Université de Mons
Faculté Polytechnique
Mathématique et Recherche Opérationnelle

Sparsity and Nonnegativity in Least Squares Problems and Matrix Factorizations

Nicolas Nadisic

A thesis presented in partial fulfillment of the requirements for the degree of
Docteur en Sciences de l'Ingénieur et Technologies

Dissertation committee:

Prof. Bernard Gosselin	Université de Mons	Chair
Prof. Nicolas Gillis	Université de Mons	Supervisor
Prof. Arnaud Vandaele	Université de Mons	Co-supervisor
Prof. Sébastien Bourguignon	École Centrale de Nantes	
Prof. Jérémy Cohen	CNRS, Université de Lyon	
Prof. François Glineur	UCLouvain	

Abstract

Many tasks in machine learning and signal processing rely on linear models where data points can be explained as the results of linear interactions between latent features. To recover these features and extract useful information, we can leverage a priori knowledge or assumptions about the underlying structure of the data. In this thesis, we focus mainly on two assumptions: nonnegativity, meaning that the data points are generated by only additive interactions of features, and sparsity, meaning that only a few features interact to generate a given data point. We also consider the assumption of separability, stating that for each feature there is at least one pure data point, that is a point generated only by this feature. We develop variants of linear models that leverage the assumptions of sparsity and nonnegativity to improve performance or to handle cases that existing approaches fail to handle. We study the theoretical implications of these assumptions and we design algorithms to tackle the corresponding optimization problems. We especially focus on exact algorithms that are guaranteed to recover the underlying solution.

The contributions of this thesis are centered around nonnegative least squares problems (NNLS) and nonnegative matrix factorization (NMF). First, we propose a new variant of separable NMF that leverages the fact that when the separability assumption holds, then there are usually more than one pure data point per feature. We propose two algorithms based on this stronger assumption. Next, we propose a branch-and-bound algorithm to solve exactly sparsity-constrained NNLS problems, as opposed to existing approaches that often rely on heuristics. We also propose a biobjective extension of this algorithm that computes a set of solutions with different tradeoffs between reconstruction error and sparsity. Then, we introduce an algorithmic framework to enforce matrix-wise sparsity constraints on NNLS problems with multiple right-hand sides, and we prove it is optimal in some cases. Finally, we explore a variant of NMF combining the assumptions of separability and sparsity and we provide a guaranteed algorithm to tackle it. This variant is the first to handle the case where a feature is an additive linear combination of other features.

Although we focus on the models and algorithms rather than on the applications, our underlying motivation is to perform feature extraction in images and notably hyperspectral unmixing, that is, the identification of materials and their distribution in a hyperspectral image. All our proposed models are validated empirically on the unmixing of real-world hyperspectral images.

Résumé

De nombreuses tâches en apprentissage automatique et en traitement du signal reposent sur des modèles linéaires où les points de données peuvent être expliqués comme les résultats d'interactions linéaires entre des composantes latentes. Pour identifier ces composantes et extraire des informations utiles à partir de données brutes, nous pouvons exploiter des connaissances a priori ou des hypothèses sur la structure sous-jacente des données. Dans cette thèse, nous nous concentrons principalement sur deux hypothèses : la non-négativité, qui signifie que les points de données sont générés par des interactions additives de composantes, et la parcimonie, qui signifie que seulement quelques composantes interagissent pour générer un point de données. Nous considérons également l'hypothèse de séparabilité, qui stipule que pour chaque composante il existe au moins un point de données pur, c'est-à-dire un point généré uniquement par cette composante. Nous développons des variantes de modèles linéaires qui tirent parti des hypothèses de parcimonie et de non-négativité pour améliorer leurs performances ou pour traiter des cas que les approches existantes ne parviennent pas à traiter. Nous étudions les implications théoriques de ces hypothèses et nous concevons des algorithmes pour résoudre les problèmes d'optimisation correspondants. Nous nous concentrons particulièrement sur les algorithmes exacts qui sont garantis de retrouver la solution sous-jacente.

Les contributions de cette thèse sont centrées sur les problèmes de moindres carrés non-négatifs (NNLS) et la factorisation non-négative de matrices (NMF). Tout d'abord, nous proposons une nouvelle variante de la NMF séparable qui exploite le fait que, lorsque l'hypothèse de séparabilité est vérifiée, il y a généralement plus d'un point de données pur par composante. Nous proposons deux algorithmes basés sur cette hypothèse plus forte. Ensuite, nous proposons un algorithme de type séparation-et-évaluation pour résoudre exactement les problèmes NNLS sous contrainte de parcimonie, par opposition aux approches existantes qui reposent souvent sur des heuristiques. Nous proposons également une extension biobjective de cet algorithme qui calcule un ensemble de solutions représentant différents compromis entre l'erreur de reconstruction et la parcimonie. Puis nous présentons un cadre algorithmique pour appliquer des contraintes de parcimonie au niveau matriciel aux problèmes NNLS multi-colonnes, et nous prouvons qu'il est optimal dans certains cas. Enfin, nous étudions une variante de NMF combinant les hypothèses de séparabilité et de parcimonie, et nous fournissons un algorithme garanti pour la traiter. Cette variante est la première à gérer le cas où une composante est une combinaison linéaire additive

d'autres composantes.

Bien que nous nous concentrions sur les modèles et les algorithmes plutôt que sur les applications, notre motivation sous-jacente est l'extraction d'information dans les images, et notamment le démélange hyperspectral, c'est-à-dire l'identification des matériaux dans une image hyperspectrale et leur distribution dans l'image. Tous les modèles que nous proposons sont validés empiriquement par le démélange d'images hyperspectrales réelles.

Acknowledgements

This thesis would not have been possible without the time and energy many people have given me. I want to thank them here.

In the first place, I thank my PhD supervisor, Nicolas Gillis, for welcoming me in his team and for dedicating so much time to me. I thank him for his patience, his support, his contagious enthusiasm, and his generosity. He is a great mentor and I am very lucky to have been his student. I also thank my co-supervisor, Arnaud Vandaele, for always taking the time to help me and to answer my questions, even the most naive ones. He is one of the best teachers I know, and I am grateful to have worked with him. If you ever need an elegant proof that something is NP-complete, he is your man. I thank Jérémy Cohen for introducing me to the world of sparse optimization and for making our collaboration as productive as enjoyable.

I am grateful to the research mentors I had before my PhD, notably Mehdi Kaytoue in INSA Lyon and Anthony Ventresque in UCD, with special thanks to Loïc Cerf for hosting me in UFMG and for teaching me so much about data mining, writing, and dupe hacks. They gave me a taste for scientific research and I would not have thought about pursuing a PhD if it was not for them.

I thank all the colleagues and collaborators I had the chance to meet with the COLORAMAP team, alphabetically: Andersen, Anthony, Christophe, Christos, François, Hien, Junjun, Maryam, Olivier, Pierre, Tim, and Valentin. Special thanks to Anthony for introducing me to Julia, to Christophe for our fruitful collaboration, to Tim for proofreading my papers and writing insightful comments often longer than the actual paper, and to Olivier for the many times he stopped working to help me track down a bug or hear me complain about politics. I also thank all the members of the MARO department, and especially Guillaume for his help when I arrived in Mons and for teaching me juggling. I am grateful to the researchers at IRIT that welcomed me in Toulouse during my short stay in 2020, especially Cédric Févotte, Emmanuel Soubies, and Nicolas Dobigeon.

I thank the members of the PhD committee for accompanying me throughout the PhD and for the insightful discussions we had, and the jury members for accepting to read and evaluate this thesis.

Agradezco a Nadia por su amor, su apoyo y su paciencia, por animarme cuando lo necesitaba, y por siempre creer en mí. Спасибо Пине Пинуш за поддержку. Грау!

Finalement, je veux remercier mes parents, ces géants qui m'ont pris sur leurs épaules pour que je voie plus loin. Tout ce que je suis, je le dois à leur éducation, à leur amour, et à leurs efforts. Je leur dédie cette thèse.

I acknowledge the support by the European Research Council (ERC Starting Grant No 679515), and by the Fonds de la Recherche Scientifique-FNRS and the Fonds Wetenschappelijk Onderzoek - Vlanderen (Excellence of Science Grant No O005318F-RG47).

"Je suis de ceux qui pensent que la science est d'une grande beauté. Un scientifique dans son laboratoire est non seulement un technicien : il est aussi un enfant placé devant des phénomènes naturels qui l'impressionnent comme des contes de fées."

– Marie Skłodowska-Curie

Contents

Notation

$\mathbb{R}$	set of real numbers
$\mathbb{R}_+$	set of real nonnegative numbers
$\mathbb{R}^m$	set of real column vectors of dimension m
$\mathbb{R}^{m\times n}$	set of real matrices of dimension $m \times n$
x_i or $x(i)$	ith entry of the vector x
$x(K)$	subvector x with indices in the set K
$x \geq 0$	the vector x is entry-wise nonnegative
$A \geq 0$	the matrix A is entry-wise nonnegative
$A(i,:)$	ith row of the matrix A
$A(:,j)$	jth column of the matrix A
$A(i,j)$	entry of the matrix A indexed by (i,j)
$A(:,J)$	submatrix of A with column indices in the set J
$A^\top$	transpose of the matrix A
$\operatorname{conv}(A)$	convex hull of the columns of matrix A, $\{y \mid y = Az, z \geq 0, e^\top z = 1\}$
$\kappa(A)$	condition number of the matrix A
$\lvert S\rvert$	Cardinality of the set S, that is number of elements in S
$\lVert x\rVert_0$	ℓ_0-"norm" of vector x, $\lvert\{i : x_i \neq 0\}\rvert$
$\lVert x\rVert_1$	ℓ_1-norm of vector $x \in \mathbb{R}^r$, $\sum_{i=1}^r \lvert x_i\rvert$
$\lVert x\rVert_2$	ℓ_2-norm of vector $x \in \mathbb{R}^r$, $\sqrt{\sum_{i=1}^r \lvert x_i\rvert^2}$
$\lVert A\rVert_F$	Frobenius norm of matrix $A \in \mathbb{R}^{m\times r}$, $\sqrt{\sum_{i=1}^m \sum_{j=1}^r A(i,j)^2}$

Acronyms

ALLS	algorithm to learn a latent simplex
HALS	hierarchical alternating least squares
HU	hyperspectral unmixing
KKT	Karush-Kuhn-Tucker
LS	least squares
MIQP	mixed-integer quadratic program
NMF	nonnegative matrix factorization
NNLS	nonnegative least squares
NNOLS	nonnegative orthogonal least squares
NNOMP	nonnegative orthogonal matching pursuit
OLS	orthogonal least squares
OMP	orthogonal matching pursuit
SNPA	successive nonnegative projection algorithm
SPA	successive projection algorithm
VCA	vertex component analysis

List of Figures

List of Tables

Chapter 1

Introduction

"I want us to make this trip a spiritual journey, where each of us seek the unknown and we learn about it. Can we agree to that?"

– *The Darjeeling Limited*, written by Wes Anderson

The main objective of data science is to extract useful knowledge and meaningful information from data. To do so, many techniques use mathematical models representing some kind of underlying structure in the data. These models sometimes represent accurately some reality, for example the physical properties of a system. However, even when they do not, they are often useful to extract some latent knowledge. To refine these models, to improve their fitting to a physical model, to produce more useful solutions, or to ease the practical solving, we often use *a priori* assumptions about the data and its underlying structures. Understanding and leveraging these assumptions to build better models is the key motivation of this doctoral study.

The class of models we focus on are *linear models*, where we assume the data is generated from linear interactions of underlying information. Our focus is also on two a priori assumptions on the weights of the interactions: *nonnegativity*, where we assume the linear interactions are only additive, and *sparsity*, where we assume each observed data point is the outcome of only a few interactions. We study the impact of these assumptions on the properties of some data models, with a particular attention to the algorithmic aspects. Specifically, we consider *least square problems* and *matrix factorizations* that we detail further in this chapter.

In this introductory chapter, we lay the foundations for the models considered throughout the thesis. We review existing works related to constraints of sparsity and nonnegativity in least squares problems and matrix factorizations. We do not aim for an exhaustive description, but rather a high-level overview including a few fundamentals and some useful pointers. The organization of the thesis is detailed at the end of this introduction, in Section 1.9 (page 29).

1.1 Constrained least squares problems

Signal processing and data mining tasks often rely on linear models, where a signal or information vector $x \in \mathbb{R}^r$ is estimated from a set of measures or observations $b \in \mathbb{R}^m$, through a relation of the form $Ax = b$, with $A \in \mathbb{R}^{m \times r}$. Here A is a coefficient matrix, often called dictionary; it represents a set of basis vectors, called features, atoms, or components. We consider here the overdetermined case where there are more observations than features, that is, we have $m > r$.

If A is known, the estimation of x can be expressed as the following optimization problem,

$$\min_x \|Ax - b\|_2^2. \tag{1.1}$$

The objective function, also called the data-fitting term, is here the squared ℓ_2-norm defined as $\|x\|_2^2 = \sum_{i=1}^r |x_i|^2$, corresponding to a *least squares* (LS) problem. Other objective functions exist and the choice of the most appropriate one for a particular application is a research question on its own. In this thesis, we stick to the ℓ_2-norm. It is the most widely used and it is adapted to an assumption of Gaussian noise, which is realistic in the applications we focus on. This problem is generally ill-posed, in the sense that the solution is non-unique or unstable. It is hard to solve when the available data is noisy, which is typically the case in real-world applications.

To account for noise and model errors and make the recovery of the original solution easier, one can benefit from *a priori* knowledge on the structure of the solution vector x. This knowledge can be expressed as constraints imposed on x during the optimization process.

For instance, nonnegative representations imply that a data point is generated from an additive combination of components [48]. This constraint is natural for many applications, for instance,

- In facial images, the faces are the nonnegative linear combination of facial features such as eyes, noses and lips [47].
- In hyperspectral images, the spectral signature of a pixel is the nonnegative linear combination of the spectral signature of the materials it contains [13], see Section 1.6 for details.

The resulting *nonnegative least squares* (NNLS) problem can be formulated as:

$$\min_x \|Ax - b\|_2^2 \quad \text{such that} \quad x \geq 0, \tag{1.2}$$

where $x \geq 0$ means x is entry-wise nonnegative.

Another example is sparsity. It consists in assuming a data point can be adequately represented using only a few atoms. It helps producing more interpretable solutions. As a constraint, sparsity means the solution vector x is required to have only a few nonzero entries. While nonnegativity is known to naturally induce sparsity [25], there is no guarantee on the sparsity of the solution of a general NNLS problem. Therefore, applications where sparsity is an important feature may benefit from explicit sparsity constraints.

The most natural sparsity measure is the ℓ_0-"norm", as it is equal to the number of nonzero elements of a vector, $\|x\|_0 = |\{i : x_i \neq 0\}|$. Note that, although often called so, ℓ_0 is not a proper norm, nor a pseudo-norm or quasi-norm, because it lacks the homogeneity property. The notation with quotation marks (ℓ_0-"norm") seems widely accepted[1]. Therefore, we can define a sparsity constraint with a maximum number of nonzero entries, k. Combined with nonnegativity, this leads to the following formulation, called *k-sparse NNLS*,

$$\min_{x \geq 0} \|Ax - b\|_2^2 \quad \text{such that} \quad \|x\|_0 \leq k. \tag{1.3}$$

In hyperspectral unmixing, this k-sparsity constraint implies that a pixel can be composed of at most k materials. We review in Section 1.2 the existing methods to solve Problem (1.3).

Although this formulation is intuitive, in some cases setting the parameter k is not straightforward. Therefore, we can also consider a *biobjective* formulation, where the objectives are on the one hand to minimize the reconstruction error, and on the other hand to maximize the sparsity (that is, minimize the ℓ_0-"norm"),

$$\min_{x \geq 0} \{\|Ax - b\|_2^2, \|x\|_0\}. \tag{1.4}$$

Sparser solutions typically lead to a higher error, and hence these objectives are conflicting, so in general there is not one optimal solution to Problem (1.4). We need a trade-off between the two objectives, thus we seek *Pareto-optimal* solutions.

Given different objectives to optimize, a solution x is said Pareto-optimal if there does not exist any solution which is at least as good as x on all objectives and strictly better than x on at least one objective. The set of all Pareto-optimal solutions for a given problem is called the *Pareto front*, see Fig. 1.1. Here, the discreteness of the ℓ_0-"norm" implies that solving Problem (1.4) conceptually reduces to solving Problem (1.3) for all possible values of k.

Another possible formulation for sparse NNLS is the minimization of the ℓ_0-"norm" with a constraint on the reconstruction error,

$$\min_{x \geq 0} \|x\|_0 \quad \text{such that} \quad \|Ax - b\|_2^2 \leq \epsilon, \tag{1.5}$$

for a given $\epsilon \in \mathbb{R}_+$. We do not focus on this formulation in this work, because in the applications we consider it is usually easier and more intuitive to determine a sparsity threshold k rather than an error threshold ϵ.

1.2 k-sparse NNLS

Because of the discreteness of the ℓ_0-"norm", the k-sparse NNLS problem (1.3) is combinatorial in nature, and NP-hard [64]. Indeed, it implies to find the optimal support (that is, the set of the indices of the non-zero entries) for x among the $\binom{r}{k}$ possible supports.

[1] Orally, it is usually pronounced "norm" while mimicking brackets with both hands.

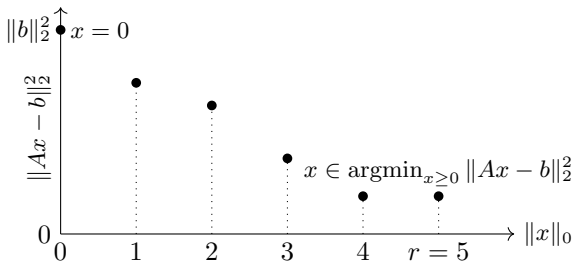


Figure 1.1: Example of the Pareto front for a biobjective k-sparse NNLS problem with $r = 5$ variables. The first solution, for $\|x\|_0 = 0$, corresponds to the zero vector. The last solution, for $\|x\|_0 = 5$, corresponds to the NNLS problem with no sparsity constraint. Here the penultimate solution is identical to the last one, meaning that the solution with no sparsity constraint has naturally 1 zero entry.

Many works were proposed for solving globally the ℓ_0-constrained least squares problem, although not specifically with nonnegativity constraints. Notably, Bienstock proposed in 1996 [12] a reformulation of Problem (1.3) into a mixed-integer program (MIP), and a branch-and-cut algorithm to solve it, using continuous relaxation at every iteration. Several contributions extended this approach, for example under different constraints, or using new methods to solve the continuous relaxation subproblem; see for example [9, 15] and the references therein. Ben Mhenni et al. [8] further extended it, using a homotopy algorithm to compute the relaxations. Although it may be possible to adapt these algorithms to the nonnegative setting, it has not been done yet, to the best of our knowledge.

For this reason, most existing works to solve sparse NNLS problems rely on approximate approaches, inexact in general but computationally efficient. They are often based on existing sparse LS methods, adapted to take on board the nonnegativity constraint. They mostly fall into two categories: greedy algorithms (discussed below), and ℓ_1-regularized methods (discussed in Section 1.3).

Greedy algorithms encompass methods that start with an empty support ($x_i = 0$ for all i), and select components one by one to enrich the support until the target sparsity k is reached. The selection of a component is done greedily, by choosing the component maximizing the decrease of the residual error. Orthogonal versions of these methods, such as *Orthogonal Least Squares* (OLS) [17] and *Orthogonal Matching Pursuit* (OMP) [68] ensure a component can be selected only once. Nonnegative variants have been proposed and successfully used; see for example [65] and the references therein. There exist theoretical conditions that guarantee the recovery of the optimal solution, but they are unfortunately restrictive in practice [73].

Another family of sparse optimization methods avoid the issues linked with the ℓ_0-"norm" by using ℓ_1*-relaxation*.

1.3 ℓ_1-relaxation

A convex relaxation of the ℓ_0-"norm" is the ℓ_1-norm defined as $\|x\|_1 = \sum_{i=1}^{r} |x_i|$. Formulations based on the ℓ_1-norm are convex problems, and as such they are easier to optimize and they benefit from the generic algorithms and strong theoretical results from convex optimization. Problems based on the ℓ_1-norm include the minimization of the reconstruction error with a constraint of the ℓ_1-norm of the solution, the minimization of the ℓ_1-norm with a constraint on the error, or the minimization of a weighted-sum form. The latter can be written as the following problem:

$$\min_x \|Ax - b\|_2^2 + \lambda \|x\|_1 \quad \text{such that} \quad x \geq 0, \tag{1.6}$$

where λ is a given nonnegative constant controlling the tradeoff between error and sparsity. This is equivalent to the LASSO [77] model with a nonnegativity constraint. Problems of this form are usually solved with iterative algorithms, such as coordinate descent. However, these methods have several drawbacks. First, they rely on the parameter λ that has no explicit meaning, and depends a lot on the application and the data properties. Setting a value to reach a given sparsity is not straightforward, and generally implies time-consuming trial and error. Second, they introduce a bias in the objective function, in the sense that the regularization term not only enforces sparsity but also shifts the objective function away from the original objective. Finally, although there exist theoretical guarantees for support recovery, such as the *Exact Recovery Condition* [78], their conditions are restrictive and often not realistic in practice. As a rule of thumb, ℓ_1-based methods tend to work well when the columns of A are not too correlated.

1.4 Least squares problems with multiple right-hand sides

In many applications, we have to deal with NNLS problems with multiple right-hand sides (MNNLS), that is, problems of the form

$$\min_X \|B - AX\|_F^2 \quad \text{such that} \quad X \geq 0, \tag{1.7}$$

where $B \in \mathbb{R}^{m \times n}$, $A \in \mathbb{R}^{m \times r}$, and $X \in \mathbb{R}^{r \times n}$. Note that many objective functions measuring the error between B and AX can be used, but in this thesis we focus on the squared Frobenius norm defined as $\sum_{i=1}^{m} \sum_{j=1}^{n} M(i,j)^2$ for a matrix $M \in \mathbb{R}^{m \times n}$. It is the most widely used in practice and it corresponds to an assumption of Gaussian noise, which is adequate in the applications we focus on.

We note $B(:,j)$ the j-th column of the matrix B. Problem (1.7) can be decomposed into n NNLS subproblems of the form (1.2), where $B(:,j)$ and $X(:,j)$ correspond respectively to b and x. For example, in the unmixing of a hyperspectral image, every column $B(:,j)$ represents a pixel (more precisely, the spectrum at a given pixel location), and the corresponding column $X(:,j)$ represents its composition, in terms of the abundances of the r materials whose spectral signatures are the columns of A.

Note that in (1.7), for the sake of conciseness, we focus only on the optimization of X, but all concepts and algorithms can be applied symmetrically on A.

To encourage sparsity in MNNLS, one can apply a sparse NNLS model column-wise, leading to

$$\min_X \|B - AX\|_F^2 \quad \text{such that} \quad X \geq 0 \text{ and } \|X(:,j)\|_0 \leq k \text{ for all } j. \tag{1.8}$$

Solving Problem (1.8) boils down to solving n independent subproblems of the form (1.3).

1.5 Nonnegative matrix factorization

The MNNLS problem is related to the nonnegative matrix factorization (NMF) problem, of the form

$$\min_{A,X} \|B - AX\|_F^2 \quad \text{such that} \quad A \geq 0 \text{ and } X \geq 0, \tag{1.9}$$

in which we aim at finding both the factors A and X, given B and a factorization rank r that is usually small compared to m and n. As explained in the previous section, many objective functions exist but in this thesis we focus on the Frobenius norm. This dimensionality reduction technique is able to identify underlying features in large datasets in an unsupervised manner. Geometrically, the columns of A can be seen as r vertices whose convex hull contains the data points (columns of B), under appropriate scaling.

Since its introduction by Paatero and Tapper in 1994 [67], and its formalization by Lee and Seung in 1999 [47], NMF has been successfully extended and applied in a variety of problems; see for example [29] and the references therein. The main advantage of NMF over similar techniques such as PCA is the production of interpretable factors, leading to a part-based representation, where every data point is expressed as an additive combination of atoms.

In general, if the rank r is part of the input, NMF is NP-hard [80]. However, if one of the factors is fixed, then the optimization of the remaining one is a convex problem of the form (1.7). The usual optimization scheme for NMF consists in alternatively optimizing A and X, by optimizing one factor while fixing the other, which is equivalent to solving MNNLS subproblems. Moreover, the optimization of one factor can be decomposed in a series of NNLS subproblems of the form (1.2). For instance, if A is fixed, then the computation of X is equivalent to solve n NNLS subproblems with, for all $j \in \{1, \ldots, n\}$, $B(:,j) = b$, and $X(:,j) = x$.

One of the features that made NMF popular, since its very introduction [47], is the natural sparsity of the produced factors. However, it is not guaranteed, and there is no easy way to control this sparsity. As many applications can benefit from an increased or explicitly required sparsity, see [43] and the references therein, it is useful to add sparsity constraints to the NMF model. Several variants of the NMF model have been introduced to take sparsity into account, and we review them in Section 1.7.

The applications of NMF are numerous, and we can cite as examples topic modeling, recommender systems, clustering, chemometrics, audio source separation, and so on; see [29] and the references therein. The models and algorithms considered in this thesis are generic and could theoretically be applied in most of these applications. However, they are mainly motivated by feature extraction in images, and in particular by hyperspectral unmixing, a remote sensing problem discussed in the next section.

1.6 Hyperspectral unmixing

Hyperspectral unmixing is a key problem in remote sensing, and the main motivation for the contributions presented in this thesis. Here we do a quick introduction to this problem, see the survey papers [13, 54] and the references therein for more details.

A hyperspectral image (HSI) is a picture of a scene acquired within a large number of spectral bands, usually between 100 and 200, instead of the usual 3 bands of RGB images. Thus, for each pixel a precise electromagnetic spectrum is recorded, which gives an information concerning the materials present in the pixel; specifically, about their reflectances (fraction of incoming light they reflect) and/or their emissivity (which is due to the fact that the materials usually have non-zero temperatures). These materials are also called *endmembers*. Unfortunately, despite their high spectral resolution, hyperspectral sensors generally have a low spatial resolution; as such, the spectrum recorded for each pixel might not correspond to the one of a single material, but rather to a mixture of the spectra of the different materials present within the pixel.

Given an HSI, *hyperspectral unmixing* (HU) thus aims to recover the abundances of each material in each pixel. If the materials present in the image are also unknown, the task of identifying them along the corresponding abundances is called *blind HU*. The standard model used to solve (blind) HU is the linear mixing model. It assumes that the spectral signature of each pixel is a linear combination of the spectral signatures of the endmembers, where the weights of the linear combination are the abundances of the endmembers in the pixel. Typically, we represent a hyperspectral image as a matrix B, where the jth column, $B(:, j)$, corresponds to the spectral signature of the jth pixel in the scene. If the spectral signatures of the endmembers are also collected as the columns of a matrix A, then according to the linear mixing model, the jth pixel can be written as $B(:, j) \approx \sum_{k=1}^{r} A(:, k)X(k, j) + N(:, j)$, where $X(k, j)$ is the abundance of the kth endmember in the jth pixel, and $N(:, j)$ represents the noise and model misfit. This is exactly the setup of Problems (1.7) and (1.9). See Fig. 1.2 for an illustration.

It is worth noting that by modelling a hyperspectral image as a matrix, we cannot use directly the spatial information of this image (relative position of pixels). To account for this information, either the image should be modeled as a 3-way tensor, or we should use appropriate regularization such as total variation regularization. This calls for other models and techniques that are out of the scope of the present work. However, we will see that even without spatial information, techniques based on least squares solving and matrix factorization are able to extract meaningful knowledge from hyperspectral images.

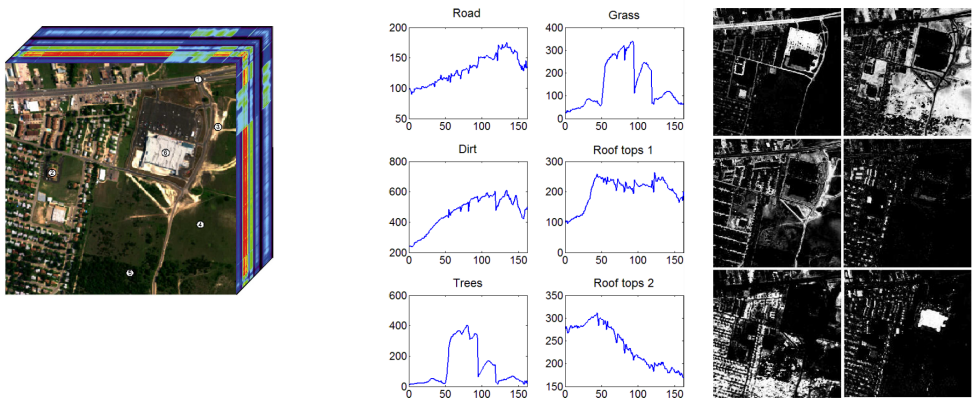


Figure 1.2: Illustration of the linear mixing model. Figure adapted from J. Bioucas Dias and N. Gillis [33].

1.7 Sparse NMF

Among the numerous variants of NMF, sparse NMF is arguably one of the most popular. In this section, we review the most important methods designed to favor sparsity in NMF problems.

In 2002, Hoyer [39] introduces *nonnegative sparse coding*, the first method to enhance sparsity for NMF. It considers an ℓ_1 regularization on X, leading to the following penalized problem:

$$\min_{A\geq 0, X\geq 0} \|B - AX\|_F^2 + \lambda \|X\|_1, \tag{1.10}$$

where λ is a given parameter, controlling the tradeoff between the reconstruction error and the ℓ_1-sparsity. Note that, because of the nonnegativity constraint, $\|X\|_1 = \sum_{i,j} X_{i,j}$. The algorithm introduced to solve (1.10) locally is based on a penalized multiplicative update (MU) for X, and a projected gradient descent for A. As $\|X\|_1$ can always be minimized without changing the reconstruction error by scaling down X and scaling up A, the optimization of (1.10) would make A grow and X tend to 0. To avoid that, at every iteration, A is rescaled so that all of its columns have unit norm.

Eggert and Korner presented in 2004 [23] a new algorithm for the same model based on a modified MU rule that implicitly normalizes the columns of A to have unit norm. Leplat et al. introduced in 2021 [49] a framework for multiplicative updates to tackle β-divergence NMF with different penalties, including a sparse variant with an ℓ_1-penalty on X and the normalization of the columns of A.

This model has several drawbacks. Particularly, the parameter λ depends on the application and the dataset, and it can be hard to tune to reach a specific sparsity. It also introduces a bias that can reduce the quality of the factorization.

To address these issues, Hoyer introduces *NMF with sparseness constraints* in 2004 [38]. This model considers a more explicit metric for sparsity, based on the ratio between the ℓ_1-norm and the ℓ_2-norm,

$$\text{spar}(x) = \frac{\sqrt{r} - \dfrac{\|x\|_1}{\|x\|_2}}{\sqrt{r} - 1},$$

where r is the dimension of x. Indeed, $1 \leq \|x\|_1/\|x\|_2 \leq \sqrt{r}$, so $\text{spar}(x)$ equals 0 when all entries of x are non-zero and have the same value, and 1 when only one entry is non-zero. This sparsity constraint is enforced with a projection on the constraint space, on either the columns of A or the rows of X, after every iteration of the optimization with a standard gradient descent.

Kim and Park presented in 2007 [43] an Alternating Nonnegative Least Squares (ANLS) algorithm for a variant of the ℓ_1-penalized problem. This variant considers a column-wise sparsity of X (respectively, a row-wise sparsity of A) of the form $\lambda\|X(:,j)\|_1^2$ (respectively, $\lambda\|A(i,:)\|_1^2$). The key of this variant is the square on the ℓ_1-norm term. Indeed, as opposed to $\|x\|_1$, $\|x\|_1^2$ is differentiable at 0 at each coordinate, which allows a direct optimization with ANLS, with no need for projection or normalization.

Cichocki et al. introduced in 2008 [18] a novel approach, based on alpha and beta divergence, and a variant of the algorithm hierarchical alternating least squares (HALS) [19].

In 2012, Gillis [31] proposed a modified version of Sparse HALS, where the λ parameter is automatically tuned during the iterations to reach a given sparsity rate (defined as the proportion of entries smaller than 10^{-6} times the larger entry).

Rapin et al. proposed in 2013 [71] a nonnegative variant of generalized morphological component analysis (GMCA). This algorithm solves Problem (1.10) using proximal calculus to solve exactly the subproblems of updating A and X. A soft-thresholding is done to enforce sparsity. The threshold λ is first set to a high value, and then decreased iteratively; this process is a key in GMCA and helps to reduce noise contamination. It also renormalizes the columns of A to unit norm. This algorithm is shown robust to noise.

To avoid the issues inherent to ℓ_1 regularization, various works considered a model of sparse NMF with a constraint on the ℓ_0-"norm" of the columns of X (or A, more rarely). Here we focus on the constraint on the columns of X for simplicity, but the approaches for A are essentially the same. This model strength is the intuitive and explicit nature of the ℓ_0 constraint that represents the maximum number of basis elements a data point can be composed of. Given a number $k \in \mathbb{N}$, this problem can be formalized as follows:

$$\min_{A,X} \|B - AX\|_F \quad \text{s.t.} \quad A \geq 0, X \geq 0, \text{ and } \|X(:,j)\|_0 \leq k \text{ for all } j. \tag{1.11}$$

Therefore, the update of X can be seen as a series of nonnegative sparse coding subproblems.

In 2006, Aharon et al. introduced *K-SVD* [2]. Its nonnegative variant *NNK-SVD* [3] is equivalent to (1.11). The update of X is done with an approximate nonnegative basis pursuit, and the update of A with a SVD approximation where negative entries are set to 0.

Morup et al. also proposed in 2008 an algorithm [55] to solve (1.11). The update of X is done with a nonnegative variant of least angle regression and selection (LARS). The update of A is done with the same self-normalizing MU rule as [23].

Peharz and Pernkopf proposed in 2012 [69] a new algorithm for (1.11). Their approach is based on a nonnegative variant of orthogonal matching pursuit (OMP), a greedy algorithm for sparse coding (see Section 1.2 for details).

Gong et al. proposed in 2018 [37] an original way to reformulate (1.11). Instead of imposing a constraint on the ℓ_0-"norm", they consider a biobjective problem, where the objectives are minimizing the reconstruction error and minimizing the ℓ_0-"norm". This problem is split up column-wisely into a series of biobjective subproblems that are solved approximately using a multi-objective evolutionary algorithm. The output of this approach is a set of solutions (pairs A_k, X_k) forming an approximation of a Pareto front. The main advantage of this method is that the analyst does not have to define a priori an arbitrary parameter or threshold.

In 2019, Cohen and Gillis [20] proposed a method to solve *exactly* the nonnegative sparse coding subproblems, using a bruteforce approach. The key contribution of this work is not the well-known bruteforce approach, but rather its use in an alternate optimization scheme for NMF, where it significantly improves the quality of the results.

1.8 Separable NMF

In this section, we review a common variant of NMF called *separable NMF*. It relies on the assumption of *separability* of the input matrix B, defined as follows.

Definition 1.1 *A matrix B is r-separable if there exists a subset of r columns of B, indexed by $\mathcal{J}$, and a nonnegative matrix $X \geq 0$, such that $B = B(:, \mathcal{J})X$.*

Equivalently, B is r-separable if B has the form $B = A[I_r, X']\Pi$, where I_r is the identity matrix of size r, X' is a nonnegative matrix, and Π is a permutation matrix. Separability is equivalent to the *pure-pixel assumption* in hyperspectral unmixing, stating that for each material present in a hyperspectral image there is a least one pixel composed of only this material.

Separable NMF consists in selecting the right r columns of B such that B can be reconstructed perfectly. In other words, it consists in finding the atoms (columns of A) *among* the data points (columns of B). It is defined as follows.

