Accelerating sparse inverse problems using structured approximations

Cássio Fraga Dantas

THÈSE DE DOCTORAT DE

L'UNIVERSITE DE RENNES 1
COMUE **U**NIVERSITE **B**RETAGNE **L**OIRE

Ecole Doctorale N°601
Mathèmatique et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Signal, Image, Vision*

Par

Cássio FRAGA DANTAS

Accelerating sparse inverse problems using structured approximations

Thèse présentée et soutenue à Rennes, le 29 novembre, 2019
Unité de recherche : Institut de Recherche en Informatique et Systèmes Aléatoires (IRISA)
Thèse N° :

Rapporteurs avant soutenance :

Emilie CHOUZENOUX	Chargée de recherche, Inria
Cédric FEVOTTE	Directeur de recherche, CNRS - IRIT

Composition du jury :

Président :		
Examinateurs :	Laurent ALBERA	Maître de conférences, l'Université de Rennes 1
	Christian JUTTEN	Professeur, Université Joseph Fourier
	Joseph SALMON	Professeur, Université de Montpellier
Dir. de thèse :	Rémi GRIBONVAL	Directeur de recherche, Inria
Invité(s)	Jérémy COHEN	Chargé de recherche, CNRS - IRISA

Acknowledgement

Coming soon.

Résumé étendu

Ce résumé présente de manière concise les différents travaux abordés dans cette thèse. Les détails techniques concernant les outils utilisés et les méthodes proposées sont donnés dans la suite du manuscrit (en anglais).

Introduction

Les problèmes inverses consistent à déterminer les causes expliquant un ensemble de données observées. Ces observations sont souvent indirectes, ce qui signifie que les données d'intérêt ont subi un processus de transformation, et le défi de ce type de problème mathématique est précisément d'inverser ce processus sous-jacent. Souvent on ne dispose que d'une quantité insuffisante d'observations pour pouvoir accomplir cette tâche. Mathématiquement, cela implique l'existence de plus d'inconnues (degrés de liberté) que d'équations (contraintes), et le problème est dit *sous-déterminé* ou simplement *mal posé*. Il est donc nécessaire de rajouter des contraintes supplémentaires en s'appuyant sur des *aprioris* sur les signaux d'entrée ou l'application visée.

Un apriori très répandu dans plusieurs domaines (que ce soit le traitement du signal, les statistiques ou l'apprentissage automatique) est la *parcimonie*. La parcimonie suppose que les signaux peuvent être décrits comme une combinaison de très peu d'éléments d'une collection (les atomes d'un dictionnaire). Ce paradigme donne lieu à ce qu'on appelle les problèmes inverses à contraintes de parcimonie, qui ont attiré une attention croissante dans les dernières années et dont les exemples d'application vont de la restauration audio ou d'images à la tomographie sismique. Il est donc crucial de pouvoir résoudre ces problèmes de façon efficace, ce qui n'est pas une tâche facile vu qu'il n'existe pas de solution analytique et qu'il faut donc recourir à des méthodes computationnelles itératives. De nombreuses études ont été menées dans cette direction et ont donné lieu à des méthodes itératives aujourd'hui bien répandues et aux bonnes performances. Cependant, lorsque la taille des problèmes augmente, en raison notamment de la vertigineuse croissance des données disponibles, la complexité computationnelle de ces algorithmes peut vite devenir un goulot d'étranglement.

Objectif général de la thèse

Accélérer la résolution de problèmes inverses linéaires à contrainte de parcimonie en combinant deux approches principales :

1. L'utilisation de dictionnaires structurés rapides à manipuler (détaillée en partie II) ;
2. L'élimination d'atomes (colonnes de la matrice dictionnaire) inutiles via les tests d'élagage sûrs (détaillée en partie III).

Cette thèse est organisée en trois parties (détaillées dans la suite) : une première partie de revue de la littérature et des outils utilisés dans le reste du manuscrit ; deux autres parties recueillant les contributions de cette thèse.

Première partie

En partie I, nous passons en revue les modèles, outils d'optimisation et d'autres concepts mathématiques qui serviront de base pour les contributions présentés en parties II et III.

Le chapitre 1 décrit différentes formulations mathématiques d'un problème inverse à contrainte de parcimonie. Les deux principales familles étudiées sont : 1) Formulations contraintes ou régularisées en norme ℓ_0 (opérateur qui compte le nombre d'entrées non-nulles d'un vecteur). Cela donne lieu à des problèmes non-convexes et par conséquent assez durs à résoudre. 2) Relaxations convexes basées sur la norme ℓ_1 (qui est donnée par la somme des valeurs absolues des entrées d'un vecteur), ce qui mène à des problèmes abordables par des techniques d'optimisation convexe. Nous mentionnons également d'autres façons de promouvoir la parcimonie de la solution et rappelons quelques garanties théoriques liées à la résolution de ces problèmes.

Dans les problèmes mentionnés précédemment, la matrice dictionnaire est supposée connue et fixe. Dans certains cas, pourtant, il peut être intéressant d'apprendre le dictionnaire sur une base de données. Cela donne lieu à la tâche d'*apprentissage de dictionnaire* que nous abordons également en chapitre 1. Nous posons les problèmes d'optimisation associés et rappelons quelques algorithmes importants de la littérature.

Au chapitre 2, nous présentons quelques outils d'optimisation utiles pour la reconstruction parcimonieuse de signaux. En particulier, nous détaillons les principaux algorithmes associés à chacune des deux familles de problèmes mentionnés au chapitre précédent. Pour la formulation en norme ℓ_0, nous nous concentrons sur les approches gloutonnes de type

Matching Pursuit (MP) et ses variations comme le Matching Pursuit Orthogonal (OMP) ainsi que ses différentes implémentations possibles. En ce qui concerne les formulations en norme ℓ_1, nous étudions les algorithmes de type gradient proximal comme ISTA (algorithme de seuillage doux itératif) et sa variante accélérée FISTA. D'autres approches connexes sont brièvement décrites, sans prétention de dresser une liste exhaustive des méthodes existantes. Finalement, nous étudions le principe de dualité Lagrangienne sur la base duquel nous dérivons le dual du problème Lasso et ses conditions d'optimalité. Ces derniers concepts seront particulièrement utiles en partie III.

Une partie importante des méthodes proposées dans cette thèse utilise des outils tensoriels. C'est pourquoi le chapitre 3 est dédié à étudier le formalisme tensoriel, ainsi qu'à donner un aperçu des concepts et outils de base de ce domaine : des différentes formes de représentation et de manipulation des tenseurs, aux principales décompositions tensorielles, en passant par des opérations tensorielles classiques (produit externe, produit de Kronecker et le produit tensoriel générique). Ces concepts seront utiles plus particulièrement dans la partie II du manuscrit. Par example, une des transformées présentées est la décomposition Canonique Polyadique (CPD) qui exprime un tenseur comme combinaison linéaire de tenseurs de rang unitaire. Cette transformée est ensuite utilisée aux chapitres 4 et 5 respectivement pour approcher des matrices en tant que somme de produits de Kronecker et pour apprendre des dictionnaires ayant ce même type de structure.

Deuxième partie

La partie II rassemble les contributions liées aux dictionnaires structurés rapides.

Au chapitre 4 nous abordons le problème d'approximation de matrices par des matrices structurées. Plus particulièrement, nous proposons une famille de matrices structurées qui s'écrivent comme une somme de R produits de Kronecker à K termes – ou, en notation plus concise, des matrices dites (R, K)-KS (*Kronecker structured*). Nous appelons ce modèle HO-SuKro (*Higher Order Sum of Kroneckers*). En plus d'émerger naturellement lors de la manipulation de donnés multi-dimensionnelles, les matrices ayant ce type de structure sont déterminées par un nombre de paramètres inférieur à celui d'une matrice non structurée, ce qui implique plusieurs avantages :

1. Coût de stockage réduit ;
2. Coût de calcul réduit en opérations de type matrice-vecteur ;
3. Complexité statistique réduite, c.à.d. un nombre inférieur de données est requis lors de l'entrainement.

Un algorithme simple est proposé pour approcher un opérateur linéaire arbitraire (de dimension finie) par une matrice (R, K)-KS, où K correspond à l'ordre tensoriel des données ciblées. Tandis que la décomposition en valeurs singulières (SVD) peut être utilisée pour le cas $K = 2$, la décomposition tensorielle canonique polyadique (CPD) rend possible la généralisation à des ordres supérieurs $K > 2$.

Au chapitre 5 nous considérons le problème d'apprentissage de dictionnaires structurés. Dans un premier temps, nous proposons une méthode de type gradient projeté pour apprendre des dictionnaires HO-SuKro, dans laquelle l'opérateur de projection utilise la CPD. Dans un deuxième temps, une approche plus efficace de type moindres carrés alternés est proposée pour cette même tâche. Ce deuxième algorithme est nettement plus rapide que son précédent car il ne quitte jamais l'espace de contrainte et ne requiert donc pas le déploiement explicite des termes de Kronecker. Nous avons également étudié les complexités computationelles des deux algorithmes proposés, et nous discutons de l'utilisation des dictionnaires structurés pour l'accélération des algorithmes traditionnels de codage parcimonieux. Un écart considérable a été observé entre les accélérations théoriques et celles obtenues en pratique, mais plusieurs pistes existent pour la réduction de cet écart.

Les résultat expérimentaux sont présentés en chapitre 6. Les algorithmes d'apprentissage de dictionnaire proposés sont évalués sur des applications de débruitage d'images. Deux types distincts d'images sont considérés : images couleurs et hyperspectrales. Des résultats encourageants ont été obtenus dans les deux cas. La contrainte de structure tensorielle imposée au dictionnaire s'est avérée bénéfique en termes de performances de débruitage par rapport à un dictionnaire non structuré, en particulier dans les scénarios de bruit élevé. En plus de servir de régularisation pour éviter le surapprentissage dans de tels scénarios très bruités, la contrainte de structure a également apporté plus de robustesse à l'entraînement sur des jeux de données de taille réduite. En ce qui concerne plus particulièrement les données hyperspectrales, une approche de débruitage plus élaborée a été proposée en intégrant une hypothèse de rang faible sur les images, ce qui a conduit à une performance de débruitage supérieure aux méthodes de l'état de l'art.

Troisième partie

La partie III contient les contributions liées aux régles d'élagage d'atomes. Après un bref rappel en chapitre 7 des approches existantes pour l'accélération de problèmes inverses avec régularisation en norme ℓ_1, au chapitre 8 nous nous intéressons plus particulièrement

aux techniques d'élimination de variables (les règles d'élagage sûres), qui sont à la base des approches proposées dans les chapitres suivants. Ces tests ont pour but de retrouver (et éliminer) des atomes ne correspondant pas au support de la solution finale, *avant même de résoudre le problème*.

Dans le but de combiner les tests d'élagage sûrs et les opérateurs structurés rapides, nous proposons, en chapitre 9, une méthodologie pour définir des tests d'élagage sûrs malgré une connaissance inexacte des atomes (les colonnes de la matrice de dictionnaire). Le formalisme proposé donne lieu à une nouvelle famille de tests d'élagage dits *stables*, qui peuvent gérer des erreurs d'approximation dans les atomes tout en gardant la sûreté du test (c'est à dire, sans aucun risque d'éliminer des atomes associés au support de la solution). La source d'erreur peut être multiple, mais nous nous intéressons au cas où cela découle de l'utilisation d'approximations structurées de la matrice de dictionnaire. Certains des principaux tests d'élagage existants dans la littérature sont étendus à ce nouveau cadre (c'est-à-dire que leur version stable est dérivée), notamment les test sphériques statiques [El Ghaoui *et al.* 2010] et dynamiques [Bonnefoy *et al.* 2015], [Fercoq *et al.* 2015].

Au chapitre 10 les tests d'élagage stables sont employés dans un algorithme rapide pour la résolution du problème lasso. Cet algorithme utilise une série d'approximations rapides de la matrice dictionnaire et peut être aisément combiné avec presque tous les algorithmes de premier ordre existants dans la littérature. Nous dérivons également un critère robuste pour déterminer les itérations auxquelles remplacer le dictionnaire en cours d'utilisation par une autre approximation plus fine.

Les résultats de simulation, au chapitre 11, montrent des réductions significatives des temps d'exécution de l'algorithme proposé en comparaison avec des algorithmes similaires qui n'utilisent pas les dictionnaires approchés et/ou les tests d'élagage. Dans une large gamme de scénarios testés, les résultats expérimentaux démontrent la complémentarité des deux stratégies combinées, justifiant l'effort de les concilier. Les dictionnaires structurés accélèrent la phase initiale de l'optimisation qui correspond à la principale faiblesse des techniques d'élagage, en particulier dans les scénarios faiblement régularisés. Dans ce cas, les tests d'élagage peuvent nécessiter plusieurs itérations avant de commencer à éliminer des atomes.

Conclusion

Ce dernier chapitre récapitule les contributions décrites dans le manuscrit et présente quelques perspectives de recherche futures.

Concernant les dictionnaires structurés discutés en partie II, les perspectives incluent : des études théoriques supplémentaires sur les propriétés de convergence des algorithmes proposés ainsi que sur le choix des différents hyperparamètres. Nous envisageons également l'extension du modèle proposé (HO-SuKro) à d'autres applications tels que l'inpainting et le défloutage d'images, à d'autres types de données comme de l'audio, des données médicales ou sismiques. Dans un sens plus large, nous identifions même des passerelles possibles avec le domaine émergent du traitement de signal sur graphes.

Des nombreuses extensions et variations sont également possibles pour l'élagage stable discuté en partie III. Par exemple, le cadre proposé pourrait être appliqué à d'autres problèmes inverses à contrainte de parcimonie tels que le groupe-lasso ou la régression logistique régularisée. Une autre perspective très intéressante est de géneraliser les tests d'élagage au cas continu [Bredies & Pikkarainen 2013] dans lequel le dictionnaire est constitué d'un nombre infini d'atomes déterminés par des paramètres continus.

Table of Contents

Introduction

How to choose one among a (potentially infinite) set of possible explanations to a given observed situation? Is it reasonable to assume that there is a better explanation?

These questions relate to the fundamental problem of *underdetermination* in philosophy of science: for a given set of observations or data, there is an infinite number of possible models explaining those same data.

A common response to this riddle is based on the *Principle of Parsimony* [Baker 2016], which favors the simplicity of the model. One of its formulations, often referred to as *Occam's razor*, reads: one should not increase, beyond what is necessary, the number of entities required to explain anything.

In mathematics, and more precisely in the context of inverse problems, the term underdetermination designates a very precise situation: there are less constraints (equations) than degrees of freedom (unknowns). It similarly implies the multiplicity of the solution. The two concepts, in philosophy and mathematics, are therefore intimately connected and, as before, a criterion has to be defined to chose one among all solutions. Parsimony appears once again as a legitimate candidate.

Inverse problems, as an archetype of science itself, consist in determining the causal factors explaining some observed data. In this context, parsimony can take multiple forms. For instance, assuming a linear model is a common simplicity prior. It consists in searching to describe the observations as a linear combination of a set of explanatory variables. Another natural form of simplicity assumption is *sparsity*. The sparsity prior consists in describing each observation as a combination of a small number of such explanatory variables.

Even though such supplementary assumptions might resolve the underdetermination issue by ensuring the uniqueness of the solution, they do not imply the existence of a tractable procedure to solve the new constrained problem in practice.

In this thesis, we are confronted with the challenge of providing efficient algorithms to tackle sparsity-constrained linear inverse problems.

Indeed, such problems are ubiquitous in signal processing, statistics and machine learning with different interpretations and terminology. Examples range from audio and image restoration to seismic tomography. But in all cases it comes down to similar optimization problems. Given the ever-growing volume of available data, it is essential to be able to efficiently solve such problems as the number of involved variables grow.

More explicitly, we are faced with the problem of approximating an observation vector $\mathbf{y} \in \mathbb{R}^n$ as a linear combination of the columns of a dictionary matrix $\mathbf{D} \in \mathbb{R}^{n \times m}$. The columns of $\mathbf{D}$ can be seen as the building blocks for reconstructing $\mathbf{y}$ and for this reason are hereby referred to as *atoms*, following the usual terminology in the sparse signal representation community. The dictionary matrix $\mathbf{D}$ is also called measurement matrix [Foucart & Rauhut 2013] or design matrix [Tibshirani 1996] respectively in compressive sensing and statistics communities. We thus seek to determine a coefficient vector $\mathbf{x} \in \mathbb{R}^m$ containing the weights of the linear combination, such that $\mathbf{y} \approx \mathbf{D}\mathbf{x}$. Other denominations for the coefficient vector $\mathbf{x}$ include: representation vector (signal processing) or regression vector (statistics). The problem is underdetermined when $n < m$ and the sparsity constraint implies that only a few atoms of the dictionary are combined in the reconstruction of the input data – which, in turn, implies that the coefficient vector $\mathbf{x}$ is composed mainly of zeros. Therefore, in the context of sparse signal processing, the dictionary matrix defines a representation domain in which the input data can be "explained" with parsimony.

Finding sparse solutions to linear inverse problems

Solving sparsity-constrained linear inverse problems is not an easy task. In general, it cannot be solved analytically and it is necessary to resort to iterative computational approaches.

Actually, it can be shown that, in general, finding the sparsest solution to an underdetermined linear system cannot be solved in polynomial time. It is a classical problem of combinatorial search; one would need to sweep exhaustively through all possible sparse subsets of non-zero elements, generating the corresponding subsystems and checking if it can be solved. The complexity of exhaustive search is exponential in m and intractable in practice.

Despite these theoretical difficulties, several algorithms exist to provide either suboptimal solutions to this problem or exact solutions to similar relaxed problems. These correspond respectively to the two following families of algorithms: which will be further detailed in the remainder of this document:

1. Greedy algorithms [Mallat & Zhang 1993, Pati *et al.* 1993, Needell & Tropp 2009, Donoho *et al.* 2012];

2. Convex relaxation based approaches [Daubechies *et al.* 2004, Beck & Teboulle 2009, Bioucas-Dias & Figueiredo 2007, Wright *et al.* 2009, Chambolle & Pock 2011].

Both families of existing algorithms can become computationally prohibitive in high-dimensional settings, which is why accelerating techniques remain an active research topic.

General objective of this thesis

Accelerate the resolution of sparsity-constrained linear inverse problems by combining two main approaches:

1. The use of structured dictionaries which are efficient to manipulate;
2. Safe screening tests, to quickly eliminate useless dictionary atoms.

Fast structured dictionaries

The main bottleneck of existing iterative algorithms in terms of computational complexity is the cost of the required matrix-vector products involving the dictionary matrix, which dominates the overall iteration cost.

A possible way to address this limitation is to constrain the dictionary matrix to a certain type of structure which would allow for fast matrix-vector products. This type of approach is inspired by the idea of designing fast transforms to implement certain structured linear transforms. A notable example is the Fast Fourier Transform (FFT) algorithm, which was decisive to ensure the feasibility of most early digital signal processing algorithms.

In practice, the dictionary can sometimes be pre-determined by the application in question and might not be structured. For instance, in inverse problems it can be determined by the physical characteristics of the system, like the forward map in electroencephalography (EEG) measurements. A possible strategy, in this case, is to perform initial iterations using a fast structured approximation of the dictionary. In that sense, we explore strategies to efficiently approximate a generic linear operator by a structured one.

In other cases, the dictionary can be learned from the data at hand. This gives rise to the so-called *Dictionary Learning* task. For such scenario, we explore algorithms to explicitly impose the desired structure during the learning process.

Different types of structure, more or less suited to a given application or type of data, can be imagined. In this kind of work, there is always a compromise between the flexibility of the structure and the provided speedup. In this thesis, we primarily explore a tensorial structure which is particularly suited to multidimensional data. The considered structure consists in dictionaries formed as sums of Kronecker (or tensorial) products of sub-dictionaries of smaller size.

Interestingly, structure can also be seen as a form of parsimony, since it corresponds to a constraint that limits the inherent complexity of the objects in question (for instance here, finite-dimensional linear operators represented by matrices). Indeed, such structured matrices are determined by a smaller set of parameters than their unstructured counterpart. There are fewer degrees of freedom or, using the same terminology as in the parsimony principle, less entities required to explain them.

Screening: Elimination of useless atoms

The fact that the coefficient vector $\mathbf{x}$ is composed mostly of zeros implies that most of the atoms (columns of $\mathbf{D}$) are not at all used to reconstruct the input signal. The so-called *screening tests* introduced by [El Ghaoui *et al.* 2010] allow us to identify such unused atoms before even solving for $\mathbf{x}$. These atoms, referred to as *inactive*, can be harmlessly removed from the dictionary to significantly simplify the considered problem. Also, the new restricted dictionary is more compact and readily-applicable in matrix-vector operations.

The screening strategy applies exclusively to the family of convex relaxation techniques. Yet, it is transparent to the underlying solver and can be readily combined with almost any existing algorithm within this particular family.

In order to combine this strategy with the use of structured dictionaries, considering that a fast but approximate structured version of the dictionary is available, we define a new class of screening tests, called *stable screening*, which are robust to approximation errors given that an error bound is provided.

We aim at a scenario in which a coarser (but faster) version of the true dictionary is used in the initial phase of an iterative optimization procedure and finer approximations are progressively adopted as approaching convergence until eventually the original dictionary takes over. This also reinforces the importance of studying techniques to provide structured approximations with controlled precision levels.

Thesis organization and contributions

Part I: The first part of this document reviews the signal models, optimization tools and other mathematical concepts which are used in the following parts.

While in chapter 1 we review the sparse signal model and the associated optimization problems, in chapter 2 we describe the main optimization tools commonly used to tackle such problems. Finally, in chapter 3 we review some useful concepts in tensor algebra.

Part II: This part gathers the contributions associated to fast structured dictionaries. Such contributions are listed below, followed by the chapter's organization.

- A new dictionary learning model is introduced where the dictionary matrix is constrained as a sum of R Kronecker products of K terms (or simply (R, K)-KS). The proposed model is termed Higher Order Sum of Kroneckers (HO-SuKro). Besides exploiting the inherent multidimensional nature of the data, the proposed dictionaries are determined by a smaller number of parameters than a fully unstructured dictionary, which implies several advantages: 1) reduced storage cost; 2) reduced computational complexity on operations involving the dictionary matrix; 3) reduced sample complexity, i.e. fewer training data is required.

- A simple algorithm is proposed to approximate an arbitrary linear operator as (R, K)-KS matrix, which is particularly suited to applications with Kth-order input data. It uses the Canonical Polyadic Decomposition (CPD) to generalize for orders $K > 2$, while the Singular Value Decomposition (SVD) is used in the case $K = 2$.

- A structured Dictionary Learning algorithm is proposed for training HO-SuKro dictionaries from data via a projected gradient procedure. Some color image denoising experiments are performed to illustrate the interest of the proposed techniques.

