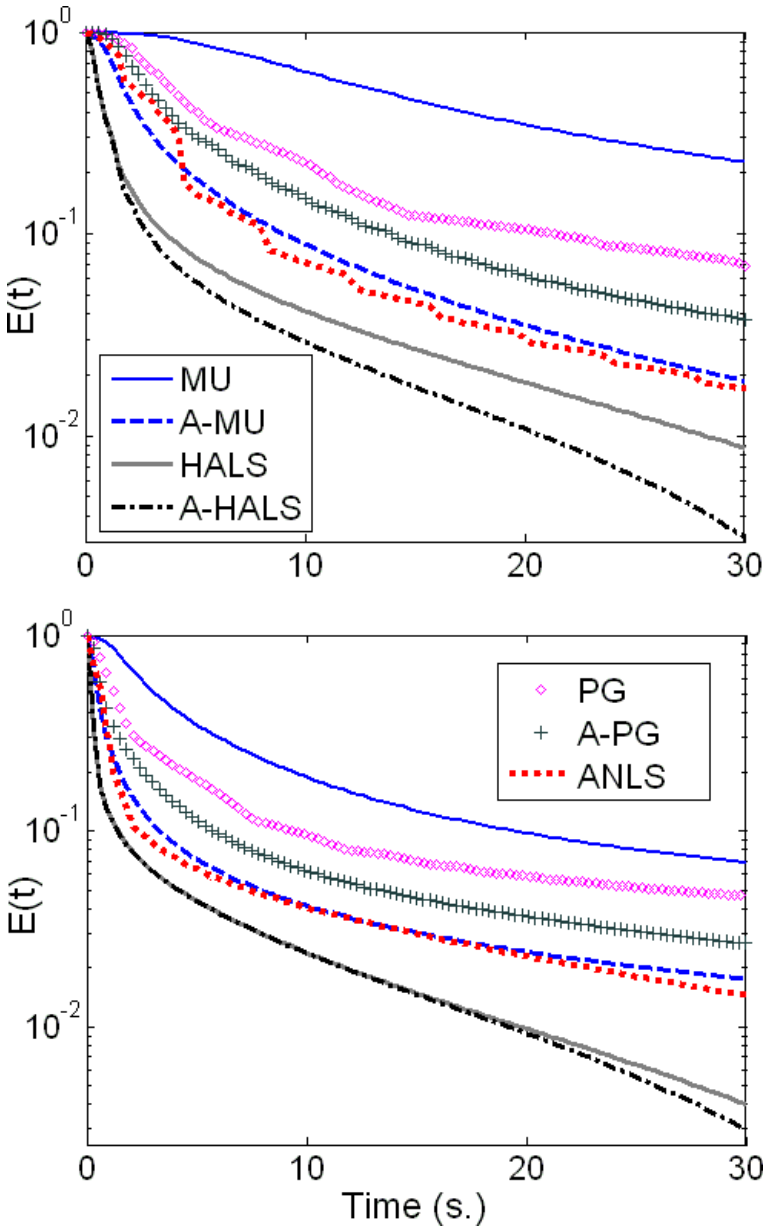

Figure 4.5: Average of functions $E(t)$ for different image datasets: Umist (top) and CBCL (bottom).

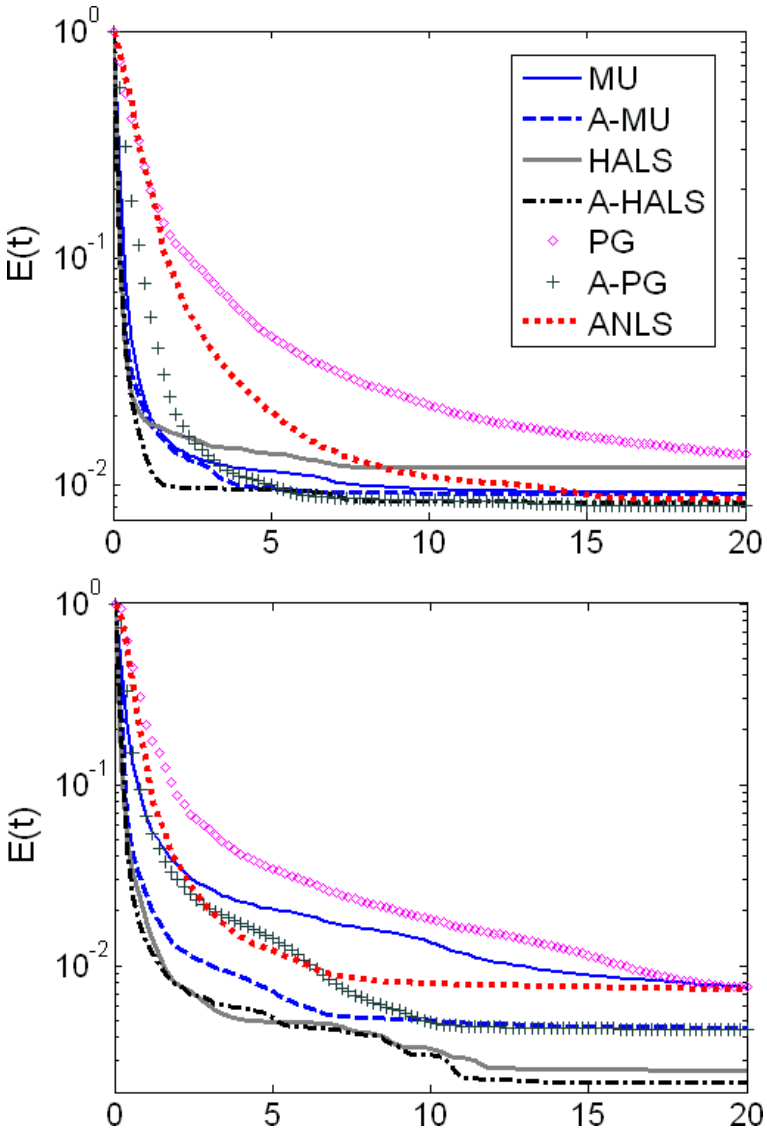

Figure 4.6: Average of functions $E(t)$ for text datasets: classic (top), reviews (bottom).

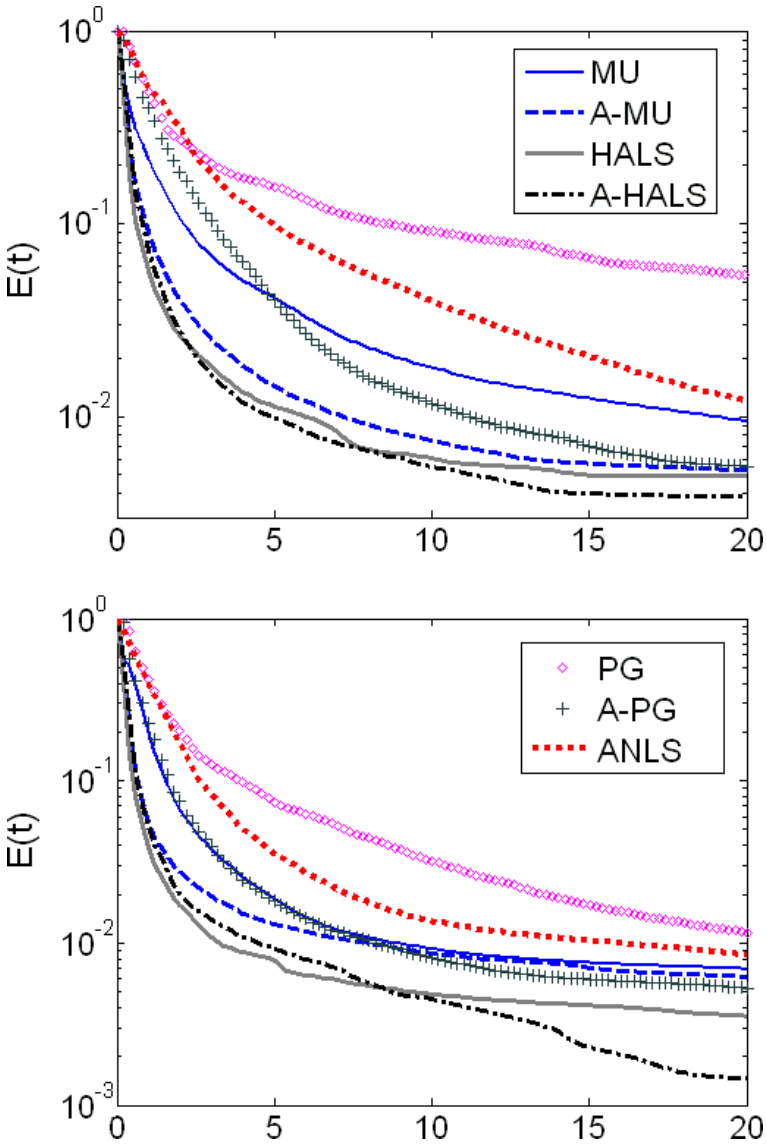

Figure 4.7: Average of functions $E(t)$ for text datasets: ohscal (top), sports (bottom).

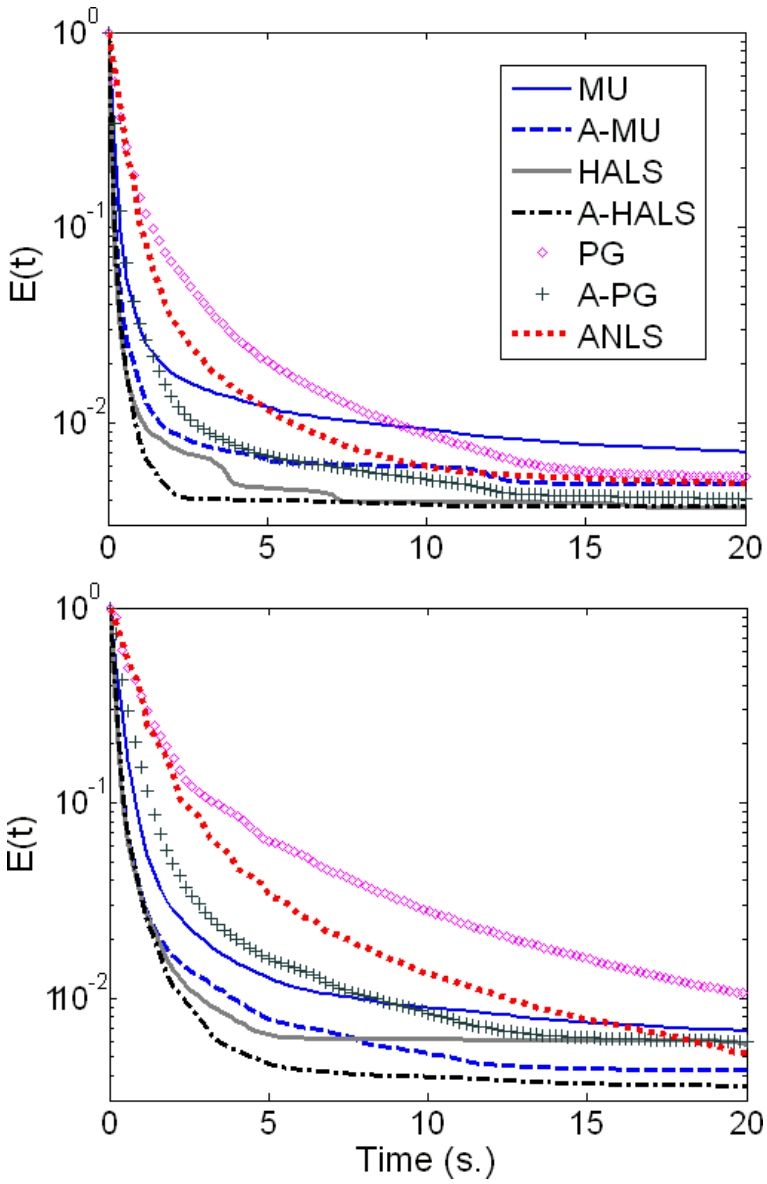

Figure 4.8: Average of functions $E(t)$ for text datasets: hitech (top) and la1 (bottom).

4.3 A Multilevel Approach

This section presents a general framework based on a multilevel strategy leading to faster convergence of NMF algorithms when dealing with data admitting some kind of simple approximate low-dimensional representations (based on linear transformations preserving nonnegativity), such as images. In fact, in these situations, a hierarchy of lower-dimensional problems can be constructed and used to compute efficiently approximate solutions of the original problem. Similar techniques have previously been used for other dimensionality reduction tasks such as PCA [134].

4.3.1 Multigrid Methods

Let us briefly introduce multigrid methods. The aim is to give the reader some insight on these techniques in order to comprehend their applications for NMF. We refer the reader to [21–23, 144] and references therein for detailed discussions on the subject.

Multigrid methods were initially used to develop fast numerical solvers for boundary value problems. Given a (partial) differential equation on a continuous domain with boundary conditions, the aim is to find an approximation of a *smooth* function f satisfying the constraints. In general, the first step is to discretize the continuous domain, i.e., choose a set of points (a *grid*) where the function values will be computed. Then, a numerical method (e.g., finite differences, finite elements) translates the continuous problem into a specific (square) system of linear equations:

$$\text{find } x \in \mathbb{R}^n \quad \text{s.t.} \quad Ax = b, \quad \text{with } A \in \mathbb{R}^{n \times n},\ b \in \mathbb{R}^n, \tag{4.12}$$

where the vector x will contain the approximate values of f on the grid points. Linear system (4.12) can be solved either by direct methods (e.g., Gaussian elimination) or iterative methods (e.g., Jacobi and Gauss-Seidel iterations). Of course, the computational cost of these methods depends on the number of points in the grid, which leads to a trade-off between precision (number of points used for the discretization) and computational cost.

Iterative methods update the solution at each step and hopefully converge to a solution of (4.12). Here comes the utility of multigrid: instead of working on a fine grid during all iterations, the solution is initially *restricted* to a coarser grid on which the iterations are cheaper. Moreover, the smoothness of function f allows to recover its low-frequency components faster on coarser grids. Solutions of the coarse grid are then *prolongated* to the finer grid and iterations can continue (higher frequency components of the error are reduced faster). Because the initial guess generated on the coarser grid is (hopefully)

a good approximation of the final solution, less iterations are needed on the fine (expensive) grid to converge. Essentially, multigrid methods make iterative methods more efficient, i.e., accurate solutions are obtained faster.

More recently, these same ideas have been applied to a broader class of problems, e.g., multiscale optimization with trust-region methods [81] and multiresolution techniques in image processing [142].

4.3.2 Description of a Multilevel Approach for NMF

The three algorithms presented in Section 4.1 (MU, ANLS and HALS) are iteratively trying to find a stationary point of NMF. Actually, most practical NMF algorithms are *iterative methods* (cf. the introduction of this chapter). In order to embed these algorithms into a multilevel strategy, one has to define the different levels and describe how the variables and the data is transferred between them.

Let each column of the matrix M be a element of the dataset (e.g., a vectorized image) belonging to $\mathbb{R}_+^m$. Given $m' < m$, we define a restriction operator $\mathcal{R}$ as a linear operator

$$\mathcal{R} : \mathbb{R}_+^m \to \mathbb{R}_+^{m'} : x \to \mathcal{R}(x) = Rx,$$

with $R \in \mathbb{R}_+^{m' \times m}$, and a prolongation $\mathcal{P}$ as a linear operator

$$\mathcal{P} : \mathbb{R}_+^{m'} \to \mathbb{R}_+^m : y \to \mathcal{P}(y) = Py,$$

with $P \in \mathbb{R}_+^{m \times m'}$. Nonnegativity of matrices R and P is a sufficient condition to preserve nonnegativity of the solutions when they are transfered from one level to another. In fact, in order to generate nonnegative solutions, one only need to require

$$\mathcal{R}(x) \geq 0, \forall x \geq 0 \text{ and } \mathcal{P}(y) \geq 0, \forall y \geq 0.$$

We also define the corresponding transfer operators on matrices, operating columnwise:

$$\mathcal{R}([x_1\, x_2 \ldots x_n]) = [\mathcal{R}(x_1)\, \mathcal{R}(x_2) \ldots \mathcal{R}(x_n)], \text{ and}$$

$$\mathcal{P}([y_1\, y_2 \ldots y_n]) = [\mathcal{P}(y_1)\, \mathcal{P}(y_2) \ldots \mathcal{P}(y_n)],$$

for $x_i \in \mathbb{R}_+^m, y_i \in \mathbb{R}_+^{m'}, 1 \leq i \leq n$.

In order for the multilevel strategy to work, the information lost when transferring from one level to another must be limited, i.e., the data matrix M has

to be well represented by $\mathcal{R}(M)$ in the lower dimensional space, which means that the reconstruction $\mathcal{P}(\mathcal{R}(M))$ must be close to M. From now on, we say that M is smooth with respect to $\mathcal{R}$ and $\mathcal{P}$ if and only if

$$s_M = \frac{||M - \mathcal{P}(\mathcal{R}(M))||_F}{||M||_F} \quad \text{is small .}$$

The quantity s_M measures how well M can be mapped by $\mathcal{R}$ into a lower-dimensional space and then brought back by $\mathcal{P}$, and still be a fairly good approximation of itself.

Based on these definitions, elaborating a multilevel approach for NMF is straightforward:

1. We are given $M \in \mathbb{R}_+^{m \times n}$ and $(U_0, V_0) \in \mathbb{R}_+^{m \times r} \times \mathbb{R}_+^{r \times n}$;

2. Compute $M' = \mathcal{R}(M) = RM \in \mathbb{R}_+^{m' \times n}$ and $U_0' = \mathcal{R}(U_0) = RU_0 \in \mathbb{R}_+^{m' \times r}$, i.e., restrict the elements of your dataset and the basis elements of the current solution to a lower-dimensional space;

3. Compute a rank-r NMF (U', V) of M' using (U_0', V_0) as initial matrices, i.e.,
$$U'V \approx M' = \mathcal{R}(M).$$
This can be done using any NMF iterative algorithm or, even better, using the multilevel strategy recursively (cf. Section 4.3.5).

4. Since
$$M \overset{(1)}{\approx} \mathcal{P}(\mathcal{R}(M)) = \mathcal{P}(M') \overset{(2)}{\approx} \mathcal{P}(U'V) = PU'V = \mathcal{P}(U')V = UV,$$
where U is computed as the prolongation of U', (U, V) is a good initial estimate for a rank-r NMF of M, *provided that* (1) M is smooth with respect to $\mathcal{R}$ and $\mathcal{P}$ (i.e., s_M is small) and (2) $U'V$ is a good approximation of $M' = \mathcal{R}(M)$ (i.e., $||M' - U'V||_F$ is small); in fact,
$$\begin{aligned} ||M - \mathcal{P}(U')V||_F &\leq ||M - \mathcal{P}(\mathcal{R}(M))||_F + ||\mathcal{P}(\mathcal{R}(M)) - \mathcal{P}(U'V)||_F \\ &\leq s_M||M||_F + ||\mathcal{P}(\mathcal{R}(M) - U'V)||_F \\ &\leq \underbrace{s_M||M||_F}_{(1)} + ||P||_F \underbrace{||\mathcal{R}(M) - U'V||_F}_{(2)}. \end{aligned}$$

5. Further improve the solution (U, V) using any NMF iterative algorithm.

Because computations needed at step 3 are relatively cheap (since $m' < m$), and in addition the low-frequency components of the error[9] are reduced faster on coarse levels (cf. Section 4.3.6), this strategy will accelerate the convergence of NMF algorithms.

4.3.3 Limitation of our Multilevel Approach

In classical multigrid methods, when solving a linear system of equations $Ax = b$, the current approximate solution x_c is not transferred from a fine level to a coarser one, because it would imply the loss of its high-frequency components; instead, the residual is transferred, which we briefly explain here. Defining the current residual $r_c = b - Ax_c$ and the error $e = x - x_c$, we have the equivalent defect equation $Ae = r_c$ and we would like to approximate e with a correction e_c in order to improve the current solution with $x_c \leftarrow x_c + e_c$. Hence the defect equation is solved approximately on the the coarser grid by restricting the residual r_c, the correction obtained on the coarser grid is interpolated and the new approximation $x_c + e_c$ is computed, see, e.g., [144, p.37]. If instead the solution is transferred directly from one level to another (as we do in this section), the corresponding scheme is in general not convergent, see [144, p.156]. In fact, even an exact solution of the system $Ax = b$ is not a fixed point, because the restriction of x is not an exact solution anymore at the coarser level (while, in that case, the residual r is equal to zero and the correction e will also be equal to zero).

Therefore, the method presented in this section should only be used as a *pre-processing/initialization* step before another (convergent) NMF algorithm. In fact, if one already has a good approximate solution (U, V) for NMF (e.g., a solution close to a stationary point), then transferring it to a coarser grid will most likely increase the approximation error because the high frequency components (such as edges in images) will be lost. Try to fix this drawback, which seems to be non-trivial, is a topic for further research. For example, it would be possible to use a 'local linearization' approach, consisting in linearizing the equation

$$R = M - (U + dU)(V + dV) \approx M - UV - UdV - dUV,$$

where dU and dV are the corrections to be computed on the coarser grids. However, several problems arise: how to take care of nonnegativity? or how to do this efficiently (in fact, computing the residual R is as expensive as computing directly a gradient direction on the fine grid with $\mathcal{O}(mnr)$ operations, see Section 4.2)?

Finally, it seems that, despite these theoretical reservations, our technique is still quite efficient (see Section 5.3.3). One intuitive reason for that good

[9] The low-frequency components refers to the parts of the data which are well-represented on coarse levels.

behavior is that NMF solutions are typically part-based and sparse (see Chapter 1 and Figure 1.2). Therefore, columns of matrix U contains relatively large 'constant components', made of their zero entries, which are perfectly transferred from one level to another, i.e., $s_U = \frac{||U-\mathcal{P}(\mathcal{R}(U))||_F}{||U||_F}$ will typically be very small (in general much smaller than s_M).

We now illustrate this technique on image datasets, more precisely, on two-dimensional gray-level images. In general, images are composed of several smooth components, i.e., regions where pixel values are similar and change continuously with respect to their location (e.g., skin on a face or, the pupil or sclera[10] of an eye), that is, a pixel value can often be approximated using the pixel values of its neighbors. This observation can be used to define the transfer operators (Section 4.3.4). For the computation of a NMF solution needed at step 3, the multilevel approach can be used recursively; three strategies (called multigrid cycles) are described in Section 4.3.5. Finally, numerical results are reported in Section 4.3.7.

4.3.4 Coarse Grid and Transfer Operators

A crucial step of multilevel methods is to define the different levels and the transformations (operators) between them. Figure 4.9 is an illustration of a standard *coarse grid* definition: we note I^1 the matrix of dimension $(2^a + 1) \times (2^b + 1)$ representing the initial image and I^l the matrix of dimension $(2^{a-l+1}+1)\times(2^{b-l+1}+1)$ representing the image at level l obtained by keeping, in each direction, only one out of every two points of the grid at the preceding level, i.e., I^{l-1}.

The transfer operators describe how to transform the images when going from finer to coarser levels, and vice versa, i.e., how to compute the values (pixel intensities) of the image I^l using values from image I^{l-1} at the finer level (restriction) or from image I^{l+1} at the coarser level (prolongation). For the *restriction*, the *full-weighting* operator is a standard choice: values of the coarse grid points are the weighted average of the values of their neighbors on the fine grid (see Figure 4.10 for an illustration). Noting $I^l_{i,j}$ the intensity of the pixel (i, j) of image I^l, it is defined as follows:

$$\begin{aligned} I^{l+1}_{i,j} = &\frac{1}{16}\Big[I^l_{2i-1,2j-1} + I^l_{2i-1,2j+1} + I^l_{2i+1,2j-1} + I^l_{2i+1,2j+1} \\ &+ 2(I^l_{2i,2j-1} + I^l_{2i-1,2j} + I^l_{2i+1,2j} + I^l_{2i,2j+1}) \\ &+ 4I^l_{2i,2j}\Big], \end{aligned} \tag{4.13}$$

[10]The white part of the eye.

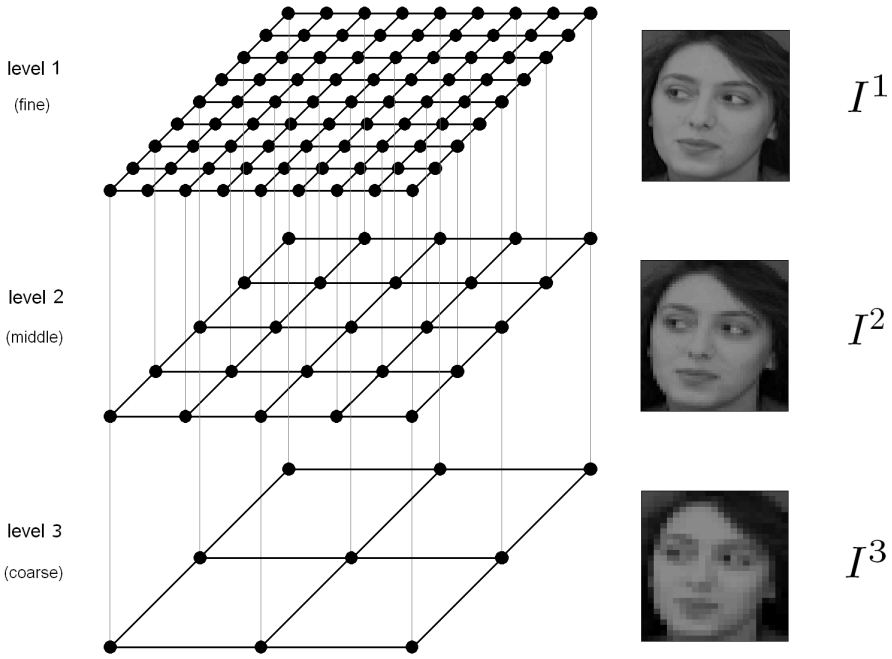

Figure 4.9: Multigrid Hierarchy. Schematic view of a grid definition for image processing (image from ORL face database, cf. Section 4.3.7).

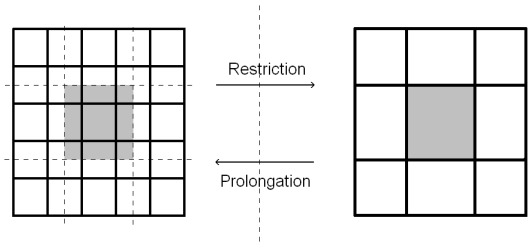

Figure 4.10: Restriction and Prolongation.

except on the boundaries of the image (when $i = 0$, $j = 0$, $i = 2^{a-l+1}$ and/or $j = 2^{b-l+1}$) where the weights are adapted correspondingly. For example, to

restrict a 3×3 image to a 2×2, $\mathcal{R}$ is defined with

$$R = \frac{1}{9} \begin{pmatrix} 4 & 2 & 0 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 4 & 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 & 4 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 0 & 2 & 4 \end{pmatrix},$$

(3×3 images needing first to be vectorized to vectors in $\mathbb{R}^9$, by concatenation of either columns or rows).

For the *prolongation*, we set the values on the fine grid points as the average of the values of their neighbors on the coarse grid:

$$I^l_{i,j} = \text{mean}_{\substack{i' \in \text{rd}(i/2) \\ j' \in \text{rd}(j/2)}} \left(I^{l+1}_{i',j'} \right), \tag{4.14}$$

where

$$\text{rd}(k/2) = \begin{cases} \{k/2\} & k \text{ even}, \\ \{(k-1)/2, (k+1)/2\} & k \text{ odd}. \end{cases}$$

For example, to prolongate a 2×2 image to a 3×3, $\mathcal{P}$ is defined with

$$P^T = \frac{1}{4} \begin{pmatrix} 4 & 2 & 0 & 2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 4 & 0 & 1 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 0 & 4 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 2 & 0 & 2 & 4 \end{pmatrix}.$$

Note that these transformations clearly preserve nonnegativity.

4.3.5 Multigrid Cycle

Now that grids and transfer operators are defined, we need to choose the procedure that is applied at each grid level as it moves through the grid hierarchy. In this section, we present three different approaches: nested iteration, V-cycle and full multigrid cycle.

In our setting, the transfer operators only change the number of rows m of the input matrix M, i.e., the number of pixels in the images of the database: the size of the images is approximatively four times smaller between each level: $m' \approx \frac{1}{4}m$. Since the computational complexity per iteration of the three algorithms (ANLS, MU and HALS) is almost proportional to m (cf. Section 4.2.1 and Appendix A), the iterations will be approximately four times cheaper. Therefore spending three quarter of the total time at the current level, and one quarter at the coarser levels leads to perform approximately the same number of iterations at each level (except at the coarsest one), which seems to give good results in practice. Table 4.3 shows the time spent and the corresponding number of iterations performed at each level.

	Level 1 (finer)	Level 2	... ...	Level $L-1$	Level L (coarser)	Total
# iterations	$3k$	$3k$	...	$3k$	$4k$	$(3L+1)k$
time	$\frac{3}{4}T$	$\frac{3}{16}T$	...	$\frac{3}{4^{L-1}}T$	$\frac{1}{4^{L-1}}T$	T

Table 4.3: Number of iterations performed and time spent at each level when allocating among L levels a total computational budget T corresponding to $4k$ iterations at the finest level.

Note that the transfer operators require $\mathcal{O}(mn)$ operations and since they are only performed once between each level, their computational cost can be neglected (at least for $r \gg 1$ and/or when a sizeable amount of iterations are performed).

Nested Iteration (NI)

To initialize NMF algorithms, we propose to factorize the image at the coarsest resolution and then use the solution as a initial guess for the next (finer) resolution. This is referred to as *nested iteration*, see Figure 4.11 for an illustration of the repartition of the iterations with three levels, Figure 4.12 for an illustration of the time spent at each level, and Algorithm 7 for the implementation. The idea is to start off the final iterations at the finer level with a better initial estimate, thus reducing the computational time required for the convergence of the iterative methods on the fine grid. The number of iterations and time spent at each level is chosen according to Table 4.3, i.e., one quarter of the time is alloted for iterations at the coarser levels followed by three quarters at the current level.

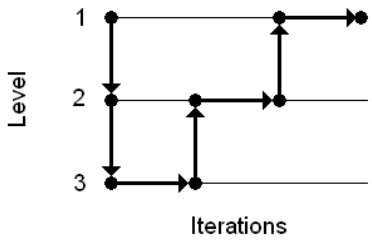

Figure 4.11: Nested Iteration. Transition between different levels for nested iteration.

Remark 4.2. When the ANLS algorithm is used, the prolongation of U' does

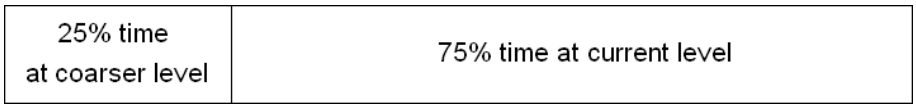

Figure 4.12: Nested Iteration. Time spent at current level.

Algorithm 7 Nested Iteration

Require: Number of levels $L \in \mathbb{N}$, data matrix $M \in \mathbb{R}_+^{m \times n}$, initial matrices $(U_0, V_0) \in \mathbb{R}_+^{m \times r} \times \mathbb{R}_+^{r \times n}$, and total time allocated to the algorithm $T \geq 0$.

1: **if** $L = 1$ **then**
2: $[U, V] = \text{NMF algorithm}(M, U_0, V_0, T)$;
3: **else**
4: $M' = \mathcal{R}(M)$; $U_0' = \mathcal{R}(U_0)$;
5: $[U', V] = \text{Nested Iteration}(L - 1, M', U_0', V_0, T/4)$;
6: $U = \mathcal{P}(U')$;
7: $[U, V] = \text{NMF algorithm}(M, U, V, 3T/4)$;
8: **end if**

not need to be computed since that algorithm only needs an initial value for one iterate. Note that this can be used in principle to avoid computing any prolongation, by setting U directly as the optimal solution of the NNLS problem $\min_{U \geq 0} ||M - UV||_F$.

V–Cycle (VC)

A drawback of nested iteration is that it does not take advantage of the smoothing properties of iterations on fine grids (high-frequency components of the error are reduced faster). It is therefore often more efficient to perform a few iterations at the fine level before going to coarser levels. The simplest choice is referred to as V-cycle and is illustrated on Figure 4.13 (with three levels) and Figure 4.14; see Algorithm 8 for the implementation. Time allocation is as follows: one quarter of the alloted time is devoted to iterations at the current level, followed by one quarter of the time for the recursive call to the immediately coarser level, and finally one half of the time again for iterations at the current level (we have therefore three quarters of the total time spent for iterations at current level, as for nested iteration).

Full Multigrid (FMG)

Combining ideas of nested iteration and V-cycle leads to a full multigrid cycle defined recursively as follows: at each level, a V-cycle is initialized with the

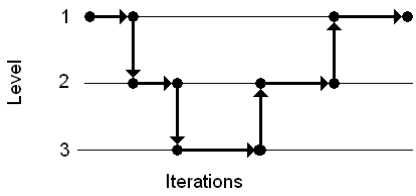

Figure 4.13: V-cycle. Transition between different levels for V-cycle.

25% time at current level	25% time at coarser level	50% time at current level

Figure 4.14: V-cycle. Time spent at current level.

Algorithm 8 V-cycle

Require: Number of levels $L \in \mathbb{N}$, data matrix $M \in \mathbb{R}_+^{m \times n}$, initial matrices $(U_0, V_0) \in \mathbb{R}_+^{m \times r} \times \mathbb{R}_+^{r \times n}$, and total time allocated to the algorithm $T \geq 0$.

if $L = 1$ **then**
  $[U, V] = \text{NMF algorithm}(M, U_0, V_0, T)$;
else
  $[U, V] = \text{NMF algorithm}(M, U_0, V_0, T/4)$;
  $M' = \mathcal{R}(M)$; $U' = \mathcal{R}(U)$;
  $[U', V] = \text{V-cycle}(L - 1, M', U', V, T/4)$;
  $U = \mathcal{P}(U')$;
  $[U, V] = \text{NMF algorithm}(M, U, V, T/2)$;
end if

solution obtained at the underlying level using a full-multigrid cycle. This is typically the most efficient multigrid strategy [144]. In this case, we propose to partition the time as follows (T is the total time): $\frac{T}{4}$ for the initialization (call of the full multigrid on the underlying level) and $\frac{3T}{4}$ for the V-cycle at the current level (see Figure 4.15 and Algorithm 9).

4.3.6 Smoothing Properties

We explained why the multilevel strategy was potentially able to accelerate iterative algorithms for NMF: cheaper computations and smoothing of the error on coarse levels. Before giving extensive numerical results in Section 4.3.7, we

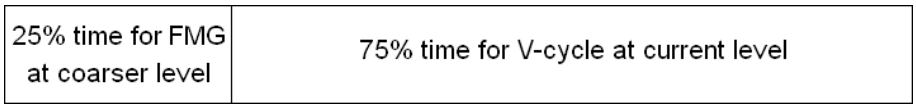

Figure 4.15: Full multigrid. Time spent at current level.

Algorithm 9 Full Multigrid

Require: Number of levels $L \in \mathbb{N}$, data matrix $M \in \mathbb{R}_+^{m \times n}$, initial matrices $(U_0, V_0) \in \mathbb{R}_+^{m \times r} \times \mathbb{R}_+^{r \times n}$, and total time allocated to the algorithm $T \geq 0$.

1: **if** $L = 1$ **then**
2: $[U, V] = \text{NMF algorithm}(M, U_0, V_0, T)$;
3: **else**
4: $U' = \mathcal{R}(U_0)$; $M' = \mathcal{R}(M)$; *
5: $[U', V] = \text{Full Multigrid}(L - 1, M', U', V_0, T/4)$;
6: $U = \text{prolongation}(U')$;
7: $[U, V] = \text{V-cycle}(L, M, U, V, 3T/4)$;
8: **end if**

*Note that the restrictions of M should be computed only once for each level.

illustrate this crucial feature of multilevel methods on the ORL face database.

Comparing three levels, Figure 4.16 displays the error (after prolongation to the fine level) for two faces and for different number of iterations (10, 50 and 100) using MU. Comparing the first row and the last row of Figure 4.16, it is clear that, in this example, the multilevel approach allows a significant smoothing of the error. Already after 10 iterations, the error obtained with the prolongated solution of the coarse level is smoother and smaller (see Figure 4.17) while it is computed much faster.

Figure 4.17 gives the evolution of the error with respect to the number of iterations performed (left) and with respect to computational time (right). In this example, the initial convergence after a given number of iteration on the three levels is comparable, while the computational cost to achieve a given accuracy is much cheaper on coarse levels. In fact, compared to the fine level, the middle (resp. coarse) level is approximately 4 (resp. 16) times cheaper.

4.3.7 Computational Results

To evaluate the performances of our multilevel approach, we present some numerical results for several standard image databases, see Table 4.3.7.

For each database, the multilevel strategy is tested using 100 runs initialized with the same random matrices for the three algorithms (ANLS, MU and

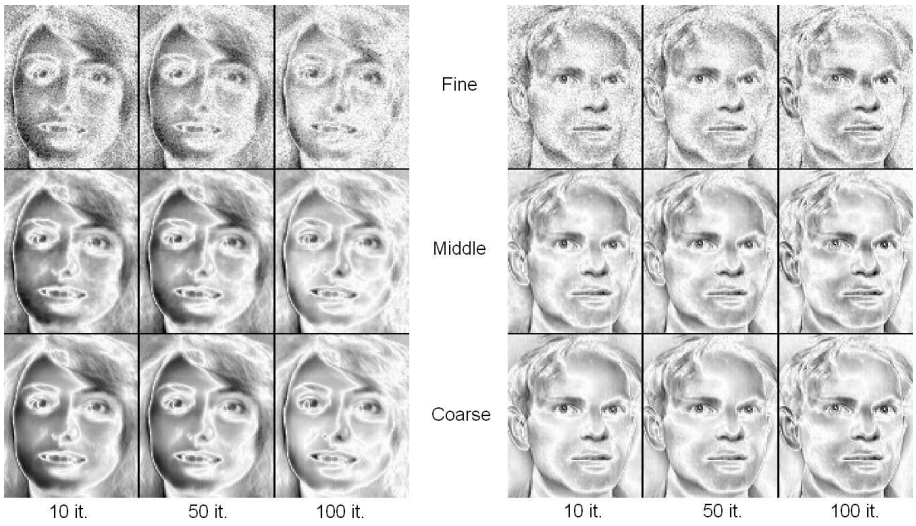

Figure 4.16: Smoothing on Coarse Levels. Example of the smoothing properties of the multilevel approach on the ORL face database. Each image represents the absolute value of the approximation error (black tones indicate a high error) of one of two faces from the ORL face database. These approximations are the prolongations (to the fine level) of the solutions obtained using the multiplicative updates on a single level, with $r = 40$ and the same initial matrices. From top to bottom: level 1 (fine), level 2 (middle) and level 3 (coarse); from left to right: 10 iterations, 50 iterations and 100 iterations.

Data	# pixels	m	n	r
ORL face[1]	112×92	10304	400	40
Umist face[2]	112×92	10304	575	40
Iris[3]	960×1280	1228800	8	4
Hubble Telescope [126]	128×128	16384	100	8

Table 4.4: Image datasets.

[1] *http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html*
[2] *http://www.cs.toronto.edu/∼roweis/data.html*
[3] *http://www.bath.ac.uk/elec-eng/research/sipg*

HALS) and the three multigrid cycles (NI, VC and FMG), with a time limit of 10 seconds. All algorithms have been implemented in MATLAB® 7.1 (R14) and tested on a 3GHz Intel® Core™2 Dual CPU PC.

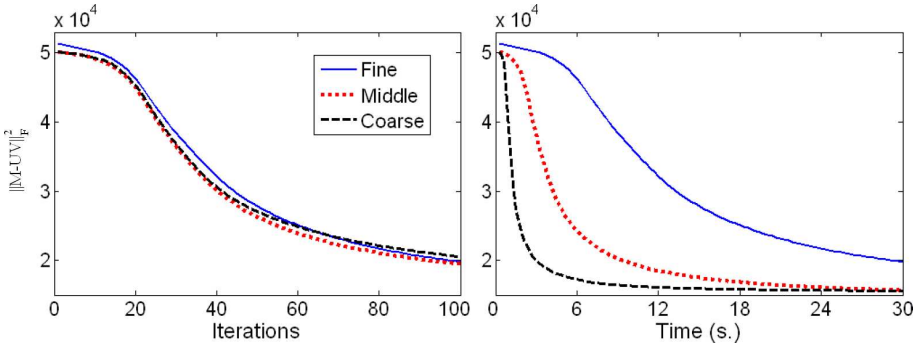

Figure 4.17: Evolution of the error on each level, after prolongation on the fine level, with respect to (left) the number of iterations performed and (right) the computational time. Same setting as in Figure 4.16.

Results

Tables 4.5, 4.6 and 4.7 give the mean error attained within 10 seconds using the different approaches.

	# lvl	ORL	Umist	Iris	Hubble
NMF	1	14960	26013	28934	24.35
NI	2	14683	25060	27834	**15.94**
	3	14591	24887	27572	16.93
	4	14580	24923	27453	17.20
VC	2	14696	25195	27957	16.00
	3	14610	24848	27620	16.12
	4	14599	24962	27490	16.10
FMG	2	14683	25060	27821	16.10
	3	14516	24672	27500	16.56
	4	**14460**	**24393**	**27359**	16.70

Table 4.5: Comparison of the mean error on the 100 runs with ANLS.

In all the cases, the multilevel approaches generate much better solutions than the original NMF algorithms; indicating that it is able to accelerate their convergence. The full multigrid cycle is, as expected, the best strategy while nested iteration and V-cycle give inferior performances (except for the Hubble images where NI with ANLS and VC for MU are better than FMG). We also observe

	# lvl	ORL	Umist	Iris	Hubble
NMF	1	34733	131087	64046	21.68
NI	2	23422	87966	37604	22.80
	3	20502	67131	33114	18.49
	4	19507	59879	31146	16.19
VC	2	23490	90064	36545	10.62
	3	20678	69208	32086	9.77
	4	19804	62420	30415	**9.36**
FMG	2	23422	87966	37504	22.91
	3	19170	58469	32120	15.06
	4	**17635**	**46570**	**29659**	11.71

Table 4.6: Comparison of the mean error on the 100 runs with MU.

	# lvl	ORL	Umist	Iris	Hubble
NMF	1	15096	27544	31571	17.97
NI	2	14517	25153	29032	17.37
	3	14310	24427	28131	16.91
	4	14280	24256	27744	16.92
VC	2	14523	25123	28732	17.37
	3	14339	24459	28001	17.02
	4	14327	24364	27670	17.04
FMG	2	14518	25153	29120	17.39
	3	14204	23950	27933	**16.69**
	4	**14107**	**23533**	**27538**	16.89

Table 4.7: Comparison of the mean error on the 100 runs with HALS.

that the additional speed up of the convergence when the number of levels is increased from 3 to 4 is less significant; the final error achieved is even increased in some cases. In general, the 'optimal' number of levels will depend on the size and the smoothness of the data.

HALS combined with the full multigrid cycle is one of the best strategies. Figure 4.18 displays the distribution of the errors for the different databases in this particular case. For the ORL and Umist databases, the multilevel strategy is extremely efficient: all the solutions generated with 2 and 3 levels are better than the original NMF algorithm. For the Iris and Hubble databases, the difference is not as clear. The reason is that the corresponding NMF problems are 'easier' because the rank r is smaller. Hence the algorithms converge faster to stationary points, and the distribution of the final errors is more concentrated.