Problem 1.1 (Separable NMF) *Given a r-separable matrix B, find $A = B(:, \mathcal{J})$ with $|\mathcal{J}| = r$ and $X \geq 0$ such that $B = AX$.*

Note that, if A is known, the computation of X is straightforward: it is a convex problem that can be solved using any nonnegative least squares (NNLS) solver[2]. However, the solution is not necessarily unique, unless A is full column-rank.

In general, NMF is NP-hard [80]. However, Arora et al. [6] proved that NMF is solvable in polynomial time under the separability assumption. In the presence of noise, which is typically the case in real-life applications, this problem is called near-separable NMF and is also solvable in polynomial time given that the noise level is sufficiently small [6]. In this case, we are given a near-separable matrix $B \approx B(:, \mathcal{J})X$ where $|\mathcal{J}| = r$ and $X \geq 0$.

The early algorithms building on this assumption emerged in the blind HU community. They include pure-pixel index (PPI) in 1995 [14], N-FINDR in 1999 [81], and vertex component analysis (VCA) in 2005 [63]. Most of these algorithms were developed based on convex geometry concepts. These early works however did not analyze noise robustness, and in fact they are not guaranteed to recover the endmembers in the presence of noise.

In analytical chemistry, Problem (2.1) is closely related to the problem of self-modeling curve resolution [41]. As in blind HU, several algorithms were developed based on geometry concepts; in particular the successive projection algorithm (SPA) [4].

More recently, and motivated by applications in machine learning (in particular, topic modeling where pure data points are referred to as anchor words), Arora et al. [6] introduced the first provably robust separable NMF algorithms. Their robustness is deterministic: under some conditions, their algorithm is guaranteed to recover an approximation of the column of A. Arora et al. were not aware of the algorithms developed within the blind HU literature. Many provably robust algorithms have followed this seminal paper, including algorithms that use linear programming [30, 35, 72], a generalization of SPA [36], fast anchor words [5], and the successive nonnegative projection algorithm (SNPA) [32]. These deterministically robust algorithms guarantee that, in the presence of noise, the vertices are recovered, up to some error bounds that depend on the noise level and the conditioning of A; see Section 2.2.1 for such a result for SPA. We refer the interested reader to [29, Chapter 7] for a detailed discussion and comparison of these algorithms.

1.9 Contributions and outline of the thesis

The starting point of this thesis is NMF, and how to leverage sparsity in this model. Let us remind the NMF problem; note that as discussed in Section 1.5 we focus on the variant using the Frobenius norm as objective function. Given $B \in \mathbb{R}^{m \times n}$ and $r \in \mathbb{N}$, NMF consists in finding $A \in \mathbb{R}^{m \times r}$ and $X \in \mathbb{R}^{r \times n}$ to solve

$$\min_{A,X} \|B - AX\|_F^2 \quad \text{such that} \quad A \geq 0 \text{ and } X \geq 0.$$

It led us to explore different research directions, some motivated by a practical need in a specific application, others by curiosity about the behavior of a model under some

[2]For example, it can be solved with the Matlab function `lsqnonneg`.

assumptions. The contributions of this thesis focus mainly on sparse models involving the ℓ_0-"norm". This non-convex constraint brings interesting properties to these models, both theoretically and in applications. The resulting optimization problems are combinatorial and hard to solve, and the development of exact algorithms to tackle them is an important part of this work. Another focus of this thesis is the separability assumption in matrix factorizations, and two chapters describe novel variants of the separable NMF model along with practical algorithms.

Let us present the contributions of this thesis and the motivations behind them; see Fig. 1.3 for a graphical overview.

- Separable NMF is a very popular model that shows good performance in many practical cases. However, existing separable NMF algorithms do not leverage the fact that, when the separability assumption holds, then there are generally multiple pure data points for each vertex instead of just one (only one algorithm called ALLS leverages this fact, and it has some practical limitations). This stronger assumption holds in many real-world datasets, notably hyperspectral images, and we can take advantage of it to design algorithms that are more robust to noise and outliers.

$\longrightarrow$ In Chapter 2, we propose two new algorithms for *smoothed separable NMF*, which is a variant of the separable NMF model where we assume that, when the separability assumption holds, then for each vertex there are more than one data point composed purely of this vertex.

- k-sparse NNLS problems are present in many tasks of machine learning and signal processing, but existing algorithms generally rely on relaxation or heuristics with limited guarantees and we lack a way to solve these problems exactly with a reasonable computing time. k-sparse NNLS problems are highly-structured combinatorial problems, so we can design an exact algorithm that leverages this structure to prune the search space.

$\longrightarrow$ In Chapter 3, we propose a *branch-and-bound algorithm* to solve exactly the k-sparse NNLS (1.3).

- In k-sparse NNLS problems, the parameter k is sometimes hard to set a priori. To avoid a tedious trial-and-error process, it would be more convenient to consider the biobjective sparse NNLS problem and compute a Pareto front with solutions representing different tradeoffs between error and sparsity. An algorithm capable of this would be useful both on its own and as a subroutine in more complex problems.

$\longrightarrow$ In Chapter 3, we also extend our branch-and-bound algorithm to solve exactly the biobjective k-sparse NNLS problem (1.4).

- In MNNLS problems, the column-wise sparsity constraint is not adequate when the sparsity varies between columns. This is notably the case in hyperspectral unmixing when pixels are composed of different numbers of materials. Existing

matrix-wise methods rely on relaxations or heuristics and have limited guarantees. There is a need for an algorithm with explicit matrix-wise sparsity constraints and optimality guarantees.

$\longrightarrow$ In Chapter 4, we propose an MNNLS model with a *matrix-wise ℓ_0-constraint*, along with an algorithm that finds an optimal solution in most cases.

- No existing separable NMF algorithm leverages the sparsity assumption whereas it often holds in practice. Moreover, no existing algorithm handles the case when a vertex (column of A) is an additive linear combination of other vertices. This can happen for example in multispectral unmixing when the number of materials present in the image exceeds the number of spectral bands of the image. Assuming a degree of sparsity of X can make the solution identifiable and allow the design of algorithms to recover it.

$\longrightarrow$ In Chapter 5, we propose *sparse separable NMF*, a new NMF variant that combines the assumptions of separability and column-wise k-sparsity, along with theoretical analysis and an algorithm to solve it.

Chapter 6 concludes this thesis and draws some perspectives for future research. Note that Chapters 4 and 5 depend on Chapter 3.

Related publications Chapters 2 to 5 are original contributions, that appear in the following publications and preprints:

61. Nadisic, N., Vandaele, A., Gillis, N. & Cohen, J. E. *Exact sparse nonnegative least squares* in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020), 5395–5399.

60. Nadisic, N., Vandaele, A., Gillis, N. & Cohen, J. E. *Exact biobjective k-sparse nonnegative least squares* in *29th European Signal Processing Conference (EUSIPCO)* (2021), 2079–2083.

59. Nadisic, N., Vandaele, A., Cohen, J. E. & Gillis, N. *Sparse separable nonnegative matrix factorization* in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECMLPKDD)* (2020), 335–350.

58. Nadisic, N., Gillis, N. & Kervazo, C. Smoothed separable nonnegative matrix factorization. *preprint arXiv:2110.05528* (2021).

57. Nadisic, N., Cohen, J. E., Vandaele, A. & Gillis, N. Matrix-wise ℓ_0-constrained sparse nonnegative least squares. *preprint arXiv:2011.11066* (2022).

Open-source codes All the algorithms developed in this thesis are available online along with the data and test scripts necessary to reproduce our experiments:
`http://nicolasnadisic.xyz/code/`.
Moreover, we developed a toolbox in Julia [10] called Giant.jl. It is still a work in progress at the time of writing, but it features several standard methods for (sparse) NNLS and NMF as well as most of the algorithms proposed in this thesis:
`https://gitlab.com/nnadisic/giant.jl`.

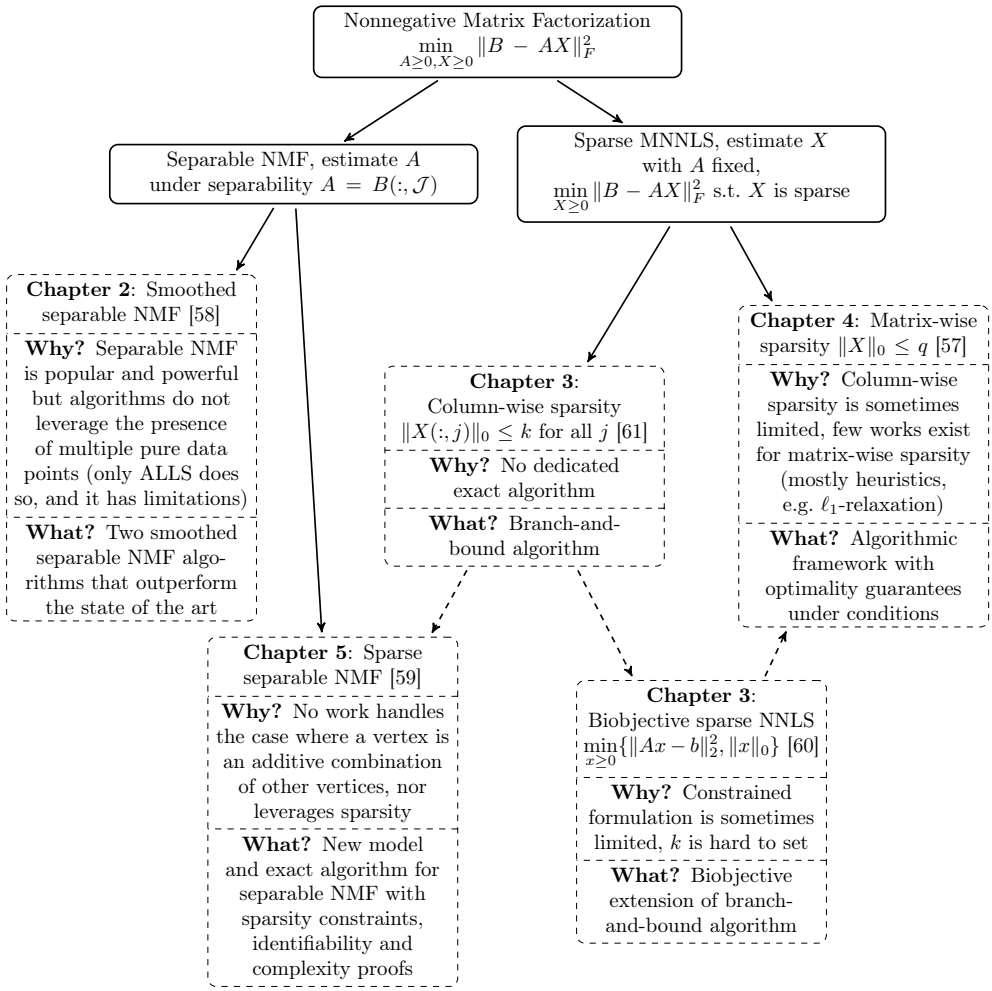


Figure 1.3: Graphical overview of the contents of the thesis. Dotted boxes correspond to our contributions. Dotted arrows denote dependencies between contributions.

Chapter 2

Smoothed Separable NMF

> *"De temps en temps on tombe sur un farfelu qui croit qu'il a inventé l'eau chaude, mais le plus souvent c'est une adaptation d'un modèle existant."*
>
> – *Kaamelott*, written by Alexandre Astier

In this chapter, we study a novel variant of near-separable NMF. Many NMF algorithms have been developed under the (near-)separability assumption, stating that there is at least one nearby data point to each vertex; two of the most widely used ones are vertex component analysis (VCA) and the successive projection algorithm (SPA). This assumption is known as the pure-pixel assumption in blind hyperspectral unmixing. More recently, Bhattacharyya and Kannan (ACM-SIAM Symposium on Discrete Algorithms, 2020) proposed an algorithm for learning a latent simplex (ALLS) that relies on the assumption that there is more than one nearby data point to each vertex. In that scenario, ALLS is probabilistically more robust to noise than algorithms based on the separability assumption. In this work, inspired by ALLS, we propose smoothed VCA (SVCA) and smoothed SPA (SSPA) that generalize VCA and SPA by assuming the presence of several nearby data points to each vertex. We illustrate the effectiveness of SVCA and SSPA over VCA, SPA and ALLS on synthetic data sets, and on the unmixing of hyperspectral images. In addition, our study highlights new theoretical results for VCA.

The content of this chapter is mainly extracted from [58]:

58. Nadisic, N., Gillis, N. & Kervazo, C. Smoothed separable nonnegative matrix factorization. *preprint arXiv:2110.05528* (2021).

2.1 Introduction

In this chapter, we first introduce a problem that generalizes nonnegative matrix factorization. It is called *simplex-structured matrix factorization* (SSMF), and consists in, given a set of data points within the convex hull of a set of vertices, estimating these vertices in the presence of noise. This problem can be formulated as follows.

Problem 2.1 *Given $B = AX + N \in \mathbb{R}^{m\times n}$ where $X \in \mathbb{R}_+^{r\times n}$ is column stochastic and N is the noise, estimate $A \in \mathbb{R}^{m\times r}$.*

X being column-stochastic means that, for all j, $\|X(:,j)\|_1 = 1$. Note that A, X and N are unknown, only B is given. Once A is estimated, X can be estimated for example using a nonnegative least squares (NNLS) algorithm. For a more detailed study of Problem (2.1), see [1] and the references therein.

In Problem (2.1), the columns of A are the vertices, while the columns of B are noisy data points within the convex hull of the columns of A,

$$\operatorname{conv}(A) = \{y \mid y = Az, z \geq 0, e^\top z = 1\},$$

where e is the vector of all ones of appropriate dimension. See Fig. 2.1 for an illustration.

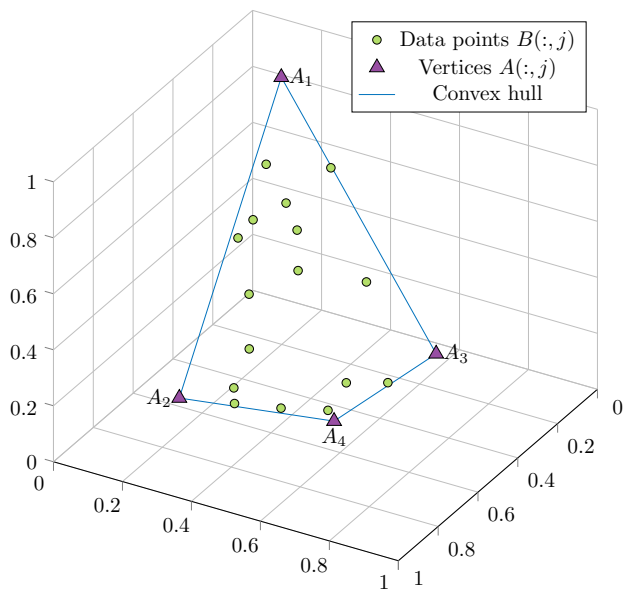


Figure 2.1: Illustration of SSMF in 3 dimensions (for $m = 3$).

In fact, for all j, we have

$$B(:,j) = AX(:,j) + N(:,j),$$

where $X(:,j) \geq 0$ and $e^\top X(:,j) = 1$ (since X is column stochastic), which means that $B(:,j) - N(:,j) \in \operatorname{conv}(A)$ for all j. To be able to estimate A in Problem (2.1), appropriate assumptions on A, X and N are required. In particular, the following assumptions are necessary:

- No column of A is contained in the convex hull of the other columns of A, otherwise it is not possible to distinguish it from a data point.

- The data points must be sufficiently spread within $\text{conv}(A)$, having data points on each facet of $\text{conv}(A)$. This implies some degree of sparsity for X.
- The noise N must be bounded.

To obtain provable or practical algorithms, the above assumptions must be carefully and rigorously complemented. Different assumptions on A, X, and N lead to different models for which different algorithms can be designed; see Section 2.2 for a literature review. We discuss below some applications of such algorithms.

Applications in data analysis and machine learning In [7, 11], the authors describe in details various applications of solving the SSMF problem (2.1); in particular topic modeling via latent Dirichlet allocation, adversarial clustering, and community detection via the mixed membership stochastic block model. Moreover, Problem (2.1) generalizes NMF and as such could be useful for all applications of NMF, such as feature extraction in sets of images, audio source separation, or chemometrics. Although the model considered in this work is very generic, we are motivated mainly by blind hyperspectral unmixing (HU), as described in Section 1.6.

Pure-pixel assumption An important class of algorithms to solve blind HU are *pure-pixel search* algorithms. They rely on the assumption that for each endmember, there is at least one pixel in which this endmember appears almost alone, that is, purely, so that the endmember signature is close to the one of the corresponding pure pixel. Two of the most effective and widely used pure-pixel search algorithms are vertex component analysis (VCA) [63] and the successive projection algorithm (SPA) [4] which will be described in Section 2.2.

Outliers One of the motivations of this chapter is to better handle outliers. In hyperspectral unmixing, an outlier can be defined broadly as a pixel whose spectra is very different from all other pixels, and that we do not consider as reliable data. It may originate from the acquisition process, for example because of a faulty cell in the sensor or electronic interference. It can also correspond to a very rare material present in the scene. In both cases, they can disturb the unmixing, especially when they are out of the convex hull of the actual endmembers.

Terminology In the NMF literature, the pure-pixel assumption is referred to as separability, and the corresponding algorithms are referred to as separable NMF algorithms, or near-separable NMF algorithms; see Section 1.8. Geometrically, the pure-pixel assumption requires that there is a data point (that is, a column of B), close to each vertex (that is, to each column of A). We will refer to such a data point as pure when the corresponding column of X is a unitary vector (that is, a column of the identity matrix). Note that in this chapter we use the terms "separable" and "near-separable" equally to designate the inexact case, that is we assume the presence of noise.

Contribution and outline When the separability assumption holds, there are typically more than one data point close to each vertex. In this work, we leverage this observation by adapting VCA and SPA, providing two new algorithms, namely smoothed VCA (SVCA) and smoothed SPA (SSPA). The idea is to aggregate (for instance average) several data points around a vertex to obtain a better estimate of that vertex. In practice, it enables our two smoothed algorithms to achieve much better separation results than their non-smoothed counterparts, and to be more tolerant to noise.

SVCA can also be interpreted as an adaptation of the algorithm for learning a latent simplex (ALLS) proposed in [11], with two major modifications that we detail in Section 2.3. This observation allows us to provide theoretical guarantees for SVCA, and hence VCA which is a special case of SVCA. To the best of our knowledge, this is the first time a theoretical guarantee for VCA is provided in the presence of noise.

We believe that this work paves the way for a whole new branch of smoothed separable NMF algorithms. Although we focus in this work on VCA and SPA, two of the most well-known separable NMF algorithms, it is straightforward to extend the methodology to more recent algorithms such as the successive nonnegative projection algorithm (SNPA) [32].

The chapter is organized as follows. In Section 2.2, we summarize the literature for solving Problem (2.1), with a focus on three algorithms, namely VCA, SPA and ALLS. In Section 2.3, we propose SVCA which is equivalent to applying VCA on a smoothed data set. SVCA is similar to ALLS, but two key differences make it empirically more efficient than ALLS. In Section 2.4, we propose SSPA which adapts SPA in the presence of multiple pure pixels. In Section 2.5, we show on synthetic and real-world hyperspectral data sets that SVCA and SSPA outperform VCA, SPA and ALLS in the presence of multiple pure pixels. In Section 2.6 we discuss some pratical aspects of SVCA and SSPA and we conclude this chapter.

2.2 Simplex-structured matrix factorization

In this section, we describe existing models and algorithms to solve Problem (2.1), that is, to solve SSMF, in order to identify the vertices of the convex hull of a set of data points, B. We focus on two key models, upon which our contribution is built:

1. Separable NMF: it assumes there is one data point close to each vertex of $\operatorname{conv}(A)$, and that the noise added to each data point is bounded. Some authors refer to this model as near-separable NMF. It is detailed in Section 1.8.

2. Learning a latent simplex: it is motivated by machine learning applications. It assumes that there is more than one data point close to each vertex of $\operatorname{conv}(A)$, but it allows much larger noise levels as it only requires the ℓ_2 norm of N to be bounded, instead of each individual column. This model is detailed in Section 2.2.2.

Other models and algorithms exist to tackle Problem (2.1) relying on different assumptions. It is worth mentioning minimum-volume NMF [26, 51], where A is

regularized such that its convex hull $\text{conv}(A)$ has the smallest possible volume, facet-based identification algorithms that identify the facets of $\text{conv}(A)$ from which its vertices are recovered [1, 28, 50, 52], and probabilistic simplex component analysis [82] that relies on a probabilistic model on the data (the columns of X are sampled using the Dirichlet distribution, and the entries of N using i.i.d. Gaussian noise). Discussing these approaches in detail is out of the scope of this article, whose focus is on the separable NMF problem.

2.2.1 Model 1: Separable NMF

As already mentioned, an important class of NMF algorithms relies on the separability assumption as detailed in Section 1.8. In Problem (2.1), separability amounts to assuming there exists an index set $\mathcal{J}$ of cardinality r such that $X(\mathcal{J},:) = I_r$ where I_r is the r-by-r identity matrix. In the following, we describe in more detail two separable NMF algorithms, VCA and SPA. They will be instrumental in proposing our new algorithms, smoothed VCA in Section 2.3 and smoothed SPA in Section 2.4.

Vertex component analysis

VCA [63] is a greedy separable NMF algorithm, that is, it identifies the indices of the subset $\mathcal{J}$ sequentially. The index set is initialized with $\mathcal{J} = \emptyset$. At each of the r iterations of VCA, a random direction belonging to the subspace spanned by the r top left singular vectors of B is generated (this is equivalent to working with the best rank-r approximation of B, and hence filters the noise). This direction is then projected onto the orthogonal complement of $B(:,\mathcal{J})$, and the index of the column of B that maximizes the absolute value of the inner product with that direction is added to $\mathcal{J}$. Algorithm 2.1 summarizes VCA. We note $\mathcal{N}(0, I_r)$ a random vector of size r generated following the normal distribution of mean 0 and variance 1.

The computational cost of VCA is $\mathcal{O}(r\,\text{nnz}(B))$ operations, where $\text{nnz}(B)$ is the number of non-zero entries of B. The main cost is to compute Y which can be done efficiently using the subspace power iteration, in $\mathcal{O}(\text{nnz}(B)r)$ operations, and to compute the products $(d_k^T P^\perp)B$ at each of the r iterations.

An important drawback of VCA is that it is not guaranteed to be deterministically robust to noise. In other words, for any noise N such that a data point goes outside $\text{conv}(A)$, there is a non-zero probability that VCA extracst this point. The reason is that VCA uses a linear function to identify the vertices of $\text{conv}(A)$; see [29, Chapter 7.4] for a numerical example.

Successive Projection Algorithm

SPA [4] is very similar to VCA. The only difference is in the selection step, when adding an index to $\mathcal{J}$. SPA selects the column of $P^\perp B$ with maximum ℓ_2 norm; see Algorithm 2.2.

It is interesting to note that VCA is equivalent to SPA if the direction u_k randomly chosen at each step is instead taken as the column of the residual $P^\perp B$ with maximum ℓ_2 norm. We will use this observation for our proposed algorithm, smoothed SPA.

Algorithm 2.1: Vertex Component Analysis (VCA) [63]

Input: The matrix $B \in \mathbb{R}^{m \times n}$, the number r of columns to extract.
Output: Index set $\mathcal{J}$ of cardinality r such that $B \approx B(:, \mathcal{J})X$ for some $X \geq 0$.

Let $\mathcal{J} = \emptyset$, $P^{\perp} = I_m$, $V = [\,]$.
Let $Y \in \mathbb{R}^{m \times r}$ be the vector space spanned by the top r left singular vectors of B.
for $k = 1 : r$ **do**
  Pick a random direction $d_k \in \mathbb{R}^m$ in the subspace spanned by Y, e.g., $d_k \sim Y\mathcal{N}(0, I_r)$.
  Compute $u_k = (d_k^T P^{\perp})B \in \mathbb{R}^n$, and let $j_k = \operatorname{argmax}_{1 \leq j \leq n} |u_k(j)|$.
  Let $\mathcal{J} = \mathcal{J} \cup \{j_k\}$.
  Update the projector $P^{\perp}$ onto the orthogonal complement of $A = B(:, \mathcal{J})$:

$$v_k = \frac{P^{\perp} B(:, j_k)}{\|P^{\perp} B(:, j_k)\|_2},$$
$$V = [V\ v_k],$$
$$P^{\perp} \leftarrow \left(I_m - VV^T\right).$$

Remark 2.1 (On the projection $P^{\perp}$) *As opposed to VCA, SPA does not work on the subspace spanned by the first r singular vectors of B, and hence we stick to this variant in this chapter. However, in practice, projecting the data onto this subspace allows noise filtering and typically leads to better numerical performance.*

Robustness of SPA As opposed to VCA, SPA is deterministically robust to noise, provided the following assumption on top of near-separability (Assumption 1.1):

Assumption 2.1 (column-wise bounded noise) *In Problem* (2.1)*, the noise satisfies $\|N(:, j)\|_2 \leq \epsilon$ for all j for some $\epsilon > 0$ sufficiently small.*

Let us state the robustness result for SPA.

Theorem 2.1 *[36, Theorem 3] Let $B = AX + N$ as in Problem* (2.1)*, and let Assumptions 1.1 and 2.1 be satisfied, that is, $A = B(:, \mathcal{J}^*)$ for some index set $\mathcal{J}^*$ of cardinality r, and $\|N(:, j)\|_2 \leq \epsilon$ for all j where $\epsilon \leq \mathcal{O}\left(\frac{\sigma_r^3(A)}{\sqrt{r}K(A)^2}\right)$. Let also the rth singular value of A be positive, that is, $\sigma_r(A) > 0$, meaning that A has rank r. Let $\mathcal{J}$ be the index set extracted by SPA. Then there exists a permutation π of $\{1, 2, \ldots, r\}$ such that for all $k = 1, 2, \ldots, r$,*

$$\|B(:, \mathcal{J}(k)) - A(:, \pi(k))\|_2 \ \leq \ \mathcal{O}\left(\frac{\epsilon K(A)^2}{\sigma_r^2(A)}\right),$$

where $\mathcal{J}(k)$ denotes the kth index in $\mathcal{J}$, and $K(A) = \max_j \|A(:, j)\|_2$.

Algorithm 2.2: Successive Projection Algorithm (SPA) [4]

Input: The matrix $B \in \mathbb{R}^{m \times n}$, the number r of columns to extract.
Output: Index set $\mathcal{J}$ of cardinality r such that $B \approx B(:, \mathcal{J})X$ for some $X \geq 0$.

Let $\mathcal{J} = \emptyset$, $P^{\perp} = I_m$, $V = [\,]$.
Let $u_1(j) = \|B(:, j)\|_2^2$ for all j.
for $k = 1 : r$ **do**
  Let $j_k = \operatorname{argmax}_{1 \leq j \leq n} u_k(j)$. (Break ties arbitrarily, if necessary.)
  Let $\mathcal{J} = \mathcal{J} \cup \{j_k\}$.
  Update the projector $P^{\perp}$ onto the orthogonal complement of $A = B(:, \mathcal{J})$:

$$v_k = \frac{P^{\perp} B(:, j_k)}{\|P^{\perp} B(:, j_k)\|_2},$$
$$V = [V\, v_k],$$
$$P^{\perp} \leftarrow \left(I_m - VV^T\right).$$

  Update the squared norms of the columns of $P^{\perp}B$: for all j,

$$u_{k+1}(j) = u_k(j) - v_k^{\top} B(:, j) = \|P^{\perp} B(:, j)\|_2^2.$$

Note that the bounds in Theorem 2.1 are relatively weak: the noise level has to be rather small to guarantee SPA to recover A approximately.

2.2.2 Model 2: Learning a latent simplex

A drawback of separable NMF algorithms, such as VCA and SPA, is that they assume that there is only one data point close to each column of A. Therefore, to estimate A, the column-wise bounded noise assumption (Assumption 2.1) is necessary; see Theorem 2.1. This is a rather strong assumption, often not met in practical situations as typically many data points are affected by large amounts of noise.

Bhattacharyya et al. [11] rather propose to leverage the fact that typically more than one data point are close to each column of A. This assumption, which is stronger than the pure-pixel one (requiring only a single pure-pixel), allows higher noise levels. It is called the *proximate latent points* assumption, and is defined as follows.

Assumption 2.2 (proximate latent points) *In Problem* (2.1)*, there exists r index sets, $\mathcal{J}_k$ for $k = 1, 2, \ldots, r$, of cardinality at least $p = \delta n$ such that*

$$\|AX(:, j) - A(:, k)\|_2 \leq \frac{4\sigma}{\delta} \quad \textit{for all } j \in \mathcal{J}_k,$$

for some $\delta \in \left[\frac{1}{n}, \frac{1}{r}\right]$ and $\sigma > 0$.

Under this assumption, instead of looking for one column of B to represent each vertex, like in VCA and SPA, algorithms should look for p of them and then estimate each vertex as the average of these p data points. We will refer to such algorithms as *smoothed separable NMF algorithms*. The main contribution of this chapter is to propose two new such algorithms; in Sections 2.3 and 2.4.

This assumption is often met in the machine learning applications mentioned in Section 2.1; see the discussions in [7, 11]. For high-resolution HSIs that satisfy the pure-pixel assumption, there are typically more than one pixel close to each endmember. This claim will be validated numerically in Section 2.5.2.

Algorithm to learn a latent simplex

To solve Problem (2.1) under Assumption 2.2, Bhattacharyya and Kannan [11] proposed an algorithm similar to VCA, which we refer to as the algorithm for learning a latent simplex (ALLS). The main difference between ALLS and VCA is the selection step. Instead of picking a single column of B, ALLS averages over p columns for some $p \in \{1, 2, \ldots, \lfloor \frac{n}{r} \rfloor\}$. More precisely, ALLS picks the p columns corresponding to the indices that maximize the absolute value of u_k; see Algorithm 2.3.

The idea behind ALLS is to apply VCA on a smoothed data set. This smoothed data set is made of $\binom{n}{p}$ data points which are the averages of all possible combinations of p data points, that is, p columns of B. Of course, constructing this smoothed data set explicitly is not practical, since $\binom{n}{p}$ grows exponentially. However, by the linearity of the selection step in VCA, this is not necessary: the smoothed data point that maximizes a linear function is the average of the p data points that have the p largest values for that function. Algorithm 2.3 summarizes ALLS.

Note that ALLS with $p = 1$ is equivalent to VCA.

Computational cost The only additional cost of ALLS compared to VCA is to average p columns of B, which requires r times $\mathcal{O}(pm)$ operations, which is negligible since $p \ll n \leq \text{nnz}(B)$.

Probabilistic robustness of ALLS

Let us describe the assumptions needed to prove the probabilistic robustness of ALLS. The condition on A is defined as follows:

Assumption 2.3 (well-separatedness of A) *In Problem* (2.1)*, the matrix A satisfies*

$$\alpha(A) = \frac{\min_{k=1,2,\ldots,r} \min_x \|A(:,k) - A(:,\bar{k})x\|_2}{K(A)} > 0, \tag{2.1}$$

where $\bar{k} = \{1, 2, \ldots, r\} \backslash \{k\}$ and $K(A) = \max_j \|A(:,j)\|_2$.

Assumption 2.3 holds if and only if $\text{rank}(A) = r$, in which case $\text{conv}(A)$ is a simplex, that is, a polytope of dimension $r-1$ with r vertices (hence the name of the algorithm).

The condition on the noise is as follows.

Algorithm 2.3: Algorithm for Learning a Latent Simplex (ALLS) [11]

Input: The matrix $B \in \mathbb{R}^{m \times n}$, the number r of columns of A, the number p of columns of B to be averaged to obtain each column of A.

Output: A matrix A' such that $B \approx A'X$ for some $X \geq 0$.

Let $A' = [\,]$, $P^{\perp} = I_m$, $V = [\,]$.

Let $Y \in \mathbb{R}^{m \times r}$ be the vector space spanned by the top r left singular vectors of B.

for $k = 1 : r$ **do**

  Pick a random direction $d_k \in \mathbb{R}^m$ in the subspace spanned by Y, e.g., $d_k \sim Y\mathcal{N}(0, I_r)$.

  Compute $u_k = \left(d_k^T P^{\perp}\right) B \in \mathbb{R}^n$.

  Let $\mathcal{S}_k$ be the set of p indices corresponding to the largest coordinates of u_k in absolute value.

  Let $A'(:,k)$ average of the columns of $B(:,\mathcal{S}_k)$.

  Update the projector $P^{\perp}$ onto the orthogonal complement of A':

$$v_k = \frac{P^{\perp} A'(:,k)}{\|P^{\perp} A'(:,k)\|_2},$$
$$V = [V\, v_k],$$
$$P^{\perp} \leftarrow \left(I_m - VV^T\right).$$

Assumption 2.4 (Spectrally bounded perturbations) *In Problem* (2.1)*,*

$$\|N\|_2 = \sigma_{\max}(N) \;\leq\; \sigma\sqrt{n},$$

where there exists some constant c such that

$$\sigma \;\leq\; \frac{\alpha^2\sqrt{\delta}}{c\, r^9} \min_j \|A(:,j)\|_2, \tag{2.2}$$

where $\alpha = \alpha(A)$ is defined in Assumption 2.3, and δ and σ in Assumption 2.2 (recall, $p = \delta n$ is the number of data points close to each column of A).

It is key to note here that the noise allowed is not column wise as in Assumption 2.1, but on the spectral norm of N, which is rather different.

We can now state the robustness theorem for ALLS.

Theorem 2.2 ([11]) *Let us consider Problem* (2.1) *under Assumptions 2.2 (proximate latent points), 2.3 (well-separatedness of A) and 2.4 (spectrally bounded perturbations). Then, with probability at least $1 - c/r^{3/2}$, ALLS computes a matrix A' such that upon permutation of its columns, for all $k = 1, 2, \ldots, r$,*

$$\|A(:,k) - A'(:,k)||_2 \;\leq\; O\left(\frac{r^4\sigma}{\alpha\sqrt{\delta}}\right). \tag{2.3}$$

Note that substituting (2.2) in (2.3) gives

$$\|A(:,k) - A'(:,k)||_2 \;\leq\; O\left(\frac{\alpha}{c\,r^5}\right) \min_j \|A(:,j)\|_2.$$

Implications for VCA Interestingly, since ALLS for $p = 1$ coincides with VCA, Theorem 2.2 provides a probabilistic robustness result for VCA which is unknown in the blind HU literature.

Bounds of SPA versus ALLS

Theorem 2.2 might look somewhat weak because of the dependence in r^9 in the bound (2.2) for σ. However, it is not known whether this bound is tight, although it is believed it could be improved [7, 11]. A similar comment applies to SPA. Moreover, these bounds assume an adversarial setting, and noise robustness under particular generative models is also an interesting direction of research, as in [82].

In any case, Theorem 2.2 only requires a bound on $\|N\|_2$ while SPA requires each column of the noise matrix N to be bounded, indicating that ALLS should perform better, in general, when p is sufficiently large. Since the theory is still not fully developed and the tightness of the theoretical bounds should be carefully studied, it is important to compare these algorithms empirically to shed light on their differences on practical problems; this will be done in Section 2.5.

2.3 Smoothed VCA

Inspired by VCA, and ALLS, we now propose smoothed VCA (SVCA); see Algorithm 2.4. SVCA has two key important differences compared to ALLS:

1. At step k, ALLS selects the p entries maximizing the absolute value of the vector of u_k, obtained as the inner product of B and a randomly generated direction $d_k^\top P^\perp$; see steps 4-5 of Algorithm 2.3. This is not equivalent to maximizing (or minimizing) the linear function $l(x) = d_k^\top P^\perp x$ over the smoothed polytope. In fact, by using the absolute value, this approach could select data points in opposite directions. For example, take the simple case with two vertices $w_1 = (-1, 0)$ and $w_2 = (1, 0)$. For any direction d, we have $|d^\top w_1| = |d^\top w_2|$ and hence it is very likely that data points close to both vertices will maximize $|d^\top x|$, and their average will be a poor approximation of both vertices.