- A more efficient alternating optimization algorithm is introduced for the HO-SuKro dictionary learning task. The proposed Alternating Least Squares algorithm markedly faster than the previously proposed one since it never leaves the constraint space and, thus, doesn't require the explicit unfolding of the Kronecker terms.

- A Hyperspectral image (HSI) denoising technique is proposed, by combining two assumptions: (i) noiseless hyperspectral images in matrix form are low-rank, and (ii)

image patches are sparse in a proper representation domain defined through a dictionary. We employ both a unstructured and a Kronecker-structured dictionary learned from patches of the noisy HSI and show that the dictionary learning approach is more efficient than state-of-the-art methods with fixed Wavelet transforms.

In chapter 4 we describe the proposed HO-SuKro structure and how to approximate a generic linear operator in this setting. Chapter 5 describes the two proposed structured dictionary learning algorithms and chapter 6 gathers the experimental results in both color image and hyperspectral image denoising tasks.

Part III: The third part gathers the contributions related to screening tests.

- We introduce a formalism for defining safe screening tests which are robust to approximation errors on the dictionary matrix given that an error bound is provided. The source of this error can be manifold, but we will focus on the case where it is a side effect of manipulating structured approximations of the true dictionary matrix. The resulting tests are called *stable screening* tests.
- Some of the main existing screening tests are extended to this new framework (i.e. their stable version is derived). Namely: 1) the seminal static sphere test in [El Ghaoui *et al.* 2010]; 2) the dynamic sphere test in [Bonnefoy *et al.* 2015]; 3) the state-of-the-art GAP sphere test in [Fercoq *et al.* 2015].
- The proposed stable screening is employed in a fast algorithm for ℓ_1-minimization problems, by making use of fast structured approximations of the problem's dictionary matrix. It uses a fast approximation of the dictionary at the initial iterations and then switches to finer approximations until eventually adopting the original dictionary. Simulation results show significant reductions in both computational complexity and execution times for a wide range of tested scenarios. A robust criterion for choosing an appropriate switching moment is proposed, based on both the duality gap saturation and the screening ratio.

After recalling the conventional safe screening technique in chapter 8, we introduce the stable screening framework in chapter 9. The proposed fast algorithm for ℓ_1-regularized problems is described in chapter 10. Finally, experimental results are reported in chapter 11.

Work conducted during this PhD led to the publications listed below. Chapters start with a note on the related publications, when relevant.

Journal paper:

C. F. Dantas and R. Gribonval. *Stable Safe Screening and Structured Dictionaries for Faster ℓ_1 Regularization.* IEEE Transactions on Signal Processing, vol. 67, no. 14, pages 3756–3769, July 2019.

Conference papers:

Cassio F. Dantas and Rémi Gribonval. *Dynamic Screening with Approximate Dictionaries.* In XXVIème colloque GRETSI, Juan-les-Pins, France, Sep 2017.

Cassio F. Dantas and Rémi Gribonval. *Faster and still safe: combining screening techniques and structured dictionaries to accelerate the Lasso.* In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4069–4073, Calgary, AB, Canada, Apr 2018.

Cassio F. Dantas, Jérémy E Cohen and Rémi Gribonval. *Learning fast dictionaries for sparse representations using low-rank tensor decompositions.* In International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA), pages 456–466, Guildford, United Kingdom, Jul 2018.

Cassio F. Dantas, Jérémy E. Cohen and Rémi Gribonval. *Learning Tensor-structured Dictionaries with Application to Hyperspectral Image Denoising.* In 27th European Signal Processing Conference (EUSIPCO), pages 1–5, A Coruña, Spain, Sep 2019.

Cassio F. Dantas, Jérémy E. Cohen and Rémi Gribonval. *Hyperspectral Image Denoising using Dictionary Learning.* In 10th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), pages 1–5, Amsterdam, The Netherlands, Sep 2019.

Part I

Preliminaries

Chapter 1

Sparsity

In this chapter we review some different formulations of a sparsity-inducing linear inverse problem. The two main families of problems are summarized in sections 1.1 and 1.2.

The ℓ_0 operator (hereafter loosely referred to as ℓ_0 norm[1]) which counts the number of non-zero entries of a vector, provides a very natural notion of sparsity. The sparsity assumption can therefore be enforced via a constraint or penalization –i.e. regularization– on the ℓ_0 norm of a solution. This operator, however, is neither convex nor continuous and the resulting problems (section 1.1) are NP-hard. Relaxed problems (section 1.2) can be obtained by replacing the ℓ_0 norm by the ℓ_1 norm, which is given by the sum of the absolute values of a vector. This norm has the advantage of being convex (and continuous), and so are the corresponding problems. For completeness, in section 1.3 we mention other ways to promote sparse solutions which are not further explored in this thesis. In section 1.4 we provide some useful existing theoretical guarantees for sparse inverse problems.

The dictionary matrix in the mentioned problems is supposed to be given. Yet, in some cases, it might be interesting to learn it from the data. This gives rise to the Dictionary Learning task, briefly reviewed in section 1.5.

1.1 ℓ_0 formulation

The problem of finding sparse solutions to an underdetermined linear system of equations $\mathbf{Dx} = \mathbf{y}$ (with $\mathbf{D} \in \mathbb{R}^{n \times m}$ a full-rank matrix and $n < m$) is commonly referred to as *Sparse coding*. Mathematically, it can be described as the following minimization problem:

$$(P_0): \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \; s.t. \; \mathbf{y} = \mathbf{Dx}, \tag{1.1}$$

where $\mathbf{x} \in \mathbb{R}^m$ is a sparse representation of a signal $\mathbf{y} \in \mathbb{R}^n$ in the dictionary $\mathbf{D} \in \mathbb{R}^{n \times m}$. The columns of $\mathbf{D}$, which are referred to as atoms, are a full and overcomplete set.

1. Strictly speaking, it is not a norm as it is not homogeneous: for $t \neq 0, \|t\mathbf{u}\|_0 = \|\mathbf{u}\|_0 \neq t\|\mathbf{u}\|_0$.

In practical applications, we allow some deviation ϵ from the model rather than seeking a perfect reconstruction:

$$(P_0^\epsilon): \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad s.t. \quad \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 \leq \epsilon, \tag{1.2}$$

An alternative form, in which the sparsity k is known (or imposed), is:

$$(P_0^k): \quad \min_{\mathbf{x}} \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 \; s.t. \; \|\mathbf{x}\|_0 \leq k, \tag{1.3}$$

Although there is a clear inverse relation between ϵ and k – the higher the ϵ, the smaller the corresponding k – such relation cannot be trivially characterized.

The $(P0)$ problem, as well as its approximate (noisy) versions (P_0^ϵ) and (P_0^k), are non-convex and hard to solve, due to the discrete and discontinuous nature of the ℓ_0 norm. Actually, these problems can be shown to be NP-hard [Natarajan 1995] as their solution requires an exhaustive search through all possible sparse solution supports, which is exponential in m. Indeed, for a given sparsity level (say, s non-zero elements), there are $\binom{m}{s}$ possible supports.

Given the infeasibility of an optimal solution, several greedy approximation techniques have been proposed to tackle these problems, as will be detailed in section 2.1.

1.2 ℓ_1 formulation

Another class of algorithms uses a convex relaxation of the ℓ_0 norm to the ℓ_1 norm: $\|\mathbf{x}\|_1 = \sum_i |x_i|$. It can be formulated in a constrained form (exact and inexact respectively)

$$(P_1): \quad \min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad s.t. \quad \mathbf{y} = \mathbf{D}\mathbf{x} \tag{1.4}$$

$$(P_1^\epsilon): \quad \min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad s.t. \quad \|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 \leq \epsilon, \tag{1.5}$$

but we will be more interested in its unconstrained formulation:

$$(P_1^\lambda): \quad \min_{\mathbf{x}} \; \tfrac{1}{2}\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1 \tag{1.6}$$

which is also known as lasso [Tibshirani 1996] or basis pursuit denoising (BPDN) [Chen *et al.* 1998]. The resulting problems are convex which means that they can be more easily tackled by standard optimization tools – even if the ℓ_1 norm is not differentiable. For

instance, (P_1) can be cast as a linear programming problem and solved using interior-point or simplex methods [Chen *et al.* 1998]. Or, in an even simpler approach, (P_1^λ) can be tackled by proximal gradient methods [Beck & Teboulle 2009]. These methods will be more thoroughly reviewed in section 2.2.

1.3 Other sparsity-promoting functions

Additional criteria are needed to narrow the choice of one among the infinitely many solutions of the underdetermined linear systems $\mathbf{Dx} = \mathbf{y}$. More generally, these criteria can be encoded by a function $g(\mathbf{x})$ penalizing the solutions which do not have a certain desirable characteristic (for instance, the sparsity). This leads to the following problem

$$(P_g): \quad \min_{\mathbf{x}} g(\mathbf{x}) \quad s.t. \quad \mathbf{y} = \mathbf{Dx} \tag{1.7}$$

Sparsity-inducing functions include the mentioned ℓ_0 and ℓ_1, but are not restricted to those. Actually, any ℓ_p-norm with $0 < p < 1$ can induce sparsity[2]:

$$\|\mathbf{x}\|_p = \left(\sum_i |x_i|^p \right)^{1/p}$$

In fact, any function $g(\mathbf{x}) = \sum_i \rho(x_i)$ with $\rho(x)$ symmetric, monotonically non-decreasing, and with a monotonic non-increasing derivative for $x \geq 0$ will serve the same purpose of enforcing sparsity [Elad 2010] [Gribonval & Nielsen 2007]. Some notable examples are:

$$\rho(x) = 1 - \exp(|x|), \qquad \rho(x) = \log(1 + |x|), \qquad \rho(x) = |x|/(1 + |x|).$$

A geometric intuition To get some more intuition, consider the problem (P_g) with $g(\mathbf{x}) = \|\mathbf{x}\|_p^p$, The linear set of equations forming the constraint defines a feasible set of solutions that are on an affine subspace. Any feasible solution therefore lies in this set, which is a hyperplane of dimension $\mathbb{R}^{m-n}$ embedded in $\mathbb{R}^m$. Among all solutions from this hyperplane, we seek the one with minimum ℓ_p norm. Geometrically, this corresponds to "blowing" a small ℓ_p ball (set of points in $\mathbb{R}^m$ in which the ℓ_p norm equals a constant value) until it first touches the feasible set. The intersecting point is the sought solution.

This procedure is illustrated in Figure 1.1 for a 3-dimensional space ($m = 3$) comparing

2. Once again, for $p < 1$ the term norm is not accurate, since the triangle inequality is not satisfied.

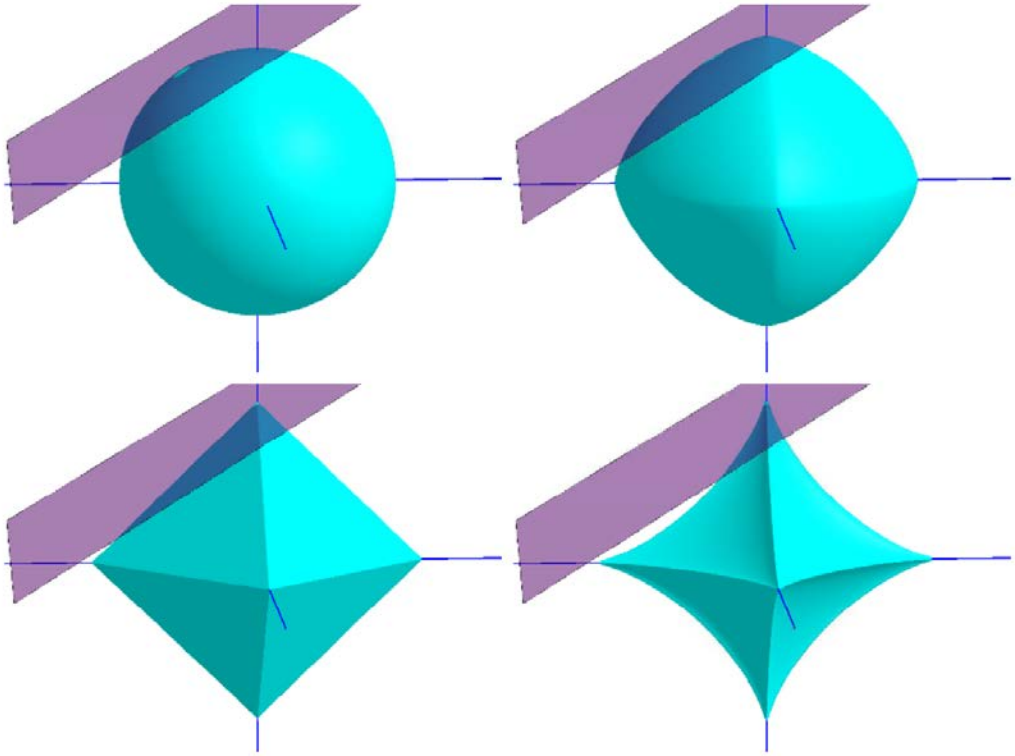


Figure 1.1 – The solution of (P_g) with $g(\mathbf{x}) = \|\mathbf{x}\|_p^p$ is given by the intersection between the ℓ_p-ball and the plane corresponding to $\mathbf{Dx} = \mathbf{y}$. This intersection is demonstrated for $p = 2$ (top left), $p = 1.5$ (top right), $p = 1$ (bottom left), and $p = 0.7$ (bottom right). When $p \leq 1$, the intersection takes place along the axes, leading to a sparse solution. Source: [Elad 2010]

several values of p. The feasibility set (solutions of the underdetermined linear system $\mathbf{y} = \mathbf{Dx}$) is given by a tilted plane. Note that for $p \leq 1$ the intersection point tend to occur in the ball-corners, which take place along the axes. In such points, 2 of the 3 coordinates are zeros, which is the tendency to sparsity we referred to. On the other hand, norms with $p > 1$ tend to give intersection points with three non-zero coordinates.

1.4 Some theoretical guarantees

The problem of finding sparse solutions to underdetermined systems of linear equations has been largely studied in the literature. The theoretical guarantees achieved so far include uniqueness conditions for the sparsest solution, conditions under which (P_0) and (P_1) have the same solution, as well as results on the impact of noise and the behavior of approximate solutions.

In subsections 1.4.1 and 1.4.2 we present some uniqueness and recoverability results on the exact recovery case ($\mathbf{y} = \mathbf{D}\mathbf{x}$). In subsection 1.4.3 we discuss the uniqueness on the (P_1^λ) problem (lasso or basis pursuit), which will be of particular interest in part III of this thesis.

1.4.1 Uniqueness via the Spark

A crucial concept for the study of uniqueness of sparse solutions is the spark of a matrix, introduced in [Donoho & Elad 2003]. Interestingly, an equivalent property, termed Kruskal rank, previously appeared in the tensor literature to study the uniqueness of tensor decompositions [Kruskal 1977].

Definition 1 (Spark [Donoho & Elad 2003])**.** *Given a matrix* $\mathbf{D}$ *we define* $\sigma = \text{spark}(\mathbf{D})$ *as the smallest possible number such that that there exists a subgroup of* σ *columns from* $\mathbf{D}$ *that are linearly dependent.*

It is important to emphasize the difference to the concept of rank of a matrix. The rank is defined as the *maximal* number of columns from $\mathbf{D}$ that are linearly *independent*. Actually, the spark of a matrix is far more difficult to obtain than its rank, as its computation (the spark) in general requires a combinatorial search over all possible subsets of columns from $\mathbf{D}$.

Spark leads to a simple criterion for uniqueness of sparse solutions.

Theorem 1 (Uniqueness - Spark [Gorodnitsky & Rao 1997])**.** *If a system of linear equations* $\mathbf{D}\mathbf{x} = \mathbf{y}$ *has a solution* $\mathbf{x}$ *such that* $\|\mathbf{x}\|_0 < \text{spark}(\mathbf{D})/2$*, this solution is necessarily the sparsest possible.*

Proof. Consider two different solutions, i.e. $\exists\ \mathbf{x}_1 \neq \mathbf{x}_2 \ \mid \ \mathbf{y} = \mathbf{D}\mathbf{x}_1 = \mathbf{D}\mathbf{x}_2$. Thus, the difference $\mathbf{x}_1 - \mathbf{x}_2$ must be in the null space of the columns of $\mathbf{D}$. Hence some group of columns from $\mathbf{D}$ must be linearly dependent.
From the definition of the spark, we have that every vector $\mathbf{z}$ in the null space has $\|\mathbf{z}\|_0 \geq \text{spark}(\mathbf{D})$ (i.e. one needs to combine at least $\text{spark}(\mathbf{D})$ columns of $\mathbf{D}$ to create linear dependency). This implies that

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_0 \geq \text{spark}(\mathbf{D}).$$

By applying the triangle inequality on the ℓ_0 norm, we have

$$\|\mathbf{x}_1\|_0 + \|\mathbf{x}_2\|_0 \geq \text{spark}(\mathbf{D}),$$

which means that any two solutions $(\mathbf{x}_1, \mathbf{x}_2)$ must have at least $\text{spark}(\mathbf{D})$ nonzero entries combined. If there exists a solution satisfying $\|\mathbf{x}_1\|_0 < \text{spark}(\mathbf{D})/2$, then any other solution $\mathbf{x}_2$ must satisfy $\|\mathbf{x}_1\|_0 > \text{spark}(\mathbf{D})/2$, implying that $\mathbf{x}_1$ is the sparsest solution. □

Clearly, the value of spark is very informative, since we can check for global optimality of a given solution by simply comparing its sparsity to the spark. Large values of spark are desirable as they make the condition in theorem 1 easier to satisfy. Some bounds can be drawn for the spark. Clearly, if there are no zero columns, then $\text{spark}(\mathbf{D}) \geq 2$. Also, from the definition of rank, any set of $\text{rank}(\mathbf{D}) + 1$ columns of $\mathbf{D}$ are necessarily linear dependent, which gives $\text{spark}(\mathbf{D}) \leq \text{rank}(\mathbf{D}) + 1 \leq n + 1$. Moreover, if the entries of $\mathbf{D}$ are random independent and identically distributed (i.i.d.) from any continuous distribution, then with probability 1 the maximal spark is achieved: $\text{spark}(\mathbf{D}) = n + 1$ [Bruckstein *et al.* 2009]. In this case, uniqueness is ensured for every solution with $\lfloor \frac{n}{2} \rfloor$ or fewer nonzero entries.

1.4.2 Uniqueness via the mutual coherence

The spark is at least as difficult to evaluate as solving (P_0). Thus, simpler ways to guarantee uniqueness are of interest. A very simple way uses the concept of mutual coherence

Definition 2 ([Mallat & Zhang 1993])**.** *The mutual coherence of a matrix* $\mathbf{D}$ *is the largest absolute normalized inner product between different columns from* $\mathbf{D}$*. Denoting* $\mathbf{d}_k$ *the* k*-th column of* $\mathbf{D}$*, the mutual coherence is given by*

$$\mu(\mathbf{D}) = \max_{1 \leq i \neq j \leq m} \frac{|\mathbf{d}_i^T \mathbf{d}_j|}{\|\mathbf{d}_i\|_2 \|\mathbf{d}_j\|_2}$$

Mutual coherence is much easier to compute than the spark, as it requires only an order of m^2 inner products for a matrix with m columns. The fact that we are to able lower bound the spark with the mutual coherence is, therefore, very useful.

Lemma 1 ([Donoho & Elad 2003])**.** *For any matrix* $\mathbf{D} \in \mathbb{R}^{n \times m}$*,*

$$\text{spark}(\mathbf{D}) \geq 1 + 1/\mu(\mathbf{D}).$$

Proof. See, for instance, [Bruckstein *et al.* 2009]. □

Lemma 1 allow us to derive a looser but more readily-applicable version of Theorem 1 using the mutual coherence.

Theorem 2 (Uniqueness - Mutual coherence [Donoho & Elad 2003])**.** *If a system of linear equations* $\mathbf{Dx} = \mathbf{y}$ *has a solution* $\mathbf{x}$ *such that* $\|\mathbf{x}\|_0 < \frac{1}{2}(1 + 1/\mu(\mathbf{D}))$*, this solution is necessarily the sparsest possible.*

So far, we have established sufficient conditions for an ℓ_0 minimizer to be unique. In principle, there is no reason for the same solution $\mathbf{x}$ to be an ℓ_1 minimizer as well – i.e. a solution to (P_1). It turns out [Gribonval & Nielsen 2003] that the condition in theorem 1 ($\|\mathbf{x}\|_0 < \frac{1}{2}(1+1/\mu(\mathbf{D}))$ is also sufficient to imply that $\mathbf{x}$ is the ℓ_1 minimizer provided that $\mathbf{D}$ has unit-norm columns.

In summary, for matrices $\mathbf{D}$ with incoherent columns, whenever (P_0) has a sufficiently sparse solution, that solution is unique and is equal to the solution of (P_1). Lower coherence values are desirable as they make the uniqueness condition easier to satisfy. It has been shown [Welch 1974] that for a general $(n \times m)$ matrix $\mathbf{D}$, the mutual coherence is lower bounded by

$$\mu(\mathbf{D}) \geq \sqrt{\frac{m-n}{n(m-1)}} \,. \tag{1.8}$$

It implies in particular for $m > n$, by plugging $m = \alpha n$ with $\alpha > 1$ in (1.8), that $\mu(\mathbf{D}) \geq 1/\sqrt{n}$. This gives a sparsity bound in theorem 2 of the order of $\sqrt{n}/2$, which is smaller than the one obtained with the spark: up to $n/2$.

1.4.3 Uniqueness in lasso

The ℓ_1-regularized unconstrained formulation, i.e. the (P_1^λ) problem or simply lasso, is convex but not strictly so when $m > n$; and, therefore, the uniqueness of the solution is not guaranteed. As a convex problem, every local minimum is also a global minimum but not necessarily unique. Multiple solutions can achieve the (same) minimum cost function. A detailed study on the uniqueness of the lasso solution is provided by [Tibshirani 2013].

Lemma 2 ([Tibshirani 2013])**.** *For any* $\mathbf{y}$*,* $\mathbf{D}$*, and* $\lambda \geq 0$*, the lasso problem* (P_1^λ) *has the following properties:*

(i) There is either a unique solution or an (uncountably) infinite number of solutions.

(ii) Every lasso solution $\mathbf{x}^\star$ *gives the same fitted value* $\mathbf{D}\mathbf{x}$*.*

(iii) If $\lambda > 0$*, then every lasso solution* $\mathbf{x}^\star$ *has the same* ℓ_1 *norm,* $\|\hat{\mathbf{x}}\|_1$*.*

Fortunately, uniqueness can be guaranteed under very mild conditions.