In order to visualize the evolution of the error through the iterations, Figure 4.19 displays the evolution of the objective function with respect to the

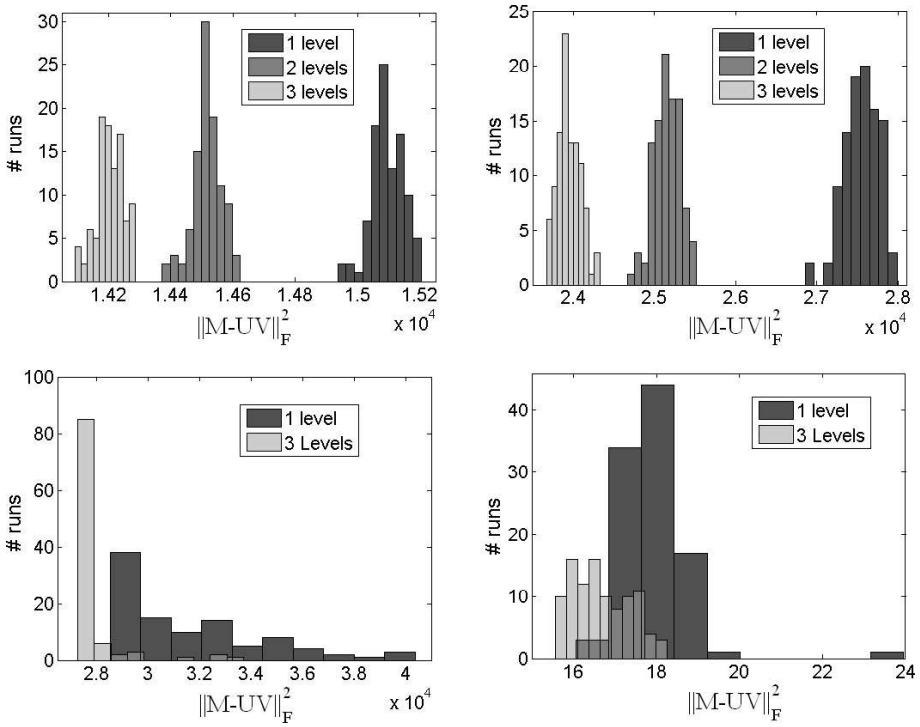

Figure 4.18: Distribution of the error among the 100 random initializations using the HALS algorithm with a full multigrid cycle: (top left) ORL, (top right) Umist, (bottom left) Iris, and (bottom right) Hubble.

number of iterations independently for each algorithm and each database using nested iteration as the multigrid cycle (which is the easiest to represent). In all the cases, the prolongations of the solutions from the lower levels generate much better solutions that the one obtained on the fine level.

These test results are very encouraging: the multilevel approach for NMF seems very efficient and allows to speed up convergence of algorithms significantly.

4.3.8 Extensions of the Multilevel Approach

Generalization

We have only used our multilevel approach for a specific objective function (sum of squared errors) to speed up three NMF algorithms (ANLS, MU and

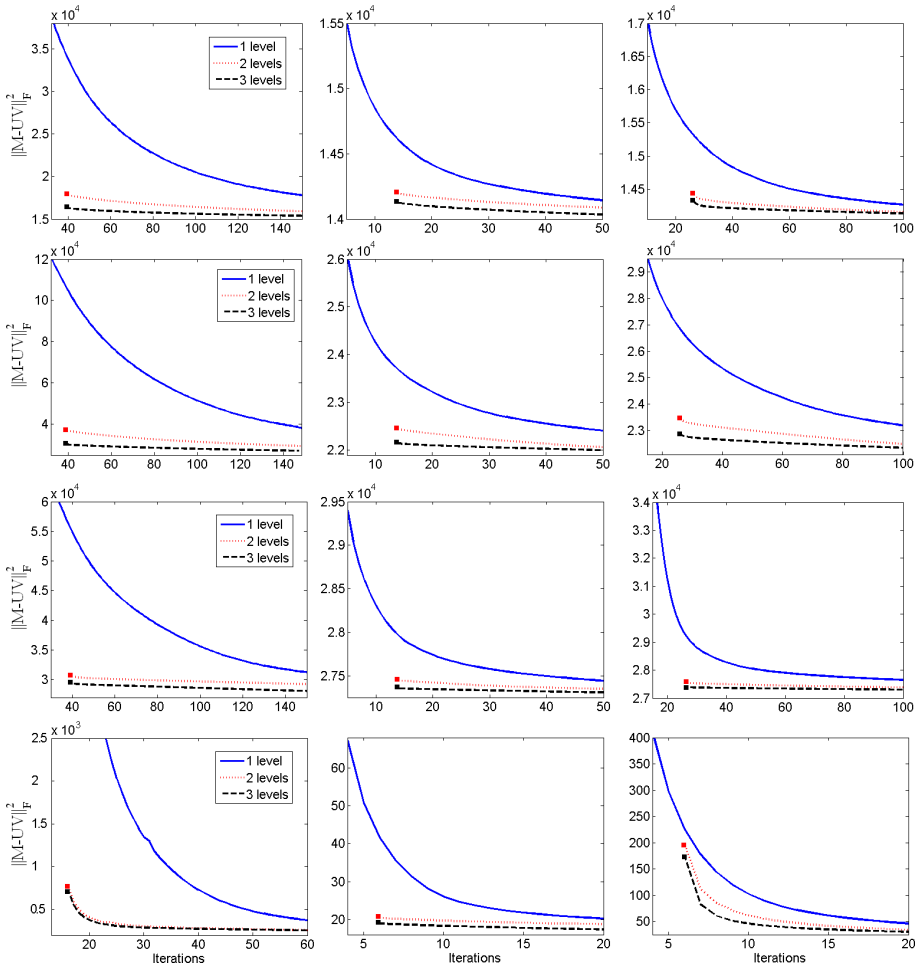

Figure 4.19: Evolution of the objective function. From left to right : MU, ANLS and HALS. From top to bottom: ORL, Umist, Iris and Hubble databases. *1 level* stands for the standard NMF algorithms. The initial points for the curves *2 levels* and *3 levels* are the prolongated solutions obtained on the coarser levels using nested iteration, cf. Section 4.3.5. All algorithms were initialized with the same random matrices.

HALS) and to factorize 2D images. However, this can be easily generalized to other objective functions, other iterative algorithms and applied to other kind of smooth data. Moreover, other types of coarse grid definition, transfer operators and grid cycle can be used and could potentially improve efficiency.

A limitation of the proposed approach is that the multigrid strategy is only applied to one dimension of the matrix (because we did not assume that the different images are related to each other in any way). However, in some applications, rows of matrix M might also be restricted to lower dimensional spaces. For example, in hyperspectral data analysis, each column of matrix M represents an image at a given wavelength, while each row represents the spectral signature of a pixel, see Chapter 7. Since spectral signatures feature smooth components as well, the multilevel strategy can be used to reduce both dimensions of the data matrix.

This idea can also be extended to nonnegative tensor factorization (NTF) (see, e.g., [153] and references therein where it is used to analyze the hyperspectral Hubble telescope images) by using multilevel techniques for higher dimensional spaces.

Initialization

Several judicious initializations for NMF algorithms have been proposed in the literature and allow to speed up convergence and improve, in general, the final solution [18, 41]. The computational cost of these good initial guesses depends on the matrix dimensions and will then be cheaper to compute on the coarsest grid. Therefore, it would be interesting to combine classical NMF initializations techniques with our multilevel approach for further speedups.

Unstructured data

A priori, applying a multilevel method to data for which we do not have any information about the matrix to factorize (and a fortiori about the solution) seems out of reach. In fact, in these circumstances, there is no sensible way to define the transfer operators.

However, it is not hopeless to extend the multilevel idea to other type of data. For example, in text mining applications, the term-by-document matrix could be restricted by stacking synonyms or similar texts together (similarly as in [134]). Of course, this implies some a priori knowledge or preprocessing of the data (which should be cheap enough to be profitable).

Conclusion

In this chapter, we have described three of the most used NMF algorithms: MU, ANLS and HALS. We have then proposed a modification of MU and HALS, motivated by the analysis of the computations needed at each iteration of these algorithms. It was experimentally shown that these modified versions outperform the original ones, and compare favorably with a state-of-the-art algorithm, namely the ANLS method of Kim and Park [99]. HALS and its

accelerated version are the most efficient variants for solving NMF problems, sometimes from afar. Besides the extensive numerical experiments, we have given a theoretical explanation for that fact, see Section 4.2.4. The reason is that NMF solutions are expected to be parts-based, i.e., in a decomposition $M \approx UV$ the supports of the columns of U (resp. rows of V) will be 'almost' disjoint, and an exact-coordinate descent method such as HALS allows to solve the nearly separable NNLS subproblems efficiently.

We have also proposed a multilevel approach to speed up NMF algorithms whose efficiency was experimentally demonstrated.

Chapter 5

Nonnegative Factorization

In the special case where we seek a rank-one factorization (i.e., when $r = 1$), NMF is known to be polynomially solvable, cf. Section 2.2.2 and Chapter 3. The central problem studied in this chapter, called rank-one nonnegative factorization (R1NF), is an extension of rank-one NMF where the matrix to be approximated by the outer product of two nonnegative vectors is now allowed to contain negative elements.

R1NF is introduced in Section 5.1, where it is shown that allowing negative elements in the matrix transforms the polynomially solvable rank-one NMF problem into a $\mathcal{NP}$-hard problem. The reduction used in the proof is based on the problem of finding a maximum-edge biclique in a bipartite graph. Because any algorithm designed to solve NMF must at least implicitly solve R1NF problems, this hardness result sheds new light on the limitations of NMF algorithms and the complexity of NMF when the factorization rank r is fixed.

In Section 5.2, stationary points of the R1NF problem used in the abovementioned reduction are shown to coincide with bicliques of the corresponding graph. Building on that fact, Section 5.3 introduces a new type of biclique finding algorithm that relies on the application of a simple nonlinear optimization scheme (exact block-coordinate descent, similar as HALS) to the equivalent R1NF problem considered earlier, which only requires for each iteration a number of operations proportional to the number of edges of the graph. This method is then compared to a greedy heuristic and an existing algorithm on some synthetic and text mining datasets, and is shown to perform competitively.

Finally, in Section 5.4, the multiplicative updates for NMF are generalized for matrices M with negative entries. This allows us to comprehend the differences between the MU and HALS algorithms for NMF, and give an explanation for the better performance of HALS.

5.1 Rank-one Nonnegative Factorization (R1NF)

Solving NMF amounts to finding r nonnegative rank-one factors $U_{:k}V_{k:}$, each having to satisfy the following equality as well as possible

$$U_{:k}V_{k:} \approx M - \sum_{i \neq k} U_{:i}V_{i:} \doteq R_k \not\geq \mathbf{0} \quad \forall k,$$

i.e., each of them should be the best possible nonnegative rank-one approximation of the corresponding residual matrix, denoted R_k. It is important to notice here that, unlike input matrix M, matrices R_k can contain negative elements. Therefore, any NMF algorithm has to solve, at least implicitly, the following subproblems

$$\min_{U_{:k} \in \mathbb{R}^m, V_{k:} \in \mathbb{R}^n} ||M - UV||_F^2 = ||R_k - U_{:k}V_{k:}||_F^2 \quad \text{such that } U_{:k} \geq 0, V_{k:} \geq 0, \tag{5.1}$$

for each k. We may wonder whether these subproblems can be solved efficiently, i.e., ask ourselves

> Is it possible to compute in polynomial time the best rank-one nonnegative approximation of a matrix which is not necessarily nonnegative?

A first observation is that computing the globally optimal value of $U_{:k}$ for a given value of $V_{k:}$ can be done in closed-form (and similarly for computing the optimal value of $V_{k:}$ for a fixed $U_{:k}$), and this constitutes an efficient strategy to solve NMF; this is the basis for the HALS algorithm, see Chapter 4.

5.1.1 Definition of R1NF and Implications for NMF

In order to shed some light on the above question, we define the problem of rank-one nonnegative factorization[1] (R1NF) to be the variant of rank-one NMF where the matrix to be factorized can be *any real matrix*, i.e., is not necessarily nonnegative. Formally, given an $m \times n$ real matrix $R \in \mathbb{R}_+^{m \times n}$, one has to find a pair of nonnegative vectors $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$ such that the nonnegative rank-one product uv^T is the best possible approximation (in the Frobenius norm) of matrix R:

$$\min_{u \in \mathbb{R}^m, v \in \mathbb{R}^n} ||R - uv^T||_F^2 \quad \text{such that } u \geq 0, v \geq 0. \tag{R1NF}$$

The next subsection shows that, in contrast with standard rank-one NMF, this problem is $\mathcal{NP}$-hard, which provides the following new insights about the NMF problem:

[1] This terminology has already been used for the problem of finding a symmetric nonnegative factorization, i.e., one where U=V, see, e.g., [87]. Also, in Chapter 3, a rank-k nonnegative factorization referred to an exact approximation of the factorized matrix. In this chapter, we assign it a different meaning.

- ◇ We cannot expect to be able to solve subproblems (5.1) in polynomial time up to global optimality, and the HALS algorithm most probably cannot be improved with a better scheme for successively computing rank-one factors $U_{:k}V_{k:}$ arising in (5.1). More generally, any algorithm for NMF cannot expect to solve at each iteration a subproblem where a given column of $U_{:k}$ and its corresponding row $V_{k:}$ are to be optimized simultaneously. This shows that, in that sense, the partition of variables for block-coordinate schemes such as alternative nonnegative least squares (ANLS, optimizing U and V alternatively) and (implicitly) HALS is best possible.
- ◇ Recall that the $\mathcal{NP}$-hardness result characterizing NMF requires both the dimensions of matrix M and the factorization rank r of M to increase, and that the complexity of NMF for a fixed rank r is currently not known (except in the polynomially solvable rank-one case), see Chapter 3. Our hardness result on (R1NF) therefore suggests that NMF is also a difficult problem for any fixed rank $r \geq 2$. Indeed, even if one was given the optimal solution of a NMF problem except for a single rank-one factor, it is not guaranteed that one would be able to find this last factor in polynomial time, since the corresponding residual matrix is not necessarily nonnegative.

5.1.2 Complexity of R1NF and the Biclique Problem

In this section, we show how the optimization version of the maximum-edge biclique problem (MB) can be formulated as a specific rank-one nonnegative factorization problem (R1NF-MB). Since the decision version of (MB) is $\mathcal{NP}$-complete [127], this implies that rank-one nonnegative factorization (R1NF) is in general $\mathcal{NP}$-hard.

The Maximum-Edge Biclique Problem in Bipartite Graphs

A *bipartite graph* G_b is a graph whose vertices can be divided into two disjoint sets V_1 and V_2 such that there is no edge between two vertices in the same set

$$G_b = (V, E) = \Big(V_1 \cup V_2, E \subseteq (V_1 \times V_2)\Big).$$

A *biclique* K_b is a complete bipartite graph, i.e., a bipartite graph where all the vertices are connected

$$K_b = (V', E') = \Big(V_1' \cup V_2', E' = (V_1' \times V_2')\Big).$$

The so-called maximum-edge biclique problem in a bipartite graph $G_b = (V, E)$ is the problem of finding a biclique $K_b = (V', E')$ in G_b (i.e., $V' \subseteq V$ and $E' \subseteq E$) maximizing the number of edges $|E'| = |V_1'||V_2'|$. The decision problem:

Given B, does G_b contain a biclique with at least B edges?

has been shown to be $\mathcal{NP}$-complete [127], and the corresponding optimization problem is at least $\mathcal{NP}$-hard.

Let $M_b \in \{0,1\}^{m\times n}$ be the biadjacency matrix of the unweighted bipartite graph $G_b = (V_1 \cup V_2, E)$ with $V_1 = \{s_1, \dots s_m\}$ and $V_2 = \{t_1, \dots t_n\}$, i.e., $M_b(i,j) = 1$ if and only if $(s_i, t_j) \in E$. We denote by $|E|$ the cardinality of E, i.e., the number of edges in G_b; note that $|E| = ||M_b||_F^2$. The set of indices of zero values in M_b will be denoted $Z = \{(i,j) \,|\, M_b(i,j) = 0\}$, and its cardinality $|Z|$, with $|E| + |Z| = mn$. With this notation, the maximum biclique problem in G_b can be formulated as

$$\begin{array}{ll} \min\limits_{u,v} & ||M_b - uv^T||_F^2 \\ & uv^T \leq M, \\ & u \in \{0,1\}^m, v \in \{0,1\}^n, \end{array} \qquad \text{(MB)}$$

where $u_i = 1$ (resp. $v_j = 1$) means that node s_i (resp. t_j) belongs to the solution, $u_i = 0$ (resp. $v_j = 0$) otherwise. The constraint $uv^T \leq M$ guarantees feasible solutions of (MB) to be bicliques of G_b. In fact,

$$M_{ij} = 0 \quad \Rightarrow \quad u_i = 0 \;\text{ or }\; v_j = 0,$$

i.e., if there is no edge between s_i and t_j, they cannot belong to the same biclique. One can also check easily that the objective is equivalent to $\max \sum_{ij} u_i v_j$ since M_b, u and v are binary: instead of maximizing the number of edges inside the biclique, one minimizes the number of edges outside.
Feasible solutions of (MB) correspond to bicliques of G_b. We will be particularly interested in *maximal* bicliques, which are bicliques not contained in a larger biclique.

Complexity of R1NF

Let us define the following rank-one nonnegative factorization problem

$$\min_{u\in\mathbb{R}^m, v\in\mathbb{R}^n} ||M_d - uv^T||_F^2 \quad \text{such that} \quad u \geq 0,\ v \geq 0 \qquad \text{(R1NF-MB)}$$

where M_d is the matrix M_b where the zero values have been replaced by $-d$, i.e.,

$$(M_d)_{ij} = \left\{ \begin{array}{ll} 1 & \text{if } (M_b)_{ij} = 1 \\ -d & \text{if } (M_b)_{ij} = 0 \end{array} \right. , \ 1 \leq i \leq m, 1 \leq j \leq n,$$

where $d \geq 1$ is a parameter. Although (R1NF-MB) is a continuous optimization problem, we are going to show that, for a sufficiently large value of d, any of

its optimal solutions has to *coincide* with a binary optimal solution of the corresponding (discrete) biclique problem (MB), which will then imply $\mathcal{NP}$-hardness of (R1NF).
Intuitively, if a $-d$ entry of M_d is approximated by a positive value, say p, the corresponding term in the squared Frobenius norm of the error is $d^2+\mathbf{2pd}+p^2$. As d increases, it becomes more and more costly to approximate $-d$ by a positive number and we will show that, for d is sufficiently large, negative values of M_d have to be approximated by zeros. Since the remaining values (not approximated by zeros) are all ones, the optimal rank-one solution will be binary.

From now on, we say that a solution (u, v) *coincides* with another solution (u', v') if and only if $uv^T = u'v'^T$, i.e., if and only if $u' = \lambda u$ and $v' = \lambda^{-1}v$ for some $\lambda > 0$.

We note M_+ the positive part of matrix M, with $(M_+)_{ij} = \max(0, M_{ij})$ and M_- the negative part of matrix M, with $(M_-)_{ij} = \max(0, -M_{ij})$ so that $M = M_+ - M_-$. We also note $\min(M)$ the minimum element of matrix M, $\min(M) = \min_{ij}(M_{ij})$.

Lemma 5.1. *Any optimal rank-one approximation with respect to the Frobenius norm (see Problem* (LRA-1) *in Section 2.2) of a matrix M for which $\min(M) \leq -||M_+||_F$ contains at least one nonpositive entry.*

Proof. If $M = 0$, the result is trivial. If not, we have $\min(M) < 0$ since $\min(M) \leq -||M_+||_F$. Suppose now $(u, v) > 0$ is a best rank-one approximation of M. Therefore, since the negative values of M are approximated by positive ones and since M has at least one negative entry, we have

$$||M - uv^T||_F^2 > ||M_-||_F^2. \tag{5.2}$$

By the Eckart-Young theorem, the optimal rank-one approximation uv^T must satisfy (see Section 2.2)

$$||M - uv^T||_F^2 \geq ||M||_F^2 - \sigma_{max}(M)^2 = ||M||_F^2 - ||M||_2^2 \,.$$

Clearly,

$$||M||_F^2 = ||M_+||_F^2 + ||M_-||_F^2 \quad \text{and} \quad ||M||_2^2 \geq \min(M)^2,$$

so that we can write

$$||M - uv^T||_F^2 \leq ||M_+||_F^2 + ||M_-||_F^2 - \min(M)^2 \leq ||M_-||_F,$$

which is in contradiction with (5.2). □

Before stating the main result of this Chapter, about the equivalence of (R1NF-MB) and (MB), let us recall that the local minima of the best rank-one approximation problem with respect to the Frobenius norm are global minima (see Theorem 2.5).

Theorem 5.2. *For $d \geq \sqrt{|E|}$, any optimal solution (u,v) of* (R1NF-MB) *coincides with an optimal solution of* (MB)*, i.e., uv^T is binary and $uv^T \leq M_b$.*

Proof. We focus on the entries of uv^T which are positive and define their support as

$$K = \Big\{ i \in \{1, 2, \ldots, m\} \;\Big|\; u_i > 0 \Big\} \quad \text{and} \quad L = \Big\{ j \in \{1, 2, \ldots, n\} \;\Big|\; v_j > 0 \Big\}. \tag{5.3}$$

We also define $u' = v(K)$, $v' = w(L)$ and $M'_d = M_d(K, L)$ to be the subvectors and submatrix with indexes in K, L and $K \times L$. Since (u, v) is optimal for M_d, (u', v') must be optimal for M'_d. Suppose there is a $-d$ entry in M'_d, then

$$\min(M'_d) = -d \leq -\sqrt{|E|} = -||(M_d)_+||_F \leq -||(M'_d)_+||_F,$$

so that Lemma 5.1 holds for M'_d. Since (u', v') is positive (i.e., it is located inside the feasible domain) and is an optimal solution of (R1NF-MB) for M'_d, it is a local minimum of problem without the nonnegativity constraints, i.e., the problem (LRA-1) of best rank-one approximation. By Theorem 2.5, this must be a global minimum. This is a contradiction with Lemma 5.1: (u', v') should contain at least one nonpositive entry. Therefore M'_d does not contain any $-d$ entry, and we have $M'_d = \mathbf{1}_{|K|\times|L|}$ which implies than $u'v'^T = M'_d$ by optimality (it is the unique rank-one solution $u'v'^T$ with objective value equals to zero) and finally allows to conclude that uv^T is binary and $uv^T \leq M_b$. □

We have just proven the following theorem:

Theorem 5.3. *Rank-one nonnegative factorization* (R1NF) *is $\mathcal{NP}$-hard.*

5.2 Stationary Points

We have shown that optimal solutions of (R1NF-MB) coincide with optimal solutions of (MB) for $d \geq \sqrt{|E|}$, whose computation is $\mathcal{NP}$-hard. In this section, we focus on stationary points of (R1NF-MB) instead: we show how they are related to the feasible solutions of (MB). This result will be used in Section 5.3 to design a new type of biclique finding algorithm.

5.2.1 Definitions and Notations

The pair (u, v) is a stationary point for problem (R1NF-MB) if and only if it satisfies its first-order optimality conditions (cf. Section 2.1), i.e., if and only if

$$u \geq 0, \quad \mu = (uv^T - M_d)v \quad \geq 0 \quad \text{and} \quad u \circ \mu = 0, \tag{5.4}$$

$$v \geq 0, \quad \lambda = (uv^T - M_d)^T u \quad \geq 0 \quad \text{and} \quad v \circ \lambda = 0. \tag{5.5}$$

Of course, we are only interested in nonzero solutions and, assuming that $u \neq 0$ and $v \neq 0$, one can check that conditions (5.4)-(5.5) are equivalent to

$$u = \max\left(0, \frac{M_d v}{||v||_2^2}\right) \qquad \text{and} \qquad v = \max\left(0, \frac{M_d^T u}{||u||_2^2}\right). \tag{5.6}$$

We define three sets of rank-one matrices:

1. Given a positive real number d, S_d is the set of nonzero stationary points of (R1NF-MB), i.e.,

$$S_d = \{uv^T \in \mathbb{R}^{m\times n} \mid (u, v) \text{ satisfies (5.4) and (5.5)}\} ;$$

2. F is the set of feasible solutions of (MB), i.e.,

$$F = \{uv^T \in \mathbb{R}^{m\times n} \mid (u, v) \text{ is a feasible for (MB)}\},$$

3. B is the set of maximal bicliques of (MB), i.e., $uv^T \in B$ if and only if $uv^T \in F$ and uv^T coincides with a maximal biclique.

5.2.2 Stationarity of Maximal Bicliques

The next theorem states that, for d sufficiently large, the only nonzero feasible solutions of (MB) that are stationary points of (R1NF-MB) are the maximal bicliques.

Theorem 5.4. *For $d > \max(m, n) - 1$, $F \cap S_d = B$.*

Proof. If $M = 0$ the result is trivial. Otherwise, let us show that $uv^T \in B$ if and only if $uv^T \in F$ and $uv^T \in S_d$. By definition, uv^T belongs to B if and only if uv^T belongs to F and is maximal, i.e.,

(*) $\nexists i$ such that $u_i = 0$ and $M_d(i, j) = 1, \forall j$ s.t. $v_j \neq 0$,

(**) $\nexists j$ such that $v_j = 0$ and $M_d(i, j) = 1, \forall i$ s.t. $u_i \neq 0$.

Since uv^T is binary and $u \neq 0$, the nonzero entries of v must be equal to each other. Noting L the support of v (see Equation (5.3)), we then have

$$v_j = \frac{||v||_1}{|L|} = C, \ \forall\, j \in L,$$

for some $C \in \mathbb{R}_+$. Moreover, $d > \max(m, n) - 1$ so that (*) is equivalent to

$$\begin{array}{ccccc} \nexists\, i & \text{such that} & u_i = 0 & \text{and} & M_d(i,:)v > 0 \\ & & \Longleftrightarrow & & \\ u_i = 0 \Rightarrow M_d(i,:)v \leq 0 & \text{and} & u_i \neq 0 \Rightarrow u_i = \frac{1}{C} = \frac{||M_d(i,:)||_1}{||v||_1} = \frac{M_d(i,:)v}{||v||_2^2}. & & \end{array}$$

These are exactly the stationarity conditions for $v \neq 0$, cf. Equation (5.6). By symmetry, (**) is equivalent to the stationarity conditions for v, so that we can conclude that $uv^T \in B$ if and only if $uv^T \in F$ and $uv^T \in S_d$. □

5.2.3 Limit Points of S_d

Theorem 5.4 implies that, for d sufficiently large, $B \subset S_d$. It would be interesting to have the converse affirmation, i.e., to show that for d sufficiently large, any stationary point of (R1NF-MB) corresponds to a maximal biclique of (MB). As we will see later, this property unfortunately does not hold. However, the following slightly weaker result can be proved: as d goes to infinity, the points in S_d get closer and closer to feasible solutions of (MB), i.e., to bicliques of the graph G_b. As a consequence, rounding stationary points of (R1NF-MB) for d sufficiently large will generate bicliques of G_b.

Lemma 5.5. *The set S_d is bounded, i.e., $\forall d > 0$, $\forall uv^T \in S_d$:*

$$||uv^T||_2 = ||u||_2||v||_2 \leq \sqrt{|E|}.$$

Proof. If $u = 0$ or $v = 0$, this is trivial; otherwise for any $uv^T \in S_d$, we have by (5.6)

$$||u||_2 = \Big|\Big| \max\Big(0, \frac{M_d v}{||v||_2^2}\Big)\Big|\Big|_2 \leq \frac{||\max(0, M_d)v||_2}{||v||_2^2} \leq \frac{||\max(0, M_d)||_F}{||v||_2} = \frac{\sqrt{|E|}}{||v||_2}.$$

□

Lemma 5.6. *For $uv^T \in S_d$, if $M_d(i,j) = -d$ and $(uv^T)_{ij} > 0$, then*

$$0 < u_i < \frac{||u||_1}{d+1} \quad \text{and} \quad 0 < v_j < \frac{||v||_1}{d+1}.$$

Proof. By (5.6), we have

$$0 < v_j||u||_2^2 = M_d(:,j)^T u \leq ||u||_1 - (d+1)u_i \;\Rightarrow\; 0 < u_i < \frac{||u||_1}{d+1}.$$

The corresponding result for v is obtained similarly. □

Theorem 5.7. *As d goes to infinity, stationary points of* (R1NF-MB) *get arbitrarily close to feasible solutions of* (MB)*, i.e., $\forall \epsilon > 0$, $\exists D$ s.t. $\forall d > D$:*

$$\max_{uv^T \in S_d} \min_{u_b v_b \in F} ||uv^T - u_b v_b^T||_F \;<\; \epsilon. \tag{5.7}$$

Proof. Let $uv^T \in S_d$. We can assume w.l.o.g. that $uv^T > 0$; otherwise, we consider the subproblem with the vectors $u(K)$ and $v(L)$ where K (resp. L) is the support of u (resp. v) and the matrix $M(K,L)$, see Equation (5.3). In fact, it is clear that if $(u(K), v(L))$ is close to a feasible solution of (MB) for $M_b(K,L)$, then (u,v) is for M_b. We also assume w.l.o.g. that $||v||_2 = 1$; in fact, if $uv^T \in S_d$, $\left(\lambda u \frac{1}{\lambda} v\right) \in S_d, \forall \lambda > 0$. Note that Lemma 5.5 implies $||u||_2 \leq \sqrt{|E|}$. By (5.6),

$$u = M_d v \quad \text{and} \quad v = \frac{M_d^T u}{||v||_2^2}. \tag{5.8}$$

Therefore, $(u/||u||_2, v) > 0$ is a pair of singular vectors of M_d associated with the singular value $||u||_2 > 0$. If $M_d = \mathbf{1}_{m\times n}$, the only pair of positive singular vectors of M_d is $\left(\frac{1}{\sqrt{m}}\mathbf{1}_m, \frac{1}{\sqrt{n}}\mathbf{1}_n\right)$ so that $uv^T = M_b$ coincides with a feasible solution of (MB).
Otherwise, when $M_d \neq \mathbf{1}_{m\times n}$, we define

$$A = \left\{ i \;\middle|\; M_d(i,j) = 1, \forall j \right\} \quad \text{and} \quad B = \left\{ j \;\middle|\; M_d(i,j) = 1, \forall i \right\}, \tag{5.9}$$

and their complements $\bar{A} = \{1,2,\ldots,m\}\backslash A$, $\bar{B} = \{1,2,\ldots,n\}\backslash B$; hence,

$$M_d(A,:) = \mathbf{1}_{|A|\times n} \quad \text{and} \quad M_d(:,B) = \mathbf{1}_{m\times|B|}.$$

These two sets clearly define the biclique $A \times B$ in graph G_b, or, equivalently, a (binary) feasible solution $(\bar{u}_A, \bar{v}_B)$ for problem (MB), where $\bar{u}_A$ is equal to one for indices in A and to zero otherwise (similarly for $\bar{v}_B$ and B). We are now going to show that, for d sufficiently large, uv^T is arbitrarily close to $\bar{u}_A\bar{v}_B$, which will prove our claim.

Using Lemma 5.6 and the fact that $||x||_1 \leq \sqrt{n}||x||_2, \forall x \in \mathbb{R}^n$, we get

$$0 < u(\bar{A}) < \frac{\sqrt{m|E|}}{d+1}\,\mathbf{1}_{|\bar{A}|} \quad \text{and} \quad 0 < v(\bar{B}) < \frac{\sqrt{n}}{d+1}\,\mathbf{1}_{|\bar{B}|}. \tag{5.10}$$

Therefore, since $||v||_2 = 1$ and $||u||_2 \leq \sqrt{|E|}$, we obtain

$$||u(\bar{A})v - 0||_F = ||u(\bar{A})||_2||v||_2 < \frac{1}{d+1}\left(m\sqrt{|E|}\right), \quad \text{and} \tag{5.11}$$

$$||uv^T(\bar{B}) - 0||_F = ||u||_2||v(\bar{B})||_2 < \frac{1}{d+1}\left(n\sqrt{|E|}\right). \tag{5.12}$$

It remains to show that $u(A)v(B)$ coincides with a biclique of the (complete) graph generated by $M_b(A,B) = \mathbf{1}_{|A|\times|B|}$ since $u(\bar{A})v$ and $uv^T(\bar{B})$ tend to zero as d goes to infinity.

Noting $k_v = \frac{||u||_1}{||u||_2^2}$ and using (5.8), we get $v(B) = k_v \mathbf{1}_{|B|}$. Combining this with (5.10) gives

$$1 - |\bar{B}|\frac{\sqrt{n}}{d+1} < ||v||_2^2 - ||v(\bar{B})||_2^2 = ||v(B)||_2^2 = |B|k_v^2 \leq ||v||_2^2 = 1. \tag{5.13}$$

Moreover, (5.8) implies $u(A) = \mathbf{1}_{|A|\times m} v^T = ||v||_1 \mathbf{1}_{|A|}$ so that

$$|B|k_v \leq u(A) = (||u(B)||_1 + ||u(\bar{B})||_1)\mathbf{1}_{|A|} < |B|k_v + |\bar{B}|\frac{\sqrt{n}}{d+1}. \tag{5.14}$$

Finally, multiplying (5.14) by k_v, combining it with (5.13) and noting that, since $||v||_2 = 1$, we have $k_v \leq 1$, we obtain

$$\left(1 - \frac{|\bar{B}|\sqrt{n}}{d+1}\right)\mathbf{1}_{|A|\times|B|} < u(A)v(B) < \left(1 + \frac{|\bar{B}|\sqrt{n}}{d+1}\right)\mathbf{1}_{|A|\times|B|}. \tag{5.15}$$

We can conclude that, for d sufficiently large, uv^T is arbitrarily close to a feasible solution $\bar{u}_A \bar{v}_B$ of (MB) which corresponds to the biclique (A, B). □

Example 5.1. Let

$$M_b = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \quad \text{and} \quad M_d = \begin{pmatrix} -d & 1 \\ 1 & 1 \end{pmatrix}.$$

Clearly, $\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$ belongs to the set B, i.e., it corresponds to a maximal biclique of the graph generated by M_b. By Theorem 5.4, for $d > 1$, it belongs to S_d, i.e., $(u, v) = ((1\ 1)^T, (0\ 1)^T)$ is a stationary point of (R1NF-MB).
For $d > 1$, one can also check that the singular values of M_d are different and that the second pair of singular vectors is positive. Since it is a positive stationary point of the unconstrained problem, it is also a stationary point of (R1NF-MB). As d goes to infinity, it must get closer to a biclique of (MB) (Theorem 5.7). Moreover, M_d is symmetric, so that the right and left singular vectors are equal to each other. Figure 5.1 shows the evolution[2] with respect to d of this positive singular vector (v_1, v_2), which is such that $(v_1\ v_2)^T(v_1\ v_2) \in S_d$. It converges to $(0\ 1)$, which means that the outer product of the left and right singular vectors converges to $\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$, which is a biclique, i.e., a member of F. We also note that this biclique is not maximal, which shows that the converse to Theorem 5.4 is false, even asymptotically as d goes to infinity.

Corollary 5.8. *For*

$$d \geq 2\max(m, n)\sqrt{|E|}, \tag{5.16}$$

any stationary point $uv^T \in S_d$ *of* (R1NF-MB) *can be rounded*[3] *to generate a*

[2]By Wedin's theorem (cf. matrix perturbation theory [141]), singular subspaces of M_d associated with a positive singular value depend continuously on d.

[3]Values smaller than 0.5 are set to 0, other values are set to 1.

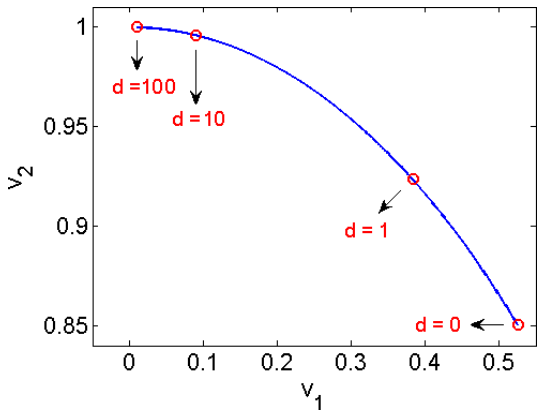

Figure 5.1: Evolution of (v_1, v_2).

biclique of the graph G_b generated by M_b.

Proof. Clearly, the condition

$$\max_{uv^T \in S_d} \min_{u_b v_b \in F} \max_{ij} (uv^T - u_b v_b)_{ij} < \frac{1}{2},$$

is sufficient to guarantee that rounding any stationary point of (R1NF-MB) will generate a biclique of G_b. Looking back at Theorem 5.7, one can check that this is satisfied (cf. Equations (5.11), (5.12) and (5.15)) for d given by (5.16) (note that w.l.o.g. $|E| \geq \max(m, n)$, i.e., that each row and each column of M_b has at least one nonzero entry, otherwise they can be removed). □

5.3 Biclique Finding Algorithm

Many real world applications rely on the discovery of maximal biclique subgraphs, e.g., web community discovery, biological data analysis and text mining [115]. Some algorithms aim at detecting all the maximal bicliques, which is computationally challenging (this is closely related to formal concept analysis, see [65]). In fact, there might be an exponential number of such bicliques and the problem is at least $\mathcal{NP}$-hard since it would solve (MB), see [2] and the references therein. For large datasets, it is in general hopeless to extract all the maximal bicliques in a reasonable computational time. Therefore, one can be interested in finding only large maximal bicliques, which is what we focus on in this section.

For example, a recent data analysis technique called binary matrix factorization (BMF) aims at expressing a binary matrix M as the product of two binary matrices [120, 154, 155]. Each rank-one factor of the decomposition corresponds to a bicluster in the bipartite graph G_b generated by M. Finding bicliques in G allows to solve BMF recursively, since bicliques of G correspond to binary rank-one underapproximations of M (see also Chapter 6).

In this section, we present a heuristic scheme designed to find large bicliques in a given graph, whose main iteration requires a number of operations proportional to the number of edges $|E|$ in the graph. It is based on the reduction from the maximum-edge biclique problem to (R1NF-MB) (Theorems 5.2, 5.4 and 5.7). We compare its performance on random graphs and text mining datasets with two other algorithms requiring $\mathcal{O}(|E|)$ operations per iteration.

5.3.1 Description

For d sufficiently large, stationary points of (R1NF-MB) are close to bicliques of (MB) (Corollary 5.8). Since (R1NF-MB) is a continuous optimization problem, any standard nonlinear optimization technique can in principle be used to compute such a stationary point. One can therefore think of applying an algorithm that finds a stationary point of (R1NF-MB) in order to localize a large biclique of the graph generated by M_b. Moreover, since the two problems have the same objective function, stationary points with larger objective functions will correspond to larger bicliques.

Of course, solving (R1NF-MB) up to global optimality, i.e., finding the best stationary point, is as hard as solving (MB). However, one can hope that the nonlinear optimization scheme used will converge to a relatively large biclique of G_b (i.e., with an objective function close to the global optimum) ; this hope will be confirmed empirically later in this section.

We choose to use the coordinate descent method presented earlier, i.e., solve alternatively the problem in the variable u for v fixed, then in the variable v for u fixed, since the optimal solutions for each of these steps can be written in closed form, cf. Equation (5.6). We also propose, instead of fixing the value of parameter d to the value recommended by Corollary 5.8, to start with a lower initial value d_0 and gradually increase it (with a multiplicative factor $\gamma > 1$) until it reaches the upper bound D equal to the recommended value. Convergence of the resulting scheme, Algorithm 10, is proved in the next Theorem.

Theorem 5.9. *The rounding of every limit point of Algorithm 10 generates a biclique of G_b, the bipartite graph generated by M_b.*

Proof. When an exact two-block coordinate descent is applied to an optimization problem with a continuously differentiable objective function and a feasible domain equal to the Cartesian product of two closed convex sets (the two blocks

correspond to $\mathbb{R}^m_+$ and $\mathbb{R}^n_+$ in this case), every limit point of the iterates is a stationary point (Theorem 2.2).

After a finite number of steps of Algorithm 10, parameter d attains the upper bound $D = 2\max(m,n)|E|$ and no longer changes, so that we can invoke this result and, using Corollary 5.8, guarantee that the resulting limit points can be rounded to generate a feasible solution of (MB), i.e., a biclique of G_b. □

Algorithm 10 Biclique Finding Algorithm based on Nonnegative Factorization

Require: Bipartite graph $G_b = (V,E)$ described by biadjacency matrix $M_b \in \{0,1\}^{m\times n}$, initial values $v_0 \in \mathbb{R}^n_{++}$ and $d_0 > 0$, parameter $\gamma > 1$.

1: **Set** parameter $D = 2\max(m,n)|E|$ and **initialize** variables $d \leftarrow d_0$, $v \leftarrow v_0$
2: **for** $k = 1, 2, \ldots$ **do**
3:

$$u \leftarrow \max\left(0, (1+d)M_b v - d||v||_1\right); \tag{5.17}$$
$$u \leftarrow u/\max(u);$$
$$v \leftarrow \max\left(0, \frac{(1+d)M_b u^T - d||u||_1}{||u||_2^2}\right); \tag{5.18}$$
$$d \leftarrow \min(\gamma d, D);$$

4: **end for**

Note that the normalization of u ($u \leftarrow u/\max(u)$) performed by Algorithm 10 only changes the scaling of the solution uv^T and allows (u,v) to converge to binary vectors. Finally, one can easily check that Algorithm 10 requires only $\mathcal{O}(|E|)$ operations per iteration, the main cost being the computation of the matrix-vector products $M_b v$ and $M_b^T u$ (the rest of an iteration requiring only $\mathcal{O}(\max(m,n))$ operations).

Parameters

It is not clear a priori how the initial value d_0 should be selected. We observed that it should not be chosen too large: otherwise, the algorithm often converges to the trivial solution: the empty biclique. In fact, in that case, the negative terms ($d||v||_1$ and $d||u||_1$) in (5.17) and (5.18) will dominate, even during the initial steps of the algorithm, and the solution will be set to zero[4].

On the other hand, the algorithm with $d = 0$ is equivalent to the power method applied to M_b, and then converges (under some mild assumptions) to

[4]In practice, we used a safety procedure which reduces the value of d whenever u (resp. v) is set to zero and reinitializes u (resp. v) to its previous value.

the best rank-one approximation of M_b [68]. Hence we observed that when d_0 is chosen too small, the iterates will in general converge to the same solution.

In order to balance positive and negative entries in M_d, we found appropriate to choose an initial value of d such that $||(M_d)_+||_F \approx ||(M_d)_-||_F$, i.e.,

$$d_0 \approx \frac{||M_b||_F}{\sqrt{|Z|}} = \sqrt{\frac{|E|}{|Z|}}, \tag{5.19}$$

(recall $|Z|$ is the number of zero entries in M_b). For our tests we chose $d_0 = 2\sqrt{\frac{|E|}{|Z|}}$, which appears to work well in practice.

Finally, the algorithm does not seem to be very sensitive to multiplicative factor γ and selecting values around 1.1 gives good results; this value will be used for the computational tests below.

5.3.2 Other Algorithms in $\mathcal{O}(|E|)$ Operations

We briefly present here two other algorithms designed to find large bicliques using $\mathcal{O}(|E|)$ operations per iteration.

Greedy Heuristic

The simplest heuristic one can imagine is to add, at each step, the vertex which is connected to the most vertices in the other side of the bipartite graph. Once a vertex is selected, the vertices which are not connected to the chosen vertex are deleted. The procedure is repeated on the remaining graph until one obtains a biclique, which is then necessarily maximal.