   Instead, to maximize (or minimize) $l(x)$, one should select the p indices maximizing u_k (or $-u_k$). In SVCA, we therefore propose to select the p indices that maximize (resp. minimize) u_k if the median of the p largest values is larger (resp. smaller) than the absolute value of the median of the p smallest values of u_k. We have observed in practical experiments that this modification of ALLS is crucial to obtain competitive results in real-world hyperspectral images. In fact, we will see that SVCA outperforms ALLS, and the main reason is this modified selection step.

2. Instead of averaging p columns of B at each step, we will also consider taking their median. We define the median of a set a vectors as the entry-wise median, thus creating a virtual vector that is in the "middle" of the columns of B considered. This allows SVCA to be much more tolerant to gross corruptions and outliers, which are often present in HSI. (In the presence of Gaussian noise, using the average is better.)

   It also allows SVCA to be more tolerant to a misspecified value of p. For example, assume a scenario where there are exactly p' data points close to each vertex. For $p < p'$, one does not leverage optimally the presence of multiple pure data points. On the other side, as soon as p is larger than p', ALLS will perform rather badly because it will average p' data points close to a vertex and $p - p'$ data points potentially far away. If instead one takes the median, the algorithm remains able to extract the vertices accurately for any $p < 2p'$. In practice, as we will show in Section 2.5, using the median performs significantly better on real data sets.

Note that SVCA has the same computational cost as VCA, SPA and ALLS, namely $\mathcal{O}(r\text{nnz}(B))$ operations.

Recovery Guarantees for SVCA SVCA is very similar to ALLS, and in fact the robustness analysis of ALLS applies to SVCA, that is, Theorem 2.2 applies to SVCA. The reason is that we guarantee SVCA to extract the data point in the smoothed data set that maximizes the absolute value $l(x) = d_k^\top P^\perp x$.

2.4 Smoothed SPA

Since SVCA is equivalent to VCA for $p = 1$, it is not guaranteed to be deterministically robust to noise. This motivates us to propose smoothed SPA. Unfortunately, it is not practical to apply SPA directly on the smoothed data set. Indeed, it would require to find the p columns of the smoothed data set with the largest ℓ_2 norm. The ℓ_2 norm being a nonlinear function, it would require to explicitly compute the $\binom{n}{p}$ data points of the smoothed data set, which is computationally prohibitive.

Instead, we replace the random selection of $u_k = Y\mathcal{N}(0, 1)$ in SVCA by the column of the residual $P^\perp B$ with maximum ℓ_2 norm, that is, $u_k = P^\perp B(:, j_k)$ for some j_k so that $\|u_k\|_2 \geq \|P^\perp B(:, j)\|_2$ for all j. This allows us to combine the best of 'both worlds': deterministic robustness under separability when $p = 1$, and the use of the proximate latent point assumption when $p > 1$ (Assumption 2.2).

Recovery Guarantees for SSPA For $p = 1$, SSPA coincides with SPA, and hence Theorem 2.1 applies to SSPA for $p = 1$, that is, it is deterministically robust to column-wise bounded noise. However, since the selection step of SSPA is deterministic, Theorem 2.2 does not apply to SSPA. A promising direction of further research would be to analyze noise robustness of SSPA for $p > 1$.

Algorithm 2.4: Smoothed Vertex Component Analysis (SVCA)

Input: The matrix $B \in \mathbb{R}^{m \times n}$, the number r of columns of $A \in \mathbb{R}^{m \times r}$, the number p of columns of B to be averaged to obtain each column of A, the aggregation method (median or mean).

Output: A matrix A such that $B \approx AX$ for some $X \geq 0$.

Let $A = [\,]$, $P^\perp = I_m$, $V = [\,]$.

Let $Y \in \mathbb{R}^{m \times r}$ be the vector space spanned by the top r left singular vectors of B.

for $k = 1 : r$ **do**

  Pick a random direction $d_k \in \mathbb{R}^m$ in the subspace spanned by Y, e.g., $d_k \sim Y\mathcal{N}(0, I_r)$.

  Compute $u_k = \left(d_k^T P^\perp\right) B \in \mathbb{R}^n$.

  if *the median of the p largest values of u_k is larger than the absolute value of the median of the p smallest values of u_k* **then**

    Let $\mathcal{S}_k$ be the set of p indices maximizing u_k.

  else

    Let $\mathcal{S}_k$ be the set of p indices minimizing u_k.

  Let $A(:, k)$ be the median (or the mean) of the columns of $B(:, \mathcal{S}_k)$.

  Update the projector $P^\perp$ onto the orthogonal complement of $A = B(:, \mathcal{J})$:

$$v_k = \frac{P^\perp B(:, j_k)}{\|P^\perp B(:, j_k)\|_2},$$
$$V = [V\ v_k],$$
$$P^\perp \leftarrow \left(I_m - VV^T\right).$$

Should you use SVCA or SSPA? SVCA has the advantage to be a randomized algorithm, and hence can be run multiple times and the best solution, according to some criterion, can be kept. SSPA is deterministic and has the advantage to have stronger theoretical guarantees for $p = 1$. From a practical point of view, one could run SVCA several times, and SSPA once, and then keep the best solution.

Comparing algorithms In these experiments, when the ground truth A^* is not available, we will use the following criterion

$$Q_F(A) = \frac{\min_{X \geq 0} \|B - AX\|_F}{\|B\|_F} \in [0, 1],$$

to evaluate the quality of a solution A. Note that we do not use the sum-to-one constraint, $X^\top e = e$, because, in many practical situations, including hyperspectral imaging, this constraint is not satisfied for all columns of X, e.g., for pixels with low luminosity; see a discussion in [32].

Algorithm 2.5: Smoothed Successive Projection Algorithm (SSPA)

Input: The matrix $B \in \mathbb{R}^{m \times n}$, the number r of columns of $A \in \mathbb{R}^{m \times r}$, the number p of columns of B to be averaged to obtain each column of A, the aggregation method (median or mean).

Output: A matrix A such that $B \approx AX$ for some $X \geq 0$.

Let $A = [\,]$, $P^\perp = I_m$, $V = [\,]$.
Let $u_1(j) = \|B(:,j)\|_2^2$ for all j.
for $k = 1 : r$ **do**
  Let $j_k = \operatorname{argmax}_{1 \leq j \leq n} u_k(j)$. (Break ties arbitrarily, if necessary.)
  Let $d_k = B(:, j_k)$.
  Compute $u_k = (d_k^T P^\perp)B \in \mathbb{R}^n$.
  if $\max_i u_k(i) \geq -\min_i u_k(i)$ **then**
    Let $\mathcal{S}_k$ be the set of p indices maximizing u_k.
  else
    Let $\mathcal{S}_k$ be the set of p indices minimizing u_k.
  Let $A(:,k)$ be the median (or the mean) of the columns of $B(:, \mathcal{S}_k)$.
  Update the projector $P^\perp$ onto the orthogonal complement of $A = B(:, \mathcal{J})$:

$$\begin{aligned} v_k &= \frac{P^\perp B(:, j_k)}{\|P^\perp B(:, j_k)\|_2}, \\ V &= [V\, v_k], \\ P^\perp &\leftarrow \left(I_m - VV^T\right). \end{aligned}$$

  Update the squared norms of the columns of $P^\perp B$: for all j,

$$u_{k+1}(j) = u_k(j) - v_k^\top B(:,j) = \|P^\perp B(:,j)\|_2^2.$$

In matrix factorization, it could be argued that in general $Q_F(A)$ is not a good measure to assess the quality of a solution A, since any A such that $\operatorname{conv}(B) \subset \operatorname{conv}(A)$ implies $Q_F(A) = 0$, even if A is very different from the ground truth A^*. In our case though, $Q_F(A)$ is relevant, since all the considered algorithms generate solutions A which columns are close to $\operatorname{conv}(B)$. Indeed, VCA, SPA and ALLS generate solutions within $\operatorname{conv}(B)$. This is not always the case for SVCA and SSPA, because of the use of the median (a non-linear operator) for the aggregation of the columns of B. In practice though, they find solutions close to $\operatorname{conv}(B)$.

2.5 Numerical experiments

In this section, we study and compare the performance of ALLS, SVCA, and SSPA. We first consider synthetic datasets and then the unmixing of real-world hyperspectral

images. The code and data are available online[1]. All algorithms are implemented in Matlab and run on a computer with an i5-8350U processor.

2.5.1 Synthetic data sets

In this section, we study the behavior of smooth separable NMF algorithms in several experimental setups. To build synthetic data sets, we first build $A \in \mathbb{R}_+^{224\times 10}$ by selecting 10 columns from the USGS hyperspectral library[2] using SPA. The condition number of the corresponding matrix is $\kappa(A) = 33.88$. The corresponding spectral signatures are shown in Fig. 2.2. Then, we generate a random $X \in \mathbb{R}_+^{10\times 1000}$ such

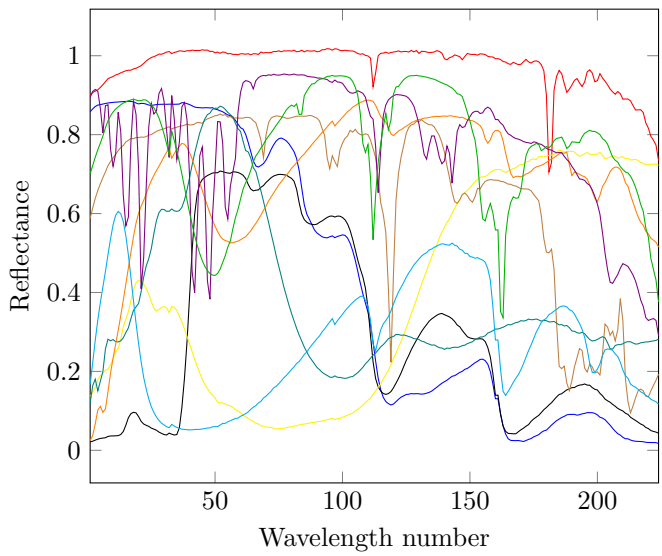


Figure 2.2: Spectral signatures (that is, columns of A) used to generate synthetic data sets.

that $X = [I_{10}, X']$, meaning there is at least one pure data point for every vertex. The coefficients of X' follow a Dirichlet distribution, which is usually a good model for the abundances in HSI [62], of parameters αe, where α controls the proportion of data points close to the vertices, see Table 2.1. The larger α, the denser the columns of X and the less likely the 'proximal latent points' assumption is to be satisfied for large p.

Finally, we let $B = AX + N$ where N is a normalized Gaussian noise: Given a noise level ϵ, we first generate $N(i,j) \sim \mathcal{N}(0,1)$ for all (i,j), then set

$$N \leftarrow \epsilon \frac{\|AX\|_F}{\|N\|_F} N,$$

[1] `https://gitlab.com/nnadisic/smoothed-separable-nmf`
[2] `https://www.usgs.gov`

Table 2.1: Generating the columns of $X \in \mathbb{R}^{10\times n}$ using the Dirichlet distribution of parameter αe, this table reports the expected percentage, δ, of pure data points close to each vertex. The jth data point is considered close to the ith vertex when $X(i,j) > 0.95$, hence this table reports the expected value of $\frac{1}{n}\big|\{j \mid X(i,j) > 0.95\}\big|$ for all i. Since the Dirichlet distribution is uniform with parameters αe, this expected value is the same for all i.

α	0.01	0.02	0.05	0.1	0.2	0.5
δ	7.7%	5.9%	2.7%	0.75%	0.06%	0%

so that ϵ is the norm of the noise relative to AX: $\|N\|_F = \epsilon\|AX\|_F$.

Given the noisy data matrix B and a parameter p, we can compute A' with the algorithms ALLS, SVCA, and SSPA. We note ALLS(p), SVCA(p) and SSPA(p) these algorithms run with parameter p. Given the computed solution A', we report the mean removed spectral angle (MRSA) to assess its quality. Given two spectral signatures $x, y \in \mathbb{R}^m$, the MRSA is defined as follows:

$$\phi(x,y) = \frac{1}{\pi}\arccos\left(\frac{(x-\bar{x})^T(y-\bar{y})}{\|x-\bar{x}\|_2\|y-\bar{y}\|_2}\right) \in [0,1],$$

where for a vector $z \in \mathbb{R}^m$, $\bar{z} = (\sum_{i=1}^m z_i)e$ and e is the vector of all ones. Given two matrices, here the groundtruth A and the estimate A', we define the MRSA as

$$\mathrm{MRSA}(A, A') = \sum_{j=1}^{r} \phi(A(:,j), A'(:,j)),$$

after the columns of A' have been reordered so as to minimize the MRSA. The smaller the MRSA, the better the solution. For ALLS and SVCA, on a given data set, we run 30 trials and keep the median of the results. SSPA is deterministic so we only run it once. Unless stated otherwise, SVCA and SSPA are equipped with the median aggregation. In the following, we consider several experimental setups to highlight the property of these algorithms.

In Fig. 2.3, we test ALLS, SVCA, and SSPA with different values of the parameter p, when the noise ϵ varies. Note that SVCA(1) and SSPA(1) are equivalent to their non-smoothed version VCA and SPA. Also note that ALLS(1) is equivalent to SVCA(1) and thus VCA. In this experiment, we observe that smoothing improves the algorithm performances. However, for ALLS, a parameter p set too large can in fact worsen the solution, especially when the noise level is small. In that case, the algorithm selects too many points and aggregates points that are close to a vertex with other points that are too far away, thus drifting the estimated vertex away from the true underlying vertex. Also, ALLS is outperformed by SVCA and SSPA, except for a very small noise; this will be confirmed in experiments on hyperspectral images in Section 2.5.2.

In Fig. 2.4, we compare the stability of ALLS and SVCA when ϵ varies by showing the best, median, and worst result among 30 runs. We also compare them to the

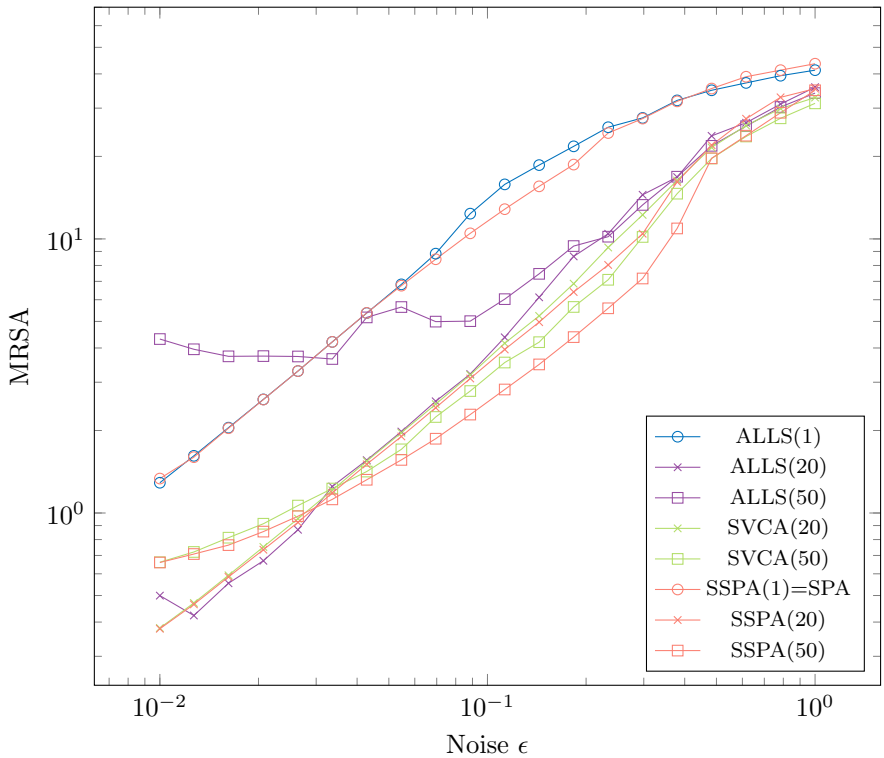


Figure 2.3: Results for ALLS, SVCA, and SSPA for different values of p, when ϵ varies, for fixed $n = 1000$ and purity $\alpha = 0.05$ ($\delta = 2.7\%$). Values for ALLS and SVCA are the medians over 30 trials. Note that ALLS(1)=SVCA(1)=VCA.

MRSA of the result of SVCA that has the smallest reconstruction error, $Q_F(A')$, and to SSPA. We see that the best result from ALLS is slightly better than other results. This is due to the use of the mean as an aggregation method, which works better with the centered Gaussian noise of the synthetic data[3], see Fig. 2.6. However, the algorithm is less stable, as the median and worst result are worst than SVCA. With SVCA, the median results are close to the best. Also, the best results in terms of reconstruction error generally coincides with the best one in terms of MRSA, showing that the reconstruction error is a good proxy for the tested algorithms, as evoked in Section 2.4 (this will be useful when the groundtruth is unknown and the MRSA cannot be computed, for example in Section 2.5.2). The deterministic SSPA is better than the median result of SVCA, but not always better than its best result.

In Fig. 2.5, we compare SVCA and SSPA with higher values of p when ϵ varies. Again, we observe that the smoothing improves the algorithm performances, but an overestimated p worsens it. Interestingly, when the noise is very high (above

[3]Using the mean instead of the median in SVCA, its best MRSA result is always better than that of ALLS in this experiment.

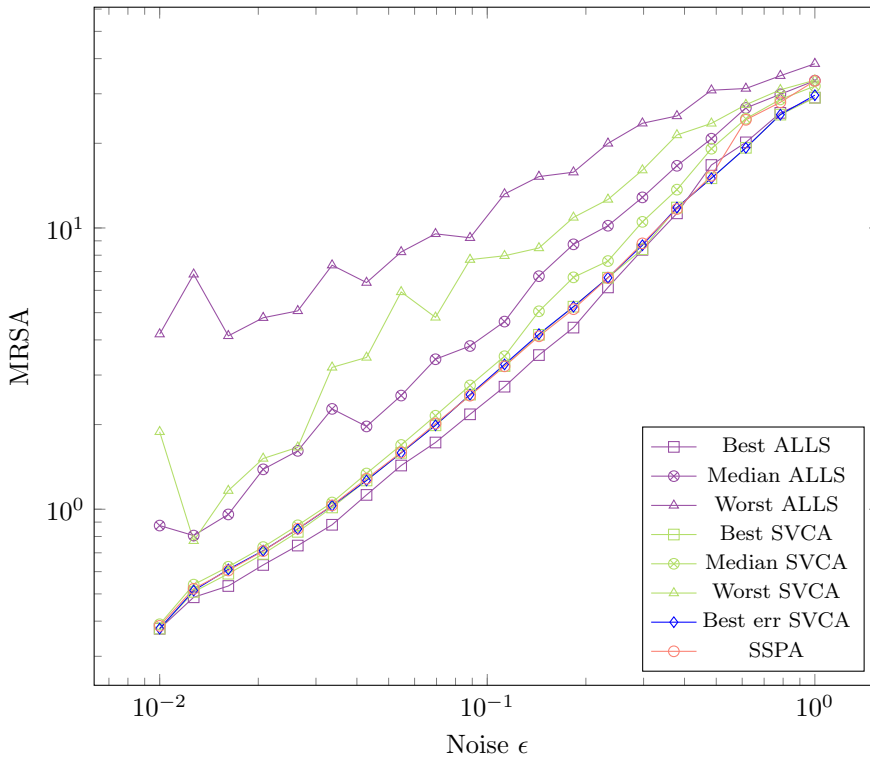


Figure 2.4: Comparison of SSPA with best, median, and worst result for ALLS and SVCA among 30 runs, when ϵ varies, for fixed $n = 1000$, purity $\alpha = 0.05$ ($\delta = 2.7\%$) and parameter $p = 25$. "Best err SVCA" represents the MRSA of the solution of SVCA that has the smallest relative reconstruction error.

10%), smoothed algorithms outperform their non-smoothed counterpart even when p is overestimated. This is due to the fact that the value of p required to obtain the best estimation of A is not only determined by the purity but also by the noise level. For instance, consider a toy example with four data points and $r = 2$:

$$B = AX + N = A \begin{bmatrix} 1 & 0 & 0.99 & 0.01 \\ 0 & 1 & 0.01 & 0.99 \end{bmatrix} + N.$$

That is, x_3 and x_4 are not pure-pixel but are almost pure. Let us further assume N to follow a centered Gaussian law. If $\epsilon = 0$ (noiseless mixing), the best estimation of $A(:,1)$ (resp. $A(:,2)$) from B is to extract $B(:,1)$ (resp. $B(:,2)$), yielding a perfect estimation. On the other hand, if ϵ is large relatively to the distance of $A(:,1)$ and $A(:,2)$, it is better to choose as an estimate of $A(:,1)$ (resp. $A(:,2)$) the average – or median – of $B(:,1)$ and $B(:,3)$ (resp. $B(:,2)$ and $B(:,4)$), as the noise power is then divided by two.

In Fig. 2.6, we compare SVCA and SSPA equipped with either the median or the

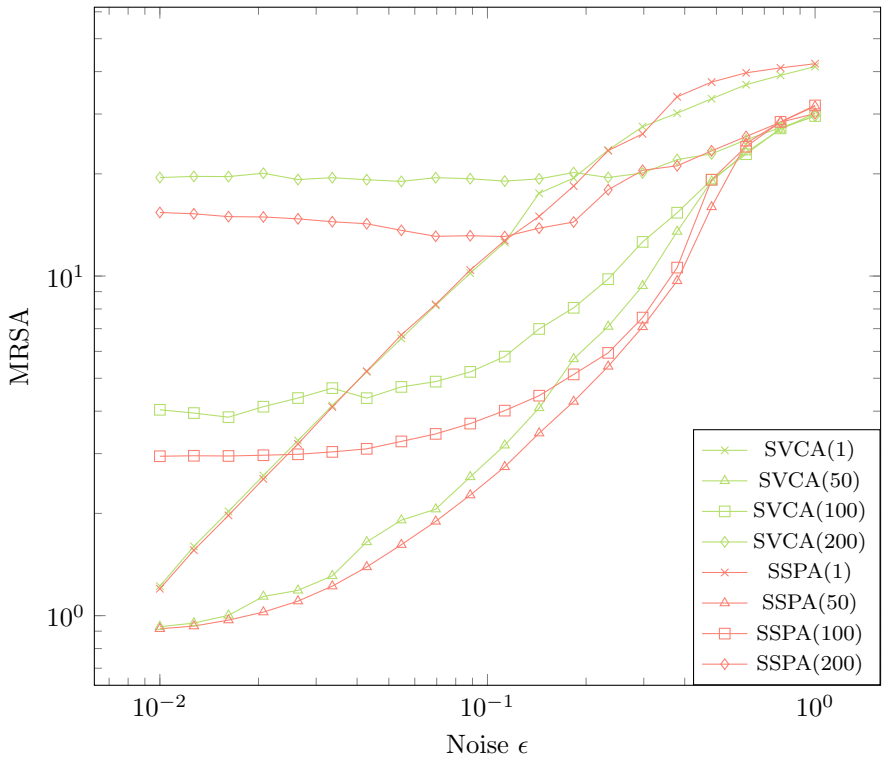


Figure 2.5: Results for SVCA and SSPA for different values of p, when ϵ varies, for fixed $n = 1000$ and purity $\alpha = 0.05$ ($\delta = 2.7\%$), Values for SVCA are the medians over 30 trials.

mean aggregation, for fixed data setup and when p varies. The reverse bell curve shows that the performance of the algorithms improves gradually as p grows, until it reaches an optimal value, after which the performance gradually worsens. We observe that the algorithms equipped with the median are more robust to an overestimation of p, but with the mean they are slightly better for smaller p. However the difference is small; this is expected as this synthetic data is generated with centered Gaussian noise, and as such the mean is expected to give the best estimation when p is well chosen. Note that the results would be different with different kind of noises, and the median could for instance be better with sparse noises. The difference is more obvious in hyperspectral images, see Section 2.5.2 and Fig. 2.8.

In Fig. 2.7, we compare SVCA and SSPA when p varies in setups with different values of purity α. As expected, we observe that the shapes of the curves are similar, and that for a fixed noise level the value of the parameter p leading to the lowest MRSA decreases as the parameter α increases.

To summarize, our synthetic experiments highlight that the two proposed algorithms generally obtain better results than ALLS. They furthermore show that SVCA

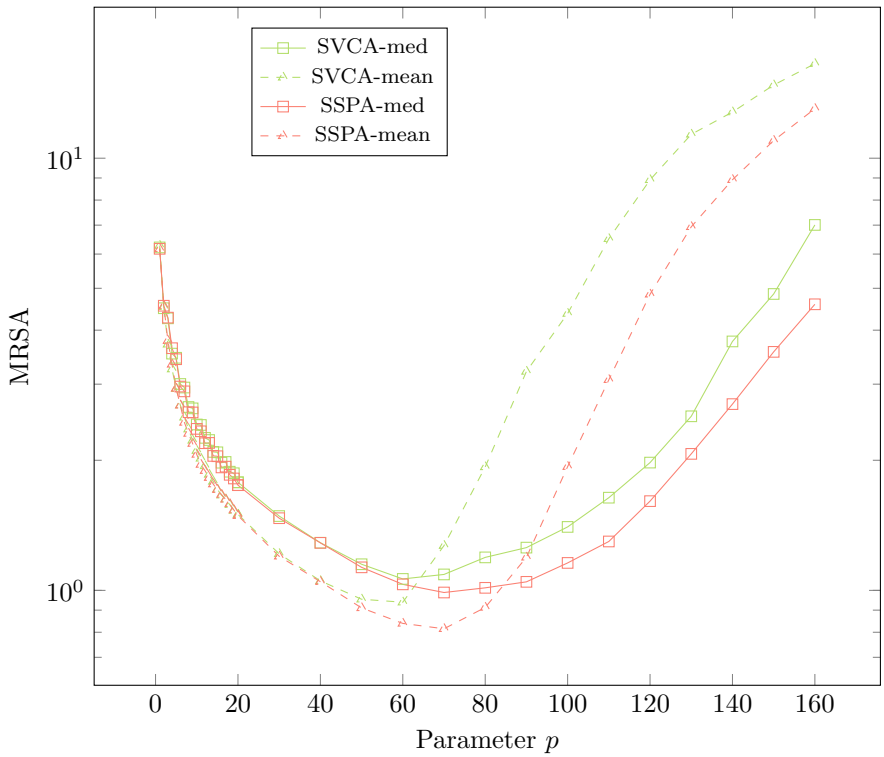


Figure 2.6: Results for SVCA and SSPA using either the median or the mean to average points, when p varies, for fixed $n = 1000$, purity $\alpha = 0.05$ ($\delta = 2.7\%$), and noise $\epsilon = 0.05$. Values for SVCA are the medians over 30 trials.

and SSPA outperform VCA and SPA when p is well chosen. Although the choice of p is important, as a bad value can worsen the results compared to the non-smoothed algorithms, the use of the median instead of the mean makes the algorithms less sensitive to this choice.

2.5.2 Hyperspectral images

In this section, we apply ALLS, SVCA, and SSPA to the unmixing of hyperspectral images, as described in Section 2.1. We consider three commonly used hyperspectral images[4], San Diego, Urban, and Terrain. In hyperspectral data sets, extremely large values are commonly associated with sensor noise or interference. To avoid overfitting the factorizations to these interferences, the pixels corresponding to the 10 largest values of any wavelength range are zeroed out. Extreme pixels generally have extreme values in many wavelenghts ranges at once, so this preprocessing removes less than 0.1% pixels. The characteristics of these images are summarized in Table 2.2.

[4]Downloaded from `http://lesun.weebly.com/hyperspectral-data-set.html`

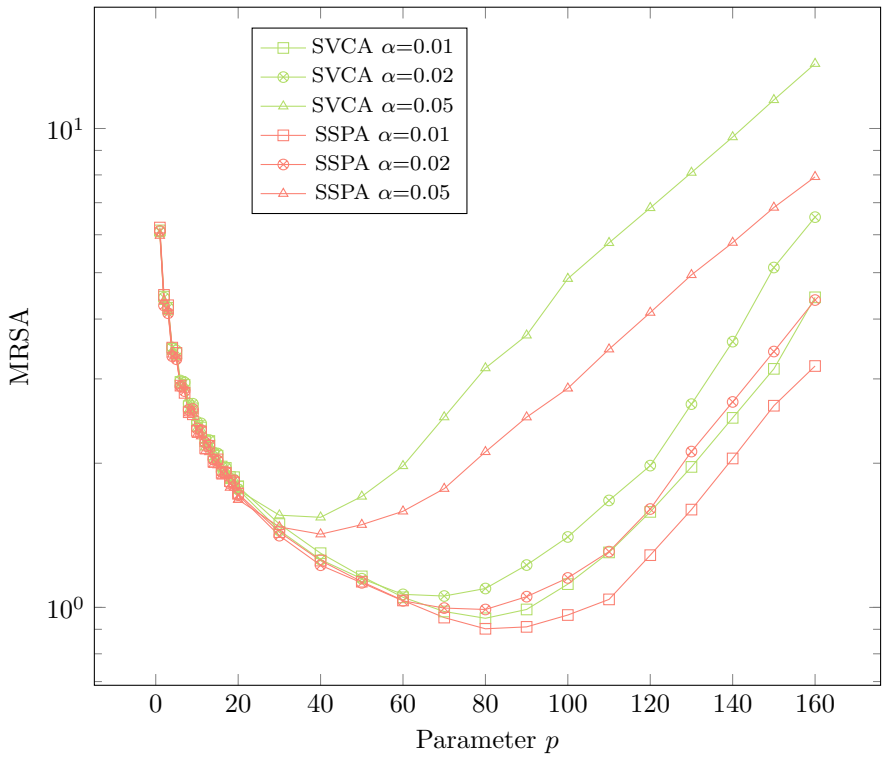


Figure 2.7: Results for SVCA and SSPA for different values of purity α, when p varies, for fixed $n = 1000$ and noise $\epsilon = 0.05$. Values for SVCA are the medians over 30 trials.

Given a data matrix $B \in \mathbb{R}^{m \times n}$, we compute $A \in \mathbb{R}^{m \times r}$ with the three algorithms. We then compute for each algorithm $X \in \mathbb{R}^{r \times n}$ with a standard coordinate descent algorithm [34] and measure the relative reconstruction error $\|B - AX\|_F / \|B\|_F$. The smaller the error, the better the solution.

Some works such as [85] proposed groundtruths for these hyperspectral images, but they are computed using numerical methods and as such do not necessarily represent reality. Therefore, we lack a reference to assess the quality of the reconstruction, for example by measuring the MRSA. This is why we use the relative reconstruction error as the criterion for the quality of a solution. This is a satisfying criterion, as illustrated in Fig. 2.4.

In Table 2.3, we report results from the experiments. We observe that, when $p > 1$, the result is always improved. When p is too large, however, the solution can be worse. We observe that the best p varies between the algorithms. For example, with Terrain, ALLS and SVCA perform best with $p = 1000$ while SSPA performs best with $p = 100$. Larger values of p seem to give more stable results, as it produces solutions with smaller deviations. SVCA outperforms ALLS in all cases. SSPA performance

Table 2.2: Summary of the hyperspectral images studied in this work

Dataset	m	n	r	Pixels zeroed out
San Diego	158	$400 \times 400 = 160000$	8	19
Urban	162	$307 \times 307 = 94249$	6	68
Terrain	188	$500 \times 307 = 153500$	6	107

is comparable to SVCA, and generally produces a better result than the median of SVCA, but never better than the best result obtained by SVCA over 30 runs.

In a few cases, SSPA produces solutions with a large error, for example in San Diego for $p = 100$ and Terrain for $p = 1000$. We believe this behavior to originate from small groups of points with a very large norm, that could correspond to a rare material or to interference.

Table 2.3: Relative reconstruction errors ($\min_{X \geq 0} \|B - AX\|_F / \|B\|_F$) resulting from the unmixing of hyperspectral images with ALLS, SVCA, and SSPA, with different values of parameter p. SVCA(1) and SSPA(1) are equivalent to VCA and SPA. For non-deterministic algorithms ALLS and SVCA, we show the minimum, median, standard deviation, and maximum of the error over 30 trials. Numbers in bold correspond to the best results for each algorithm on each dataset.

		SanDiego			Urban			Terrain		
	p	Min	Med ± std	Max	Min	Med ± std	Max	Min	Med ± std	Max
ALLS	1	4.72	5.60 ± 0.67	8.25	**5.39**	**9.14** ± 1.93	12.26	3.94	4.88 ± 0.73	7.08
	100	**4.27**	**5.35** ± 1.72	10.91	6.37	9.28 ± 3.18	19.40	**3.84**	4.87 ± 0.87	6.85
	1000	4.64	6.14 ± 1.12	8.68	6.78	9.71 ± 2.16	14.20	**3.84**	**4.71** ± 1.19	8.81
	2000	4.87	5.91 ± 1.62	11.79	6.96	9.93 ± 1.60	12.85	3.96	4.89 ± 0.88	7.63
	5000	5.51	7.42 ± 2.64	13.88	7.68	10.37 ± 1.89	14.98	4.28	5.26 ± 0.80	6.88
SVCA	1	3.95	5.42 ± 0.61	6.90	6.25	9.13 ± 1.78	12.23	4.03	5.11 ± 1.25	8.70
	100	**3.44**	4.92 ± 0.77	6.96	**5.08**	**6.10** ± 1.27	10.13	3.52	4.04 ± 0.67	6.52
	1000	3.82	4.95 ± 0.59	6.82	5.82	6.77 ± 1.23	10.84	**3.18**	**3.92** ± 0.38	4.70
	2000	3.73	**4.40** ± 0.51	5.81	5.66	6.36 ± 0.67	7.83	3.38	4.12 ± 0.45	4.95
	5000	4.01	4.66 ± 0.73	7.01	5.69	6.94 ± 1.21	11.74	3.70	4.19 ± 0.30	4.82
SSPA	1		5.90			9.46			5.01	
	100		9.29			6.65			**4.03**	
	1000		5.82			6.22			8.05	
	2000		**4.32**			6.11			7.86	
	5000		4.65			**5.91**			5.38	

In Fig. 2.8, we compare SVCA and SSPA using either the median or the mean for the unmixing of the hyperspectral image Urban, with a varying p. We observe that $p > 1$ always leads to better results for all algorithms and all data sets. Also, the median aggregation almost always gives better results than the mean. While the curves are not as regular as with synthetic data, we observe a similar tendency that the solution improves when p grows, until a certain point or zone after which it worsens again. However, SSPA-med has an irregular behaviour for $p = 200$. This can be explained by the fact that SSPA is a greedy algorithm, so if it makes a bad choice in the first iterations, it will likely never compensate. Also, it is deterministic, so the

error is not averaged over several runs.

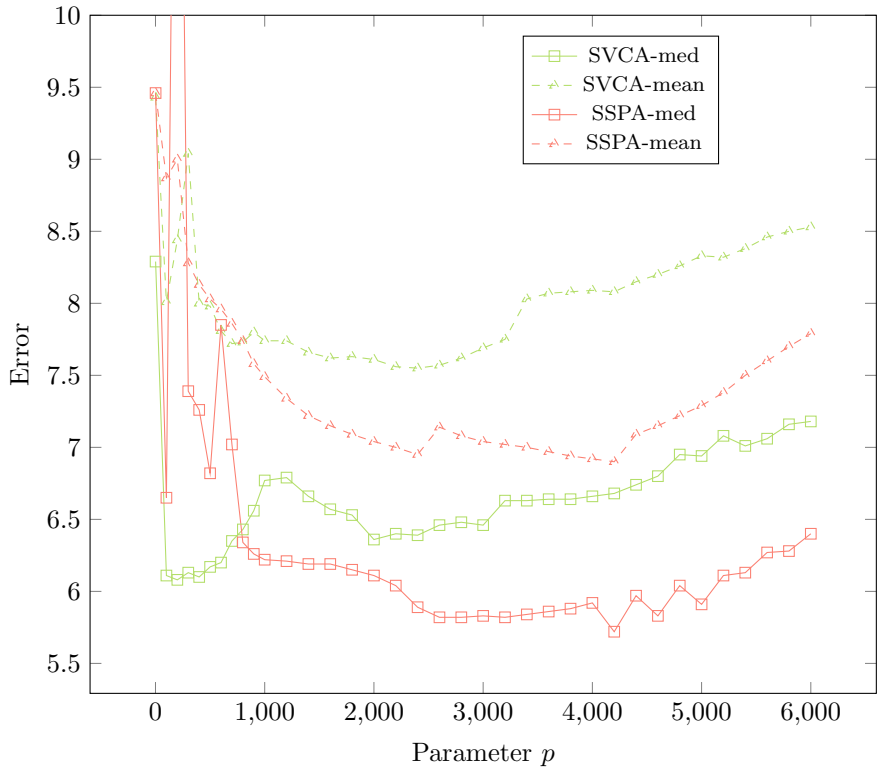


Figure 2.8: Results of the unmixing of the hyperspectral image Urban. Values for SVCA are the medians over 30 trials. One point is out of the plot; for $p = 200$, SVCA-med has an error of 14.55%.