Lemma 3 (Lasso uniquennes - sufficient condition [Tibshirani 2013])**.** *If the entries of* $\mathbf{D} \in \mathbb{R}^{n\times m}$ *are drawn from a continuous probability distribution on* $\mathbb{R}^{n\times m}$*, then for any* $\mathbf{y}$ *and* $\lambda > 0$*, the lasso solution is unique with probability one.*

According to this result, we do not have to worry about uniqueness when the atoms come from a continuous distribution, regardless of the sizes of n and m. Actually, this result can be extended to other data fidelity terms (not only the squared error): it holds for ℓ_1 penalized minimization with any differentiable, strictly convex loss function.

In the case of discretely distributed atoms, even though the solution *might* not be unique, we can still guarantee that any two lasso solutions must have the same signs over their common support [Tibshirani 2013]. Concerning the support of different solutions, it is useful to define the following set:

Definition 3 (Equicorrelation set)**.** *For a lasso problem with* $\mathbf{D} \in \mathbb{R}^{n\times m}$*,* $\mathbf{y} \in \mathbb{R}^n$ *and* $\lambda \geq 0$*, and any solution* $\mathbf{x}^\star$*, we define the equicorrelation set* $\mathcal{E}$ *by*

$$\mathcal{E} = \left\{ i \in \{1, \ldots, m\} \ : \ |\mathbf{d}_i^T(\mathbf{y} - \mathbf{D}\mathbf{x}^\star)| = \lambda \right\}.$$

Lemma 4 (Support of lasso solutions)**.** *The support of a lasso solution is always a subset of the equicorrelation set.*

Actually, a more explicit sufficient condition for uniqueness of the lasso solution is that $\text{null}(\mathbf{D}_\mathcal{E}) = \{0\}$ (or, equivalently, $\text{rank}(\mathbf{D}_\mathcal{E}) = |\mathcal{E}|$) where $\mathbf{D}_\mathcal{E}$ stands for the dictionary matrix restricted to the columns belonging to the equicorrelation set and $|\mathcal{E}|$ is the cardinality of this set.

1.4.4 Exact recovery condition

The definition of an Exact Recovery Condition (ERC) in [Tropp 2004] brings an additional and interesting insight towards the performance limits of the OMP (a classical greedy approach for ℓ_0 minimization –to be detailed in section 2.1.1– which iteratively increases

the solution support by picking the "most promising" atom at each iteration) and the basis pursuit (or the lasso, i.e. the ℓ_1-regularized least-squares formulation).

Definition 4 (ERC [Tropp 2004])**.** *For a given support $\mathcal{S}$ and for a matrix $\mathbf{D}$, the Exact Recovery Condition (ERC) is given by*

$$\text{ERC}(\mathbf{D}, \mathcal{S}) : \max_{i \notin \mathcal{S}} \|\mathbf{D}_{\mathcal{S}}^{\dagger} \mathbf{d}_i\|_1 < 1.$$

This condition considers linear systems of equations of the form $\mathbf{D}_{\mathcal{S}}\mathbf{x} = \mathbf{d}_i$, with $\mathbf{d}_i$ being a column from $\mathbf{D}$ that is outside the support $\mathcal{S}$. The ERC states that the minimum ℓ_2-energy solution[3] (i.e. $\mathbf{D}_{\mathcal{S}}^{\dagger}\mathbf{d}_i$) to all these systems should have an ℓ_1-norm length smaller than 1. The ERC provides a guarantee on the success of pursuit techniques:

Theorem 3. $\text{ERC}(\mathbf{D}, \mathcal{S})$ *is a sufficient condition for both OMP and BP to recover the sparsest solution of a system $\mathbf{Dx} = \mathbf{y}$, with $\mathcal{S}$ being the solution support (i.e. $\mathbf{y} = \mathbf{D}_{\mathcal{S}}\mathbf{x}_{\mathcal{S}}$).*

It is possible to show that the ERC is at least as general as the mutual coherence result in Theorem 2.

Theorem 4. *For a matrix $\mathbf{D}$ with mutual-coherence $\mu(\mathbf{D})$, then for all supports $\mathcal{S}$ with cardinality equal or smaller than $\frac{1}{2}(1 + 1/\mu(\mathbf{D}))$, the ERC is satisfied.*

1.5 Dictionary Learning

In the context of sparse representations, a problem that has attracted great attention is the choice of a dictionary to efficiently sparsify a certain class of signals of interest.

The first dictionaries to be used were existing transforms – such as the Fourier, wavelet, short-time Fourier transform (STFT), and Gabor transforms, see e.g., [Mallat & Zhang 1993], [Chen *et al.* 1998]. This type of dictionary is characterized by an analytic formulation, and an important advantage of this approach is that the resulting dictionary usually features a fast implementation (e.g. the Fast Fourier Transform) which does not involve explicit multiplication by the dictionary matrix. On the other hand, the dictionary can only be as successful as its underlying model, and indeed, these models tend to be over-simplistic compared to the complexity of natural phenomena. For example, not all signals can be sparsely approximated as a sum of sinusoids, particularly signals containing discontinuities. In such cases, a Fourier dictionary would not lead to satisfactory results.

3. Or the least-squares estimation, when $\mathbf{d}_i \notin \text{span}(\mathbf{D}_{\mathcal{S}})$, which is typically the case.

Data-driven techniques, on the other hand, can be used to learn dictionaries that are better adapted to specific instances of the data, replacing the use of generic models. As with machine learning techniques [Bishop 2006], the idea is that the structure of complex natural phenomena can be more accurately extracted from the data itself than by using a prior mathematical description.

This new paradigm gives rise to the dictionary learning task. The idea here is to learn a suitable dictionary from some training data. In this setting, rather than a single input signal $\mathbf{y}$, we now consider a set of N input signals arranged as the columns of a *training data matrix*: $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \dots \mathbf{y}_N] \in \mathbb{R}^{n \times N}$.

Given $\mathbf{Y}$, we seek to simultaneously learn: 1) a representation domain that efficiently sparsifies the input data (in the dictionary matrix $\mathbf{D} \in \mathbb{R}^{n \times m}$); 2) the corresponding sparse representations of each input signal (as the columns of a *representation matrix* $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_N] \in \mathbb{R}^{m \times N}$).

This task can be formalized, among other possible formulations, as the following optimization problem:

$$\min_{\mathbf{D} \in \mathcal{S}_\mathbf{D}, \mathbf{X}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \sum_{i=1}^{N} g(\mathbf{x}_i). \tag{1.9}$$

The goal is to minimize the representation error over the training data – measured by the first term, the *data fidelity* term – while enforcing the sparsity of the columns of $\mathbf{X}$ via a generic sparsity-enforcing penalization $g : \mathbb{R}^m \mapsto \mathbb{R}$ – for instance, the ℓ_0 or ℓ_1 norms of the columns $\mathbf{x}_i$ (see chapter 1). Similarly to the sparsity-inducing problems introduced in chapter 1, analogous constrained formulations could also be considered. Finally, the dictionary columns are usually restricted to unit Euclidean norm in order to avoid a scale ambiguity problem.[4] This means that the constraint set $\mathcal{S}_\mathbf{D}$ is, most often, simply the set of matrices with unit Euclidean norm columns. An exception is made for the so-called structured dictionary learning methods (overviewed in section 1.5.2), in which the dictionary is further constrained to a certain structure of interest.

As formulated in (1.9), the dictionary learning problem can be regarded as a matrix factorization problem. The input data matrix $\mathbf{Y}$ is factorized into a product between a dictionary $\mathbf{D}$ and a sparse coefficient matrix $\mathbf{X}$:

$$\mathbf{Y} \approx \mathbf{D}\mathbf{X} \ , \tag{1.10}$$

4. Otherwise, one could arbitrarily reduce $g(\mathbf{x}_i)$ by transferring energy from $\mathbf{x}_i$ to the atom $\mathbf{d}_i$.

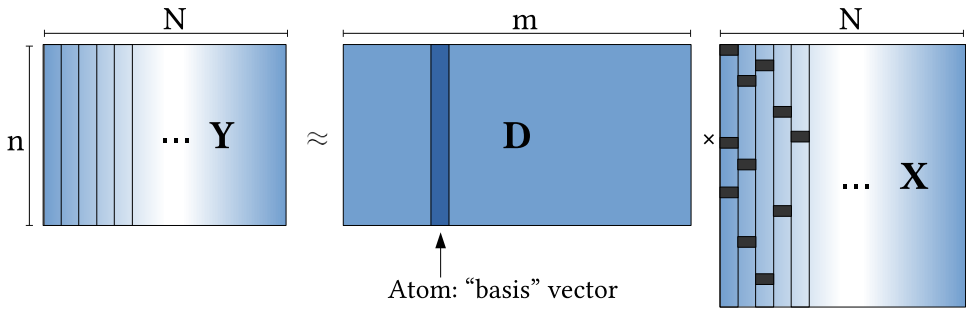


Figure 1.2 – Dictionary learning problem: matrix formulation.

where each data sample in $\mathbf{Y}$ is approximated as a linear combination of only a few columns of $\mathbf{D}$, referred to as *atoms*. This matrix formulation is illustrated in Figure 1.2.

The considered problem is jointly non-convex in $\mathbf{D}$ and $\mathbf{X}$ which explains the predominance of sub-optimal approaches to tackle it in the literature (see [Rubinstein *et al.* 2010a] or [Tosic & Frossard 2011] for a survey). Since there are two variables to simultaneously estimate from the data, most of the existing training methods adopt an alternating optimization strategy with two steps:

Sparse coding step: For a fixed dictionary ($\mathbf{D}$), find the best sparse approximation ($\mathbf{X}$) for the input data.

$$\min_{\mathbf{X}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \sum_i g(\mathbf{x}_i) \quad = \quad \min_{\{\mathbf{x}_i\}_{i=1}^N} \sum_i \|\mathbf{y}_i - \mathbf{D}\mathbf{x}_i\|_2^2 + g(\mathbf{x}_i). \tag{1.11}$$

This is usually handled as a set of N independent sparse representation problems as the ones introduced in sections 1.1 and 1.2, one for each input data $\mathbf{y}_i$. In practice, any of the formulations presented in sections 1.1 and 1.2 can be used and, similarly, any of the corresponding optimization algorithms which will be described in sections 2.1 and 2.2 can be applied.

Dictionary update step: For a fixed sparse representation matrix ($\mathbf{X}$), optimize the dictionary atoms for better data approximation:

$$\min_{\mathbf{D} \in \mathcal{S}_\mathbf{D}} \|\mathbf{D}\mathbf{X} - \mathbf{Y}\|_F^2. \tag{1.12}$$

Each of the several existing dictionary learning algorithms proposes a different approach to solve this problem, especially if the constraint $\mathcal{S}_\mathbf{D}$ is nontrivial. Some of these techniques are briefly described in sections 1.5.1 and 1.5.2.

1.5.1 Unconstrained Dictionary Learning

The following dictionary learning algorithms employ the mentioned alternating minimization strategy. The sparse coding step can be performed by any standard technique and is not detailed here. The dictionary update step is usually performed in an unrestricted manner while unit-norm constraint in $\mathbf{D}$ is handled as a post-processing step.

Method of Optimal Directions (MOD)

The Method of Optimal Directions (MOD) was introduced by [Engan *et al.* 1999], and was one of the first proposed methods for dictionary learning. The dictionary update step uses the well-known analytic solution for the linear least squares (LS) problem

$$\mathbf{D} = \mathbf{Y}\mathbf{X}^{\dagger}, \tag{1.13}$$

where † denotes the Moore-Penrose pseudo-inverse[5] ($\mathbf{X}^{\dagger} = \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$).

Despite its conceptual simplicity, the method suffers from the relatively high complexity of the required matrix inversion. Several subsequent works have thus focused on reducing this complexity.

K-SVD

The K-SVD algorithm, introduced by [Aharon *et al.* 2006], is probably the most popular unconstrained dictionary learning algorithm. The main contribution of the K-SVD is that the dictionary update, rather than using a matrix inversion, is performed atom-by-atom in a simple and efficient process. Further acceleration is provided by updating both the current atom and its associated sparse coefficients simultaneously. The K-SVD algorithm takes its name from the Singular Value Decomposition (SVD) process that forms the core of the atom update step.

The dictionary optimization step updates one column (atom) at a time, fixing all columns except one, say $\mathbf{d}_k$, and updating its value along with the corresponding coefficients in the matrix $\mathbf{X}$ to optimally reduce the mean squared error. This is markedly different from all previous methods, which freeze $\mathbf{X}$ while finding a better $\mathbf{D}$.

Fixing all columns of $\mathbf{D}$ except $\mathbf{d}_k$, and denoting $\mathbf{x}_T^k$ the k-th row of $\mathbf{X}$, which contains the coefficients related to the atom $\mathbf{d}_k$, the penalty term in the objective function (1.12)

5. Note that if an atom is never used (i.e. a row of $\mathbf{X}$ is all zeros), the pseudo-inverse $\mathbf{X}^{\dagger}$ does not exist. In this case, any value can be assigned to the corresponding atom with no effect in the cost function.

can be rewritten as:

$$\|\mathbf{Y} - \mathbf{DX}\|_F^2 = \left\|\mathbf{Y} - \sum_{i=1}^{m} \mathbf{d}_i \mathbf{x}_T^i\right\|_F^2 = \left\|\left(\mathbf{Y} - \sum_{i \neq k} \mathbf{d}_i \mathbf{x}_T^i\right) - \mathbf{d}_k \mathbf{x}_T^k\right\|_F^2 = \|\mathbf{E}_k - \mathbf{d}_k \mathbf{x}_T^k\|_F^2 \quad (1.14)$$

where the product $\mathbf{DX}$ has been decomposed as the sum of m rank-1 matrices. All terms but the one associated with the k-th atom are grouped and $\mathbf{E}_k$ stands for the error for each of the N samples when the k-th atom is removed.

The minimization of (1.14) for $\mathbf{d}_k$ and $\mathbf{x}_T^k$ corresponds to a rank-1 approximation of $\mathbf{E}_k$, which is optimally solved by the SVD truncated to its first component[6]. To avoid introduction of new non-zeros in $\mathbf{X}$, the update process is performed using only the examples whose current representations use the atom $\mathbf{d}_k$.

1.5.2 Structured Dictionaries

The learning techniques mentioned so far lead to non-structured dictionaries which are relatively costly to apply; therefore these methods remain limited to signals of relatively small size. The most recent contributions to the field, thus, focus on parametric models in the training process, which produce structured dictionaries [Dumitrescu & Irofti 2018]. The idea is to preserve the best of both scenarios: the computational efficiency of the analytic dictionaries and the flexibility of the learned ones. Naturally, any kind of structure imposition corresponds to a restriction on the search space, leading to less adapted dictionaries in general. The challenge is ultimately to better handle this compromise, providing a higher computational reduction by paying as little as possible in terms of performance. As a byproduct, structured dictionaries also require fewer training symbols, since they have fewer parameters to be learned [Gribonval *et al.* 2015b, Shakeri *et al.* 2019].

Early proposals Countless types of dictionary structures may be imagined. Learning a dictionary as a union of orthonormal bases was proposed in [Lesage *et al.* 2005]. This type of structure allows efficient sparse-coding via block coordinate relaxation (BCR) [Sardy *et al.* 2000] . Another proposition is the *signature dictionary* [Aharon & Elad 2008], in which the dictionary is described by a compact image called signature. Each of its $\sqrt{n} \times \sqrt{n}$ sub-blocks constitute an atom of the dictionary. The reduced number of parameters (only one per atom) also makes this model more restrictive than other structures.

6. This result follows from the Eckart-Young theorem [Eckart & Young 1936].

Sparsity-related More recently, in [Magoarou & Gribonval 2015] the authors impose the dictionary to be the product of several sparse matrices. The total complexity in this case is determined by the number of non-zero values on the factor matrices. Still on the sparsity line, [Rubinstein *et al.* 2010b] proposes to learn a dictionary in the form $\mathbf{D} = \mathbf{\Phi}\mathbf{A}$, where $\mathbf{\Phi}$ is a pre-specified *base dictionary* that has a fast implementation (e.g. DCT) and $\mathbf{A}$ is a column sparse matrix. In other words, the atoms in this dictionary are sparse linear combinations of atoms from the *base dictionary*. In [Sulam *et al.* 2016] this idea is taken further, replacing the fixed base dictionary by an adaptable multi-scale one (*cropped* Wavelets are used). This brings more flexibility to the model, while keeping its scalability.

Shift invariance Shift invariant dictionaries [Mailhé *et al.* 2008, Thiagarajan *et al.* 2008, Jost *et al.* 2006, Barthelemy *et al.* 2012] bring the advantage of being insensitive to the way a long signal is cut into smaller patches for processing. They also often have fast representation algorithms based on the FFT. In [Pope *et al.* 2013, Rusu *et al.* 2014] the shift invariance is promoted by explicitly imposing a circulant structure on the dictionary matrix. In [Chabiron *et al.* 2015], the dictionary atoms are the composition of several circular convolutions using sparse kernels. A closely related idea has gained great popularity recently: the convolutional sparse representation model [Bristow *et al.* 2013, Papyan *et al.* 2017b, Garcia-Cardona & Wohlberg 2018] where the dictionary is a concatenation of circulant matrices. Its popularity is partly due to the strong connections with the prominent topic of convolutional neural networks [Papyan *et al.* 2017a, Papyan *et al.* 2018].

Low rank and tensor structure A straightforward approach for complexity reduction is introducing a low-rank restriction on the dictionary. This approach has notably been applied for face recognition [Ma *et al.* 2012, Li *et al.* 2013, Zhang *et al.* 2013, Zheng *et al.* 2016]. A seemingly unrelated approach, proposed in [Hawe *et al.* 2013, Roemer *et al.* 2014], is to learn separable dictionaries which are formed as the Kronecker product of two smaller terms, i.e. $\mathbf{D} = \mathbf{D}_1 \boxtimes \mathbf{D}_2$. This particular structure arises naturally when treating two-dimensional data, such as images. In this case, the direct and transpose operators can be obtained directly from the sub-dictionaries, which have much smaller dimensions, thus leading to considerable complexity savings. As will be shown in chapter 4, the Kronecker-product constraint can be formulated as a rank-one constraint, so that the two approaches (separability and low-rankness) are actually related. [Dantas *et al.* 2017] also leveraged this relationship to extend the proposal in [Hawe *et al.* 2013] to a sum of

Kronecker products, where the number of summing terms provides a trade-off between complexity and flexibility.

The separable structure was also proposed for 3-dimensional data, i.e. the Kronecker product of three terms $\mathbf{D} = \mathbf{D}_1 \boxtimes \mathbf{D}_2 \boxtimes \mathbf{D}_3$ [Zubair & Wang 2013, Peng *et al.* 2014], and even for an arbitrary tensor order [Caiafa & Cichocki 2013, Ghassemi *et al.* 2017] based on the Tucker decomposition, a model coined as Tucker Dictionary Learning. In this thesis (part II), we generalize this model to a sum of Kronecker products of any order, aiming at multidimensional data of arbitrary order.

Chapter 2

Optimization tools

In this chapter, we review some useful optimization tools for sparse signal reconstruction.

Two main families of algorithms are explored in sections 2.1 and 2.2, always focusing on the algorithms which will intervene in the remainder of this document. In section 2.1, we review algorithms tackling the ℓ_0 problems with a greedy strategy, and, in section 2.2, convex optimization algorithms tackling the relaxed ℓ_1 problems. Other related approaches are briefly overviewed in each section, but with no pretension to make an exhaustive list of the existing methods.

In section 2.3, the Lagrangian duality framework is reviewed and the lasso dual problem is derived along with its optimality conditions. These concepts will be particularly useful in part III of this thesis.

2.1 Greedy pursuit

The greedy approach is a long-established algorithmic paradigm which consists in making locally optimal choices at each stage. In some sense, it renounces to treat the problem as a whole (usually because it is intractable) to approach it as smaller short-term subproblems. Naturally, the greedy strategy does not usually lead to a globally optimal solution, but can make hard problems approachable in practice.

Greedy algorithms have been proposed for addressing the NP-hard problem of finding the sparsest solution of an underdetermined system of equations – not only the exact reconstruction case (P_0) but also its inexact counterparts (P_0^ϵ) and (P_0^k). The general procedure consists in iteratively expanding the support set by adding one (or several) most promising atom in each step. The column that maximally reduces approximation error given the current support is chosen at each iteration. We focus on the most prominent greedy approach for ℓ_0 minimization, orthogonal matching pursuit (OMP) and some of its different possible implementations in section 2.1.1. In section 2.1.2, other greedy pursuit algorithms are briefly discussed.

2.1.1 (Orthogonal) Matching Pursuit

Matching pursuit algorithms build the support incrementally. At each iteration, one more atom is added to the support set of the solution candidate $\mathbf{x}$. We terminate the algorithm either after a fixed number of iterations K or when the magnitude of the residual $\|\mathbf{y}-\mathbf{D}\mathbf{x}\|_2^2$ reaches a specified threshold.

Algorithm 1 Matching Pursuit Algorithms - Pseudocode

1: **INPUTS:** Input vector $\mathbf{y}$, Dictionary $\mathbf{D}$
2: **Initialize**: residual as the input vector; solution as zero vector; support as empty set.
3: **repeat**
4: **Select atom**: add to support highest correlated atom with current residual.
5: **Update solution estimate**: Update coefficients within the support.
6: **Update residual**: calculate residual w.r.t. current reconstruction.
7: **until**
8: **OUTPUTS:** Sparse coefficient vector $\mathbf{x}$

Notation A vector $\mathbf{x}$ restricted to a support set $\mathcal{S}$ (i.e. containing only the entries indexed by the set $\mathcal{S}$) is denoted $\mathbf{x}_\mathcal{S}$. Similarly, $\mathbf{D}_\mathcal{S}$ denotes a matrix $\mathbf{D}$ restricted to the columns indexed by the set $\mathcal{S}$, and $\mathbf{D}_{\mathcal{S},\mathcal{S}}$ a matrix restricted to both columns and lines indexed by $\mathcal{S}$. The j-th entry of a vector $\mathbf{x}$ is denoted x_j and the entry in column i and line j of a matrix $\mathbf{D}$ is denoted $D_{i,j}$. By convention, the unspecified entries of a vector are assumed to be zero.

Matching pursuit

For compactness, we do not provide the detailed algorithm for Matching Pursuit (MP). Instead, we provide that of Orthogonal Matching Pursuit in Alg. 2, which is very similar. The only difference is in the *solution update* step. In Matching pursuit, only the coefficient corresponding to the last selected atom is updated. All other coefficients (from previously selected atoms) are kept unchanged. This comes down to replacing line 8 in Alg. 2 by:

$$\mathbf{x}^{k+1} = \mathbf{x}^k; \qquad x_{j^\star}^{k+1} = x_{j^\star}^k + \mathbf{d}_{j^\star}^T \mathbf{r}^k / \|\mathbf{d}_{j^\star}\|_2^2$$

In Matching pursuit, differently from its Orthogonal version, a same atom can be selected more than once. Also, although being less complex, MP generally leads to less accurate approximations than OMP.