Motzkin-Strauss Formalism

In [56], Ding and co-authors extend the generalized Motzkin-Strauss formalism, defined for cliques, to bicliques by defining the optimization problem

$$\max_{\mathbf{x}\in F_x^\alpha, \mathbf{y}\in F_y^\beta} \mathbf{x}^T M_b \mathbf{y}$$

where $F_x^\alpha = \{x \in \mathbb{R}_+^n | \sum_{i=1}^n x_i^\alpha = 1\}$, $F_y^\beta = \{y \in \mathbb{R}_+^n | \sum_{i=1}^n y_i^\beta = 1\}$ and $1 < \alpha, \beta \ll 2$.
Multiplicative updates for this problem are then provided:

$$\mathbf{x} \leftarrow \left(\mathbf{x} \circ \frac{M_b \mathbf{y}}{\mathbf{x}^T M_b \mathbf{y}}\right)^{\frac{1}{\alpha}}, \quad \mathbf{y} \leftarrow \left(\mathbf{y} \circ \frac{M_b^T \mathbf{x}}{\mathbf{x}^T M_b \mathbf{y}}\right)^{\frac{1}{\beta}}. \tag{MS}$$

This algorithm does not necessarily converge to a biclique: if α and β are not sufficiently small, it may only converge to a dense bipartite subgraph (a

bicluster). In particular, for $\alpha = \beta = 2$, it converges to an optimal rank-one solution of the unconstrained problem, as Algorithm 10 does for $d = 0$. For our tests, we chose $\alpha = \beta = 1.05$ as recommended in [56].

In order to evaluate the quality of the solutions provided by this algorithm when it did not converge to a biclique, we used the following two post-processing procedures to convert a bicluster into a biclique:

1. Greedy (MS): extract from the generated bicluster a biclique using the greedy heuristic presented above.

2. Recursive (MS): use the algorithm recursively on the extracted bicluster, i.e., rerun it on the positive submatrix while decreasing the values of parameters α and β with $\alpha \leftarrow 1 + \frac{\alpha-1}{2}$ and $\beta \leftarrow 1 + \frac{\beta-1}{2}$.

5.3.3 Results

Synthetic Data

We first present numerical experiments with random graphs: for each density (0.1, 0.3, 0.5, 0.7 and 0.9), 100 bipartite graphs with 200 vertices (100 on each side, i.e., $m = n = 100$) were randomly generated (the probability that an edge belongs to the graph is equal to the density). We then performed, for each graph, 100 runs with the same random initializations and each algorithm was allotted 100 iterations, except for the greedy heuristic which was always run until completion and only once for each graph (since it does not require a random initialization). Actual amounts of CPU time spent by Algorithms MS and 10 were comparable, as expected from their similar iteration complexity, while the greedy heuristic was faster.

Figure 5.2 displays the performance profile for these experiments [57], where the performance function at $\rho \leq 1$ is defined as the percentage, among all graphs and all runs, of bicliques whose sizes (i.e., number of edges) is larger than ρ times the size the largest biclique found by any algortihm in the corresponding graph, i.e.,

$$\text{performance}(\rho) = \frac{\#\{\text{bicliques} \mid \text{size} \geq \rho \times \text{size of best biclique found}\}}{\#\text{runs}}.$$

On such a performance profile, the higher the curve, the better ; more specifically, the left part of the graph measures efficiency, i.e., how often a given algorithm produces the best biclique among its peers, while the right part estimates robustness, i.e., how far from the best non-optimal solutions are. These two aspects are also reported more quantitatively in Table 5.1, which displays the value of the performance function at $\rho = 1$ (Efficicency, i.e., how often a given algorithm finds a biclique with largest size) and the smallest value of ρ such that the performance function is equal to 100% (Robustness, i.e., the relative size of the worst biclique found).

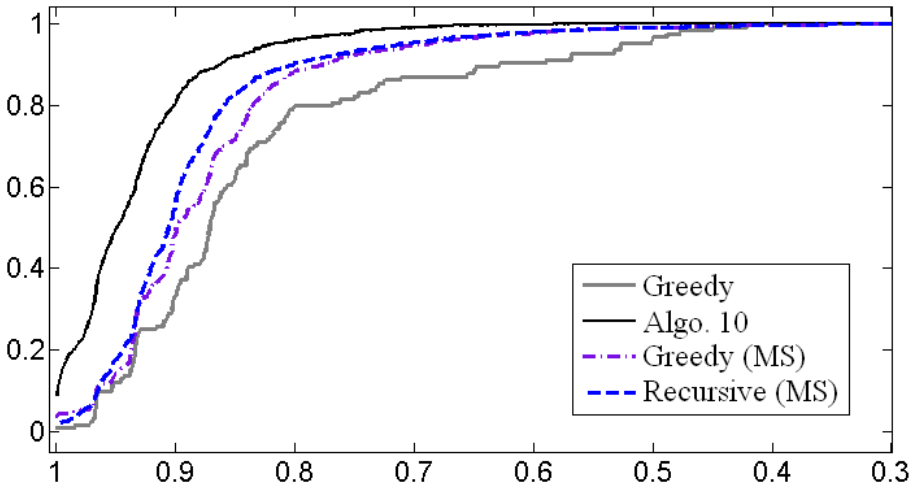

Figure 5.2: Performance profile for random graphs (densities from 0.1 to 0.9).

We observe on the performance profile that both Algorithm 10 and (MS) perform better than the greedy heuristic. The variant of (MS) using recursive post-processing performs slightly better than the one based on the use of the greedy heuristic. Nevertheless, Algorithm 10 generates in general better solutions: it is more efficient (9% of its solutions are 'optimal', twice better than the greedy (MS)) and more robust (all solutions are at most a factor 0.56 away from the best solution, better than 0.42 and 0.30 of other algorithms).

	Greedy	Algo. 10	Greedy M.-S.	Rec. M.-S.
Both (Fig. 5.2)	1% | 0.42	**9%** | **0.56**	4% | 0.30	2% | 0.30
Sparse (Fig. 5.3)	0% | 0.33	**24%** | **0.39**	14% | 0.28	14% | 0.28
Dense (Fig. 5.3)	2% | 0.76	**16%** | **0.80**	6% | 0.68	2% | 0.70

Table 5.1: Efficiency | Robustness.

It is worth noting that the algorithms behave quite differently on sparse and dense graphs. Using the same setting as before, Figure 5.3 displays performance profiles for sparse graphs (on the left, with densities 0.05, 0.1, 0.15 and 0.2) and dense graphs (on the right, with densities 0.8, 0.85, 0.9 and 0.95). For sparse graphs, both versions of (MS) seem to coincide and the greedy heuristic performs significantly worse. For dense graphs, the greedy heuristic coincides with the greedy (MS) and performs almost as well as the recursive (MS). However, in all cases, Algorithm 10 performs better. It is more efficient: it finds

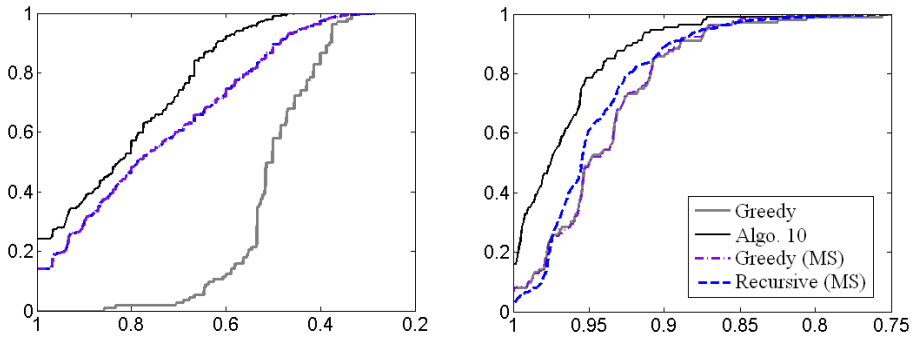

Figure 5.3: Performance profiles for random graphs: sparse (left, from 0.05 to 0.2) and dense (right, from 0.8 to 0.95).

the best solution in 24% (resp. 16%) of the runs for sparse (resp. dense) graphs while (MS) only achieves 14% (resp. 6%) and the greedy heuristic 0% (resp. 2%). It is also more robust: all solutions are at most a factor 0.39 (resp. 0.80) away from the best solution for sparse (resp. dense) graphs, bigger than the best factor 0.33 (resp. 0.76) of the other algorithms.

Text Datasets

If parameter D in Algorithm 10 is chosen smaller than the value recommended by Corollary 5.8, the algorithm is no longer guaranteed to converge to a biclique. However, the negative entries in M_d will force the corresponding entries of the solutions of (R1NF-MB) to be small (cf. Theorem 5.7). Therefore, instead of a biclique, one gets a dense submatrix of M_b, i.e., a *bicluster*. Algorithm 10 can then be used as a *biclustering algorithm* and the density of the corresponding submatrix will depend on the choice of parameter D between 0 and $2\max(m,n)|E|$. We test this approach on the six text mining datasets (with sparse matrices) described in Table 5.2.

Figure 5.4 compares Algorithms 10 and MS for varying values of their parameters: for the Motzkin-Strauss formalism, we tested $\alpha = \beta \in [1.3, 1.9]$ with step size 0.025 and, for Algorithm 10, $D \in d_0 10^{[3,9]}$ with step size 0.25 (d_0 given by Equation (5.19)). For each value, we performed 10 runs (same initializations for both algorithms and 500 iterations) and plotted all the non-dominated solutions (i.e., for which no other solution has both larger size and higher density) for each dataset. We observe that our approach consistently generates better results since its curves dominate the ones of the Motzkin-Strauss formalism, i.e., the biclusters it finds are denser for the same size or larger for the same density.

Data	m	n	$\|E\|$	sparsity
classic	7094	41681	223839	99.92
sports	8580	14870	1091723	99.14
reviews	4069	18483	758635	98.99
hitech	2301	10080	331373	98.57
ohscal	11162	11465	674365	99.47
la1	3204	31472	484024	99.52

Table 5.2: Text mining datasets [156] (sparsity is given in %: $100 * |Z|/(mn)$).

Finally, we mention that Algorithm 10 can be further enhanced in the following ways:

- ◇ It is applicable to non-binary matrices, i.e., weighted graphs. Theorem 5.2 can easily be adapted using $d \geq ||M_+||_F$ (Lemma 5.1), and one can show that the resulting algorithm will converge to the optimal rank-one approximation of a positive submatrix of M.

- ◇ It is possible to give more weight to a given side of the biclique by adding regularization terms to the cost functions. For example, on can consider the following objective function

$$\min_{u,v \geq 0} \quad ||M_d - uv^T||_F^2 + \alpha||u||_2^2 + \beta||v||_2^2$$

  which our algorithm can handle after some straightforward modifications (namely, the optimal solution for u when v is fixed can still be written in closed-form, and vice versa).

- ◇ If $M_b \in \{0,1\}^{n \times n}$ is the adjacency matrix of a (non bipartite) graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$, i.e., $M_b(i,j) = 1 \Leftrightarrow (v_i, v_j) \in E$, one can check that formulation (MB) corresponds to the maximum-edge biclique problem in any graph. This only requires that the diagonal entries of M_b are set to zero (no self loop in the graph) since a vertex cannot simultaneously belong to both sides of a biclique. Therefore, all the results of this chapter are actually valid for not necessarily bipartite graphs.

5.4 Multiplicative Updates for Nonnegative Factorization

In this section, the multiplicative updates (MU) of Lee and Seung presented in Section 4.1.1 to find approximate solutions of NMF are generalized to not necessarily nonnegative matrices: given a matrix $M \in \mathbb{R}^{m \times n}$ and a factorization

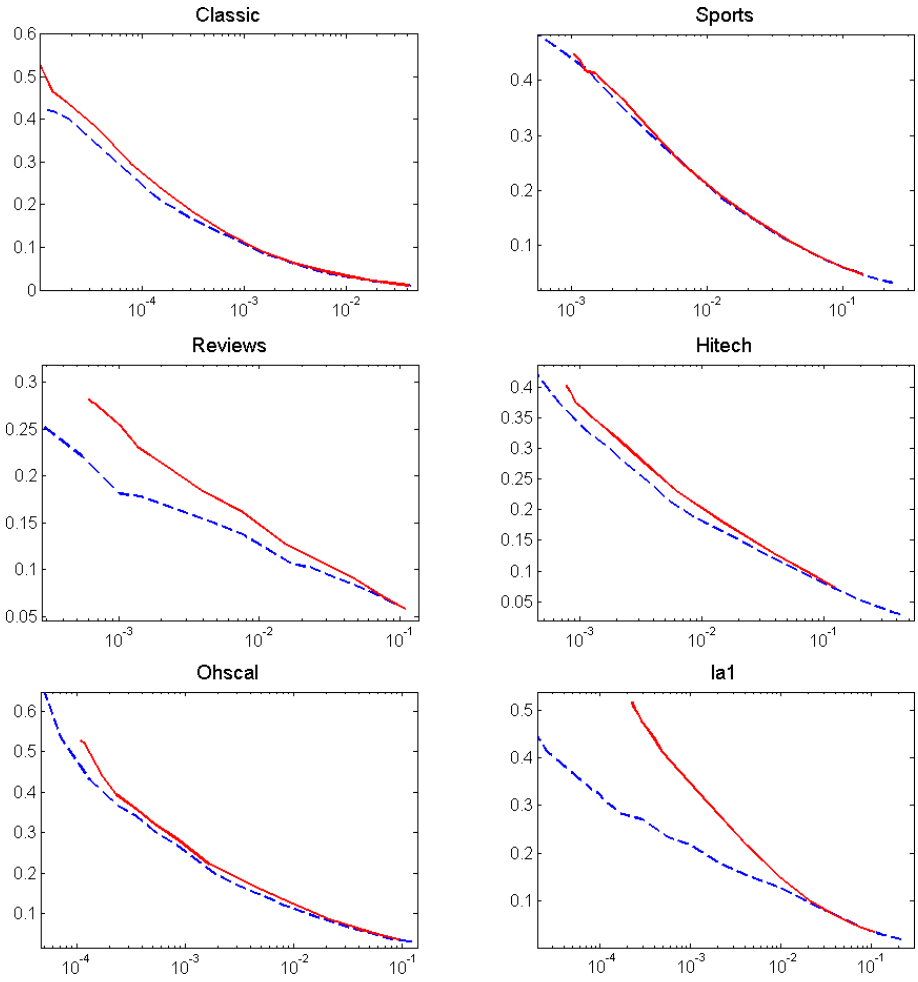

Figure 5.4: Normalized size vs. density for the Motzkin-Strauss formalism (dashed line) and Algorithm 10 based on (R1NF-MB) (solid line). The x-axis indicates the normalized sizes of the extracted clusters (i.e., number of entries in the extracted submatrix divided by the number of entries in the original matrix) while the y-axis indicates the density of these clusters (number of nonzero entries divided by the total number of entries) for the text datasets of Table 5.2.

rank r, nonnegative factorization (NF) is defined as

$$\min_{U\in\mathbb{R}^{m\times r},V\in\mathbb{R}^{r\times n}} ||M-UV||_F^2 \quad \text{such that } U\geq 0, V\geq 0. \qquad \text{(NF)}$$

Other than providing a way of computing approximate solutions of (NF), this result will also help us to understand why the updates of Lee and Seung are not very efficient in practice, see Section 5.4.1.

Of course, any real matrix M can be written as the difference of two nonnegative matrices: $M = P - N$ with $P, N \geq 0$. Using the same idea as in Section 4.1.1, we get the following multiplicative update rules :

Theorem 5.10. *For $U, V \geq 0$ and $M = P - N$ with $P, N \geq 0$, the cost function $||M - UV||_F$ is nonincreasing under the following update rules:*

$$U \leftarrow U \circ \frac{[PV^T]}{[UVV^T + NV^T]}, \qquad V \leftarrow V \circ \frac{[U^T P]}{[U^T UV + U^T N]}. \tag{5.20}$$

Proof. We only treat the proof for U since the problem is perfectly symmetric. The cost function can be split into m independent components related to each row of the error matrix, each depending on a specific row of P, N and U, which we call respectively p, n and u. Hence, we can treat each row of U separately, and we only have to show that the function

$$F(u) = \frac{1}{2}||p - n - uV||_F^2.$$

is nonincreasing under the following update

$$u_0 \leftarrow u_0 \circ \frac{[pV^T]}{[u_0 VV^T + nV^T]}, \ \forall u_0 > 0. \tag{5.21}$$

F is a quadratic function so that

$$F(u) = F(u_0) + (u - u_0)\nabla F(u_0) + \frac{1}{2}(u - u_0)\nabla^2 F(u_0)(u - u_0)^T, \quad \forall u_0,$$

with $\nabla F(u_0) = (-p + n + u_0 V)V^T$ and $\nabla^2 F(u_0) = VV^T$. Let G be a quadratic model of F around u_0:

$$G(u) = F(u_0) + (u - u_0)\nabla F(u_0) + \frac{1}{2}(u - u_0)K(u_0)(u - u_0)^T$$

with $K(u_0) = \mathrm{diag}\Big(\frac{[u_0 VV^T + nV^T]}{[u_0]}\Big)$. G has the following nice properties (see below):

(1) G is an upper approximation of F, i.e., $G(u) \geq F(u), \forall u$;

(2) The global minimum of $G(u)$ is nonnegative and given by (5.21).

Therefore, the global minimum of G, given by (5.21), provides a new iterate which guarantee the monotonicity of F. In fact,

$$F(u_0) = G(u_0) \geq \min_v G(u) = G(u^*) \geq F(u^*).$$

It remains to show that (1) and (2) hold.
(1) $G(u) \geq F(u)\ \forall u$. This is equivalent to $K(u_0) - VV^T$ positive semidefinite (PSD). Lee and Seung have proved [106] that $A = \mathrm{diag}\Big(\frac{[u_0 VV^T]}{[u_0]}\Big) - VV^T$ is PSD (see also [89]). Since $B = \mathrm{diag}\Big(\frac{[nV^T]}{[u_0]}\Big)$ is a diagonal nonnegative matrix for $u_0 > 0$ and $nV^T \geq 0$, $A + B = K(u_0) - VV^T$ is also PSD.
(2) The global minimum of G is given by (5.21):

$$\begin{aligned} u^* = \mathrm{argmin}_u G(u) &= u_0 - K^{-1}(u_0)\nabla F(u_0) \\ &= u_0 - u_0 \circ \frac{[-pV^T + (u_0 VV^T + nV^T)]}{[u_0 VV^T + nV^T]} \\ &= u_0 \circ \frac{[pV^T]}{[u_0 VV^T + nV^T]}. \end{aligned}$$

□

As with standard multiplicative updates (cf. Theorem 4.2), convergence can be guaranteed with a simple modification:

Theorem 5.11. *For every constant $\epsilon > 0$ and for $M = P - N$ with $P, N \geq 0$, $||M - UV||_F$ is nonincreasing under*

$$U \leftarrow \max\Big(\epsilon, U \circ \frac{[PV^T]}{[UVV^T + NV^T]}\Big),\ V \leftarrow \max\Big(\epsilon, V \circ \frac{[U^T P]}{[U^T UV + U^T N]}\Big), \tag{5.22}$$

for any $(U, V) \geq \epsilon$. Moreover, every limit point of a sequence of points obtained by using update (5.22) *is a stationary point of the optimization problem* (4.2).

Proof. We use exactly the same notation as in the proof of Theorem 5.20, so that

$$F(u_0) = G(u_0) \geq \min_{u \geq \epsilon} G(u) = G(u^*) \geq F(u^*), \quad u_0 \geq \epsilon$$

remains valid. By definition, $K(u_0)$ is a diagonal matrix implying that $G(u)$ *is the sum of r independent quadratic terms*, each depending on a single entry of v. Therefore,

$$\mathrm{argmin}_{u \geq \epsilon} G(u) = \max\Big(\epsilon, u_0 \circ \frac{[pV^T]}{[u_0 VV^T + nV^T]}\Big),$$

and the monotonicity is proved.
Let $(\bar{U}, \bar{V})$ be a limit point of a sequence $\{(U^k, V^k)\}$ generated by (5.22). The monotonicity implies that $\{||M - U^k V^k||_F\}$ converges to $||M - \bar{U}\bar{V}||_F$ since the cost function is bounded from below. Moreover,

$$\bar{U}_{ik} = \max\left(\epsilon, \alpha_{ik}\, \bar{U}_{ik}\right), \quad \forall i, k \tag{5.23}$$

where

$$\alpha_{ik} = \frac{P_{i:}\bar{V}_{k:}^T}{\bar{U}_{i:}\bar{V}\bar{V}_{k:}^T + N_{i:}\bar{V}_{k:}^T},$$

which is well-defined since $\bar{U}_{i:}\bar{V}\bar{V}_{k:}^T > 0$. One can easily check that the stationarity conditions of (4.2) for $\bar{U}$ are

$$\bar{U}_{ik} \geq \epsilon, \quad \alpha_{ik} \leq 1 \quad \text{and} \quad (\bar{U}_{ik} - \epsilon)\,(\alpha_{ik} - 1) = 0, \quad \forall i, k.$$

Finally, by (5.23), we have either $\bar{U}_{ik} = \epsilon$ and $\alpha_{ik} \leq 1$, or $\bar{U}_{ik} > \epsilon$ and $\alpha_{ik} = 1$,$\forall i, k$. The same can be done for $\bar{V}$ by symmetry. □

In order to implement the updates (5.20), one has to choose the matrices P and N. It is clear that $\forall P, N \geq 0$ such that $M = P - N$, there exists a matrix $C \geq 0$ such that the two components P and N can be written $P = M_+ + C$ and $N = M_- + C$. When C goes to infinity, the above updates do not change the matrices U and V, which seems to indicate that smaller values of C are preferable. Indeed, in the case $r = 1$, one can prove that $C = 0$ is an optimal choice:

Theorem 5.12. *For any $P, N \geq 0$ such that $M = P - N$, and $\forall u \in \mathbb{R}_+^n, v \in \mathbb{R}_+^m$*

$$||M - u_1 v^T||_F \leq ||M - u_2 v^T||_F \leq ||M - uv^T||_F, \tag{5.24}$$

for

$$u_1 = v \circ \frac{[M_+ v]}{[uv^T v + M_- v]} \quad \text{and} \quad u_2 = v \circ \frac{[Pv]}{[uv^T v + Nv]}.$$

Proof. The second inequality of (5.24) is a consequence of Theorem 5.20. For the first one, we treat the inequality separately for each entry of u, i.e., we prove that

$$||M_{i:} - u_{1i} v^T||_F \leq ||M_{i:} - u_{2i} v^T||_F, \quad \forall i.$$

Let define u_i^* as the optimal solution of the unconstrained problem, i.e.,

$$u_i^* = \text{argmin}_{u_i} ||M_{i:} - u_i v^T||_F = \frac{M_{i:} v}{v^T v},$$

and $a, b, d \geq 0$, $e > 0$, as

$$a = (M_+)_{i:} v,\ b = (M_-)_{i:} v,\ d = (P - M_+)_{i:} v \ \text{ and } \ e = u_i v^T v.$$

Noting that $P - M_+ = N - M_-$, we have

$$u_i^* = u_i \Big(\frac{a-b}{e}\Big),\ u_{1i} = u_i \Big(\frac{a}{e+b}\Big) \ \text{ and } \ u_{2i} = u_i \Big(\frac{a+d}{e+b+d}\Big).$$

Suppose $u_i^* \geq u_i$: then we have

$$\frac{a-b}{e} \geq 1 \ \Rightarrow \ a-b-e \geq 0 \ \Rightarrow \ \frac{a-b}{e} \geq \frac{a}{e+b}.$$

Moreover, we have1 $\leq \frac{a+d}{e+b+d}$ since u_{2i} is a better solution than u_i (Theorem 5.20). Finally,

$$1 \leq \frac{a+d}{e+b+d} \leq \frac{a}{e+b} \leq \frac{a-b}{e} \quad \Rightarrow \quad u_i \leq u_{2i} \leq u_{1i} \leq u_i^*.$$

The case $u_i^* \leq u_i$ is similar. □

Unfortunately, this result does not hold for $r > 1$. Surprisingly, this is even true for nonnegative matrices, i.e., one can improve the effect of a standard Lee and Seung multiplicative update by using a well-chosen matrix C.

Example 5.2. With the following matrices

$$M = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}, U = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}, V = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \end{pmatrix}, C = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix},$$

we have $||M - U'V||_F < ||M - U''V||_F$ where U' (resp. U'') is updated following (5.20) using $P = M + C$ and $N = C$ (resp. $P = M$ and $N = 0$).

However, in practice, it seems that the choice of a proper matrix C is nontrivial and we were not able to use this modification to accelerate the speed of convergence.

5.4.1 How good are the Multiplicative Updates?

In this section, we use Theorem 5.10 to interpret the multiplicative updates and show why the HALS algorithm performs much better in practice.

An Improved Version of the MU

The aim of the MU is to improve a current solution $(U, V) \geq 0$ by optimizing alternatively U (V fixed), and vice-versa. In order to prove the monotonicity of the MU, $||M - UV||_F$ was shown to be nonincreasing under an update of a single row of U (resp. column of V) since the objective function can be split

into m (resp. n) independent quadratic terms, each depending on the entries of a row of U (resp. column of V); cf. proof of Theorem 5.10.

However, there is no guarantee, a priori, that the algorithm is also nonincreasing with respect to an individual update of a column of U (resp. row of U). In fact, each entry of a column of U (resp. row of V) depends on the other entries of the same row (resp. column) in the cost function. The next theorem states that this property actually holds.

Corollary 5.13. *For $U, V \geq 0$, $||M - UV||_F$ is nonincreasing under*[5]

$$U_{:k} \leftarrow U_{:k} \circ \frac{[MV_{k:}^T]}{[UVV_{k:}^T]}, \qquad V_{k:} \leftarrow V_{k:} \circ \frac{[U_{:k}^T M]}{[U_{:k}^T UV]}, \quad \forall k, \tag{5.25}$$

i.e., under the single update of any column of U or any row of V using the MU.

Proof. This is a consequence of Theorem 5.10 using $P = M$ and $N = \sum_{i \neq k} U_{:i} V_{i:}$. In fact, $||M - UV||_F = ||(M - \sum_{i \neq k} U_{:i} V_{i:}) - U_{:k} V_{k:}||_F$. □

Corollary 5.13 sheds light on a very interesting fact: the multiplicative updates are also trying to optimize alternatively the columns of U and the rows of V using a specific cyclic order: first, the columns of U and then the rows of V. We can now point out two ways of improving the MU:

1. When updating a column of U (resp. a row of V), the columns (resp. rows) already updated are not taken into account: the algorithm uses their old values;

2. The multiplicative updates are not optimal for the column and row subproblems: $P \neq M_+$ and $N \neq M_-$ (cf. Theorem 5.12). Moreover, there is actually a closed-form solution for these subproblems (cf. HALS algorithm, Section 4.1.3).

Therefore, using Theorem 5.12, we have the following new improved updates

Corollary 5.14. *For $U, V \geq 0$, $||M - UV||_F$ is nonincreasing under*

$$U_{:k} \leftarrow U_{:k} \circ \frac{[(R_k)_+ V_{k:}^T]}{[U_{:k} V_{k:} V_{k:}^T + (R_k)_- V_{k:}^T]}, \quad V_{k:} \leftarrow V_{k:} \circ \frac{[U_{:k}^T (R_k)_+]}{[U_{:k}^T U_{:k} V_{k:} + U_{:k}^T (R_k)_-]}, \ \forall k, \tag{5.26}$$

with $R_k = M - \sum_{i \neq k} U_{:i} V_{i:}$. Moreover, the updates (5.26) perform better than the updates (5.25), but worse than HALS which is optimal.

[5]The proof of convergence of the MU (see Theorem 5.10) can also be used to derive this result. In fact, the quadratic upper approximation (called auxiliary function) used in the proof is a sum of r independent quadratic terms. Therefore, minimizing only one term minimizes the sum of all terms which implies this result.

Proof. This is a consequence of Theorem 5.10 and 5.12 using $P = (R_k)_+$ and $N = (R_k)_-$. In fact, $||M - UV||_F = ||R_k - U_{:k}V_{k:}||_F$. □

Figure 5.5 shows an example of the behavior of the different algorithms: the original MU (Section 4.1.1), the improved version[6] (Corollary 5.14) and the *optimal* HALS method (Section 4.1.3). The test was carried out on a commonly used data set for NMF: the cbcl face database (cf. Section 4.2.3 and Figure 1.2) for which we set $r = 40$ and we used the same *scaled* (see Equation (4.7)) random initialization and the same cyclic order (same as the MU, i.e., first the columns of U then the rows of V) for the three algorithms. We observe that the MU converges significantly less rapidly than the two other

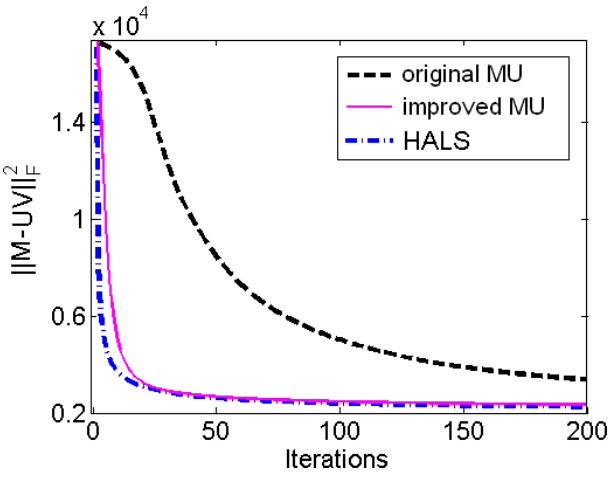

Figure 5.5: Comparison of the MU, the improved MU (5.26) and HALS on the cbcl face database.

algorithms. There do not seem to be good reasons to use either the MU or the method of Corollary 5.14 since there is a closed-form solution (HALS) for the corresponding subproblems.

Finally, the HALS algorithm has the same computational complexity (see Section 4.2) and performs provably much better than the popular multiplicative updates of Lee and Seung. Of course, because of the $\mathcal{NP}$-hardness of NMF and the existence of numerous locally optimal solutions, it is not possible to give a theoretical guarantee that HALS will converge to a better solution than the MU: although its iterations are *locally* more efficient, they could still ultimately converge to a worse local optimum.

[6]The improved MU are actually computationally more expensive since the residual matrices R_k have to be computed explicitly (even though still in $\mathcal{O}(mnr)$ operations). However, we only use it as an illustration of the poor performances of the original MU.

Conclusion

We have introduced nonnegative factorization (NF), a generalization of nonnegative matrix factorization (NMF), and proved it is $\mathcal{NP}$-hard in the rank-one case by reduction from the maximum-edge biclique problem. Since finding each rank-one factor in any NMF decomposition implicitly amounts to solving a rank-one NF problem, this suggests that NMF is a $\mathcal{NP}$-hard problem for any fixed factorization rank and that no polynomial time algorithm based on the successive optimization of the rank-one factors can be designed, giving more credence to algorithms based on alternating optimization (e.g., HALS or ANLS).

We also presented a heuristic algorithm for detecting large bicliques whose iterations require $\mathcal{O}(|E|)$ operations. It is based on results linking stationary points of a specific rank-one nonnegative factorization problem (R1NF-MB) and the maximum-edge biclique problem. We experimentally demonstrated its efficiency and robustness on random graphs and text mining datasets.

Finally, we generalized the NMF multiplicative updates to NF, and proved their convergence. This allowed us to give an explanation of the better performances of algorithm HALS over MU commonly observed in practice (cf. Chapter 4).

Chapter 6

Nonnegative Matrix Underapproximation

Finding a rank-one nonnegative matrix factorization, i.e., solving NMF with $r = 1$, is notably easier than for higher factorization ranks: while the general problem is $\mathcal{NP}$-hard, computing a globally optimal rank-one approximation can be done in polynomial time, see Section 2.2. In principle, we might try exploit this result to find factorizations of higher ranks by applying it recursively: after identification of an optimal rank-one NMF solution (u, v), one could subtract the uv^T factor from M and apply the same technique to $M - uv^T$ to recover the next rank-one factor. Unfortunately, this idea cannot work: the difference between M and its rank-one approximation may contain negative values (typically roughly half of them), so that the next SVD factor will no longer provide a nonnegative solution. Moreover, there is no hope of replacing SVD by another efficient technique for this step since we showed that it is $\mathcal{NP}$-hard to find the optimal nonnegative rank-one approximation to a matrix which is not nonnegative, cf. Chapter 5 (Theorem 5.3).

If we wish to keep the principle of a recursive algorithm finding one rank-one factor at a time, we have to add a constraint ensuring that the uv^T factor, when subtracted from M, gives a nonnegative remainder, i.e., we need to have $uv^T \leq M$. Therefore we introduce a similar *upper bound constraint* $UV \leq M$ to the general (NMF) problem and obtain a new problem we call nonnegative matrix underapproximation (NMU): given $M \in \mathbb{R}_+^{m \times n}$and $1 \leq r < \min(m, n)$, the NMU optimization problem is defined as

$$\min_{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{r \times n}} ||M - UV||_F^2 \text{ such that } U \geq 0,\ V \geq 0 \text{ and } UV \leq M. \quad \text{(NMU)}$$

Assuming we are able to solve it for $r = 1$, an underapproximation of any rank can then be built by following the recursive procedure outlined above. More

precisely, if $(U_{:1}, V_{1:})$ is a rank-one underapproximation for M, i.e., $U_{:1}V_{1:} \approx M (= R_1)$ and $U_{:1}V_{1:} \leq M$, we have that $R_2 = M - U_{:1}V_{1:}$ is nonnegative. R_2 can then be underapproximated by $U_{:2}V_{2:} \leq R_2$, leading to $R_3 = R_2 - U_{:2}V_{2:}$, and so on. After r steps, we get an underapproximation of rank r

$$\begin{aligned} M &\geq U_{:1}V_{1:} + U_{:2}V_{2:} + \cdots + U_{:r}V_{r:} \\ &= [U_{:1}\ U_{:2}\ \ldots\ U_{:r}][V_{1:};\ V_{2:};\ \ldots\ ;V_{r:}] \\ &= UV. \end{aligned}$$

Besides enabling this recursive procedure, we notice that NMU leads to a more *localized* part-based decomposition, in the sense that different basis elements tend to describe disjoint parts of the input data (i.e., involving different nonzero entries). This is a consequence of the underapproximation constraints which impose the extracted parts (the basis elements $U_{:k}$) to really be common features of the columns of M since

$$M_{:j} \gtrapprox \sum_k U_{:k}V_{kj}, \quad \forall j\ .$$

Basis elements can only be combined to approximate a column of M if each of them represents a part of this column, i.e., none of the parts selected with a positive weight can involve a nonzero entry corresponding to a zero entry in the input column $M_{:j}$. The following example demonstrates this behavior.

Example 6.1 (Swimmer Database)**.** The swimmer image dataset consists of 256 binary images of a body with 4 limbs which can be each in 4 different positions. NMF is expected to find a part-based decomposition of these images, i.e., isolate different constitutive parts of the images (the body and the limbs) in each of its basis elements.

Figure 6.1 displays a sample of such images along with the basis elements obtained with NMF and NMU. While NMF elements are rather sparse, they are mixtures of several limbs. By contrast, NMU returns a even sparser solution and is able to extract a single body part for each of its elements.

The chapter is organized as follows. Section 6.1 studies the sparsity of the solutions generated by NMU, Section 6.2 reviews some earlier related work, Section 6.3 proves that it is $\mathcal{NP}$-hard and Section 6.4 present some convex approaches to tackle the problem. In Section 6.5, we describe an algorithm to solve NMU using a technique based on Lagrangian relaxation. We test this approach on several standard image datasets, and show both qualitatively and quantitatively that it provides part-based and sparse representations that are comparable and sometimes superior to those obtained with standard sparse nonnegative matrix factorization techniques (sparse NMF), see Section 6.6.

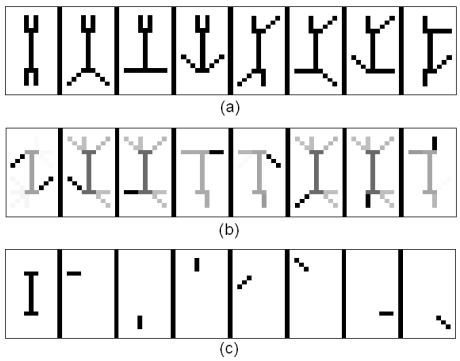

Figure 6.1: Basis elements generated for the swimmer image dataset with $r = 8$: (a) Sample images from the dataset, (b) NMF and (c) NMU; see Section 6.6 for the algorithms used to compute the factorizations.

6.1 Sparsity

The fact that NMU decompositions naturally generate sparser solutions than NMF can be explained as follows: since the zero entries of M can only be underapproximated by zeros, we have

$$M_{ij} = 0 \;\Rightarrow\; (UV)_{ij} = 0 \;\Rightarrow\; U_{ik} = 0 \text{ or } V_{kj} = 0,\ \forall k$$

which shows that when the input matrix is sparse, many components of the NMU factors will have to be equal to zero. This observation can be made more formal: defining the sparsity $s(M)$ of a m-by-n matrix M as the proportion of its zero entries, i.e.,

$$s(M) = \frac{\#\text{zeros}(M)}{mn} \;\in\; [0, 1],$$

we have the following theorem relating sparsity of M and its NMU factors.

Theorem 6.1. *For any nonnegative rank-one underapproximation $(u, v) \in \mathbb{R}^m_+ \times \mathbb{R}^n_+$ of $M \in \mathbb{R}^{m\times n}_+$ we have*

$$s(u) + s(v) \geq s(M).$$

Proof. For a rank-one matrix uv^T, the number of nonzeros is exactly equal to the product of the number of nonzeros in vectors u and v. Therefore we have that $(1 - s(uv^T)) = (1 - s(u))(1 - s(v))$ which implies $s(uv^T) = s(u) + s(v) - s(u)s(v) \leq s(u) + s(v)$. Since underapproximation uv^T satisfies $0 \leq uv^T \leq M$, it must have more zeros than M and we have

$$s(M) \leq s(uv^T) \leq s(u) + s(v),$$

proving our claim. □

Recall the recursive definition of the residuals $R_{k+1} = R_k - U_{:k}V_{k:}$ and $R_1 = M$. The following corollary relates their sparsity and the sparsity of the whole rank-r approximation with that of the NMU factors.

Corollary 6.2. *For any nonnegative underapproximation* $(U, V) \in \mathbb{R}_+^{m\times r} \times \mathbb{R}_+^{r\times n}$ *of* $M \in \mathbb{R}_+^{m\times n}$ *we have for each factor*

$$s(U_{:k}) + s(V_{k:}) \geq s(R_k) \geq s(M), \quad 1 \leq k \leq r,$$

and $s(U) + s(V) \geq s(M)$.

Proof. We have $0 \leq U_{:k}V_{k:} \leq R_k \leq M$, which implies by the previous theorem the first set of inequalities. Observing that $S(U) = \frac{1}{r}\sum_k s(U_{:k})$ and $S(V) = \frac{1}{r}\sum_k s(V_{k:})$ is sufficient to prove the second inequality. □

Sparsity of the residuals R_k is monotonically nondecreasing at each step, since $M = R_1 \geq R_2 \geq \cdots \geq 0$. Moreover, the following theorem can guarantee an increase in sparsity at each step.

Theorem 6.3. *For any locally optimal nonnegative rank-one underapproximation* $(u, v) \in \mathbb{R}_+^m \times \mathbb{R}_+^n$ *of* $M \in \mathbb{R}_+^{m\times n}$, *define sets* I *and* J *(supports of vectors* u *and* v*) by*

$$I = \{i \mid u_i > 0\},\ J = \{j \mid v_j > 0\},$$

and define matrix $R(I, J)$ *to be the submatrix of residual* $R = M - uv^T$ *whose row and column indices belong respectively to* I *and* J *(corresponding to the submatrix of* M *that is not approximated by zeros). Then there is at least one zero in each row and each column of submatrix* $R(I, J)$.

Proof. Simply observe that if $R(i, J) > 0$ (resp. $R(I, j) > 0$) for some $i \in I$ (resp. $j \in J$), u_i (resp. v_j) can be increased to obtain a strictly better solution, which contradicts the local optimality assumption. □

This ability of NMU to generate sparse part-based decomposition will be experimentally confirmed in Section 6.6, and Chapter 7.

6.2 Related Work

The problem of rank-one underapproximation has been first introduced by Levin in [108] in the case of *positive stochastic* matrices. He introduced a specific objective function different from the Frobenius norm and used a logarithmic change of variables in order to design an iterative method based on the corresponding optimality conditions.

In [68], the rank-one underapproximation problem is cast as a convex problem using again different objective functions which will be summarized in Section 6.4. Solutions are then used to initialize standard NMF algorithms in order to accelerate their convergence and, in general, find better final solutions as compared to those obtained with random initializations. Similar behavior was observed for other judicious initializations in [18, 41].

More recently, Dong et al. [58] studied the same problem with the additional constraint that the rank of the residual must be strictly smaller than the rank of the factorized matrix. Using the Wedderburn rank reduction formula, they proposed a numerical procedure which is able to compute the maximum rank splitting of a nonnegative matrix. However, the underlying optimization problem is $\mathcal{NP}$-hard [148] and their algorithm is not guaranteed to find a solution in all cases.

Biggs et al. [14] also introduced a recursive procedure to solve NMF problems: their idea is to locate and then approximate nearly rank-one submatrices of M. However, the problem of locating maximum rank-one submatrices is also shown to be $\mathcal{NP}$-hard, and their algorithm is not guaranteed to find globally optimal solutions.

6.3 Computational Complexity

We now prove that NMU is $\mathcal{NP}$-hard, even in the rank-one case, unlike NMF. In order to do this, the rank-one version of the problem is proved to be equivalent to the maximum-edge biclique problem, which is $\mathcal{NP}$-hard (cf. Section 5.1.2). The result is then generalized to NMU with arbitrary factorization rank r using a simple construction.

Recall that the maximum-edge biclique problem (MB) in a bipartite graph can be formulated as follows

$$\begin{aligned} \min_{u,v} \quad & \sum_{i,j} (M_{ij} - u_i v_j)^2 \\ & u_i v_j \leq M_{ij}, \quad \forall i, j \ , \\ & u \in \{0,1\}^m, \quad v \in \{0,1\}^n, \end{aligned} \tag{MB}$$

and is $\mathcal{NP}$-hard. For $r = 1$, (NMU) can be written as

$$\begin{aligned} \min_{u \in \mathbb{R}^m, v \in \mathbb{R}^n} \quad & \sum_{i,j} (M_{ij} - u_i v_j)^2 \\ & u_i v_j \leq M_{ij}, \ \forall i, j \ , \\ & u \geq 0, \ v \geq 0, \end{aligned} \tag{NMU-1}$$

which is very close to (MB): the difference is that vectors u and v are required to be *binary* for (MB) and *nonnegative* for (NMU-1). The next lemma proves that the two problems are actually equivalent.

Lemma 6.4. *For $M \in \{0,1\}^{m\times n}$, every optimal solution (u,v) of* (NMU-1) *is such that uv^T is binary, i.e., $uv^T \in \{0,1\}^{m\times n}$, and can then be trivially transformed into a binary optimal solution $(u',u') \in \{0,1\}^m \times \{0,1\}^n$ of* (MB).

Proof. For $M = 0$, this is trivial. Otherwise, suppose (u,v) is an optimal solution of (NMU-1). Let define (u',v') as

$$u'_i = \begin{cases} 1 & \text{if } u_i \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad v'_j = \begin{cases} 1 & \text{if } v_j \neq 0 \\ 0 & \text{otherwise} \end{cases},$$

and analyze the different possibilities: as $u_iv_j \leq M_{ij}$, we have either

- $M_{ij} = 1$ and $0 < u_iv_j \leq 1 \Rightarrow u'_iv'_j = 1$;
- $M_{ij} = 1$ and $u_iv_j = 0 \Rightarrow u'_iv'_j = 0$;
- $M_{ij} = 0 \Rightarrow u_iv_j = 0 \Rightarrow u'_iv'_j = 0$.