In Fig. 2.9, we show the abundances maps corresponding to the unmixing of Urban. They indicate the proportion of every of the 6 extracted endmembers in the pixels of the image. We see that the smoothed algorithms obtain a better separation than the non-smoothed ones. For example, the fourth endmember extracted by SVCA and SSPA corresponds to grass, and it is well separated by these algorithms, while VCA and SPA mix it with asphalt and dirt. The second endmember extracted by SVCA and SSPA corresponds to metallic rooftops, and it is well separated while VCA mixes it with other materials and SPA does not clearly identify it and produces a blurred picture. The corresponding spectral signatures can be found in Fig. 2.10.

The results from our experiments on other hyperspectral images are in Figs. 2.11 to 2.13 for Terrain, and in Figs. 2.14 to 2.16 for San Diego.

2.6 Discussion and conclusion

Let us discuss some additional points regarding SVCA and SSPA.

Which algorithm should one use? SVCA allows to generate different solutions, among which the best solution w.r.t. reconstruction error can be found. Therefore, in practice and when time and resources allow, we recommend running SSPA once and SVCA several times, with different values of p, and keep the best solution.

Which aggregation should one use? Apart from the average and the median, other aggregation methods could perform better depending on the noise statistics and data set at hand. For instance, the authors in [42] use an aggregation on manifold in the different context of sparse matrix factorization to better take into account the structure of the columns of A.

How to select p? Choosing a value for the parameter p is crucial and not trivial; a strategy to determine it is an interesting direction of research. It could also be useful to consider a different value of p for every vertex, as the number of proximal latent points typically varies for each vertex.

Spectral variability in blind HU In blind HU, an issue with separable NMF algorithms is that they identify a single pixel to represent a material. It is however well-known that the spectral signature of an endmember may vary across the pixels of the image, for example because of differences in light intensity or orientation. This is known as *spectral variability*. By construction, most separable NMF algorithms, such as VCA and SPA, will identify pure pixels that do not represent well the average behaviour of a material, but rather a pure pixel located at the boundary of the convex hull of the variations of the spectral signature of that endmember. Therefore, working on the smoothed data set, which averages every subset of p data points, allows to better represent this average behaviour.

Designing other smoothed separable NMF algorithms The proximal latent points assumption could be used to generalize other separable NMF algorithms. In this work, we validated the idea on the two most widely used separable NMF algorithms, namely VCA and SPA, but the same idea can be applied to any separable NMF algorithms such as SNPA [32].

Conclusion In this chapter, we investigated the smoothed separable NMF model that strengthens the separability assumption by assuming the presence of several data points nearby each column of the basis matrix A. Inspired by the existing algorithm ALLS, we developed smoothed variants of two separable NMF algorithms, namely VCA and SPA. Empirically, we showed that our smoothed methods outperform both the non-smoothed ones and ALLS, for both synthetic data sets and for the unmixing of real-world hyperspectral images. This shows that the proximal latent points

assumption is verified in hyperspectral images, and that smoothed separable NMF algorithms are a more effective tool for hyperspectral unmixing. Further works include the design of other smoothed separable NMF algorithms, and the use of SVCA and SSPA for other applications, in particular in machine learning [7, 11].

(a) VCA, error= 6.24%

(b) SVCA p=200, error= 5.24%

(c) SPA, error= 9.46%

(d) SSPA p=4200, error= 5.72%

Figure 2.9: Abundance maps of the unmixing of the Urban hyperspectral image (that is, reshaped rows of X) with different algorithms. Endmembers have been reordered for easier comparison. Parameters p have been chosen as the best from Fig. 2.8. Error corresponds to $\min_{X \geq 0} \|B - AX\|_F / \|B\|_F$. For VCA and SVCA, we show the best solution over 30 trials.

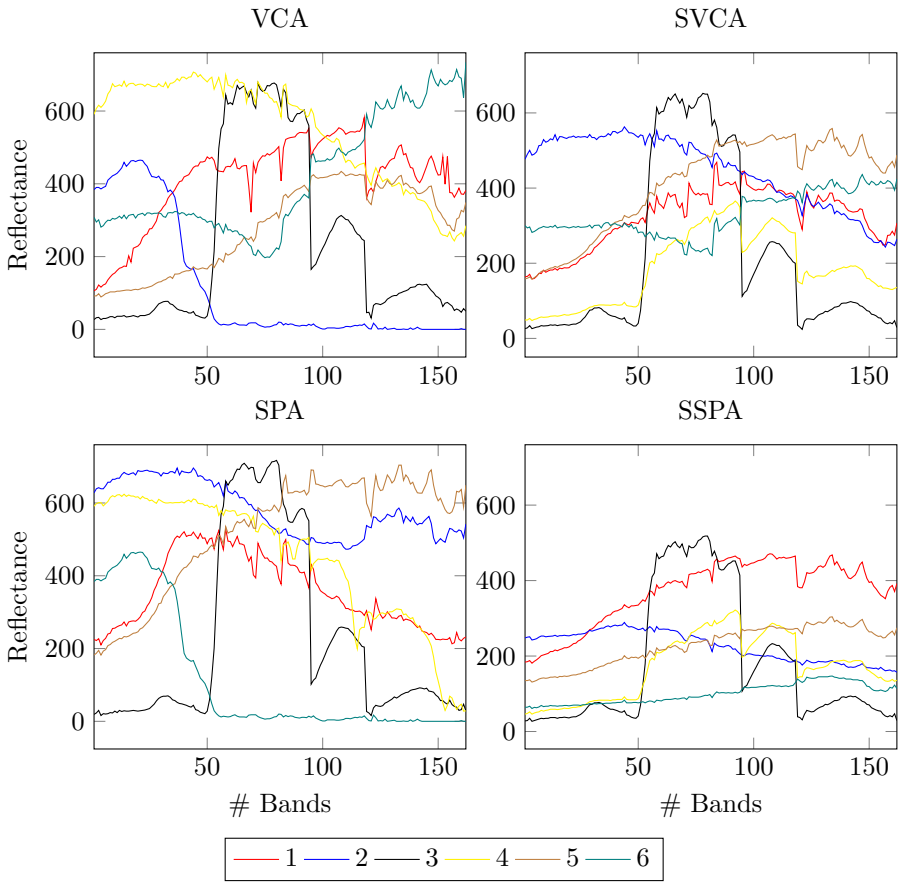


Figure 2.10: Spectral signatures from the unmixing of the Urban hyperspectral image (that is, columns of W) with different algorithms. Parameters p have been chosen as the best from Fig. 2.8. For VCA and SVCA, we show the best solution over 30 trials. Spectral signatures correspond to the abundance maps shown in Fig. 2.9, in the same order. For example, for SSPA, the spectral signatures correspond to (1) dirt, (2) roof tops, (3) trees, (4) grass, (5) a mixture of road and dirt, and (6) roads.

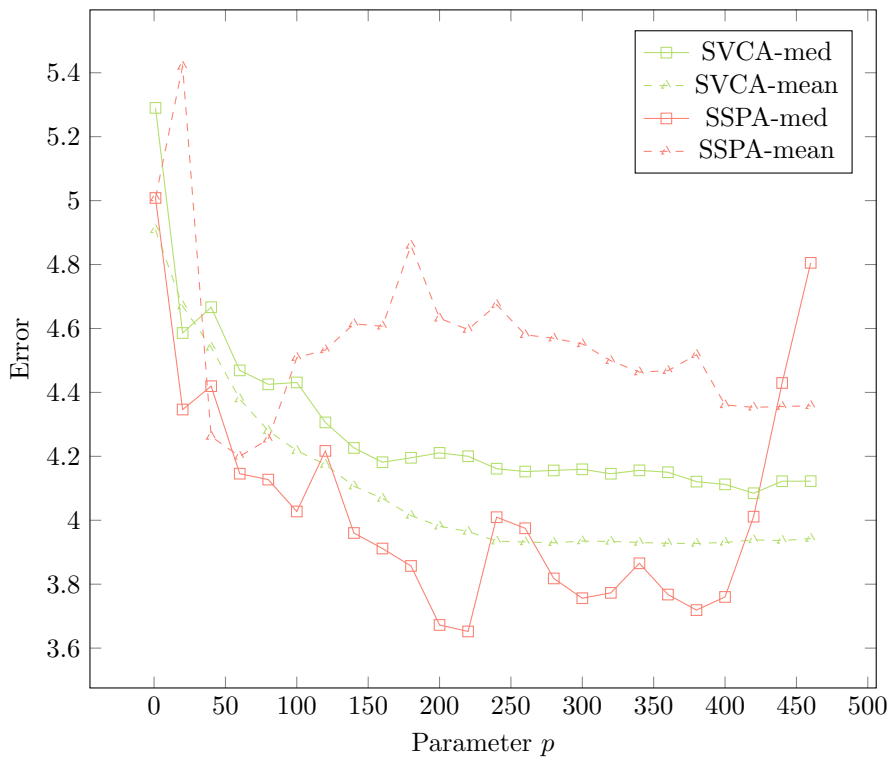
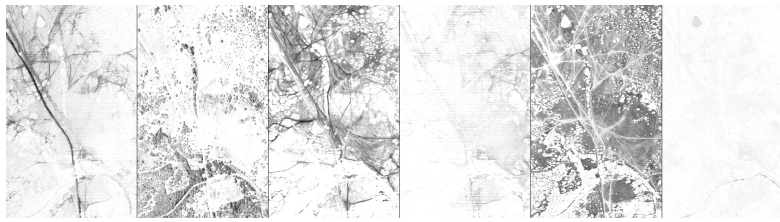
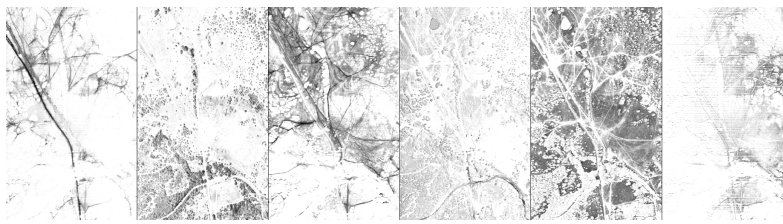


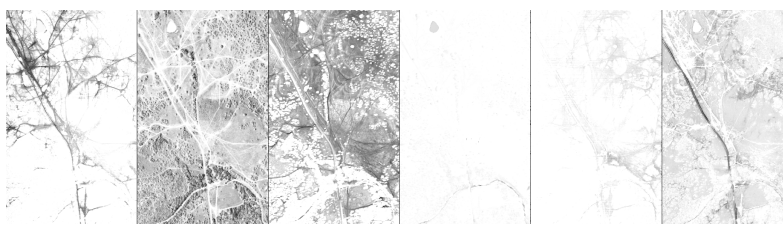
Figure 2.11: Results of the unmixing of the hyperspectral image Terrain. Values for SVCA are the medians over 30 trials.



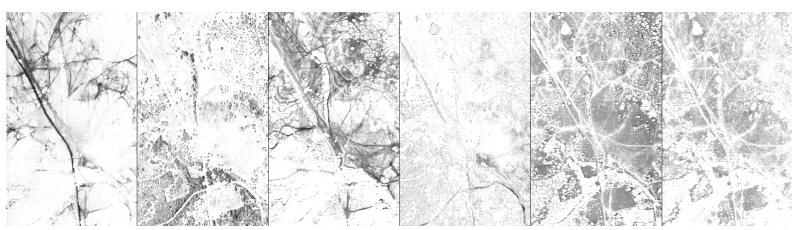
(a) VCA, error= 4.03%



(b) SVCA p=420, error= 3.25%



(c) SPA, error= 5.01%



(d) SSPA p=220, error= 3.65%

Figure 2.12: Abundance maps of the unmixing of the Terrain hyperspectral image (that is, reshaped rows of H) with different algorithms. Endmembers have been reordered for easier comparison. Parameters p have been chosen as the best from Fig. 2.11. Error corresponds to $\min_{H\geq 0} \|X - WH\|_F / \|X\|_F$. For VCA and SVCA, we show the best solution over 30 trials.

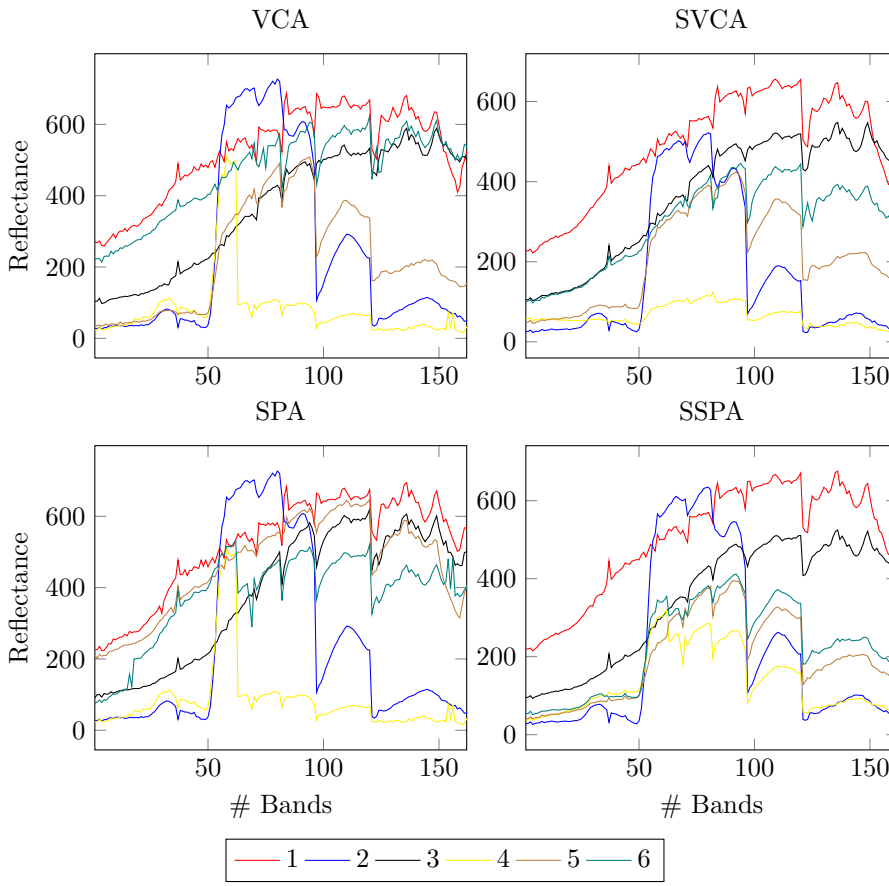


Figure 2.13: Spectral signatures from the unmixing of the Terrain hyperspectral image (that is, columns of W) with different algorithms. Parameters p have been chosen as the best from Fig. 2.11. For VCA and SVCA, we show the best solution over 30 trials. Spectral signatures correspond to the abundance maps shown in Fig. 2.12, in the same order.

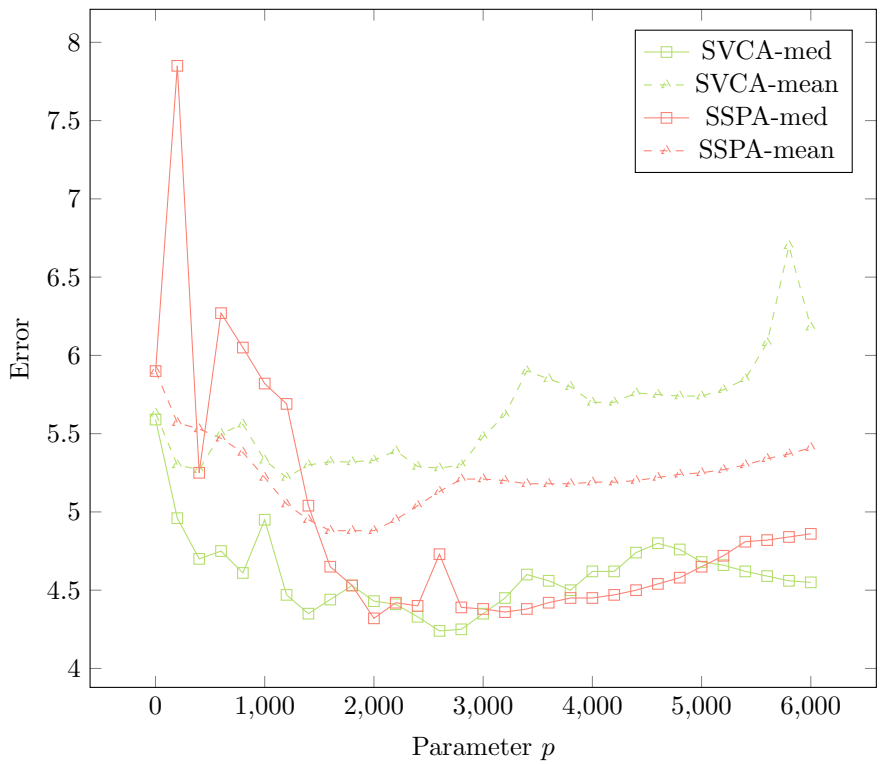
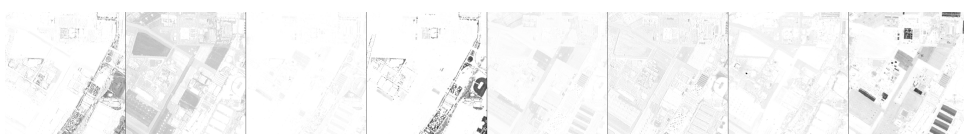


Figure 2.14: Results of the unmixing of the hyperspectral image San Diego. Values for SVCA are the medians over 30 trials.

(a) VCA, error= 3.95%

(b) SVCA p=2600, error= 3.74%



(c) SPA, error= 5.90%

(d) SSPA p=2000, error= 4.32%

Figure 2.15: Abundance maps of the unmixing of the San Diego hyperspectral image (that is, reshaped rows of H) with different algorithms. Endmembers have been reordered for easier comparison. Parameters p have been chosen as the best from Fig. 2.14. Error corresponds to $\min_{H \geq 0} \|X - WH\|_F / \|X\|_F$. For VCA and SVCA, we show the best solution over 30 trials.

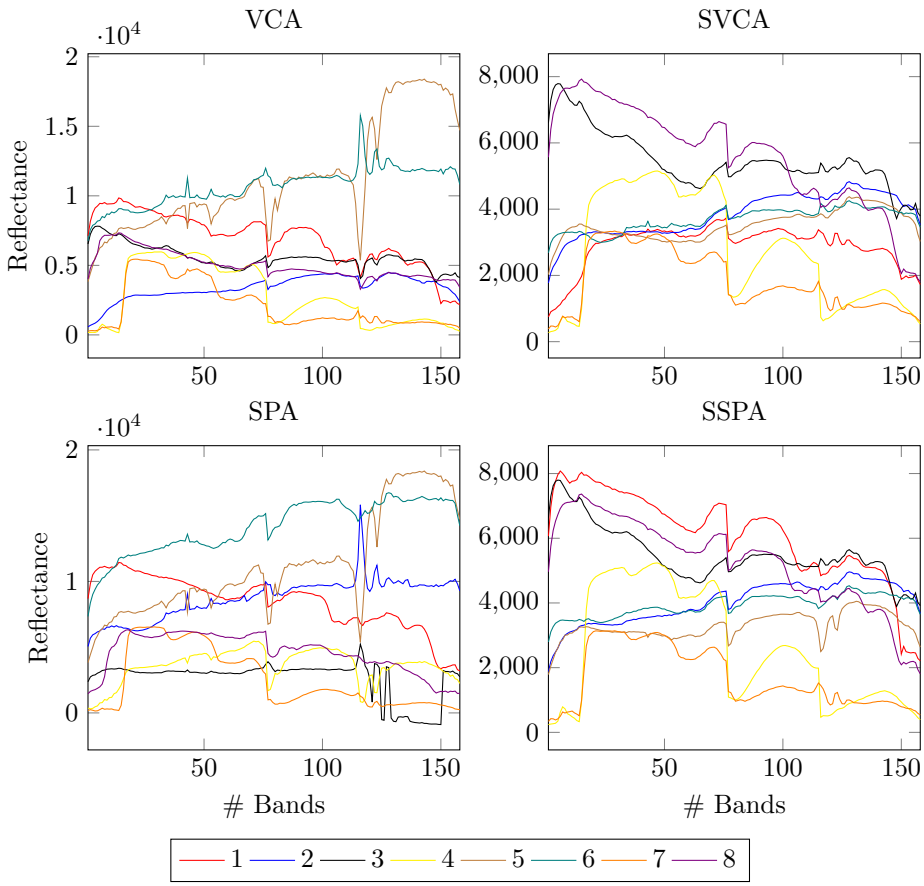


Figure 2.16: Spectral signatures from the unmixing of the San Diego hyperspectral image (that is, columns of W) with different algorithms. Parameters p have been chosen as the best from Fig. 2.14. For VCA and SVCA, we show the best solution over 30 trials. Spectral signatures correspond to the abundance maps shown in Fig. 2.15, in the same order.

Chapter 3

Branch-and-bound for Sparse NNLS

> *"Beware of bugs in the above code; I have only proved it correct, not tried it."*
>
> – Donald Knuth

In this chapter, we propose a novel algorithm to solve exactly the k-sparse NNLS problem (1.3). As a reminder, it consists in

$$\min_{x \geq 0} \|Ax - b\|_2^2 \quad \text{such that} \quad \|x\|_0 \leq k,$$

given $A \in \mathbb{R}^{m \times r}$, $b \in \mathbb{R}^m$, and a sparsity parameter $k \in \mathbb{N}$. We also propose an extension of this algorithm to solve the biobjective sparse NNLS problem (1.4), that we recall here,

$$\min_{x \geq 0} \{\|Ax - b\|_2^2, \|x\|_0\}.$$

This algorithm is based on a dedicated branch-and-bound strategy, and it is able to compute an optimal solution even in complicated cases such as noisy or ill-conditioned data, where traditional approaches fail. We show that our proposed algorithm scales well, despite the combinatorial nature of the problem. We illustrate the advantages of the proposed technique on synthetic data sets, as well as a real-world hyperspectral image.

The content of this chapter is mainly extracted from [60, 61]:

60. Nadisic, N., Vandaele, A., Gillis, N. & Cohen, J. E. *Exact biobjective k-sparse nonnegative least squares* in *29th European Signal Processing Conference (EUSIPCO)* (2021), 2079–2083.

61. Nadisic, N., Vandaele, A., Gillis, N. & Cohen, J. E. *Exact sparse nonnegative least squares* in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020), 5395–5399.

3.1 Introduction

As discussed in Section 1.2, the k-sparse NNLS problem (1.3) is key in many applications. This combinatorial problem is hard to solve exactly, with $\binom{r}{k}$ possible supports for x. In most cases practitioners use heuristics such as greedy algorithms that have limited guarantees. However, in critical applications, computing an exact solution can be useful. Notably, with an exact solution we know that the reconstruction error comes from model misfit or acquisition noise and not from the optimization algorithm.

A brute-force algorithm would solve a NNLS subproblem for each possible support and select the solution with minimal error. This is the approach used by [20] in the context of sparse NMF. However, the number of possible supports grows exponentially with r and k. To avoid computing all of them, the solution we propose can prune the search space using a *branch-and-bound* strategy. With this technique, the problem is still solved exactly, but much fewer NNLS subproblems are solved, leading to reasonable computing times even for large n and k.

The k-sparse NNLS problem can also be reformulated as a mixed-integer quadratic program (MIQP), and as such can be solved with a generic solver such as CPLEX. We will discuss such reformulation in next section. However, this problem features a strong structure that generic solvers are mostly indifferent to, and we will see in this chapter the advantages of a specific algorithm designed to leverage this structure.

3.2 Formulation as a MIQP

A previous work by Bourguignon et al. [15] provides a MIQP reformulation for the k-sparse least squares problem. We reused this formulation and added nonnegativity constraints, leading to the following problem:

$$\min_x \frac{1}{2} x^\top A^\top A x - b^\top A x \text{ s.t. } \begin{cases} x \geq 0 \\ 1^\top z \leq k \\ x_i \leq M z_i \text{ for all } i \end{cases}, \tag{3.1}$$

where

- $z \in \{0,1\}^r$ is a vector of auxiliary binary variables, that allows us to formulate linearly the k-sparsity constraint.
- $M \in \mathbb{R}_+$ is the so-called "big-M" constant.

This is a mixed-integer quadratic problem (MIQP) with linear constraints. In (3.1), the second constraint ensures that the binary vector z has at most k entries set to 1, while the third constraint ensures that for each zero entry of z the corresponding entry of x is also zero. The value of M can affect significantly the performance of the solver, and it should be as small as possible to accelerate the solving, while being larger than the largest entry of x to ensure that the solution is in the feasible set.

This formulation can be tackled directly by a generic solver, but we will see in the experiments of Section 3.5 that this is less performant than a specifically-designed algorithm. We present such algorithm in the next section.

3.3 The algorithm Arborescent

In this section we present the main contribution of this chapter, that is, a branch-and-bound algorithm to solve the k-sparse NNLS problem exactly. This algorithm is called Arborescent[1]. In Arborescent, the possible patterns of zeros (that is, the possible supports) in the solution vector are enumerated on a tree, as shown in Figure 3.1. Each node represents an over-support $\mathcal{K}$ of x. The entries of x indexed by $\mathcal{K}$ are those that are not constrained to be 0. The cardinality of $\mathcal{K}$ is equal to the current tree depth. Exploring a node means solving the following NNLS subproblem

$$f^*(\mathcal{K}) = \min ||A(:,\mathcal{K})x(\mathcal{K}) - b||_2^2 \quad \text{such that} \quad x(\mathcal{K}) \geq 0, \tag{3.2}$$

where $x(\mathcal{K})$ is the subvector of x whose entries are specified by the index subset $\mathcal{K} \subseteq \{1, 2, \ldots, r\}$. The value $f^*(\mathcal{K})$ is the *error* associated with the node corresponding to $\mathcal{K}$. In Arborescent, an *active-set* algorithm [70] is used as the NNLS solver. The key property of active-set methods is that they solve the NNLS problem exactly, without a need for parameter tuning. Note that active-set methods allow for a *warm start*, that is, the algorithm can be initialized at the current node with the solution from a previous node. This significantly speeds up the iterative process since it starts close to the optimal solution.

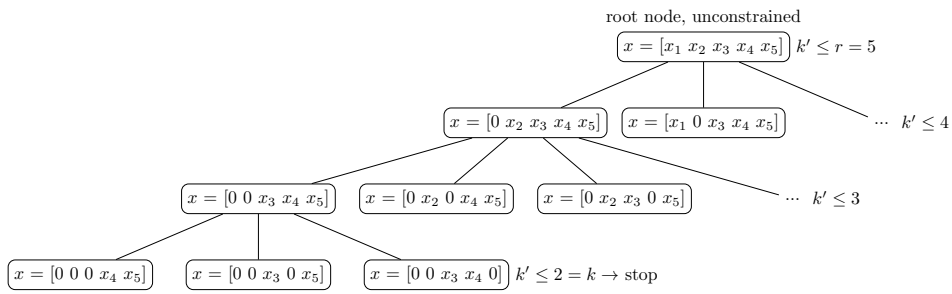


Figure 3.1: Example of the Arborescent search tree, for $r = 5$ and $k = 2$.

Branch. The root of the tree represents the unconstrained problem. No component of x is constrained to zero, that is, $\mathcal{K} = \{1, 2, \ldots, r\}$. At this root node, the support of x is required to have cardinality at most $k' = r$ (unconstrained). From this root we generate one child node for each component of x, where the respective component is not in the support (and thus constrained to zero). The cardinality of the support of x in these children nodes is therefore at most $k' = r - 1$. The tree is constructed recursively, by constraining one additional component to be equal to zero per node. When the tree has reached the depth were the cardinality of the support set is $k' = k$, the desired sparsity is obtained and the leaf nodes at this depth represent feasible solutions to (1.3).

[1]The name stands for Arborescent Realizes a Branch-and-bound Optimization to Require Explicit Sparsity Constraints to be Enforced in NNLS Tasks.

As described above, several nodes of the Arborescent tree would correspond to the same over-support $\mathcal{K}$. For example, in Figure 3.1 there would be a redundant node $\{0, 0, x_3, x_4, x_5\}$ as the child of both $\{0, x_2, x_3, x_4, x_5\}$ and $\{x_1, 0, x_3, x_4, x_5\}$. To avoid this redundant computations, we fix the order of the entries of x in the root node and keep this order in all nodes. This way, we can deduce trivially if a node has already been explored. In the root node, we order the entries of x in order of increasing magnitude of the entries of the unconstrained solution. Let x^* be the optimal solution of the NNLS problem (1.2), we reorder the entries of x^* so that $x_1^* \leq x_2^* \leq \cdots \leq x_n^*$.

Bound. As an additional entry of x is constrained to be zero for each child node, the error of a given node will always be greater than or equal to the error of its parent, by construction. As soon as we reached a leaf of the tree, we obtain a feasible solution whose error is therefore an upper bound for (1.3). Hence if a given node has an error greater than this upper bound, all the children of this node will also have a higher error, and it can be safely pruned. A key element of a branch-and-bound algorithm is the strategy used to explore the tree. A relevant exploration can quickly lead to a good admissible solution, and thus to a bound allowing the pruning of large parts of the search space. As described above, in Arborescent, the solution x^* to the unconstrained problem is computed in the root node and the components of x^* are sorted in ascending order. Thereafter, the exploration is done depth-first, and "left-first". Among all children of a node, the node furthest left is explored first, that is, the components that are closest to zero in the unconstrained solution are constrained first. This is based on the hypothesis that if a component is close to zero in the solution of the unconstrained problem, it is more likely to be zero in the optimal solution of the constrained problem. We have observed empirically that this approach outperforms more complex strategies, including greedy node selection (that is, selecting the node with the lowest error).

Pseudocode. Arborescent is detailed in Algorithm 3.1. The set P is the pool of nodes. On line 5, it is initialized with the root node, that has a full support (no component is constrained). On line 7, a node is selected in P, following the strategy described above, and removed from P on line 8. On line 9, the NNLS subproblem defined by A and b limited to the corresponding over-support $\mathcal{K}$ is solved using the initialization x. If the error is larger that the current lowest error, no child node can be optimal, and the node is pruned (line 11). Otherwise, the exploration continues. If the desired sparsity level k is not reached, a new node is generated for every component of the over-support (lines 14 and 15). If the value of k is reached, the error of the current node is compared to the lowest error found so far (line 17), and if it is lower, it replaces it (line 18).

Computational complexity. In the worst case, Arborescent has to build and explore the whole tree, that is, to solve a number of NNLS subproblems equal to $\sum_{l=k}^{r} \binom{r}{l}$. In the best case, all nodes except the leftmost ones are pruned, so the number of NNLS subproblems to solve is $\sum_{l=k}^{r} l$. In practice, the number or nodes explored is far from the worst case; see Section 3.5.2.

Algorithm 3.1: Arborescent

Input: $A \in \mathbb{R}_+^{m \times n}$, $b \in \mathbb{R}_+^m$, $k \in \{1, 2, \ldots, r\}$
Output: $x_{best} = \arg\min_{x \geq 0} ||Ax - b||_2^2$ s.t. $||x||_0 \leq k$

```
Init x_0 ← NNLS(A, b)
Sort the elements in x_0 in increasing order
Init K_0 ← {1, ..., r}
Init best_error ← +∞
Init P ← {(K_0, x_0)}
while P ≠ ∅ do
    (K, x_parent) = P.select()
    P ← P \ {(K, x_parent)}
    (error, x) ← NNLS(A(:, K), b, x_parent(K))
    if error > best_error then
        prune (do nothing)
    else
        if size(K) > k then
            foreach i ∈ K do
                P ← P ∪ {(K \ {i}, x) }
        else
            if error < best_error then
                best_error ← error
                x_best ← x
return x_best
```

3.4 Biobjective extension

Although Arborescent was designed to solve the k-sparse NNLS problem, it can be easily extended to compute the whole Pareto front for the biobjective problem (1.4), that we remind here,

$$\min_{x \geq 0} \{\|Ax - b\|_2^2, \|x\|_0\}.$$

Indeed, while exploring the search tree and computing intermediary nodes, it also computes automatically the optimal k'-sparse solutions for all $k' \in \{k, \dots, r\}$. Our extension (that we call Arbo-Pareto) therefore consists in maintaining a list of the optimal k'-sparse solutions, making a comparison for every node explored, and updating the list when a better solution is found. This almost does not affect the computational cost of the algorithm as the cost of comparison and update is negligible compared to the cost of the evaluation of a node.

To show that Arborescent indeed computes these solutions, we prove by contradiction that it cannot prune an optimal k'-sparse solution. Suppose the node γ is the optimal k'-sparse solution for a given $k' > k$, and that it is pruned by Arborescent. Because it is pruned, its error must be larger than the error of some feasible k-sparse solution α, $f^*(\gamma) > f^*(\alpha)$. However, there exists necessarily a parent node of α that is k'-sparse; we call it β. By construction, $f^*(\alpha) > f^*(\beta)$, so we have $f^*(\gamma) > f^*(\beta)$, meaning that γ is not the optimal k'-sparse solution. This contradicts the hypothesis.

Arbo-Pareto is detailed in Algorithm 3.2. NNLS refers to the active-set method described above. The set P is the pool of nodes, initialized with the root node (with no entry constrained) on line 4. A node is selected from P on line 8, and removed from P on line 9. On line 10, the NNLS subproblem restricted to the over-support $\mathcal{K}$ is solved using the parent solution as initialization. If the error at the current node is worse than the current best feasible solution, then no descending node can be optimal, and we prune the current node (line 13). Otherwise, we continue the exploration. If the sparsity target k is not reached, we generate one node for every entry of the over-support (lines 16 and 17). We then compare the error of the current node with the error of the current best k'-sparse solution, on line 18. If it is lower, we update this error (line 19) and the Pareto front (line 20).

Note that, if $k = 1$, Arbo-Pareto computes the whole Pareto front. Otherwise, it only computes the part with $k' \in \{k, \dots, r\}$. Arbo-Pareto can be used on its own to solve biobjective k-sparse NNLS problems. In Chapter 4, we present its use as a subroutine in a MNNLS algorithm with a matrix-wise sparsity constraint.

3.5 Experiments

The code of Arborescent and the test scripts are provided in an online repository[2]. All experiments were performed on a personal computer with an i5-1135G7 processor, with a clock frequency of 2.40GHz. All algorithms are implemented in Julia.

Experiments concerning the biobjective extension of Arborescent can be found in Chapter 4.