Orthogonal matching pursuit

In Orthogonal Matching Pursuit, the coefficients of the entire support are recalculated every time an atom is added (i.e. every iteration). The update corresponds to solving a least squares problem on $\mathbf{x}$ restricted to the current support $\mathcal{S}$ for optimally approximating $\mathbf{y}$, which gives: $\mathbf{x}_{\mathcal{S}} = \mathbf{D}_{\mathcal{S}}^{\dagger}\mathbf{y}$, where $\dagger$ denotes the Moore-Penrose pseudoinverse.

Algorithm 2 $\mathbf{x} = \mathrm{OMP}(\mathbf{D}, \mathbf{y})$

1: ▷ *Initialization*
2: $\mathbf{r}^0 = \mathbf{y}; \quad \mathbf{x}^0 = \mathbf{0}; \quad \mathcal{S}^0 = \mathrm{support}(\mathbf{x}^0) = \emptyset; \quad k = 0;$
3: **while** stopping criterion is not met **do**
4: ▷ *Select atom*
5: $j^\star = \mathrm{argmax}_j \, |\mathbf{d}_j^T \mathbf{r}^k| / \|\mathbf{d}_j\|_2$
6: $\mathcal{S}^{k+1} = \mathcal{S}^k \cup \{j^\star\}$
7: ▷ *Update solution estimate*
8: $\mathbf{x}_{\mathcal{S}^{k+1}}^{k+1} = \mathbf{D}_{\mathcal{S}^{k+1}}^{\dagger}\mathbf{y}$
9: ▷ *Update residual*
10: $\mathbf{r}^{k+1} = \mathbf{y} - \mathbf{D}_{\mathcal{S}^{k+1}}\mathbf{x}_{\mathcal{S}^{k+1}}^{k+1}$
11: $k = k + 1$
12: **end while**

Select atom (line 5-6 in in Alg. 2) At each iteration, the atom the most correlated with the current residual is selected. Its index, denoted $j^\star$, is added to the support $\mathcal{S}$. This criterion is equivalent to choosing the atom which minimizes the representation error $\epsilon(j)$ when reconstructing the residual. By choosing wisely the j-th coefficient x_j, we have:

$$\epsilon(j) = \min_{x_j} \|x_j \mathbf{d}_j - \mathbf{r}\|_2^2 = \|\mathbf{r}\|_2^2 - \frac{(\mathbf{d}_j^T \mathbf{r})^2}{\|\mathbf{d}_j\|_2^2}.$$

Therefore, $j^\star = \mathrm{argmin}_j \, \epsilon(j) = \mathrm{argmax}_j \, |\mathbf{d}_j^T \mathbf{r}| / \|\mathbf{d}_j\|_2$.

Update solution estimate (line 8 in in Alg. 2) This is done by minimizing the approximation error $\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2$ with respect to $\mathbf{x}$, restricting its support to $\mathcal{S}$.

$$\begin{aligned} & \mathbf{x} = \underset{\mathbf{x}'}{\mathrm{argmin}} \, \|\mathbf{y} - \mathbf{D}\mathbf{x}'\|_2^2, \ s.t. \ \mathrm{support}(\mathbf{x}') = \mathcal{S} \\ \iff \quad & \mathbf{x}_{\mathcal{S}} = \underset{\mathbf{x}'_{\mathcal{S}}}{\mathrm{argmin}} \, \|\mathbf{y} - \mathbf{D}_{\mathcal{S}}\mathbf{x}'_{\mathcal{S}}\|_2^2 = \mathbf{D}_{\mathcal{S}}^{\dagger}\mathbf{y} \end{aligned}$$

Update redidual (line 10 in in Alg. 2) We denote $\mathbf{r} = \mathbf{y} - \mathbf{D}\mathbf{x}$ the residual. Since the support $\mathcal{S}$ is known, we can simply calculate $\mathbf{r} = \mathbf{y} - \mathbf{D}_{\mathcal{S}}\mathbf{x}_{\mathcal{S}}$.

Least Squares via the Cholesky Update

The OMP least squares step can be computed more efficiently using the Cholesky decomposition.

Definition 5 (Cholesky factorization)**.** *Let* $\mathbf{A}$ *be a symmetric, positive-definite matrix. There exists a real lower triangular matrix* $\mathbf{L}$ *such that* $\mathbf{A} = \mathbf{L}\mathbf{L}^T$*.*

We maintain the Cholesky decomposition of $\mathbf{G}^k = \mathbf{G}_{\mathcal{S}^k,\mathcal{S}^k} = \mathbf{D}_{\mathcal{S}^k}^T\mathbf{D}_{\mathcal{S}^k}$, the Gram matrix of $\mathbf{D}$ restricted to the current support $\mathcal{S}^k$ at iteration k. It can be used to solve the LS equation

$$\begin{aligned}
& \mathbf{x}_{\mathcal{S}^k} = \mathbf{D}_{\mathcal{S}^k}^{\dagger}\mathbf{y} \\
\iff \quad & \mathbf{x}_{\mathcal{S}^k} = \left(\mathbf{D}_{\mathcal{S}^k}^T\mathbf{D}_{\mathcal{S}^k}\right)^{-1}\mathbf{D}_{\mathcal{S}^k}^T\mathbf{y} \\
\iff \quad & \left(\mathbf{D}_{\mathcal{S}^k}^T\mathbf{D}_{\mathcal{S}^k}\right)\mathbf{x}_{\mathcal{S}^k} = \mathbf{D}_{\mathcal{S}^k}^T\mathbf{y} \\
\iff \quad & \mathbf{L}^k(\mathbf{L}^k)^T\mathbf{x}_{\mathcal{S}^k} = \mathbf{D}_{\mathcal{S}^k}^T\mathbf{y}
\end{aligned}$$

At each iteration, we need to update the Cholesky decomposition $\mathbf{L}^k$, compute $\mathbf{b} := \mathbf{D}_{\mathcal{S}^k}^T\mathbf{y}$, solve $\mathbf{L}^k\mathbf{u} = \mathbf{b}$ and then solve $(\mathbf{L}^k)^T\mathbf{x} = \mathbf{u}$.

The core insight of this acceleration is that one can determine the Cholesky decomposition of $\mathbf{G}^{k+1}$ from that of $\mathbf{G}^k$ with little extra computational effort. First, note that $\mathbf{G}^{k+1}$ can be related to $\mathbf{G}^k$ as follows

$$\mathbf{G}^{k+1} = \mathbf{D}_{\mathcal{S}^{k+1}}^T\mathbf{D}_{\mathcal{S}^{k+1}} = \begin{bmatrix} \mathbf{D}_{\mathcal{S}^k}^T\mathbf{D}_{\mathcal{S}^k} & \mathbf{D}_{\mathcal{S}^k}^T\mathbf{d}_{j^\star} \\ \mathbf{d}_{j^\star}^T\mathbf{D}_{\mathcal{S}^k} & \mathbf{d}_{j^\star}^T\mathbf{d}_{j^\star} \end{bmatrix} = \begin{bmatrix} \mathbf{G}^k & \mathbf{v} \\ \mathbf{v}^T & a_{j^\star} \end{bmatrix}$$

with $\mathbf{v} = \mathbf{D}_{\mathcal{S}^k}^T\mathbf{d}_{j^\star}$, $a_{j^\star} = \mathbf{d}_{j^\star}^T\mathbf{d}_{j^\star}$ and $j^\star$ the index of the atom selected at iteration $k+1$. One can now easily verify that the Cholesky update ($\mathbf{L}^{k+1}$ from $\mathbf{L}^k$) is given by:

$$\mathbf{L}^{k+1} = \begin{bmatrix} \mathbf{L}^k & 0 \\ \mathbf{w}^T & \sqrt{a_{j^\star} - \mathbf{w}^T\mathbf{w}} \end{bmatrix}$$

where $\mathbf{w}$ is given by solving $\mathbf{L}^k\mathbf{w} = \mathbf{v}$. For the first iteration, $\mathbf{L}^1 = a_{j^\star}$ is trivial.

Algorithm 3 $\mathbf{x} = \text{OMP-Cholesky}(\mathbf{D}, \mathbf{y})$

▷ *Initialization*
$\mathbf{r}^0 = \mathbf{y}; \quad \mathbf{x}^0 = \mathbf{0}; \quad \mathcal{S}^0 = \text{support}(\mathbf{x}^0) = \emptyset; \quad k = 0;$
while stopping criterion is not met **do**
    ▷ *Select atom*
    $j^\star = \text{argmax}_j \, |\mathbf{d}_j^T \mathbf{r}^k| / \|\mathbf{d}_j\|_2$
    ▷ *Cholesky update*
    if $k > 0$ **then**
        $\mathbf{w} = \text{Solve for } \mathbf{w} \;\; \left\{ \mathbf{L}^k \mathbf{w} = \mathbf{D}_{\mathcal{S}^k}^T \mathbf{d}_{j^\star} \right\}$
        $\mathbf{L}^{k+1} = \begin{bmatrix} \mathbf{L}^k & 0 \\ \mathbf{w}^T & \sqrt{a_{j^\star} - \mathbf{w}^T \mathbf{w}} \end{bmatrix}$
    end if
    $\mathcal{S}^{k+1} = \mathcal{S}^k \cup \{j^\star\}$
    ▷ *Update solution estimate*
    $\mathbf{x}_{\mathcal{S}^{k+1}}^{k+1} = \text{Solve for } \mathbf{x}_{\mathcal{S}^{k+1}} \;\; \left\{ \mathbf{L}^{k+1} (\mathbf{L}^{k+1})^T \mathbf{x}_{\mathcal{S}^{k+1}} = \mathbf{D}_{\mathcal{S}^{k+1}}^T \mathbf{y} \right\}$
    ▷ *Update residual*
    $\mathbf{r}^{k+1} = \mathbf{y} - \mathbf{D}_{\mathcal{S}^{k+1}} \mathbf{x}_{\mathcal{S}^{k+1}}^{k+1}$
    $k = k + 1$
end while

Batch OMP

When a large number of signals must be reconstructed using the same dictionary (for instance, in a Dictionary Learning task), it is worthwhile to consider pre-computation to reduce the total amount of work involved in coding the entire set. The Batch-OMP implementation [Rubinstein *et al.* 2008] follows this approach. [1]

The key observation is that the atom selection step at each iteration does not require knowing $\mathbf{r}$ or $\mathbf{x}$, but only $\mathbf{D}^T\mathbf{r}$. The idea is therefore to replace the explicit computation of $\mathbf{r}$ and its multiplication by $\mathbf{D}^T$ with a lower-cost computation of $\mathbf{D}^T\mathbf{r}$. Denoting $\boldsymbol{\alpha} = \mathbf{D}^T\mathbf{r}$, $\boldsymbol{\alpha}^0 = \mathbf{D}^T\mathbf{y}$, $\mathbf{G}_\mathcal{S} = \mathbf{D}^T\mathbf{D}_\mathcal{S}$ and $\mathbf{G}_{\mathcal{S},\mathcal{S}} = \mathbf{D}_\mathcal{S}^T\mathbf{D}_\mathcal{S}$, we can write

$$\begin{aligned} \boldsymbol{\alpha} &= \mathbf{D}^T(\mathbf{y} - \mathbf{D}_\mathcal{S}\mathbf{D}_\mathcal{S}^\dagger\mathbf{y}) \\ &= \boldsymbol{\alpha}^0 - \mathbf{G}_\mathcal{S}\mathbf{D}_\mathcal{S}^\dagger\mathbf{y} \\ &= \boldsymbol{\alpha}^0 - \mathbf{G}_\mathcal{S}(\mathbf{D}_\mathcal{S}^T\mathbf{D}_\mathcal{S})^{-1}\mathbf{D}_\mathcal{S}^T\mathbf{y} \\ &= \boldsymbol{\alpha}^0 - \mathbf{G}_\mathcal{S}(\mathbf{G}_{\mathcal{S},\mathcal{S}})^{-1}\boldsymbol{\alpha}_\mathcal{S}^0 \end{aligned} \tag{2.1}$$

1. The denomination Batch-OMP proposed by the authors is somewhat misleading. One would expect several input signals $\mathbf{y}$ to be processed simultaneously (in a batch), which is not the case.

This means that given the pre-computed $\boldsymbol{\alpha}^0$ and $\mathbf{G}$, we can compute $\boldsymbol{\alpha}$ each iteration without explicitly computing $\mathbf{r}$. The modified update step requires only multiplication by the matrix $\mathbf{G}_{\mathcal{S}}$ instead of the complete dictionary $\mathbf{D}^T$. The former is considerably less costly for small supports. Note that the matrix $\mathbf{G}_{\mathcal{S},\mathcal{S}}$ can be inverted using the progressive Cholesky factorization discussed above ($\mathbf{G}_{\mathcal{S}^k,\mathcal{S}^k} = \mathbf{G}^k$ in the previous notation).

The limitation of this approach is that since the residual is never explicitly computed, an error-based stopping criterion becomes challenging to employ. Fortunately, this problem can be circumvented by deriving a similar incremental formula for the ℓ_2 residual error (i.e. $\|\mathbf{r}\|_2^2$).

$$\begin{aligned}\mathbf{r}^{k+1} &= \mathbf{y} - \mathbf{D}\mathbf{x}^{k+1} \\ &= \mathbf{y} - \mathbf{D}\mathbf{x}^{k} + \mathbf{D}\mathbf{x}^{k} - \mathbf{D}\mathbf{x}^{k+1} \\ &= \mathbf{r}^k + \mathbf{D}(\mathbf{x}^k - \mathbf{x}^{k+1})\end{aligned} \tag{2.2}$$

The orthogonalization process in OMP ensures that at each iteration, the residual is orthogonal to the current signal approximation. We thus have for all k,

$$(\mathbf{r}^k)^T \mathbf{D}\mathbf{x}^k = 0. \tag{2.3}$$

Using expression (2.2) and property (2.3), we obtain after some manipulation the following recurrence for the squared approximation error:

$$\|\mathbf{r}^{k+1}\|_2^2 = \|\mathbf{r}^k\|_2^2 - (\mathbf{x}^{k+1})^T \mathbf{G}\mathbf{x}^{k+1} + (\mathbf{x}^k)^T \mathbf{G}\mathbf{x}^k \tag{2.4}$$

The complete algorithm, combined with the Cholesky update, is presented as Algorithm 4. Note that the algorithm becomes simpler when the stopping criterion is given by a fixed number of iterations. For better readability, we introduce the notations: $\boldsymbol{\beta} = \mathbf{G}\mathbf{x} = \mathbf{G}_{\mathcal{S}}\mathbf{x}_{\mathcal{S}}$, and $\delta = \mathbf{x}^T\mathbf{G}\mathbf{x} = \mathbf{x}_{\mathcal{S}}^T\boldsymbol{\beta}_{\mathcal{S}}$. Under this notation, the error update is given by $\|\mathbf{r}^{k+1}\|_2^2 = \|\mathbf{r}^k\|_2^2 - \delta^{k+1} + \delta^k$.

Note that the computation of δ^k at each iteration is extremely cheap, as intermediate variable $\boldsymbol{\beta}$ has to be computed anyway for the update of $\boldsymbol{\alpha}$. Therefore, the only added work in computing δ^k is the dot product between this vector $\boldsymbol{\beta}$ and the sparse vector $\mathbf{x}^k$, which requires a negligible amount of work. More details concerning the computational complexities are given below.

Algorithm 4 $\mathbf{x} = \text{Batch-OMP-Cholesky}(\mathbf{D}, \mathbf{y}, \mathbf{G} = \mathbf{D}^T\mathbf{D})$

▷ *Initialization*
$\mathbf{x}^0 = \mathbf{0}$; $\mathcal{S}^0 = \text{support}(\mathbf{x}^0) = \emptyset$; $k = 0$; $\boldsymbol{\alpha}^0 = \mathbf{D}^T\mathbf{y}$; $\|\mathbf{r}^0\|_2^2 = \mathbf{y}^T\mathbf{y}$;
while stopping criterion is not met **do**
    ▷ *Select atom*
    $j^\star = \text{argmax}_j \, |\alpha_j| / \sqrt{G_{j,j}}$
    ▷ *Cholesky update*
    if $k > 0$ **then**
        $\mathbf{w} = \text{Solve for } \mathbf{w} \;\; \left\{\mathbf{L}^k\mathbf{w} = \mathbf{G}_{\mathcal{S}^k, j^\star}\right\}$
        $\mathbf{L}^{k+1} = \begin{bmatrix} \mathbf{L}^k & 0 \\ \mathbf{w}^T & \sqrt{Gj^\star, j^\star - \mathbf{w}^T\mathbf{w}} \end{bmatrix}$
    end if
    $\mathcal{S}^{k+1} = \mathcal{S}^k \cup \{j^\star\}$
    ▷ *Update solution estimate*
    $\mathbf{x}^{k+1}_{\mathcal{S}^{k+1}} = \text{Solve for } \mathbf{x}_{\mathcal{S}^{k+1}} \;\; \left\{\mathbf{L}^{k+1}(\mathbf{L}^{k+1})^T\mathbf{x}_{\mathcal{S}^{k+1}} = \boldsymbol{\alpha}^0_{\mathcal{S}^{k+1}}\right\}$
    ▷ *Update* $\boldsymbol{\alpha}$
    $\boldsymbol{\beta} = \mathbf{G}_{\mathcal{S}^{k+1}}\mathbf{x}_{\mathcal{S}^{k+1}}$
    $\boldsymbol{\alpha} = \boldsymbol{\alpha}^0 - \boldsymbol{\beta}$
    ▷ *Update residual norm*
    if error-based stopping criterion **then**
        $\delta^{k+1} = \mathbf{x}^T_{\mathcal{S}^{k+1}}\boldsymbol{\beta}_{\mathcal{S}^{k+1}}$
        $\|\mathbf{r}^{k+1}\|_2^2 = \|\mathbf{r}^k\|_2^2 - \delta^{k+1} + \delta^k$
    end if
    $k = k + 1$
end while

Complexity Analysis

The detailed computational complexities of the different OMP implementations are shown in Table 2.1. The total results (both per iteration and for a total of K iterations) have been slightly simplified for improved readability. We consider the complexity of solving a $k \times k$ lower triangular system of equations (via back-substitution) to be k^2.

In OMP-Batch we suppose that $\mathbf{G}$ is pre-computed. Similarly, in OMP-Cholesky we suppose all $a_j = \mathbf{d}_j^T \mathbf{d}_j$ to be pre-calculated. Basically, OMP-Batch reduces the iteration cost by the expense of some extra initialization and memory cost. Namely: a total of $2nm$ operations for initializing $\boldsymbol{\alpha}^0$ and $2n$ for $\|\mathbf{r}\|_2^2$.

Since $m > n \gg k$, we can see that the complexities are dominated by the full matrix-vector products (those that are not restricted to the solution support) with a cost of $2nm$. This motivates the study of structured dictionaries (part II), which can mitigate this cost.

Table 2.1 – OMP complexity analysis

Operation	**OMP Naive**	**OMP Cholesky**	**OMP Batch**
Initialization	–	–	$2nm + 2n$
Select atom	$2nm + 2m$	$2nm + 2m$	$2m$
Cholesky update	–	$2nk + k^2$	k^2
Update solution	$2n(k^2 + k) + k^3 + 2k^2$	$2n + 2k^2$	$2k^2$
Update $\boldsymbol{\alpha}$	–	–	$2mk + m$
Update residual	$2nk + n$	$2nk + n$	$2k$
Total per iteration	$\backsim 2mn + 2nk^2 + k^3$	$\backsim 2mn + 4nk + 3k^2$	$\backsim 2mk + 3k^2$
Total K iterations	$2mnK + nK^3 + K^4$	$2mnK + 2nK^2 + K^3$	$2mn + mK^2 + K^3$

2.1.2 Other greedy pursuit approaches

Further research has led to more sophisticated pursuit methods than OMP. These techniques rely on several improvements to the basic greedy approaches, which include:

1. selecting multiple atoms per iteration;

2. pruning the set of active atoms at each step;

3. theoretical analysis, e.g. using the Restricted Isometry Property (RIP) [Candes & Tao 2006] .

To cite some methods, stagewise orthogonal matching pursuit (StOMP) [Donoho *et al.* 2012] selects multiple atoms at each step. The regularized orthogonal matching pursuit (ROMP) [Needell & Vershynin 2009], [Needell & Vershynin 2010] was the first provide a RIP-based analysis. OMP with replacement (OMPR) [Jain *et al.* 2011] introduces the idea of pruning the current support. Compressive sampling matching pursuit (CoSaMP) [Needell & Tropp 2009] assembles all the listed ideas to essentially obtain optimal performance guarantees. A similar algorithm, called subspace pursuit, was proposed in [Dai & Milenkovic 2009] with equivalent guarantees. CoSaMP is generally faster and more effective than OMP, except perhaps when the number of nonzeros in the representation is very small. Compared to the convex optimization approaches presented in the next section, CoSaMP can be faster but is usually less effective (in terms of sparsity/reconstruction error tradeoff).

2.2 Convex relaxation

In this section we present some optimization strategies that can tackle sparse representation problems in the ℓ_1 convex relaxation form. We will focus our analysis on first-order proximal gradient approaches (section 2.2.1), and briefly discuss some other convex approaches in section 2.2.2.

2.2.1 Proximal algorithms

Let us first consider a more general problem of minimizing the sum of two functions:

$$\min_{\mathbf{x}\in\mathbb{R}^m} \{F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x})\}. \tag{2.5}$$

We assume that f is convex and differentiable with a β-Lipschitz continuous gradient ∇f with $\beta \in]0, +\infty[$, i.e.,

$$\forall(\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{R}^m \times \mathbb{R}^m, \quad \|\nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2)\|_2 \leq \beta\|\mathbf{x}_1 - \mathbf{x}_2\|_2.$$

We do not suppose that g is differentiable, instead we suppose simply that g is a lower semicontinuous convex function [Rockafellar 1970, section 7] from $\mathbb{R}^m$ to $]-\infty, +\infty]$ with $\operatorname{dom} g := \{\mathbf{x} \in \mathbb{R}^m : g(\mathbf{x}) < +\infty\} \neq \emptyset$. Let us denote $\Gamma_0(\mathbb{R}^m)$ the space of all such functions.

Let us introduce the concept of a *proximity operator*, an extension of the notion of a projection operator proposed by [Moreau 1962]:

Definition 6 (Proximity operator)**.** *Let $g \in \Gamma_0(\mathbb{R}^m)$. For every $\mathbf{x} \in \mathbb{R}^m$ and for any $\gamma > 0$, the minimization problem*

$$\text{prox}_{\gamma g}(\mathbf{x}) = \underset{\mathbf{y}\in\mathbb{R}^m}{\text{argmin}}\, \gamma g(\mathbf{y}) + \tfrac{1}{2}\|\mathbf{x}-\mathbf{y}\|_2^2$$

admits a unique solution, which is denoted by $\text{prox}_{\gamma g}(\mathbf{x})$. The operator $\text{prox}_{\gamma g} : \mathbb{R}^m \to \mathbb{R}^m$ thus defined is the proximity operator of γg.