Therefore, $u_iv_j \leq u'_iv'_j \leq M_{ij}$ which implies

$$||M - u'v'^T||_F \leq ||M - uv^T||_F.$$

By optimality of (u,v), we must have $uv^T = u'v'^T \in \{0,1\}^{m\times n}$. Therefore, $(u',v') = (u/\max(u), v/\max(v))$ is an optimal binary solution of (NMU-1) which is then also an optimal solution of (MB) (note that we must have $\max(u) = \max(v)^{-1}$). □

Corollary 6.5. (NMU-1) *is $\mathcal{NP}$-hard.*

We now generalize Corollary 6.5 to the more general case of (NMU) with $r > 1$.

Theorem 6.6. (NMU) *is $\mathcal{NP}$-hard.*

Proof. Let $M \in \{0,1\}^{m\times n}$ be the adjacency matrix of a bipartite graph G_b. We define the matrix A as

$$A = \operatorname{diag}(M,r) = \begin{pmatrix} M & 0 & \dots & 0 \\ 0 & M & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \dots & & M \end{pmatrix},$$

which is the adjacency matrix of another bipartite graph G_b^r which is the graph G_b repeated r times. Let (U,V) be an optimal solution of (NMU). Since

$UV = \sum_{k=1}^{r} U_{:k}V_{k:}$, we have $UV \leq A \Rightarrow U_{:k}V_{k:} \leq A$. Therefore $(U_{:k}, V_{k:})$ is a feasible solution of (NMU-1) for the matrix A, i.e., for the graph G_b^r. Hence, each $(U_{:k}, V_{k:})$ corresponds to a biclique $B_G^k = (V_1^k \cup V_2^k, E^k)$ of G_b^r with

$$U_{ik} \neq 0 \Leftrightarrow s_i \in V_1^k \quad \text{and} \quad V_{kj} \neq 0 \Leftrightarrow t_j \in V_2^k.$$

By optimality of (U, V) and since there are at least r independent maximum biclique in G_b^r, each $(U_{:k}, V_{k:})$ must coincide with a maximum biclique of G_b^r which corresponds to a maximum biclique of G_b. This is due to the fact that, because G_b^r is the graph G_b repeated r times, a biclique clearly cannot span two disjoint subgraphs of G_b^r. Therefore, (NMU) is $\mathcal{NP}$-hard since any instance of (MB) can be polynomially reduced to an instance of (NMU). □

6.4 Convex Formulations

In [68], the rank-one NMU problem is analyzed in order to build a recursive algorithm. In particular, the aim is to find alternative cost functions such that the corresponding problem is convex, hence solvable in polynomial time.

A first possibility is to use the following formulation

$$\min_{u,v \geq 0} \Big\{ \max_{i,j} |M_{ij} - u_i v_j| \Big\}, \quad uv^T \leq M, \tag{6.1}$$

which can be solved with a sequence of linear systems of inequalities. This is done in two stages:

1. Reduce the problem to the case $M > 0$. Since

   $$M_{kl} = 0 \Rightarrow u_k = 0 \text{ or } v_l = 0,$$

   $M_{kl} = 0$ implies that either the k^{th} row or the l^{th} column of M is approximated by 0. Hence

   $$\min\Big(\max_j(M_{kj}), \max_i(M_{il}) \Big) \leq \max_{i,j}(M_{ij} - u_i v_j),$$

   for any feasible solution (u, v) of (6.1). Therefore, if the maximum entry of the k^{th} row (resp. l^{th} column) of M is greater that the maximum entry of the l^{th} column (resp. k^{th} row) of M, one can then set $v_l = 0$ (resp. $u_k = 0$). Indeed, if any optimal solution features one u_k set to zero (resp. v_l), v_l (resp. u_k) could also be set to zero without deteriorating the solution.
   We then successively remove columns or rows corresponding to the zero entries of M and we end up with a reduced problem with strictly positive matrix M.

2. For $M > 0$, one can check that there always exists a positive optimal solution $(u, v) > 0$ (if an entry of u or of v is equal to zero in an optimal solution, it can be replaced by a sufficiently small positive constant such that uv^T remains smaller than M). We then introduce the variables $v'_j = u_j^{-1}\ \forall j$ and w.l.o.g. we impose $u_i \geq 1\ \forall i$; if the following linear system of inequalities

$$\begin{aligned} u_i &\leq v'_j M_{ij}, \quad \forall i, j, \\ v'_j M_{ij} - u_i &\leq B v'_j, \quad \forall i, j, \\ u_i &\geq 1, \quad \forall i, \end{aligned}$$

is solvable, it means that the optimal value of (6.1) is lower than B (and a solution of this system is a feasible solution of (6.1) with optimal value lower than B), greater than B otherwise. Hence (6.1) can be solved using a sequence of such linear systems of inequalities, e.g., with a bisection algorithm for values of B starting in $[0, \max_{ij}(M_{ij})]$.

Unfortunately, this formulation is not really interesting for practical applications since it focuses only on approximating the large entries of M.

For the specific case of *positive* matrices ($M > 0$), (NMU) has also been shown to be $\mathcal{NP}$-hard for any factorization rank [68]. However, it is possible to design other cost functions for which the problem is not as hard. Consider for example a cost function based on the ratio between M and its approximation

$$\min_{u,v>0} \sum_{i,j} \left| \log\left(\frac{M_{ij}}{u_i v_j}\right) \right|.$$

By the following logarithmic change of variables:

$$c_{ij} = \log(M_{ij}),\ x_i = \log(u_i) \ \text{ and } \ y_j = \log(v_j),\ \forall i, j$$

one can compute the optimal rank-one *positive* matrix factorization with the above cost function with the following linear program :

$$\min_{x,y} \sum_{i,j} |c_{ij} - x_i - y_j|.$$

Observing that the additional underapproximation constraints can be written $x_i + y_j \leq c_{ij}\ \forall i, j$, we get a linear program

$$\begin{aligned} \max_{x,y}\ & n \sum_i x_i + m \sum_j y_j \\ & x_i + y_j \leq c_{ij}, \quad \forall i, j. \end{aligned} \tag{Flow}$$

Note that this is the dual of a flow problem, namely, the Hitchcock problem [88] so that it can be solved efficiently with dedicated algorithms [63].
Another possible formulation for the rank-one positive matrix factorization is

$$\min_{u,v>0} \sum_{i,j} \max\left(\frac{M_{ij}}{u_i v_j}, \frac{u_i v_j}{M_{ij}}\right)$$

which can be cast as a Geometric Program [61]. Recall that a geometric program has the following form

$$\begin{aligned} \min_x \ & f_0(x) \\ & f_i(x) \leq 1, \\ & x > 0, \end{aligned}$$

where f_i are *posynomials*, i.e.,

$$f_i(x) = \sum_k c_{ik} x_1^{a_{ik1}} x_2^{a_{ik2}} ... x_n^{a_{ikn}} \quad \text{with} \quad c_{ik} \geq 0, \, \forall i, k.$$

and that it can be *convexified* through a logarithmic change of variables (see, e.g., [20]). Hence, the rank-one underapproximation problem can be formulated as the following instance

$$\begin{aligned} \min_{u,v>0} \ & \sum_{i,j} \frac{M_{ij}}{u_i v_j} \\ & \frac{u_i v_j}{M_{ij}} \leq 1. \end{aligned} \tag{6.2}$$

Unfortunately, (Flow) and (6.2) can only be applied to positive matrices. Some additional work has to be done if one wants to deal with nonnegative matrices. For example, one could think of extracting, from the nonnegative matrix, a rectangular positive submatrix (cf. Example 6.2) on which those methods can be applied. Of course, we would like to extract the largest submatrix possible: this amounts exactly to finding a biclique with maximum weight[1], which is $\mathcal{NP}$-hard [48]. However, there exists heuristic algorithms for this problem and good approximate solutions can be computed (see, e.g., [3,68,90] and Section 5.3). Combining these ideas, a recursive NMU algorithm can be designed, see Algorithm 11.

[1]Since matrix M is not necessarily binary, it is the biadjacency matrix of a weighted bipartite graph G_b (i.e., to each edge (s_i, t_j) in G_b we assign a weight corresponding to the (i, j) entry of M) and we look for the biclique whith maximum weight.

Algorithm 11 Combinatorial Recursive NMU [68]

Require: $R_1 = M$, $r > 0$.
Ensure: (U, V) s.t. $UV \leq M$ with $UV \approx M$.

1: **for** $k = 1$: r **do**
2: Extract a positive submatrix $\tilde{R}_k$ from R_k;
3: Perform a rank-one underapproximation $(\tilde{U}_{:k}, \tilde{V}_{k:})$ of $\tilde{R}_k$ using (Flow) or (6.2);
4: Extend $(\tilde{U}_{:k}, \tilde{V}_{k:})$ to $(U_{:k}, V_{k:})$ by adding zeros at the entries corresponding to the ones of R_k not in $\tilde{R}_k$;
5: Compute $R_{k+1} = R_k - U_{:k}V_{k:} \geq 0$.
6: **end for**

Example 6.2. One iteration of Algorithm 11 with (Flow) :

$$
\begin{aligned}
M &= \begin{pmatrix} 1 & 3 & 0 & 7 & 0 \\ 0 & 8 & 2 & 0 & 0 \\ \boxed{9} & 6 & \boxed{4} & 0 & \boxed{1} \\ \boxed{8} & 0 & \boxed{5} & 0 & \boxed{3} \end{pmatrix} \\
&\geq \begin{pmatrix} 0 \\ 0 \\ \boxed{0.8} \\ \boxed{1} \end{pmatrix} \begin{pmatrix} \boxed{8} & 0 & \boxed{5} & 0 & \boxed{1.25} \end{pmatrix} \\
&= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \boxed{6.4} & 0 & \boxed{4} & 0 & \boxed{1} \\ \boxed{8} & 0 & \boxed{5} & 0 & \boxed{1.25} \end{pmatrix}.
\end{aligned}
$$

In [68], Algorithm 11 is used as an initialization technique for standard NMF algorithms to accelerate convergence and, in general, is observed to improve the final solution compared to random initializations. The same kind of behavior was already observed for other judicious initializations [18, 41].

An attempt is also made in [68] to compute higher rank factorizations using convex formulations. For example, using geometric programming, one could try to maximize (U, V) while imposing the underapproximation constraint :

$$
\begin{aligned}
\min_{U>0, V>0} \; & f(U, V) \\
& (UV)_{ij} \leq M_{ij}, \forall i, j
\end{aligned} \tag{6.3}
$$

where $f(U, V)$ is any posynomial nonincreasing in U and V, e.g., $\sum_{ik} U_{ik}^{-1} + \sum_{kj} V_{kj}^{-1}$.

A first disadvantage of this formulation is that it can only deal with positive matrices and is not able to set variables to zero. Moreover, we observed that if $f(U, V)$ is symmetric with respect to each rank-one factor of UV, i.e., that for any permutation σ of $[1, 2, \dots r]$

$$\tilde{U}_{:k}\tilde{V}_{k:} = U_{:\sigma(k)}V_{\sigma(k):}\ \forall k \Rightarrow f(\tilde{U}, \tilde{V}) = f(U, V), \tag{6.4}$$

then the optimal solution obtained (U^*, V^*) had all the columns of U^* equal to each other, and similarly all the rows of V^* equal to each other.

Theorem 6.7. *For any formulation* (6.3) *with a symmetric* $f(U, V)$ *(cf. Equation* (6.4)*), there exists an optimal solution* (U^*, V^*) *with the columns of* U^* *and the rows of* V^* *equal to each other.*

Proof. Consider the case $r = 2$, the proof for $r > 2$ is similar. Let

$$(U, V) = ([U_{:1}\, U_{:2}], [V_{1:}; V_{2:}])$$

be an optimal solution of (6.3) with $U_{:1} \neq U_{:2}$ or $V_{1:} \neq V_{2:}$. In the convex reformulation of the Geometric Program (6.3), the change of variables is given by

$$X_{:1} = \log(U_{:1}),\ X_{:2} = \log(U_{:2}),$$

$$Y_{1:} = \log(V_{1:}) \text{ and } Y_{2:} = \log(V_{2:})$$

where the logarithm is taken componentwise. The symmetry assumptions implies that the permutation of the columns of U and simultaneously of the corresponding rows of V generates another optimal solution. Therefore, in the convex reformulation, the mean (a convex combination) of these permuted solutions

$$\left(\left[\frac{X_{:1} + X_{:2}}{2}, \frac{X_{:1} + X_{:2}}{2}\right], \left[\frac{Y_{1:} + Y_{2:}}{2}, \frac{Y_{1:} + Y_{2:}}{2}\right]\right)$$

is also an optimal solution. Hence

$$U^*_{:1} = U^*_{:2} = \exp\left(\frac{\log U_{:1} + \log U_{:2}}{2}\right) = \sqrt{U_{:1} \circ U_{:2}},$$

$$V^*_{1:} = V^*_{2:} = \exp\left(\frac{\log V_{1:} + \log V_{2:}}{2}\right) = \sqrt{V_{1:} \circ V_{2:}}$$

with component-wise square root, must be an optimal solution of (6.3). □

Moreover, a similar reasoning shows that any other reasonable convex reformulation with a symmetric objective function, approximate or exact, will suffer from the same drawback and lead to rank-one solutions.

Nevertheless, it is possible to design asymmetric cost functions. For example, we tried to give different weights to each rank-one factor in the cost

function (e.g., $f(U,V) = \sum_{ik} U_{ik}^{-k} + \sum_{kj} V_{kj}^{-k}$). Unfortunately, we observe experimentally that the optimal solutions are such that the columns of U and the rows of V are multiple of each other although we have no theoretical proof for this fact. We also tried to introduce random weights but it does not give satisfactory results in practice.

It therefore seems hopeless to extend those convex formulations for higher ranks. We suspect that it would be quite difficult, if not impossible, to find an appropriate convex formulation for NMU (and a fortiori for NMF) for $r > 1$ mostly because of the symmetry of the problem. The same observation was also made by d'Aspremont and co-authors in [45]. To conclude, we give another example of such failure for NMF, based on semidefinite programming.

6.4.1 Tentative of SDP formulation for NMF

Let us note $N = r + m + n$,

$$Y = \begin{pmatrix} I_r \\ U \\ V^T \end{pmatrix} \in \mathbb{R}^{N \times r},$$

for $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{r \times n}$, and

$$\mathcal{Y} = YY^T = \begin{pmatrix} I_r & U^T & V \\ U & UU^T & UV \\ V^T & V^TU^T & V^TV \end{pmatrix} = \begin{pmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{12}^T & Y_{22} & Y_{23} \\ Y_{13}^T & Y_{23}^T & Y_{33} \end{pmatrix} \in \mathbb{S}_+^N,$$

where $\mathbb{S}_+^N$ is the set of N-by-N real symmetric positive semidefinite matrices. Using the correspondence with the variables (U,V) in (NMF), we would like to minimize $||M - UV||_F = ||M - Y_{23}||_F$ while requiring $U = Y_{12}^T \geq 0$ and $V = Y_{13} \geq 0$. Let us then write the following optimization problem

$$\begin{aligned} \min_{\mathcal{Y} \in \mathbb{S}^N} \quad & ||M - Y_{23}||_F \\ \text{such that} \quad & \mathcal{Y} \succcurlyeq 0,\ \mathcal{Y} \geq 0, \\ & Y_{11} = I_r, \\ & \operatorname{rank} \mathcal{Y} = r. \end{aligned} \tag{NMF$_{SDP}$}$$

Theorem 6.8. *From any optimal solution of* (NMF$_{SDP}$)*, one can construct an optimal solution of* (NMF)*, and vice-versa.*

Proof. Let (U^*, V^*) be an optimal solution of (NMF). It is clear that

$$\mathcal{Y} = \begin{pmatrix} I_r \\ U^* \\ V^{*T} \end{pmatrix} \begin{pmatrix} I_r \\ U^* \\ V^{*T} \end{pmatrix}^T$$

is a feasible solution of (NMF$_{SDP}$), implying that for any optimal solution $\mathcal{Y}^*$ of (NMF$_{SDP}$) we have $||M - Y_{23}^*||_F \leq ||M - U^*V^*||_F$.

Let $\mathcal{Y}^*$ be an optimal solution of (NMF$_{SDP}$). Since $\mathcal{Y}^* \in \mathbb{S}^N$ and $\operatorname{rank} \mathcal{Y} = r$, there exists a matrix $Y \in \mathbb{R}^{N\times r}$ with $\mathcal{Y}^* = YY^T$ (Y could be computed using the SVD, but it is not necessarily unique). Noting

$$Y = \begin{pmatrix} A \\ B \\ C \end{pmatrix} \in \mathbb{R}^{N\times r}, \text{ with } A \in \mathbb{R}^{r\times r}, B \in \mathbb{R}^{m\times r} \text{ and } C \in \mathbb{R}^{n\times r},$$

we have

$$\mathcal{Y}^* = YY^T = \begin{pmatrix} AA^T & AB^T & AC^T \\ BA^T & BB^T & BC^T \\ CA^T & CB^T & CC^T \end{pmatrix} \in \mathbb{S}_+^N,$$

with

- $AA^T = I_r$, i.e, A orthogonal.
- $BA^T \geq 0$ and $AC^T \geq 0$.
- $||M - BC^T||_F = ||M - B(A^TA)C^T||_F = ||M - (BA^T)(AC^T)||_F$ is minimized.

Hence we have (BA^T, AC^T) is a feasible solution for (NMF) and

$$||M - (BA^T)(AC^T)||_F \geq ||M - U^*V^*||_F,$$

for any optimal solution (U^*, V^*) of (NMF).

Combining the above two inequalities, $||M - U^*V^*||_F = ||M - Y_{23}^*||_F = ||M - (BA^T)(AC^T)||_F$ for any optimal solution $\mathcal{Y}^*$ of (NMF$_{SDP}$) and any optimal solution (U^*, V^*) of (NMF), the proof is complete. □

Notice that the formulation of (NMF$_{SDP}$) allows to remove degrees of freedom of (NMF). In fact, for any feasible solution (U, V) of (NMF), any equivalent solution (UA, A^TV) with A orthogonal, $UA \geq 0$ and $A^TV \geq 0$, will correspond to the same solution $\mathcal{Y}$ in (NMF$_{SDP}$).

One could now relax the above $\mathcal{NP}$-hard problem, e.g., by removing the rank constraint as it is usually done. However, for the reasons explained in Theorem 6.7, any optimal solution for the relaxation will correspond to rank-one matrices $U = Y_{12}^T$ and $V = Y_{13}$ because the formulation is symmetric.

Another idea is to use the nuclear norm minimization heuristic [3,129]. Since minimizing the trace of Y is equivalent to minimizing its nuclear norm, we get

the following SDP heuristic approach to approximate solutions of (NMF$_{SDP}$):

$$\begin{aligned} \min_{\mathcal{Y}\in\mathbb{S}^N} \quad & \text{trace}(\mathcal{Y}) \\ \text{such that} \quad & \mathcal{Y} \succcurlyeq 0,\ \mathcal{Y} \geq 0, \\ & Y_{11} = I_r, \\ & ||M - Y_{23}||_F \leq \epsilon, \end{aligned} \qquad \text{(Heuristic for NMF}_{SDP}\text{)}$$

where ϵ is a desired approximation accuracy (e.g., for 5% relative accuracy, one could use $\epsilon = 0.05||M||_F$). Notice that, in the rank-one case, this formulation is closely related to the SDP formulation of the biclique problem proposed by Ames and Vavasis [3].

Unfortunately, again, any solution of this Heuristic for NMF$_{SDP}$ generates rank-one matrices $U = Y_{12}^T$ and $V = Y_{13}$. We were not able to break this symmetry.

6.5 An algorithm for NMU based on Lagrangian relaxation

Since (NMU), like (NMF), is a $\mathcal{NP}$-hard problem, we can not expect to solve it up to guaranteed global optimality in a reasonable (e.g., polynomial) computational time (unless $P = NP$). In this section, we propose a nonlinear optimization scheme based on Lagrangian relaxation in order to compute approximate solutions of (NMU).

Drop the $m \times n$ underapproximation constraints $UV \leq M$ of (NMU) and add them into the objective function with the corresponding Lagrange multipliers (dual variables, forming a matrix) $\Lambda \in \mathbb{R}_+^{m\times n}$, to obtain the Lagrangian function $L(U, V, \Lambda)$

$$L(U, V, \Lambda) = \frac{1}{2}||M - UV||_F^2 + \sum_{i=1}^{m}\sum_{j=1}^{n} \Lambda_{ij}(UV - M)_{ij},$$

where a factor of $\frac{1}{2}$ was introduced to make the presentation nicer. The Lagrangian relaxation subproblem (LR$_\Lambda$) consists in minimizing L for a fixed value of the Λ multipliers, leading to the corresponding Lagrangian dual function $f(\Lambda)$

$$f(\Lambda) = \min_{U,V\geq 0} L(U, V, \Lambda) \qquad \text{(LR}_\Lambda\text{)}$$

where $f(\Lambda)$ is well-defined because the minimum of $L(U, V, \Lambda)$ is always attained, due to the fact that f is bounded below and the search space can be

restricted to a compact set. Indeed, considering each rank-one factor individually $(U_{:k}, V_{k:})$ and imposing w.l.o.g. $||U_{:k}||_F^2 = ||V_{k:}||_F^2 = ||U_{:k}||_F||V_{k:}||_F = ||U_{:k}V_{k:}||_F$, we have

$$\begin{aligned} ||U_{:k}||_F^2 = ||V_{k:}||_F^2 &\leq ||UV||_F \\ &\leq ||M-\Lambda||_F + ||M-\Lambda-UV||_F, \\ &\leq 2||M-\Lambda||_F \ \forall k, \end{aligned}$$

where we have used the trivial solution $(U, V) = (0, 0)$ to bound $||M-\Lambda-UV||_F$ (cf. derivations of Section 6.5.1).

Standard application of Lagrangian duality tells us that

$$(\text{NMU}) \equiv \min_{U,V\geq 0} \ \sup_{\Lambda\geq 0} \ L(U,V,\Lambda) \quad \geq \quad \sup_{\Lambda\geq 0} \ \min_{U,V\geq 0} \ L(U,V,\Lambda) \ = \ \sup_{\Lambda\geq 0} f(\Lambda),$$

where the problem on the left of the inequality is equivalent to our original NMU formulation and the problem on the right is its Lagrangian dual, whose solution will provide a (hopefully tight) lower bound on the optimal (NMU). This new problem is a nondifferentiable optimization problem with the nice property that its objective $f(\Lambda) = \min_{U,V\geq 0} L(U, V, \Lambda)$ is concave and its maximization (over a convex set) is then a convex problem (see [4] and the references therein).

We describe in the next section a general solution technique, which consists in repeatedly applying the following two steps:

1. Given multipliers Λ, compute (U, V) to (approximately) minimize $L(U, V, \Lambda)$, i.e., solve (LR_Λ); this is discussed in Section 6.5.1;

2. Given solution (U, V), update multipliers Λ; this is described in Section 6.5.2.

6.5.1 Solving the Lagrangian Relaxation

The following derivations

$$\begin{aligned} L(U,V,\Lambda) &= \sum_{i,j} \frac{1}{2}(M-UV)_{ij}^2 + \sum_{i,j} \Lambda_{ij}(UV-M)_{ij} \\ &= \frac{1}{2}\sum_{i,j} M_{ij}^2 - \sum_{i,j} M_{ij}(UV)_{ij} + \frac{1}{2}\sum_{i,j}(UV)_{ij}^2 \\ &\quad + \sum_{i,j} \Lambda_{ij}(UV)_{ij} - \sum_{i,j} \Lambda_{ij}M_{ij} \\ &= \frac{1}{2}||(M-\Lambda)-UV||_F^2 - \frac{1}{2}||\Lambda||_F^2, \end{aligned}$$

show that minimizing $L(U, V, \Lambda)$ for a fixed Λ is equivalent to minimizing $||(M-\Lambda)-UV||_F^2$. Matrix $R = M-\Lambda$ is not necessarily nonnegative, therefore finding

$U \geq 0$ and $V \geq 0$ such that $R \approx UV$ is a more general problem than NMF. It is actually the problem of nonnegative factorization (NF) studied in Chapter 5 where it was formulated as

$$\min_{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{r \times n}} ||R - UV||_F^2 \text{ such that } U \geq 0 \text{ and } V \geq 0, \tag{NF}$$

with $R \in \mathbb{R}^{m \times n}$ and $1 \leq r < \min(m, n)$, was shown to be $\mathcal{NP}$-hard even for $r = 1$, see Theorem 5.3.

Standard algorithms for NMF can easily be adapted to handle an input matrix that is not nonnegative, i.e., solve a NF problem (see, e.g., the generalized multiplicative updates in Chapter 5). We decide to use hierarchical alternating least squares (HALS) since it is one of the most efficient NMF algorithm, see Chapter 4. HALS alternatively updates each column of U and each row of V with the following optimal closed-form solutions:

$$\begin{aligned} U_{:k}^* &= \text{argmin}_{U_{:k} \geq 0} \ ||(M - \Lambda) - UV||_F^2 \\ &= \max\Big(0, \frac{A_{:k} - \sum_{l=1, l \neq k}^r U_{:l} B_{lk}}{B_{kk}}\Big), \end{aligned} \tag{6.5}$$

with $A = (M - \Lambda)V^T$ and $B = VV^T$, and

$$\begin{aligned} V_{k:}^* &= \text{argmin}_{V_{k:} \geq 0} \ ||(M - \Lambda) - UV||_F^2 \\ &= \max\Big(0, \frac{C_{k:} - \sum_{l=1, l \neq k}^r D_{kl} V_{l:}}{D_{kk}}\Big), \end{aligned} \tag{6.6}$$

with $C = U^T(M - \Lambda)$ and $D = U^T U$. The main computational cost of one HALS iteration is the evaluation of A and C: they each require $2mnr$ (floating point) operations. One can check that the resulting total number of operations is $4mnr + \mathcal{O}((m + n)r^2)$.

6.5.2 Update of the multipliers Λ

The second step of our algorithm consists in updating Λ in order to find better (i.e., higher) solutions to the Lagrangian dual problem. Using the knowledge that any optimal solution (Λ^*, V^*, W^*) of the Lagrangian dual must satisfy the following complementarity slackness conditions

$$\Lambda_{ij}^*(M - U^*V^*)_{ij} = 0 \ \forall i, j,$$

as well as feasibility conditions

$$\Lambda_{ij}^* \geq 0 \text{ and } (M - U^*V^*)_{ij} \geq 0,$$

we see that the update rule for the multipliers Λ should satisfy the following:

- ◇ if $(M - UV)_{ij} > 0$, Λ_{ij} should be decreased and eventually reach zero if $(M - U^*V^*)_{ij} > 0$,
- ◇ if $(M - UV)_{ij} < 0$, Λ_{ij} should be increased to give more importance to $(M - UV)_{ij}$ in the cost function, hopefully in order to get a feasible solution such that $(M - U^*V^*)_{ij} \geq 0$.

In the sequel, we use the following rule to update Λ, which satisfies the above requirements:

$$\Lambda \leftarrow \max(0, \Lambda - \mu_k(M - UV)), \quad \mu_k \to 0,$$

where μ_k is a predefined sequence of step lengths decreasing to zero; Λ can be initialized to zero. This update is inspired from the concept of subgradient methods [138]; in fact, one can easily check that the quantity $(UV - M)$ is a subgradient of

$$f(\Lambda) = \min_{U,V \geq 0} L(U, V, \Lambda)$$

with respect to Λ, i.e., if $(\bar{U}, \bar{V}) = \operatorname{argmin}_{U,V \geq 0} L(U, V, \bar{\Lambda})$, we have

$$f(\Lambda) \leq f(\bar{\Lambda}) + \left\langle \bar{U}\bar{V} - M, \Lambda - \bar{\Lambda} \right\rangle, \quad \forall \Lambda.$$

Two questions now arise

- ◇ Since an iterative algorithm is used to solve (approximately) the Lagrangian relaxation problem (cf. section 6.5.1), after how many of these HALS iterations do we stop and proceed to update the multipliers Λ?
- ◇ How do we choose the sequence of step lengths μ_k?

Subgradient methods usually assume that the Lagrangian relaxation problem (LR_Λ) can be solved exactly and can guarantee their convergence to an optimal solution provided an appropriate sequence of step sizes is selected (see, e.g., [4]), for example $\{\mu_k\}$ satisfying the conditions

$$0 \leq \mu_k \to 0 \quad \text{such that} \quad \sum_{k=0}^{\infty} \mu_k^2 < +\infty \ \text{ while } \ \sum_{k=0}^{\infty} \mu_k = +\infty.$$

In the sequel, we choose to use $\mu_k = \frac{1}{k}$, which is such a suitable sequence. However, in our case, we cannot expect to solve (LR_Λ) in a reasonable amount of time since the problem is $\mathcal{NP}$-hard. It would even probably be too expensive to wait for the stabilization of (U, V) (e.g., getting close to a stationary but not necessarily optimal point). We therefore suggest to update (U, V) only a constant number of times T between each update of Λ, which leads to Algorithm L-NMU. Note that because we do not solve (LR_Λ) exactly, Algorithm L-NMU is not guaranteed to converge to an optimal solution of the Lagrangian dual but, as we will see, it produces satisfactory solutions in practice.

Algorithm 12 L-NMU Lagrangian NMU

Require: $M \in \mathbb{R}_+^{m\times n}$, $r > 0$, $U \in \mathbb{R}_+^{m\times r}$, $V \in \mathbb{R}_+^{r\times n}$, maxiter, T.
Ensure: (U, V) s.t. $UV \lesssim M$.

1: $\Lambda = 0$;
2: **for** $k = 1 :$ maxiter **do**
3: Update (U, V) using T iterations of HALS (6.5)-(6.6);
4: Update $\Lambda \leftarrow \max(0, \Lambda - \frac{1}{k}(M - UV))$;
5: **end for**

The additional computational cost of one iteration of algorithm L-NMU when compared with one iteration of HALS for NMF consists in the computation of $M - \Lambda$ (needed in step 3) and the update of Λ (at step 4), which require $2mnr + \mathcal{O}(mn)$ operations (and, in the special case $r = 1$, $5mn$ operations).

Remark 6.1. Because convergence is not theoretically guaranteed, Algorithm L-NMU may end up with solutions that do not completely satisfy the underapproximation constraint. Although our numerical experiments show that this has no detrimental influence on the quality of the obtained sparse part-based representations (see Section 6.6), we give here a simple technique to transform such a solution into a feasible underapproximation. Indeed, it is enough to consider the following QP problem (convex quadratic objective function, linear inequality constraints) which only involves the U factor

$$U^* = \text{argmin}_{U\geq 0, UV\leq M} ||M - UV||_F^2. \tag{6.7}$$

Because of its convexity, this problem can be solved up to global optimality in a very efficient manner, and replacing the original U factor by the optimal solution U^* leads to a feasible solution (U^*, V) to (NMU).

Remark 6.2. Because update rule (6.7) is exact and computable in practice, it would be natural to consider a simpler algorithm based on its alternative application to the U and V factors, without using the Lagrangian relaxation technique, hoping to converge to a solution of (NMU). Unfortunately, we observed that this is quite inefficient in practice. In fact,

- ◇ it is relatively computationally expensive to solve these linearly constrained quadratic programs (with $mn + mr$ and $mn + nr$ inequalities), at least compared to the HALS closed-form update rules (6.5)-(6.6);
- ◇ since the underapproximation constraint is imposed at each step, this algorithm has much less freedom to converge to good solutions: iterates rapidly get stuck on the boundary of the feasible domain, typically with (too) many zeros and a lower rank. For example, assuming M has one zero

in each column, we have that for any positive matrix U the corresponding optimal V is equal to 0:

$$\forall j, \exists i \text{ s.t. } M_{ij} = 0 \Rightarrow \forall j, \exists i \text{ s.t. } \sum_k U_{ik} V_{kj} = 0 \Rightarrow V_{kj} = 0, \forall k, j.$$

Therefore, such an algorithm can only work if we decide a priori which values in U and V should be equal to zero, i.e., if we find a good sparsity pattern for the solution, which is precisely where the difficulty of the problem lies. Note that the same behavior is observed if a HALS-type algorithm is used instead of (6.7) (i.e., updating columns of U and rows of V alternatively): after the update of one column of U, the residual will have one zero in each row (cf. Theorem 6.3) which will prevent the other columns of U to be nonzero (except if the sparsity pattern is chosen a priori).

Remark 6.3. The L-NMU algorithm described above is not particularly well-suited to deal with very sparse input matrices. In fact, one has to store a potentially dense $m \times n$ matrix with the Lagrangian variables Λ. Nevertheless, we have obtained [11] encouraging results when applying NMU to sparse anomaly detection problems in text mining. Moreover, it is possible to take advantage of the input sparsity pattern and design a computationally cheaper method. First note that the Lagrangian variables associated with a zero of M will be nondecreasing in the course of the algorithm, since

$$0 \leq (U^{(k)} V^{(k)})_{ij} \text{ and } M_{ij} = 0 \quad \Rightarrow \quad \Lambda_{ij}^{(k)} \leq \Lambda_{ij}^{(k+1)},$$

where superscript $^{(k)}$ denotes the solution at step k. Therefore one can significantly reduce the computational cost by defining

$$\Lambda_{ij}^{(k)} = -g(k) \quad \text{for all } i, j \text{ s.t. } M_{ij} = 0,$$

where $g(k)$ is an arbitrary positive nondecreasing function, e.g., $g(k) = \gamma^k$ with $\gamma > 1$. This is actually what was done in Algorithm 10 in Section 5.3 where $g(k)$ corresponds to the parameter d and at each step we used $d \leftarrow \min(\gamma d, D)$. This leads to an algorithm in $\mathcal{O}(|E|)$ operations, where $|E|$ is the number of nonzero entries in M.

6.6 NMU vs sparse NMF

We have argued in Section 6.1 that NMU is potentially able to extract a better part-based representation of the data and that its factors should be sparser than those of the standard NMF, at the detriment of the approximation error. In this

section, we support these claims by reporting results of computational experiments involving two variants of Algorithm L-NMU on several image datasets.

A direct comparison between NMU and NMF is not very informative in itself: while the former will provide a sparser part-based representation, the latter will feature a lower approximation error. This does not really tell us whether the improvements in the part-based representation and sparsity are worth the increase in approximation error. For that reason, we chose to compare NMU with two other sparse nonnegative matrix factorizations techniques, described below, in order to better assess whether the increase in sparsity achieved by NMU is worth the loss in reconstruction accuracy.

6.6.1 Sparse NMF

We selected and tested the following two sparse nonnegative matrix factorization techniques that are frequently used in the literature.

1. Hoyer describes in [91] an algorithm relying on additional explicit sparsity constraints on the factors, enforced at each iteration by means of a projection. The approximation error is reduced via a combination of projected gradient and multiplicative updates. For our experiments, we use the MATLAB® code provided by the author[2].

   It should be pointed out that Hoyer is using a different definition of sparsity: for any nonzero n dimensional vector x, his measure of sparsity $sh(x)$ is defined as

   $$sh(x) = \frac{\sqrt{n} - ||x||_1/||x||_2}{\sqrt{n} - 1} \quad \in \quad [0, 1]. \tag{6.8}$$

   Hence, a vector with a single nonzero entry is perfectly sparse

   $$\text{sh}([0 \ldots 0\ k\ 0 \ldots 0]) = 1, \quad \forall k \neq 0,$$

   while a vector with all entries equal to each other is completely dense

   $$\text{sh}([k \ldots k]) = 0, \quad \forall k \neq 0.$$

   In our experiments, we report sparsity using both the standard $s(\cdot)$ indicator and Hoyer's $sh(\cdot)$ measure.

2. Instead of enforcing sparsity at every iteration, a sparsity-inducing penalty term can be introduced in the objective function [111]. In particular, it is well-known that adding l_1-norm penalty terms induce sparser solutions (see, e.g., [33, 97, 98]), and we therefore solve the following problem:

   $$\min_{U,V \geq 0} ||M - UV||_F^2 + \mu_U ||U||_1 + \mu_V ||V||_1, \tag{sNMF}$$

[2]This code was downloaded from `http://www.cs.helsinki.fi/u/phoyer/software.html`.

where $||A||_1 = \sum_{ik} |A_{ik}|$ and μ_U and μ_V are two positive parameters controlling the sparsity of U and V. In order to solve (sNMF), we use the HALS algorithm which can easily be adapted to handle the additional l_1-norm penalty terms (see, e.g., [33, 89]). This algorithm will be referred to as sNMF.

Technical details for the first technique are more complicated, but it allows the sparsity of the factors to be chosen a priori. The second technique is conceptually simpler but requires the determination of appropriate penalizing parameters by other means.

6.6.2 Tested algorithms

Algorithm L-NMU proposed in Section 6.5 can be used to compute underapproximations for any given factorization rank r. This opens the possibility of building a rank-r underapproximation in several different ways: one simple option consists in applying algorithm L-NMU directly to the rank-r problem – we call this method *global NMU* (G-NMU). Another option consists in applying the recursive technique outlined in the introduction, used to motivate the introduction of underapproximations. More specifically, this means running algorithm L-NMU successively r times to compute r rank-one approximations, subtracting each approximation from the input matrix before computing the next one – we call this method *recursive NMU* (R-NMU). Note that many other variants are possible (e.g., computing two rank-$\frac{r}{2}$ approximations, computing $\frac{r}{2}$ successive rank-two approximations, etc.) but we only tested the two above-mentioned variants, which represent two extreme cases (no recursion and maximum recursion). In practice, it is not clear how to determine in advance which variants is the most appropriate; it will highly depend on the data and the application of interest.

In both cases, our implementation of algorithm L-NMU computes two HALS steps between each update of the multipliers Λ (i.e., we fixed $T = 2$). Most of the computational work done in one iteration of L-NMU consists in computing $M - \Lambda$, performing the two HALS steps and updating Λ; more specifically, one can estimate the computational cost of one iteration of G-NMU to $10mnr + \mathcal{O}((m+n)r^2)$ operations, while an R-NMU iteration takes $13mn + \mathcal{O}((m+n)r)$ operations (repeated r times in the recursive procedure).

For each dataset, we test five nonnegative factorization algorithms: NMF based on HALS updates (NMF), global NMU (G-NMU), recursive NMU (R-NMU), sparse NMF with l_1-penalty terms (sNMF) and the algorithm of Hoyer. We also report the results of Principal Component Analysis (PCA) to serve as a reference (recall that the approximation error of the resulting unconstrained low-rank approximation, computed here with a singular value decomposition, is globally minimal for the given rank, but that its factors are neither nonnegative, nor sparse).

6.6.3 Iteration limits and CPU time

Each of the five iterative algorithms described above requires a limit on the number of its iterations; these limits were chosen in order to roughly allocate the same CPU time to each algorithm. More specifically, the standard NMF was given a 600-iterations limit, which corresponds to the computation of 600 HALS updates. The sparse sNMF, based on a slightly modified HALS update, was also allowed 600 iterations. Because a HALS update involves $4mnr + \mathcal{O}((m+n)r^2)$ operations, we can deduce the following iteration budgets for G-NMU and R-NMU from the leading terms in their corresponding operation counts: G-NMU is allowed $600 \times \frac{4}{10} = 240$ L-NMU iterations while R-NMU can take $600 \times \frac{4}{13} \approx 180$ iterations.

An exception to the equal CPU time rule was made for the algorithm of Hoyer. Results obtained after an amount of CPU time similar to that of the other algorithms were too poor to be compared in a meaningful way. Indeed, because this method is based on a projected gradient method and multiplicative updates (both $\mathcal{O}(mnr)$ operations per iteration), which are known to converge at a typically much slower rate (see Chapter 4), a relatively high limit of 1000 iterations had to be fixed, although the resulting CPU time is then much larger than for the other methods (for example, on the CBCL dataset, 600 iterations of HALS took $\sim 80s$. while 1000 iterations of the algorithm of Hoyer needed $\sim 260s$.).

6.6.4 Testing methodology

Recall we decided to test algorithms sNMF and Hoyer to assess the quality of the sparsity-accuracy compromise proposed by our NMU approaches. To achieve this, we decided to associate with each NMU variant a solution of sNMF/Hoyer featuring the same level sparsity, and compare the resulting approximation errors. We therefore report results for eight algorithms on each dataset: PCA, NMF, G-NMU, sNMF with the same sparsity as G-NMU, which we denote by sNMF{G-NMU}, Hoyer{G-NMU}, R-NMU, sNMF{R-NMU} and Hoyer{R-NMU}.

In order to enforce a sparsity similar to the NMU solution in Hoyer's code, we compute the *sh* measure of the NMU factors and input it as a parameter of the method (see description in subsection 6.6.1); note however that we could only enforce this for the sparsest of the two NMU factors[3]. In the case of sNMF, sparsity cannot be directly controlled, and penalty parameters are found using the following adaptive procedure, which proved to work well in practice: μ_U and μ_V are initialized to 0.1 and, after each iteration, μ_U (resp. μ_V) is increased by 5 percent if $S(U)$ (resp. $S(V)$) is below the target sparsity, and is decreased

[3]Ideally, we would have imposed sparsity for both factors, but the implementation we used seemed to return poor results in that situation.

by 5 percent otherwise.

All algorithms were run 10 times with the same initial random matrices and only the best solution with respect to the Frobenius norm of the error is reported. When testing with gray-level images, the input matrices M where normalized to have their entries varying between 0 and 1, with 0 representing white and 1 representing black (when trying to decompose M as a sum of parts, this make more sense than the opposite convention, since the dark regions are the constitutive parts of the objects in the image datasets we analyze). Finally, before computing reported sparsity measures of the factors, any sufficiently small[4] entry is rounded to zero (indeed, because algorithms are stopped by the iteration limit before convergence, true zeros are typically not all reached). All tests were run within the MATLAB® 7.1 (R14) version, on a 3GHz Intel® Core™2 Dual CPU PC.

6.6.5 CBCL Face Database

The CBCL face image dataset was used for the illustrative example of Figure 1.2 and is made of 2429 gray-level images of faces represented with 19×19 pixels. We look for an approximation of rank $r = 49$.

Figure 6.2 displays the basis elements for NMF, G-NMU, R-NMU and sNMF{G-NMU} (which was the best solution obtained in term of sparsity vs. error among all four sNMF and Hoyer variants). Both G-NMU, R-NMU and sNMF achieve a better part-based representation than NMF, generating sparser solutions. An interesting feature of R-NMU is that it extracts parts successively in order of "importance": the first basis element is a 'mean' face (which is dense) while the next ones describe different complementary parts (which become sparser as the recursion moves on, cf. Corollary 6.2 and Theorem 6.3).

A more quantitative assessment is provided in Table 6.1, reporting for the eight algorithms tested the relative error (in percent) of their solutions

$$\text{relative error} = \frac{||M - UV||_F}{||M||_F}$$

in the second column ("Plain") and the corresponding sparsity measures (in percent) of factors U and V in the last four columns.

As expected, PCA returns the smallest error, albeit with very dense factors. NMF already features much sparser factors (slightly half of the entries in U are equal to zero), at the cost of a relatively modest increase in the approximation error ($7.43 \rightarrow 8.12$). G-NMU provides an even sparser solution (three quarters of zero entries), increasing again the approximation error ($8.12 \rightarrow 12.45$). The

[4] We declare an entry of a factor to be sufficiently small if it is less than 0.1% of the largest entry in its column.