[2] `http://gitlab.com/nnadisic/giant.jl`

Algorithm 3.2: Arbo-Pareto

Input: $A \in \mathbb{R}_+^{m \times r}$, $b \in \mathbb{R}_+^m$, $k \in \{1, 2, \ldots, n\}$
Output: Pareto front $\mathcal{S}$

Init $\mathcal{K}_0 \leftarrow \{1, ..., r\}$
Init $x_0 \leftarrow \text{NNLS}(A, b)$
Sort the entries in x_0 in ascending order
Init $P \leftarrow \{(\mathcal{K}_0, x_0)\}$
Init $\mathcal{E}_i \leftarrow +\infty$ for all $i \in \{k, \ldots, r\}$
Init $\mathcal{S}_i \leftarrow \vec{0}$ for all $i \in \{k, \ldots, r\}$
while $P \neq \emptyset$ **do**
    $(\mathcal{K}, x_{parent}) = P.\text{select}()$
    $P \leftarrow P \setminus \{(\mathcal{K}, x_{parent})\}$
    $x, error \leftarrow \text{NNLS}(A(:, \mathcal{K}), b, x_{parent}(\mathcal{K}))$
    $k' = size(\mathcal{K})$
    if $error > \mathcal{E}_k$ **then**
        prune (do nothing)
    else
        if $k' > k$ **then**
            foreach $i \in \mathcal{K}$ **do**
                $P \leftarrow P \cup \{(\mathcal{K} \setminus \{i\}, x)\ \}$
        if $error < \mathcal{E}_{k'}$ **then**
            $\mathcal{E}_{k'} \leftarrow error$
            $S_{k'} \leftarrow x$

3.5.1 Comparison on synthetic datasets

We first consider synthetic k-sparse NNLS problems for which we compare Arborescent to three algorithms:

- A homotopy algorithm, that tackles the ℓ_1-penalized problem (1.6). It is described in more detail in Section 4.2.2. In a nutshell, the homotopy algorithm is a heuristic that computes a set of solutions representing different tradeoffs between reconstruction error and sparsity. Among these solutions, we choose the best k-sparse one, and we debias it by running an NNLS algorithm restricted to its support.
- Nonnegative OMP (NNOMP) [65], a standard greedy algorithm for k-sparse NNLS.
- CPLEX[3], a generic MIP solver that solves the k-sparse NNLS problem as formulated in Section 3.2. The "big-M" parameter is set to 1.1, which is reasonable given that $0 \leq x_i \leq 1$ for all i in the synthetic data we generate, see below. Theoretically, CPLEX should always find the same solution as Arborescent, as both methods solve the problem exactly.

Synthetic test cases are built by generating a random matrix $A \in \mathbb{R}_+^{m \times r}$ and a random k-sparse vector $x_{true} \in \mathbb{R}_+^r$, computing $b = Ax_{true}$, and trying to find again x_{true} with A, b and k as parameters of the sparse NNLS algorithm. We consider four cases: well-conditioned A and noiseless b, well-conditioned A and noisy b, ill-conditioned A and noiseless b, ill-conditioned A and noisy b. For well-conditioned A, each entry of A is generated using the uniform distribution in [0,1] (`rand(m,n)` in Matlab). For ill-conditioned A, we proceed in the same way, then compute the SVD of $A = U\Sigma V^T$, replace Σ by values between 10^{-4} and 1 equally spaced in a logscale (`logspace(-4,0,n)` in Matlab), and finally take $A = U\Sigma V^T$. For x, we also use the uniform distribution in [0,1]. For the noise e added to b, we generate a vector where each entry is generated using the normal distribution of mean 0 and variance 1 (`randn(m,1)` in Matlab), then rescale $e \leftarrow 0.05\frac{e}{||e||_2}||b||_2$ so that $||e||_2 = 0.05||b||_2$ (the noise level is 5%). We generate such data sets for three values of m: 1000, 100, and 10, with fixed $r = 10$ and $k = 6$, for a total of 12 test settings. For each setting, 100 data sets are randomly generated, and processed by the 4 algorithms. For each algorithm, we measure the median of the relative error $\frac{||Ax-b||_2}{||b||_2}$ over the 100 runs, as well as the median computational time, and the number of *successes*, defined as the number of times the support of x_{true} is recovered.

Tables 3.1 to 3.3 present the results of the experiments. As expected, the exact methods Arborescent and CPLEX always produce the same results, and they have 100% of success and a zero error in all noiseless settings. This is particularly interesting when A is ill-conditioned, because then the heuristics homotopy and NNOMP often fail to identify the support. For noisy data, Arborescent and CPLEX always

[3]See `http://cplex.com`. Although it is a closed-source commercial software, it is available for free for academics.

	Well-cond A, Noiseless b			Well-cond A, Noisy b			Ill-cond A, Noiseless b			Ill-cond A, Noisy b		
	Err.	Time	Succ.	Err.	Time	Succ.	Err.	Time	Succ.	Err.	Time	Succ.
Homotopy	0	0.11	100	5.03	0.14	48	0	0.19	73	5.08	0.23	36
NNOMP	0	0.21	100	4.98	0.21	97	3.18	0.21	13	5.94	0.22	8
CPLEX	0	8.38	100	4.98	9.31	99	0	5.27	100	4.97	6.69	88
Arbo.	0	1.16	100	4.98	2.8	99	0	1.35	100	4.97	3.43	88

Table 3.1: Results for $m = 1000$. Err. is the relative error $\|Ax - b\|_2 / \|b\|_2$ in percent. Time is in milliseconds. Succ. is the number of successes, that is, the number of times the algorithm recovered the support of x_{true} over 100 runs.

	Well-cond A, Noiseless b			Well-cond A, Noisy b			Ill-cond A, Noiseless b			Ill-cond A, Noisy b		
	Err.	Time	Succ.	Err.	Time	Succ.	Err.	Time	Succ.	Err.	Time	Succ.
Homotopy	0	0.11	100	4.98	0.12	30	0	0.14	79	5.16	0.19	24
NNOMP	0	0.21	100	4.85	0.2	78	3.4	0.19	14	5.85	0.2	2
CPLEX	0	3.37	100	4.85	4.27	80	0	2.89	100	4.83	4.31	70
Arbo.	0	0.97	100	4.85	1.68	80	0	0.97	100	4.83	2.45	70

Table 3.2: Results for $m = 100$. Measures are defined as in table 3.1.

	Well-cond A, Noiseless b			Well-cond A, Noisy b			Ill-cond A, Noiseless b			Ill-cond A, Noisy b		
	Err.	Time	Succ.	Err.	Time	Succ.	Err.	Time	Succ.	Err.	Time	Succ.
Homotopy	0	0.1	83	3.62	0.17	16	0	0.13	70	3.15	0.17	14
NNOMP	1.63	0.13	24	3.94	0.2	11	2.47	0.16	4	4.19	0.19	2
CPLEX	0	2.43	100	2.66	4	36	0	2.6	100	2.57	3.46	19
Arbo.	0	0.75	100	2.66	2.18	36	0	1.26	100	2.57	1.93	19

Table 3.3: Results for $m = r = 10$. Measures are defined as in table 3.1.

perform better than the heuristics. When $m = 100$ with ill-conditioned A and noisy b, they still show 70% of success while the heuristics have respectively 24% and 2%. When $m = r = 10$, the success rate is low but always better than the heuristics. The homotopy seems to handle ill-conditioned A better than NNOMP, while NNOMP handles noise better. In all settings, Arborescent is faster than CPLEX in producing the same result, and it is slower than the heuristics.

The experiments clearly show the advantages of using an exact method over a heuristic, that is, to recover the true solution more often, especially with ill-conditioned and noisy data. The tradeoff is a higher computational cost. They also show the advantage of Arborescent over a generic solver such as CPLEX, that is a faster computation to produce the same exact result.

3.5.2 Scalability of Arborescent

To test the scalability of Arborescent compared to CPLEX and to the brute-force approach, we perform two new tests with the same data model as in Section 3.5.1, in the well-conditioned and noiseless case, with fixed $m = 1000$. The brute-force approach consists in generating all the $\binom{r}{k}$ possible supports for the given k, solve the corresponding NNLS subproblem for each support, and keep the solution with the smaller reconstruction error. For a given test, we measure the computing time as the median time over 10 runs; note that for a given setting all 3 algorithms produce the same solution.

First, we vary r between 4 and 50, and run each k-sparse algorithm with $k = r/2$,

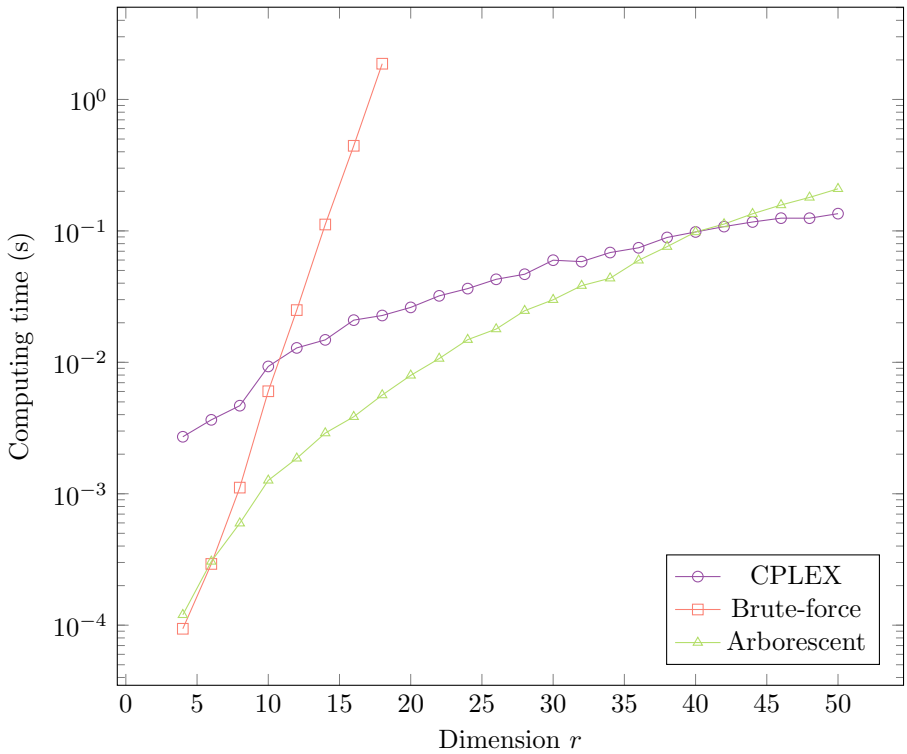


Figure 3.2: Evolution of the computing time for different values of r, with $k = r/2$, and fixed $m = 1000$. Measures for brute-force stop after $r = 18$

see Fig. 3.2. The computing time of the brute-force approach grows exponentially with r. The computing time of Arborescent also grows much slower than brute-force, meaning that our branch-and-bound strategy is able to prune the search tree efficiently. For $r < 40$, Arborescent is faster than CPLEX, but the curves cross at $r = 40$, meaning that for higher dimensions the search tree is too deep and our branch-and-bound strategy is not adapted anymore. Interestingly, for $r \leq 10$, even the brute-force approach is faster than CPLEX. This can be explained by the fixed costs of the solver CPLEX, that become non-negligible when the number $\binom{r}{k}$ of supports to explore is very small.

Second, we fix $r = 20$ and vary k between 2 and 20, see Fig. 3.3. The brute-force approach shows a curve directly proportional to the number $\binom{r}{k}$ of supports to explore. Arborescent is faster than brute force, except for k equals to 2, 19, and 20. In those cases, $\binom{r}{k}$ is so small that brute force is fast enough. Arborescent is also always faster than CPLEX. Its computing time does not vary much with k (except for $k > 18$ where it becomes very fast), showing again its good pruning capabilities.

In the application we focus on, hyperspectral unmixing, the dimension r is often smaller than 10 and rarely larger than 20. Therefore, our approach Arborescent

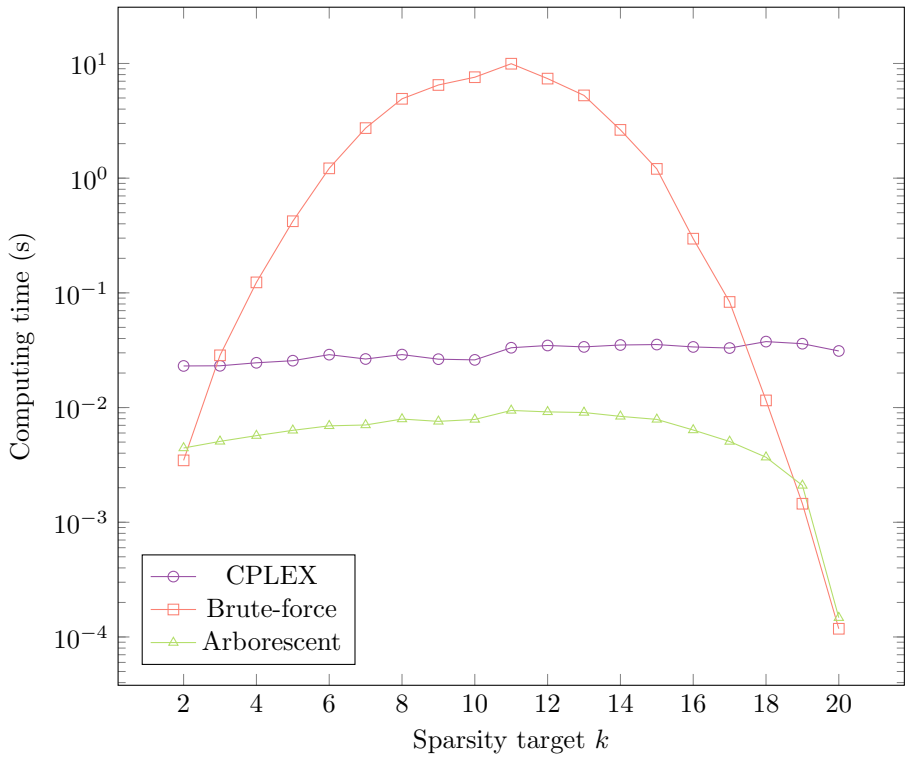


Figure 3.3: Evolution of the computing time for different values of k, with fixed $m = 1000$ and $r = 20$.

is more adapted than the generic solver CPLEX. In other applications with larger dimensions, it may not always be the case.

3.5.3 Application on hyperspectral unmixing

In this section, we use Arborescent to identify the materials present in the pixels of a hyperspectral image. We consider 3 widely used hyperspectral images, Jasper, Samson, and Urban with precomputed dictionaries[4] [85], see Table 3.4 for details. Given $B \in \mathbb{R}^{m\times n}$ and $A \in \mathbb{R}^{m\times r}$, we compute $X \in \mathbb{R}^{r\times n}$ by solving the following MNNLS problem with a column-wise k-sparse constraint

$$\min_X \|B - AX\|_F \quad \text{s.t.} \quad X \geq 0, \text{ and } \|X(:,j)\|_0 \leq k \text{ for all } j.$$

This reduces to solving n independent k-sparse NNLS subproblems of the form (1.3) with each subproblem corresponding to one pixel. We solve them with the four sparse methods presented in Section 3.5.1, and also with an active-set method (noted AS)

[4]Downloaded from `http://lesun.weebly.com/hyperspectral-data-set.html`

Dataset	m	n	r	k
Jasper	198	$100 \times 100 = 1000$	4	2
Samson	156	$95 \times 95 = 9025$	3	2
Urban	162	$307 \times 307 = 94249$	6	2

Table 3.4: Summary of the datasets, for which the data is $B \in \mathbb{R}^{m \times n}$ and the dictionary is $A \in \mathbb{R}^{m \times r}$. We compute $X \in \mathbb{R}^{r \times n}$ such that each column $X(:, j)$ is k-sparse.

	Algorithm	AS	Homotopy	NNOMP	CPLEX	**Arbo.**
Jasper	Time	0.34	0.45	0.39	20.21	1.7
	Error	5.71	6.99	7.49	5.94	5.94
	Sparsity	2.23	1.78	1.72	1.81	1.81
Samson	Time	0.18	0.19	0.31	16.34	0.68
	Error	3.3	3.34	6.76	3.34	3.34
	Sparsity	2.19	1.85	1.6	1.85	1.85
Urban	Time	5.93	4.68	3.29	231.58	52.22
	Error	7.67	8.62	8.97	7.89	7.89
	Sparsity	2.62	1.9	1.7	1.9	1.9

Table 3.5: Results of the unmixing of hyperspectral images. Time is the median running time over 10 runs, in seconds. Error is the relative error $||B - AX||_F / ||B||_F$ in percent. Sparsity is the average k-sparsity per column.

that solves the NNLS problems without sparsity constraint, to serve as a baseline. We set the "big-M" constant required by CPLEX as $M = 1.5\|x_{\text{ho}}\|_{\text{inf}}$, where x_{ho} is the k-sparse solution computed by the homotopy.

For each algorithm, we report the running time (median over 10 runs), the relative reconstruction error $\frac{||B-AX||_F}{||B||_F}$, and the average column-wise sparsity. Table 3.5 shows the results of the unmixing of the 3 hyperspectral images. We first note that, even without sparsity constraint, the solutions computed are already sparse. Also, by imposing a column-wise k-sparsity of 2, the sparse methods generate solutions with an average column-wise sparsity smaller than 2, meaning that some columns are naturally 1-sparse. The heuristics homotopy and NNOMP enforce sparsity at the cost of a larger reconstruction error, and their speed is on the same order as AS. The exact methods are able to enforce sparsity while limiting the error increase. They are slower than the heuristics, but Arborescent is significantly faster than CPLEX. For Samson, the homotopy produces a solution with the same error and sparsity as the exact methods, meaning that the problem is well-conditioned and "easy" enough so that this heuristic can recover the true solution, or a very close one.

3.6 Conclusion

We proposed Arborescent, a dedicated branch-and-bound algorithm to tackle the k-sparse nonnegative least squares problem (1.3) exactly. It works in very general settings, where existing approaches such as ℓ_1-penalized or greedy algorithms fail to identify the support of the optimal solution. The branch-and-bound technique allows for substantial pruning of the search space, so this combinatorial problem can be solved exactly with a reduction of computation time over a generic solver. The combination of relative speed, scalability, and an exact solution provide a broadly applicable technique for medium-scale sparse nonnegative least squares problems.

We also proposed Arbo-Pareto, a biobjective extension of Arborescent. It is able to compute the Pareto front of the biobjective sparse NNLS problem (1.4), that is compute a set a solutions with different tradeoffs between error and sparsity. It uses the fact that when Arborescent compute a k-sparse solution to a given NNLS problem, it also computes as a side-effect the k'-sparse solutions for $k' \in \{k, k+1, \ldots, r\}$. This extension will be central in the matrix-wise sparsity-constrained approach presented in the next chapter.

In practice, computing time is rarely critical in hyperspectral unmixing tasks, as they are usually performed offline and once and for all. Also, such images are often taken from airplanes and require specific and expensive hardware, so the acquisition cost is usually very high and dominates the unmixing cost. Therefore, it is worth spending more computing time in exchange for a guaranteed result, and the end-user is assured that the potential residual error comes from model misfit or acquisition noise, and not from the optimization process.

Chapter 4

Matrix-wise ℓ_0-constrained NNLS

"There's a fine line between fishing and just standing on the shore like an idiot."

– Steven Wright

In this chapter, we focus on the sparse MNNLS problem, that we remind here,

$$\min_X \|B - AX\|_F^2 \quad \text{such that} \quad X \geq 0 \text{ and } X \text{ is sparse.}$$

As opposed to most previous works that enforce sparsity of X column- or row-wise, we introduce a novel formulation with a matrix-wise sparsity constraint. Then, we present a two-step algorithm to tackle this problem. The first step divides sparse MNNLS in subproblems, one per column of the original problem. It then uses different algorithms to produce, either exactly or approximately, a Pareto front for each subproblem, that is, to produce a set of solutions representing different tradeoffs between reconstruction error and sparsity. The second step selects solutions among these Pareto fronts in order to build a sparsity-constrained matrix that minimizes the reconstruction error. We perform experiments on facial and hyperspectral images, and we show that our proposed two-step approach provides more accurate results than state-of-the-art sparse coding heuristics applied both column-wise and globally.

The content of this chapter is mainly extracted from [57]:

57. Nadisic, N., Cohen, J. E., Vandaele, A. & Gillis, N. Matrix-wise ℓ_0-constrained sparse nonnegative least squares. *preprint arXiv:2011.11066* (2022).

4.1 Introduction

In Section 1.4 we saw that in NNLS problems with multiple right-hand sides, sparsity is often enforced column- or row-wise. For example, a column-wise k-sparsity

constraint lead to Problem (1.8) that we remind here,

$$\min_X \|B - AX\|_F^2 \quad \text{such that} \quad X \geq 0 \text{ and } \|X(:,j)\|_0 \leq k \text{ for all } j.$$

However, in some applications, setting the sparsity parameter k is tricky, as the relevant value can vary for different columns. For example, in hyperspectral unmixing, pixels are often composed of different numbers of materials. Therefore, we consider a more global approach, that we coin as *matrix-wise q-sparse MNNLS*:

$$\min_X \frac{1}{2}\|B - AX\|_F^2 \quad \text{such that} \quad X \geq 0 \text{ and } \|X\|_0 \leq q, \tag{4.1}$$

where $\|X\|_0 = \sum_j \|X(:,j)\|_0$ and q is a matrix-wise sparsity parameter, hence enforcing an *average* sparsity q/n for the columns of X.

Note that (4.1) could theoretically be solved by any column-wise k-sparse NNLS algorithm, because (4.1) is equivalent to the vectorized form

$$\min_{\bar{x}} \|\operatorname{vec}(B) - \underbrace{(A \otimes I)}_{\Omega} \bar{x}\|_2^2 \quad \text{such that} \quad \bar{x} \geq 0 \text{ and } \|\bar{x}\|_0 \leq q, \tag{4.2}$$

where $\otimes$ is the Kronecker product, I is the identity matrix of appropriate dimension, and $\operatorname{vec}(B)$ denotes the column vector obtained by stacking the columns of B on top of one another. Problem (4.2) is a k-sparse NNLS problem, but in practice its dimensions make it difficult to solve directly. Denoting $\Omega = A \otimes I$, we have $\Omega \in \mathbb{R}^{(mn)\times(rn)}$, which is particularly problematic in hyperspectral unmixing where the dimension n can reach tens of thousands.

It is possible to implement some k-sparse NNLS algorithms in a non-naive way to solve (4.2) efficiently without actually allocating Ω, and we detail such implementation of a greedy algorithm in Section 4.3.2. However, even in this case, when n is large then the problem to solve is huge and the computing time can become too high, see Section 4.4 for an experimental illustration.

Related work Most approaches that tackle sparse MNNLS were actually introduced in the context of sparse NMF, see our brief review in Section 1.7. Some similar models have been studied, such as simultaneous sparse approximation [74, 79], where X is constrained to be block-sparse, that is to have sparse columns sharing the same support. The assumptions of these models, the algorithms to solve them, and their applications are far from our focus, so detailing them is out of the scope of this work.

Contribution and outline of the chapter The main goal of this work is to describe a novel method able to solve efficiently the matrix-wise q-sparse MNNLS problem (4.1), even in large dimensions. This method can be summarized by two main steps:

1. Problem (4.1) is divided in n subproblems of the form (1.4) and, for each of them, the Pareto front is computed with existing algorithms.

2. One solution per column (hence per Pareto front) is selected to build a solution to Problem (4.1), that is, a q-sparse matrix. This combinatorial step is solved exactly with a dedicated algorithm.

To the best of our knowledge, this work is the first to tackle specifically Problem (4.1). Note that the algorithms used in the first step are not original contributions. The contributions lie rather in the use of these existing algorithms to generate the Pareto fronts of the subproblems, and the combination of these fronts with a novel algorithm to obtain a q-sparse matrix.

This chapter is organized as follows. In Section 4.2, we detail the three algorithms used to generate Pareto fronts. In Section 4.3, we present the main contribution of this work, that is, an algorithm to solve Problem (4.1). We illustrate the effectiveness of our proposed method with experiments on real-life facial and hyperspectral image datasets in Section 4.4, and conclude in Section 4.5.

4.2 Computing the Pareto fronts

In the following, we rely on three types of sparse NNLS algorithms to generate Pareto fronts, that is, to solve Problem (1.4), that we remind here,

$$\min_{x \geq 0} \{\|Ax - b\|_2^2, \|x\|_0\}.$$

The branch-and-bound algorithm Arborescent and its biobjective extension have been detailed in Chapter 3. Greedy and homotopy algorithms are discussed below.

4.2.1 Greedy algorithms

Greedy algorithms are one of the most popular approaches for solving approximately the k-sparse NNLS problem (1.3), and they are briefly described in Section 1.2. Interestingly, because they select components one after the other, greedy algorithms can be used as a proxy to compute an approximation of the Pareto front of the corresponding biobjective sparse NNLS problem. Indeed, the solution at the i-th iteration of the algorithm is i-sparse. By running the algorithm with a sparsity target k, we also compute, as a side effect, some k'-sparse solutions for $k' \in \{1, \ldots, k\}$. Therefore, to obtain an approximation of the Pareto front, it suffices to return all intermediate solutions instead of only the final one.

In this chapter, we will only focus on Nonnegative OMP (NNOMP) for conciseness, but this approach could be easily generalized to similar greedy algorithms, see Section 1.2. NNOMP was first introduced by [16], but many variants exist, and implementations details can have a significant impact on the performance of this algorithm; we refer the interested reader to [65].

4.2.2 Homotopy algorithm

Sparse methods based on the ℓ_1-norm are very popular. They are discussed in Section 1.3 and we remind here the formulation of the ℓ_1-penalized NNLS problem (1.6),

$$\min_x \|Ax - b\|_2^2 + \lambda \|x\|_1 \quad \text{such that} \quad x \geq 0.$$

The parameter λ is a nonnegative constant controlling the tradeoff between error and sparsity. Choosing a suitable value for λ for a given application is not straightforward, as there are no explicit relation between this value and the actual sparsity of the solution.

To overcome this issue, *homotopy* algorithms have been introduced. They generate the full *regularization path* of a given ℓ_1-penalized NNLS problem, that is, the set of the solutions for all possible values of λ. They allow the user to choose the relevant solution within this path, each solution representing a different trade-off between sparsity and reconstruction error. The first homotopy algorithm has been introduced by [66], for sparse least squares with no nonnegativity constraint. Variants have been developed by [22] and [21]. [44] introduced a variant to deal specifically with the nonnegativity constraint.

In a nutshell, the homotopy algorithm uses the KKT conditions (necessary conditions for optimality) to first find the value $\lambda_{\max}$ for which the optimal solution of ℓ_1-penalized NNLS is $x = 0$ for any $\lambda \geq \lambda_{\max}$, and then to compute the next smaller values of λ for which the support (that is, the set of non-zero entries) of the optimal solution changes (one zero entry becomes non-zero, or the other way around). This is similar in spirit to active-set methods, akin to the simplex algorithm for linear programming. These values of λ are called *breakpoints*, between which the support of the optimal solution does not change; see Figs. 4.1 and 4.2 for an illustration.

The strength of the homotopy algorithm is to generate the full *regularization path*, that is, the set of optimal solution for all possible values of λ, for the same cost as a standard active-set algorithm. The solutions in this path represent different tradeoffs between reconstruction error and sparsity, and as such this path can be seen as an approximation of the Pareto front in Problem (1.4).

Although the homotopy algorithm is not an original contribution, we include below its detailed description and development. This is not necessary to understand our contribution, and the reader focusing on our proposed algorithm for matrix-wise q-sparse MNNLS can skip to Section 4.3 (page 86).

Optimality conditions

The homotopy algorithm uses the first-order optimality conditions, that is, the KKT conditions, to determine the breakpoints and the supports of the corresponding solutions. Because x is nonnegative, we have $\|x\|_1 = \sum_i x_i = e^T x$, where e is the vector of all ones whose dimension will be clear from context. Therefore, the ℓ_1-penalized NNLS problem can be written as follows

$$\min_{x \geq 0} f(x), \text{ where } f(x) = \frac{1}{2}\|Ax - b\|_2^2 + \lambda e^T x. \tag{4.3}$$

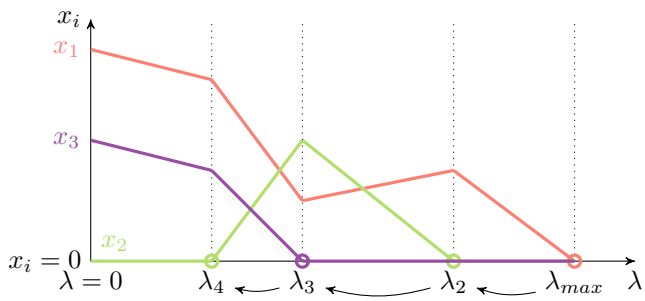


Figure 4.1: Example of a solution path of a homotopy algorithm, depending on λ, for a problem of 3 variables. Vertical dotted lines correspond to breakpoints.

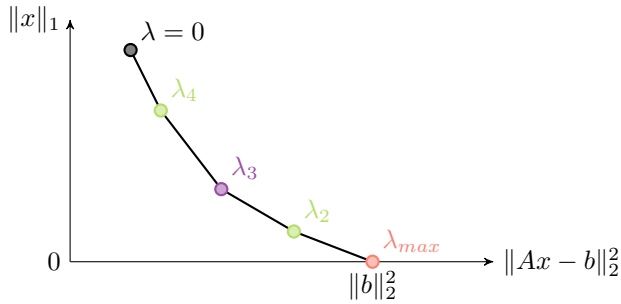


Figure 4.2: Trade-off between the ℓ_1-norm and the error $\|Ax - b\|_2^2$ corresponding to the solution path in Fig. 4.1.

The KKT conditions are $x \geq 0$,

$$\nabla f(x) = \underbrace{A^\top A}_{P} x - \underbrace{A^\top b}_{\ell} + \lambda e \leq 0, \tag{4.4}$$

$$x_i (A^\top A x - A^\top b + \lambda e)_i = 0 \text{ for all } i. \tag{4.5}$$

Eq. (4.5) is the complementary condition; for every entry of x, either the entry itself or the corresponding gradient entry is equal to zero. To simplify the notation, let us define $P = A^\top A$ and $\ell = A^\top b$.

As f is a convex function and the feasible set contains a Slater point (e.g., $x = e$), the KKT conditions are necessary and sufficient. Therefore, any solution x satisfying them is optimal. Suppose we know the optimal support, that is, the set $\mathcal{K}$ such that $x(\mathcal{K}) > 0$ and $x(\bar{\mathcal{K}}) = 0$, where $\bar{\mathcal{K}} = \{1, 2, \ldots, n\} \setminus \mathcal{K}$. Then

$$\begin{aligned} x(\mathcal{K}) > 0 &\Rightarrow P(\mathcal{K}, \mathcal{K}) x(\mathcal{K}) - \ell(\mathcal{K}) + \lambda e = 0 \\ &\Rightarrow x(\mathcal{K}) = P(\mathcal{K}, \mathcal{K})^{-1} (\ell(\mathcal{K}) - \lambda e) \geq 0, \end{aligned} \tag{$\mathcal{C}1$}$$

and

$$Px - \ell + \lambda e \geq 0 \Rightarrow P(\bar{\mathcal{K}}, \mathcal{K}) x(\mathcal{K}) - \ell(\bar{\mathcal{K}}) + \lambda e \geq 0. \tag{$\mathcal{C}2$}$$

Replacing $x(\mathcal{K})$ in $(\mathcal{C}2)$ by $(\mathcal{C}1)$, we have

$$P(\bar{\mathcal{K}},\mathcal{K})[P(\mathcal{K},\mathcal{K})^{-1}(\ell(\mathcal{K})-\lambda e)]-\ell(\bar{\mathcal{K}})+\lambda e\geq 0.$$

Let us simplify the notation. Let

$$a_{\mathcal{K}}=P(\mathcal{K},\mathcal{K})^{-1}\ell(\mathcal{K}),\ \text{and}\ b_{\mathcal{K}}=P(\mathcal{K},\mathcal{K})^{-1}e. \tag{4.6}$$

We can rewrite $(\mathcal{C}1)$ as

$$a_{\mathcal{K}}-\lambda b_{\mathcal{K}}\geq 0. \tag{$\mathcal{C}1b$}$$

Note that the dimension of $a_{\mathcal{K}}$ and $b_{\mathcal{K}}$ is the cardinality of $\mathcal{K}$. Let

$$c_{\mathcal{K}}=P(\bar{\mathcal{K}},\mathcal{K})a_{\mathcal{K}}-\ell(\bar{\mathcal{K}}),\ \text{and}\ d_{\mathcal{K}}=P(\bar{\mathcal{K}},\mathcal{K})b_{\mathcal{K}}-e.$$

We can rewrite $(\mathcal{C}2)$ as

$$c_{\mathcal{K}}-\lambda d_{\mathcal{K}}\geq 0. \tag{$\mathcal{C}2b$}$$

Note that the dimension of $c_{\mathcal{K}}$ and $d_{\mathcal{K}}$ is the cardinality of $\bar{\mathcal{K}}$. Moreover, Equations $(\mathcal{C}1b)$ and $(\mathcal{C}2b)$ are linear in λ: Given $\mathcal{K}$, we can easily compute $a_{\mathcal{K}}$, $b_{\mathcal{K}}$, $c_{\mathcal{K}}$, $d_{\mathcal{K}}$.

Algorithm description

The goal of the homotopy algorithm is to compute breakpoints and their corresponding supports. It starts with an empty support, corresponding to the zero vector for any $\lambda\geq\lambda_{max}$, and iteratively adds or removes entries to the support while decreasing the value of λ.

The first step to build the regularization path is to find the first breakpoint $\lambda_{\max}$, that is, the minimum value of λ such that the solution is the zero vector. If the optimal solution is $x=0$, that is, $\mathcal{K}=\varnothing$ and $\bar{\mathcal{K}}=\{1,2,\ldots,n\}$, then from Eq. (4.4) we have $\lambda\geq\max_i \ell_i$. Therefore, the first breakpoint is

$$\lambda_{\max}=\max_i \ell_i=\max_i (A^\top b)_i=\max_i A(:,i)^T b.$$

The index $i_1=\arg\max_i \ell_i$ is the first to enter the support[1], thus for $\lambda_2\leq\lambda<\lambda_{\max}$ we have $\mathcal{K}=\{i_1\}$.

From a given support $\mathcal{K}_j$, the next breakpoint $\lambda_{j+1}\leq\lambda_j$ is the largest value of λ that violates one of the conditions $(\mathcal{C}1)$ or $(\mathcal{C}2)$. If $(\mathcal{C}1)$ is violated then a variable will leave the support, that is, a positive entry will become zero. Denoting k^* the index of this entry, we have $\mathcal{K}_{j+1}=\mathcal{K}_j\setminus\{k^*\}$. If $(\mathcal{C}2)$ is violated then a variable will enter the support, that is, a zero entry will become positive, $\mathcal{K}_{j+1}=\mathcal{K}_j\cup\{k^*\}$.

Let us consider $(\mathcal{C}1)$. We have $a_{\mathcal{K}}(k)-\lambda b_{\mathcal{K}}(k)\geq 0$ for all k so

$$\lambda_{j+1}\geq\max_{\{k|b_{\mathcal{K}}(k)<0\}}\frac{a_{\mathcal{K}}(k)}{b_{\mathcal{K}}(k)}.$$

[1]If two of more columns maximize the value of ℓ_i, we have to pick one to start the homotopy. If we always pick the one with smallest index, then the algorithm behaves normally, except that at the next iteration we will have $\lambda_{j+1}=\lambda_j$. This is similar to Bland's rule for the simplex algorithm.

Similarly, for $(\mathcal{C}2)$ we have $c_{\mathcal{K}}(k) - \lambda d_{\mathcal{K}}(k) \geq 0$ for all k so

$$\lambda_{j+1} \geq \max_{\{k|d_{\mathcal{K}}(k)<0\}} \frac{c_{\mathcal{K}}(k)}{d_{\mathcal{K}}(k)}.$$

Therefore,

$$\lambda_{j+1} = \max \left(\underbrace{\max_{\{k|b_{\mathcal{K}}(k)<0\}} \frac{a_{\mathcal{K}}(k)}{b_{\mathcal{K}}(k)}}_{\text{Case 1}}, \underbrace{\max_{\{k|d_{\mathcal{K}}(k)<0\}} \frac{c_{\mathcal{K}}(k)}{d_{\mathcal{K}}(k)}}_{\text{Case 2}} \right).$$

The algorithm is detailed formally in Algorithm 4.1. Note that, inside the algorithm loop, once a support $\mathcal{K}$ is identified, getting the corresponding optimal solution is straightforward. From Eq. (4.6), if $a_{\mathcal{K}}$ is nonnegative, then the unbiased optimal solution of the NNLS problem is the vector x^* such that $x^*(\bar{\mathcal{K}}) = 0$ and $x^*(\mathcal{K}) = a_{\mathcal{K}}$. If $a_{\mathcal{K}}$ has negative entries, then we can compute the unbiased solution with a standard NNLS solver, with $x^*(\bar{\mathcal{K}}) = 0$ and $x^*(\mathcal{K}) = \arg\min_{x \geq 0} \|P(\mathcal{K}, \mathcal{K})x - l(\mathcal{K})\|_2^2$.