An important property of the proximity operator is the firm non-expansiveness:

$$\begin{aligned}&(\forall \mathbf{x}_1 \in \mathbb{R}^m)(\forall \mathbf{x}_2 \in \mathbb{R}^m) \\ &\|\,\text{prox}_g(\mathbf{x}_1) - \text{prox}_g(\mathbf{x}_2)\|_2^2 + \|(\mathbf{x}_1 - \text{prox}_g(\mathbf{x}_1)) - (\mathbf{x}_2 - \text{prox}_g(\mathbf{x}_2))\|_2^2 \leq \|\mathbf{x}_1 - \mathbf{x}_2\|_2^2.\end{aligned} \tag{2.6}$$

It can be shown [Combettes & Wajs 2005] that Problem (2.5) admits at least one solution and that its solutions are characterized by the fixed point equation $\mathbf{x} = \text{prox}_{\gamma g}\left(\mathbf{x} - \gamma \nabla f(\mathbf{x})\right)$. This equation suggests an iterative procedure called *proximal gradient algorithm*

$$\mathbf{x}^{k+1} = \text{prox}_{\gamma_k g}\left(\mathbf{x}^k - \gamma_k \nabla f(\mathbf{x}^k)\right) \tag{2.7}$$

for values of the step-size parameter γ_k in a suitable bounded interval. It is also referred to as *forward-backward splitting* since it can be broken up into a forward (explicit) gradient step using the function f, and a backward (implicit) step using the function g.

When the function g is a characteristic function (see Definition 9 below) of a certain convex and closed subset $\mathcal{A}$ of the domain, i.e. $g(\mathbf{x}) = \mathbb{I}_{\mathcal{A}}(\mathbf{x})$, the proximal step reduces to a projection over $\mathcal{A}$ and the proximal gradient algorithm in (2.7) reduces to the *projected gradient algorithm.*

Step-size Two main strategies that ensure convergence are used in practice. Basically, they both provide sufficient condition for the algorithm to produce a nonincreasing sequence of function values $F(\mathbf{x}^k)$ (we refer the reader to [Beck & Teboulle 2009] [Beck 2017, section 10.4.2] for the technical details, skipped here for conciseness).

- **Constant step**: Choose, for all k, $\gamma_k = \gamma \in]0, 2/\beta]$. Recall that β is the Lischitz constant associated to ∇f.

- **Backtracking**: At iteration k the choice of γ_k goes by: starting with a relatively large estimate of the step size and iteratively shrinking it (i.e., "backtracking") until a decrease of the objective function is observed that adequately corresponds to the expected decrease (related to the gradient magnitude at that point)[2]. The goal is to obtain a constant γ_k satisfying:

$$f(\mathbf{x}^{k+1}) \leq f(\mathbf{x}^k) + \langle \nabla f(\mathbf{x}^k), \mathbf{x}^{k+1} - \mathbf{x}^k \rangle + \frac{1}{2\gamma_k} \|\mathbf{x}^{k+1} - \mathbf{x}^k\|_2^2. \tag{2.8}$$

  where $\mathbf{x}^{k+1}$ is obtained from $\mathbf{x}^k$ using (2.7). While (2.8) is not satisfied, we set $\gamma_k = \eta\gamma_k$ with a shrinkage factor $\eta < 1$, and repeat.

Convergence rates and Nesterov acceleration The proximal gradient algorithm, with both the described step-size strategies, has a rate of convergence $\mathcal{O}(1/k)$ for the objective function. That is, after k iterations $F(\mathbf{x}_k) - F(\mathbf{x}^\star) \leq C/k$ for some positive constant C and a solution $\mathbf{x}^\star$ – see, for instance, [Beck 2017, theorem 10.21].
In the smooth setting (i.e. for $g(\mathbf{x}) \equiv 0$) a gradient method with convergence rate of $\mathcal{O}(1/k^2)$ has been proposed by [Nesterov 1983]. This convergence rate can be shown to be optimal for a first order method [Nemirovsky & Yudin 1983]. The remarkable fact is that the accelerated method demands roughly the same computational effort as the traditional gradient method (namely, one single gradient evaluation per iteration). The acceleration only requires the memorization of one extra solution iterate (not only the last estimate but also the previous one) and the computation of an additional point $\mathbf{z}^k$ that is smartly chosen and easy to compute. First we define the following sequences[3]:

$$t_0 = 0, \; t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}. \tag{2.9}$$

2. Backtracking line search for proximal gradient descent is similar to gradient descent but operates only on f, the smooth part of F. The firm non-expansiveness property of the proximity operator (see equation (2.6)) is crucial for this extension to the proximal gradient case.

3. Sequence (2.9) is just one example of suitable sequence, the one originally proposed in [Nesterov 1983]. A suitable sequence is one for which $\{F(\mathbf{x}_k)\}_{k\in\mathbb{N}}$ converges to $F(\mathbf{x}^\star)$ in the same $O(1/k^2)$ rate. Actually, one can choose for instance any sequence $\{t_k\}_{k\geq 0}$ satisfying for all $k \geq 0$: (a) $t_k \geq \frac{k+2}{2}$; (b) $t_{k+1}^2 - t_{k+1} \leq t_k^2$ (see [Beck 2017, section 10.7.2]). Also, weak convergence has been shown [Chambolle & Dossal 2015] for any sequence $t_k = \frac{k+a-1}{a}$ with $a > 2$.

Now the algorithm is simply defined by the following equations, with an arbitrary initial point $\mathbf{x}_0 = \mathbf{z}_0$,

$$\begin{aligned}\mathbf{x}^{k+1} &= \mathbf{z}^k + \gamma_k \mathbf{D}^T(\mathbf{y} - \mathbf{D}\mathbf{z}^k) \\ \mathbf{z}^{k+1} &= \mathbf{x}^{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(\mathbf{x}^{k+1} - \mathbf{x}^k)\end{aligned}$$

Informally, Nesterov's Accelerated Gradient Descent performs a regular gradient step to go from $\mathbf{z}^k$ to $\mathbf{x}^{k+1}$, and then it "slides" a little further than $\mathbf{x}^{k+1}$ in the direction given by the previous point $\mathbf{x}^k$.
This approach extends naturally to the proximal gradient setting by simply adding the proximal step on top of the accelerated gradient procedure (see Algorithm 6 for an example). The convergence rate of $\mathcal{O}(1/k^2)$ is also obtained in this case – see, for instance, [Beck 2017, theorem 10.34]).

Iterative Shrinkage-Thresholding algorithm

Applying the proximal gradient approach to the ℓ_1-regularized least-squares problem (P_1^λ), we obtain the classic Iterative Shrinkage-Thresholding Algorithm (ISTA) [Daubechies *et al.* 2004]. Using the previous notations we have:

$$f(\mathbf{x}) = \tfrac{1}{2}\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2, \qquad g(\mathbf{x}) = \lambda\|\mathbf{x}\|_1 \tag{2.10}$$

f is smooth with

$$\nabla f(\mathbf{x}) = \mathbf{D}^T(\mathbf{D}\mathbf{x} - \mathbf{y}) \tag{2.11}$$

which is β-Lipschitz continuous, with $\beta = \|\mathbf{D}^T\mathbf{D}\|$ and $\|\cdot\|$ denotes the matrix spectral norm given by the highest singular value. Also, the proximity operator associated to the ℓ_1-norm is given by the soft-thresholding operation denoted ST and defined coordinate-wise as follows

$$\Big(\mathrm{prox}_{\gamma\lambda\|\cdot\|_1}(\mathbf{x})\Big)_i = \mathrm{sign}(x_i)\cdot\max(0, |x_i| - \gamma\lambda) = \Big(\mathrm{ST}_{\gamma\lambda}(\mathbf{x})\Big)_i \tag{2.12}$$

ISTA can be compactly described as follows for step-size of γ_k at iteration k.

$$\mathbf{x}^{k+1} = \mathrm{ST}_{\gamma_k \lambda}\left(\mathbf{x}^k + \gamma_k \mathbf{D}^T(\mathbf{y} - \mathbf{D}\mathbf{x}^k)\right) \tag{2.13}$$

A detailed description is given in Algorithm 5.

Algorithm 5 $\mathbf{x} = \mathrm{ISTA}(\mathbf{D}, \mathbf{y})$

Fix $\mathbf{x}^0 \in \mathbb{R}^m$
while stopping criterion is not met **do**
    Choose step size γ_k
    ▷ *Proximal gradient step*
    $\mathbf{x}^{k+1} = \mathrm{ST}_{\gamma_k \lambda}\left(\mathbf{x}^k + \gamma_k \mathbf{D}^T(\mathbf{y} - \mathbf{D}\mathbf{x}^k)\right)$
    $k = k + 1$
end while

Fast Iterative Shrinkage-Thresholding algorithm

Applying the Nesterov acceleration to the ISTA algorithm leads to its fast variant FISTA [Beck & Teboulle 2009], which is described in Algorithm 6. The main difference with respect to the ISTA is that the proximal gradient step is not employed on the previous iterate $\mathbf{x}^k$, but rather on the intermediate variable $\mathbf{z}^k$ which consists of a very specific linear combination of the previous two iterates $\{\mathbf{x}^k, \mathbf{x}^{k-1}\}$. This simple modification allows to achieve faster convergence guarantees with some minor computational and memory overhead – two previous iterates have to be kept in memory instead of one.

Algorithm 6 $\mathbf{x} = \mathrm{FISTA}(\mathbf{D}, \mathbf{y})$

Fix $\mathbf{x}^0 \in \mathbb{R}^m$; Set $\mathbf{z}^0 = \mathbf{x}^0$; $k = 0$; $t_0 = 0$;
while stopping criterion is not met **do**
    Choose step size γ_k
    ▷ *Proximal gradient step*
    $\mathbf{x}^{k+1} = \mathrm{ST}_{\gamma_k \lambda}\left(\mathbf{z}^k + \gamma_k \mathbf{D}^T(\mathbf{y} - \mathbf{D}\mathbf{z}^k)\right)$
    ▷ *Update intermediate variable*
    $t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}$
    $\mathbf{z}^{k+1} = \mathbf{x}^{k+1} + \left(\frac{t_k - 1}{t_{k+1}}\right)(\mathbf{x}^{k+1} - \mathbf{x}^k)$
    $k = k + 1$
end while

Convergence rate While ISTA has a convergence rate of $\mathcal{O}(1/k)$, its improved variant FISTA attains a convergence rate of $\mathcal{O}(1/k^2)$ [Beck & Teboulle 2009], under the same conditions on step-size γ_k discussed in subsection 2.2.1.

Other proximal gradient algorithms for the lasso Several variations of proximal gradient algorithms have been proposed for the lasso. They can differ from ISTA and FISTA in several points, but in the end they are very similar in spirit and the theoretical basis remain the same. For instance in SpaRSA [Wright *et al.* 2009] the step-size is chosen by applying the so-called Brazilai-Borwein rule. TwIST [Bioucas-Dias & Figueiredo 2007], like FISTA, combines two previous iterates to accelerate the convergence, but in a slightly different way. Similarly, [Chambolle & Pock 2011] use a primal-dual framework to derive an algorithm comparable to FISTA but with different rules for updating the intermediate variables and stepsize. See [Bonnefoy *et al.* 2015, Table I] for a summary of the update steps in the mentioned algorithms.

Complexity analysis

The detailed computational complexities are shown in Table 2.2. Similarly to the OMP-like algorithms in section 2.1.1, the computational complexities are dominated by the matrix-vector operations involving the dictionary matrix. This is the case in both ISTA and FISTA. The requested additional computation for latter in the update of the intermediate variable $\mathbf{z}$ is clearly marginal.

Table 2.2 – (F)ISTA complexity analysis

Operation	**ISTA**	**FISTA**
Gradient step	$4nm + m + n$	$4nm + m + n$
Proximal step	$3m$	$3m$
Update $\mathbf{z}^k$	–	$3n$
Total per iteration	$4mn + 4m + n$	$4mn + 4m + 4n$

2.2.2 Other convex approaches

Linear programming and Interior-point methods

Some of the earliest approaches to solve ℓ_1-minimization problems were proposed in [Chen *et al.* 1998]. The authors exploit the fact that the (P_1) problem (equality-constrained ℓ_1-

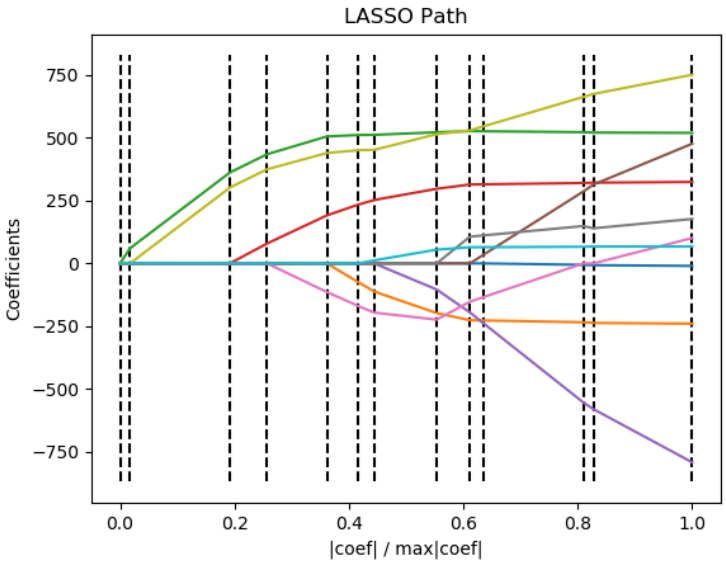


Figure 2.1 – Lasso coefficients (colored lines) as a function of the regularization penalty λ. Between two breakpoint (dashed lines) each coefficient is given by a linear function. Source: scikit-learn.org

minimization problem) can be cast as a linear programming (LP) problem to solve it using classic simplex methods, interior-point methods.

The ℓ_1-regularized least-squares problem (P_1^λ), in turn, can be cast as a quadratic programming (QP) problem which can also be tackled by interior-point methods [Chen *et al.* 1998] [Kim *et al.* 2007]. In general, interior-point methods are not competitive with the proximal gradient methods of section 2.2.1 on problems with very sparse solutions. On the other hand, their performance is insensitive to the sparsity of the solution or the value of the regularization parameter, which can make them interesting in specific scenarios.

Homotopy methods

Homotopy methods [Osborne *et al.* 2000] solve the lasso problem for all possible regularization values at once (also called the lasso path), for a fixed dictionary and input signal. A key observation is that, as the regularization penalty decreases, new atoms are progressively added to the solution which becomes less and less sparse –even if some atom removal may also occur. Actually, by inspecting the solution coefficients x_i as a function of the regularization λ, one observes a piecewise linear behavior as illustrated in figure 2.1. The task, thus, comes down to characterizing the breakpoints – i.e. when and which atom has to enter or leave the support.

The most prominent method of this type is the least-angle regression algorithm (LARS) [Efron *et al.* 2004], which is quite competitive when the goal is to characterize a full regularization path. It starts with the highest possible regularization (such that the solution is trivially the all-zeros vector) $\lambda = \|\mathbf{D}^T\mathbf{y}\|_\infty$ and then progressively reduces λ while testing if an atom has to be added to or removed from the support. The next breakpoint is thus identified and the solution coefficients are recalculated for the new support (which is enough to completely characterize the coefficients behavior since the last breakpoint, due to the mentioned linearity). There are two necessary conditions for this algorithm to work properly, which can also be an obstacle sometimes: (i) one single event can happen at each breakpoint, either one single atom enters or leaves the support $\mathcal{S}$, (ii) $\mathbf{D}_\mathcal{S}^T\mathbf{D}_\mathcal{S}$ has to be invertible for all supports throughout the lasso path, which is actually a sufficient condition for unicity of the solution. LARS can therefore be inapplicable in some cases. In addition, numerical precision issues can easily harm its performance when conditions (i) or (ii) are barely satisfied (i.e. two very close events or an ill-conditioned $\mathbf{D}_\mathcal{S}^T\mathbf{D}_\mathcal{S}$).

Active set methods

This set of algorithms comes as complement to the previously described solvers, since the proposed strategy can be combined to almost any existing lasso solver.

The active set methods (also called working sets) [Kim & Park 2010,Kowalski *et al.* 2011, Loth 2011,Johnson & Guestrin 2015] decompose the original problem into several smaller sub-problems –restricted to a subset of the atoms, the active set. Then, if the optimality conditions (see section 2.3.4) are not verified, the support is increased by including more promising atoms. Usually, the atoms the more correlated to the residual are chosen.

The motivation of such methods is that each sub-problem can be more readily solved (using any existing lasso solver) and, as such, one hopes to accelerate the resolution of the initial problem.

This type of correlation-based criterion is heuristic and does not guarantee to select the exact support at a given step. The choice of an efficient heuristic is, therefore, crucial to the success of this type of strategy.

(Block) Coordinate Descent

Coordinate (or block coordinate) descent (BCD) is a quite general and efficient optimization strategy, which consist in optimizing with respect to one coordinate (or a block of coordinates) at a time and looping until convergence. The interest of this type of method

is the considerably low cost of each individual minimization, both in terms of computational and memory requirements. Some paralelization can also envisioned [Fercoq & Richtárik 2015]. This strategy is particularly useful when the dictionaries are not associated to a fast implicit operator (e.g. FFT, wavelets) but rather given by explicit matrices.

In the context of sparse regression problem, (block) coordinate descent has been first suggested by [Fu 1998] and has since had a great success especially in the statistical machine learning community [Tseng 2001] [Friedman *et al.* 2007] [Wu & Lange 2008] [Shalev-Shwartz & Zhang 2016]. Different BCD strategies exist depending on how one iterates over coordinates: it can be a cyclic rule as used by [Friedman *et al.* 2007], random [Shalev-Shwartz & Zhang 2016], or greedy (meaning that the updated coordinate is the one leading to the best improvement on the objective or on a surrogate) [Wu & Lange 2008]. The latter rule, recently studied by [Tseng & Yun 2008] [Nutini *et al.* 2015] is historically known as the Gauss-Southwell (GS) rule [Southwell 1940].

2.3 Duality

In this section, after introducing some basic definitions, we review the Lagrangian duality framework (subsection 2.3.1) and the optimality conditions (subsection 2.3.2). These concepts are then applied to the lasso problem in subsections 2.3.3 and 2.3.4.

Preliminary definitions and notations

In constrained optimization problems, it is useful to define cost functions with value $+\infty$ outside the feasible region. For all $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, the domain of f, denoted by $\operatorname{dom} f$, is the set of points $\mathbf{x}$ such that $f(\mathbf{x}) < +\infty$, i.e $\operatorname{dom} f := \{\mathbf{x} \in \mathbb{R}^n : f(\mathbf{x}) < +\infty\}$.

Definition 7 (Proper function)**.** *A function f is called* proper *if* $\operatorname{dom} f \neq \emptyset$ *(i.e $f \not\equiv +\infty$) and if f never takes the value $-\infty$.*

Throughout this section, we consider only proper functions $f : \mathbb{R}^n \to (-\infty, +\infty]$. Other useful definitions:

Definition 8 (Convex function)**.** *A function $f : \mathbb{R}^n \to (-\infty, +\infty]$ is convex if* $\operatorname{dom} f$ *is a convex set and if for all $\mathbf{x}, \mathbf{y} \in \operatorname{dom} f$, and $0 \geq \alpha \geq 1$, we have*

$$f(\alpha\mathbf{x} + (1-\alpha)\mathbf{y}) \leq \alpha f(\mathbf{x}) + (1-\alpha) f(\mathbf{y}).$$

Definition 9 (Characteristic function)**.** *The* infinite indicator function *(or* characteristic function*)* $\mathbb{I}_A : \mathbb{R}^n \to (-\infty, +\infty]$ *is defined as follows for a subset A of a domain X:*

$$\mathbb{I}_A(\mathbf{u}) := \begin{cases} 0, & \mathbf{u} \in A; \\ +\infty, & \mathbf{u} \notin A; \end{cases} \tag{2.14}$$

Definition 10 (Elementwise comparison)**.** *For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we write $\mathbf{x} \preceq \mathbf{y}$ if $\forall i \in \{1, \ldots, n\},\ x_i \leq y_i$. We define $\mathbf{x} \succeq \mathbf{y}$ analogously.*

Definition 11 (Affine hull)**.** *The affine hull,* $\mathrm{aff}(A)$*, of a set $A \subset \mathbb{R}^n$ is the smallest affine set containing A or, equivalently, the set of all affine combinations of elements of A.*

Definition 12 (Relative interior)**.** *Denoting $B(\mathbf{x}, \epsilon)$ a ball of radius ϵ centered on $\mathbf{x}$, the relative interior of a set A, denoted* $\mathrm{relint}(A)$*, is defined as:*

$$\mathrm{relint}(A) := \{\mathbf{x} \in A : \exists \epsilon > 0, B(\mathbf{x}, \epsilon) \cap \mathrm{aff}(A) \subseteq A\}.$$

Let us now introduce an important concept which generalizes the gradient for non-smooth functions. Informally, the subgradient of a function is the usual gradient, except at non-regular points. There, the subgradient is any slope that lies "under" the function. The set of all such slopes is called the subdifferential.

Definition 13 (Subgradient and subdifferential)**.** *Consider $f : \mathbb{R}^n \to \mathbb{R}$. We call $g \in \mathbb{R}^n$ a subgradient of f at a point $x \in \mathrm{dom}\, f$ if:*

$$\forall \mathbf{z} \in \mathrm{dom}\, f,\ f(\mathbf{z}) \geq f(\mathbf{x}) + \langle \mathbf{g}, \mathbf{z} - \mathbf{x} \rangle$$

The subdifferential $\partial f(\mathbf{x})$ of f at $\mathbf{x}$ is the set of all subsubgradients of f at $\mathbf{x}$.

Remark 1. *This is a global definition compared to the gradient which is local.*

Property 1. *Let $f : \mathbb{R}^n \to (-\infty, \infty]$ be a convex function, differentiable at $\mathbf{x}$. Then, $\partial f(\mathbf{x}) = \{\nabla f(\mathbf{x})\}$.*

Unconstrained optimality conditions A point $\mathbf{x} \in \mathbb{R}^n$ is called a minimizer of a function $f : \mathbb{R}^n \to \mathbb{R}$ if $f(\mathbf{x}) \leq f(\mathbf{y}),\ \forall y \in \mathrm{dom}\, f$. The set of minimizers of f is denoted $\mathrm{argmin}\, f$.