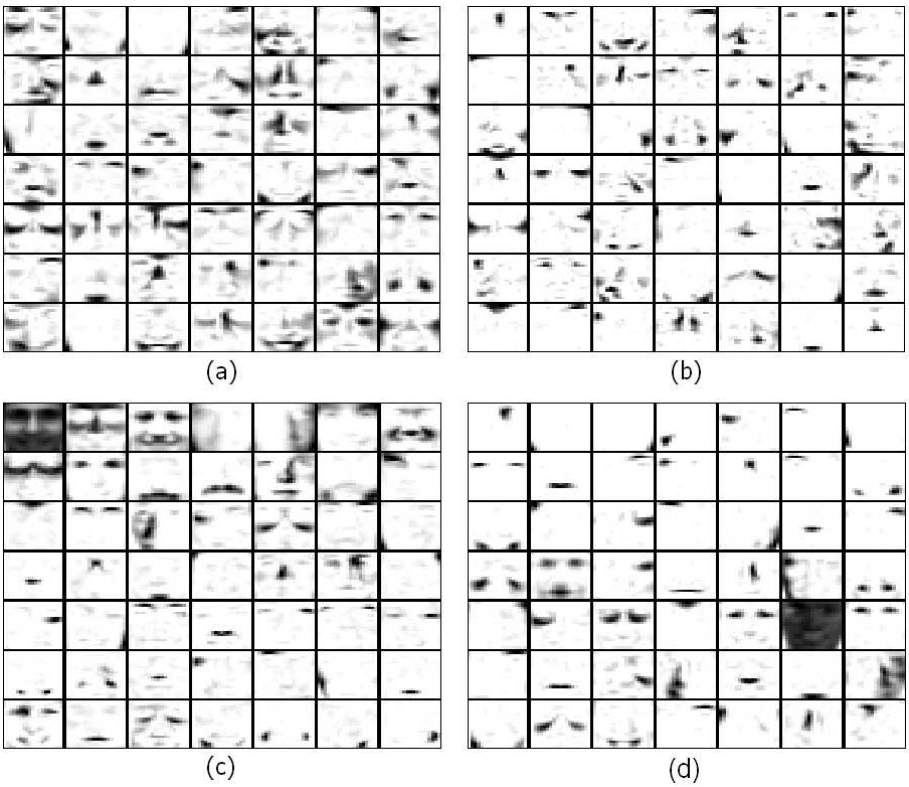

Figure 6.2: Basis elements ($U_{:k}$) generated for the CBCL image dataset: (a) NMF, (b) G-NMU, (c) R-NMU and (d) sNMF with sparsity of G-NMU.

factors recursively computed by R-NMU are in comparison not as sparse: as explained above, this is because R-NMU focuses on obtained representative parts, including relatively dense ones for the first few steps of the recursion. However, it features a much sparser weight vectors, giving again more credit to the hypothesis that better parts are extracted. The corresponding approximation error is higher than for other methods, because the intrinsically greedy approach taken by R-NMU cannot be globally better than a method that optimizes all the factors simultaneously.

Is the increased sparsity provided by G-NMU worth the increase in approximation error? Looking at the corresponding results for sNMU{G-NMU} and Hoyer{G-NMU}, i.e., for sparse NMF and Hoyer's algorithms with a similar target sparsity, it might seem at first that the answer is negative: the other methods return solutions with similar number of nonzeros (slightly higher for

	Plain	Improved	$s(U)$	$S(V)$	$sh(U)$	$sh(V)$
PCA	7.43	7.43	0	0	22	22
NMF	8.12	8.11	56	11	66	22
G-NMU	12.45	8.76	74	14	74	21
sNMF{G-NMU}	8.68	8.44	74	14	74	30
Hoyer{G-NMU}	9.33	8.78	69	6	73	16
R-NMU	16.42	10.89	53	52	63	64
sNMF{R-NMU}	10.23	9.49	50	50	56	57
Hoyer{R-NMU}	8.83	8.56	54	12	64	22

Table 6.1: Comparison of the relative approximation error and sparsity for the CBCL image dataset.

Hoyer) and a lower approximation error (8.68 and 9.33 instead of 12.45). Actually, this was expected: because it tries to return an underapproximation, i.e., factors such that $UV \lesssim M$, the entries in the error term $M - UV$ are mostly nonnegative, while the other techniques, with no underapproximation constraint, obtain a smaller norm of the error by choosing the entries of $M-UV$ to be roughly half negative, half positive. It is therefore not completely fair to compare directly the error of the NMU approach to the other techniques.

In order to compensate for this, a simple rescaling could be used, i.e., multiplying UV by a scalar since $UV \lesssim M$ (cf. Equation (4.7)). However, we chose a different procedure that has the advantage of benefiting all algorithms, including those whose error was not suffering from the underapproximation constraint. Once a solution is computed by one of the eight algorithms, we fix the zero entries of U and V and optimize the approximation error, i.e., $\min_{U,V\geq 0} ||M - UV||_F^2$, on the remaining (nonzero) entries (again, HALS can easily be adapted to handle this situation). In essence, this allows us to compare the sparsity patterns of the different solutions. We perform 100 additional HALS steps on each solution, and report the new relative approximation error in the third column of Table 6.1 ("Improved"). Note that sNMF and Hoyer's errors are also improved by this procedure; this can be explained by the fact that they were also not directly trying to minimize the approximation error (Hoyer had to take into account its sparsity constraint, and sNMF was influenced by the penalty terms added to the approximation error).

Looking now at the NMU solutions in a fairer comparison, we observe that their approximation error becomes very close to that of sNMF and Hoyer, in particular for G-NMU, and not very far from the denser NMF, so that we can conclude that the sparsity-approximation error compromise it offers is worthwhile.

6.6.6 Swimmer Database

For the swimmer image dataset described in Example 6.1 (256 images with 20×11 pixels), the eight basis elements obtained with the different algorithms are displayed on Figure 6.3 and the corresponding approximation errors and sparsity measures are reported in Table 6.2.

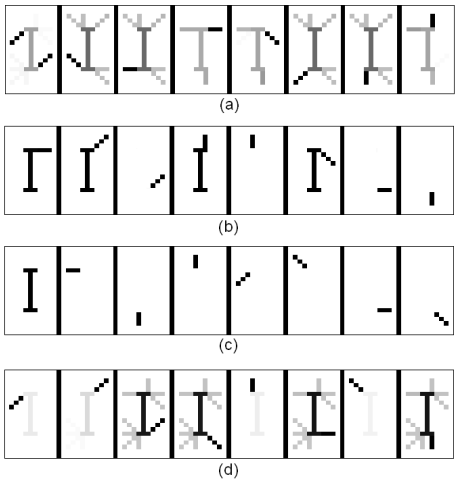

Figure 6.3: Basis elements for the swimmer image dataset: (a) NMF, (b) G-NMU, (c) R-NMU and (d) sNMF with sparsity of R-NMU.

As mentioned earlier, our two NMU algorithms are the only methods able to extract truly independent parts, while NMF and sNMF generate combinations of multiple parts. Note however that the solution generated by sNMF bears some similarity to the one of G-NMU.

6.6.7 Hubble Space Telescope Spectral Images

The next image dataset consists of 100 spectral images (128×128 pixels) of the Hubble telescope at different frequencies [126, 153], see Figure 6.4. With the choice $r = 8$, NMF generates a nearly exact factorization (relative error 0.29%), because the spectral reflectance of the Hubble telescope results from the additive linear combination of the reflectance of eight constitutive materials. Figure 6.5 and Table 6.3 provide the visual and computational results for this dataset.

Because NMF is already a nearly exact reconstruction (Table 6.3), the NMU constraints are somehow redundant: NMF and G-NMU are basically equivalent and return solutions with very similar sparsity measures (albeit with a slightly

Error	Plain	Improved	$s(U)$	$S(V)$	$sh(U)$	$sh(V)$
PCA	37.98	37.98	77*	0	67	17
NMF	40.41	40.41	84	45	73	67
G-NMU	47.70	46.85	94	75	85	78
sNMF{G-NMU}	50.52	42.04	89	66	84	73
Hoyer{G-NMU}	42.04	41.91	90	45	80	63
R-NMU	50.92	50.71	98	66	93	65
sNMF{R-NMU}	41.66	41.17	85	66	80	79
Hoyer{R-NMU}**	/	/	/	/	/	/

Table 6.2: Comparison of the relative approximation error and sparsity for the swimmer image dataset. *This value is very close to the percentage of zero rows in the matrix M (corresponding to pixels that are equal to zero in all images): in general, PCA factors feature a zero component when all the entries of either one row or one column of the input matrix are equal to zero. **When imposing the sparsity level of R-NMU ($sh(U) = 0.93$), Hoyer's algorithm was not able to converge, probably because it is not well adapted to handle high sparsity constraints. Note that sNMF is also sensitive to high sparsity requirements: high penalty terms sometimes lead to optimal zero factors ($U_{:k} = 0$ for some k), which had to be reinitialized.

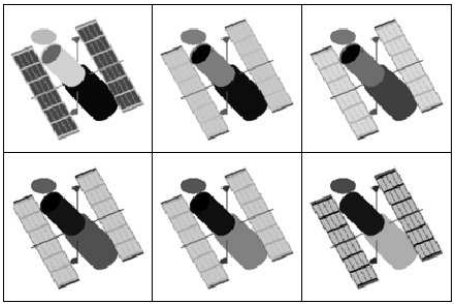

Figure 6.4: Sample of Hubble space telescope spectral images.

lower error for G-NMU). For that reason, sNMF{G-NMU} and Hoyer{G-NMU} return results nearly identical to NMF and are omitted from the table.

Recursive R-NMU extracts parts in order of importance: first, a global picture of the telescope and then its different constitutive parts. This allows it to generate the sparsest solution, with several basis elements representing well-delimited constitutive parts of the telescope not identified by the other methods (see also Chapter 7).

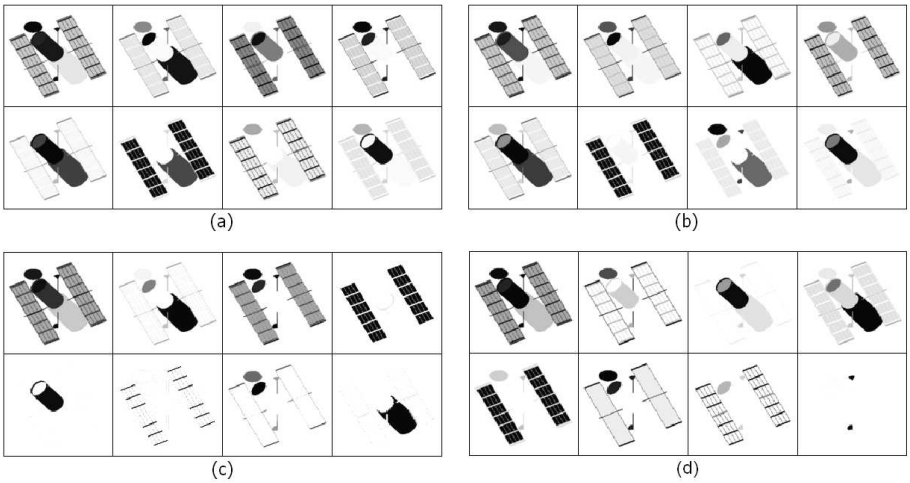

Figure 6.5: Basis for the Hubble telescope: (a) NMF, (b) G-NMU, (c) R-NMU and (d) sNMF with sparsity of R-NMU.

Error	Plain	Improved	$s(U)$	$s(V)$	$sh(U)$	$sh(V)$
PCA	0.01	0.01	57	0	62	25
NMF	0.29	0.29	64	5	57	35
G-NMU	0.52	0.11	64	4	60	31
R-NMU	3.74	1.37	79	30	71	62
sNMF{R-NMU}	0.48	0.37	73	28	66	64
Hoyer{R-NMU}	0.77	0.68	75	0	71	12

Table 6.3: Comparison of the relative approximation error and sparsity for the Hubble telescope image dataset.

6.6.8 Kuls Illuminated Faces

A static scene was illuminated from many directions with a moving light source to produce the Kuls image dataset[5]. It consists of 20 images (64×64 pixels) of a face. Because the images are very similar, most of the information (more than 70 percent) can be expressed with only one factor. The remaining information resides in the different orientations of the lighting. Computational and visual results for a rank-5 factorization are given by Table 6.4 and Figure 6.6. We observe that NMF and G-NMU obtain similar results: even though they are both able to extract several faces with different lighting orientations, they do

[5]Available at `http://www.robots.ox.ac.uk/~amb/`.

not extract a sparse and part-based representation.

R-NMU first extracts a face illuminated from all directions, and then complementary parts representing different orientations of the lighting (successively on the fourth row of Figure 6.6: global then light from the right, left, bottom and top). This nice recursive extraction of the information is a direct consequence of the underapproximation constraints. Although sNMF (with the same sparsity requirement as R-NMU) is also able to extract a part-based representation with a slightly better approximation error, only two components are well-identified (left and right lighting mixed with top and bottom lighting).

Error	Plain	Improved		$s(U)$	$s(V)$	$sh(U)$	$sh(V)$
PCA	4.36	4.36		0	0	23	15
NMF	4.38	4.38		1	7	9	38
G-NMU	6.27	4.49		3	20	8	48
sNMF{G-NMU}	4.42	4.41		2	20	8	47
Hoyer{G-NMU}	4.60	4.71		2	25	8	53
R-NMU	8.13	5.73		29	31	38	67
sNMF{R-NMU}	5.24	5.01		29	31	32	59
Hoyer{R-NMU}	6.82	6.54		0	71	6	92

Table 6.4: Comparison of the relative approximation error and sparsity for the Kuls image dataset.

Some properties of R-NMU will be analyzed in the next chapter, and will provide an explanation for such a nice extraction. In fact, we will see in Chapter 7 that rows of matrix M which share similar information will be extracted together with the R-NMU approach (see Theorem 7.4). For the Kuls faces, each row of matrix M corresponds to a pixel, i.e., to a specific location on the image. Hence the rows corresponding to pixels located close to each other in the images will have similar 'shapes' (because the lighting orientation is similar), which is the reason why they are extracted together by R-NMU.

Conclusion

In order to solve the NMF problem in a recursive way, we have introduced a new problem, namely nonnegative matrix underapproximation (NMU), which was shown to be $\mathcal{NP}$-hard using its equivalence with the biclique problem. The additional constraints of NMU are shown to induce sparser factors and to lead naturally to a better part-based representation of the data, while keeping

Figure 6.6: Basis for the Kuls image dataset, from top to bottom: sample of images, NMF, G-NMU, R-NMU, sNMF with sparsity of R-NMU.

a fairly good reconstruction. We proposed an algorithm based on Lagrangian relaxation to find approximate solutions to NMU.

We tested two factorization methods based on this algorithm, one with full

recursion (R-NMU), the other without recursion (G-NMU), on several standard image datasets. After suitable post-processing, we observed that the factors computed by these methods indeed offer a good compromise between their achieved sparsity and the resulting approximation error, comparable or sometimes superior to that of two standard sparse nonnegative matrix factorization techniques.

These two variants can be contrasted in the following way: where G-NMU mainly focuses on finding sparse factors with small reconstruction error, in the same spirit as sNMF and Hoyer, R-NMU typically computes an even sparser factorization corresponding to a better part-based representation, albeit with a moderate increase in the reconstruction error (due to the greedy approach). Moreover, this second variant is useful in situations where the factorization rank is not fixed a priori: the fact that it is recursive allows the user to stop the procedure as soon as the reconstruction error becomes satisfactory, without having to recompute a completely different solution from scratch every time a higher-rank factorization needs to be considered.

In the next chapter, we use R-NMU for hyperspectral image analysis, and provide some further theoretical and experimental evidences that it is indeed able to efficiently cluster such data automatically.

Chapter 7

Hyperspectral Data Analysis using Underapproximation

A hyperspectral image is a set of images of the same object or scene taken at different wavelengths. Each image is acquired by measuring the reflectance (i.e., the fraction of incident electromagnetic power reflected) of each individual pixel at a given wavelength, see the illustration on Figure 7.1. A crucial aspect

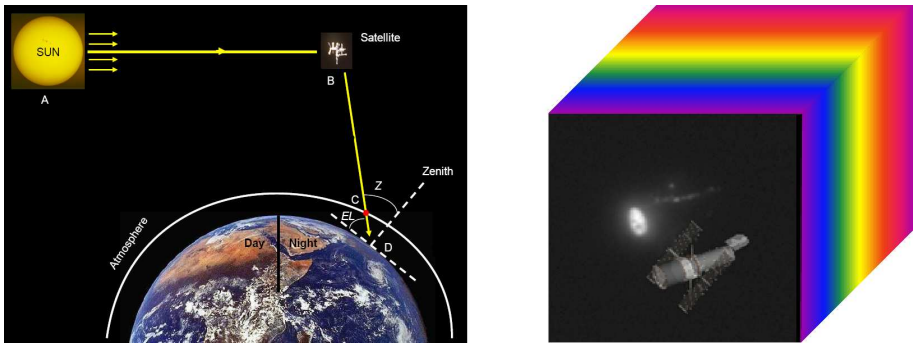

Figure 7.1: Illustration of hyperspectral image acquisition: (left) reflectance measurements, and (right) hyperspectral image.

of hyperspectral image analysis is the identification of materials present in the object or scene being imaged. This can be for example used to classify the thousands of objects in orbit around the earth (e.g., military and commercial satellites, debris).

Dimensionality reduction techniques such as PCA are widely used as a preprocessing step for hyperspectral image analysis in order to reduce the computational cost of classification algorithms (such as k-means or nearest neighbor, see, e.g., [24]) while keeping the pertinent information. In this context, it is often preferable to take advantage of the intrinsic properties of hyperspectral data: *the spectral signature of each pixel results from the additive combination of the nonnegative spectral signatures of its constitutive materials*. Taking these nonnegativity constraints into account enhances interpretability of the extracted factors. This can be done using nonnegative matrix factorization.

Let the matrix M be constructed as follows: each 2D image corresponding to a wavelength is vectorized and is a column $M_{:j}$, while each row $M_{i:}$ corresponds to the spectral signature of a pixel, see Figure 7.2. Then a decomposition

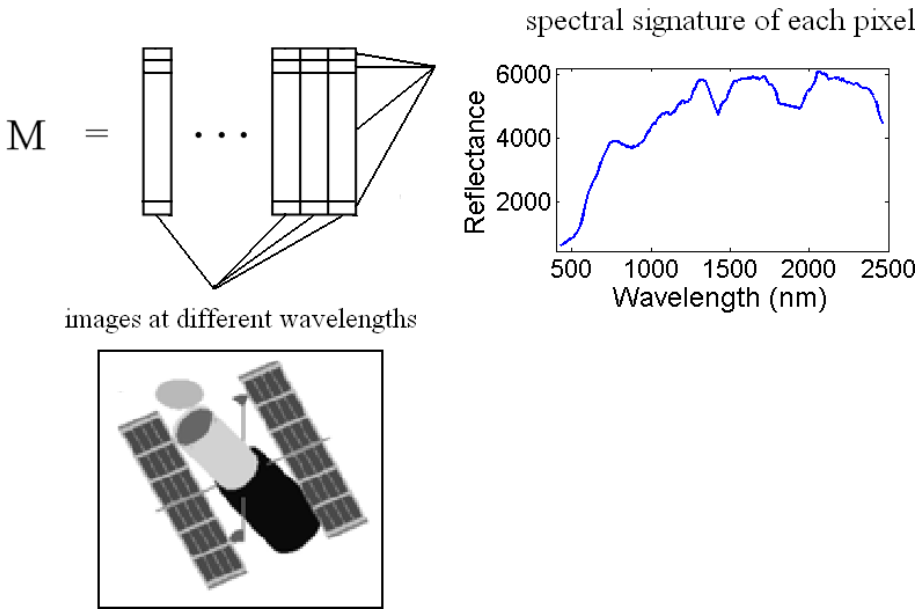

Figure 7.2: Construction of matrix M.

$(U, V) \geq 0$ of M can be interpreted as follows

$$M_{i:} \approx \sum_k U_{ik} V_{k:} \quad \forall i,$$

and the spectral signature of each pixel ($M_{i:}$, a row of M) is approximated with a nonnegative linear combination (with weights U_{ik}, representing abundances) of end-members signatures ($V_{k:}$, rows of V) which hopefully correspond to the signatures of the constituent materials of the hyperspectral image.

This model[1] has been successfully applied for identification of the materials and the spectral unmixing in hyperspectral images [126, 153]. However, NMF features some drawbacks. In particular,

1. NMF is a $\mathcal{NP}$-hard nonlinear optimization problem with many local minimizers, see Chapter 3. In practice, NMF is solved using iterative schemes based on nonlinear optimization techniques, see Chapter 4.

2. The optimal solution is in general non-unique[2] which makes the problem ill-posed [103]. Additional constraints are often added to reduce the degrees of freedom, e.g., smoothness [94, 126], sparsity [94], orthogonality [54, 109], minimum-volume [119], sum-to-one constraint of the rows of U [118].

3. One needs to recompute a solution from scratch when the rank of the approximation is modified.

In this chapter, we use recursive NMU from Chapter 6 which overcomes some of these drawbacks[3] (2. and 3. above) as a dimensionality reduction technique to analyze hyperspectral data. We provide some theoretical evidences that this technique is in fact able to automatically detect materials in hyperspectral images and illustrate this on a simple example. We then propose a new approach based on ℓ_1-norm minimization (instead on the ℓ_2-norm considered in Chapter 6), and explain why it is theoretically more appealing: it is potentially able to extract the materials in a more efficient and robust way. An algorithm is proposed with the same computational complexity as the one presented in Section 6.5. Finally, we experimentally show the efficiency of these new strategies on hyperspectral images associated with space object material identification, and on HYDICE and related remote sensing images.

7.1 The Ideal Case

If we assume that each pixel contains only one material, the corresponding matrix has the following form:

Assumption 7.1. $M \in \mathbb{R}_+^{m \times n}$ with $M = WH$ where

[1] (NMF) is closely related to an older approach based on the geometric interpretation of the distribution of spectral signatures: they are located inside a low-dimensional simplex whose vertices are the pure pixel signatures (i.e., the signatures of each individual material) [17, 40]. This is also related to the geometric interpretation of the nonnegative rank given in Chapter 3.

[2] Any invertible matrix D such that $UD \geq 0$ and $D^{-1}V \geq 0$ generates an equivalent solution.

[3] Unless $\mathcal{P} = \mathcal{NP}$, drawback 1 can not be 'resolved' since the underlying problem of spectral unmixing is of combinatorial nature [92].

1. $W \in \{0,1\}^{m \times r}$ is a binary matrix of dimension m by r, with $r \leq \min(m,n)$. W is full column rank, and has one and only one element equals to one in each row with

$$W_{ik} = 1 \quad \Longleftrightarrow \quad \text{pixel } i \text{ contains material } k.$$

   This implies that its columns are orthogonal $W_{:i}^T W_{:j} = 0 \; \forall i \neq j$.

2. $H \in \mathbb{R}_+^{r \times n}$ is of full row rank r.

Of course, recovering W and H in this setting is trivial and, in practice, because of blurring and other mixing effects, limited resolution and mixed materials, the spectral signature of each pixel will be a mixture of spectral signatures of several materials (in particular, pixels located at the boundary of materials) plus noise. However, classifying each pixel into a single category amounts to approximating M with a matrix satisfying Assumption 7.1. This problem is referred to as orthogonal NMF (oNMF) and is equivalent to k-means clustering [54].

We now show that, under some mild assumptions, the underapproximation technique is able to retrieve the underlying structure in the ideal case, when each pixel corresponds to only one material. This will shed some light on the behavior of the recursive algorithm based on underapproximations presented in Chapter 6, and justify its efficiency when dealing with non-ideal hyperspectral images.

Recall that the rank-one NMU problem is the following

$$\min_{u \in \mathbb{R}_+^m, \, v \in \mathbb{R}_+^n} \quad ||M - uv^T||_F^2 \quad \text{such that} \quad uv^T \leq M. \tag{NMU-1}$$

It is convex in u and v separately, and the corresponding optimal solutions can actually be trivially computed: for $v \geq 0$,

$$u^*(v) = \text{argmin}_{u \geq 0, uv^T \leq M} ||M - uv^T||_F, \quad u_i^*(v) = \min_{\{j \,|\, v_j \neq 0\}} \left\{ \frac{M_{ij}}{v_j} \right\} \forall i, \tag{7.1}$$

and for $u \geq 0$,

$$v^*(u) = \text{argmin}_{v \geq 0, uv^T \leq M} ||M - uv^T||_F, \quad v_j^*(u) = \min_{\{i \,|\, u_i \neq 0\}} \left\{ \frac{M_{ij}}{u_i} \right\} \forall j, \tag{7.2}$$

and they correspond to the stationarity conditions of (NMU-1). One can check that these first-order stationarity conditions (7.1) and (7.2) are the same for (many) other norms than $||.||_F$, such as the ℓ_1-norm which will be analyzed later in Section 7.3.

7.1.1 First Rank-One Factor

As for PCA, the first rank-one factor of NMU will reduce the error the most and will already be a fairly good approximation of the hyperspectral data.

Lemma 7.1. *Let (u, v) be a nontrivial stationary point of* (NMU-1) *(i.e., $u \neq 0$ and $v \neq 0$), then the residual $R = M - uv^T$ has at least one zero by row and by column.*

Proof. This follows directly from Equations (7.1) and (7.2). □

Lemma 7.2. *Let (u, v) be a nontrivial stationary point of* (NMU-1) *for $M = WH$ satisfying Assumption 7.1, then the residual $R = M - uv^T$ can be written as $R = WH'$ for some $H' \geq 0$.*

Proof. Because columns of W are binary and orthogonal, each row of M is equal to a row of H. Therefore, the entries of u corresponding to the rows of M equal to each other must take the same value, i.e., $\forall i \in \{1, 2, \ldots, r\}, \forall k, l \in \operatorname{supp}(W_{:i}) : \; u_k = u_l$. In fact, one can check that for $v \neq 0$, the solution of Equation (7.1) is unique. It follows that $u = Wd$, for some $d \in \mathbb{R}^r_+$, and then $R = WH - (Wd)v^T = W[H - dv^T]$. The facts that R is nonnegative and that W is binary and orthogonal implies that $H' = H - dv^T \geq 0$. □

Corollary 7.3. *Let (u, v) be a nontrivial stationary point of* (NMU-1) *and $M > 0$, then $u > 0$ and $v > 0$. Moreover, the residual $R = M - uv^T$ can be written as $R = WH'$ for some $H' \geq 0$ with at least one zero by row and by column in H'.*

Proof. Positivity of u and v follows directly from Equations (7.1) and (7.2) while structure of the residual matrix R is a consequence of Lemmas 7.1 and 7.2. □

Let us use notation of Corollary 7.3. We observe that it is typically very unlikely for the sparsity pattern of a row of H' to be contained in the sparsity pattern of another row, i.e., that

$$I = \overline{\operatorname{supp}}(H'_{i:}) \subset \overline{\operatorname{supp}}(H'_{j:}), \quad \text{for some } \; i \neq j, \tag{7.3}$$

for some non-empty set $I \subset \{1, 2, \ldots, n\}$. There are two basic reasons for this fact

1. We know there is at least one zero by row and by column in H' (Corollary 7.3). Clearly,

$$H'(i, I) = H'(j, I) = 0 \iff H(i, I) = \alpha \, H(j, I),$$

for some constant $\alpha > 0$. In fact, recall that $H'(i, I) = H(i, I) - d_i v(I)$ so that $H'(i, I) = H'(j, I) = 0$ if and only if $H(i, I) - d_i v(I) = H(j, I) -$

$d_j v(I) = 0$, i.e., $H(i,I) = \frac{d_i}{d_j} H(j,I)$. If $|I| \geq 2$ and if we assume that H is generated randomly, the probability of having $H(i,I) = \alpha H(j,I)$ is zero (randomly generated vectors in two dimensions or more are multiple of each other with probability zero). If $|I| = 1$, it means that at least one row of H' has only one zero element. We know that there are at least n zeros in H' (one by column) and at least one zero in each of the r rows of H'. There are still at least $(n-r)$ zeros to be placed in the r rows of H'. Assuming there are only $(n-r)$ zeros (typically, there are many more zeros in the residual) and that they are distributed uniformly among the rows of H', we can compute the probability of having the sparsity pattern of one row contained in the sparsity pattern of another[4] in case $|I| = 1$ in Equation (7.3) above. Figure 7.3 displays this probability for $n = 210$ (which is the number of spectral bands for HYDICE images, cf. Section 7.4) with respect to r (number of materials). For example, for $r \leq 25$ (i.e., less than 25 materials present in the image), the probability for one row of H' to have only one zero and for another row of H' to have a zero at the same position is smaller than 10^{-2}.

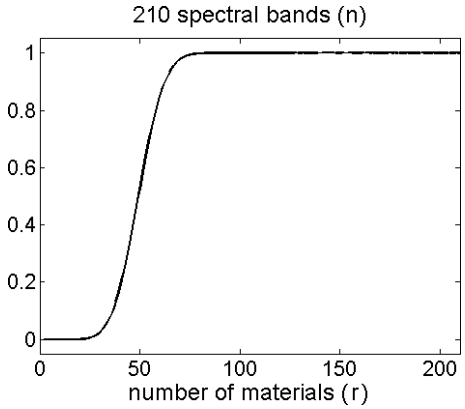

Figure 7.3: Probability for at least one row of H' to have one zero element, while another row has a zero at the same position for $n = 210$, assuming that H is randomly generated, and that H' contains (only) n zeros which are uniformly distributed (with at least one zero by row).

2. In practice, it is observed that the zeros are not distributed uniformly among the rows of H', and, typically, they have approximately the same

[4]We added up the probabilities to have i rows with only one zero element multiplied by the probability for at least one of these zeros to be located at the same position of another one (either in one of these i rows with one zero, or in the remaining $r - i$).

number of zeros $\mathcal{O}(\frac{n}{r})$, located at different positions.

Using another cost function (the sum of the logarithms of the ratios between the entries of dv^T and $H > 0$), it can be proved that the problem is equivalent to a flow problem (using a logarithmic change of variables), see Problem (Flow) in Section 6.4. In contrast with item 1. above, if H is a square matrix ($r = n$) and (d, v) an optimal solution of (Flow), the zeros of $H - dv^T$ will be located on the diagonal of H' (up to a permutation)[5] and no row will share common zeros. When using the Frobenius norm as an objective function, it seems that the zeros follow the same sparsity pattern, even though we do not have a proof for this fact. However, notice that problem (Flow) shares the same stationarity conditions as (NMU-1) (see Equations (7.1) and (7.2)), so that optimal solutions of (Flow) are stationary points of (NMU-1).

Conclusion. After the first NMU recursion, the residual R can be written in the same form as $M = WH$ (cf. Assumption 7.1) with $R = WH'$, and it is highly probable that

$$\overline{\text{supp}}(H'_{i:}) \not\subseteq \overline{\text{supp}}(H'_{j:}) \quad \forall i \neq j,$$

i.e., that no row of H' has its sparsity pattern contained in the sparsity pattern of another row. If this property holds, we will say that H' *satisfies the sparsity pattern assumption*.

7.1.2 Next Rank-One Factors

Assuming that $M = WH$ satisfies Assumption 7.1 and that H satisfies the sparsity pattern assumption, we show that the recursion outlined above will eventually locate each material individually.

Theorem 7.4. *Let (u, v) be a nontrivial stationary point of* (NMU-1) *for $M = WH$ satisfying Assumption 7.1 and $\overline{\text{supp}}(H_{i:}) \not\subseteq \overline{\text{supp}}(H_{j:}) \forall i \neq j$, i.e., H satisfies the sparsity pattern assumption. Then $R = M - uv^T = WH'$, with $u = Wd$ for some $d \in \mathbb{R}^r_+$ so that $H' = H - dy^T \geq 0$. Moreover,*

$$\text{supp}(u) = \cup_{i \in \Omega} \text{supp}(W_{:i}), \quad \textit{for some } \Omega \subset \{1, 2, \ldots, r\},$$

and

$$\Omega = \{i\} \iff H'_{i:} = 0 \iff d_i\, v = H^T_{i:}, \tag{7.4}$$

for some $1 \leq i \leq r$. $\Omega = \{i\}$ amounts to extracting the material i.

[5] This is related to the way assignment problems are solved [63]. One can show that a solution (d, v) of (Flow) (cf. Section 6.4) is optimal if and only if the matrix D with $d_{ij} = \log(H_{ij}) - \log(d_i) - \log(v_j)$ has, up to a permutation of its rows or columns, zeros on its diagonal (this is how the Hungarian method used to solve assignment problems has been designed).

Proof. The first part is a consequence of Corollary 7.3. It remains to show that Equation (7.4) holds. The second equivalence is trivial since $H'_{i:} = H_{i:} - d_i v^T$ is equal to zero for some i if and only if $d_i v = H_{i:}^T$. For the first equivalence, observe that $d_i v = H_{i:}^T$ implies that $\Omega = \{i\}$ because of the underapproximation constraints and the sparsity pattern assumption. In fact, since v has the same support as $H_{i:}$, we have $\forall j \neq i, \exists k$ s.t. $H_{jk} = 0$ and $v_k > 0$ implying $d_j = 0$. Finally, it is clear that for $\Omega = \{i\}$, the solution obtained with Equation (7.2) is $v = \frac{1}{d_i} H_{i:}^T$. □

Theorem 7.4 implies that, at each step of the NMU recursion, *a set of materials are extracted together*. Moreover, a material is extracted alone if and only if the corresponding row of H' is set to zero. Since the recursive approach outlined above will eventually end up with a zero matrix (say, after r_u steps), we will have that

$$M = \sum_{i=1}^{r_u} u_i v_i^T,$$

and, under the sparsity pattern assumption (at each step of the recursion),

$$\forall 1 \leq i \leq r, \exists 1 \leq j \leq r_u \quad \text{s.t.} \quad \operatorname{supp}(u_j) = \operatorname{supp}(W_{:i}).$$

In fact, for the residual $R = WH'$ to be equal to zero, all the rows of H' must be identically zero. This feature of the NMU recursion will be experimentally verified in Section 7.4.

Remark 7.1. The sparsity pattern assumption is a sufficient but not a necessary condition for exact recovery. For example, if two rows with the same sparsity pattern are extracted together, it is likely that the corresponding optimal solution will not be exactly equal to one of these two rows (because there are linearly independent) and therefore, at the next step, it is likely that they will satisfy the sparsity pattern assumption.

7.1.3 Illustration of Basis Recovery with NMU vs NMF

Let us construct the following synthetic data: 4 binary orthogonal images of 5×5 pixels (which are the columns of W, $W \in \{0,1\}^{25\times 4}$, see the top image of Figure 7.5) are randomly mixed ($H \in \mathbb{R}^{4\times 25}$ is randomly generated with uniform distribution between[6] 0 and 1) to generate a 25×25 matrix $M = WH$ satisfying Assumption 7.1. Figure 7.4 displays a sample of the 25 images contained in the columns of M, which then result from the nonnegative linear combination of the columns of W. Figure 7.5 displays the original images and the basis elements obtained with NMF and NMU. As was mentioned in Section 7.1.1, the first rank-one factor of NMU will reduce the error the most, and we

[6] We used the function rand(4,25) of MATLAB®.

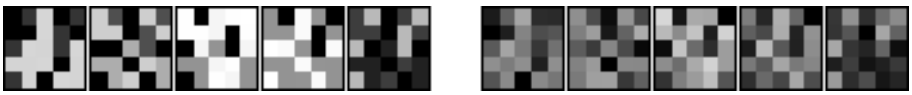

Figure 7.4: Sample of images of the data matrix M: clean (left) and with mixed pixels (right).

include it and list a total of five for NMU (since each end-member has been extracted individually after 5 steps, the residual error is equal to zero, i.e., the approximation is exact, see Theorem 7.4). We observe that NMF is not able

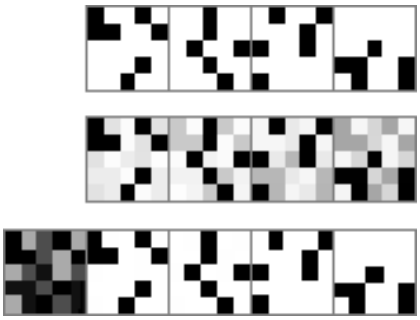

Figure 7.5: From top to bottom: four original images (i.e., columns of W), basis elements obtained with NMF and with NMU.

to extract perfectly the four original basis elements (even though the objective function is equal to zero; the reason is the non-uniqueness of the solution: NMF retrieves a mixture of the basis elements) while NMU is able to do the extraction[7].

7.2 The Non-Ideal Case

As we have already mentioned, practical problems don't have the nice structure mentioned in Assumption 7.1 and the spectral signature of most pixels results from a combination of several materials. What can we expect of NMU in that case? Since the data matrix is positive, the first rank-one factor will still be a mixture of all materials (cf. Lemma 7.1). It seems more difficult to provide theoretical guarantees for the next factors in more general settings

[7]Notice that for $n = 25$ and $k = 4$, the probability for two rows of H' to satisfy the sparsity pattern assumption is larger than $1-10^{-8}$, if we assume that the zeros are uniformly distributed, and that H is randomly generated (which is the case here), see Section 7.1.1.

and this will be a topic for further research. However, extracting a single constitutive material would allow one to approximate all the pixels containing it (removing from their spectral signature this component) and, since NMU aims at extracting components explaining the data as closely as possible in order to reduce the error the most, this indicates that NMU is encouraged to extract constitutive materials in non-ideal cases.

For example, let us add to the matrix W in the illustration of the previous paragraph a randomly generated matrix, uniformly distributed between 0 and 0.5. This means that each pixel is now a mixture of several materials but one material is still predominant. Figure 7.6 displays the visual result: NMF performs even worse, while NMU is still able to extract the original parts fairly well. It actually provides a soft clustering for each pixel[8], as it will also be shown in Section 7.4.

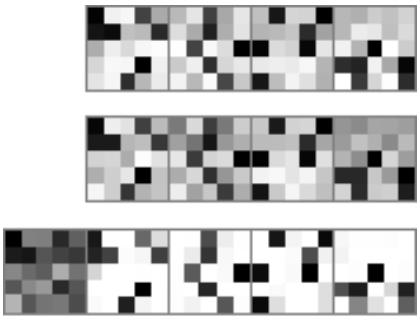

Figure 7.6: From top to bottom: 4 original images (i.e., columns of W), basis elements obtained with NMF and with NMU.

7.3 ℓ_0-Pseudo-Norm Minimization and ℓ_1-Norm Relaxation

Ideally, each basis element extracted with the recursive approach outlined previously should correspond to a different material present in the hyperspectral image: we would like that each extracted rank-one factor corresponds to only one material, i.e., that only a submatrix of M (a set of rows of M) corresponding to pixels containing the same material is approximated at each step.

[8]Soft clustering means that a single element of the dataset can be assigned to different clusters; a (nonnegative) weight being attached to each cluster (e.g., corresponding to the probability of the pixel to belong to the cluster). In a hyperspectral image, it makes sense since the pixels can be composed of different materials (the clusters) with different abundances (the weights, summing to one).

Unfortunately, the ℓ_2-norm is not appropriate for this purpose: it is very sensitive to 'outliers', i.e., it cannot neglect some entries of the matrix M and set only a subset of the entries of the residual error to zero. It is more likely that it will try to approximate several materials at the same time in order to avoid large entries in the residual error. For this reason, we will see that the ℓ_2-norm based algorithm typically first extracts several materials together.

If the ℓ_0-'norm' is used instead, i.e., if the number of zero entries in the residual is maximized, one can check that for a matrix satisfying Assumption 7.1, this will lead to an *exact recovery in r steps*; because extracting one material (i.e., taking $v = H_{i:}^T$ for some i at each step) will lead to the highest number of zeros in the residual $R = M - uv^T$ (rows corresponding to the extracted material are identically zero; plus one zero by row and by column for the other ones). Unfortunately, ℓ_0-'norm' minimization is very difficult to work with (non-differentiable, non-convex even when one factor is fixed, i.e., $||M - uv^T||_0$ for u or v fixed). Moreover, in practice, because of noise and blur, the ℓ_0-'norm' would not be appropriate since rows of M representing the same material cannot be approximated exactly. However, the ℓ_1-norm, often used as a convex heuristic to approximate ℓ_0-'norm'[9], is known to be less sensitive to outliers and is then disposed to let some entries of the error large, in order to approximate better other entries. We will experimentally observe in Section 7.4 that using ℓ_1-norm allows us to extract materials individually in a more efficient manner, i.e., using a smaller number of recursive steps.

7.3.1 Algorithm for ℓ_1-Norm Minimization

Using the idea of Lagrangian duality presented in Secton 6.5, we propose to solve[10]

$$\min_{u \in \mathbb{R}_+^m,\, v \in \mathbb{R}_+^n} \quad ||(M - \Lambda) - uv^T||_1 = \sum_{ij} |(M - \Lambda) - uv^T|_{ij}, \tag{7.5}$$

where $\Lambda \in \mathbb{R}_+^{m \times n}$ represents the Lagrangian multipliers associated with the underapproximation constraints. Let us apply the same procedure as in Section 6.5, i.e., alternate optimization over u, v and Λ. Fixing v and Λ and noting $A = M - \Lambda$, u can be optimized by solving the following m independent problems

$$\min_{u_i \geq 0} ||A_{i:} - u_i v^T||_1 = \sum_j |A_{ij} - u_i v_j| = \sum_{j \,\in\, \mathrm{supp}(y)} v_j \left| \frac{A_{ij}}{v_j} - u_i \right| + \sum_{j \,\notin\, \mathrm{supp}(v)} |A_{ij}|,$$

[9] The convex envelope of $||x||_0$ on the set $S = \{x \in \mathbb{R}^n \mid ||x||_2 \leq 1\}$ is $||x||_1$, i.e., $||x||_1$ is the largest convex function smaller than $||x||_0$ on S, see [129] and the references therein.

[10] Note that Λ does not correspond to the Lagrangian dual variables of $\min_{u \geq 0, v \geq 0, uv^T \leq M} ||M - uv^T||_1 = \sum_{ij} |M - uv^T|_{ij}$. However, this formulation is closely related to the Lagrangian relaxation and allows us to use the same derivations as for Algorithm 12.

which can be solved by computing the weighted median of z with $z_j = (A_{ij}/v_j)\,\forall j$ with weights v_j. The same can be done for v by symmetry, and we propose to update u and v with (steps 6 and 7 of Algorithm 12)

$$u_i = \max\Big(0, \text{weighted-median}\Big(\frac{[(M-\Lambda)^i(J)]}{[v(J)]}, v(J)\Big)\Big)\ \forall i, \quad J = \text{supp}(v),$$

and

$$v_j = \max\Big(0, \text{weighted-median}\Big(\frac{[(M-\Lambda)_j(I)]}{[u(I)]}, u(I)\Big)\Big)\ \forall j, \quad I = \text{supp}(u).$$

The weighted median of an n dimensional vector can be computed in $\mathcal{O}(n)$ operations, cf. [86] and the references therein, so that the algorithm can be implemented in $\mathcal{O}(mn)$ operations per iteration when the data matrix M has dimension $m \times n$. We will refer to this algorithm as ℓ_1-NMU. The ℓ_2-norm version (where u and v are taken as the optimal solution of the ℓ_2-norm minimization problem, see Algorithm 13) has the same computational complexity even though in practice ℓ_1-NMU will be slower, but only up to a constant factor[11].

The rest of the algorithm is implemented as follows, see Algorithm 13. Matrix Λ is updated with a subgradient type update (step 10, as in Algorithm 12). If Λ is too large, it might happen that u and/or v are set to zero leading to a trivial stationary point. We propose to reduce the value of Λ if that happens and to set u and v to their old values (step 12). Iterates (u, v) are initialized with the optimal rank-one solution of the ℓ_2-norm unconstrained problem (i.e., the optimal rank-one approximation of the residual, step 2, corresponding to $\Lambda = 0$); Λ is initialized with the nonnegative part of the residual matrix (step 4). Since the algorithm is not guaranteed to generate a feasible solution[12], only the nonnegative part of the residual is considered (step 15).

Note that the updates of u and v share some similarities with the power method (applied to $M - \Lambda$, with projection on the nonnegative orthant) which computes the maximum singular value and its corresponding left and right singular vectors, see Section 2.2.1. It seems that Algorithm 12 behaves similarly as the power method in the sense that it converges in general relatively fast. Extensive experiments on a host of data and applications allow us to conclude that 100 iterations at each step of the recursion are sufficient (i.e., maxiter = 100 which will be used for the numerical experiments, see Section 7.4).