Algorithm 4.1: Homotopy algorithm for sparse NNLS

Input: $A \in \mathbb{R}^{m \times r}$, $b \in \mathbb{R}^m$
Output: Breakpoints λ_j, corresponding supports $\mathcal{K}_j$ and solutions x_j^*, for all j

$i_1 \leftarrow \arg\max_i \ell_i$
$K \leftarrow \{i_1\}$; $\lambda_1 = \lambda_{\max} = \ell_{i_1}$, $j \leftarrow 1$
while $\lambda_j > 0$ **do**
  $j \leftarrow j + 1$
  Compute a_K, b_K, c_K, d_K
  $\lambda_{C1} \leftarrow \max_{\{k|b_{\mathcal{K}}(k)<0\}} \frac{a_{\mathcal{K}}(k)}{b_{\mathcal{K}}(k)}$, and $k^*_{C1} \leftarrow$ corresponding argmax
  $\lambda_{C2} \leftarrow \max_{\{k|d_{\mathcal{K}}(k)<0\}} \frac{c_{\mathcal{K}}(k)}{d_K(k)}$, and $k^*_{C2} \leftarrow$ corresponding argmax
  if $\lambda_{C1} \geq \lambda_{C2}$ **then**
    $\lambda_j \leftarrow \lambda_{C1}$
    $\mathcal{K} \leftarrow \mathcal{K} \setminus \{k^*_{C1}\}$
  else if $\lambda_{C1} < \lambda_{C2}$ **then**
    $\lambda_j \leftarrow \lambda_{C2}$
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{k^*_{C2}\}$
  $x_j^*(\bar{\mathcal{K}}) \leftarrow 0$
  if $a_{\mathcal{K}} > 0$ **then**
    $x_j^*(\mathcal{K}) \leftarrow a_K$
  else
    $x^*_{\mathcal{K}} = \arg\min_{x \geq 0} \|A(:, \mathcal{K})x - b(\mathcal{K})\|_2^2$

Computational cost

As explained by [44], the time complexity of one iteration of the homotopy algorithm is the same as one iteration of the standard active-set algorithm [46], that is, $\mathcal{O}(r^3)$ operations. It is dominated by the computation of $a_{\mathcal{K}}$, that entails solving a linear system in at most r variables. The number of iterations equals the number of breakpoints, which is in practice similar to those of the active-set. In the worst case, active-set methods might require an exponential number of iterations, up to $\mathcal{O}(2^r)$, as the simplex algorithm for linear programming. However, in practice, we have observed that it typically requires much less iterations, of the order of $\mathcal{O}(r)$. In particular, when P^{-1} is diagonally dominant, we have $b_{\mathcal{K}} > 0$, so $(\mathcal{C}1)$ is never violated. As a result, when λ decreases, no positive entry of x becomes zero. We only add entries to the support, so the homotopy algorithm will be done in at most r iterations. In practice, even when this condition is not met, we have observed that adding entries to the support happens far more often than removing entries.

As the homotopy algorithm solves a series of ℓ_1-penalized NNLS problems, there exist conditions under which it is guaranteed to recover the correct supports, that is, the supports of the solutions of the corresponding ℓ_0-constrained NNLS problems; see [40] and the references therein.

4.3 Solving matrix-wise q-sparse MNNLS

In this section, we study how to tackle matrix-wise q-sparse MNNLS, that is, Problem (4.1). First, we present the key contribution of this work, that is a two-step algorithm to solve Problem (4.1). Then, we show how the greedy algorithm NNOMP can be implemented specifically for Problem (4.1) to avoid the costly reformulation from Eq. (4.2).

4.3.1 A two-step algorithm for matrix-wise q-sparse MNNLS

In this section we present the main contribution of this chapter, that is, a two-step algorithm to solve Problem (4.1). This algorithm is called Salmon[2] and it is detailed in Algorithm 4.2. The motivation behind it is to divide the large matrix-wise problem into small one-column subproblems, solve them, and then combine their solutions to build a global solution.

Step 1 corresponds to lines 1 to 11. It consists in, given the data matrix B and the dictionary A, running an algorithm to generate a Pareto front for every column of B, that is, with input A and $b = B(:\, j)$ for all j. This can be done by any of the three Pareto-front-generating methods presented in Section 4.2; exactly with Arborescent and approximately with NNOMP and the homotopy algorithm. From these fronts, we build a cost matrix C where each column represents a column j of X, each row represents a k-sparsity between 0 and r, and each entry is the reconstruction error of

[2]The name stands for Salmon Applies ℓ_0-constraints Matrix-wise On NNLS problems.

Algorithm 4.2: The algorithm Salmon

Input: Data matrix $B \in \mathbb{R}^{m \times n}$, dictionary $A \in \mathbb{R}^{m \times r}$, sparsity target $q \in \mathbb{N}$
Output: $X^* \in \mathbb{R}^{r \times n}$

Init matrix $C \in \mathbb{R}^{r \times n}$, for all i and j, $C(i,j) \leftarrow \infty$
Init solutions $Sol_{i,j}$ for all i and j
for each $j \in \{1, \ldots, n\}$
  $\{x_t^*\}_{\forall t} \leftarrow$ front_generator(A,$B(:,j)$) `// Arborescent, NNOMP, or homotopy`
  for each t
    $k \leftarrow \|x_t^*\|_0$
    $err \leftarrow \|B(:,j) - Ax_t^*\|_2^2$
    for each $i \in \{k, \ldots, r\}$
      if $err < C(i,j)$ **then**
        $C(i,j) \leftarrow err$
        $Sol_{i,j} \leftarrow x_t^*$
Init cursors, for all $j \in \{1, \ldots, n\}, k_j \leftarrow 0$
Init matrix $\mathcal{G} \in \mathbb{R}^{r \times n}$
for each $i \in \{1, \ldots, r\}, j \in \{1, \ldots, n\}$
  $\mathcal{G}(i,j) \leftarrow (C(0,j) - C(i,j))/i$
while $\sum_j k_j < q$ **and** $\max_{(i,j)} \mathcal{G}(i,j) > 0$ **do**
  $i^*, j^* \leftarrow \operatorname{argmax}_{(i,j)} \mathcal{G}(i,j)$
  $\delta \leftarrow i^* - k_{j^*}$
  $k_{j^*} \leftarrow i^*$
  for each $i \in \{1, \ldots, i^*\}$
    $\mathcal{G}(i,j^*) \leftarrow 0$
  for each $i \in \{i^* + 1, \ldots, r\}$
    $\mathcal{G}(i,j^*) \leftarrow \frac{C(i^*,j^*) - C(i,j^*)}{i - i^*}$
  if $\sum_j k_j > q - r + 1$ **then**
    for each (i,j) such that $i \in \{k_j + 1, \ldots, \min(r, k_j + q - \sum_{l>j} k_l)\}$
      $\mathcal{G}(i,j) \leftarrow 0$
for each $j \in \{1, \ldots, n\}$
  $X^*(:,j) \leftarrow Sol_{k_j,j}$

the k-sparse solution of column j. Formally, for all $i \in \{0, \dots, r\}$ and $j \in \{1, \dots, n\}$,

$$C(i,j) \quad \approx \quad \min_{x \geq 0} \|B(:,j) - Ax\|_2^2 \text{ s.t. } \|x\|_0 \leq i,$$

and $Sol_{i,j}$ stores the corresponding argmin, that is the best i-sparse solution for column j.

Remark 4.1 *The algorithms used to generate the Pareto front in step 1 do not necessarily generate one k-sparse solution for a each k; they may generate more than one solution, or no solution at all for a given k. For example, NNOMP selects columns of A sequentially, but some of them may not have a positive weight when solving the corresponding NNLS. The loop on line 8 ensures that, if there exists some k for which no k-sparse solution is generated, then the $(k-1)$-sparse solution is used instead (or the $(k-2)$-sparse one if no $(k-1)$-sparse solution is generated, and so on). The condition on line 9 ensures that, if there are several k-sparse solutions for a given k, then only the best one is kept.*

Once C is computed, step 2 consists in selecting one solution per column to build the solution matrix X, that is, it consists in choosing the sparsity level for each column of X. This selection step is a combinatorial problem, similar to an assignment problem. Let us define the binary variables $z_{i,j} \in \{0,1\}$ for $i \in \{0,1,\dots r\}$ and $j \in \{1,2,\dots n\}$ such that $z_{i,j} = 1$ if and only if the jth column of X is i-sparse. The variable z encodes which sparsity level is selected for each column of X. Given the cost matrix C computed in step 1, step 2 requires to solve the following integer program

$$\begin{aligned} &\min_{z \in \{0,1\}^{r \times n}} \sum_{i,j} z_{i,j} C(i,j) \\ &\text{such that } \sum_i z_{i,j} = 1 \text{ for all } j, \quad \text{and} \quad \sum_{i,j} i\, z_{i,j} \leq q. \end{aligned} \tag{4.7}$$

The objective is to minimize the reconstruction error, while the first constraints impose that each column of X has a single sparsity level, and the second that the total number of non-zero entries of X does not exceed q.

We propose a greedy selection algorithm to solve (4.7), see lines 12 to 28 of Algorithm 4.2. As we will prove in Theorem 4.1, this greedy strategy is nearly optimal. It works as follows. For each column j, the scalar k_j indicates its sparsity level at the current iteration, that is, at every iteration, the jth column of X is k_j-sparse. We initialize the algorithm with the 0-sparse solution (the vector of all zeros) for each column (line 12), that is, $k_j = 0$ for all j. Note that $\sum_j k_j$ corresponds to the current sparsity level of the solution.

At each iteration, we will decrease the sparsity of a single column of X. To pick that column in an optimal way, let us define the matrix $\mathcal{G}$, with the same dimensions as C, as follows: at any iteration, the entry $\mathcal{G}(i,j)$ is equal to the potential gain in reconstruction error if the jth column of X goes from k_j-sparse to i-sparse divided

by the sparsity difference between these two solutions (that is, $i - k_j$). In particular, at the first iteration (line 15), when $X = 0$, we have

$$\mathcal{G}(i,j) = \frac{C(0,j) - C(i,j)}{i} \quad \text{for all } i,j.$$

Let us denote by (i^*, j^*) the position of the largest entry of $\mathcal{G}$. Given a current solution, the column that will decrease the error $\|B - AX\|_F^2$ the most by decreasing its sparsity is the one corresponding to the column of $\mathcal{G}$ with the largest entry, that is, the j^*th column. Finding this entry is cheap in practice, as we update a sorted list of the maximum entry of each column of $\mathcal{G}$. We denote the quantity δ as the difference in sparsity of the selected column before and after it has been updated, that is, $\delta = i^* - k_{j^*}$. After the value of k_{j^*} has been updated to i^*, we update the entries of the j^*th column of $\mathcal{G}$ accordingly (line 23). Note that $\mathcal{G}(i, j^*) = 0$ for all $i \leq i^*$.

To avoid generating a too dense solution (recall $\sum_j k_j$ must be smaller than q), the procedure on lines 24 to 26 sets to zero the entries of $\mathcal{G}$ whose selection would lead to a total sparsity $\sum_j k_j$ larger than q. Note that thanks to the condition on line 24, this procedure is executed only for the last few iterations of Salmon. Indeed, the maximum increase δ is r, so this procedure would not be useful as long as $\sum_j k_j \leq q - r$.

When the sum of the sparsity levels equals q, or when all entries of $\mathcal{G}$ are zero, we stop the procedure and we build the final q-sparse solution X by selecting for each column the solution corresponding to its final sparsity level (line 27). As a side note, only by changing the stopping condition from a sparsity threshold to an error threshold, we can adapt this algorithm to tackle ℓ_0-minimization under a constraint on the reconstruction error.

Although the selection algorithm of step 2 is greedy, since we perform an optimal selection at each iteration, it is able to generate a near-optimal solution to Problem (4.7), as shown below.

Theorem 4.1 (Near-optimality of the selection step) *Given that C is non-increasing by columns, that is, $C(i,j) \leq C(i',j)$ for all $i' \leq i$ and all j, the proposed selection step of Salmon (lines 12 to 28 of Algorithm 4.2) computes a near-optimal solution of* (4.7) *in the following sense. Denoting*

- $f(z) = \sum_{i,j} z_{i,j} C(i,j)$ *the value of the objective function of* (4.7) *for a solution z,*
- z^* *an optimal solution to Problem* (4.7)*, and*
- z_{Sal} *the solution computed by the selection step of Salmon,*

we have

$$f(z^*) \leq f(z_{Sal}) \leq f(z^*) + \max_j \|C(:,j)\|_\infty.$$

Proof 4.1 *First, note that our proposed greedy algorithm generates a feasible solution of* (4.7) *since we make sure that $\sum_j k_j$ remains smaller than q, and hence, by optimality of z^*, $f(z^*) \leq f(z_{Sal})$.*

Let us now show that $f(z_{Sal}) \leq f(z^*) + \max_j \|C(:,j)\|_\infty$. *Our greedy procedure is similar to a dynamic programming approach. In fact, let us denote the optimal objective function value of* (4.7) *as* $f^*(q)$ *that depends on the parameter* q*, the global sparsity level allowed; note that* $f(z^*) = f^*(q)$. *The greedy algorithm is initialized with* $z_{0,j} = 1$ *for all* j*, that is,* $X = 0$*, which is optimal for* $q = 0$ *(it is the only feasible solution), and hence gives* $f^*(0) = \sum_j C(0,j)$. *It then progressively decreases the sparsity to reduce the objective the most at each iteration. At each iteration of the greedy algorithm, the support of a single column of* X *is increased in order to maximize the ratio between the decrease in objective function value and the decrease of sparsity. Since the columns of* X *do not interact with each other in the objective function and since* C *is non-increasing in each column, the greedy solution cannot possibly be improved as long as* $\delta \leq q - \sum_j k_j$*, that is, as long as this optimal way of picking a column is allowed by the global sparsity level. In summary, our greedy algorithm produces intermediate optimal solutions of* (4.7) *with objective* $f^*(\sum_j k_j + \delta)$ *as long as* $\sum_j k_j + \delta \leq q$.

The only moment when the greedy algorithm might fall short of global optimality is during the last iterations: if at some point the optimal move is to increase the support of a column in such a way that the total sparsity would exceed q *(that is,* $\sum_j k_j + \delta > q$*), then our greedy algorithm may not be optimal because, to allow that move, we might need to reduce the support of another column, which the greedy approach does not allow. Making that move anyway would generate an optimal solution with global sparsity* $q' = \sum_j k_j + \delta > q$ *which would not be feasible for* (4.7). *Observe that*

- $f(z_{Sal}) \leq f^*(q' - \delta)$ *since the greedy algorithm keeps improving the* $(q' - \delta)$*-sparse solution in its next iterations (although possibly not optimally).*
- $f^*(q') \geq f^*(q' - \delta) - \max_j \|C(:,j)\|_\infty$ *since the move from* $q' - \delta$ *to* q' *using the greedy strategy is optimal and, in the worst case, will decrease the sparsity level of a column from* r *to* 0 *reducing the error by at most* $\max_j \|C(:,j)\|_\infty$.
- $f^*(q') \leq f^*(q) \leq f^*(q' - \delta)$ *since* $q' - \delta \leq q \leq q'$.

Combining these observations, we obtain

$$f^*(q) = f(z^*) \geq f^*(q') \geq f^*(q' - \delta) - \max_j \|C(:,j)\|_\infty \geq f(z_{Sal}) - \max_j \|C(:,j)\|_\infty,$$

which gives the result. □

Note that, in most practical cases, such as hyperspectral unmixing, we have $n \gg r$ and hence $\max_j \|C(:,j)\|_\infty$ is negligible compared to $f(z^*)$. In addition, it is rather unlikely to reach this worst case, as it implies that the greedy algorithm needs to increase the sparsity level of one column from 0 to r at the last step. In practice, when $\delta > 1$, it is usually small (equal to 2 or 3). In fact, in our experiments, we observed that δ is in most cases equal to 1. When δ is equal to 1 in the last r steps, the greedy algorithm is globally optimal (or, more generally, when $\delta = q - \sum_k k_j$ in the last step). For example, for the datasets used in the numerical experiments (Section 4.4) and with the three sparse NNLS algorithms to generate the Pareto fronts

(that is, the matrix C), the greedy selection algorithm generated a guaranteed globally optimal solution (that is, $\delta = q - \sum_k k_j$ in the last step) in 19 out of 22 cases.

In practice, the global optimality of Salmon to solve the sparse MNNLS problem (4.1) therefore heavily relies on step 1. If step 1 is done with Arborescent, then the Pareto fronts are computed optimally and Salmon computes a near-optimal solution to Problem (4.1). Otherwise, it only computes an approximate solution, although there exist some conditions under which NNOMP or the homotopy algorithm do recover the true Pareto fronts, see Sections 4.2.1 and 4.2.2.

Computational cost of Salmon The cost of step 1 depends on the algorithm used to generate the Pareto fronts. In all cases, the n biobjective subproblems are solved independently, so the cost of step 1 grows linearly with n. For one subproblem, that is, to generate one Pareto front, we have that

- The cost of Arborescent depends on the number of nodes explored in the branch and bound. In the worst case, it is of the same order as the bruteforce algorithm, and requires to solve $\binom{r}{k}$ NNLS subproblems, while, in the best case, it is of the order of r [61]. Empirically, the cost is far from the worst case but grows faster than linear with r.
- The cost of NNOMP is in $\mathcal{O}(mr^4)$ operations [83].
- The cost of the homotopy algorithm is of the same order as an active-set algorithm and requires at least $\mathcal{O}(r^4)$ operations, see Section 4.2.2.

Given C, the selection step consists in building $\mathcal{G}$ in $\mathcal{O}(rn)$ operations, then iterating q times (in the worst case) to select a solution. As we maintain updated a sorted list to avoid recomputing the maximum at each iteration, this is done in $\mathcal{O}(q \log(n))$ operations. Since q is in the order of rn in the worst case, the cost of the selection step is dominated by the cost of the Pareto-front-generating step.

4.3.2 Adapting NNOMP for matrix-wise q-sparse MNNLS

Let us now adapt NNOMP for the matrix-wise q-sparse MNNLS problem. To avoid the overcost of solving the vectorized problem (4.2) directly with NNOMP, we can adapt NNOMP to handle the matrix form of Problem (4.1). We detail the adaptation in Algorithm 4.3; note that it is not an original contribution and seems to be common knowledge in the sparse approximation community. The solution produced by this algorithm is strictly equivalent to the one produced by direct solving of (4.2). Our goal in introducing it is to show how using NNOMP within the two-step algorithm Salmon is advantageous compared to using NNOMP directly on Problem (4.1).

On line 1, we initialize X as a zero matrix, S as an empty support, and the residual R_S as equal to B. Then, we select greedily entries to add to the support, while the cardinality of the support is lower than q and the residual greater than zero (line 2). The greedy selection on line 3 consists in choosing the entry that maximizes the decrease of the residual error; this entry is added to the support on line 4. Then, we update X on line 5 (this is an NNLS problem, that we solve with an active-set

Algorithm 4.3: The greedy algorithm NNOMP adapted for matrix-wise q-sparse MNNLS.

Input: $B \in \mathbb{R}^{m \times n}$, $A \in \mathbb{R}^{m \times r}$, $q \in \mathbb{N}$
Output: X, approximate solution to Problem (4.1)

1 $X \leftarrow 0^{r \times n}$; $S \leftarrow \emptyset$; $R_S \leftarrow B$
2 **while** $|S| < q$ *and* $\max_{(i,j)}(A^\top R_S)_{i,j} > 0$ **do**
3 $\quad (i^*, j^*) \leftarrow \operatorname{argmax}_{(i,j) \notin S} A^\top R_S$
4 $\quad S \leftarrow S \cup (i^*, j^*)$
5 $\quad X \leftarrow \min_X \|A_S - B_S X_S\|_F^2$
6 $\quad R_S \leftarrow B - AX$
7 $\quad S \leftarrow \{(i,j) : X(i,j) > 0\}$

algorithm), and the residual R_S on line 6; note that in practice we only need to update the j^*-th column of X and R_S. On line 7 we perform a support compression, that is, we restrain the support to the non-zero entries of X. This last step is necessary because of the nonnegativity constraint, that may put to zero an entry that was selected at a previous iteration.

Other greedy algorithms could be adapted similarly, but here we focus only on NNOMP for conciseness. Also, our goal is to study how the original NNOMP compares to NNOMP used within the two-step approach Salmon, rather than comparing different greedy algorithms with each other. The homotopy algorithm may also be similarly adaptable, but this is not trivial and out of the scope of this work. Adapting Arborescent in such a way does not seem possible as it would result in a combinatorial explosion.

4.4 Experiments

In this section, we study the performance of the proposed algorithm on the unmixing of 7 datasets: 3 faces datasets and 4 hyperspectral images.

4.4.1 Data

In the faces datasets, each column of B corresponds to a pixel and each row to an image (that is, $B(i,j)$ is the intensity of pixel j in image i). It is well-known that NMF will extract facial features as the rows of matrix X [47]. As no groundtruth is available, we first compute A with SNPA [32], an algorithm for separable NMF, setting the factorization rank r as in the literature. We then compute X with our sparsity-enhancing method. Imposing sparsity on X means that we require that only a few pixels are contained in each facial feature [38]. We consider the 3 widely used face datasets CBCL[3], Frey[4], and Kuls[5].

[3]Downloaded from `http://poggio-lab.mit.edu/codedatasets`
[4]Downloaded from `https://cs.nyu.edu/~roweis/data.html`
[5]Downloaded from `http://www.robots.ox.ac.uk/`

Similarly a hyperspectral image is an image-by-pixel matrix where each image corresponds to a different wavelength. The columns of A represent the spectral signature of the pure materials (also called endmembers) present in the image [13], and we use the ground truth A from [85]. We compute X, whose columns represent the abundance of materials in each pixel. It makes sense to impose X to be sparse as most pixels contain only a few endmembers [54]. We consider the 4 widely used datasets[6] Jasper, Samson, Urban, and Cuprite. The characteristics of these datasets are summarized in Table 4.1.

Table 4.1: Summary of the datasets, for which $B \in \mathbb{R}^{m \times n}$ and $A \in \mathbb{R}^{m \times r}$.

Dataset	Type	m	n	r
CBCL	Faces	2429	$19 \times 19 = 361$	49
Frey	Faces	1965	$20 \times 28 = 560$	36
Kuls	Faces	20	$64 \times 64 = 4096$	5
Jasper	Hyperspectral	198	$100 \times 100 = 1000$	4
Samson	Hyperspectral	156	$95 \times 95 = 9025$	3
Urban	Hyperspectral	162	$307 \times 307 = 94249$	6
Cuprite	Hyperspectral	188	$250 \times 191 = 47750$	12

4.4.2 Methods

All methods have been implemented in Julia and run on a computer with a processor Intel Core i5-2520M @2.50GHz. The code and experiment scripts are provided in an online repository[7].

We compare the following methods:

- AS denotes an active-set algorithm that solves exactly the NNLS problem without sparsity constraint. It serves as a baseline to compare with the sparsity-constrained methods.
- ℓ_1-CD denotes a coordinate descent algorithm with an ℓ_1 penalty. The penalty parameter λ is fixed and the same for the whole matrix, as in [43]. It has been tuned manually to reach the target sparsity. We compute the unbiased solution by running an active-set NNLS algorithm restrained to the non-zero elements of the ℓ_1-penalized solution.
- Hcw denotes the homotopy algorithm described in Section 4.2.2, that solves the column-wise k-sparse problem, as defined in (1.8) For each column, we generate the regularization path, take the best k-sparse solution, and unbias it as described above.
- H+S corresponds to the two-step algorithm Salmon using the homotopy algorithm in step 1 (solutions are unbiased as above).

[6] Downloaded from `http://lesun.weebly.com/hyperspectral-data-set.html`
[7] `https://gitlab.com/nnadisic/giant.jl`

- OGcw stands for orthogonal greedy and denotes the NNOMP algorithm described in Section 4.2.1, that solves the column-wise k-sparse problem.
- OGg denotes the matrix-wise variant of NNOMP described in Section 4.3.2.
- OG+S corresponds to the two-step algorithm Salmon using NNOMP in step 1. We generate the whole Pareto fronts, hence the column-wise sparsity target for NNOMP is $k = r$.
- ARBOcw denotes the branch-and-bound algorithm Arborescent described in Chapter 3, that solves the column-wise k-sparse problem.
- ARBO+S corresponds to the two-step algorithm Salmon using Arborescent in step 1. We generate the whole Pareto fronts, hence the column-wise sparsity target for Arborescent is $k = 1$.

We choose the parameter k by trial-and-error. Unless stated otherwise, we define the sparsity parameter of matrix-wise methods as $q = k \times n$, which is equivalent to an average column-wise k-sparsity constraint.

For every dataset, we run the nine methods, and measure the average column-sparsity of the given solutions (defined as the number of non-zero entries divided by the number of columns), the relative reconstruction error $\frac{\|B-AX\|_F}{\|B\|_F}$, and the computing time, for which we measure the median over 10 runs. We set a timeout of 6000 seconds. Note that, for a given dataset and with the same parameters, a given algorithm always gives the same output. For the 3 methods based on NNOMP, we normalize the columns of the matrix A before the computation.

4.4.3 Results and interpretation

The results of the experiments are summarized in Table 4.2. We first note the natural sparsity of the data: without sparsity constraint, AS already produces very sparse solutions, and column-wise methods produce solutions with an average sparsity below the sparsity target k, meaning that some columns are naturally sparser than k. We observe that the column-wise methods give relatively bad results in terms of reconstruction error, while the variants of Salmon are able to enforce sparsity while limiting the loss in error. H+S is only slightly slower than Hcw, meaning that the selection (step 2) takes less time than the homotopy (step 1). On the other hand, OG+S and ARBO+S are slower than their column-wise counterparts because they need to be run with a different sparsity target, respectively $k = r$ and $k = 1$, to compute the whole Pareto front. The computing times of H+S and OG+S seem proportional, and differ by a factor between 2 and 3, which is consistent with our discussion about computational cost in Section 4.3.1. ℓ_1-CD is very fast, and produces good solutions with some datasets (Jasper, Samson), but it is outperformed by Salmon in all cases and sometimes produces solutions with high error (CBCL, Cuprite).

Comparing OGg and OG+S, we observe that OGg is faster for tall matrices, while OG+S is faster for short and fat matrices. Also, OG+S is always better in terms of reconstruction error, meaning that performing the heuristic column-wise

Table 4.2: Results of the experiments, for the unmixing of facial and hyperspectral datasets. Time is in seconds, relative error in percent, and sparsity is the average number of non-zero entries per column. Numbers in bold represent, for a given setting, the error of ARBO+S and the best error among the other sparse methods. For the variants of Salmon, a star * indicates that the greedy selection (step 2 of Salmon) is optimal (which can be checked easily: it requires $\delta = q - \sum_k k_j$ at the last iteration). Jasper is processed once with all algorithms for $k = q/n = 2$, and once with matrix-wise algorithms for $q/n = 1.8$ (which is not possible with column-wise algorithms).

		AS	ℓ_1-CD	Hcw	**H+S**	OGcw	OGg	**OG+S**	ARBOcw	**ARBO+S**
CBCL	Time	0.2	0.1	0.71	0.81	0.08	0.31	3.7	timeout	timeout
$r = 49$	Error	12.04	17.37	16.19	13.22*	13.12	12.35	**12.3***	-	-
$k = 3$	Sparsity	6.64	3	2.69	3	2.37	3	3	-	-
Frey	Time	0.22	0.08	1.12	1.27	0.18	0.61	3.97	timeout	timeout
$r = 36$	Error	19.35	21.76	23.22	20.75*	21.35	19.86	**19.8**	-	-
$k = 6$	Sparsity	12.29	6	5.52	6	4.64	6	6	-	-
Kuls	Time	0.17	0.12	0.18	0.17	0.28	1.82	0.5	0.67	1.41
$r = 5$	Error	19.05	19.61	20.13	19.12*	19.46	**19.06**	**19.06***	19.42	**19.06***
$k = 3$	Sparsity	3.45	3	2.86	2.99	2.7	3	3	2.76	3
Jasper	Time	0.34	0.22	0.38	0.48	0.39	6.08	1.12	1.21	1.93
$r = 4$	Error	5.71	**5.72**	6.99	**5.72***	7.49	5.76	5.73	6.18	**5.71***
$k = 2$	Sparsity	2.27	2	1.78	1.99	1.72	2	2	1.78	2
Jasper	Time	-	0.18	-	0.44	-	5.26	1.15	-	1.7
$r = 4$	Error	-	7.87	-	5.95*	-	6.06	**5.77***	-	**5.74***
$q/n = 1.8$	Sparsity	-	1.8	-	1.79	-	1.8	1.8	-	1.8
Samson	Time	0.22	0.24	0.2	0.26	0.31	3.67	0.57	0.52	0.8
$r = 3$	Error	3.3	**3.3**	3.34	**3.3***	6.76	3.32	**3.3***	3.4	**3.3***
$k = 2$	Sparsity	2.2	2	1.85	2	1.6	1.99	1.99	1.83	2
Urban	Time	5.08	4.31	4.86	7.79	3.38	958	16.4	33.5	73.1
$r = 6$	Error	7.67	8.13	8.62	7.83*	8.97	8.07	**7.76***	8.27	**7.71***
$k = 2$	Sparsity	2.63	2	1.9	2	1.7	2	2	1.83	2
Cuprite	Time	5.19	3.32	7.86	10.1	5.06	620	31.5	784	4829
$r = 12$	Error	1.74	3.17	2.37	2.01	2.32	1.97	**1.89***	1.93	**1.83***
$k = 4$	Sparsity	6.61	4	3.92	4	3.53	4	4	3.81	4

and then recombining solutions is more efficient than applying the same heuristic matrix-wise. As regards Arborescent, we see a clear improvement of reconstruction error with Salmon, at the cost of a larger computation time. The two-step approach of Salmon allows Arborescent to be applied for matrix-wise q-sparse MNNLS, which would be impossible otherwise. Another advantage of the matrix-wise formulation is the capacity to tune the sparsity parameter more finely; this is illustrated on Jasper with $q/n = 1.8$, for which we reach an average sparsity similar to that of column-wise metods and still get lower errors. For the three variants of Salmon, the stars indicates that the selection (step 2) is done optimally as discussed in Theorem 4.1; here it is the case for 19 out of 22 settings.

The abundance maps corresponding to the second material extracted in the hyperspectral image Jasper are shown in Fig. 4.4. The abundance maps of the features extracted in the faces dataset Kuls are shown in Fig. 4.3. The other abundance maps from our experiments are available in [57][8].

[8] `https://arxiv.org/abs/2011.11066`

The features extracted in Fig. 4.3 correspond to different directions of light. Without sparsity constraint, the quality of the images is good, but the features are not well separated. Column-wise methods and ℓ_1-CD fail to retrieve the features and produce noisier images with pixelated regions. Salmon, using any of the 3 possible methods for step 1, produces better-separated features, with a better spatial coherence.

In Fig. 4.4, we consider only the second material extracted. It corresponds mainly to water. However, with most methods, it is mixed with pixels from the road on the right. Salmon variants separate it better while maintaining the spatial coherence. Salmon using the homotopy in step 1 produces an almost perfect separation, and eliminates the non-water pixels while sharpening the edge between water and non-water pixels. In these images, it is clear that some pixels are composed of only one endmember (for instance, water) while others are mixed (trees and grass mixed). This is the kind of setting where Salmon can bring significant improvement.

What algorithm should one use for step 1 of Salmon? When the dimension r is small or when the computing time is not critical, the best option is to use Arborescent in step 1 as it is the only algorithm to compute the Pareto fronts exactly. In other cases, using NNOMP in step 1 generally produces better solutions than using the homotopy algorithm. The homotopy algorithm is the fastest so it is appropriate when processing very large datasets with a limited time.

4.5 Conclusion

In this chapter, we focused on the multiple nonnegative least squares problem with a matrix-wise ℓ_0 constraint (4.1). We introduced Salmon (Algorithm 4.2), that first computes for each column a Pareto front (step 1) and then applies a provably near-optimal selection strategy to build a solution matrix X (step 2). We computed the Pareto fronts with three existing algorithms: one exact but slow branch-and-bound algorithm (Arborescent from Chapter 3), and two fast heuristics. We illustrated the advantages of Salmon for the unmixing of real-world facial and hyperspectral images, for which it outperformed state-of-the-art methods.

(a) AS (no sparsity constraint)

(b) ℓ_1-CD

(c) Hcw

(d) H+S

Figure 4.3: (1/2) Abundance maps (that is, reshaped rows of X) from the unmixing of the faces dataset Kuls by different algorithms.

(e) OGcw

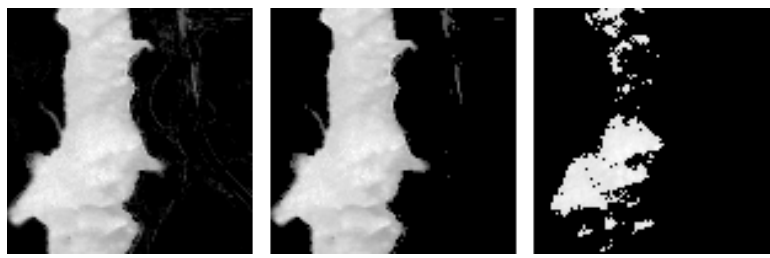
(f) OGg

(g) OG+S

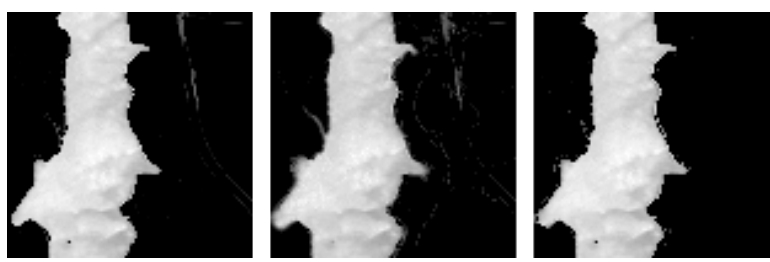
(h) ARBOcw

(i) ARBO+S

Figure 4.3: (2/2) Abundance maps (that is, reshaped rows of X) from the unmixing of the faces dataset Kuls by different algorithms.