Theorem 5 (Fermat's rule)**.** *Consider a convex function f, then*

$$\mathbf{x} \in \operatorname{argmin} f \iff \mathbf{0} \in \partial f(\mathbf{x})$$

Proof. $\mathbf{x} \in \operatorname{argmin} f \iff \forall \mathbf{y},\ f(\mathbf{y}) \geq f(\mathbf{x}) + \langle \mathbf{0}, \mathbf{y} - \mathbf{x} \rangle \iff \mathbf{0} \in \partial f(\mathbf{x})$ □

2.3.1 Lagrangian Duality

Primal Problem

We consider the following optimization problem,

$$\min_{\mathbf{x} \in \operatorname{dom} \mathcal{P}} f(\mathbf{x}) \qquad \text{s.t.} \qquad \begin{cases} g_i(\mathbf{x}) \leq 0 & \forall i = 1 \dots q \\ h_j(\mathbf{x}) = 0 & \forall j = 1 \dots p \end{cases} \tag{$\mathcal{P}$}$$

In the sequel, we refer to g_i as the *inequality constraints* while we refer to h_j as the *equality constraints*. The problem ($\mathcal{P}$) is often referred to as the *primal problem* (the dual problem will be defined later). We suppose $\mathcal{P}$ to be a convex problem, which means

- f and g_i ($\forall i = 1 \dots q$) are convex functions.
- h_j ($\forall j = 1 \dots p$) are affine functions; i.e. $h_j(\mathbf{x}) = \mathbf{a}_j^T \mathbf{x} - \mathbf{b}_i$.

The domain $\operatorname{dom} \mathcal{P} \triangleq \operatorname{dom} f \cap_{i=1}^{q} \operatorname{dom} g_i \cap_{j=1}^{p} \operatorname{dom} h_j \subset \mathbb{R}^n$ is a convex set as it is an intersection of convex sets. The *primal feasible set* is a (convex[4]) subset of the domain in which all the contraints are satisfied. We assume this set to be non-empty.

Definition 14 (Feasible primal)**.** *A point $\mathbf{x} \in \mathbb{R}^n$ is said to be feasible for problem ($\mathcal{P}$) if (1) $\mathbf{x} \in \mathcal{D}$; (2) $g_i(\mathbf{x}) \leq 0$ for all $i = 1 \dots q$; (3) $h_j(\mathbf{x}) = 0$ for all $j = 1 \dots p$.*

The *primal optimal value* $p^\star \in [-\infty, +\infty)$ associated to ($\mathcal{P}$) is the infimum of $f(\mathbf{x})$ over the feasible set. A *primal solution* $\mathbf{x}^\star$ is a feasible primal point such that $p^\star = f(\mathbf{x}^\star)$. Note that $p^\star$ might be attained by multiple (even infinitely many) $\mathbf{x}^\star$. Also, there is no guarantee about the existence of a primal solution, i.e. that the primal value be attained.

4. The feasible set is also an intersection of convex sets: $\operatorname{dom} \mathcal{P}$ is convex; the constraints g_i define sublevel sets $\{\mathbf{x} \mid g_i(\mathbf{x}) \leq 0\}$ of convex functions, which are convex sets; and the constraints h_j define hyperplanes $\{\mathbf{x} \mid \mathbf{a}_i^T \mathbf{x} = \mathbf{b}_i\}$.

Remark [Equivalent formulations] Problem $(\mathcal{P})$ could be equivalently formulated with only inequality constraints. Indeed, each equality constraint $h_j(\mathbf{x}) = 0$ could be rewritten as a pair of inqualities $h_j(\mathbf{x}) \leq 0$ and $-h_j(\mathbf{x}) \leq 0$.
Similarly, $(\mathcal{P})$ could be rewritten as an unconstrained problem, using the characteristic function $\mathbb{I}_A(\cdot)$ introduced earlier, which gives: $\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) + \sum_{i=1}^q \mathbb{I}_{g_i(\mathbf{x})\leq 0} + \sum_{j=1}^p \mathbb{I}_{h_j(\mathbf{x})=0}$. We will, however, stick with our initial formulation.

Lagrangian

Definition 15 (Lagrangian)**.** *We define the* Lagrangian *associated with the problem* $(\mathcal{P})$ *as the function* $\mathcal{L} : \operatorname{dom}\mathcal{P} \times \mathbb{R}^q_+ \times \mathbb{R}^p \to \mathbb{R}$ *such that*

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\nu}) = f(\mathbf{x}) + \sum_{i=1}^{q} \mu_i g_i(\mathbf{x}) + \sum_{j=1}^{p} \nu_j h_j(\mathbf{x}), \tag{2.15}$$

where $\boldsymbol{\mu}, \boldsymbol{\nu}$ *are referred to as* dual variables*.*

Interpretation of the Lagrangian. Consider the unconstrained version of problem $(\mathcal{P})$ using characteristic functions:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \widetilde{f}(\mathbf{x}), \quad \text{with} \quad \widetilde{f}(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{q} \mathbb{I}_{g_i\leq 0}(\mathbf{x}) + \sum_{j=1}^{p} \mathbb{I}_{h_i=0}(\mathbf{x}) \tag{2.16}$$

The Lagrangian function now appears as a relaxed version of $\widetilde{f}(\mathbf{x})$ where the indicators $\mathbb{I}$ have been linearized. [Boyd & Vandenberghe 2004] qualifies these indicators functions as "infinitely hard" displeasure function, while the linearized version is said "soft".

Lagrange dual function

Definition 16 (Lagrange dual function)**.** *Given a couple of dual variables* $\boldsymbol{\mu} \in \mathbb{R}^q_+, \boldsymbol{\nu} \in \mathbb{R}^p$*, the* Lagrange dual function $D : \mathbb{R}^q \times \mathbb{R}^p \to \mathbb{R}$ *is defined as the infimum of the Lagrangian:*

$$D(\boldsymbol{\mu}, \boldsymbol{\nu}) = \inf_{\mathbf{x}\in\mathcal{D}} \mathcal{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\nu}). \tag{2.17}$$

We have the following property [Boyd & Vandenberghe 2004, section 5.1.2]

Property 2. *The Lagrange dual function* D *is concave, even if problem* $(\mathcal{P})$ *is not convex.*

Proof. Let $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2 \in \mathbb{R}^q$, $\boldsymbol{\nu}_1, \boldsymbol{\nu}_2 \in \mathbb{R}^p$ and $t \in [0,1]$. Define $\boldsymbol{\mu} \in \mathbb{R}^q$, $\boldsymbol{\nu} \in \mathbb{R}^p$ such that $(\boldsymbol{\mu}, \boldsymbol{\nu}) = t(\boldsymbol{\mu}_1, \boldsymbol{\nu}_1) + (1-t)(\boldsymbol{\mu}_2, \boldsymbol{\nu}_2)$. We have

$$\begin{aligned} D(\boldsymbol{\mu}, \boldsymbol{\nu}) &= \inf_{\mathbf{x} \in \mathcal{D}} \mathcal{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\nu}) \\ \text{by linearity of } \mathcal{L} \text{ w.r.t. } \boldsymbol{\mu} \text{ and } \boldsymbol{\nu}, \text{ we obtain} \\ &= \inf_{\mathbf{x} \in \mathcal{D}} t\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}_1, \boldsymbol{\nu}_1) + (1-t)\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}_2, \boldsymbol{\nu}_2) \\ &\geq \inf_{\mathbf{x} \in \mathcal{D}} t\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}_1, \boldsymbol{\nu}_1) + \inf_{\mathbf{x} \in \mathcal{D}} (1-t)\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}_2, \boldsymbol{\nu}_2) \\ &= tD(\boldsymbol{\mu}_1, \boldsymbol{\nu}_1) + (1-t)D(\boldsymbol{\mu}_2, \boldsymbol{\nu}_2), \end{aligned} \tag{2.18}$$

which concludes the proof. □

The dual function provides lower bounds on the optimal value $p^\star$ of the primal problem $(\mathcal{P})$. This property is called *weak duality*.

Property 3 (Weak duality)**.** *For any $\boldsymbol{\mu} \in \mathbb{R}^q_+, \boldsymbol{\nu} \in \mathbb{R}^p$, we have*

$$D(\boldsymbol{\mu}, \boldsymbol{\nu}) \leq p^\star. \tag{2.19}$$

Proof. Let $\mathbf{x}_0$ be a feasible point (see Definition 14) for problem $(\mathcal{P})$. We have:

$$\mathcal{L}(\mathbf{x}_0, \boldsymbol{\mu}, \boldsymbol{\nu}) = f(\mathbf{x}_0) + \sum_{i=1}^{q} \underbrace{\mu_i}_{\geq 0} \underbrace{g_i(\mathbf{x}_0)}_{\leq 0} + \sum_{j=1}^{p} \nu_j \underbrace{h_j(\mathbf{x}_0)}_{=0}$$

Then $\sum_{i=1}^q \mu_i g_i(\mathbf{x}_0) + \sum_{j=1}^p \nu_j h_j(\mathbf{x}_0) \leq 0$ so

$$\mathcal{L}(\mathbf{x}_0, \boldsymbol{\mu}, \boldsymbol{\nu}) \leq f(\mathbf{x}_0). \tag{2.20}$$

Hence

$$D(\boldsymbol{\mu}, \boldsymbol{\nu}) = \inf_{\mathbf{x} \in \mathcal{D}} \mathcal{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\nu}) \leq \mathcal{L}(\mathbf{x}_0, \boldsymbol{\mu}, \boldsymbol{\nu}) \leq f(\mathbf{x}_0). \tag{2.21}$$

Since this holds for any feasible $\mathbf{x}_0$, taking the infimum over $\mathbf{x}_0$ we obtain (2.19). □

Lagrange dual problem

As stated in property 3, the Lagrange dual function gives a lower bound on the optimal value of the problem $(\mathcal{P})$. A natural question is therefore *What is the best lower bound?* This leads to the definition of the following optimization problem.

Definition 17 (Lagrange dual problem)**.**

$$d^\star = \max_{\boldsymbol{\mu} \succeq \mathbf{0}, \boldsymbol{\nu}} D(\boldsymbol{\mu}, \boldsymbol{\nu}) \tag{$\mathcal{D}$}$$

Property 4. *The dual problem* $(\mathcal{D})$ *is convex (even if* $(\mathcal{P})$ *is not).*

A pair $(\boldsymbol{\mu}, \boldsymbol{\nu})$ is called *dual feasible* when $\boldsymbol{\mu} \succeq \mathbf{0}$ and $D(\boldsymbol{\mu}, \boldsymbol{\nu}) > -\infty$. We denote $d^\star \in \mathbb{R}$ the *dual optimal value* associated to $(\mathcal{D})$. We refer to $(\boldsymbol{\mu}^\star, \boldsymbol{\nu}^\star)$ as *dual optimal* if they are dual feasible and reach the optimal value $d^\star = D(\boldsymbol{\mu}^\star, \boldsymbol{\nu}^\star)$.

2.3.2 Optimality conditions

Strong duality

In the general case, using the weak duality property, one can check that $d^\star \leq p^\star$. Strong duality refers to pairs of primal-dual problems for which $d^\star = p^\star$.

When strong duality holds, it is therefore sufficient to solve the dual problem (which is convex) to obtain the solution to the primal problem (which may not be convex). The fact that strong duality allows to solve a formally simpler problem than the primal optimization problem is one of the main argument for using optimization algorithms in the dual space, such as dual ascent, ADMM and such.

Slater's constraint qualification

A natural question to ask: are there verifiable conditions on the primal problem, so that it is guaranteed that strong duality holds? A family of such conditions is referred to as Slater's constraint qualification, designed for convex primal problems.

Theorem 6 (Slater's conditions)**.** *Given a primal problem* $(\mathcal{P})$*, if*

- *Problem* $(\mathcal{P})$ *is convex.*
- $\exists \mathbf{x} \in \operatorname{relint}(\operatorname{dom} \mathcal{P})$ *such that*

$$\forall i, g_i(\mathbf{x}) < 0 \tag{2.22}$$

$$\forall j, h_j(\mathbf{x}) = 0 \tag{2.23}$$

then strong duality holds. In addition, the dual optimal value is attained, i.e., there exists a dual feasible pair $(\boldsymbol{\mu}^\star, \boldsymbol{\nu}^\star)$ *with* $D(\boldsymbol{\mu}^\star, \boldsymbol{\nu}^\star) = d^\star = p^\star$.

Slater's constraint qualification can be understood as "if there exist a *strictly* feasible point in the interior of the domain $\operatorname{dom}\mathcal{P}$, then strong duality holds". Note that the inequality constraints have to be strictly satisfied.

Duality gap and Dual certificates

Because of the weak duality, if a feasible pair of dual variables $(\boldsymbol{\mu}, \boldsymbol{\nu})$ is exhibited, then $D(\boldsymbol{\mu}, \boldsymbol{\nu})$ provides a lower bound on $p^\star$. This pair can therefore be used to *certify* the optimality of a solution candidate $\mathbf{x}$.

Definition 18. *A pair* $(\boldsymbol{\mu}, \boldsymbol{\nu})$ *is a dual certificate of* $p^\star$ *iff*

$$(\boldsymbol{\mu}, \boldsymbol{\nu}) \in \operatorname{dom} D \tag{2.24}$$

Note that for any $\mathbf{x}$ and given a dual certificate $(\boldsymbol{\mu}, \boldsymbol{\nu})$,

$$0 \leq f(x) - p^\star \leq f(\mathbf{x}) - D(\boldsymbol{\mu}, \boldsymbol{\nu}) =: \epsilon \tag{2.25}$$

Here, we can say, without knowing the true value of $p^\star$, that $f(\mathbf{x})$ is ϵ-suboptimal. The quantity $f(\mathbf{x}) - D(\boldsymbol{\mu}, \boldsymbol{\nu})$ is referred as the *duality gap* and can be used to as stopping criterion in primal-dual algorithms, where dual certificates are computed at each iteration. If strong duality holds, the duality gap becomes even more interesting, as it can be made arbitrarily small.

Complementary slackness

Let $(\mathbf{x}^\star, \boldsymbol{\mu}^\star, \boldsymbol{\nu}^\star)$ be an admissible tuple such that $f(\mathbf{x}^\star) = D(\boldsymbol{\mu}^\star, \boldsymbol{\nu}^\star)$ (*i.e.* strong duality holds). Under such conditions, we have the following properties:

Property 5. $\inf_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\mu}^\star, \boldsymbol{\nu}^\star) = \mathcal{L}(\mathbf{x}^\star, \boldsymbol{\mu}^\star, \boldsymbol{\nu}^\star)$*, i.e.* $\mathbf{x}^\star$ *is a minimizer of the Lagrangian function taken at optimal dual variables.*

Property 6 (Complementary Slackness)**.** $\mu_i^\star g_i(\mathbf{x}^\star) = 0, \ \forall i = 1, \ldots, q.$

Proof. A proof is obtained by studying the following (trivial) inequalities.

$$
\begin{aligned}
f(\mathbf{x}^\star) &= D(\boldsymbol{\mu}^\star, \boldsymbol{\nu}^\star) && (2.26)\\
&= \inf_{\mathbf{x}} f(\mathbf{x}) + \sum_i \mu_i^\star g_i(\mathbf{x}) + \sum_j \nu_j^\star h_j(\mathbf{x}) && (2.27)\\
&\leq f(\mathbf{x}^\star) + \sum_i \mu_i^\star g_i(\mathbf{x}^\star) + \sum_j \nu_j^\star h_j(\mathbf{x}^\star) && (2.28)\\
&\leq f(\mathbf{x}^\star) && (2.29)
\end{aligned}
$$

which means that both inequalities are actually equalities. The first inequality implies that that $\mathbf{x}^\star$ minimizes $\mathcal{L}(\mathbf{x}, \boldsymbol{\mu}^\star, \boldsymbol{\nu}^\star)$. The second inequality implies that

$$\sum_i \mu_i^\star g_i(\mathbf{x}^\star) = 0 \tag{2.30}$$

And since each term in this sum is nonpositive, we have that $\mu_i^\star g_i(\mathbf{x}^\star) = 0, \ \forall i$. □

Complementary slackness states that when strong duality holds, at a minimizer of the primal problem and a maximizer of a dual problem, the constraints are either active with a null Lagrange multiplier, or the constraints are inactive with a positive multiplier.

$$\mu_i^\star > 0 \implies g_i(\mathbf{x}^\star) = 0 \qquad \text{or, equivalently,} \qquad g_i(\mathbf{x}^\star) < 0 \implies \mu_i^\star = 0 \tag{2.31}$$

Karush-Kuhn-Tucker (KKT) optimality conditions

In a convex problem, the KKT conditions are a set of sufficient conditions for optimality. If, in addition, strong duality holds, then the KKT conditions become also necessary and provide a mean to assess the optimality of a triplet of variables $(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\nu})$. Basically, the KKT conditions group up feasibility, complementary slackness and the annulation of the gradient at the optimum – or the analogue condition on the subgradient if f, g_i, h_i are not differentiable, which is the Fermat's rule.

Theorem 7 (KKT Conditions)**.** *Denote as KKT conditions the following set of equations:*

$$\forall i,\ g_i(\mathbf{x}) \leq 0 \tag{KKT.1}$$
$$\forall j,\ h_j(\mathbf{x}) = 0 \tag{KKT.2}$$
$$\forall i,\ \mu_i \geq 0 \tag{KKT.3}$$
$$\forall i,\ \mu_i g_i(\mathbf{x}) = 0 \tag{KKT.4}$$
$$\mathbf{0} \in\ \partial_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\nu}). \tag{KKT.5}$$

If the primal problem is convex, any tuple $(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\nu})$ *that satisfies the KKT conditions is solution to the primal and dual problems. Reciprocally, if strong duality holds, any solution* $(\mathbf{x}^\star, \boldsymbol{\mu}^\star, \boldsymbol{\nu}^\star)$ *of the primal and dual problems must satisfy the KKT conditions.*

Necessity and sufficiency In whole generality, KKT conditions are not always necessary nor sufficient. However, we have the following implications:

- Convexity of the primal problem ⇒ sufficiency of KKT.
- (Slater's conditions ⇒) Strong duality ⇒ necessity of KKT, regardless of convexity.

Note that in (KKT.5) a more general condition on the subgradient was used. In the simpler case when f, g_i, h_i are differentiable, this condition becomes

$$\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\nu}) = 0. \tag{KKT.5'}$$

2.3.3 Lasso dual

In this subsection we apply the previous concepts to derive the dual of the lasso problem. Let us recall the lasso problem, for an observation $\mathbf{y} \in \mathbb{R}^n$ and a dictionary $\mathbf{D} \in \mathbb{R}^{n \times m}$:

$$\min_{\mathbf{x} \in \mathbb{R}^m}\ \tfrac{1}{2}\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_1 \tag{P_1^λ}$$

As a foreword, let us pose an important property of problem (P_1^λ).

Property 7. *Strong duality holds for the lasso problem.*

Proof. The cost function is convex, and there are no inequality constraints. Therefore, Slater's conditions are trivially satisfied. □

Since there are no constraints in (P_1^λ), a simple trick is needed for deriving the lasso dual with the usual Lagrangian framework. An equality constraint can be artificially created via the introduction of a new variable:

$$\min_{\mathbf{x},\mathbf{z}} \tfrac{1}{2}\|\mathbf{y}-\mathbf{z}\|_2^2 + \lambda\|\mathbf{x}\|_1 \quad \text{s.t.} \quad \mathbf{z} = \mathbf{D}\mathbf{x} \tag{$\mathcal{P}_{\text{lasso}}$}$$

We can now easily derive the Lagrangian function

$$\mathcal{L}(\mathbf{x},\mathbf{z},\boldsymbol{\nu}) = \tfrac{1}{2}\|\mathbf{y}-\mathbf{z}\|_2^2 + \lambda\|\mathbf{x}\|_1 + \boldsymbol{\nu}^T(\mathbf{z}-\mathbf{D}\mathbf{x}). \tag{2.32}$$

The dual function is given by:

$$D(\boldsymbol{\nu}) = \min_{\mathbf{x},\mathbf{z}} \mathcal{L}(\mathbf{x},\mathbf{z},\boldsymbol{\nu}) \tag{2.33}$$

After some calculation (postponed to Appendix A.1), the following closed form solution is obtained for the Dual function:

$$D(\boldsymbol{\nu}) = \begin{cases} \frac{1}{2}\left(\|\mathbf{y}\|_2^2 - \|\mathbf{y}-\boldsymbol{\nu}\|_2^2\right) \text{ if } \|\frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda}\|_\infty \leq 1 \\ -\infty \end{cases} \tag{2.34}$$

Lasso dual problem This yields the following dual problem $\max_{\boldsymbol{\nu}} D(\boldsymbol{\nu})$:

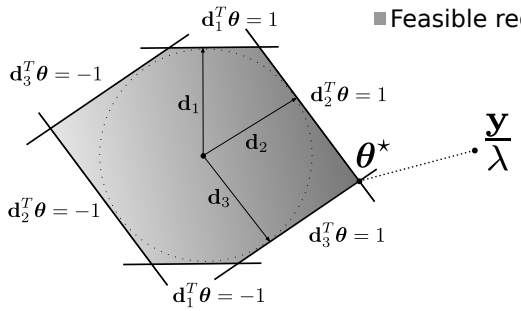
$$\max_{\boldsymbol{\nu}} \ \frac{1}{2}\left(\|\mathbf{y}\|_2^2 - \|\mathbf{y}-\boldsymbol{\nu}\|_2^2\right) \quad \text{s.t.} \quad \|\tfrac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda}\|_\infty \leq 1. \tag{$\mathcal{D}_{\text{lasso}}$}$$

For convenience, one may set $\boldsymbol{\theta} \triangleq \frac{\boldsymbol{\nu}}{\lambda}$, which yields:

$$\max_{\boldsymbol{\theta}} \ \frac{1}{2}\left(\|\mathbf{y}\|_2^2 - \lambda^2\|\frac{\mathbf{y}}{\lambda} - \boldsymbol{\theta}\|_2^2\right) \quad \text{s.t.} \quad \|\mathbf{D}^T\boldsymbol{\theta}\|_\infty \leq 1. \tag{2.35}$$

Note that (2.35) is a projection problem. Indeed, we are trying to find $\boldsymbol{\theta}$ in the feasible set $\|\mathbf{D}^T\boldsymbol{\theta}\|_\infty \leq 1$ which minimizes the distance to $\mathbf{y}/\lambda$. Therefore, the solution $\boldsymbol{\theta}^\star$ is the projection of $\mathbf{y}/\lambda$ on the feasible set $\|\mathbf{D}^T\boldsymbol{\theta}\|_\infty \leq 1$.

This is illustrated on figure 2.2. The constraint $\|\mathbf{D}^T\boldsymbol{\theta}\|_\infty \leq 1$ can be interpreted as a set of m constraints, one for each atom of the dictionary. Each of those leads to a pair of linear boundaries on the constraint set.