[11] Implementation of both algorithms is available at `http://www.core.ucl.ac.be/~ngillis/`.

[12] This feature is actually an advantage for practical applications. In fact, this gives the algorithm some flexibility when dealing with noisy data. However, one can obtain a feasible stationary point by using updates (7.1) and (7.2) as a post-processing step.

Algorithm 13 Recursive NMU for ℓ_1- and ℓ_2-norms

Require: $M \in \mathbb{R}_+^{m\times n}$, $r > 0$, maxiter, norm $= 1$ or 2.
Ensure: $(U, V) \in \mathbb{R}_+^{m\times r} \times \mathbb{R}_+^{n\times r}$ s.t. $UV \lesssim M$.

for $k = 1 : \mathrm{r}$ **do**
  $[u, v] =$ optimal rank-one approximation(M);
  $U_{:k} \leftarrow u$; $V_{k:} \leftarrow v$;
  $\Lambda \leftarrow \max(0, -(M - uv^T))$;
  for $p = 1 :$ maxiter **do**
    if norm $= 2$ **then**
      $u \leftarrow \max\left(0, \frac{(M-\Lambda)v}{||v||_2^2}\right)$;
      $v \leftarrow \max\left(0, \frac{(M-\Lambda)^T u}{||u||_2^2}\right)$;
    else if norm $= 1$ **then**
      $u_i = \max\left(0, \text{w-median}\left(\frac{[(M-\Lambda)^i(J)]}{[v(J)]}, v(J)\right)\right)\ \forall i, J = \text{supp}(v)$;
      $v_j = \max\left(0, \text{w-median}\left(\frac{[(M-\Lambda)_j(I)]}{[u(I)]}, u(I)\right)\right)\ \forall j, I = \text{supp}(u)$;
    end if
    if $u \neq 0$ and $v \neq 0$ **then**
      $U_{:k} \leftarrow u$; $V_{k:} \leftarrow v$;
      $\Lambda \leftarrow \max(0, \Lambda - \frac{1}{p}(M - uv^T))$;
    else
      $\Lambda \leftarrow \frac{\Lambda}{2}$; $u \leftarrow U_{:k}$; $v \leftarrow V_{k:}$;
    end if
  end for
  $M = \max(0, M - U_{:k}V_{k:})$;
end for

Since only a rank-one matrix is computed at each step, (NMU-1) is in general well-posed in the sense that the optimal solution is *unique* (up to a scaling factor)[13]. In fact, for any rank-one nonnegative matrix A, there exists one and only one $(u, v) \geq 0$ such that $||u||_2 = 1$ and $A = uv^T$. In our experiments, we observed that NMU is much less sensitive to initialization and that, in general, when we allow several restarts of the algorithm with different initializations, it ends up with similar solutions (hopefully, close to the optimum). This is typically not the case with the standard NMF formulation because of non-uniqueness [103].

[13]Note that (NMF) with $r = 1$ is also well-posed; in fact, the optimal solution is unique if and only if the maximum singular value of M ($\sigma_1(M)$) is strictly greater than the second biggest singular value ($\sigma_2(M) < \sigma_1(M)$), cf. Section 2.2.

7.4 Numerical Experiments

In this section, Algorithms for ℓ_2- and ℓ_1-norm minimization proposed in Section 7.3 (ℓ_2-NMU and ℓ_1-NMU) are used as dimensionality reduction techniques for hyperspectral data in order to achieve *classification* (selecting from the basis elements the different clusters), and *spectral unmixing* (using nonnegative least squares).

In the first part, we analyze carefully the Urban HYDICE image (Sections 7.4.2) and a Hubble space telescope simulated image (Section 7.4.3) developed in [153]. To serve as a comparison, we also give the classification results obtained with k-means[14]; we plan to compare the NMU strategy with more sophisticated techniques in the future.

In the second part, we provide some visual results for aerial images of a desert region and of the San Diego airport (Sections 7.4.4 and 7.4.4), and for an eye image with four bands (Section 7.4.4).

7.4.1 Classification and Spectral Unmixing

NMU can be used as a standard dimensionality reduction technique and any type of post-processing procedure can be used to extract the constitutive parts of spectral data, e.g., k-means, nearest neighbor, etc. However, we have shown why NMU is potentially able to extract these parts automatically. Therefore, the simplest approach would be to visually select each cluster from the generated basis elements (i.e., manually selected the basis elements representing a single material). We stick to this approach and select, from the basis elements, each individual cluster: from the U matrix obtained with NMU, we only keep a subset of the columns, each corresponding to an individual material.

The second post-processing step is to normalize U. In fact, as NMF, NMU is invariant to the scaling of the columns of U ($\forall k\ U_{:k}V_{k:} = (\alpha U_{:k})(\alpha^{-1}V_{k:})\ \forall \alpha > 0$). Moreover, in the context of hyperspectral image analysis, rows of U have a physical interpretation: U_{ij} is the abundance of material j in pixel i. Therefore, $U_{ij} \leq 1\ \forall i, j$ and the columns of U are normalized with

$$U_{:j} = \frac{U_{:j}}{\max_i(U_{ij})} \quad \forall j.$$

This means that, for each rank-one factor extracted with the NMU procedure, the maximum abundance of each pixel for the corresponding spectral signature $V_{k:}$ is at most 1. Moreover, since rows of U correspond to abundances, $\sum_j U_{ij} = 1\ \forall i$ and we can scale the rows of U as follows

$$U_{i:} \leftarrow \frac{U_{i:}}{||U_{i:}||_1 + \epsilon}, \quad \epsilon \lll 1,$$

[14] We used the MATLAB® *kmeans* function, with default parameters.

so that they sum to one (except if they are identically zeros). This allows us to equilibrate the relative importance of each pixel in each basis element. With this procedure, we end up with a soft clustering: each pixel i is composed of several materials with the corresponding abundances given by $U_{i:}$.

Once the pixels have been classified, one might be interested in recovering the spectral signatures of each individual material (corresponding to the rows of matrix V in a decomposition $M \approx UV$), called the end-members. A standard approach is to solve a nonnegative least squares problem of the form

$$\min_{V \geq 0} \quad ||M - UV||_F^2, \tag{NNLS}$$

where U represents the basis vectors, with dedicated algorithms [27].

Remark 7.2. The rows of V in the NMU solution don't correspond directly to the spectral signatures of the end-members, because several materials are sometimes extracted together, in particular at the first steps of the algorithm. In order to get the spectral signature of an end-member from the NMU solution, one has to sum up all the rows of V corresponding to a column of U containing the desired end-member. For example, for $M > 0$, the first row of V always has to be used to obtain a spectral signature since all materials are present in the first basis element with $U_{:1} > 0$, see Corollary 7.3. It is therefore more convenient and more accurate to compute the spectral signatures by solving (NNLS).

7.4.2 Urban HYDICE Image

We consider first the Urban hyperspectral image[15] taken with HYper-spectral Digital Imagery Collection Experiment (HYDICE) air-borne sensors. We analyze the data where the noisy bands have been removed (162 bands left, originally 210), and the data cube has dimension $307 \times 307 \times 162$. Figures 7.7 and 7.8 give the basis elements of the ℓ_2- and ℓ_1-NMU decompositions. The Urban data is mainly composed of 6 types of materials: road, dirt, trees, roofs, grass and metal, as reported in [85]. Table 7.1 gives the index of the NMU basis elements corresponding to single materials.

Clusters	Road	Dirt	Trees	Roofs	Grass	Metal
ℓ_2 basis #	18	23	3	4	6	8
ℓ_1 basis #	16	17	2	5	6	7

Table 7.1: Basis elements obtained: cluster selection.

[15] Available at `http://www.agc.army.mil/hypercube/`.

Figure 7.9 shows the classification obtained from the basis elements obtained with NMU (cf. Figures 7.7 and 7.8, and Table 7.1), and with k-means which is only able to extract two materials correctly.

Figure 7.10 displays the results of the spectral unimxing procedure for both NMU algorithms (ℓ_2 and ℓ_1) which are compared to 6 end-members, as listed as the 'true' endmembers for this data in [85].

In this example, NMU performs relatively well and is able to detect all the materials individually, which can then be used to classify the pixels and finally recover the end-member signatures. We also note that, as predicted, the ℓ_2-NMU needs more recursion than ℓ_1-NMU to extract all materials individually (23 vs. 17). For example, it is interesting to observe that ℓ_1-NMU actually extracts the grass as two separate basis elements (6 and 10, cf. Figure 7.8). The reason is that the spectral signatures of the pixels in these two basis elements differ (especially after the 100^{th} band in the hyperspectral image): there are two types of grass with similar spectral signatures, and they are different enough to be assigned to different clusters (see Figure 7.11). Because ℓ_1-NMU is able to extract parts separately in a more efficient way (cf. Section 7.3), it is able to detect this 'anomaly' while ℓ_2-NMU is not. It also gives an explanation of the differences in the spectral signatures of the grass in Figure 7.10.

Number of Spectral Bands

In Section 7.1, a sufficient condition was presented to recover each material individually: the number of spectral bands (n) should be sufficiently larger than the number of materials (r). This is clearly satisfied by the Urban data since for $n = 162$ and $r = 6$, the probability to satisfy the sparsity pattern assumption is larger than $1 - 10^{-12}$. However, it was also explained why this condition is not necessary, see Section 7.1. What happens then if we reduce the number of bands? For example, suppose we keep only 9 bands from the 162 original clean bands of the urban data[16]. Figure 7.12 displays the basis elements for ℓ_2-NMU. Surprisingly, the algorithm is still able to separate the materials: trees are recovered in basis element 2, roofs 3, road 4 (mixed with dirt), grass 5, metal 7, and dirt 8.

When less than 6 bands are kept, the algorithm cannot detect all the materials individually. Figure 7.13 displays basis elements using 5 bands. The grass and the trees are extracted together (basis element 2), because their spectral signatures are similar; the roads and the dirt also are extracted together (basis element 4), so are the road and roofs (basis element 7), while the roofs and the metal (basis elements 3 and 6 respectively) are extracted individually.

[16]We selected the bands so that they are well distributed in the spectral domain. However, one could use more sophisticated techniques (e.g., subset selection algorithms such as [100]) or using dimensionality reduction techniques preserving nonnegativity (such as NMF) as a pre-processing step.

Finally, it seems that, as long as the spectral signatures of the different materials can be distinguished, the number of spectral bands does not need to be significantly larger than the number of materials in order for NMU to be able to perform classification. However, when this is not the case (e.g., when more noise and blur are present), more spectral bands are needed to distinguish the different materials, which seems to be a natural requirement (these observations will also be illustrated in the next section).

7.4.3 Simulated Hubble Space Telescope Data

Figure 7.14 displays some sample images of the simulated Hubble database which consists of 100 spectral images of the Hubble telescope, 128×128 pixels each, with added blur and noise[17] [126]. It is composed of 8 materials[18], see Figure 7.15.

Figure 7.16 shows the basis elements obtained with NMU, Figure 7.17 displays the classification obtained with k-means (which is able to extract five constitutive materials), Table 7.2 gives the classification of the basis elements and Figure 7.18 shows the end-members extraction: original vs. noisy and blurred.

Clusters	Al	S. Cell	Glue	Cu	H. Side	H. Top	Edge	Bolts
ℓ_2 basis #	2	3	4	6	7	8	13	11
ℓ_1 basis #	2	3	4	7	6	9	11	8

Table 7.2: Basis elements obtained: cluster selection for the Hubble telescope database with noise and blur, see Figure 7.16.

Spectral signatures of black rubber edge and bolts are not recovered very accurately (or not at all in the case of the ℓ_2-norm). The reason is that they are the smallest and thinest parts: they get mixed with surrounding materials which make them difficult to extract. Moreover, the spectral signature of the bolts is very similar to the one of copper stripping and therefore, when noise and blur are added, they are extracted together (basis elements 11 for ℓ_2-norm and 8 for ℓ_1-norm).

As for the Urban dataset, ℓ_2-NMU extracts more mixed materials and therefore needs more recursions to get all the parts separated than the ℓ_1-NMU, which seems to do a better job (especially for the black rubber edge).

[17] Point spread function on 5 by 5 pixels and with standard deviation of 1, and white Gaussian noise $\sigma = 1\%$ of the values of M and Poisson noise $\sigma = 1\%$ of the mean value of M.

[18] These are true Hubble satellite material spectral signatures provided to us by the NASA Johnson Space Center.

Number of Spectral Bands

Let reduce the number of spectral bands to 12, and compare the basis elements obtained with ℓ_2-NMU on the clean vs. the noisy and blurry images. Figure 7.19 displays the basis elements. Clearly, in the noisy and blurry case, the algorithm is no longer able to extract all the materials. The reason is that there is not enough spectral bands left. Because of blur (spectral signatures of materials are mixed together) and noise (spectral signatures are perturbed), 12 bands is not enough to distinguish all the materials, as already explained in Section 7.4.2. Quite naturally, the more the number of bands, the more robust the algorithm will be with respect to noise and blur.

7.4.4 Visual Experiments

In this section, we first provide some visual results[19] for datasets analyzed in a recent comparative study of dimensionality reduction techniques for hyperspectral images [24], and for which 'ground truth' data is not available. However, it allows to experimentally reinforce our claims about the properties of NMU; namely that it is indeed able to detect materials, or at least to separate some of them.

We then display results for an hyperspectral image of an eye with 4 spectral bands, and propose a way to post-process the basis elements obtained with NMU in order to achieve clustering.

Aerial Image of a Desert Region

This is a HYDICE terrain data set with 166 clean bands (originally 210), each containing 500×307 pixels. Figure 7.20 displays the first 6 basis elements. NMU is easily able to extract trees (basis element 2), roads (basis element 3) and grass (basis element 6).

San Diego Airport

The San Diego airport hyperspectral image contains 158 clean bands, and 400×400 pixels for each spectral image. Figure 7.21 displays the first 8 basis elements obtained by the NMU decomposition. In this case, it is less clear what the different materials are. It should probably be necessary to apply more sophisticated post-processing techniques to classify the pixels. However, we observe that

- ◇ Basis elements 4, 7 and 8 contain the roofs;
- ◇ Basis element 6 mostly contains roads (including the parking lots);

[19] We will only display basis elements obtained with ℓ_2-NMU because the results obtained with ℓ_1-NMU are comparable.

- ◇ Basis element 2 contains the grass and some roofs;
- ◇ Basis element 3 is mostly composed of another type of road surface (including boarding and landing zones).

Eye Image

We are given (only) 4 spectral images with 1040×1392 pixels (spectral bands correspond to IR, red, green and blue channels)[20]. The data comes from the West Virginia University multispectral image iris database [132]. Figure 7.22 displays the basis elements obtained with ℓ_2-NMU. The first basis element represents the pupil, a part of the iris and the eyelashes; the second some kind of substructure in the iris and the skin; and the third the pupil.

Segmentation and clustering multispectral eye images are useful in the analysis of iris recognition algorithms in biometrics, see, e.g., [19]. A possible way to post-process the NMU basis elements in order to achieve clustering is to compare their support. Recall that each basis element should represent a set of 'materials'. Therefore, if we want to identify these materials, the following simple procedure can be used

1. Compare the supports of each pair of basis elements[21], i.e., compute $\text{supp}(u_i) \cap \text{supp}(u_j)\, \forall i \neq j$.
2. Define non-empty intersections as new basis elements. If no new basis elements are identified, go to step 3.; otherwise go back to step 1.
3. $\forall i \neq j$ such that $\text{supp}(u_i) \subset \text{supp}(u_j)$, set $\text{supp}(u_j) \leftarrow \text{supp}(u_j) \setminus \text{supp}(u_i)$. If some basis elements have been modified, go back to step 1., otherwise go to step 4.
4. Materials are identified as the remaining basis elements, which define disjoint clusters.

One can check that this procedure will always terminate within a finite number of steps. Notice that, in practice, some kind of thresholding should be used in order to define the supports, and the intersections containing a small number of pixels should be considered as empty.

In the example of Figure 7.22, we have (after thresholding)

$$\text{supp}(u_1) \cap \text{supp}(u_2) \approx \emptyset, \quad \text{supp}(u_3) \subset \text{supp}(u_1) \quad \text{and} \quad \text{supp}(u_2) \cap \text{supp}(u_3) = \emptyset,$$

so that the above procedure provides us with the clustering displayed in Figure 7.23.

[20] The data comes from the West Virginia University multispectral image iris database. The circle around the pupil and the matrix inside the pupil were embedded in the image.

[21] There are $\binom{r}{2} = \frac{r(r-1)}{2} = \mathcal{O}(r^2)$ such pairs (at the first step), and each comparison requires $\mathcal{O}(mn)$ operations.

Conclusion

In this chapter, we have used recursive NMU for dimensionality reduction in the context of hyperspectral data analysis. We gave theoretical and experimental evidence showing that NMU is able to perform automatically soft-clustering of hyperspectral images. A main advantage is that no sparsity parameters have to be tuned and parts-based representation is naturally achieved. It would be interesting to compare NMU with other dimensionality reduction techniques such as PCA, NMF, ICA, etc., see [24]. Another direction of research would be to design automatic classification algorithms, based on the properties of NMU, to classify the pixels; as we proposed in Section 7.4.4. It would be particularly interesting to study these properties in more depth and see if it is possible to obtain stronger theoretical guarantees for the factors generated by NMU. Finally, NMU can be generalized to higher order tensors, which could be called *nonnegative tensor underapproximation* (NTU). For example, for a third order tensor $\mathcal{T}$ of dimension $m \times n \times p$, we would use a third vector $w \in \mathbb{R}^p_+$ in order to underapproximate $\mathcal{T}$ with a rank-one tensor $u \otimes v \otimes w$, i.e., $\mathcal{T}_{ijk} \approx u_i v_j w_k$. The optimal solutions for u, v and w separately can still be written in closed-forms and Algorithm 12 can be easily generalized. Notice that the hyperspectral images analyzed in this section can also be represented as third order tensors (two spatial dimensions and the wavelength, see Figure 7.1). However, abundance maps typically don't have a rank-one structure because constitutive elements can be spread anyhow over the scene being imaged. Hence they cannot be well represented by a rank-one matrix uv^T, and using NTU does not give good results. Nevertheless, it could be useful in other situations. For example, if we also had images taken at different time steps. This situation can be encountered for the analysis of a chemical reaction where the concentrations of the different constitutive elements evolve with time.

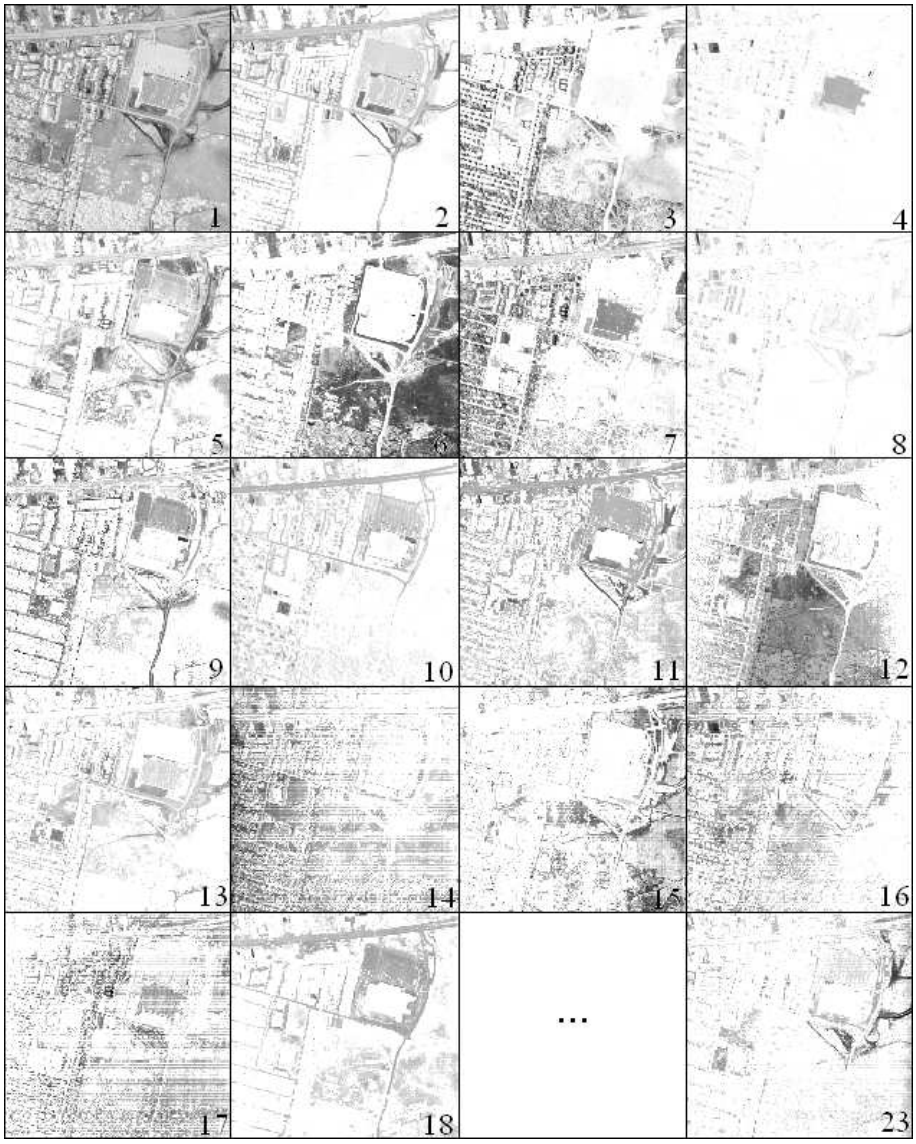

Figure 7.7: Basis elements (columns of matrix U) for the Urban dataset extracted with ℓ_2-NMU; dark tones indicate a high value of an entry (0 is white, 1 black); numbers indicate the position of the factor in the NMU decomposition.

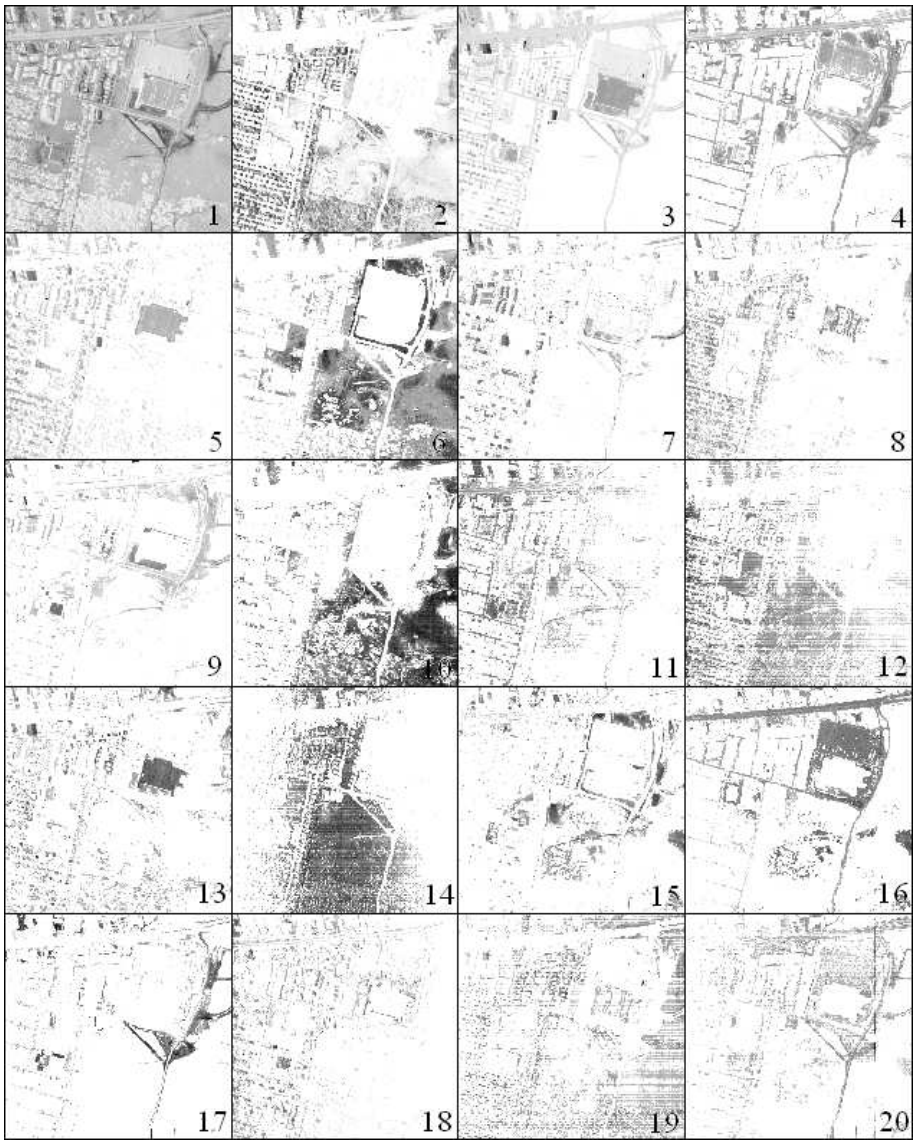

Figure 7.8: Basis elements of ℓ_1-NMU for the Urban dataset extracted, identified as in Figure 7.7.

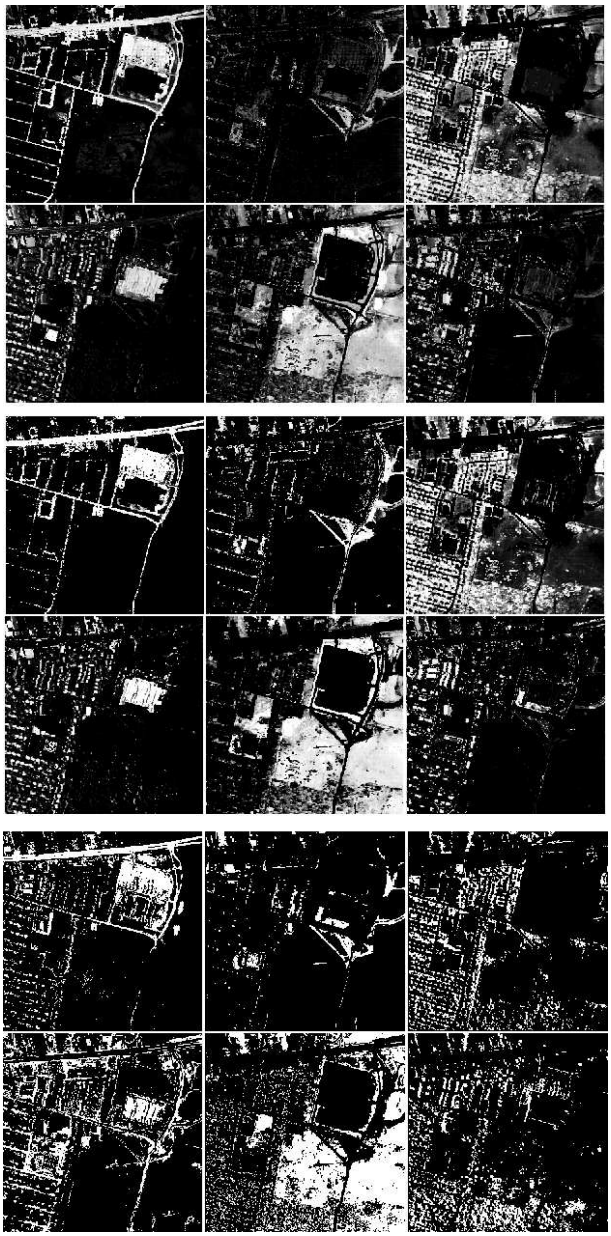

Figure 7.9: Post-processed basis elements of NMU for Urban dataset with ℓ_2-norm (top) and ℓ_1-norm (middle), and classification obtained with k-means able to extract properly the grass, the trees and the dirt (bottom). Light tones represent high degree of membership.

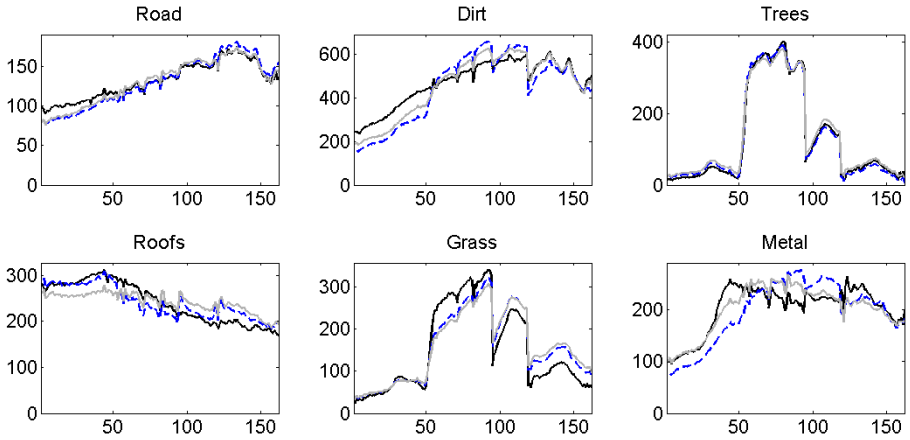

Figure 7.10: End-members extraction: ℓ_2-NMU (gray solid) and ℓ_1-NMU (dashed) vs. 6 end-members from the image using N-FINDR5 [151] plus manual adjustment (dark solid) from [85]. The x-axis gives the wavelength bands while y-axis gives the reflectance values (intensities of reflected light).

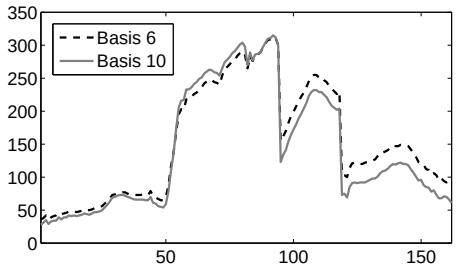

Figure 7.11: Spectral signatures of grass: weighted average of the spectral signatures of the pixels present in basis elements 6 and 10 of the ℓ_1-norm solution.

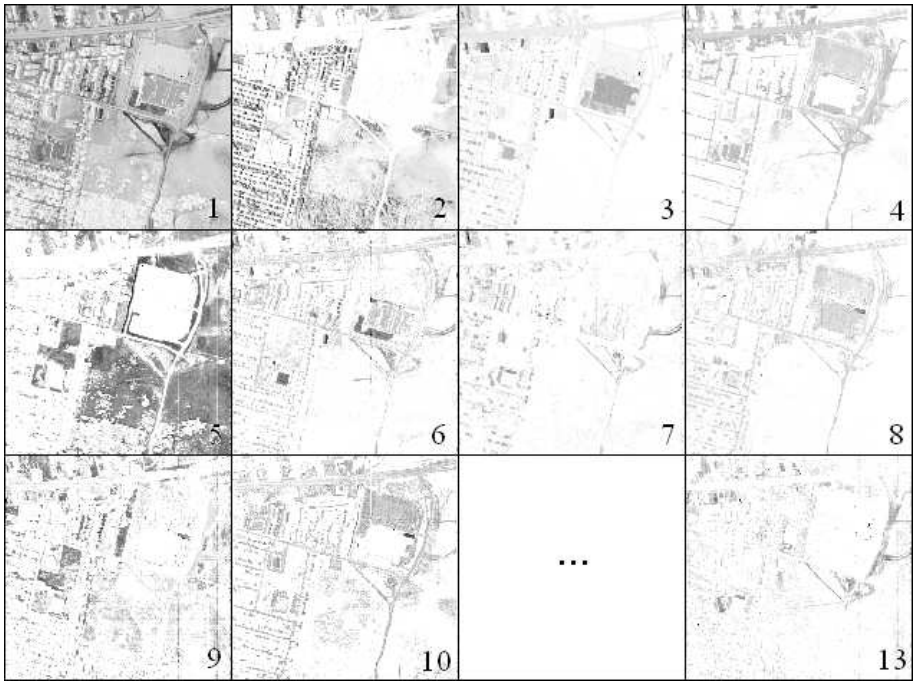

Figure 7.12: Basis elements of ℓ_2-NMU for the Urban dataset using only 9 bands (bands 1, 20, 40, ..., 160).

Figure 7.13: Basis elements of ℓ_2-NMU for the Urban dataset using only 5 bands (1,41,81,121,161).

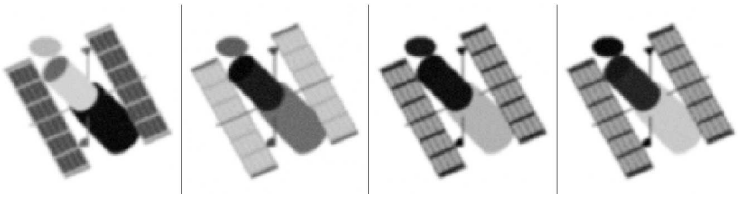

Figure 7.14: Sample of images in the Hubble tensor with blur and noise.

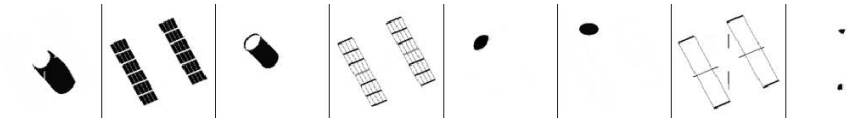

Figure 7.15: The 8 materials for the Hubble telescope data provided to us by NASA. From left to right: Aluminum, Solar Cell, Green Glue, Copper Stripping, Honeycomb Side, Honeycomb Top, Black Rubber Edge and Bolts.

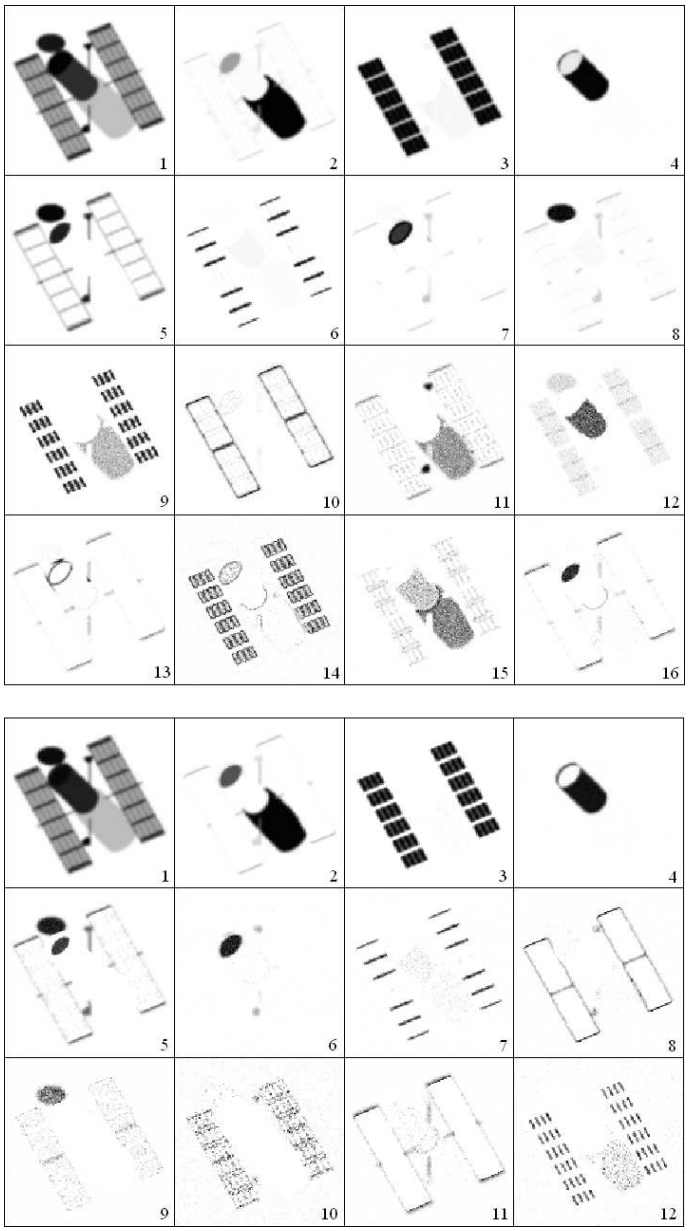

Figure 7.16: Basis elements of NMU for Hubble telescope with ℓ_2-norm (top) and ℓ_1-norm (bottom) with added blur and noise.

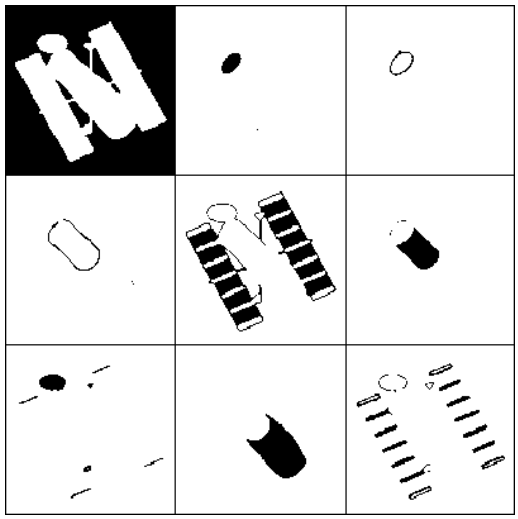

Figure 7.17: K-means classification for Hubble telescope with added blur and noise, not able to separate the Honeycomb Top, Black Rubber Edge and Bolts.

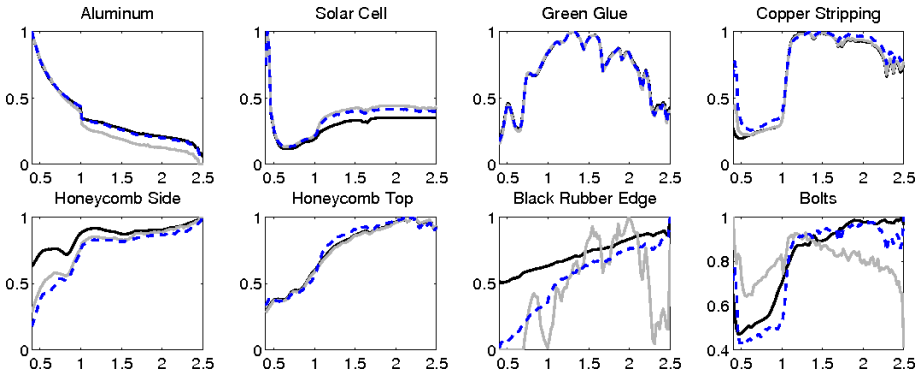

Figure 7.18: Endmembers for the Hubble satellite data with noise and blur. NMU with ℓ_2 (gray solid) and NMU with ℓ_1 (dashed) vs. 8 true end-members (black solid).

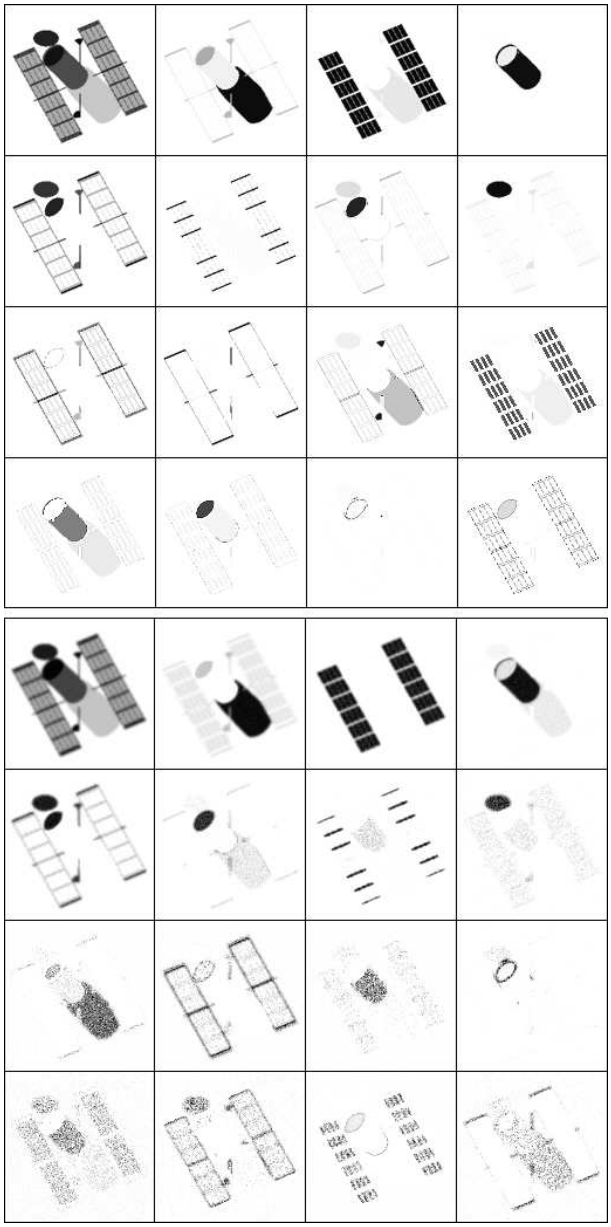

Figure 7.19: Basis elements obtained with ℓ_2-NMU on the Hubble telescope hyperspectral image using only 12 spectral bands ($1 + 9i$, $0 \leq i \leq 11$): (top) clean image and (bottom) noisy and blurry image (same settings as before).

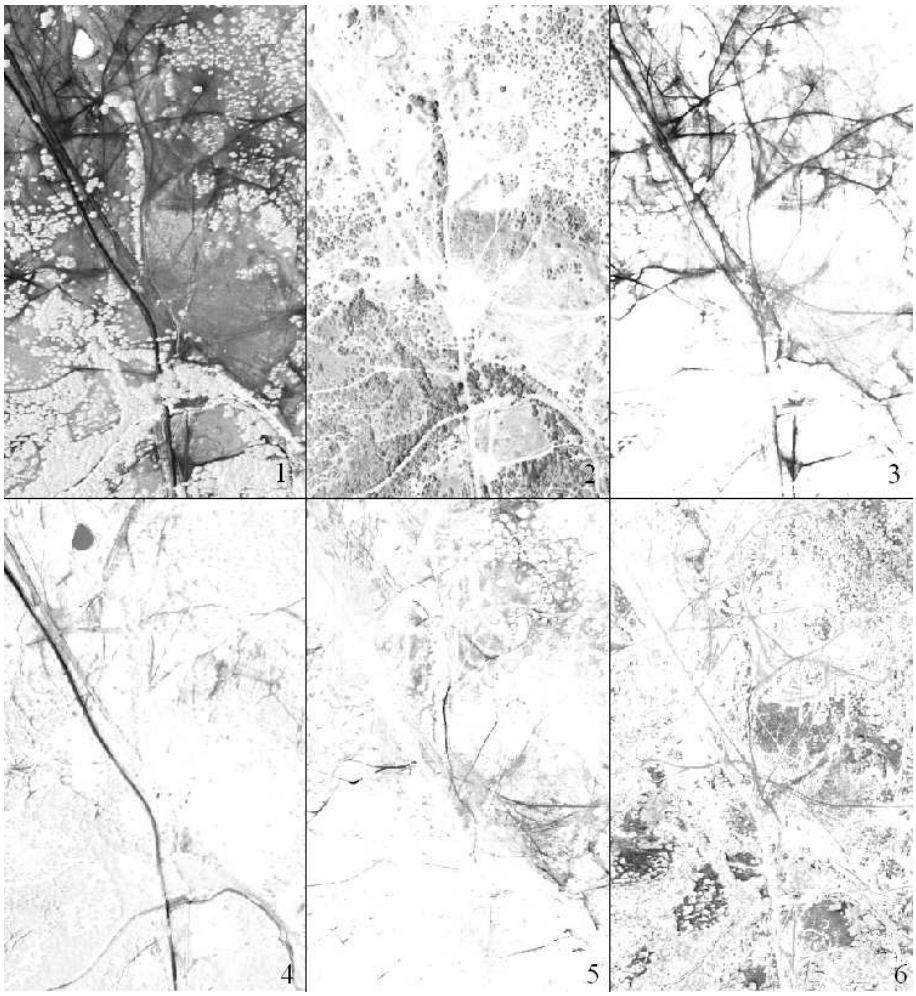

Figure 7.20: Aerial Image of a Desert Region basis elements obtained with ℓ_2-NMU.