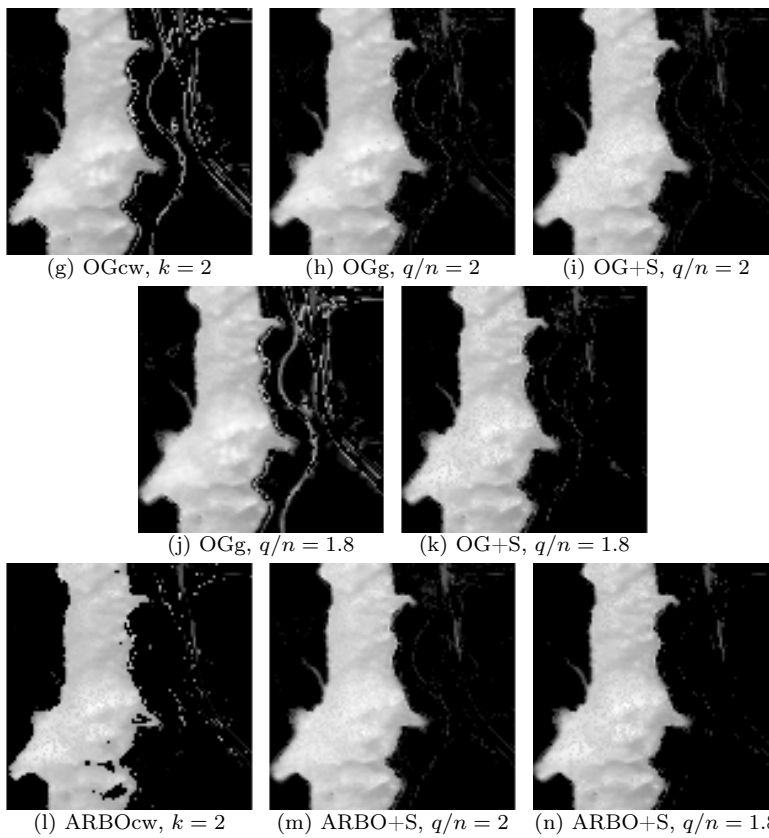


(a) AS (b) ℓ_1-CD, $q/n = 2$ (c) ℓ_1-CD, $q/n = 1.8$



(d) Hcw, $k = 2$ (e) H+S, $q/n = 2$ (f) H+S, $q/n = 1.8$

Figure 4.4: (1/2) Abundance map of the second material (that is, the second row of X reshaped) from the unmixing of the hyperspectral image Jasper by different algorithms. This material corresponds to water.



(g) OGcw, $k = 2$ (h) OGg, $q/n = 2$ (i) OG+S, $q/n = 2$

(j) OGg, $q/n = 1.8$ (k) OG+S, $q/n = 1.8$

(l) ARBOcw, $k = 2$ (m) ARBO+S, $q/n = 2$ (n) ARBO+S, $q/n = 1.8$

Figure 4.4: (2/2) Abundance map of the second material (that is, the second row of X reshaped) from the unmixing of the hyperspectral image Jasper by different algorithms. This material corresponds to water.

Chapter 5

Sparse Separable NMF

"Le véritable anarchiste marche toujours entre les clous, parce qu'il a horreur de discuter avec les agents."

– Georges Brassens

In this chapter, we propose a new variant of nonnegative matrix factorization (NMF) that combines the assumptions of separability and sparsity. We call this variant sparse separable NMF (SSNMF), which we prove to be NP-complete, as opposed to separable NMF which can be solved in polynomial time (see Chapter 2). The main motivation to consider this new model is to handle underdetermined blind source separation problems, such as multispectral image unmixing. We introduce an algorithm to solve SSNMF, based on the successive nonnegative projection algorithm (SNPA, an effective algorithm for separable NMF) and the algorithm Arborescent presented in Chapter 3. We prove that, in noiseless settings and under mild assumptions, our algorithm recovers the true underlying sources. This is illustrated by experiments on synthetic data sets and the unmixing of a multispectral image.

The content of this chapter is mainly extracted from [59]:

59. Nadisic, N., Vandaele, A., Cohen, J. E. & Gillis, N. *Sparse separable nonnegative matrix factorization* in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECMLPKDD)* (2020), 335–350.

5.1 Introduction

In this chapter we consider the separable NMF problem as discussed in Section 1.8. As a reminder, separable NMF consists in the following: given a r-separable matrix B, find $A = B(:, \mathcal{J})$ with $|\mathcal{J}| = r$ and $X \geq 0$ such that $B = AX$.

The central contribution of this chapter is a novel variant of this model with an additional constraint of k-sparsity on the columns of X. We study the theoretical properties of this new model, and we propose a guaranteed algorithm to tackle it. First, let us describe an existing algorithm for separable NMF, namely the successive

nonnegative projection algorithm (SNPA). This algorithm is the foundation for our proposed method.

5.1.1 Successive Nonnegative Projection Algorithm

Various algorithms have been developed to tackle the (near-)separable NMF problem. In Chapter 2, we studied vertex component analysis (VCA) and the successive projections algorithm (SPA). Other examples are the fast canonical hull algorithm [45], or the successive nonnegative projections algorithm (SNPA) [32]. Such algorithms start with an empty matrix A and a residual matrix $R = B$, and then alternate between two steps: a greedy selection of one column of R to be added to A, and an update of R using B and the columns extracted so far. As SNPA was shown, both theoretically and empirically, to perform better and to be more robust than its competitors [32], it is the one we study here in detail. The main difference of SNPA over VCA and SPA is that, instead of orthogonal projections, it performs at each iteration a projection on the convex hull of the origin and the column of B extracted so far. Moreover, SNPA is able to handle the underdetermined case when $\text{rank}(A) < m$ which will be key for our problem setting (see below for more details).

SNPA is presented in Algorithm 5.1. SNPA selects, at each step, the column of

Algorithm 5.1: SNPA [32]

Input: A near-separable matrix $B \in \mathbb{R}^{m \times n}$, the number r of columns to be extracted, and a strongly convex function f with $f(0) = 0$ (by default, $f(x) = \|x\|_2^2$).

Output: A set of r indices $\mathcal{J}$, and a matrix $X \in \mathbb{R}_+^{r \times n}$ such that $B \approx B(:, \mathcal{J})X$.

1 Init $R \leftarrow B$
2 Init $\mathcal{J} = \{\}$
3 Init $t = 1$
4 **while** $R \neq 0$ & $t \leq r$ **do**
5 $p = \text{argmax}_j \, f(R(:, j))$
6 $\mathcal{J} = \mathcal{J} \cup \{p\}$
7 **foreach** j **do**
8 $X^*(:, j) = \text{argmin}_{h \in \Delta} \, f(B(:, j) - B(:, \mathcal{J})h)$
9 $R(:, j) = B(:, j) - B(:, \mathcal{J})X^*(:, j)$
10 t = t + 1

B maximizing a function f (which can be any strongly convex function such that $f(0) = 0$, and $f = ||.||_2^2$ is the most common choice). Then, the columns of B are projected onto the convex hull of the origin and the columns extracted so far, see step 8 where we use the notation

$$\Delta = \Big\{ x \mid x \geq 0, \sum_i x_i \leq 1 \Big\},$$

whose dimension is clear from the context. After r steps, given that the noise is sufficiently small and that the columns of A are vertices of $\text{conv}(A)$, SNPA is guaranteed to identify A. An important point is that SNPA requires the columns of X to satisfy $\|X(:,j)\|_1 \leq 1$ for all j. This assumption can be made without loss of generality by properly scaling the columns of the input matrix to have unit ℓ_1 norm; see the discussion in [32].

5.1.2 Model limitations

Unfortunately, some data sets cannot be handled successfully by separable NMF, even when all data points are linear combinations of a subset of the input matrix. In fact, in some applications, some columns of the basis matrix A, that is, the atoms, might not be vertices of $\text{conv}(A)$. This may happen when one seeks a matrix A which is not full column rank. For example, in multispectral unmixing, m is the number of spectral bands which can be smaller than r, which is the number of materials present in the image; see Section 5.6.2 for more details. Therefore, it is possible for some columns of A to be contained in the convex hull of the other columns, that is, to be additive linear combinations of others columns of A; see Fig. 5.1 for illustrations in three dimensions (that is, $m = 3$).

These difficult cases cannot be handled with separable NMF, because it assumes the data points to be linear combinations of vertices, so an "interior vertex" cannot be distinguished from another data point. However, if we assume the *sparsity* of the mixture matrix X, we may be able to identify these interior vertices. To do so, we introduce a new model, extending the approach of SNPA using additional sparsity constraints. More precisely, we consider a constraint of column-wise k-sparsity on the factor X, meaning that a data point is expressed as the combination of at most k atoms. For example, in multispectral unmixing, this constraint means that a pixel is composed of at most k materials. We introduce in Section 5.2 a proper definition of this new problem, which we coin as sparse separable NMF (SSNMF). As a reminder, a brief review of sparse NMF is presented in Section 1.7.

5.1.3 Contributions and outline

In this chapter, we study the SSNMF model from a theoretical and a practical point of view. Our contributions can be summarized as follows:

- In Section 5.2, we introduce the SSNMF model.
- In Section 5.3, we prove that, unlike separable NMF, SSNMF is NP-complete.
- In Section 5.4, we propose an algorithm to tackle SSNMF, based on SNPA and the exact k-sparse NNLS solver Arborescent from Chapter 3.
- In Section 5.5, we prove that our algorithm is correct under reasonable assumptions, in the noiseless case.
- In Section 5.6, experiments on both synthetic and real-world data sets illustrate the relevance and efficiency of our algorithm.

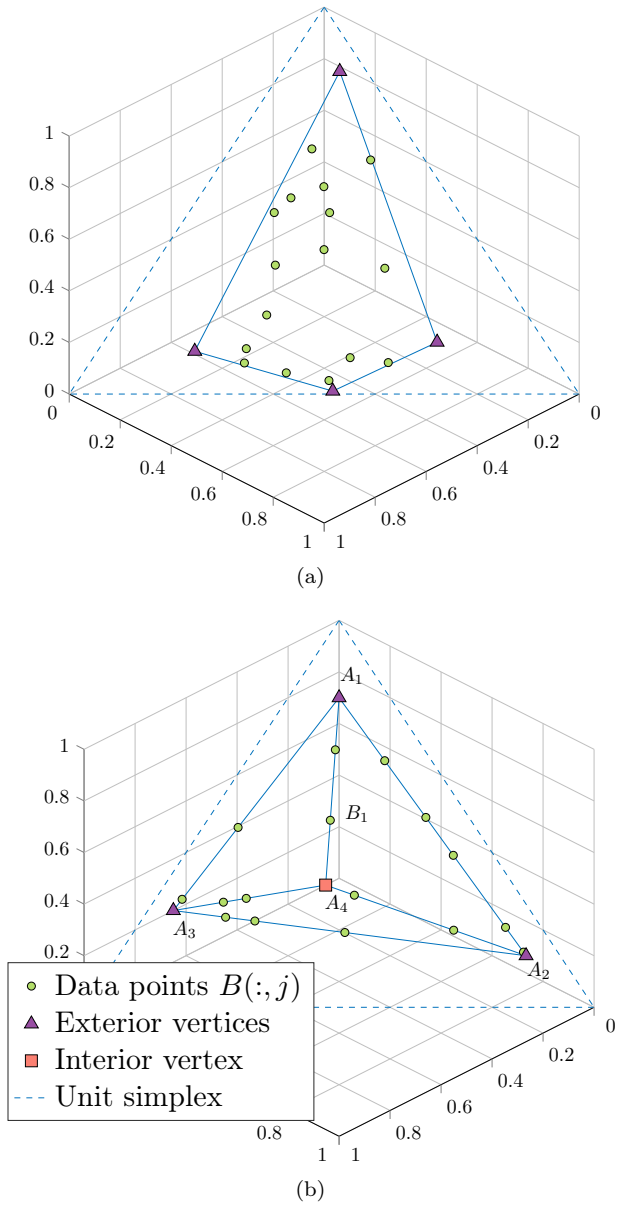


Figure 5.1: On the top (a), all vertices are exterior, and SNPA is assured to identify them all. On the bottom (b), the data points are 2-sparse combinations of 4 points, one of which (vertex 4) is "interior" hence it cannot be identified with separable NMF.

5.2 Sparse separable NMF

We explained in the previous section why separable NMF does not allow for the identification of "interior vertices", as they are nonnegative linear combinations of other vertices. However, if we assume a certain column-wise sparsity on the coefficient matrix X, they may become identifiable. For instance, the vertex A_4 of Fig. 5.1b can be expressed as a combination of the three exterior vertices (A_1, A_2, and A_3), but not as a combination of any two of these vertices. Moreover, some data points cannot be explained using only pairs of exterior vertices, while they can be if we also select the interior vertex A_4.

5.2.1 Problem statement and complexity

Definition 5.1 *A matrix B is k-sparse r-separable if there exists a subset of r columns of B, indexed by $\mathcal{J}$, and a nonnegative matrix $X \geq 0$ with $\|X(:,j)\|_0 \leq k$ for all j such that $B = B(:,\mathcal{J})X$.*

Definition 5.1 corresponds to Definition 1.1 with the additional constraint that X has k-sparse columns, that is, columns with at most k non-zero entries. A natural assumption to ensure that we can identify A (that is, find the set $\mathcal{J}$), is that the columns of A are not k-sparse combinations of any other columns of A; see Section 5.5 for the details.

Problem 5.1 (SSNMF) *Given a k-sparse r-separable matrix B, find $A = B(:,\mathcal{J})$ with $|\mathcal{J}| = r$ and a column-wise k-sparse matrix $X \geq 0$ such that $B = AX$.*

As opposed to separable NMF, given $\mathcal{J}$, computing X is not straightforward. It requires to solve the following ℓ_0-constrained optimization problem

$$X^* = \underset{X \geq 0}{\operatorname{argmin}} f(B - B(:,\mathcal{J})X) \text{ such that } \|X(:,j)\|_0 \leq k \text{ for all } j. \tag{5.1}$$

Because of the combinatorial nature of the ℓ_0-"norm", this k-sparse projection is a difficult subproblem with $\binom{r}{k}$ possible solutions. It is known to be NP-hard, see Chapter 3. In particular, a brute-force approach could tackle this problem by solving $\mathcal{O}(r^k)$ NNLS problems. However, this combinatorial subproblem can be solved exactly and at a reasonable cost by the branch-and-bound algorithm Arborescent (see Chapter 3), given that r is sufficiently small, which is typically the case in practice. Even when k is fixed, the following result shows that no provably correct algorithm exists for solving SSNMF in polynomial time (unless P=NP):

Theorem 5.1 *SSNMF is NP-complete for any fixed $k \geq 2$.*

Proof 5.1 *The proof is given in Section 5.3. Note that the case $k = 1$ is trivial since each data point is a multiple of a column of A.*

However, in Section 5.5, we show that under a reasonable assumption, SSNMF can be solved in polynomial time when k is fixed.

5.2.2 Related work

To the best of our knowledge, the only work presenting an approach to tackle SSNMF is the one by Sun and Xin (2011) [75] — and it does so only partially. It studies the blind source separation of nonnegative data in the underdetermined case. The problem tackled is equivalent to NMF in the case $m < r$. The assumptions used in this work are similar to ours, that is, separability and sparsity. However, the setup considered is less general than SSNMF because the sparsity assumption (on each column of X) is limited to $k = m - 1$, while the only case considered theoretically is the case $r = m + 1$ with only one interior vertex.

The proposed algorithm first extracts the exterior vertices using the method LP-BSS from [56], and then identifies the interior vertex using a brute-force geometric method. More precisely, they select an interior point, and check whether at least two of the $m-1$ hyperplanes generated by this vertex with $m-2$ of the extracted exterior vertices contain other data points. If it is the case, then they conclude that the selected point is an interior vertex, otherwise they select another interior point. For example, when $m = 3$, this method consists in constructing the segments between the selected interior point and all the exterior vertices. If two of these segments contain at least one data point, then the method stops and the selected interior point is chosen as the interior vertex. Looking at Figure 5.1b, the only interior point for which two segments joining this point and an exterior vertex contain data points is A_4. Note that, to be guaranteed to work, this method requires at least two hyperplanes containing the interior vertex and $m - 2$ exterior vertices to contain data points. This will not be a requirement in our method.

5.3 Proof of NP-completeness of SSNMF

The main purpose of this section is to provide the proof of Theorem 5.1. More precisely, we prove the NP-completeness of SSNMF with $k = 2$, which we denote 2-SSNMF. The decision version of this problem is formally defined as follows.

Problem 5.2 *2-SSNMF*
Given: a natural number $r > 0$ and a 2-sparse r-separable matrix B
Question: does there exist a dictionary matrix $A = B(:, \mathcal{J})$ with $|\mathcal{J}| \leq r$ and a column-wise 2-sparse matrix $X \geq 0$ such that $B = AX$.

NP-completeness of SSNMF for any $2 \leq k \leq r$ follows directly as it would allow to solve 2-SSNMF by simply adding artificial columns of A (for example orthogonal to the ones used in the 2-sparse decomposition). In order to prove the NP-hardness of 2-SSNMF, we first demonstrate a polynomial time reduction from the well known NP-complete problem *SET-COVER* [27] to 2-SSNMF.

Problem 5.3 *SET-COVER*
Given: A finite set $S = \{1, ..., n\}$, a collection $C = \{C_1, ..., C_m\}$ of subsets of S and a positive integer $K \leq m$.
Question: Does $C' \subseteq C$ exist with $|C'| \leq K$ such that every element of S belongs to at least one member of C'.

From an instance (S, C, K) of SET-COVER, let us construct an instance (B, r) of 2-SSNMF in polynomial time.

- The natural number $r > 0$ is defined as

$$r = \sum_{i=1}^{m} |C_i| + 2 + K.$$

- The matrix B is the concatenation of three matrices B_1, B_2, B_3 such that $B = [B_1, B_2, B_3]$.
  - For each subset C_i of Problem 5.3 with $i = 1, ..., m$, we have the data point $B_1(:, i)$ defined as follows:

$$B_1(:, i) = \begin{pmatrix} 0 & -h_i \end{pmatrix}^T$$

    with $h_i = \frac{i}{m+1}$. Hence, B_1 is a 2-by-m matrix.
  - For each element $j = 1, ..., n$ of the ground set S, we have the data point $B_2(:, j)$ defined as follows:

$$B_2(:, j) = \begin{pmatrix} b_j & b_j^2 \end{pmatrix}^T$$

    with $b_j = \frac{1}{m+1+aj}$ and $a = \frac{1}{n}$. These points belong to the curve $y = x^2$. B_2 is a 2-by-n matrix.
  - When the jth element of the ground set S is a member of the ith subset C_i, we add a data point in B_3 as follows:

$$B_3(:, l) = \begin{pmatrix} \frac{h_i}{b_j} & \frac{h_i^2}{b_j^2} \end{pmatrix}^T$$

    with h_i and b_j as previously defined. Moreover, we add two more columns to B_3 for the data points $(0, 0)$ and $(0, -1)$. Hence, B_3 is a 2-by-$(2 + \sum_{i=1}^{m} |C_i|)$ matrix. Note that the intersection of the curve $y = x^2$ with the linear equation connecting $(0, -h_i)$ and (b_j, b_j^2) is precisely the point $(\frac{h_i}{b_j}, \frac{h_i^2}{b_j^2})$. We show in Lemma 5.2 that all these points never overlap. It implies that a straight line between the ith point of B_1 and a point in B_3 is passing through the jth point of B_2 if and only if j is in the subset C_i.

Lemma 5.2 *All the columns of B_3 are different.*

Proof 5.2 *Suppose it is not the case and that for (i, j) and (i', j') with $i \neq i'$ and $j \neq j'$, we have $\frac{h_i}{b_j} = \frac{h_{i'}}{b_{j'}}$, which means that, after rearrangement*

$$\frac{i}{i'} = \frac{m + 1 + aj'}{m + 1 + aj}. \tag{5.2}$$

For $j, j' = 1, ..., n$, the right-hand side of (5.2) varies as follows:

$$1 - a\left(\frac{n-1}{m+1+an}\right) \leq \frac{m+1+aj}{m+1+aj'} \leq 1 + a\left(\frac{n-1}{m+1+a}\right),$$

and when $a = \frac{1}{n}$, we have $a\left(\frac{n-1}{m+1+an}\right) < a\left(\frac{n-1}{m+1+a}\right) < \frac{1}{m+1}$, which means that the variation of the right-hand side around 1 is as follows

$$1 - \frac{1}{m+1} < \frac{m+1+aj}{m+1+aj'} < 1 + \frac{1}{m+1}.$$

For $i, i' = 1, ..., m+1$, the closest value to 1 of $\frac{i}{i'}$ is $\frac{m}{m+1}$, that is $1 - \frac{1}{m+1}$. Therefore, the choice of $a = \frac{1}{n}$ prevents the right-hand side of (5.2) to be equal to its left-hand side. It results that all the values $\frac{h_i}{b_j}$ are different for $i = 1, ..., m+1$ and $j = 1, ..., n$.

Lemma 5.3 *All the columns of B are situated inside the convex hull of the columns of B_3.*

Proof 5.3 *Except for $(0, -1)$, the columns of B_3 are located on the moment curve $y = x^2$. It results that these points are the vertices of a convex polygon, known under the name of cyclic polytope. The intersection of the y-axis and the line connecting any two points of the set $\{(x, y)|y = x^2, x \geq 1\}$ is located strictly below the point $(0, -1)$. Following the definitions of h_i and b_j, we have $\frac{h_i}{b_j} \geq 1$ for any $i = 1, ..., m$ and $j = 1, ..., n$, which means that even with the addition of $(0, -1)$, the points of B_3 still form a convex polygon. It is then easy to check that the points of B_1 and B_2 are inside the convex hull of B_3.*

Lemma 5.4 *The 2-SSNMF instance is a yes-instance if and only if the SET-COVER instance is a yes-instance.*

Proof 5.4 ***The if part.*** *Suppose we have an optimal cover $C' \subseteq C$ of the SET-COVER instance with $|C'| \leq K$. From this solution, we build a solution to the 2-SSNMF instance as follows:*

- *For the dictionary matrix A, we concatenate B_3 and the columns of B_1 corresponding to the subsets in C'. By this way, the number of columns of A is less or equal than $r = \sum_{i=1}^{m} |C_i| + 2 + K$.*
- *With B_3 being in the dictionary, it is easy to construct the columns of X corresponding to $[B_1, B_3]$ in B: it is trivial for B_3 and, for B_1, the two nonnegative entries of a column of X are the two coefficients of the convex combination of $(0, 0)$, $(0, -1)$. Moreover, since A also contains the K columns of B_1 corresponding to the cover C', every column coming from B_2 in B can be expressed as the convex combination of exactly two columns of A (see the reduction above). By this way, we have $X \geq 0$, a column-wise 2-sparse matrix, such that $B = AX$.*

The only if part. *Suppose that we have a solution (A, X) of the 2-SSNMF instance such that $B = AX$, A having at most r columns and $X \geq 0$ being a column-wise 2-sparse matrix. From this factorization, we show how to extract a cover C' made of at most K subsets. All the columns of B_3 are necessarily in A since they are the vertices of a convex polygon (see Lemma 5.3). Since, by construction, no convex combination of two points in B_3 can reach the n points in B_2, we must have $A = [B_3, A']\Pi$ with the columns of A' coming either from B_1 or from B_2. The number of columns of A' is therefore $r - (\sum_{i=1}^{m} |C_i| + 2) = K$. It remains to show how to construct a solution to the SET-COVER instance from A'. For every point in B_2, it is possible to find a point in B_1 and a point B_3 such that the three points are lined up (it is always possible to find such points since we suppose that every element of the ground set belongs to at least one subset in the SET-COVER instance). It means that we can replace all the columns coming from B_2 in A' by columns of B_1 without increasing the size of A'. In order to maintain the equality $B = AX$, it is easy to update the matrix X accordingly while keeping it column-wise 2-sparse. Finally, with the K columns of A' coming from B_1, we have identified a cover C' composed of K subsets for the SET-COVER instance.*

Proof 5.5 *Proof of Theorem 5.1. 2-SSNMF is in NP since we can check in polynomial time that a given pair (A, X) is a solution of a 2-SSNMF instance. With the reduction from the SET-COVER problem presented above and Lemma 5.4, we can conclude that 2-SSNMF is NP-hard.*

Illustration of the reduction. From the *SET-COVER* instance: $n = 5$, $m = 4$, $K = 2$, $C_1 = \{2, 4\}$, $C_2 = \{1, 2, 3\}$, $C_3 = \{3, 4\}$ and $C_4 = \{4, 5\}$, the reduction presented above leads to the following *2-SSNMF* instance: $r = 13$ and $B = [B_1, B_2, B_3]$ with

$$B_1 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -h_1 & -h_2 & -h_3 & -h_4 \end{pmatrix}, B_2 = \begin{pmatrix} b_1 & b_2 & b_3 & b_4 & b_5 \\ b_1^2 & b_2^2 & b_3^2 & b_4^2 & b_5^2 \end{pmatrix},$$

$$\text{and } B_3 = \begin{pmatrix} \frac{h_1}{b_2} & \frac{h_1}{b_4} & \frac{h_2}{b_1} & \frac{h_2}{b_2} & \frac{h_2}{b_3} & \frac{h_3}{b_3} & \frac{h_3}{b_4} & \frac{h_4}{b_4} & \frac{h_4}{b_5} & 0 & 0 \\ \frac{h_1^2}{b_2^2} & \frac{h_1^2}{b_4^2} & \frac{h_2^2}{b_1^2} & \frac{h_2^2}{b_2^2} & \frac{h_2^2}{b_3^2} & \frac{h_3^2}{b_3^2} & \frac{h_3^2}{b_4^2} & \frac{h_4^2}{b_4^2} & \frac{h_4^2}{b_5^2} & 0 & -1 \end{pmatrix},$$

where $h_i = \frac{i}{5}$ for $i = 1, ...4$ and $b_j = (5 + \frac{j}{5})^{-1}$ for $j = 1, ...5$ (see Figure 5.2).
A solution to the *SET-COVER* instance is $C' = \{C_2, C_4\}$ and the corresponding *2-SSNMF* solution is

$$A = \begin{pmatrix} \frac{h_1}{b_2} & \frac{h_1}{b_4} & \frac{h_2}{b_1} & \frac{h_2}{b_2} & \frac{h_2}{b_3} & \frac{h_3}{b_3} & \frac{h_3}{b_4} & \frac{h_4}{b_4} & \frac{h_4}{b_5} & 0 & 0 & 0 & 0 \\ \frac{h_1^2}{b_2^2} & \frac{h_1^2}{b_4^2} & \frac{h_2^2}{b_1^2} & \frac{h_2^2}{b_2^2} & \frac{h_2^2}{b_3^2} & \frac{h_3^2}{b_3^2} & \frac{h_3^2}{b_4^2} & \frac{h_4^2}{b_4^2} & \frac{h_4^2}{b_5^2} & 0 & -1 & -h_2 & -h_4 \end{pmatrix}, \text{ and}$$

$$X = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & \beta_{2,1} & 0 & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & 0 & \beta_{2,2} & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & 0 & 0 & \beta_{2,3} & 0 & 0 & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & I_{11} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_{4,4} & 0 & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \beta_{4,5} & \\
1-\alpha_1 & 1-\alpha_2 & 1-\alpha_3 & 1-\alpha_4 & 0 & 0 & 0 & 0 & 0 & \\
\alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & 0 & 0 & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & 1-\beta_{2,1} & 1-\beta_{2,2} & 1-\beta_{2,3} & 0 & 0 & 0...0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\beta_{4,4} & 1-\beta_{4,5} & 0...0
\end{pmatrix},$$

for which we have $B = AX$ when $\alpha_i = h_i$, $\beta_{i,j} = \frac{b_j^2}{h_i}$.

5.4 Proposed algorithm: Brassens

In the following, we assume that the input matrix B is k-sparse r-separable. Our algorithm, called Brassens[1], is presented formally in Algorithm 5.2.

Algorithm 5.2: Brassens

Input: A k-sparse-near-separable matrix $B \in \mathbb{R}^{m \times n}$, and the desired sparsity level k.

Output: A set of r indices $\mathcal{J}$, and a matrix $X \in \mathbb{R}_+^{r \times n}$, such that $\|X(:,j)\|_0 \leq k$ for all j and $B = \tilde{B}(:,\mathcal{J})X$.

1 $\mathcal{J} = \text{SNPA}(B, \infty)$
2 $\mathcal{J}' = \text{kSSNPA}(B, \infty, \mathcal{J})$
3 **foreach** $j \in \mathcal{J}'$ **do**
4 **if** $\min_{||h||_0 \leq k, h \geq 0} f(B(:,j) - B(:, \mathcal{J}' \setminus j)h) > 0$ **then**
5 $\mathcal{J} = \mathcal{J} \cup \{j\}$
6 $X = \text{arborescent}(B, B(:,\mathcal{J}), k)$

On line 1 we apply the original SNPA to select the exterior vertices; it is computationally cheap and ensures that these vertices are properly identified. The symbol ∞ means that SNPA stops only when the residual error is zero. For the noisy case, we replace the condition $R \neq 0$ by $R > \delta$, where δ is a user-provided noise-tolerance threshold.

Then, we adapt SNPA to impose a k-sparsity constraint on X: the projection step (line 8 of Algorithm 5.1) is replaced by a k-sparse projection step that imposes the columns of X to be k-sparse by solving (5.1). We call *kSSNPA* this modified version of SNPA. Note that, if $k = r$, kSSNPA reduces to SNPA.

On line 2 we apply kSSNPA to select candidate interior vertices. We provide it with the set $\mathcal{J}$ of exterior vertices so that they do not need to be identified again.

[1]It stands for Brassens Relies on Assumptions of Separability and Sparsity for Elegant NMF Solving.

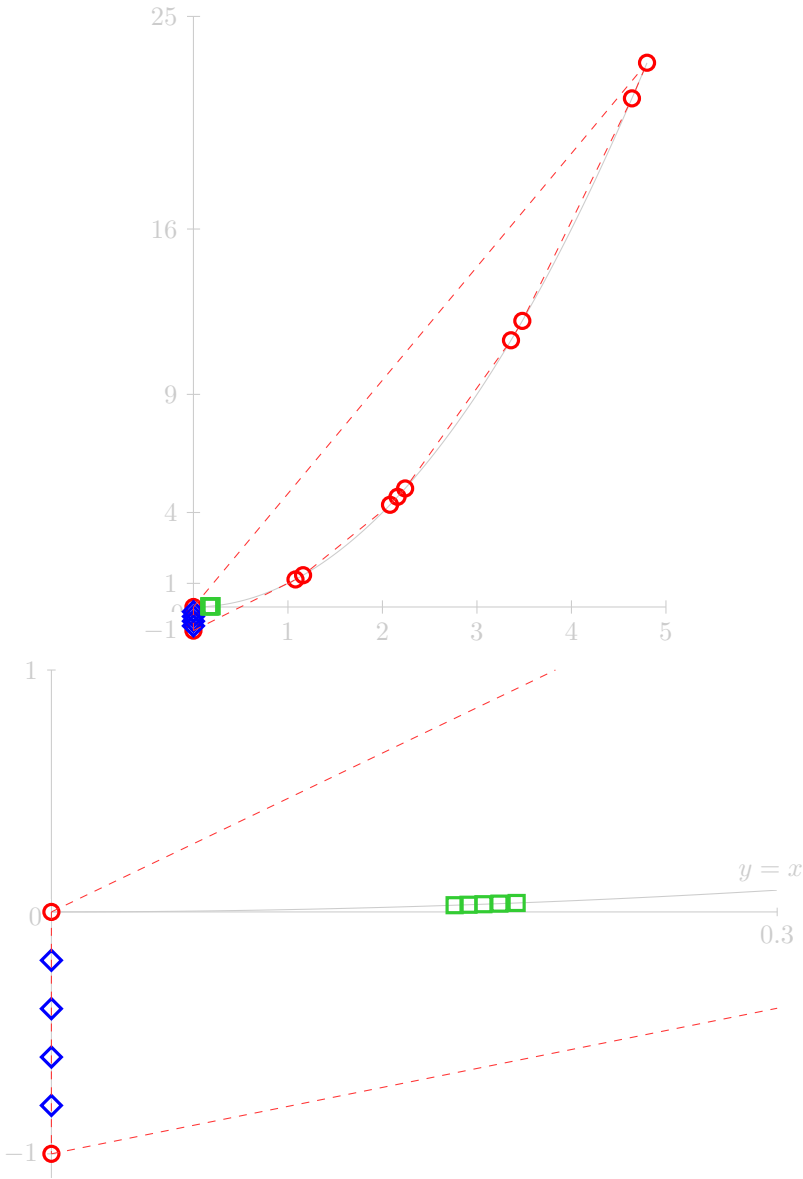


Figure 5.2: Example of a 2-SSNMF instance constructed from the following SET-COVER instance: $n = 5$, $m = 4$, $C_1 = \{2, 4\}$, $C_2 = \{1, 2, 3\}$, $C_3 = \{3, 4\}$ and $C_4 = \{4, 5\}$. The second picture is a zoom of the first picture on the $[0, 0.3] \times [-1, 1]$ box. Red circles correspond to the points of B_3, blue diamonds to the points of B_1 and green squares to the points of B_2. The red dashed lines are the edges of the convex hull of the points in B_3.

kSSNPA extracts columns of B as long as the norm of the residual $\|B - B(:, \mathcal{J}')X\|_F$ is larger than zero. At this point, all vertices have been identified: the exterior vertices have been identified by SNPA, while the interior vertices have been identified by kSSNPA because we will assume that they are not k-sparse combinations of any other data points; see Section 5.5 for the details. Hence, the error will be equal to zero if and only if all vertices have been identified. However, some selected interior points may not be interior vertices, because the selection step of kSSNPA chooses the point that is furthest away from the *k-sparse hull* of the selected points, that is, the union of the convex hulls of the subsets of k already selected points. For example, in Fig. 5.1b, if A_1, A_2, and A_3 are selected, the k-sparse hull is composed of the 3 segments $[A_1, A_2]$, $[A_2, A_3]$, and $[A_3, A_1]$. In this case, although only point A_4 is an interior vertex, point B_1 is selected before point A_4, because it is located further away from the k-sparse hull.

On lines 3 to 5, we apply a postprocessing to the selected points by checking whether they are k-sparse combinations of other selected points; this is a k-sparse NNLS problem solved with Arborescent (from Chapter 3). If they are, then they cannot be vertices and they are discarded, such as point B_1 in Figure 5.1b which belongs to the segment $[A_1, A_4]$.

Note that this "postprocessing" could be applied directly to the whole data set by selecting data points as the columns of A if they are not k-sparse combinations of other data points. However, this is not reasonable in practice, as it is equivalent to solving n times a k-sparse NNLS subproblem in $n - 1$ variables. The kSSNPA step can thus be interpreted as a safe screening technique, similarly as done in [24] for example, in order to reduce the number of candidate atoms from all the columns of B to a subset $\mathcal{J}'$ of columns. In practice, we have observed that kSSNPA is very effective at identifying good candidates points; see Section 5.6.1.

5.5 Analysis of Brassens

In this section, we first discuss the assumptions that guarantee Brassens to recover A given the k-sparse r-separable matrix B, and then discuss the computational complexity of Brassens.

5.5.1 Correctness

In this section, we show that, given a k-sparse r-separable matrix B, the algorithm Brassens provably solves SSNMF, that is, it is able to recover the correct set of indices $\mathcal{J}$ such that $A = B(:, \mathcal{J})$, under a reasonable assumption.

Clearly, a necessary assumption for Brassens to be able to solve SSNMF is that no column of A is a k-sparse nonnegative linear combinations of other columns of A, otherwise kSSNPA might set that column of A to zero, hence might not be able to extract it.

Assumption 5.1 *No column of A is a nonnegative linear combination of k other columns of A.*

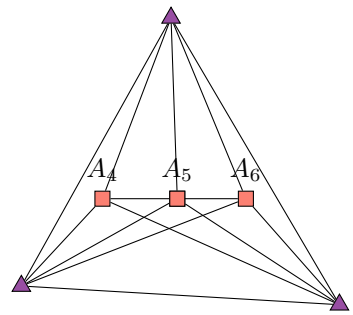


Figure 5.3: There are three interior vertices. One of them (A_5) is a combination of the others two.

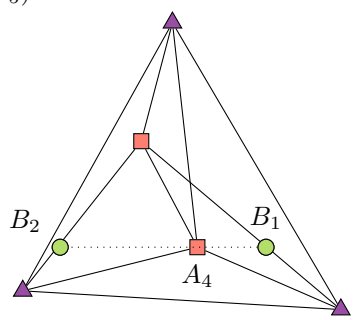


Figure 5.4: There are two interior vertices. One of them (A_4) is a 2-sparse combination of data points (B_1 and B_2).

Interestingly, unlike the standard separable case (that is, $k = r$), and although it is necessary in our approach with Brassens, Assumption 5.1 is not necessary in general to be able to uniquely recover A. Take for example the situation of Fig. 5.3, with three aligned points in the interior of a triangle, so that $r = 6$, $m = 3$ and $k = 2$. The middle point of these three aligned points is a 2-sparse combination of the other two, by construction. If there are data points on each segment joining these three interior points and the exterior vertices, the only solution to SSNMF with $r = 6$ is the one selecting these three aligned points. However, Assumption 5.1 is a reasonable assumption for SSNMF.