Feasible region

Constraints

$$\|\mathbf{D}^T\boldsymbol{\theta}\|_\infty \leq 1 \iff \begin{cases} -1 \leq \mathbf{d}_1^T\boldsymbol{\theta} \leq 1 \\ -1 \leq \mathbf{d}_2^T\boldsymbol{\theta} \leq 1 \\ -1 \leq \mathbf{d}_3^T\boldsymbol{\theta} \leq 1 \end{cases}$$

Figure 2.2 – Lasso dual problem as a projection problem with $n = 2$, $m = 3$ ($\boldsymbol{\theta} \in \mathbb{R}^2$, $\mathbf{D} \in \mathbb{R}^{2\times 3}$). Each atom leads to a pair of linear boundaries on the feasible region (in gray).

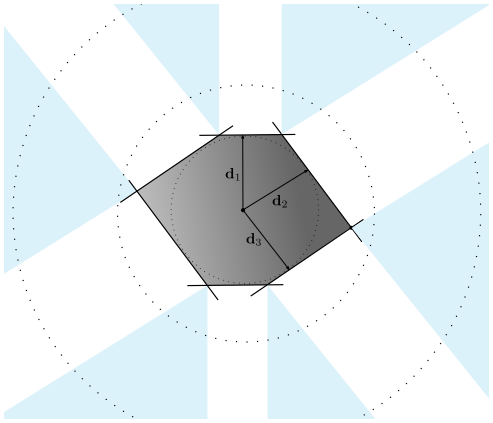


Figure 2.3 – All points lying in the blue zones are projected onto one of the corners of the polytope (which implies a primal solution with two non-zero entries). The remaining points, lying in the white stripes, are projected onto an edge (implying 1-sparse primal solutions). The further we get from the center of the polytope, the more considerable the blue area becomes (compared to the white area). Note how the outer circle crosses more blue, proportionally (in radians), than the inner circle.

Discussion As a projection problem, its solution will typically lie in the boundary of the feasible set (except for the trivial case of $\mathbf{y}/\lambda$ lying inside the feasible region, in which case the primal solution is the zero vector). The atoms associated to saturated constraints, i.e. the borders which are touched by the dual solution, are those associated to nonzero coefficients in a primal solution. Another intuition can be drawn from this problem when varying the regularization parameter λ: as λ gets smaller, $\mathbf{y}/\lambda$ gets further from the polytope defined by the constraints. The further $\mathbf{y}/\lambda$ lies from the polytope, the higher the chance that its projection lies in a corner (intersection between two atom constraints) rather than in a simple edge, as illustrated in Figure 2.3. This goes in line with the fact that weaker regularizations leads to less sparse solutions. Naturally, this analysis is quite limited in $\mathbb{R}^2$, since we only have edges and corners (respectively corresponding to 1-sparse and 2-sparse solutions). In higher dimensions, more possibilities arise. For instance, in $\mathbb{R}^3$ we have faces, edges and corners (respectively associated to 1, 2 and 3-sparse solutions).

2.3.4 Lasso optimality conditions

The optimality conditions of the lasso can be obtained in different ways. Here, we will show two ways of doing so.

The first one (in subsection *Fermat's rule*) is more straight-forward and simply applies the Fermat's rule, which provides necessary and sufficient optimality conditions for convex non-differentiable unconstrained problems (such as the lasso).

The second one (in subsection *KKT conditions*) uses the KKT conditions within the primal-dual framework discussed so far. Although it leads to a more convoluted derivation, it also provides some additional insights which will be useful for obtaining dual certificates and calculating duality gaps, as well as for the screening tests in part III of this thesis.

Fermat's rule

Fermat's rule for the lasso gives:

$$\begin{aligned} & \mathbf{0} \in \partial_{\mathbf{x}}(\tfrac{1}{2}\|\mathbf{y}-\mathbf{D}\mathbf{x}\|_2^2 + \lambda\|\mathbf{x}\|_1) \\ \iff & \tfrac{1}{\lambda}\mathbf{D}^T(\mathbf{y}-\mathbf{D}\mathbf{x}) \in \partial_{\mathbf{x}}\|\mathbf{x}\|_1. \end{aligned} \tag{2.36}$$

The subdifferential of the ℓ_1 norm is fairly easy to calculate. We can, for instance, exploit its separability to reason coordinate-wise $\varphi(\mathbf{x}) = \|\mathbf{x}\|_1 = \sum_i |x_i| = \sum_i \varphi_i(x_i)$. We use the subdifferential of the absolute value scalar function

$$\partial\varphi_i(x_i) \in \begin{cases} [-1,1] & \text{if } x_i = 0 \\ \{1\} & \text{if } x_i > 0 \\ \{-1\} & \text{if } x_i < 0 \end{cases}. \tag{2.37}$$

Fermat's rule thus becomes:

$$\mathbf{d}_i{}^T \frac{(\mathbf{y}-\mathbf{D}\mathbf{x})}{\lambda} = \begin{cases} \operatorname{sign}(x_i) & \text{if } x_i \neq 0 \\ \in [-1,1] & \text{otherwise} \end{cases} \tag{2.38}$$

where $\mathbf{d}_i$ is the i-th column of the matrix $\mathbf{D}$.

KKT conditions

Let us now write the KKT conditions for the lasso, considering the primal and dual problems ($\mathcal{P}_{\text{lasso}}$) and ($\mathcal{D}_{\text{lasso}}$).

- (KKT.1) (inequality constraints must be satisfied) does not apply here.
- (KKT.2) (equality contraints must be satisfied) gives $\mathbf{z} = \mathbf{D}\mathbf{x}$.
- (KKT.3) (lagrangian multipliers μ_i related to inequality constraints must be non-negative) does not apply.
- (KKT.4) (complementary slackness) does not apply.
- (KKT.5) (stationarity conditions, i.e. gradients vanish) see below.

The last KKT condition reads

$$\nabla_{\mathbf{z}} \mathcal{L}(\mathbf{x}, \mathbf{z}, \boldsymbol{\nu}) \;=\; 0 \tag{2.39a}$$

$$\mathbf{0} \;\in\; \partial_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{z}, \boldsymbol{\nu}). \tag{2.39b}$$

Eq. (2.39a) simply gives $\mathbf{z} = \mathbf{y} - \boldsymbol{\nu}$.

In Eq. (2.39b) we used the subdifferential since the ℓ_1 norm is not differentiable. By reasoning similarly to equation (2.38) in the previous subsection, we obtain:

$$\mathbf{d}_i{}^T \frac{\boldsymbol{\nu}}{\lambda} = \begin{cases} \operatorname{sign}(x_i) & \text{if } x_i \neq 0 \\ \in [-1, 1] & \text{otherwise.} \end{cases} \tag{2.40}$$

Summary Using the notation $\boldsymbol{\theta} = \frac{\nu}{\lambda}$.

$$\mathbf{d}_i{}^T \boldsymbol{\theta} = \begin{cases} \operatorname{sign}(x_i) & \text{if } x_i \neq 0 \\ \in [-1, 1] & \text{otherwise.} \end{cases} \tag{2.41}$$

Finally, combining (KKT.2) and (2.39a) we obtain the following relation between the primal $\mathbf{x}$ and dual $\boldsymbol{\theta}$ variables at the optimum:

$$\lambda \boldsymbol{\theta} = \mathbf{y} - \mathbf{D}\mathbf{x}. \tag{2.42}$$

Trivial lasso solution

In view of the dual problem (2.35), it is now easy to notice that if $\lambda > \|\mathbf{D}^T\mathbf{y}\|_\infty$, then the zero vector is the unique lasso primal solution, i.e. $\mathbf{x}^\star = \mathbf{0}$. Indeed, under such conditions, $\mathbf{y}/\lambda$ lies inside the dual feasible set and the projection problem is trivially solved, with solution $\boldsymbol{\theta}^\star = \mathbf{y}/\lambda$. From the optimality conditions (2.41) we obtain for every atom $\mathbf{d}_i$, $\mathbf{d}_i^T\boldsymbol{\theta}^\star = \frac{\mathbf{d}_i^T\mathbf{y}}{\lambda} \leq \frac{\|\mathbf{D}^T\mathbf{y}\|_\infty}{\lambda} < 1$, which implies that all entries $x_i = 0$.

Dual certificates

If we are able to map a primal estimate $\mathbf{x}$ to a dual feasible $\boldsymbol{\theta}$, we can calculate the *duality gap* $f(\mathbf{x}) - D(\boldsymbol{\theta})$, which provides an optimality bound for $\mathbf{x}$ (i.e. how far $f(\mathbf{x})$ is from $f(\mathbf{x}^\star)$). In addition, since strong duality holds for the lasso, the duality gap tends to zero as the primal-dual pair converge to the optimal $(\mathbf{x}^\star, \boldsymbol{\theta}^\star)$.

Now, how can we exhibit a feasible dual point for the lasso problem?

Proposition 1 (Dual scaling)**.**

$$\forall \mathbf{z} \in \mathbb{R}^m \setminus \{\mathbf{0}\},\ \boldsymbol{\theta} = \frac{\mathbf{z}}{\|\mathbf{D}^T\mathbf{z}\|_\infty} \tag{2.43}$$

is always dual feasible.

Proof. By construction $\|\mathbf{D}^T\boldsymbol{\theta}\|_\infty = 1$. Recall that here a feasible dual variable needs to satisfy the set constraint in (2.35), $\|\mathbf{D}^T\boldsymbol{\theta}\|_\infty \leq 1$. □

So, given a solution estimate $\mathbf{x}$, we can calculate a dual feasible $\boldsymbol{\theta}$ by taking $\mathbf{z} = \mathbf{y} - \mathbf{D}\mathbf{x}$ in (2.43). This dual point $\boldsymbol{\theta}$ is interesting because it can be shown that as $\mathbf{x} \to \mathbf{x}^\star$, it also converges to the optimal $\boldsymbol{\theta} \to \boldsymbol{\theta}^\star$. Indeed, from (2.41) and (2.42), we know respectively that $\lambda = \|\mathbf{D}^T(\mathbf{y} - \mathbf{D}\mathbf{x}^\star)\|_\infty$ and $\boldsymbol{\theta}^\star = (\mathbf{y} - \mathbf{D}\mathbf{x}^\star)/\lambda$.

Chapter 3

Tensor formalism

There are many ways to define a tensor. Probably the simplest one is to see it as a multiway array of data, as usually done in the data science community. Although this definition doesn't quite capture the entire complexity and generality of tensors, it has the advantage of providing an immediate visualization of the concept. For instance, let us define a three-way array (the extension to a higher number of ways is straightforward).

Definition 19. *(Multidimensional array) A three way array $\boldsymbol{\mathcal{A}}$ of size $I_1 \times I_2 \times I_3$ is an element of the finite-dimensional real vector space $\mathbb{R}^{I_1 \times I_2 \times I_3}$. It is uniquely defined by the set of its coordinates A_{i_1,i_2,i_3} where $1 \leq i_1 \leq I_1$, $1 \leq i_2 \leq I_2$ and $1 \leq i_3 \leq I_3$.*

In this interpretation, tensors are a generalization of vectors and matrices to a higher order. But in a more general way, tensors can be seen as a generalization of linear maps. In this setting, a tensor is identified to a multilinear map from vector spaces to a resulting vector space [Landsberg 2012, Section 2.3]. This is a basis-independent definition, which embodies a more general class of objects than finite-dimensional arrays. In the same way as linear maps are more general than matrices, tensors are more general than multiway arrays. In the case of finite-dimensional real vector spaces with fixed bases, just like linear maps can be represented by matrices, tensors can be represented by arrays of numerical values – or coordinates – related to the specific bases.

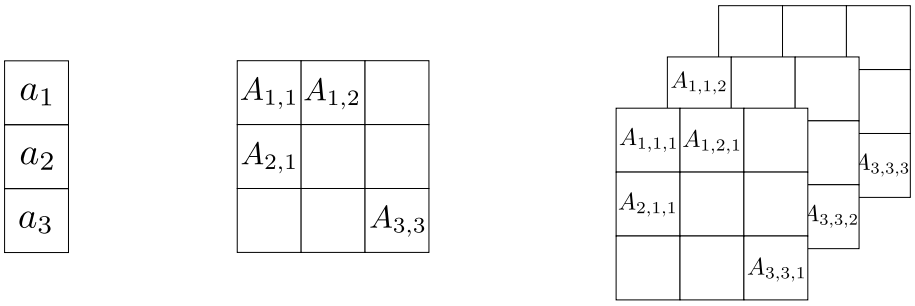


Figure 3.1 – Exemples of one, two and three-dimensional arrays.

Foreword: Having mentioned this more general view, it is important to emphasize that the multiway-array interpretation is sufficient for understanding most of the concepts and tools evoked in this thesis (except maybe the tensor product). In section 3.2.3 we provide another general definition of tensors based on the tensor product. But, in practice, we will deal with finite real multi-way arrays. For further readings on multiway array processing we refer the reader to [Bro 1998] [Kolda & Bader 2009] and for a more in-depth algebraic approach on tensors we refer the reader to [Hackbusch 2012] [Schwartz 1975].

3.1 Array manipulation

The concepts and operations defined in this section are much easier to understand with examples. We will often consider a generic multiway array as an illustration:

$$\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \quad \text{with entries } T_{i_1,i_2,\ldots,i_N},\ i_1 \in \{1,\ldots,I_1\},\ \ldots\ , i_N \in \{1,\ldots,I_N\}$$

Three-way arrays are sufficient to illustrate most of the concepts we need, so we will sometimes set $N=3$, giving $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$.

Let us introduce some vocabulary and basic concepts.

Order, modes, ways The order of a tensor is the number N of dimensions, or equivalently the number of indexes necessary to identify each entry of the array. It is also known as ways or modes. For instance, a matrix is an array of order two (or a 2-mode tensor). A 3-way array is given by a cube of data.

Frobenius norm The Euclidian norm of a vectorized matrix yields its Frobenius norm. Analogously, we define the squared Frobenius norm of a multiway array $\boldsymbol{\mathcal{T}}$ as the sum of the squares of all its element: $\|\boldsymbol{\mathcal{T}}\|_F^2 = \sum_{i_1,\ldots,i_N} T_{i_1,\ldots,i_N}^2$.

Subarrays: fibers and slices Subarrays are formed when a subset of the indices is fixed. A colon is used to indicate all elements of a given mode.

Fibers are defined by fixing every index but one, they are the higher-order analogue of matrix rows and columns. A column is a mode-1 fiber, a row is a mode-2 fiber, higher-order fibers are indistinctly called tube fibers. In a third-order tensor $\boldsymbol{\mathcal{T}}$ we denote respectively $\mathbf{t}_{:,i_2,i_3}$, $\mathbf{t}_{i_1,:,i_3}$ and $\mathbf{t}_{i_1,i_2,:}$ the possible fibers.

Slices are defined by fixing all but two indices, giving two-dimensional sections of

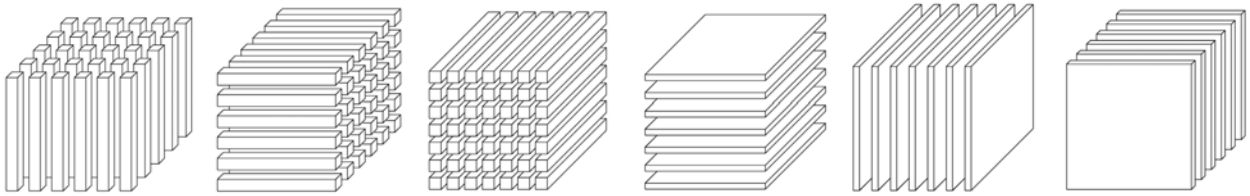


Figure 3.2 – Fibers and slices of a three-mode array.

a tensor. In a third-order tensor, there are three possible slices $\mathbf{T}_{i_1,:,:}$ (horizontal) $\mathbf{T}_{:,i_2,:}$ (lateral) and $\mathbf{T}_{:,:,i_3}$ (frontal). They are illustrated in Figure 3.2.

It is usual to represent a three way array by concatenating its frontal slices. For example, $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{2,3,2}$ with entries T_{i_1,i_2,i_3} can be represented as follows:

$$\left[\begin{array}{ccc|ccc} T_{1,1,1} & T_{1,3,1} & T_{1,3,1} & T_{2,1,2} & T_{2,2,2} & T_{2,3,2} \\ T_{2,1,1} & T_{2,2,1} & T_{2,3,1} & T_{2,1,2} & T_{2,2,2} & T_{2,3,2} \end{array}\right]$$

For more concrete illustrations, we will use a three-way array $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{3\times4\times2}$ given by

$$\boldsymbol{\mathcal{X}} = \left[\begin{array}{cccc|cccc} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{array}\right]$$

3.1.1 Vectorization

The vectorization operation rearranges the elements of a multiway array in a one-dimensional array. A convention has to be adopted on the order that the different indices are swept. It is important to emphasize that the ordering convention is not really important as long as it is consistent across related calculations [Kolda & Bader 2009]. The usual definition states that an array should be vectorized by sweeping the first mode with all other modes fixed, then the second and so on. In a data cube this means stacking the columns of the first frontal slice ($i_2 = 1$), then doing the same for the second slice ($i_2 = 2$) until the end. For a generic three-mode tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1,I_2,I_3}$ we have vec : $\mathbb{R}^{I_1,I_2,I_3} \to \mathbb{R}^{I_1I_2I_3}$ given by

$$\text{vec}(\boldsymbol{\mathcal{T}})_{(i_3-1)I_1I_2+(i_2-1)I_1+i_1} = T_{i_1,i_2,i_3}$$

For our concrete example with $\boldsymbol{\mathcal{X}}$ the resulting vector is:

$$\text{vec}(\boldsymbol{\mathcal{X}}) = \begin{bmatrix} 1 \\ 2 \\ 3 \\ \vdots \\ 24 \end{bmatrix}.$$

3.1.2 Tensorization

In general, tensorization is the process of forming a higher-order array by rearranging the elements of a smaller-order one (usually a vector or a matrix). Although there are plenty possible tensorization operations, we will be interested in a reverse operation to the vectorization. This particular tensorization operation transforms a vector into a tensor. For it to be completely specified, the dimensions of the resulting tensor has to provided. Given a vector $\mathbf{t} \in \mathbb{R}^{I_1 \cdots I_N}$, we denote $\text{tens}_{I_1,\ldots,I_N}(\mathbf{t}) \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ the resulting tensor.

This tensorization operation is such that, for any $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1,\ldots,I_N}$:

$$\text{tens}_{I_1,\ldots,I_N}(\text{vec}(\boldsymbol{\mathcal{T}})) = \boldsymbol{\mathcal{T}}$$

3.1.3 Matricization

Matricization, also called *unfolding* or *flattening*, is the process of reordering the elements of a multi-way array into a matrix. Among all the possible rearrangements that would lead to a matrix, some of them stand out. Here, we will focus on the special case of *mode-n unfolding*. A more general treatment of matricization can be found in [Kolda 2006].

Mode-n unfolding

The mode-n unfolding of a tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ is denoted by $\mathbf{T}_{(n)}$ and arranges its mode-n fibers as the columns of the resulting matrix. Now, the column ordering convention may vary, but once again, the most important is that it remains consistent across related calculations. We will adopt a standard convention [Kolda & Bader 2009] which sweeps the remaining indexes similarly to the vectorization operation, i.e. first modes first (i_1, then i_2 and so on).

The n-mode unfolding of a tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ gives a matrix $\mathbf{T}_{(n)} \in \mathbb{R}^{I_n \times I_1 \ldots I_{n-1} I_{n+1} \ldots I_N}$. In our example with tensor $\boldsymbol{\mathcal{X}}$, the three possible unfoldings are given by:

$$\mathbf{X}_{(1)} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix}$$

$$\mathbf{X}_{(2)} = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{bmatrix}$$

$$\mathbf{X}_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 & 24 \end{bmatrix}$$

Unvectorization

Similarly to the tensorization case, we will also consider an operation that reverses the vectorization of a matrix. So, it inputs a vector and outputs a matrix. For this operation to be fully specified, the dimensions of the resulting matrix have to be provided.

Given a vector $\mathbf{x} \in \mathbb{R}^I$, we denote $\text{unvec}_{I_1,I_2}(\mathbf{x})$ the output matrix of size $(I_1 \times I_2)$, provided that $I = I_1 I_2$. This particular matricization operation is such that, for any matrix $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2}$:

$$\text{unvec}_{I_1,I_2}(\text{vec}(\mathbf{X})) = \mathbf{X}.$$

Actually, this can be seen as an special case of the tensorization operation defined in section 3.1.2, since for any vector $\mathbf{x}$ we have that $\text{unvec}_{I_1,I_2}(\mathbf{x}) = \text{tens}_{I_1,I_2}(\mathbf{x})$. Nevertheless, we choose to explicitly distinguish the matrix case by keeping the distinct notation unvec.

3.1.4 Contractions

Tensor contraction is an operation involving two tensors which results in another tensor with order equal to the sum of the orders of each tensor minus two. In multi-way arrays, it consists of a summation over a pair of indices.

Matrix multiplication can be seen as a particular case of tensor contraction. However, for higher-order arrays there are more possible combinations of indexes to be contracted

than in the matrix case.

Given two tensors $\boldsymbol{\mathcal{T}}$ and $\boldsymbol{\mathcal{T}}'$ of order N and N', we denote $\boldsymbol{\mathcal{T}} \bullet_{n,n'} \boldsymbol{\mathcal{T}}'$ a contraction in which the summation is made over the n-th index of $\boldsymbol{\mathcal{T}}$ and the n'-th index of $\boldsymbol{\mathcal{T}}'$. The result $\boldsymbol{\mathcal{U}}$ is a tensor of order $N + N' - 2$ with entries:

$$U_{i_1,\ldots,i_{n-1},i_{n+1},\ldots,i_N,j_1,\ldots,j_{n'-1},j_{n'+1},\ldots,j_{N'}} = \sum_l T_{i_1,\ldots,i_{n-1},l,\ldots,i_N} T'_{j_1,\ldots,j_{n'-1},l,\ldots,j_{N'}}$$

Note that a convention was adopted for the mode ordering on the resulting tensor.

n-Mode product

For the purposes of this thesis, we are interested in a particular type of tensor contraction called the *n-mode product*, which consists in multiplying a tensor by a matrix in mode n. It has been the usual practice to always sum over the second matrix index. As a consequence, this kind of contraction is fully determined by a single index and we denote it by $\times_n$.