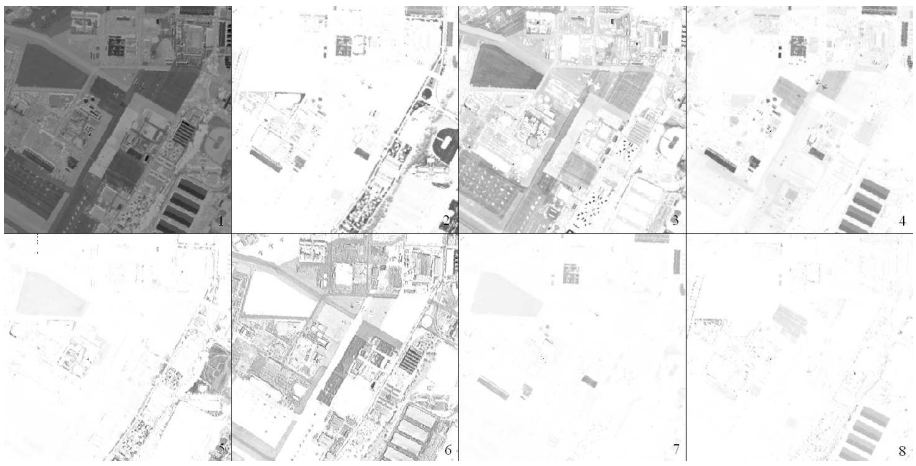

Figure 7.21: San Diego Airport basis elements obtained with ℓ_2-NMU.

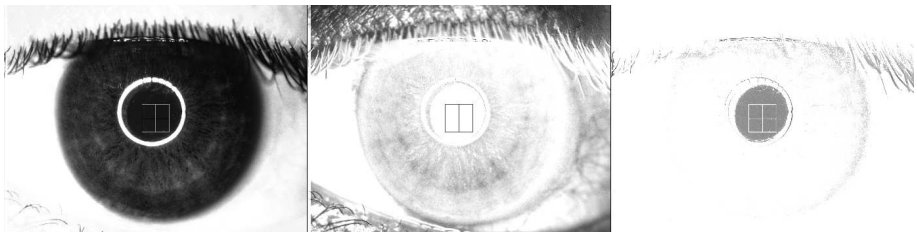

Figure 7.22: Eye basis elements obtained with ℓ_2-NMU.

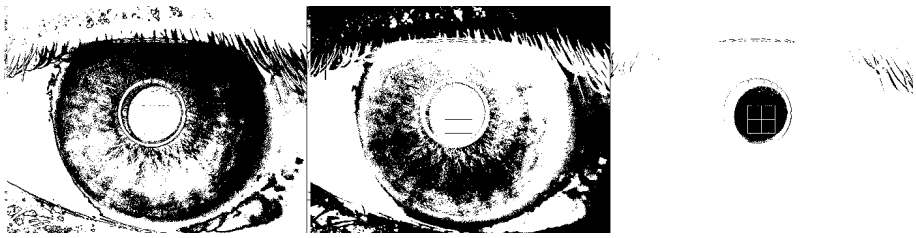

Figure 7.23: Clustering of the eye, based on the NMU decomposition.

Chapter 8

Weights and Missing Data

Approximating a matrix with one of lower rank is a key problem in data analysis and is widely used for linear dimensionality reduction, see the introduction of the thesis (Chapter 1). In some cases, it might be necessary to attach a weight to each entry of the data matrix corresponding to its relative importance [64]. This is for example the case in the following situations:

- ◇ The matrix to be approximated is obtained via a sampling procedure and the number of samples and/or the expected variance vary among the entries, e.g., 2-D digital filter design [116], or microarray data analysis [117].
- ◇ Some data is missing/unknown, which can be taken into account assigning zero weights to the missing/unknown entries of the data matrix. This is for example the case in collaborative filtering, notably used to design recommender systems [135] (in particular, the Netflix prize competition has demonstrated the effectiveness of low-rank matrix factorization techniques [101]), or in computer vision to recover structure from motion [93, 139], see also [28]. This problem is often referred to as *PCA with missing data* (PCAMD) [83, 139], and can be viewed as a *low-rank matrix completion problem with noise*, i.e., approximate a given noisy data matrix featuring missing entries with a low-rank matrix[1].
- ◇ A greater emphasis must be placed on the accuracy of the approximation on a localized part of the data, a situation encountered for example in image processing [89, Chapter 6].

Finding a low-rank matrix that is the closest to the input matrix according to these weights is an optimization problem called *weighted low-rank approx-*

[1] In our settings, the rank of the approximation is fixed a priori.

imation (WLRA). Formally, it can be formulated as follows: first, given an m-by-n nonnegative weight matrix $W \in \mathbb{R}_+^{m\times n}$, we define the weighted Frobenius norm of an m-by-n matrix A as $||A||_W = (\sum_{i,j} W_{ij}A_{ij}^2)^{\frac{1}{2}}$. Then, given an m-by-n real matrix $M \in \mathbb{R}^{m\times n}$ and a positive integer $r \leq \min(m,n)$, we seek an m-by-n matrix X with rank at most r that approximates M as closely as possible, where the quality of the approximation is measured by the weighted Frobenius norm of the error:

$$p^* = \inf_{X\in\mathbb{R}^{m\times n}} ||M-X||_W^2 \text{ such that } X \text{ has rank at most } r.$$

Since any m-by-n matrix with rank at most r can be expressed as the product of two matrices of dimensions m-by-r and r-by-n, we will use the following more convenient formulation featuring two unknown matrices $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{r\times n}$ but no explicit rank constraint:

$$p^* = \inf_{U\in\mathbb{R}^{m\times r}, V\in\mathbb{R}^{r\times n}} \quad ||M-UV||_W^2 = \sum_{ij} W_{ij}(M-UV)_{ij}^2 \,. \qquad \text{(WLRA)}$$

Even though (WLRA) is suspected to be $\mathcal{NP}$-hard [93,140], this has never, to the best of our knowledge, been studied formally.

In this chapter, we focus only on the computational complexity of this problem in the rank-one case[2] (i.e., for $r = 1$) and prove the following two results.

Theorem 8.1. *When $M \in \{0,1\}^{m\times n}$, and $W \in]0,1]^{m\times n}$, it is $\mathcal{NP}$-hard to find an approximate solution of rank-one* (WLRA) *with objective function accuracy less than $2^{-11}(mn)^{-6}$.*

Theorem 8.2. *When $M \in [0,1]^{m\times n}$, and $W \in \{0,1\}^{m\times n}$, it is $\mathcal{NP}$-hard to find an approximate solution of rank-one* (WLRA) *with objective function accuracy less than $2^{-12}(mn)^{-7}$.*

It is then $\mathcal{NP}$-hard to find an approximate solution to the following problems: (1) rank-one (WLRA) with positive weights, and (2) rank-one approximation of a matrix with missing data.

Even though this chapter does not directly deal with nonnegativity constraints, the matrices used in the reductions are nonnegative (see Theorems 8.1 and 8.2). Moreover, when the data matrix M is nonnegative, it is easy to show that, without loss of generality, any optimal solution to rank-one (WLRA) can be assumed to be nonnegative (cf. derivations of Example 8.1). Therefore, all the complexity results of this chapter also apply to NMF; in particular, it

[2]The obtained results can be easily generalized to any fixed rank r, see Remark 8.2.

is $\mathcal{NP}$-hard to find an approximate solution to *rank-one weighted NMF*[3] and to *rank-one NMF with missing data*, in contrast with rank-one NMF which is polynomially solvable (cf. Chapter 3).

Remark 8.1. The following more general problem is sometimes also referred to as WLRA:

$$\inf_{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{r \times n}} ||M - UV||_P^2,$$

where $||A||_P^2 = \text{vec}(A)^T P \text{vec}(A)$, with $\text{vec}(A)$ a vectorization of matrix A and P an mn-by-mn positive semidefinite matrix, see [136] and the references therein; our WLRA formulation hence reduces to having P diagonal and nonnegative.

The chapter is organized as follows. We first review existing results about the complexity of (WLRA) in Section 8.1. In Section 8.2, we prove Theorem 8.1 and, in Section 8.3, Theorem 8.2 using polynomial-time reductions from the maximum-edge biclique problem (see Problem (MB) in Section 5.1.2). We conclude with a discussion and some open questions.

8.1 Previous Results

Weighted low-rank approximation is known to be much more difficult than the corresponding unweighted problem (i.e., when W is the matrix of all ones), which is solved using the SVD, see Section 2.2. In fact, it has been previously observed that the weighted problem might have several local minima which are not global [140].

Example 8.1. Let

$$M = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}, \quad \text{and} \quad W = \begin{pmatrix} 1 & 100 & 2 \\ 100 & 1 & 2 \\ 1 & 1 & 1 \end{pmatrix}.$$

In the case of a rank-one factorization ($r = 1$) and a nonnegative matrix M, one can impose without loss of generality that $U \geq 0$ and $V \geq 0$. In fact, one can easily check that any solution UV is improved by taking its component-wise absolute value $|UV| = |U||V|$. Moreover, we can impose without loss of generality that $||U||_2 = 1$, so that only two degrees of freedom remain. Indeed, for a given

$$U(x, y) = \begin{pmatrix} x \\ y \\ \sqrt{1 - x^2 - y^2} \end{pmatrix}, \quad \text{with} \quad \begin{cases} x \geq 0, y \geq 0 \\ x^2 + y^2 \leq 1 \end{cases},$$

[3]Weighted NMF amounts to solving (WLRA) with additional nonnegativity constraints on $U \geq 0$ and $V \geq 0$, see, e.g., [89, Chapter 6].

the corresponding optimal $V^*(x,y) = \text{argmin}_V \, ||M - U(x,y)V||_W^2$ can be computed easily (it reduces to a weighted least squares problem). Figure 8.1 displays the surface of the objective function $||M - U(x,y)V^*(x,y)||_W$ with respect to parameters x and y; we distinguish 4 local minima, close to $(\frac{1}{\sqrt{2}}, 0)$, $(0, \frac{1}{\sqrt{2}})$, $(0,0)$ and $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. We will see later in Section 8.2 how this example has been

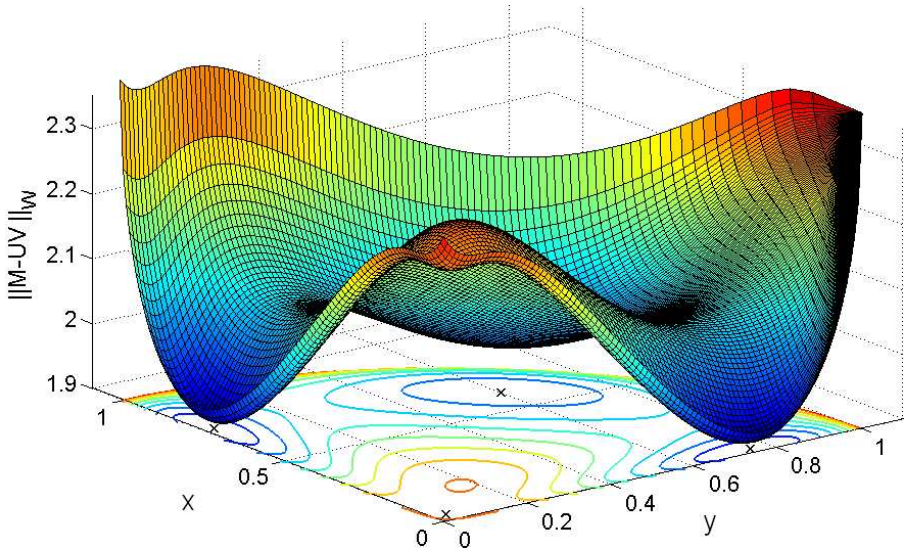

Figure 8.1: Objective function of (WLRA) with respect to the parameters (x, y).

generated.

However, if the rank of the weight matrix $W \in \mathbb{R}_+^{m \times n}$ is equal to one, i.e., $W = st^T$ for some $s \in \mathbb{R}_+^m$ and $t \in \mathbb{R}_+^n$, (WLRA) can be reduced to an unweighted low-rank approximation. In fact,

$$\begin{aligned} ||M - UV||_W^2 &= \sum_{i,j} W_{ij} \, (M - UV)_{ij}^2 = \sum_{i,j} s_i t_j \, (M - UV)_{ij}^2 \\ &= \sum_{i,j} \left(\sqrt{s_i t_j} \, M_{ij} - (\sqrt{s_i} \, U_{i:})(\sqrt{t_j} \, V_{:j}) \right)^2. \end{aligned}$$

Therefore, if we define a matrix M' such that $M'_{ij} = \sqrt{s_i t_j} \, M_{ij} \; \forall i, j$, an optimal weighted low-rank approximation (U, V) of M can be recovered from a solution (U', V') to the unweighted problem for matrix M' using $U_{i:} = U'_{i:}/\sqrt{s_i} \; \forall i$ and

$V_{:j} = V'_{:j}/\sqrt{t_j}\ \forall j$.

When the weight matrix W is binary, WLRA amounts to approximating a matrix with missing data. This problem is closely related to *low-rank matrix completion* (MC), see [25] and references therein, which can be defined as

$$\min_X \text{rank}(X) \text{ such that } X_{ij} = M_{ij} \text{ for } (i,j) \in \Omega \subset \{1,2,\ldots,m\}\times\{1,2,\ldots,n\}, \tag{MC}$$

where Ω is the set of entries for which the values of M are known. (MC) has been shown to be $\mathcal{NP}$-hard [29], and it is clear that an optimal solution X^* of (MC) can be obtained by solving a sequence of (WLRA) problems with the same matrix M, with

$$W_{ij} = \begin{cases} 1 & \text{if } (i,j) \in \Omega \\ 0 & \text{otherwise} \end{cases},$$

and for different values of the target rank ranging from $r = 1$ to $r = \min(m,n)$. The smallest value of r for which the objective function $||M - UV||^2_W$ of (WLRA) vanishes provides an optimal solution for (MC). This observation implies that it is $\mathcal{NP}$-hard to solve (WLRA) for each possible value of r (from 1 to $\min(m,n)$) since it would solve (MC). However, this does not imply that (WLRA) is $\mathcal{NP}$-hard when r is fixed, and in particular when $r = 1$. In fact, checking whether (MC) admits a rank-one solution can be done easily[4].

Rank-one (WLRA) can be equivalently reformulated as

$$\inf_A ||M - A||^2_W \quad \text{such that} \quad \text{rank}(A) \leq 1,$$

and, when W is binary, it is then the problem of finding, if possible, the best rank-one approximation of a matrix with missing entries. To the best of our knowledge, the complexity of this problem has never been studied formally; it will be shown to be $\mathcal{NP}$-hard in the Section 8.3.

Another closely related result is the $\mathcal{NP}$-hardness of the structure from motion problem (SFM), in the presence of noise and missing data [122]. Several points of a rigid object are tracked with cameras (we are given the projections of the 3-D points on the 2-D camera planes; missing data arises because the points might not always be visible by the camera, e.g., in case of rotation), and the aim is to recover the structure of the object and the positions of the 3-D points. SFM can be written as a rank-four (WLRA) problem with a binary weight matrix[5] [93]. However, this result does not imply anything on the complexity analysis of rank-one (WLRA).

[4]The solution $X = uv^T$ can be constructed observing that the vector u must be multiple of each column of M.

[5]Except that the last row of V must be all ones, i.e., $V_{r:} = \mathbf{1}_{1\times n}$.

An important feature of (WLRA) is exposed by the following example.

Example 8.2. Let

$$M = \begin{pmatrix} 1 & ? \\ 0 & 1 \end{pmatrix}$$

where ? indicates that an entry is missing, i.e., that the weight associated with this entry is 0 (1 otherwise). Observe that $\forall (u,v) \in \mathbb{R}^m \times \mathbb{R}^n$,

$$\text{rank}(M) = 2 \text{ and } \text{rank}(uv^T) = 1 \quad \Rightarrow \quad ||M - uv^T||_W > 0.$$

However, we have

$$\inf_{(u,v)\in\mathbb{R}^m\times\mathbb{R}^n} ||M - uv^T||_W = 0.$$

In fact, one can check that with

$$u^{(k)} = \begin{pmatrix} 1 \\ 10^{-k} \end{pmatrix} \text{ and } v^{(k)} = \begin{pmatrix} 1 \\ 10^{k} \end{pmatrix}, \text{ we have } \lim_{k\to+\infty} ||M - u^{(k)}{v^{(k)}}^T||_W = 0.$$

This indicates that when W has zero entries the set of optimal solutions of (WLRA) might be empty: there might not exist an optimal solution. In other words, the (bounded) infimum might not be attained. At the other end, the infimum is always attained for $W > 0$ since $||.||_W$ is then a norm.

For this reason, these two cases will be analyzed separately: in Section 8.2, we study the computational complexity of the problem when $W > 0$, and, in Section 8.3, when W is binary (the problem with missing data).

8.2 Weighted Low-Rank Approximation

In order to prove $\mathcal{NP}$-hardness of rank-one (WLRA) with positive weights ($W > 0$), let us consider the following instance:

$$p^* = \min_{u\in\mathbb{R}^m, v\in\mathbb{R}^n} ||M - uv^T||_W^2, \tag{W-1d}$$

with $M \in \{0,1\}^{m\times n}$ the biadjacency of a bipartite graph $G_b = (V, E)$ and the weight matrix defined as

$$W_{ij} = \begin{cases} 1 & \text{if } M_{ij} = 1 \\ d & \text{if } M_{ij} = 0 \end{cases}, \ 1 \le i \le m, 1 \le j \le n,$$

with $d \ge 1$ a parameter. We are going to show that as d increases the optimal solutions of (W-1d) gets closer to optimal solutions of the maximum-edge biclique problem (MB) (cf. Section 5.1.2).

Intuitively, increasing the value of d makes the zero entries of M more important in the objective function, which leads them to be approximated by

small values. This observation will be used to show that, for d sufficiently large, the optimal value p^* of (W-1d) will be close to the minimum $|E|-|E^*|$ of (MB) (Lemma 8.5). Recall $|E| = ||M||_F^2$ denotes the cardinality of the edge set of G_b and $|E^*|$ is the cardinality of a maximum-edge biclique of G_b.

In fact, as the value of parameter d increases, the local minima of (W-1d) get closer to the 'locally' optimal solutions of (MB), which are binary vectors describing the maximal bicliques in G_b, i.e., bicliques not contained in larger bicliques. Example 8.1 illustrates the situation: the graph G_b corresponding to matrix M is represented in Figure 8.2 and contains four maximal bicliques

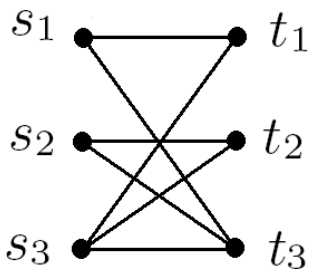

Figure 8.2: Graph corresponding to the matrix M of Example 8.1.

$\{s_1, s_3, t_1, t_3\}$, $\{s_2, s_3, t_2, t_3\}$, $\{s_3, t_1, t_2, t_3\}$ and $\{s_1, s_2, s_3, t_3\}$, and the weight matrix W that was used is similar to the case $d = 100$ in problem (W-1d). We now observe that (W-1d) has four local optimal solutions as well (cf. Figure 8.1) close to $(\frac{1}{\sqrt{2}}, 0)$, $(0, \frac{1}{\sqrt{2}})$, $(0, 0)$ and $(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. There is a one to one correspondence between these solutions and the four maximal bicliques listed above (in this order). For example, for $(x, y) = (\frac{1}{\sqrt{2}}, 0)$ we have $U(x, y) = (\frac{1}{\sqrt{2}}\, 0\, \frac{1}{\sqrt{2}})^T$, $V^*(x, y)$ is approximately equal to $(\sqrt{2}\, 0\, \sqrt{2})^T$, and this solution corresponds to the maximal biclique $\{s_1, s_3, t_1, t_3\}$.

This idea is similar to the one used in Chapter 5 to prove Theorem 5.3, i.e., to prove $\mathcal{NP}$-hardness of the rank-one nonnegative factorization problem $\min_{u \in \mathbb{R}^m_+, v \in \mathbb{R}^n_+} ||M - uv^T||_F$, where the zero entries of M were replaced by sufficiently large negative ones.

Let us now prove this formally. It is first observed that for any (u, v) such that $||M - uv^T||_W^2 \leq |E|$, the absolute value of the row or the column of uv^T corresponding to a zero entry of M must be smaller than a constant inversely proportional to $\sqrt[4]{d}$.

Lemma 8.3. *Let (i, j) be such that $M_{ij} = 0$, then $\forall (u, v)$ such that $||M -$*

$uv^T||_W^2 \leq |E|$,

$$\min\Big(\max_{1\leq k\leq n}|u_iv_k|, \max_{1\leq p\leq m}|u_pv_j|\Big) \leq \sqrt[4]{\frac{4|E|^2}{d}}.$$

Proof. Without loss of generality u and v can be scaled such that $||u||_2 = ||v||_2$ without changing the product uv^T. First, observe that since $||.||_W$ is a norm,

$$||uv^T||_W - \sqrt{|E|} = ||uv^T||_W - ||M||_W \leq ||M - uv^T||_W \leq \sqrt{|E|}.$$

Since all entries of W are larger than 1 ($d \geq 1$), we have

$$||u||_2||v||_2 = ||uv^T||_F \leq ||uv^T||_W \leq \sqrt{4|E|},$$

and then $||u||_2 = ||v||_2 \leq \sqrt[4]{4|E|}$.

Moreover $d(0 - u_iv_j)^2 \leq ||M - uv^T||_W^2 \leq |E|$, so that $|u_iv_j| \leq \sqrt{\frac{|E|}{d}}$ which implies that either $|u_i| \leq \sqrt[4]{\frac{|E|}{d}}$ or $|v_j| \leq \sqrt[4]{\frac{|E|}{d}}$. Combining above inequalities with the fact that $(\max_{1\leq k\leq n}|v_k|)$ and $(\max_{1\leq p\leq m}|u_p|)$ are bounded above by $||u||_2 = ||v||_2 \leq \sqrt[4]{4|E|}$ completes the proof. □

Using Lemma 8.3, we can associate any point (u, v) such that $||M-uv^T||_W^2 \leq |E|$ with a biclique of G_b, the graph generated by the biadjacency matrix M.

Corollary 8.4. *For any pair (u, v) such that $||M - uv^T||_W^2 \leq |E|$, the set*

$$\Omega(u, v) = I \times J, \text{ with } I = \{\, i \mid \exists j \text{ s.t. } |u_iv_j| > \alpha \,\}, J = \{\, j \mid \exists i \text{ s.t. } |u_iv_j| > \alpha \,\},$$

where $\alpha = \sqrt[4]{\frac{4|E|^2}{d}}$, defines a biclique of G_b.

We can now provide lower and upper bounds on the optimal value p^* of (W-1d), and show that it is not too different from the optimal value $|E| - |E^*|$ of (MB).

Lemma 8.5. *Let $0 < \epsilon \leq 1$. For any value of parameter d such that $d \geq \frac{2^6|E|^6}{\epsilon^4}$, the optimal value p^* of* (W-1d) *satisfies*

$$|E| - |E^*| - \epsilon < p^* \leq |E| - |E^*|.$$

Proof. Let (u, v) be an optimal solution of (W-1d) (there always exists at least one optimal solution, cf. Section 8.1), and let us note $p = |E| - |E^*| \geq 0$. Since any optimal solution of (MB) plugged in (W-1d) also achieves an objective function equal to p, we must have

$$p^* = ||M - uv^T||_W^2 \leq p = |E| - |E^*|,$$

which gives the upper bound.

By Corollary 8.4, the set $\Omega = \Omega(u, v)$ defines a biclique of (MB) with $|\Omega| \leq |E^*|$ edges. By construction, the entries in M which are not in Ω are approximated by values smaller than α. If $\alpha = \sqrt[4]{\frac{4|E|^2}{d}} \leq 1$, i.e., $d \geq 4|E|^2$ which is satisfied for $0 < \epsilon \leq 1$, the error corresponding to a one entry of M not in the biclique Ω is at least $(1-\alpha)^2$. Since there are at least $p = |E| - |E^*|$ such entries, we have

$$(1-\alpha)^2 p \leq ||M - uv^T||_W^2. \tag{8.1}$$

Moreover

$$(1-\alpha)^2 p > (1-2\alpha)p = p - 2\alpha p \geq p - 2\alpha|E| \geq p - \epsilon,$$

since $2\alpha|E| \leq \epsilon \iff d \geq \frac{2^6|E|^6}{\epsilon^4}$, which gives the lower bound. □

This result implies that for $\epsilon = 1$, i.e., for $d \geq (2|E|)^6$, we have $|E| - |E^*| - 1 < p^* \leq |E| - |E^*|$, and therefore computing p^* exactly would allow to recover $|E^*|$ (since $\lceil p^* \rceil = |E| - |E^*|$), which is $\mathcal{NP}$-hard. Since the reduction from (MB) to (W-1d) is polynomial (it uses the same matrix M and a weight matrix W whose description has polynomial length), we conclude that solving (W-1d) exactly is $\mathcal{NP}$-hard. The next result shows that even solving (W-1d) approximately is $\mathcal{NP}$-hard.

Corollary 8.6. *For any $d > (2mn)^6$, $M \in \{0,1\}^{m\times n}$, and $W \in \{1, d\}^{m\times n}$, it is $\mathcal{NP}$-hard to find an approximate solution of rank-one* (WLRA) *with objective function accuracy less than $1 - \frac{(2mn)^{3/2}}{d^{1/4}}$.*

Proof. Let $d > (2mn)^6$, $0 < \epsilon = \frac{(2mn)^{3/2}}{d^{1/4}} < 1$, and $(\bar{u}, \bar{v})$ be an approximate solution of (W-1d) with objective function accuracy $(1-\epsilon)$, i.e., $p^* \leq \bar{p} = ||M - \bar{u}\bar{v}^T||_W^2 \leq p^* + 1 - \epsilon$. Since $d = \frac{(2mn)^6}{\epsilon^4} \geq \frac{(2|E|)^6}{\epsilon^4}$, Lemma 8.5 applies and we have

$$|E| - |E^*| - \epsilon \;<\; p^* \leq\; \bar{p} \;\leq\; p^* + 1 - \epsilon \;\leq\; |E| - |E^*| + 1 - \epsilon.$$

We finally observe that $\bar{p}$ allows to recover $|E^*|$, which is $\mathcal{NP}$-hard. In fact, adding ϵ to the above inequalities gives $|E| - |E^*| < \bar{p} + \epsilon \leq |E| - |E^*| + 1$, and therefore

$$|E^*| = |E| - \left\lceil \bar{p} + \epsilon \right\rceil + 1.$$

□

We are now in position to prove Theorem 8.1, which deals with the hardness of rank-one (WLRA) with bounded weights.

Proof of Theorem 8.1. Let us use Corollary 8.6 with $W \in \{1, d\}^{m\times n}$, and define $W' = \frac{1}{d}W \in \{\frac{1}{d}, 1\}^{m\times n}$. Clearly, replacing W by W' in (W-1d) simply amounts to multiplying the objective function by $\frac{1}{d}$, with $||M - uv^T||^2_{W'} = \frac{1}{d}||M - uv^T||^2_W$. Taking $d^{1/4} = 2(2mn)^{3/2}$ in Corollary 8.6, we obtain that for $M \in \{0, 1\}^{m\times n}$ and $W \in]0, 1]^{m\times n}$, it is $\mathcal{NP}$-hard to find an approximate solution of rank-one (WLRA) with objective function accuracy less than $\frac{1}{d}\Big(1 - \frac{(2mn)^{3/2}}{d^{1/4}}\Big) = \frac{1}{2d} = 2^{-11}(mn)^{-6}$. □

Remark 8.2. Using the same construction as in Theorem 6.6 from Chapter 6, this rank-one $\mathcal{NP}$-hardness result can be generalized to any factorization rank, i.e., approximate (WLRA) for any fixed rank r is $\mathcal{NP}$-hard.

Remark 8.3. The bounds on d have been quite crudely estimated, and can be improved. Our goal was only to show existence of a polynomial-time reduction from (MB) to rank-one (WLRA).

8.3 Low-Rank Approximation with Missing Data

Unfortunately, the above $\mathcal{NP}$-hardness proof does not include the case when W is binary, corresponding to missing data in the matrix to be approximated (or to low-rank matrix completion with noise). This corresponds to the following problem

$$\inf_{U\in\mathbb{R}^{m\times r}, V\in\mathbb{R}^{r\times n}} \quad ||M - UV||^2_W = \sum_{ij} W_{ij}(M - UV)^2_{ij}\,, \quad W \in \{0, 1\}^{m\times n}. \tag{LRAMD}$$

In the same spirit as before, we consider the following rank-one version of the problem

$$p^* = \inf_{u\in\mathbb{R}^m, v\in\mathbb{R}^n} ||M - uv^T||^2_W, \tag{MD-1d}$$

with input data matrices M and W defined as follows

$$M = \left(\begin{array}{c|c} M_b & \mathbf{0}_{s\times Z} \\ \hline \mathbf{0}_{Z\times t} & dI_Z \end{array}\right) \text{ and } W = \left(\begin{array}{c|c} \mathbf{1}_{s\times t} & B_1 \\ \hline B_2 & I_Z \end{array}\right),$$

where $M_b \in \{0, 1\}^{s\times t}$ is the biadjacency matrix of the bipartite graph $G_b = (V, E)$, $d > 1$ is a parameter, $Z = st - |E|$ is the number of zero entries in M_b, $m = s + Z$ and $n = t + Z$ are the dimensions of M and W.

Binary matrices $B_1 \in \{0, 1\}^{s\times Z}$ and $B_2 \in \{0, 1\}^{Z\times t}$ are constructed as follows: assume the Z zero entries of M_b can be enumerated as

$$\{M_b(i_1, j_1), M_b(i_2, j_2), \ldots, M_b(i_Z, j_Z)\},$$

and let k_{ij} be the (unique) index k ($1 \leq k \leq Z$) such that $(i_k, j_k) = (i, j)$ (therefore k_{ij} is only defined for pairs (i, j) such that $M_b(i, j) = 0$, and establishes a bijection between these pairs and the set $\{1, 2, \ldots, Z\}$). We now define matrices B_1 and as follows: for every index $1 \leq k_{ij} \leq Z$, we have

$$B_1(i, k_{ij}) = 1, B_1(i', k_{ij}) = 0 \; \forall i' \neq i \text{ and } B_2(k_{ij}, j) = 1, B_2(k_{ij}, j') = 0 \; \forall j' \neq j \, .$$

Equivalently, each column of B_1 (resp. row of B_2) corresponds to a different zero entry $M_b(i, j) = 0$, and contains only zeros except for a one in position i within the column (resp j within the row).

In the case of Example 8.1, we get

$$M = \left(\begin{array}{ccc|c} 1 & 0 & 1 & \\ 0 & 1 & 1 & \mathbf{0}_{3\times 2} \\ 1 & 1 & 1 & \\ \hline \multicolumn{3}{c|}{\mathbf{0}_{2\times 3}} & d\, I_2 \end{array}\right) \text{ and } W = \left(\begin{array}{ccc|cc} & & & 1 & 0 \\ & \mathbf{1}_{3\times 3} & & 0 & 1 \\ & & & 0 & 0 \\ \hline 0 & 1 & 0 & \multicolumn{2}{c}{\multirow{2}{*}{I_2}} \\ 1 & 0 & 0 & & \end{array}\right),$$

i.e., the matrix to be approximated can be represented as

$$\left(\begin{array}{ccc|cc} 1 & 0 & 1 & 0 & ? \\ 0 & 1 & 1 & ? & 0 \\ 1 & 1 & 1 & ? & ? \\ \hline ? & 0 & ? & d & ? \\ 0 & ? & ? & ? & d \end{array}\right).$$

For any feasible solution (u, v) of (MD-1d), we also note

$$u = \begin{pmatrix} u_b \\ u_d \end{pmatrix} \in \mathbb{R}^m, \; u_b \in \mathbb{R}^s \text{ and } u_d \in \mathbb{R}^Z,$$

$$v = \begin{pmatrix} v_b \\ v_d \end{pmatrix} \in \mathbb{R}^n, \; v_b \in \mathbb{R}^t \text{ and } v_d \in \mathbb{R}^Z.$$

We will show that this formulation ensures that, as d increases, the zero entries of the matrix M_b (upper left of matrix M, which is the biadjacency matrix of G_b) have to be approximated with smaller values. Hence, as for (W-1d), we will be able to prove that the optimal value p^* of (MD-1d) will have to get close to the minimum $|E| - |E^*|$ of (MB), implying its $\mathcal{NP}$-hardness.

Intuitively, when d is large, the lower right matrix dI_Z of M will have to be approximated by a matrix with large diagonal entries since they correspond to one entries in the weight matrix W. Hence $u_d(k_{ij})v_d(k_{ij})$ has to be large for all $1 \leq k_{ij} \leq Z$. We then have at least either $u_d(k_{ij})$ or $v_d(k_{ij})$ large for all k_{ij} (recall each k_{ij} corresponds to a zero entry in M at position (i, j), cf. definition of B_1 and B_2 above). By construction, we also have two entries $M(s + k_{ij}, j) = 0$

and $M(i, t+k_{ij}) = 0$ with nonzero weights corresponding to the nonzero entries $B_1(i, k_{ij})$ and $B_2(k_{ij}, j)$, which then have to be approximated by small values. If $u_d(k_{ij})$ (resp. $v_d(k_{ij})$) is large, then $v_b(j)$ (resp. $u_b(i)$) will have to be small since $u_d(k_{ij})v_b(j) \approx 0$ (resp. $u_b(i)v_d(k_{ij}) \approx 0$). Finally, either $u_b(i)$ or $v_b(j)$ has to be small, implying that $M_b(i, j)$ is approximated by a small value, because (u_b, v_b) is bounded independently of the value of d.

We now proceed as in Section 8.2. Let us first give an upper bound for the optimal value p^* of (MD-1d).

Lemma 8.7. *For $d > 1$, the optimal value p^* of* (MD-1d) *is bounded above by $|E| - |E^*|$, i.e.,*

$$p^* = \inf_{u\in\mathbb{R}^m, v\in\mathbb{R}^n} ||M - uv^T||_W^2 \leq |E| - |E^*|. \tag{8.2}$$

Proof. Let us build the following feasible solution (u, v) of (MD-1d) where (u_b, v_b) is an optimal solution of (MB) and (u_d, v_d) is defined as

$$u_d(k_{ij}) = \begin{cases} d^K & \text{if} \quad u_b(i) = 0, \\ d^{1-K} & \text{if} \quad u_b(i) = 1, \end{cases} \quad \text{and} \quad v_d(k_{ij}) = \begin{cases} d^K & \text{if} \quad v_b(j) = 0, \\ d^{1-K} & \text{if} \quad v_b(j) = 1, \end{cases}$$

with $K \in \mathbb{R}$ and k_{ij} the index of the column of B_1 and the row of B_2 corresponding to the zero entry (i, j) of M_b (i.e., $(i, j) = (i_{k_{ij}}, j_{k_{ij}})$).

One can check that

$$(uv^T) \circ W = \begin{pmatrix} u_b {v_b}^T & d^{1-K} B_1 \\ d^{1-K} B_2 & dI_Z \end{pmatrix},$$

where $\circ$ is the component-wise (or Hadamard) product between two matrices, so that

$$p^* \leq ||M - uv^T||_W^2 = |E| - |E^*| + \frac{2Z}{d^{2(K-1)}}, \quad \forall K. \tag{8.3}$$

Since $d > 1$, taking the limit $K \to +\infty$ gives the result. □

We now prove a property similar to Lemma 8.3 for any solution with objective value smaller that $|E|$.

Lemma 8.8. *Let $d > \sqrt{|E|}$ and (i, j) be such that $M_b(i, j) = 0$, then the following holds for any pair (u, v) such that $||M - uv^T||_W^2 \leq |E|$:*

$$\min\Big(\max_{1\leq k\leq n} |u_i v_k|, \max_{1\leq p\leq m} |u_p v_j| \Big) \leq \frac{\sqrt{2}\,|E|^{\frac{3}{4}}}{\big(d - \sqrt{|E|}\big)^{\frac{1}{2}}}. \tag{8.4}$$

Proof. Without loss of generality we set $||u_b||_2 = ||v_b||_2$ by scaling u and v without changing uv^T. Observing that

$$||u_b||_2||v_b||_2 - \sqrt{|E|} = ||u_b v_b^T||_F - ||M_b||_F \leq ||M_b - u_b v_b^T||_F \leq ||M - uv^T||_W \leq \sqrt{|E|},$$

we have $||u_b||_2||v_b||_2 \leq 2\sqrt{|E|}$, and $||u_b||_2 = ||v_b||_2 \leq \sqrt{2}|E|^{\frac{1}{4}}$.
Assume $M_b(i,j)$ is zero for some pair (i,j) and let $k = k_{ij}$ denote the index of the corresponding column of B_1 and row of B_2 (i.e., such that $B_1(i,k) = B_2(k,j) = 1$). By construction, $u_d(k)v_d(k)$ has to approximate d in the objective function. This implies $(u_d(k)v_d(k) - d)^2 \leq |E|$ and then

$$u_d(k)v_d(k) \geq d - \sqrt{|E|} > 0.$$

Suppose $|u_d(k)|$ is greater than $|v_d(k)|$ (the case $|v_d(k)|$ greater than $|u_d(k)|$ is similar), this implies $|u_d(k)| \geq (d - |E|^{\frac{1}{2}})^{\frac{1}{2}}$. Moreover $u_d(k)v_j$ has to approximate zero in the objective function, since $B_2(k,j) = 1$, implying

$$(u_d(k)v_j - 0)^2 \leq |E| \quad \Rightarrow \quad |u_d(k)v_j| \leq \sqrt{|E|}.$$

Hence

$$|v_j| \leq \frac{\sqrt{|E|}}{|u_d(k)|} \leq \frac{|E|^{\frac{1}{2}}}{\left(d - \sqrt{|E|}\right)^{\frac{1}{2}}}, \tag{8.5}$$

and since $(\max_{1 \leq p \leq m} |u_p|)$ is bounded by $||u_b||_2 \leq \sqrt{2}|E|^{\frac{1}{4}}$, the proof is complete. □

One can now associate to any point with objective value smaller than $|E|$ a biclique of G_b, the graph generated by the biadjacency matrix M_b.

Corollary 8.9. *Let $d > \sqrt{|E|}$, then for any pair (u,v) such that $||M - uv^T||_W^2 \leq |E|$, the set*

$$\Omega(u,v) = I \times J, \text{ with } I = \{\, i \mid \exists j \text{ s.t. } |u_i v_j| > \beta \,\}, J = \{\, j \mid \exists i \text{ s.t. } |u_i v_j| > \beta \,\}, \tag{8.6}$$

where $\beta = \frac{\sqrt{2}\,|E|^{\frac{3}{4}}}{\left(d - \sqrt{|E|}\right)^{\frac{1}{2}}}$, defines a biclique of G_b.

The next lemma gives a lower bound for the value of p^*.

Lemma 8.10. *Let $0 < \epsilon \leq 1$. For any value of parameter d that satisfies $d > \frac{8|E|^{\frac{7}{2}}}{\epsilon^2} + |E|^{\frac{1}{2}}$, the infimum p^* of* (MD-1d) *satisfies*

$$|E| - |E^*| - \epsilon < p^*.$$

Proof. If $|E| = |E^*|$, the result is trivial since $p^* = 0$. Otherwise, suppose $p^* \leq |E| - |E^*| - \epsilon$ and let $\beta = \frac{\sqrt{2}\,|E|^{\frac{3}{4}}}{\left(d - \sqrt{|E|}\right)^{\frac{1}{2}}}$. First observe that $d > \frac{8|E|^{\frac{7}{2}}}{\epsilon^2} + |E|^{\frac{1}{2}}$ is equivalent to $2|E|\beta < \epsilon$. Then, by continuity of (MD-1d), for any δ such that $\delta < \epsilon$, there exists a pair (u, v) such that

$$||M_d - uv^T||_W^2 \leq |E| - |E^*| - \delta.$$

In particular, let us take $\delta = 2|E|\beta < \epsilon$. We can now proceed as for Lemma 8.5. By Corollary 8.9, $\Omega(u, v)$ corresponds to a biclique of G_b, with at most $|E^*|$ edges. Then, for $\beta \leq 1$, i.e., for $d \geq 2|E|^{\frac{3}{2}} + |E|^{\frac{1}{2}}$ satisfied for $0 < \epsilon \leq 1$,

$$(1 - \beta)^2(|E| - |E^*|) \leq ||M - uv^T||_W^2 \leq |E| - |E^*| - \delta.$$

Dividing the above inequalities by $|E| - |E^*| > 0$, we obtain

$$1 - 2\beta < (1 - \beta)^2 \leq 1 - \frac{\delta}{|E| - |E^*|} \leq 1 - \frac{\delta}{|E|} \Rightarrow \delta < 2|E|\beta,$$

a contradiction. □

Corollary 8.11. *For any $d > 8(mn)^{7/2} + \sqrt{mn}$, $M \in \{0, 1, d\}^{m \times n}$, and $W \in \{0, 1\}^{m \times n}$, it is $\mathcal{NP}$-hard to find an approximate solution of rank-one* (WLRA) *with objective function accuracy $1 - \frac{2\sqrt{2}(mn)^{7/4}}{(d - \sqrt{mn})^{1/2}}$.*

Proof. Let $d > 8(mn)^{7/2} + \sqrt{mn}$, $0 < \epsilon = \frac{2\sqrt{2}(mn)^{7/4}}{(d - \sqrt{mn})^{1/2}} < 1$, and $(\bar{u}, \bar{v})$ be an approximate solution of (W-1d) with absolute error $(1 - \epsilon)$, i.e., $p^* \leq \bar{p} = ||M - \bar{u}\bar{v}^T||_W^2 \leq p^* + 1 - \epsilon$. Lemma 8.10 applies because $d = \frac{8(mn)^{7/2}}{\epsilon^2} + \sqrt{mn} \geq \frac{8(st)^{7/2}}{\epsilon^2} + \sqrt{st} \geq \frac{8|E|^{7/2}}{\epsilon^2} + |E|^{1/2}$. Using Lemmas 8.7 and 8.10, the rest of the proof is identical as the one of Theorem 8.1. Since the reduction from (MB) to (MD-1d) is polynomial (description of matrices W and M has polynomial length, since the increase in matrix dimensions from M_b to M is polynomial), we conclude that finding such an approximate solution for (MD-1d) is $\mathcal{NP}$-hard. □

We can now easily derive Theorem 8.2, which deals with the hardness of rank-one (WLRA) with a bounded matrix M.

Proof of Theorem 8.2. Replacing M by $M' = \frac{1}{d}M$ in (MD-1d) gives an equivalent problem with objective function multiplied by $\frac{1}{d^2}$, since $\frac{1}{d^2}||M - uv^T||_W^2 = ||M' - \frac{uv^T}{d}||_W^2$. Taking $d = 2^5(mn)^{7/2} + \sqrt{mn}$ in Corollary 8.11, we find that it is $\mathcal{NP}$-hard to compute an approximate solution of rank-one (WLRA) for $M \in [0, 1]^{m \times n}$ and $W \in \{0, 1\}^{m \times n}$, and with objective function accuracy less than $\frac{1}{d^2}\left(1 - \frac{2\sqrt{2}(mn)^{7/4}}{(d - \sqrt{mn})^{1/2}}\right) = \frac{1}{2d^2} \geq 2^{-12}(mn)^{-7}$. □

Concluding Remarks

In this chapter, we have studied the complexity of the weighted low-rank approximation problem (WLRA), and proved that finding an approximate solution is $\mathcal{NP}$-hard, already in the rank-one case, both for positive and for binary weights (the latter also corresponding to low-rank matrix completion with noise, or PCA with missing data).