Unfortunately, Assumption 5.1 is not sufficient for Brassens to provably recover A. In fact, we need the following stronger assumption.

Assumption 5.2 *No column of A is a nonnegative linear combination of k other columns of B.*

This assumption guarantees that a situation such as the one shown on Figure 5.4 where one of the columns of A is a 2-sparse combination of two data points is not possible. In fact, in that case, if Brassens picks these two data points before the

interior vertex A_4 in between them, it will not be able to identify A_4 as it is set to zero within the projection step of kSSNPA.

Interestingly, in the standard separable case, that is, $k = r$, the two assumptions above coincide; this is the condition under which SNPA is guaranteed to work. Although Assumption 5.2 may appear much stronger than Assumption 5.1, they are actually generically equivalent given that the entries of the columns of X are generated randomly (that is, non-zero entries are picked at random and follow some continuous distribution). For instance, for $m = 3$ and $k = 2$, it means that no vertex is on a segment joining two data points. If the data points are generated randomly on the segments generated by any two columns of A, the probability for the segment defined by two such data points to contain a column of A is zero. In fact, segments define a set of measure zero in the unit simplex.

We can now provide a recovery result for Brassens.

Theorem 5.2 *Let $B = AX$ with $A = B(:, \mathcal{J})$ be a k-sparse r-separable matrix so that $|\mathcal{J}| = r$; see Definition 5.1. We have that*

- *If A satisfies Assumption 5.2, then the factor A with r columns in SSNMF is unique (up to permutation and scaling) and Brassens recovers it.*

- *If A satisfies Assumption 5.1, the entries of X are generated at random (more precisely, the position of the non-zero entries are picked at random, while their values follows a continuous ditribution) and $k < \text{rank}(B)$, then, with probability one, the factor A with r columns in SSNMF is unique (up to permutation and scaling) and Brassens recovers it.*

Proof 5.6 *Uniqueness of A in SSNMF under Assumption 5.2 is straightforward: since the columns of A are not k-sparse combinations of other columns of B, they have to be selected in the index set $\mathcal{J}$. Otherwise, since columns of A are among the columns of B, it would not possible to reconstruct B exactly using k-sparse combinations of $B(:, \mathcal{J})$. Then, since all other columns are k-sparse combinations of the r columns of A (by assumption), no other columns needs to be added to $\mathcal{J}$ which satisfies $|\mathcal{J}| = r$.*

Let us show that, under Assumption 5.2, Brassens recovers the correct set of indices $\mathcal{J}$. kSSNPA can only stop when all columns of A have been identified. In fact, kSSNPA stops when the reconstruction error is zero, while, under Assumption 5.2, this is possible only when all columns of A are selected (for the same reason as above). Then, the postprocessing will be able to identify, among all selected columns, the columns of A, because they will be the only ones that are not k-sparse combinations of other selected columns.

The second part of the proof follows from standard probabilistic results: since $k < \text{rank}(B)$, the combination of k data points generates a subspace of dimension smaller than that of $\text{col}(B)$. Hence, generating data points at random is equivalent to generating such subspaces at random. Since these subspaces form a space of measure zero in $\text{col}(B)$, the probability for these subspaces to contain a column of A is zero, which implies that Assumption 5.2 is satisfied with probability one.

5.5.2 Computational cost

Let us derive an upper bound on the computational cost of Brassens. First, recall that solving an NNLS problem up to any precision can be done in polynomial time. For simplicity and because we focus on the non-polynomial part of Brassens, we denote $\bar{\mathcal{O}}(1)$ the complexity of solving an NNLS problem. In the worst case, kSSNPA will extract all columns of B. In each of the n iterations of kSSNPA, the problem (5.1) needs to be solved. When $|\mathcal{J}| = \mathcal{O}(n)$, this requires to solve n times (one for each column of B) a k-sparse least squares problem in $|\mathcal{J}| = \mathcal{O}(n)$ variables. The latter requires in the worst case $\mathcal{O}(n^k)$ operations by trying all possible index sets; see the discussion after (5.1). In total, kSSNPA will therefore run in the worst case in time $\bar{\mathcal{O}}(n^{k+2})$.

Therefore, when k is fixed (meaning that k is considered as a fixed constant) and under Assumption 5.2, Brassens can solve SSNMF in polynomial time. Note that this is not in contradiction with our NP-completeness results when k is fixed (Theorem 5.1) because our NP-completeness proof does not rely on Assumption 5.2.

In summary, to make SSNMF hard, we need either k to be part of the input, or the columns of A to be themselves k-sparse combinations of other columns of A.

5.6 Experiments

The code and data are available online[2]. All experiments have been performed on a personal computer with an i5 processor, with a clock frequency of 2.30GHz. All algorithms are single-threaded. All are implemented in Matlab, except the sparse NNLS solver Arborescent, which is implemented in C++ with a Matlab MEX interface.

As far as we know, no algorithm other than Brassens can tackle SSNMF with more than one interior point (see Section 5.2.2) hence comparisons with existing works are unfortunately limited. For example, separable NMF algorithms can only identify the exterior vertices; see Section 5.1. However, we will compare Brassens to SNPA on a real multispectral image in Section 5.6.2, to show the advantages of the SSNMF model over separable NMF. In Section 5.6.1, we illustrate the correctness and efficiency of Brassens on synthetic data sets.

5.6.1 Synthetic data sets

In this section, we illustrate the behaviour of Brassens in different experimental setups. The generation of a synthetic data set is done as follows: for a given number of dimensions m, number of vertices r, number of data points n, and data sparsity k, we generate matrices $A \in \mathbb{R}_+^{m \times r}$ such that the last $r - m$ columns of A are linear combinations of the first m columns, and $X \in \mathbb{R}_+^{r \times n}$ such that $X = [I_r, X']$ and $\|X(:, j)\|_0 \leq k$ for all j. We use the uniform distribution in the interval [0,1] to generate random numbers (columns of A and columns of X), and then normalize the columns of A and X to have unit ℓ_1 norm. We then compute $B = AX$. This way, the matrix B is k-sparse r-separable, with r vertices, of which $r - m$ are interior vertices

[2] `https://gitlab.com/nnadisic/ssnmf`

(in fact, the first m columns of A are linearly independent with probability one as they are generated randomly). We then run Brassens on B, with the parameter k, and no noise-tolerance. For a given setup, we perform 30 rounds of generation and solving, and we measure the median of the running time and the median of the number of candidates extracted by kSSNPA. This number of candidates corresponds to $|\mathcal{J}'|$ in Algorithm 5.2, that is, the number of interior points selected by kSSNPA as potential interior vertices. Note that a larger number of candidates only results in an increased computation time, and does not change the output of the algorithm which is guaranteed to extract all vertices (Theorem 5.2).

Fig. 5.5 shows the behaviour of Brassens when n varies, with fixed $m = 3$, $k = 2$, and $r = 5$. To the best of our knowledge, this case is not handled by any other algorithm in the literature. Both the number of candidates and the run time grow slower than linear. The irregularities in the plot are due to the high variance between runs. Indeed, if vertices are generated in a way that some segments between vertices are very close to each other, Brassens typically selects more candidates before identifying all columns of A.

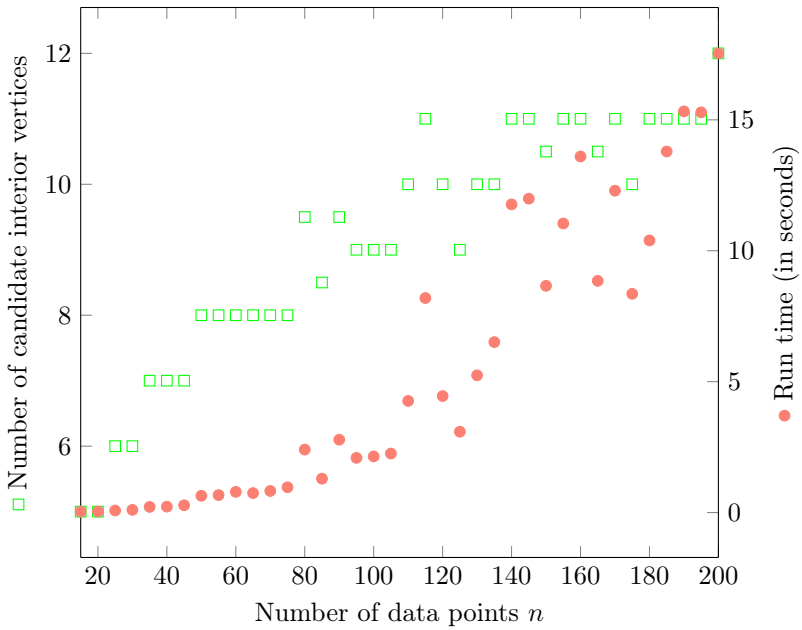


Figure 5.5: Results for Brassens on synthetic data sets for different values of n, with fixed $m = 3$, $k = 2$, and $r = 5$ with 3 exterior and 2 interior vertices. The values showed are the medians over 30 experiments.

In Table 5.1 we compare the performance of Brassens for several sets of parameters. The number of candidates grows relatively slowly as the dimensions (m, n) of the problem increase, showing the efficiency of the screening performed by kSSNPA. However, the run time grows rather fast when the dimensions (m, n) grow. This is

because, not only the number of NNLS subproblems to solve increase, but also their size. In all cases, as guaranteed by Theorem 5.2, Brassens was able to correctly

Table 5.1: Results for Brassens on synthetic data sets (median over 30 experiments).

m	n	r	k	Number of candidates	Run time in seconds
3	25	5	2	5.5	0.26
4	30	6	3	8.5	3.30
5	35	7	4	9.5	38.71
6	40	8	5	13	395.88

identify the columns of A. Again, as far as we know, no existing algorithms in the literature can perform this task.

To summarize, our synthetic experiments show the efficiency of the screening done by kSSNPA, and the capacity of Brassens to handle medium-scale data sets.

5.6.2 Blind multispectral unmixing

A multispectral image is an image composed of various wavelength ranges, called *spectral bands*, where every pixel is described by its *spectral signature*. This signature is a vector representing the amount of energy measured for this pixel in every considered spectral band. As opposed to hyperspectral images discussed in Section 1.6 and in previous chapters, multispectral images usually have a small number of bands (between 3 and 15). Similarly as in hyperspectral unmixing, multispectral unmixing consists in identifying the different materials present in that image, and here we also rely on the linear mixing model.

Let us apply Brassens to the unmixing of the well-known Urban satellite image [86], composed of 309×309 pixels. The original cleaned image has 162 bands, but we only keep 3 bands, namely the bands 2, 80, and 133 – these were obtained by selecting different bands with SPA applied on B^T – to obtain a data set of size $3 \times 94\,249$. The question is: can we still recover materials by using only 3 bands? (The reason for this choice is that this data set is well known and the ground truth is available, which is not the case of most multispectral images with only 3 bands.) We first normalize all columns of B so that they sum to one. Then, we run Brassens with a sparsity constraint $k = 2$ (this means that we assume that a pixel can be composed of at most 2 materials, which is reasonable for this relatively high resolution image) and a noise-tolerance threshold of 4%; this means that we stop SNPA and kSSNPA when $\|B - B(:, \mathcal{J})X\|_F \leq 0.04\|B\|_F$. Brassens extracts 5 columns of the input matrix. For comparison, we run SNPA with $r = 5$. Note that this setup corresponds to underdetermined blind unmixing, because $m = 3 < r = 5$. It would not be possible to tackle this problem using standard NMF algorithms (that would return a trivial solution such as $B = I_3 B$). It can be solved with SNPA, but SNPA cannot identify interior vertices.

SNPA extracts the 5 vertices in 3.8 seconds. Brassens extracts 5 vertices, including one interior vertex, in 33 seconds. The resulting abundance maps are showed in Fig. 5.6. They correspond to the reshaped rows of X, hence they show which pixel

contains which extracted material (they are more easily interpretable than the spectral signatures contained in the columns of A). The materials they contain are given in Table 5.2, using the ground truth from [85]. We see that Brassens produces a better solution, as the materials present in the image are better separated: the first three abundance maps of Brassens are sparser and correspond to well-defined materials. The last two abundances maps of SNPA and of Brassens are similar but extracted in a different order. The running time of Brassens is reasonable, although ten times higher than SNPA.

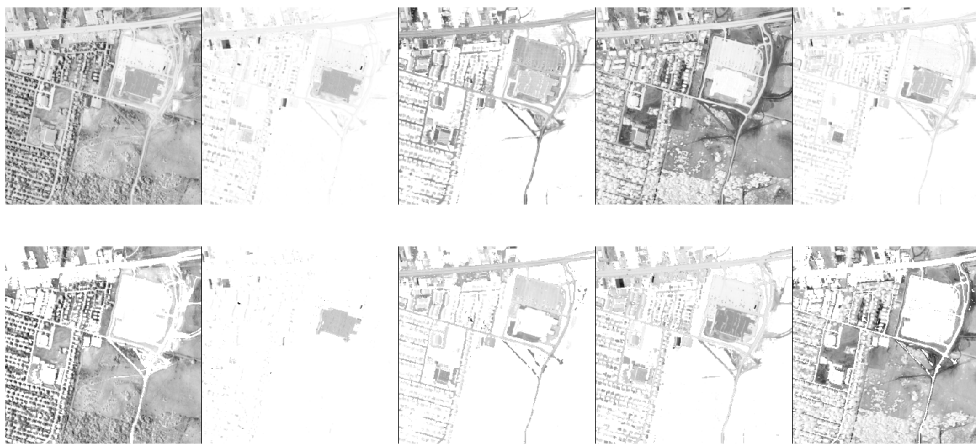


Figure 5.6: Abundances maps of materials (that is, reshaped rows of X) extracted by SNPA (top) and by Brassens (bottom) in the Urban image with only 3 spectral bands.

Table 5.2: Interpretation of the unmixing results from Figure 5.6.

Image	Materials extracted by SNPA	Materials extracted by Brassens
1	Grass + trees + roof tops	Grass + trees
2	Roof tops 1	Roof tops 1
3	Dirt + road + roof tops	Road
4	Dirt + grass	Roof tops 1 and 2 + road
5	Roof tops 1 + dirt + road	Dirt + grass

5.7 Conclusion

In this chapter, we introduced SSNMF, a new variant of the NMF model combining the assumptions of separability and sparsity. We proved this variant to be NP-complete, as opposed to separable NMF. We presented Brassens, an algorithm able to solve exactly SSNMF. It is based on SNPA and on Arborescent, the exact sparse NNLS solver presented in Chapter 3. We showed its efficiency for various setups and

in the successful unmixing of a multispectral image. The present work provides a new way to perform underdetermined blind source separation, under mild hypothesis, and a new way to regularize NMF. It makes NMF identifiable even when atoms of A are nonnegative linear combinations of other atoms (as long as these combinations have sufficiently many non-zero coefficients). Further work includes the theoretical analysis of the proposed model and algorithm in the presence of noise.

Chapter 6

Conclusion

> *"It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts."*
>
> – *Sherlock Holmes, A Scandal in Bohemia*, Arthur Conan Doyle

In this chapter, we summarize the contributions presented in this thesis and we draw some perspectives for future research. We invite the reader to go back to Fig. 1.3 on page 32 for a graphical overview of this thesis. Our contributions are centered around the NMF problem,

$$\min_{A,X} \|B - AX\|_F^2 \quad \text{such that} \quad A \geq 0 \text{ and } X \geq 0.$$

- In Chapter 1, we discussed the motivation for the contributions of this thesis and gave some technical background about the models considered.

- In Chapter 2, we studied smoothed separable NMF, a variant of the separable NMF model to estimate A. It is based on the observation that, when the separability assumption holds, then for each vertex $A(:, j)$ there are more than one data point $B(:, l)$ close to this vertex. Based on the algorithm ALLS that leverages this observation, we proposed the two algorithms smoothed VCA and smoothed SPA. We showed empirically that they outperform both their non-smoothed counterparts (VCA and SPA) and ALLS by being more tolerant to noise and outliers in both synthetic experiments and the unmixing of hyperspectral images.

- In Chapter 3, we proposed a branch-and-bound algorithm called Arborescent to solve exactly the k-sparse NNLS problem (1.3). It can notably be used to compute X with a column-wise k-sparsity constraint ($\|X(:, j)\|_0 \leq k$ for all j), given A. We showed empirically that Arborescent works in very noisy and ill-conditioned settings, where other approaches completely fail to recover the true solution. It is able to enforce sparsity while limiting the loss in reconstruction

error. When the dimension r is small, which is generally the case in hyperspectral unmixing, Arborescent performs exact solving in a reasonable computing time. We also proposed its extension to solve the biobjective k-sparse NNLS problem (1.4).

- In Chapter 4, we proposed an MNNLS model with a matrix-wise ℓ_0-constraint. Given A, it sums up to estimating X such that $\|X\|_0 \leq q$ for some parameter q. The motivation for this model is to enforce sparsity in a finer way than the column-wise constraint. We proposed Salmon, a two-step algorithm to solve it. Salmon consists in (i) dividing the sparse MNNLS problem into n one-column biobjective subproblems and compute a Pareto front for each subproblem, and (ii) select one solution per column (hence per Pareto front) to build a sparsity-constrained matrix. Step (i) can be done with Arborescent to compute exactly the Pareto front, or with a greedy algorithm or the homotopy algorithm for an approximate but faster solving. Step (ii) is done with a dedicated selection algorithm that we proved near-optimal despite its greedy nature. With experiments on real-world faces datasets and hyperspectral images, we showed that Salmon produces better solutions and that the matrix-wise ℓ_0-constraint is more appropriate and effective that the column-wise constraint in these settings.

- In Chapter 5, we proposed sparse separable NMF, a new NMF variant to estimate A that combines the assumptions of separability and column-wise k-sparsity of X. This model handles the underdetermined setting where $r > m$. It is also the first algorithm to handle the "interior vertex case" where one vertex is a linear combination of other vertices. We proved this new model to be NP-complete, as opposed to separable NMF that is solvable in polynomial time. We proposed an algorithm called Brassens to solve it, and proved that Brassens is guaranteed to recover the vertices in the noiseless case and under mild assumptions. We showed both theoretically and empirically on real-world multispectral images that our proposed method handles well underdetermined problems even with interior vertices.

Perspectives

We believe that several perspectives are opened by the contributions of this thesis.

Smoothed separable NMF can pave the way for a whole new class of separable NMF algorithms. The assumption of this new model generally holds in real-world datasets where the separability assumption holds, and many applications could benefit from new algorithms that are more robust to noise and outliers. We also believe this new assumption can be easily taken into account in existing algorithms for separable NMF.

The exact methods for ℓ_0-constrained optimization Arborescent and Salmon could inspire new approaches in other sparse optimization problems, for example simultaneous sparse approximation [74, 79] where the columns of X are constrained to be sparse and to share the same support. This could be useful, for example, in the case of

a fixed over-complete dictionary A, to select only a subset of columns of A to explain the data. In addition, these combinatorial algorithms could inspire novel methods for enforcing other discrete constraints in least square problems or matrix factorization. This is not common in the current state of the art, and it could be useful in many applications. For instance, constraining a factor to take values within a discrete set (for instance, the integers between 1 and 5) would be relevant in applications such as recommender systems [53, 76] and may be solvable with a branch-and-bound-like method to prune the search space. Binary constraints [84] also lead to combinatorial problems and may be tackled in a similar way.

Focusing back on ℓ_0-constraints, we could benefit from recent advances such as the work by Ben Mhenni et al. [8]. They developed branch-and-bound algorithms for sparse least squares problems concomitantly with our development of Arborescent, and their approach is more scalable in some settings. Adapting these algorithms to the nonnegative setting would be of great interest and they could handle the cases where Arborescent shows limitations, for example, when r is very large and k is small. We may also be able to extend them to tackle biobjective sparse problems.

Finally, we applied our novel models and methods only for feature extraction in images and for the unmixing of hyperspectral images. Sparsity and nonnegativity are known to be realistic assumptions in many other applications such as topic modeling, chemometrics, or audio source separation. Experimenting with our algorithms on these new types of data could shed a different light on the properties of these datasets, determine better the strengths and limitations of our approaches, and motivate further contributions.

> *"We can only see a short distance ahead, but we can see plenty there that needs to be done."*
>
> – Alan Turing

Bibliography

1. Abdolali, M. & Gillis, N. Simplex-structured matrix factorization: Sparsity-based identifiability and provably correct algorithms. *SIAM Journal on Mathematics of Data Science* **3,** 593–623 (2021).
2. Aharon, M., Elad, M. & Bruckstein, A. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing* **54,** 4311–4322 (2006).
3. Aharon, M., Elad, M. & Bruckstein, A. *K-SVD and its non-negative variant for dictionary design* in *Wavelets XI* (San Diego, California, USA, 2005).
4. Araújo, U., Saldanha, B., Galvão, R., Yoneyama, T., Chame, H. & Visani, V. The successive projections algorithm for variable selection in spectroscopic multicomponent analysis. *Chemometrics and Intelligent Laboratory Systems* **57,** 65–73 (2001).
5. Arora, S., Ge, R., Halpern, Y., Mimno, D., Moitra, A., Sontag, D., Wu, Y. & Zhu, M. *A practical algorithm for topic modeling with provable guarantees* in *Proceedings of the 30th International Conference on Machine Learning* (2013), 280–288.
6. Arora, S., Ge, R., Kannan, R. & Moitra, A. *Computing a nonnegative matrix factorization – provably* in *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing* (2012), 145–162.
7. Bakshi, A., Bhattacharyya, C., Kannan, R., Woodruff, D. P. & Zhou, S. *Learning a latent simplex in input sparsity time* in *Proceedings of the International Conference on Learning Representations (ICLR)* (2021).
8. Ben Mhenni, R., Bourguignon, S. & Ninin, J. Global optimization for sparse solution of least squares problems. *Optimization Methods and Software,* 1–30 (2021).
9. Bertsimas, D. & Shioda, R. Algorithm for cardinality-constrained quadratic optimization. *Computational Optimization and Applications* **43,** 1–22 (2009).
10. Bezanson, J., Edelman, A., Karpinski, S. & Shah, V. B. Julia: A fresh approach to numerical computing. *SIAM review* **59,** 65–98 (2017).
11. Bhattacharyya, C. & Kannan, R. *Finding a latent k–simplex in $O^*(k \cdot nnz(data))$ time via subset smoothing* in *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms* (2020), 122–140.

12. Bienstock, D. Computational study of a family of mixed-integer quadratic programming problems. *Mathematical Programming* **74,** 121–140 (1996).

13. Bioucas-Dias, J. M., Plaza, A., Dobigeon, N., Parente, M., Du, Q., Gader, P. & Chanussot, J. Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* **5,** 354–379 (2012).

14. Boardman, J. W., Kruse, F. A. & Green, R. O. *Mapping target signatures via partial unmixing of AVIRIS data* in *Proc. Summary JPL Airborne Earth Science Workshop, Pasadena, CA* (1995), 23–26.

15. Bourguignon, S., Ninin, J., Carfantan, H. & Mongeau, M. Exact sparse approximation problems via mixed-integer programming: Formulations and computational performance. *IEEE Transactions on Signal Processing* **64,** 1405–1419 (2015).

16. Bruckstein, A., Elad, M. & Zibulevsky, M. On the uniqueness of nonnegative sparse solutions to underdetermined systems of equations. *IEEE Transactions on Information Theory* **54,** 4813–4820 (2008).

17. Chen, S., Billings, S. A. & Luo, W. Orthogonal least squares methods and their application to non-linear system identification. *International Journal of Control* **50,** 1873–1896 (1989).

18. Cichocki, A., Phan, A. H. & Caiafa, C. *Flexible HALS algorithms for sparse non-negative matrix/tensor factorization* in *2008 IEEE Workshop on Machine Learning for Signal Processing* (2008), 73–78.

19. Cichocki, A., Zdunek, R. & Amari, S.-i. in *Independent component analysis and signal separation* (eds Davies, M. E., James, C. J., Abdallah, S. A. & Plumbley, M. D.) 169–176 (2007).

20. Cohen, J. E. & Gillis, N. *Nonnegative low-rank sparse component analysis* in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), 8226–8230.

21. Donoho, D. L. & Tsaig, Y. Fast solution of ℓ_1-norm minimization problems when the solution may be sparse. *IEEE Transactions on Information theory* **54,** 4789–4812 (2008).

22. Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., *et al.* Least angle regression. *The Annals of statistics* **32,** 407–499 (2004).

23. Eggert, J. & Korner, E. *Sparse coding and NMF* in *IEEE International Joint Conference on Neural Networks* **4** (2004), 2529–2533.

24. El Ghaoui, L., Viallon, V. & Rabbani, T. *Safe feature elimination in sparse supervised learning* tech. rep. (UC/EECS-2010-126, EECS Dept., University of California at Berkeley, 2010).

25. Foucart, S. & Koslicki, D. Sparse recovery by means of nonnegative least squares. *IEEE Signal Processing Letters* **21,** 498–502 (2014).

26. Fu, X., Ma, W.-K., Huang, K. & Sidiropoulos, N. D. Blind separation of quasi-stationary sources: Exploiting convex geometry in covariance domain. *IEEE Transactions on Signal Processing* **63,** 2306–2320 (2015).
27. Garey, M. R. & Johnson, D. S. *Computers and Intractability, vol. 29* (WH Freeman, New York, 2002).
28. Ge, R. & Zou, J. *Intersecting faces: Non-negative matrix factorization with new guarantees* in *Proceedings of the 32nd International Conference on Machine Learning* (2015), 2295–2303.
29. Gillis, N. *Nonnegative Matrix Factorization* (SIAM, Philadelphia, 2020).
30. Gillis, N. Robustness analysis of Hottopixx, a linear programming model for factoring nonnegative matrices. *SIAM Journal on Matrix Analysis and Applications* **34,** 1189–1212 (2013).
31. Gillis, N. Sparse and unique nonnegative matrix factorization through data preprocessing. *Journal of Machine Learning Research* **13,** 3349–3386 (2012).
32. Gillis, N. Successive nonnegative projection algorithm for robust nonnegative blind source separation. *SIAM Journal on Imaging Sciences* **7,** 1420–1450 (2014).
33. Gillis, N. The why and how of nonnegative matrix factorization. *Regularization, Optimization, Kernels, and Support Vector Machines* **12,** 257–291 (2014).
34. Gillis, N. & Glineur, F. Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization. *Neural computation* **24,** 1085–1105 (2012).
35. Gillis, N. & Luce, R. Robust near-separable nonnegative matrix factorization using linear optimization. *Journal of Machine Learning Research* **15,** 1249–1280 (2014).
36. Gillis, N. & Vavasis, S. A. Fast and robust recursive algorithms for separable nonnegative matrix factorization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **36,** 698–714 (2014).
37. Gong, M., Jiang, X., Li, H. & Tan, K. C. Multiobjective sparse non-negative matrix factorization. *IEEE Transactions on Cybernetics,* 1–14 (2018).
38. Hoyer, P. O. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research* **5,** 1457–1469 (2004).
39. Hoyer, P. O. *Non-negative sparse coding* in *Proceedings of the 12th IEEE Workshop On Neural Networks for Signal Processing* (2002), 557–565.
40. Itoh, Y., Duarte, M. F. & Parente, M. Perfect recovery conditions for non-negative sparse modeling. *IEEE Transactions on Signal Processing* **65,** 69–80 (2017).
41. Jiang, J.-H., Liang, Y. & Ozaki, Y. Principles and methodologies in self-modeling curve resolution. *Chemometrics and Intelligent Laboratory Systems* **71,** 1–12 (2004).

42. Kervazo, C., Liaudat, T. & Bobin, J. Faster and better sparse blind source separation through mini-batch optimization. *Digital Signal Processing* **106,** 102827 (2020).

43. Kim, H. & Park, H. Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis. *Bioinformatics* **23,** 1495–1502 (2007).

44. Kim, J., Ramakrishnan, N., Marwah, M., Shah, A. & Park, H. *Regularization paths for sparse nonnegative least squares problems with applications to life cycle assessment tree discovery* in *IEEE 13th International Conference on Data Mining* (2013), 360–369.

45. Kumar, A., Sindhwani, V. & Kambadur, P. *Fast conical hull algorithms for near-separable non-negative matrix factorization* in *Proceedings of the 30th International Conference on Machine Learning* (2013).

46. Lawson, C. L. & Hanson, R. J. *Solving least squares problems* (Society for Industrial and Applied Mathematics, 1995).

47. Lee, D. D. & Seung, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature* **401,** 788–791 (1999).

48. Lee, D. D. & Seung, H. S. *Unsupervised learning by convex and conic coding* in *Advances in Neural Information Processing Systems* (1997), 515–521.

49. Leplat, V., Gillis, N. & Idier, J. Multiplicative Updates for NMF with β-Divergences under Disjoint Equality Constraints. *SIAM Journal on Matrix Analysis and Applications* **42,** 730–752 (2021).

50. Lin, C.-H., Chi, C.-Y., Wang, Y.-H. & Chan, T.-H. A fast hyperplane-based minimum-volume enclosing simplex algorithm for blind hyperspectral unmixing. *IEEE Transactions on Signal Processing* **64,** 1946–1961 (2015).

51. Lin, C.-H., Ma, W.-K., Li, W.-C., Chi, C.-Y. & Ambikapathi, A. Identifiability of the simplex volume minimization criterion for blind hyperspectral unmixing: The no-pure-pixel case. *IEEE Transactions on Geoscience and Remote Sensing* **53,** 5530–5546 (2015).

52. Lin, C.-H., Wu, R., Ma, W.-K., Chi, C.-Y. & Wang, Y. Maximum volume inscribed ellipsoid: A new simplex-structured matrix factorization framework via facet enumeration and convex optimization. *SIAM Journal on Imaging Sciences* **11,** 1651–1679 (2018).

53. Luo, X., Zhou, M., Xia, Y. & Zhu, Q. An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. *IEEE Transactions on Industrial Informatics* **10,** 1273–1284 (2014).

54. Ma, W.-K., Bioucas-Dias, J. M., Chan, T.-H., Gillis, N., Gader, P., Plaza, A. J., Ambikapathi, A. & Chi, C.-Y. A signal processing perspective on hyperspectral unmixing: Insights from remote sensing. *IEEE Signal Processing Magazine* **31,** 67–81 (2014).

55. Morup, M., Madsen, K. H. & Hansen, L. K. *Approximate L0 constrained non-negative matrix and tensor factorization* in *2008 IEEE International Symposium on Circuits and Systems* (IEEE, 2008), 1328–1331.

56. Naanaa, W. & Nuzillard, J.-M. Blind source separation of positive and partially correlated data. *Signal Processing* **85,** 1711–1722 (2005).

57. Nadisic, N., Cohen, J. E., Vandaele, A. & Gillis, N. Matrix-wise ℓ_0-constrained sparse nonnegative least squares. *preprint arXiv:2011.11066* (2022).

58. Nadisic, N., Gillis, N. & Kervazo, C. Smoothed separable nonnegative matrix factorization. *preprint arXiv:2110.05528* (2021).

59. Nadisic, N., Vandaele, A., Cohen, J. E. & Gillis, N. *Sparse separable nonnegative matrix factorization* in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECMLPKDD)* (2020), 335–350.

60. Nadisic, N., Vandaele, A., Gillis, N. & Cohen, J. E. *Exact biobjective k-sparse nonnegative least squares* in *29th European Signal Processing Conference (EUSIPCO)* (2021), 2079–2083.

61. Nadisic, N., Vandaele, A., Gillis, N. & Cohen, J. E. *Exact sparse nonnegative least squares* in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020), 5395–5399.

62. Nascimento, J. M. & Bioucas-Dias, J. M. Hyperspectral unmixing based on mixtures of Dirichlet components. *IEEE Transactions on Geoscience and Remote Sensing* **50,** 863–878 (2011).

63. Nascimento, J. M. & Bioucas-Dias, J. M. Vertex component analysis: A fast algorithm to unmix hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing* **43,** 898–910 (2005).

64. Nguyen, T. T., Soussen, C., Idier, J. & Djermoune, E.-H. *NP-hardness of ℓ_0 minimization problems: revision and extension to the non-negative setting* in *13th International conference on Sampling Theory and Applications (SampTA)* (2019), 1–4.

65. Nguyen, T. T., Idier, J., Soussen, C. & Djermoune, E.-H. Non-negative orthogonal greedy algorithms. *IEEE Transactions on Signal Processing,* 1–16 (2019).

66. Osborne, M. R., Presnell, B. & Turlach, B. A. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis* **20,** 389–403 (2000).

67. Paatero, P. & Tapper, U. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics* **5,** 111–126 (1994).

68. Pati, Y. C., Rezaiifar, R. & Krishnaprasad, P. S. *Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition* in *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers* (1993), 40–44.

69. Peharz, R. & Pernkopf, F. Sparse nonnegative matrix factorization with l0-constraints. *Neurocomputing* **80,** 38–46 (2012).

70. Portugal, L. F., Judice, J. J. & Vicente, L. N. A comparison of block pivoting and interior-point algorithms for linear least squares problems with nonnegative variables. *Mathematics of Computation* **63,** 625–643 (1994).

71. Rapin, J., Bobin, J., Larue, A. & Starck, J.-L. Sparse and non-negative BSS for noisy data. *IEEE Transactions on Signal Processing* **61,** 5620–5632 (2013).

72. Recht, B., Re, C., Tropp, J. & Bittorf, V. *Factoring nonnegative matrices with linear programs* in *Advances in Neural Information Processing Systems (NIPS)* (2012), 1214–1222.

73. Soussen, C., Gribonval, R., Idier, J. & Herzet, C. Joint k-step analysis of orthogonal matching pursuit and orthogonal least squares. *IEEE Transactions on Information Theory* **59,** 3158–3174 (2013).

74. Stojnic, M., Parvaresh, F. & Hassibi, B. On the reconstruction of block-sparse signals with an optimal number of measurements. *IEEE Transactions on Signal Processing* **57,** 3075–3085 (2009).

75. Sun, Y. & Xin, J. Underdetermined sparse blind source separation of nonnegative and partially overlapped data. *SIAM Journal on Scientific Computing* **33,** 2063–2094 (2011).

76. Thanh, O. V., Gillis, N. & Lecron, F. Bounded simplex-structured matrix factorization (2022).

77. Tibshirani, R. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society: Series B (Methodological)* **58,** 267–288 (1996).

78. Tropp, J. A. Just relax: convex programming methods for identifying sparse signals in noise. *IEEE Transactions on Information Theory* **52,** 1030–1051 (2006).

79. Tropp, J. A., Gilbert, A. C. & Strauss, M. J. Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit. *Signal Processing* **86,** 572–588 (2006).

80. Vavasis, S. A. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization* **20,** 1364–1377 (2010).

81. Winter, M. E. *N-FINDR: An algorithm for fast autonomous spectral end-member determination in hyperspectral data* in *Proceedings of the SPIE Conference on Imaging Spectrometry V* **3753** (1999), 266–276.

82. Wu, R., Ma, W.-K., Li, Y., So, A. M.-C. & Sidiropoulos, N. D. Probabilistic simplex component analysis. *preprint arXiv:2103.10027* (2021).

83. Yaghoobi, M., Wu, D. & Davies, M. E. Fast non-negative orthogonal matching pursuit. *IEEE Signal Processing Letters* **22,** 1229–1233 (2015).

84. Zhang, Z., Li, T., Ding, C. & Zhang, X. *Binary matrix factorization with applications* in *Seventh IEEE International Conference On Data Mining (ICDM 2007)* (2007), 391–400.

85. Zhu, F. Hyperspectral unmixing: Ground truth labeling, datasets, benchmark performances and survey. *preprint arXiv:1708.05125* (2017).

86. Zhu, F., Wang, Y., Xiang, S., Fan, B. & Pan, C. Structured sparse method for hyperspectral unmixing. *ISPRS Journal of Photogrammetry and Remote Sensing* **88,** 101–118 (2014).