Nevertheless, some questions remain open. In particular,

- ◇ When W is the matrix of all ones, WLRA can be solved in polynomial-time. We have shown that, when the ratio between the largest and the smallest entry in W is large enough, the problem is $\mathcal{NP}$-hard (Theorem 8.1). It would be interesting to investigate the gap between these two facts, i.e., what is the minimum ratio of the entries of W so that WLRA is $\mathcal{NP}$-hard?

- ◇ When $\text{rank}(W) = 1$, WLRA can be solved in polynomial-time (cf. Section 8.1) while it is $\mathcal{NP}$-hard for general matrix W (with rank up to $\min(m, n)$). But what is the complexity of (WLRA) if the rank of the weight matrix W is fixed and greater than one, e.g., if $\text{rank}(W) = 2$?

- ◇ When data is missing, the rank-one matrix approximation problem is $\mathcal{NP}$-hard in general. Nevertheless, it has been observed [27] that when the given entries are sufficiently numerous, well-distributed in the matrix, and affected by a relatively low level of noise, the original uncorrupted low-rank matrix can be recovered accurately, with a technique based on convex optimization (minimization of the nuclear norm of the approximation, which can be cast as a semidefinite program). It would then be especially interesting to analyze the complexity of the problem given additional assumptions on the data matrix, for example on the noise distribution, and deal in particular with situations related to applications.

Conclusion

In this conclusion, we first summarize our results and give some directions for further research. We then review again the three main themes of this thesis: complexity, algorithms and applications, and try to stand back and make some final general comments.

Summary and Further Research

In this thesis, we have explored a relatively recent problem in linear algebra, namely nonnegative matrix factorization (NMF). It is a linear dimensionality reduction technique for nonnegative data, and requires factors of the corresponding low-rank matrix approximation to be nonnegative. These additional constraints enhance compression (through sparsity) and interpretability (because basis elements are sparse and can be interpreted in the same way as the data, e.g., as images or texts, and because they lead to an additive and part-based representation).

We started this thesis with some theoretical considerations, and studied in Chapter 3 the problem of determining the smallest value of the rank for which an exact nonnegative matrix factorization exists (called the nonnegative rank). Defining a new closely related quantity (the restricted nonnegative rank), and using a geometric interpretation, we were able to understand better the complexity of the nonnegative rank computation, and to provide some new lower and upper bounds for the nonnegative rank. The geometric interpretation also gave new insights on the NMF problem. For example, it allows the reduction of the dimension of the search space, hence it might be a starting point for new efficient NMF algorithms (see also [148]). However, there are still many open questions related to the computational complexity of both NMF and nonnegative rank computation. For example, are these problems difficult when the rank of the matrix to be factored is fixed (to three, or higher values)? Is NMF difficult when the factorization rank is fixed (to two, or higher values)? The nonnegative rank can also be used to characterize the size of extended formulations, and it would be interesting to investigate further in this direction. For

example, would it be possible to apply nonnegative rank computation algorithms on the slack matrices of integer programs in order to design extended formulations? Could an approximate nonnegative factorization (i.e., NMF) be used to find approximate extended formulations? The fact that slack matrices have exponential size (in the dimension of the polytope) is a drawback that will have to be overcome.

In Chapter 4, three standard NMF algorithms (MU, ANLS and HALS) were studied, along with two new approaches to speed up their convergence: one based on the reorganization of the updates performed at each iteration, and the other based on a multilevel approach valid for specific classes of datasets. Using the properties of the NMF optimization problem and the structure of the solutions (part-based), we gave an explanation for the good performances of HALS. Further research on this topic includes the design of more effective algorithms. It also seems non-trivial to come up with good stopping criteria to be used to decide when to switch between blocks of variables in the general framework of an inexact coordinate descent scheme. This could potentially improve the current approaches (see Section 4.2).

In Chapter 5, the rank-one subproblems arising in NMF were studied. They amount to approximating a not necessarily nonnegative matrix with a rank-one nonnegative matrix (R1NF), which we proved to be $\mathcal{NP}$-hard using a reduction from the maximum-edge biclique problem. Using a link between the stationary points of R1NF and the bicliques of a graph, we proposed a new efficient biclique finding algorithm. We also generalized the MU algorithm of NMF for matrices with negative entries, and gave an explanation for the worse performance of multiplicative updates compared to the HALS algorithm.

In Chapter 6, a new recursive approach to solve NMF was introduced. Based on the introduction of underapproximation constraints, it enables the extraction of features in a recursive way, like PCA, but preserving nonnegativity, and the computation of sparse solutions, enhancing interpretability. It was explained in Chapter 7 how and why this technique is useful for the analysis of hyperspectral data. In particular, we showed that a variant based on ℓ_1-norm minimization was able to recursively extract constitutive materials in hyperspectral images in a robust and efficient way, as experimentally demonstrated on images associated with space object material identification and on HYDICE and related remote sensing images. It would be interesting to generalize some of these results to tensors. In particular, could the recursive approach based on underapproximation applied to tensors be useful in some applications?

Finally, in Chapter 8, low-rank matrix approximation with weights or missing data, but without nonnegativity constraints on the factors, was proved to be $\mathcal{NP}$-hard, already when computing an approximate rank-one solution. These results also apply to NMF, i.e., weighted NMF and NMF with missing data are both $\mathcal{NP}$-hard, even to approximate in the rank-one case.

Complexity

The standard low-rank matrix approximation problem can be defined as

$$\min_{X \in \mathbb{R}^{m \times n}} \quad ||M - X||_F^2 \quad \text{such that} \quad \text{rank}(X) \leq r. \tag{LRA}$$

Using the singular value decomposition, one can obtain an optimal solution (see Theorem 2.3), and also characterize the set of all stationary points. In fact, letting

$$S = \{ \ (\sigma, u, v) \in \mathbb{R}_+ \times \mathbb{R}^m \times \mathbb{R}^n \mid ||u||_2 = ||v||_2 = 1, M^T u = \sigma v, Mv = \sigma u \ \}$$

be the set of singular triplets of M, all stationary points are of the form

$$X_s = \sum_{i=1}^{r} \sigma_i u_i v_i^T,$$

where $(\sigma_i, u_i, v_i) \in S \ \forall i$ and $(\sigma_i, u_i, v_i) \neq (\sigma_j, u_j, v_j) \forall i \neq j$, with at least $\binom{\min(m,n)}{r}$ such points if M is full-rank, see, e.g., [89, p.29]. One nice feature of (LRA) is that any local minimum is also global (as for convex optimization problems), and all other stationary points are saddle points, see Theorem 2.5.

A recurring theme in this thesis is that, as soon as one perturbs the problem[6] by either modifying the domain (i.e., adds constraints on matrix X) or changing the objective function, the structure of (LRA) is typically lost, and the problem becomes difficult to solve, often even in the rank-one case $r = 1$. Intuitively, the reason is that there are still exponentially many stationary points, including many local minima, while global optimality can no longer be checked easily, see, e.g., Figure 8.1 for an illustration when weights are attached to the entries of matrix M. In this thesis, we have considered several variants of LRA, which can be equivalently reformulated as

$$\min_{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{r \times n}} \quad ||M - UV||_F^2, \tag{LRA}$$

and all were shown to be $\mathcal{NP}$-hard. Table 8.1 summarizes these computational complexity results[7].

At first sight, these computational complexity results are quite discouraging. However, it appears in practice that approximate solutions are often satisfactory. In some situations, it is even possible to have a guarantee of optimality. For example, in case of missing data, a semidefinite programming reformulation

[6] Perturbed variants of other related polynomially solvable problems, such as solving linear systems or linear programming, have also been shown to be NP-hard, see the work of Jiri Rohn and co-authors [130, 131].

[7] Notice that $\mathcal{NP}$-hardness for $r = 1$ implies $\mathcal{NP}$-hardness for r fixed (see construction of Theorem 6.6), and $\mathcal{NP}$-hardness for r fixed implies $\mathcal{NP}$-hardness for r not fixed.

Variant	$r = 1$	r fixed	r not fixed
LRA	polynomial	polynomial	polynomial
SPCA - Sparsity on U	$\mathcal{NP}$-hard [46]	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard
NMF - $U, V \geq 0$, $M \geq 0$	polynomial	open	$\mathcal{NP}$-hard [148]
NF - $U, V \geq 0$, $M \ngeq 0$	$\mathcal{NP}$-hard (Th. 5.3)	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard
NMU - $U, V \geq 0$, $UV \leq M$	$\mathcal{NP}$-hard (Th. 6.5)	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard
WLRA - $\min \|\|M - UV\|\|_W^2$	$\mathcal{NP}$-hard (Th. 8.1)	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard
LRAMD - missing entries	$\mathcal{NP}$-hard (Th. 8.2)	$\mathcal{NP}$-hard	$\mathcal{NP}$-hard [29]

Table 8.1: Computational complexity of several low-rank matrix approximation problems.

based on the nuclear norm minimization allows to recover the solution of the original problem in some specific circumstances [25]. Along the same line, the maximum-edge biclique problem, used in the reductions for our $\mathcal{NP}$-hardness proofs, can be solved with a convex reformulation when the bipartite graph contains a planted biclique, i.e., a biclique with significantly more edges than the other bicliques [3]. Hence there might exist classes of instances for which these low-rank matrix approximation problems can be solved in polyomial time, and it would be particularly interesting to characterize them.

Algorithms

In Chapter 4, we have presented several algorithms for solving

$$\min_{U \in \mathbb{R}^{m \times r}, V \in \mathbb{R}^{r \times n}} ||M - UV||_F^2, \quad \text{such that} \quad U \geq 0, V \geq 0, \qquad \text{(NMF)}$$

and used them throughout the thesis. It appears that most NMF algorithms are two-block coordinate descent schemes, i.e., successively optimizing U for V fixed and then V for U fixed. There are two reasons for that fact:

1. The corresponding subproblems are convex nonnegative least squares (NNLS), hence efficiently solvable, potentially up to global optimality.

2. The gradient of these subproblems can be updated efficiently ; in fact, we have shown that it is at least twice less expensive than reconstructing it from scratch (and it is in general even less expensive), see Section 4.2.

We are actually not aware of any efficient algorithm updating both matrices U and V simultaneously.

We have then observed that an exact coordinate descent applied on these NNLS subproblems, i.e., the hierarchical alternating least squares algorithm

(HALS), was the best strategy among the ones we have tested, which was motivated by the following two facts:

1. Structure of NNLS. The problem $\min_{U \geq 0} ||M - UV||_F^2$ (resp. $\min_{V \geq 0} ||M - UV||_F^2$) can be decoupled into m (resp. n) completely independent NNLS subproblems in r variables, corresponding to each row (resp. column) of M, each with Hessian matrix VV^T (resp. U^TU).

2. Structure of NMF solutions. Because of the nonnegativity constraints, NMF solutions are typically part-based, i.e., rows of V (resp. columns of U) shares few nonzero entries. Therefore matrix VV^T (resp. U^TU) typically features large diagonal entries, and the coupling between variables in the NNLS subproblems is rather low, with a beneficial effect on the efficiency of coordinate descent schemes.

Hence HALS combines cheap (because it only uses first-order information) and effective (because subproblems are almost separable) updates.

This illustrates the general fact that it is crucial to take into account the properties of a given problem to select and design efficient algorithms. In particular, we have proposed two approaches to speed up significantly NMF algorithms: reorganizing ordering of the updates in Section 4.2, and using a multilevel approach in Section 4.3. We believe it should be possible to go further in this direction and take advantage of other properties of NMF. For example, we typically observe convergence of the sparsity patterns of the factors before actual convergence. When the positions of the zeros in U and V stop changing after several updates, one could simply fix all these zero entries and update the remaining entries using (unconstrained) least squares, as done in active-set like methods (see Appendix A).

Applications

NMF is a dimensionality reduction technique used as a preprocessing step for classification tasks, and can be applied in many different situations, e.g., image processing in (see Chapter 1), text mining (see Chapter 5 and [11]), and hyperspectral image analysis (see Chapter 7). The advantage of NMF over standard low-rank approximation techniques such as PCA is its interpretability (part-based and sparse representation, see Chapter 1) which makes its post-processing much easier. However, there are two main pitfalls to using NMF in practice:

1. Complexity of the underlying optimization problem. In fact, Vavasis showed $\mathcal{NP}$-hardness of NMF [148], and many closely related variants appears to be $\mathcal{NP}$-hard as well (see Complexity above). In addition to the theoretical reasons outlined earlier, NMF can be interpreted as a

clustering technique (such as k-means [53]), closely related to difficult combinatorial optimization problems [55]. Another closely related problem is the NMF of symmetric matrices with one factor forced to be equal to the transpose of the other ($M = UU^T$), which amounts to finding the decomposition of completely positive matrices (see Chapter 2).

2. Non-uniqueness. Solutions to NMF are in general non-unique. This can be shown for example using its geometric interpretation presented in Section 3.4.1: there might exist many different solutions to the equivalent nested polytopes problem (see also [103]), i.e., there might exist optimal solutions (U, V) and (U', V') with $||M - UV||_F = ||M - U'V'||_F$ and $UV \neq U'V'$. In particular, when restarting several times an NMF algorithm, we often obtain quite different solutions with nearby objective function values, and it is rather difficult to decide which one is the best for the application of interest.
   Moreover, for a solution (U, V), it is possible that a non-monomial (i.e., not a permutation of a diagonal matrix) invertible matrix D exists such that $(UD, D^{-1}V)$ is nonnegative, and these 'equivalent' solutions could potentially lead to different interpretations (in particular if the sparsity patterns of matrices UD and $D^{-1}V$ are different from those of U and V respectively).

The second pitfall can sometimes be alleviated by incorporating prior information into the model. In fact, in many applications, additional information on the structure of the solutions is available, such as sparsity or smoothness, cf. introduction of Chapter 7.

For example, we have proposed in Chapters 6 and 7 a new framework based on nonnegative underapproximations to analyze hyperspectral images which makes the optimal solution of the underlying optimization problem unique. Ideally, for each application, one should try to design an appropriate model in order to reduce the degrees of freedom in NMF, allowing the algorithms to obtain solutions that fit the situation at hand. Objective functions different from the Frobenius norm can also be used to improve performances, as shown in Section 7.3 (ℓ_1-norm).

Bibliography

[1] A. Aggarwal, H. Booth, J. O'Rourke, and S. Suri. Finding minimal convex nested polygons. *Information and Computation*, 83(1):98–110, 1989.

[2] G. Alexe, S. Alexe, Y. Crama, S. Foldes, P.L. Hammer, and B. Simeone. Consensus algorithms for the generation of all maximal bicliques. *Discrete Applied Mathematics*, 145(1):11–21, 2004.

[3] B. Ames and S.A. Vavasis. Nuclear norm minimization for the planted clique and biclique problems. arXiv:0901.3348v1, 2009.

[4] K.M. Anstreicher and L.A. Wolsey. Two "well-known" properties of subgradient optimization. *Mathematical Programming*, 120(1):213–220, 2009.

[5] S. Arora and B. Barak. *Computational Complexity*. Cambridge University Press, New York, 2009.

[6] L.B. Beasley and T.J. Laffey. Real rank versus nonnegative rank. *Linear Algebra and its Applications*, 431 (12):2330–2335, 2009.

[7] A. Ben-Tal and A. Nemirovski. On polyhedral approximations of the second-order cone. *Mathematics of Operations Research*, 26(2):193–205, 2001.

[8] A. Berman and R.J. Plemmons. Rank factorization of nonnegative matrices. *SIAM Review*, 15 (3):655, 1973.

[9] A. Berman and R.J. Plemmons. *Nonnegative matrices in the mathematical sciences*. SIAM, 1994.

[10] M.W. Berry, M. Browne, A. Langville, V.P. Pauca, and R.J. Plemmons. Algorithms and Applications for Approximate Nonnegative Matrix Factorization. *Computational Statistics and Data Analysis*, 52:155–173, 2007.

[11] M.W. Berry, N. Gillis, and F. Glineur. Document Classification Using Nonnegative Matrix Factorization and Underapproximation. In *Proc. of*

the IEEE Int. Symp. on Circuits and Systems (ISCAS), pages 2782–2785, 2009. ISBN: 978-1-4244-3828-0.

[12] D.P. Bertsekas. *Nonlinear Programming: Second Edition*. Athena Scientific, Massachusetts, 1999.

[13] J. Bhadury and R. Chandrasekaran. Finding The Set of All Minimal Nested Convex Polygons. In *Proceedings of the 8th Canadian Conf. on Computational Geometry*, pages 26–31, 1996.

[14] M. Biggs, A. Ghodsi, and S.A. Vavasis. Nonnegative Matrix Factorization via Rank-One Downdate. In *25th Int. Conf. on machine learning (ICML)*, 2008.

[15] L. Blum. Computing over the reals: Where Turing meets Newton. *Notices of the American Mathematical Society*, 51:1024–1034, 2004.

[16] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and real computation*. Springer-Verlag, New York, 1998.

[17] J.W. Boardman. Geometric mixture analysis of imaging spectrometry data. In *Proc. IGARSS 4, Pasadena, Calif., pp. 2369–2371*, 1994.

[18] C. Boutsidis and E. Gallopoulos. SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41:1350–1362, 2008.

[19] C. Boyce, A. Ross, M. Monaco, L. Hornak, and X. Li. Multispectral iris analysis: A preliminary study. In *Proceedings of the Conf. on Computer Vision and Pattern Recognition Workshop (CVPRW'06), pp. 51–60*, 2006.

[20] S. Boyd, A. Hassibi, S.J. Kim, and L. Vandenberghe. A Tutorial on Geometric Programming. *Optimization and Engineering*, 8(1):67–127, 2007.

[21] J.H. Bramble. *Multigrid methods*. Number 294 Pitman Research Notes in Mathematic Series. Longman Scientific & Technical, UK, 1995.

[22] A. Brandt. Guide to multigrid development. *W. Hackbusch and U. Trottenberg, eds., Multigrid Methods, Lecture Notes in Mathematics, Springer*, 960:220–312, 1982.

[23] W.L. Briggs. *A Multigrid Tutorial*. SIAM, Philadelphia, 1987.

[24] K. Burgers, Y. Fessehatsion, S. Rahmani, J.Y. Seo, and T. Wittman. A comparative analysis of dimension reduction algorithms on hyperspectral data. Available at `http://www.math.ucla.edu/~wittman`, 2009.

[25] E.J. Candès and B. Recht. Exact Matrix Completion via Convex Optimization. *Foundations of Computational Mathematics*, 9:717–772, 2009.

[26] E. Carlini and F. Rapallo. Probability matrices, non-negative rank, and parameterization of mixture models. *Linear Algebra and its Applications*, 433:424–432, 2010.

[27] D. Chen and R.J. Plemmons. Nonnegativity Constraints in Numerical Analysis. In *A. Bultheel and R. Cools (Eds.), Symposium on the Birth of Numerical Analysis, World Scientific Press.*, 2009.

[28] P. Chen. Optimization Algorithms on Subspaces: Revisiting Missing Data Problem in Low-Rank Matrix. *Int. J. of Computer Vision*, 80(1):125–142, 2008.

[29] A.L. Chistov and D.Yu. Grigoriev. Complexity of quantifier elimination in the theory of algebraically closed fields. *Proceedings of the 11th Symposium on Mathematical Foundations of Computer Science, Lecture Notes in Computer Science, Springer*, 176:17–31, 1984.

[30] M.T. Chu. On the nonnegative rank of Euclidean distance matrices, II. `http://www4.ncsu.edu/~mtchu/Research/Papers/letter04.pdf`, 2010.

[31] M.T. Chu and M.M. Lin. Low-Dimensional Polytope Approximation and Its Applications to Nonnegative Matrix Factorization. *SIAM J. Sci. Comput.*, 30 (3):1131–1155, 2008.

[32] A. Cichocki, S. Amari, R. Zdunek, and A.H. Phan. *Non-negative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation.* Wiley-Blackwell, 2009.

[33] A. Cichocki and A.-H. Phan. Fast local algorithms for large scale Nonnegative Matrix and Tensor Factorizations. *IEICE Trans. on Fundamentals of Electronics*, Vol. E92-A No.3:708–721, 2009.

[34] A. Cichocki, R. Zdunek, and S. Amari. Non-negative Matrix Factorization with Quasi-Newton Optimization. In *Lecture Notes in Artificial Intelligence, Springer*, volume 4029, pages 870–879, 2006.

[35] A. Cichocki, R. Zdunek, and S. Amari. Hierarchical ALS Algorithms for Nonnegative Matrix and 3D Tensor Factorization. In *Lecture Notes in Computer Science, Vol. 4666, Springer, pp. 169-176*, 2007.

[36] K.L. Clarkson. Algorithms for polytope covering and approximation. In *Proceedings of the Third Workshop on Algorithms and Data Structures*, pages 246–252, 1993.

[37] J.E. Cohen and U.G. Rothblum. Nonnegative ranks, Decompositions and Factorization of Nonnegative Matrices. *Linear Algebra and its Applications*, 190:149–168, 1993.

[38] P. Comon. Independent component analysis, A new concept? *Signal Processing*, 36:287–314, 1994.

[39] M. Conforti, G. Cornuéjols, and G. Zambelli. Extended formulations in combinatorial optimization. *4OR: A Quarterly J. of Operations Research*, 10 (1):1–48, 2010.

[40] M.D. Craig. Minimum-volume tranforms for remotely sensed data. *IEEE Trans. on Geoscience and Remote Sensing*, 32 (3):542–552, 1994.

[41] J. Curry, A. Dougherty, and S. Wild. Improving non-negative matrix factorizations through structured initialization. *Pattern Recognition*, 37(11):2217–2232, 2004.

[42] G. Das. *Approximation schemes in computational geometry*. PhD thesis, University of Wisconsin-Madison, 1990.

[43] G. Das and M. Goodrich. On the Complexity of Optimization Problems for Three-Dimensional Convex Polyhedra and Decision Trees. *Computational Geometry: Theory and Applications*, 8:123–137, 1997.

[44] G. Das and D.A. Joseph. The Complexity of Minimum Convex Nested Polyhedra. In *Proc. of the 2nd Canadian Conf. on Computational Geometry*, pages 296–301, 1990.

[45] A. d'Aspremont. Semidefinite Optimization with Applications in Sparse Multivariate Statistics. Talk at BIRS, Banff, available at `http://www.princeton.edu/~aspremon/Banff07.pdf`, 2007.

[46] A. d'Aspremont, L. El Ghaoui, M.I. Jordan, and G.R.G. Lanckriet. A Direct Formulation for Sparse PCA Using Semidefinite Programming. *SIAM Rev.*, 49(3):434–448, 2007.

[47] M.E. Daube-Witherspoon and G. Muehllehner. An iterative image space reconstruction algorithm suitable for volume `ECT`. *IEEE Trans. Med. Imaging*, 5:61–66, 1986.

[48] M. Dawande, P. Keskinocak, and S. Tayur. On the biclique problem in bipartite graphs. GSIA Working Paper 1996-04, Carnegie Mellon University, Pittsburgh, PA 15213, USA, 1996.

[49] D. de Caen, D.A. Gregory, and N.J. Pullman. The boolean rank of zero-one matrices. In *Proc. 3rd Caribbean Conf. on Combinatorics and Computing, pp. 169-173*, 1981.

[50] K. Devarajan. Nonnegative Matrix Factorization: An Analytical and Interpretive Tool in Computational Biology. *PLoS Computational Biology*, 4(7), e1000029, 2008.

[51] I.S. Dhillon, D. Kim, and S. Sra. Fast Newton-type Methods for the Least Squares Nonnegative Matrix Approximation problem. In *Proc. of SIAM Conf. on Data Mining*, 2007.

[52] I.S. Dhillon and S. Sra. Nonnegative Matrix Approximations: Algorithms and Applications. Technical report, University of Texas (Austin), 2006. Dept. of Computer Sciences.

[53] C. Ding, X. He, and H.D. Simon. On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering. In *SIAM Int'l Conf. Data Mining (SDM'05)*, pages 606–610, 2005.

[54] C. Ding, T. Li, W. Peng, and H. Park. Orthogonal nonnegative matrix tri-factorizations for clustering. In *Proc. SIGKDD, Int. Conf. on Knowledge Discovery and Data Mining*, pages 126–135, 2006.

[55] C. Ding, L. Tao, and M.I. Jordan. Nonnegative matrix factorization for combinatorial optimization: Spectral clustering, graph matching, and clique finding. *IEEE Int. Conf. on Data Mining*, pages 183–192, 2008.

[56] C. Ding, Y. Zhang, T. Li, and S.R. Holbrook. Biclustering Protein Complex Interactions with a Biclique Finding Algorithm. In *Sixth IEEE Int. Conference on Data Mining*, pages 178–187, 2006.

[57] E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. *Math. Program., Ser. A*, 91:201–213, 2002.

[58] B. Dong, M.M. Lin, and M.T. Chu. Nonnegative rank factorization via rank reduction. Preprint, 2008.

[59] D. Donoho and V. Stodden. When does non-negative matrix factorization give a correct decomposition into parts? In *In Advances in Neural Information Processing 16*, 2003. MIT Press.

[60] M. Dür. Copositive Programming - a Survey. In *Book of Abstracts of the 14th Belgian-French-German Conference on Optimization (BFG'09), Leuven*, 2010.

[61] R.J. Duffin, E.L. Peterson, and C. Zener. *Geometric Programming: Theory and Applications*. Wiley, New York, 1967.

[62] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[63] L.R. Ford Jr. and D.R. Fulkerson. *Flows in Networks*. Princeton University press, Paris, 1962.

[64] K.R. Gabriel and S. Zamir. Lower Rank Approximation of Matrices by Least Squares With Any Choice of Weights. *Technometrics*, 21(4):489–498, 1979.

[65] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, New York, 1997.

[66] Y. Gao and G. Church. Improving molecular cancer class discovery through sparse non-negative matrix factorization. *Bioinformatics*, 21(21):3970–3975, 2005.

[67] M.R. Garey and D.S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. Freeman, San Francisco, 1979.

[68] N. Gillis. Approximation et sous-approximation de matrices par factorisation positive: algorithmes, complexité et applications. Master's thesis, Université catholique de Louvain, 2007. In French.

[69] N. Gillis and F. Glineur. Nonnegative Factorization and The Maximum Edge Biclique Problem. CORE Discussion paper 2008/64, 2008.

[70] N. Gillis and F. Glineur. Nonnegative Matrix Factorization and Underapproximation. Communication at 9th Int. Symposium on Iterative Methods in Scientific Computing, Lille, France, 2008.

[71] N. Gillis and F. Glineur. A Multilevel Approach for Nonnegative Matrix Factorization. CORE Discussion paper 2010/47, 2010.

[72] N. Gillis and F. Glineur. Accelerated Multiplicative Updates and Hierarchical ALS Algorithms for Nonnegative Matrix Factorization. Preprint, 2010.

[73] N. Gillis and F. Glineur. Low-Rank Matrix Approximation with Weights or Missing Data is NP-hard. CORE Discussion paper 2010/75, 2010.

[74] N. Gillis and F. Glineur. On the Geometric Interpretation of the Nonnegative Rank. CORE Discussion paper 2010/51, 2010.

[75] N. Gillis and F. Glineur. Using underapproximations for sparse nonnegative matrix factorization. *Pattern Recognition*, 43(4):1676–1687, 2010.

[76] N. Gillis and R.J. Plemmons. Dimensionality Reduction, Classification, and Spectral Mixture Analysis using Nonnegative Underapproximation. In *SPIE conference Volume 7695, paper 46, Orlando*, 2010.

[77] N. Gillis and R.J. Plemmons. Dimensionality Reduction, Classification, and Spectral Mixture Analysis using Nonnegative Underapproximation. *Optical Engineering*, 2011. in press.

[78] F. Glineur. Computational experiments with a linear approximation of second order cone optimization. Image Technical Report 0001, Service de Mathématique et de Recherche Opérationnelle, Faculté Polytechnique de Mons, 2000.

[79] M.X. Goemans. Smallest compact formulation for the permutahedron. Talk at ISMP, Chicago, 2009.

[80] G.H. Golub and C.F. Van Loan. *Matrix Computation, 3rd Edition*. The Johns Hopkins University Press Baltimore, 1996.

[81] S. Gratton, A. Sartenaer, and P. Toint. Recursive trust-region methods for multiscale nonlinear optimization. *SIAM J. on Optimization*, 19:414–444, 2008.

[82] L. Grippo and M. Sciandrone. On the convergence of the block nonlinear Gauss-Seidel method under convex constraints. *Operations Research Letters*, 26:127–136, 2000.

[83] B. Grung and R. Manne. Missing values in principal component analysis. *Chemom. and Intell. Lab. Syst.*, 42:125–139, 1998.

[84] D. Guillamet and J. Vitrià. Non-negative matrix factorization for face recognition. *Lecture Notes in Computer Science, Volume 2504/2002*, 2504:336–344, 2002.

[85] Z. Guo, T. Wittman, and S. Osher. L1 unmixing and its application to hyperspectral image enhancement. In *Proc. SPIE Conf. on Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XV*, 2009.

[86] C. Gurwitz. Weighted median algorithms for l_1 approximation. *BIT Numerical Mathematics*, 30 (2):301–310, 1990.

[87] J. Hannah and T.J. Laffey. Nonnegative factorization of completely positive matrices. *Linear Algebra and its Applications*, 55:1–9, 1983.

[88] F.L. Hitchcock. The Distribution of a Product from Several Sources to Numerous Localities. *J. Math. Phys.*, 10:224–230, 1941.

[89] N.-D. Ho. *Nonnegative Matrix Factorization - Algorithms and Applications*. PhD thesis, Université catholique de Louvain, 2008.

[90] D.S. Hochbaum. Approximating clique and biclique problems. *J. Algorithms*, 29(1):174–200, 1998.

[91] P.O. Hoyer. Nonnegative matrix factorization with sparseness constraints. *J. Machine Learning Research*, 5:1457–1469, 2004.

[92] A. Ifarraguerri and C.-I. Chang. Multispectral and hyperspectral image analysis with convex cones. *IEEE Trans. on Geoscience and Remote Sensing*, 37 (2):756–770, 1999.

[93] D. Jacobs. Linear fitting with missing data for structure-from-motion. *Vision and Image Understanding*, 82:57–810, 2001.

[94] S. Jia and Y. Qian. Constrained nonnegative matrix factorization for hyperspectral unmixing. *IEEE Trans. on Geoscience and Remote Sensing*, 47 (1):161–173, 2009.

[95] I.T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[96] V. Kaibel and K. Pashkovich. Constructing Extended Formulations from Reflection Relations. arXiv:1011.3597v1, 2010.

[97] H. Kim and H. Park. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics*, 23(12):1495–1502, 2007.

[98] H. Kim and H. Park. Non-negative Matrix Factorization Based on Alternating Non-negativity Constrained Least Squares and Active Set Method. *SIAM J. Matrix Anal. Appl.*, 30(2):713–730, 2008.

[99] J. Kim and H. Park. Toward Faster Nonnegative Matrix Factorization: A New Algorithm and Comparisons. In *Proceedings of IEEE Int. Conf. on Data Mining*, pages 353–362, 2008.

[100] B. Klingenberg, J. Curry, and A. Dougherty. Non-negative matrix factorization: Ill-posedness and a geometric algorithm. *Pattern Recognition*, 42(5):918–928, 2009.

[101] Y. Koren, R. Bell, and C. Volinsky. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer*, 42(8):30–37, 2009.

[102] N. Krislock and H. Wolkowicz. *Euclidean Distance Matrices and Applications*. Handbook of Semidefinite, Cone and Polynomial Optimization: Theory, Algorithms, Software and Applications, Miguel Anjos and Jean Lasserre (eds.), 2010. in preparation.

[103] H. Laurberg, M.G. Christensen, M.D. Plumbley, L.K. Hansen, and S.H. Jensen. Theorems on positive data: On the uniqueness of NMF. *Computational Intelligence and Neuroscience*, 2008. Article ID 764206.

[104] C.L. Lawson and R.J. Hanson. *Solving Least Squares Problems*. Prentice-Hall, 1974.

[105] D.D. Lee and H.S. Seung. Learning the Parts of Objects by Nonnegative Matrix Factorization. *Nature*, 401:788–791, 1999.

[106] D.D. Lee and H.S. Seung. Algorithms for Non-negative Matrix Factorization. *In Advances in Neural Information Processing*, 13, 2001.

[107] T. Lee and A. Shraibman. *Lower Bounds in Communication Complexity*. Foundations and Trends in Theoretical Computer Science, 2009.

[108] B. Levin. On Calculating Maximum Rank One Underapproximations for Positive Arrays. Technical report, Columbia University, 1985. Div. of Biostatistics.

[109] H. Li, C. Adal, W. Wang, D. Emge, and A. Cichocki. Non-negative matrix factorization with orthogonality constraints and its application to raman spectroscopy. *J. of VLSI Signal Processing*, 48:83–97, 2007.

[110] L. Li and Y.-J. Zhang. FastNMF: highly efficient monotonic fixed-point nonnegative matrix factorization algorithm with good applicability. *J. Electron. Imaging*, Vol. 18 (033004), 2009.

[111] S.Z. Li, X.W. Hou, H.J. Zhang, and Q.S. Cheng. Learning spatially localized parts-based representation. In *Proceedings of IEEE Int. Conf. on Computer Vision and Pattern Recognition*, pages 207–212, 2001.

[112] C.-J. Lin. On the Convergence of Multiplicative Update Algorithms for Nonnegative Matrix Factorization. In *IEEE Trans. on Neural Networks*, 2007.

[113] C.-J. Lin. Projected Gradient Methods for Nonnegative Matrix Factorization. *Neural Computation*, 19:2756–2779, 2007. MIT press.

[114] M.M. Lin and M.T. Chu. On the nonnegative rank of Euclidean distance matrices. *Linear Algebra and its Applications*, 433:681–689, 2010.

[115] G. Liu, K. Sim, and J. Li. Efficient Mining of Large Maximal Bicliques. *Springer Berlin, Lecture Notes in Computer Science*, pages 437–448, 2006.

[116] W.-S. Lu, S.-C. Pei, and P.-H. Wang. Weighted low-rank approximation of general complex matrices and its application in the design of 2-D digital filters. *IEEE Trans. Circuits Syst. I*, 44:650–655, 1997.

[117] I. Markovsky and M. Niranjan. Approximate low-rank factorization with structured factors. *Computational Statistics & Data Analysis*, 54:3411–3420, 2010.

[118] Y.M. Masalmah and M. Veléz-Reyes. A full algorithm to compute the constrained positive matrix factorization and its application in unsupervised unmixing of hyperspectral imagery. In *Proc. SPIE, Vol. 6966; doi:10.1117/12.779444*, 2008.

[119] L. Miao and H. Qi. Endmember extraction from highly mixed data using minimum volume constrained nonnegative matrix factorization. *IEEE Trans. on Geoscience and Remote Sensing*, 45 (3):765–777, 2007.

[120] P. Miettinen. The boolean column and column-row matrix decompositions. *Data Mining and Knowledge Discovery*, 17(1):39–56, 2008.

[121] J.S.B. Mitchell and S. Suri. Separation and Approximation of Polyhedral Surfaces. *Operations Research Letters*, 11:255–259, 1992.

[122] D. Nister, F. Kahl, and H. Stewenius. Structure from Motion with Missing Data is NP-Hard. In *IEEE 11th Int. Conf. on Computer Vision*, 2007.

[123] J. Orlin. Contentment in graph theory: Covering graphs with cliques. *Indagationes Mathematicae (Proceedings)*, 80 (5):406–424, 1977.

[124] P. Paatero and U. Tapper. Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.

[125] P.M. Pardalos and S.A. Vavasis. Quadratic programming with one negative eigenvalue is NP-hard. *J. of Global Optimization*, 1:15–22, 1991.

[126] V.P. Pauca, J. Piper, and R.J. Plemmons. Nonnegative matrix factorization for spectral data analysis. *Linear Algebra and its Applications*, 406 (1):29–47, 2006.

[127] R. Peeters. The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics*, 131(3):651–654, 2003.

[128] M.J.D. Powell. On Search Directions for Minimization Algorithms. *Mathematical Programming*, 4:193–201, 1973.

[129] G.B. Recht, M. Fazel, and P.A. Parrilo. Guaranteed Minimum Rank Solutions to Linear Matrix Equations via Nuclear Norm Minimization. *SIAM Review*, 52(3):471–501, 2010.

[130] J. Rohn. Linear Programming with Inexact Data is NP-Hard. *Zeitschrift fuer Angewandte Mathematik und Mechanik*, 78, Supplement 3:S1051–S1052, 1998.

[131] J. Rohn and V. Kreinovich. Nonnegative factorization of completely positive matrices. *SIAM J. on Matrix Analysis and Applications*, 16(2):415–420, 1995.

[132] A. Ross. Iris dataset obtained from west virginia university (wvu). Available at `http://www.wvu.edu/`, 2010.

[133] S. Sahni. Computationally related problems. *SIAM J. Computing*, 3(4):262–279, 2000.

[134] S. Sakellaridi, H.-R. Fang, and Y. Saad. Graph-based Multilevel Dimensionality Reduction with Applications to Eigenfaces and Latent Semantic Indexing. preprint, 2009.

[135] B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Riedl. Item-Based Collaborative Filtering Recommendation Algorithms. In *10th Int. WorldWideWeb Conference*, 2001.

[136] M. Schuermans. *Weighted Low Rank Approximation: Algorithms and Applications*. PhD thesis, Katholieke Universiteit Leuven, 2006.

[137] F. Shahnaz, M.W. Berry, A., V.P. Pauca, and R.J. Plemmons. Document clustering using nonnegative matrix factorization. *Information Processing and Management*, 42:373–386, 2006.

[138] N.Z. Shor. *Minimization Methods for Non-differentiable Functions*. Springer Series in Computational Mathematics, 1985.

[139] H. Shum, K. Ikeuchi, and R. Reddy. Principal component analysis with missing data and its application to polyhedral object modeling. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 17(9):854–867, 1995.

[140] N. Srebro and T. Jaakkola. Weighted Low-Rank Approximations. In *20th ICML Conf. Proceedings*, 2004.

[141] G.W. Stewart. Perturbation Theory for The Singular Value Decomposition. Technical report, UMIACS, 1990. Dept. of Computer Sciences.

[142] D. Terzopoulos. Image Analysis Using Multigrid Relaxation Methods. *J. Math. Phys.*, PAMI-8(2):129–139, 1986.

[143] L.B. Thomas. Rank factorization of nonnegative matrices. *SIAM Review*, 16(3):393–394, 1974.

[144] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Elsevier Academic Press, London, 2001.

[145] M.H. Van Benthem and M.R. Keenan. Fast algorithm for the solution of large-scale non-negativity constrained least squares problems. *J. Chemometrics*, 18:441–450, 2004.

[146] S.A. Vavasis. Quadratic programming is in NP. *Information Processing Letters*, 36:73–77, 1990.

[147] S.A. Vavasis. Complexity issues in global optimization: A survey. *In Horst, R. and Pardalos, P., editors, Handbook of Global Optimization - Kluwer Academic Publishers*, pages 27–41, 1995.

[148] S.A. Vavasis. On the complexity of nonnegative matrix factorization. *SIAM J. on Optimization*, 20(3):1364–1377, 2009.

[149] C.A. Wang. Finding Minimal Nested Polygons. *BIT*, 31:230–236, 1991.

[150] V. Watts. Fractional biclique covers and partitions of graphs. *Electronic J. of Combinatorics*, 13:#R74, 2006.

[151] M. Winter. N-findr: an algorithm for fast autonomous spectral end-member determination in hyperspectral data. In *Proc. SPIE Conf. on Imaging Spectrometry V*, 1999.

[152] M. Yannakakis. Expressing Combinatorial Optimization Problems by Linear Programs. *Computer and System Sciences*, 43:441–466, 1991.

[153] Q. Zhang, H. Wang, R.J. Plemmons, and V.P. Pauca. Tensor methods for hyperspectral data analysis: a space object material identification study. *J. Optical Soc. Amer. A*, 25(12):3001–3012, 2008.

[154] Z.-Y. Zhang, T. Li, C. Ding, X.W. Ren, and X.-S. Zhang. Binary matrix factorization for analyzing gene expression data. *Data Mining and Knowledge Discovery*, 20 (1):28–52, 2009.

[155] Z.-Y. Zhang, T. Li, C. Ding, and X.-S. Zhang. Binary matrix factorization with applications. In *Proceedings of 2007 IEEE Int. Conf. on Data Mining*, 2007.

[156] S. Zhong and J. Ghosh. Generative model-based document clustering: a comparative study. *Knowledge and Information Systems*, 8 (3):374–384, 2005.

[157] G.M. Ziegler. *Lectures on Polytopes*. Springer-Verlag, 1995.

Appendix A

Active set methods for NNLS

In a nutshell, active set methods for nonnegative least squares work in the following iterative fashion [104, Algorithm NNLS, p. 161]

0. Choose the set of active (zero) and passive (nonzero) variables.
1. Get rid of the nonnegativity constraints and solve the unconstrained least squares problem (LS) corresponding to the set of passive (nonzero) variables (the solution is obtained by solving a linear system, i.e., the normal equations);
2. Check the optimality conditions, i.e., the nonnegativity of passive variables, and the nonnegativity of the gradients of the active variables (see Chapter 1). If they are not satisfied:
3. Exchange variables between the set of active and the set of passive variables in such a way that the objective function is decreased at each step; and go to 1.

In (NMF), the problem of computing the optimal U (resp. V) for a given fixed V (resp. U) can be decoupled into m (resp. n) independent NNLS subproblems in r variables:

$$\min_{U_{i:}\in\mathbb{R}^r_+} ||M_{i:} - U_{i:}V||_F^2,\ 1 \le i \le m \quad (\text{resp. } \min_{V_{:j}\in\mathbb{R}^r_+} ||M_{:j} - UV_{:j}||_F^2,\ 1 \le j \le n).$$

Each of them amounts to solving a sequence of linear subsystems (with at most r variables, cf. step 1 above) of

$$U_{i:}(VV^T) = M_{i:}V^T,\ 1 \le i \le m \quad (\text{resp. } (U^TU)V_{:j} = U^TM_{:j},\ 1 \le j \le n).$$

In the worst case, one might have to solve every possible subsystem, which requires $\mathcal{O}(g(r))$ operations with[1] $g(r) = \sum_{i=1}^{r} \binom{r}{i} i^3 = \Theta(2^r r^3)$. Note that

[1] One can check that $(2^{(r-3)} - 1)(r-2)^3 \le g(r) \le 2^r r^3$.

VV^T and MV^T (resp. U^TU and U^TM) can be computed once for all to avoid redundant computations. Finally, one ANLS step requires at most $\mathcal{O}(mnr + (m+n)s(r)r^3)$ operations per iteration, where $s(r) \leq 2^r$. In the worst case, $s(r)$ is in $\Theta(2^r)$ while in practice it is in general much lower (as is the case for the simplex method for linear programming).

When m is reduced by a certain factor (e.g., four as in our multilevel approach presented in Section 4.3), the computational cost is not exactly reduced by the same factor, because the leading $(m+n)$ factor above also depends on n. However, in our applications, when m (number of pixels) is much larger than n (number of images), one can roughly consider the cost per iteration to be reduced by the same factor since $\frac{m+n}{4} \approx \frac{m}{4}$.