The n-mode product between a tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1,\ldots,I_N}$ and a matrix $\mathbf{M}^{J\times I_n}$ gives an N-mode tensor of size $(I_1 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_N)$ with elements

$$(\boldsymbol{\mathcal{T}} \times_n \mathbf{M})_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_N} = \sum_{l=1}^{I_n} T_{i_1,\ldots,i_{n-1},l,i_{n+1},\ldots,i_N} M_{jl}$$

The idea can also be expressed in terms of unfolded tensors:

$$\boldsymbol{\mathcal{U}} = \boldsymbol{\mathcal{T}} \times_n \mathbf{M} \quad \Longleftrightarrow \quad \mathbf{U}_{(n)} = \mathbf{M}\mathbf{T}_{(n)}.$$

For series of multiplications in distinct modes, the order of the multiplication is irrelevant, i.e. $\boldsymbol{\mathcal{T}} \times_{n_1} \mathbf{M_1} \times_{n_2} \mathbf{M_2} = \boldsymbol{\mathcal{T}} \times_{n_2} \mathbf{M_2} \times_{n_1} \mathbf{M_1}$ $(n_1 \neq n_2)$.

The n-mode product can also be defined between a tensor and a vector $\mathbf{v} \in \mathbb{R}^{I_n}$. The resulting tensor $\boldsymbol{\mathcal{T}} \times_n \mathbf{v}$, calculated similarly to the matrix case, is of order $N - 1$ with sizes $(I_1 \times \ldots I_{n-1} \times I_{n+1} \times \cdots \times I_N)$.

3.2 Tensor operations

We start by introducing two basic operations in the context of multiway array processing: the outer product (subsection 3.2.1) and the Kronecker product (subsection 3.2.2). We later define the more general notion of tensor product, in subsection 3.2.3. Despite being

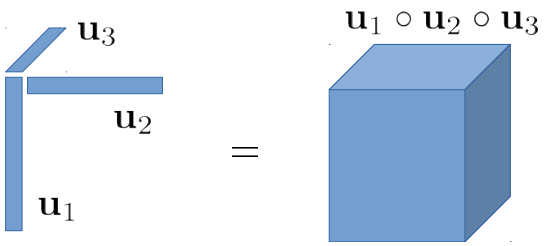


Figure 3.3 – The outer product of three vector leads to a three-way array.

the foundation of tensor algebra (one can for instance define tensors from the notion of tensor product) it is often skipped due to its more abstract nature. Instead, authors often favor its easier-to-grasp basis-dependent special cases: the outer product, which may be seen as a tensor product of vectors in the canonical basis of each vector space; and the Kronecker product, which is the expression of the tensor product for matrices.

3.2.1 Outer product

The outer product between two vectors $\mathbf{u} \in \mathbb{R}^{I_1}$ and $\mathbf{v} \in \mathbb{R}^{I_2}$ is a well-known operation that gives a matrix $(I_1 \times I_2)$ matrix:

$$\mathbf{u} \circ \mathbf{v} = \mathbf{u}\mathbf{v}^T = \begin{bmatrix} u_1 v_1 & \dots & u_1 v_m \\ \vdots & \ddots & \vdots \\ u_n v_1 & \dots & u_n v_m \end{bmatrix}$$

This concept can be generalized to the outer product between N vectors $\mathbf{u}_1 \circ \cdots \circ \mathbf{u}_N$, with $\mathbf{u}_n \in \mathbb{R}^{I_n}$. The result is a N-mode array of size $(I_1 \times \cdots \times I_N)$, with entries given by

$$(\mathbf{u}_1 \circ \cdots \circ \mathbf{u}_N)_{i_1, i_2 \dots, i_N} = u_{i_1} u_{i_2} \dots u_{i_N}, \quad \text{for all } 1 \leq i_n \leq I_n$$

The outer product can be seen as a building block of tensors, as they allow to generate elementary N-mode tensors from N vectors. Such elementary tensors are called rank-one tensors (in analogy to rank-one matrices, which are outer product between two vectors). We will get back to this point in section 3.2.3, where we formalize the intuition that these can be seen as elementary tensors. Figure 3.3 shows an example of a third-order array resulting from such an outer product.

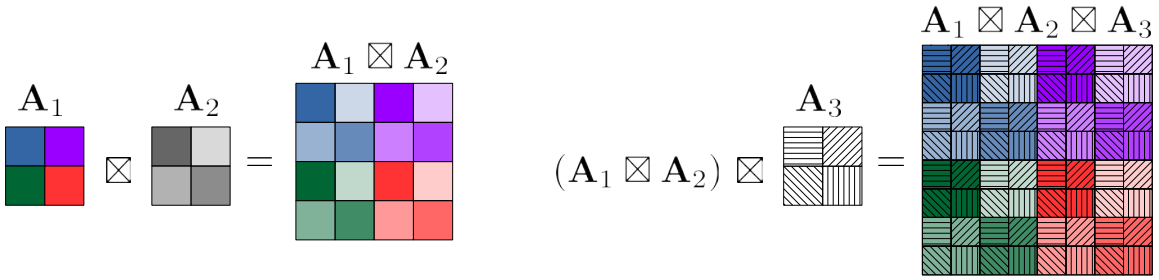


Figure 3.4 – Illustration of the Kronecker product of two (left) and three (right) matrices.

3.2.2 Kronecker product

Before defining the Kronecker product between two matrices, let us define it for two vectors $\mathbf{u} \in \mathbb{R}^{I_1}$, $\mathbf{v} \in \mathbb{R}^{I_2}$. We denote $\mathbf{u} \boxtimes \mathbf{v}$ the result, which is a vector of dimension $I_1 I_2$ given by

$$\mathbf{u} \boxtimes \mathbf{v} = \begin{bmatrix} u_1 \mathbf{v} \\ \vdots \\ u_{I_1} \mathbf{v} \end{bmatrix} = \begin{bmatrix} u_1 v_1 \\ \vdots \\ u_1 v_{I_2} \\ \vdots \\ u_{I_1} v_1 \\ \vdots \\ u_{I_1} v_{I_2} \end{bmatrix}.$$

This is closely related to the outer product. Indeed, we have the following relation:

$$\mathbf{u} \boxtimes \mathbf{u} = \text{vec}\left((\mathbf{u} \circ \mathbf{v})^T\right)$$

Now, for two matrices $\mathbf{A} \in \mathbb{R}^{I_1 \times J_1}$ and $\mathbf{B} \in \mathbb{R}^{I_2 \times J_2}$, the Kronecker product gives a matrix denoted $\mathbf{A} \boxtimes \mathbf{B}$ of size $(I_1 I_2 \times J_1 J_2)$ defined by

$$\mathbf{A} \boxtimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & a_{1,2}\mathbf{B} & \dots & a_{1,J_1}\mathbf{B} \\ a_{2,1}\mathbf{B} & a_{2,2}\mathbf{B} & \dots & a_{2,J_1}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I_1,1}\mathbf{B} & a_{I_1,2}\mathbf{B} & \dots & a_{I_1,J_1}\mathbf{B} \end{bmatrix}$$

which can be related to the Kronecker product between vectors

$$\mathbf{A} \boxtimes \mathbf{B} = \begin{bmatrix} \mathbf{a}_1 \boxtimes \mathbf{b}_1 & \dots & \mathbf{a}_1 \boxtimes \mathbf{b}_{J_2} & \mathbf{a}_2 \boxtimes \mathbf{b}_1 & \dots & \mathbf{a}_2 \boxtimes \mathbf{b}_{J_2} & \dots & \mathbf{a}_{J_1} \boxtimes \mathbf{b}_1 & \dots & \mathbf{a}_{J_1} \boxtimes \mathbf{b}_{J_2} \end{bmatrix}$$

This notion can be extended for N matrices, say $\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_N$ with $\mathbf{A}_n \in \mathbb{R}^{I_n \times J_n}$, by performing successive Kronecker products. For instance $\mathbf{A}_1 \boxtimes \mathbf{A}_2 \boxtimes \mathbf{A}_3 = (\mathbf{A}_1 \boxtimes \mathbf{A}_2) \boxtimes \mathbf{A}_3$ and, generalizing: $\mathbf{A}_1 \boxtimes \mathbf{A}_2 \cdots \boxtimes \mathbf{A}_N = ((\mathbf{A}_1 \boxtimes \mathbf{A}_2) \boxtimes \mathbf{A}_3) \cdots \boxtimes \mathbf{A}_N$. The resulting matrix is of size $(I_1 I_2 \ldots I_N \times J_1 J_2 \ldots J_N)$. Actually, we will see just below that the Kronecker product is associative, meaning that the order in which these multiple product are performed does not matter. Figure 3.4 illustrates the Kronecker product between two and three matrices.

Some useful properties

Let us now introduce some important properties of the Kronecker product, focusing on the ones which will useful in the remainder of this document. For a more complete list of properties, we refer the reader to [Horn & Johnson 2012, chapter 4].

Property 8 (Bilinearity)**.** *The Kronecker product is linear w.r.t. each of its entries, i.e. for all* $\mathbf{A} \in \mathbb{R}^{I_1 \times J_1}$, $\mathbf{B} \in \mathbb{R}^{I_2 \times J_2}$, $\mathbf{C} \in \mathbb{R}^{I_3 \times J_3}$, $\alpha \in \mathbb{R}$*:*

$$\begin{aligned}
&\mathbf{A} \boxtimes (\mathbf{B} + \mathbf{C}) = \mathbf{A} \boxtimes \mathbf{B} + \mathbf{A} \boxtimes \mathbf{C} \\
&(\mathbf{A} + \mathbf{B}) \boxtimes \mathbf{C} = \mathbf{A} \boxtimes \mathbf{C} + \mathbf{B} \boxtimes \mathbf{C} \\
&(\alpha \mathbf{A}) \boxtimes \mathbf{B} = \mathbf{A} \boxtimes (\alpha \mathbf{B}) = \alpha (\mathbf{A} \boxtimes \mathbf{B}) \\
&\mathbf{A} \boxtimes \mathbf{0} = \mathbf{0} \boxtimes \mathbf{A} = \mathbf{0}
\end{aligned}$$

Property 9 (Associativity)**.** *The Kronecker product is associative, i.e. for all* $\mathbf{A} \in \mathbb{R}^{I_1 \times J_1}$, $\mathbf{B} \in \mathbb{R}^{I_2 \times J_2}$, $\mathbf{C} \in \mathbb{R}^{I_3 \times J_3}$*:*

$$(\mathbf{A} \boxtimes \mathbf{B}) \boxtimes \mathbf{C} = \mathbf{A} \boxtimes (\mathbf{B} \boxtimes \mathbf{C})$$

However, it is important to point out that the Kronecker product is not commutative: $\mathbf{A} \boxtimes \mathbf{B} \neq \mathbf{B} \boxtimes \mathbf{A}$, even though these two terms can be related through permutation matrices [Henderson & Searle 1981].

Property 10 (Transpose)**.** *Transposition is distributive over the Kronecker product:*

$$(\mathbf{A} \boxtimes \mathbf{B})^T = \mathbf{A}^T \boxtimes \mathbf{B}^T$$

Property 11 (Mixed-product)**.** *The product of two Kronecker products yields another Kronecker product. If* $\mathbf{A}$*,* $\mathbf{B}$*,* $\mathbf{C}$ *and* $\mathbf{D}$ *are matrices of such size that one can form the*

matrix products $\mathbf{AC}$ *and* $\mathbf{BD}$*, then:*

$$(\mathbf{A} \boxtimes \mathbf{B})(\mathbf{C} \boxtimes \mathbf{D}) = (\mathbf{AC}) \boxtimes (\mathbf{BD})$$

The following property follows directly from the mixed product property 11.

Property 12 (Inverse of a Kronecker product)**.** $\mathbf{A} \boxtimes \mathbf{B}$ *is invertible if and only if both* $\mathbf{A}$ *and* $\mathbf{B}$ *are invertible, in which case the inverse is given by*

$$(\mathbf{A} \boxtimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \boxtimes \mathbf{B}^{-1}$$

The invertible product property holds for the Moore–Penrose pseudoinverse as well:

$$(\mathbf{A} \boxtimes \mathbf{B})^{\dagger} = \mathbf{A}^{\dagger} \boxtimes \mathbf{B}^{\dagger}$$

Relation with mode products

Suppose matrices $\mathbf{A}_n \in \mathbb{R}^{I_n \times J_n}$. A very interesting property of the Kronecker product emerges when it is multiplied by a vectorized matrix, say $\mathbf{X} \in \mathbb{R}^{J_1 \times J_2}$:

$$(\mathbf{A}_2 \boxtimes \mathbf{A}_1) \operatorname{vec}(\mathbf{X}) = \operatorname{vec}(\mathbf{A}_1 \mathbf{X} \mathbf{A}_2^T)$$

which means that each of the composing matrices $\mathbf{A}_1$ and $\mathbf{A}_2$ apply separately to the columns and rows of the matrix $\mathbf{X}$. This identity can also be expressed in terms of mode products:

$$\mathbf{y} = (\mathbf{A}_2 \boxtimes \mathbf{A}_1) \operatorname{vec}(\mathbf{X}) \iff \mathbf{Y} = \mathbf{X} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2,$$

denoting $\mathbf{y} \in \mathbb{R}^{I_1 I_2}$ the resulting vector and $\mathbf{Y}$ its analogous before vectorization, i.e. $\mathbf{y} = \operatorname{vec}(\mathbf{Y})$.

It can be extended to a higher order. When applying a Kronecker product of N matrices to an N-way array $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$, we have

$$\mathbf{y} = (\mathbf{A}_N \boxtimes \cdots \boxtimes \mathbf{A}_1) \operatorname{vec}(\boldsymbol{\mathcal{X}}) \iff \boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \times_1 \mathbf{A}_1 \cdots \times_N \mathbf{A}_N, \tag{3.1}$$

with $\mathbf{y} \in \mathbb{R}^{I_1 \cdots I_N}$ and $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, such that $\mathbf{y} = \operatorname{vec}(\boldsymbol{\mathcal{Y}})$.

This property is particularly useful when treating multi-dimensional data. A linear operator that can be written as a Kronecker product is sometimes referred to as a *separable operator*, since it acts independently on each mode of the data. We will explore this point in part II and push it further by allowing operators written as a sum of Kronecker products.

Another related property will be useful in part II. It is simply a different way of expressing (3.1) by using the n-mode unfolding to isolate the n-th Kronecker term:

$$\begin{aligned} & \boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \times_1 \mathbf{A}_1 \cdots \times_N \mathbf{A}_N \\ \iff & \mathbf{Y}_{(n)} = \mathbf{A}_n \mathbf{X}_{(n)} \left(\mathbf{A}_N \boxtimes \cdots \boxtimes \mathbf{A}_{n-1} \boxtimes \mathbf{A}_{n+1} \boxtimes \cdots \boxtimes \mathbf{A}_1\right)^T. \end{aligned} \tag{3.2}$$

It brings to light the linear dependency between $\mathbf{Y}_{(n)}$ and $\mathbf{A}_n$.

3.2.3 Tensor product

Before giving a formal definition of the tensor product, let us provide some intuition.

Given vector spaces $U_1, \ldots, U_N$ and W the tensor product $\otimes$ is basically a way of building a new vector space $U_1 \otimes \cdots \otimes U_N$ in which multilinear maps $U_1 \times \cdots \times U_N \to W$ become linear maps $U_1 \otimes \cdots \otimes U_N \to W$.

The multilinear maps defined below are very interesting since they act linearly in each of the original vector spaces *separately*. Hence, the interest of the tensor product is that now, in this new vector space $U_1 \otimes \cdots \otimes U_N$, such multilinear operations can be treated with usual linear algebra tools.

Definition 20 (Multilinearity)**.** *For $U_1, \ldots, U_N$ and W vector spaces and a field $\mathbb{K}$ (e.g. $\mathbb{R}$ or $\mathbb{C}$), a map $f : U_1 \times \cdots \times U_N \to W$ is said to be multilinear if*

$$f(\mathbf{x}_1, ..., \alpha\mathbf{x}_n + \beta\mathbf{y}_n, ..., \mathbf{x}_N) = \alpha f(\mathbf{x}_1, ..., \mathbf{x}_n, ..., \mathbf{x}_N) + \beta f(\mathbf{x}_1, ..., \mathbf{y}_n, ..., \mathbf{x}_N), \quad \forall n, \forall \alpha, \beta \in \mathbb{K}.$$

Let us consider the case of two vector spaces U and V for simplicity. Multilinearity now translates into bilinearity. A tensor product $\otimes$ defines a new vector space $U \otimes V$ and an operation $\otimes$ on vectors such that, for every bilinear map $f : U \times V \to W$, there is a corresponding linear map $g : U \otimes V \to W$ with $f(\mathbf{u}, \mathbf{v}) = g(\mathbf{u} \otimes \mathbf{v})$ for any vectors $\mathbf{u} \in U$, $\mathbf{v} \in V$ and $\mathbf{u} \otimes \mathbf{v} \in U \otimes V$.

The following theorem (proved in [Schwartz 1975, p4-5]) ensures the existence and uniqueness (up to an isomorphism) of one such operator. It provides a formal definition

of the tensor product for the case of two vector spaces (bilinearity), but it also holds for multiple spaces (multilinearity).

Theorem 8 (Tensor product [Schwartz 1975])**.** *If U and V are vector spaces, then:*

1. *(existence of the tensor product) there exists a vector space $U \otimes V$ and a bilinear map $\otimes : U \times V \to U \otimes V$ having the following property: for any vector space W and any bilinear map $f : U \times V \to W$, there exists a unique linear map $g : U \otimes V \to W$ such that $f(\mathbf{u}, \mathbf{v}) = g(\mathbf{u} \otimes \mathbf{v})$ for all $\mathbf{u} \in U$ and $\mathbf{v} \in V$.*
   One such bilinear map is called a ***tensor product*** *of U and V.*
2. *(uniqueness of the tensor product up to a composition with an isomorphism) If $\otimes' : U \times V \to U \otimes' V$ is another map with the same property, there exists a unique linear bijection $h : U \otimes V \to U \otimes' V$ such that $\otimes' = h \circ \otimes$ and $\otimes = h^{-1} \circ \otimes'$ (denoting $\circ$ the composition, not to be confused with the outer product denoted equally).*

A formal definition of tensors can be derived from the notion of tensor product.

Definition 21 (Tensor)**.** *A tensor is a vector of a tensor product space $(U \otimes V, \otimes)$, which is the vector space generated from U and V by a tensor product $\otimes$.*

To illustrate this abstract concept of tensor product, let us present a concrete example.

Example 1. *Take $U = \mathbb{R}^n$, $V = \mathbb{R}^m$ and $W = \mathbb{R}$. In this particular case, the outer product, denoted $\circ$, is a tensor product. This can be verified by showing that a generic bilinear map $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ can be associated to a linear map $g : \mathbb{R}^n \circ \mathbb{R}^m \to \mathbb{R}$. Since f is a bilinear form, it can be represented by a matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$, such that $\forall \mathbf{u} \in \mathbb{R}^n, \mathbf{v} \in \mathbb{R}^m$*

$$f(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T \mathbf{A} \mathbf{v} = \sum_{i,j} u_i A_{i,j} v_j$$

The corresponding map $g(\mathbf{u} \circ \mathbf{v})$ takes as input vectors $\mathbf{u} \circ \mathbf{v} \in \mathbb{R}^n \circ \mathbb{R}^m$ in the form

$$\mathbf{u} \circ \mathbf{v} = \mathbf{u}\mathbf{v}^T = \begin{bmatrix} u_1 v_1 & \dots & u_1 v_m \\ \vdots & \ddots & \vdots \\ u_n v_1 & \dots & u_n v_m \end{bmatrix}$$

$g(\cdot)$ can be exhibited as the following linear form

$$g(\mathbf{u} \circ \mathbf{v}) = \operatorname{tr}(\mathbf{A}^T(\mathbf{u} \circ \mathbf{v})) = \operatorname{vec}(\mathbf{A})^T \operatorname{vec}(\mathbf{u} \circ \mathbf{v}) = \sum_{i,j} A_{i,j} u_i v_j$$

Obviously, the tensor product is more general than the outer product since the spaces U and V need not be $\mathbb{R}^n$. In addition, the outer product is not the only possible tensor product in this setting. Another example is the Kronecker product between vectors:

$$\mathbf{u} \boxtimes \mathbf{v} = \begin{bmatrix} u_1\mathbf{v} \\ \vdots \\ u_n\mathbf{v} \end{bmatrix}$$

The corresponding linear form $g : \mathbb{R}^n \boxtimes \mathbb{R}^m \to \mathbb{R}$ is simply

$$g(\mathbf{u} \boxtimes \mathbf{v}) = \operatorname{vec}(\mathbf{A}^T)^T(\mathbf{u} \boxtimes \mathbf{v}) = \sum_{i,j} A_{i,j} u_i v_j$$

In conformation to Theorem 8 the two exhibited tensor products are isomorphic: $\operatorname{vec}((\mathbf{u} \circ \mathbf{v})^T) = \mathbf{u} \boxtimes \mathbf{v}$.

Naturally, not every bilinear map defined in $U \times V$ is a tensor product. The following result [Schwartz 1975, Corolary 1bis] provides a necessary and sufficient condition for one such map to be a tensor product.

Theorem 9. *A bilinear map $\boxdot$ from $U \times V$ to a vector space denoted $U \boxdot V$ is a tensor product if and only if there exist bases $(\mathbf{u}_i)_{i \in I}$, $(\mathbf{v}_j)_{j \in J}$ of U and V such that $(\mathbf{u}_i \boxdot \mathbf{v}_j)_{(i,j) \in I \times J}$ form a basis of $U \boxdot V$ (in which case it is true for any basis of U and V).*

In view of theorem 9, it is now easy to show that the outer product (example 2) and the Kronecker product (example 3) are tensor products, respectively in the space of finite 1-dimensional and 2-dimensional arrays.

Example 2 (Outer product)**.** *Let $U = \mathbb{R}^n$, $V = \mathbb{R}^m$ and consider the canonical basis $(\mathbf{e}_i)_{1 \leq i \leq n}$ of $\mathbb{R}^n$ and $(\mathbf{e}_j)_{1 \leq j \leq m}$ of $\mathbb{R}^m$, the outer product $\circ : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^{n \times m}$ gives: $(\mathbf{e}_i \circ \mathbf{e}_j) = \mathbf{E}_{i,j}$ where $\mathbf{E}_{i,j} \in \mathbb{R}^{n \times m}$ denotes a matrix with entry $E_{i,j} = 1$ and all other entries zero. This set $(\mathbf{E}_{i,j})_{1 \leq i \leq n,\ 0 \leq j \leq m}$ is clearly a basis of $\mathbb{R}^{n \times m}$.*

Example 3 (Kronecker product)**.** *The same can be done for the Kronecker product, with* $U = \mathbb{R}^{n_1 \times m_1}$ *and* $V = \mathbb{R}^{n_2 \times m_2}$ *and basis* $(\mathbf{E}_{i,j})_{ij}$ *of* $\mathbb{R}^{n \times m}$ *defined above. The Kronecker product* $\boxtimes : \mathbb{R}^{n_1 \times m_1} \times \mathbb{R}^{n_2 \times m_2} \to \mathbb{R}^{n_1 n_2 \times m_1 m_1}$ *gives:*
$(\mathbf{E}_{i,j} \boxtimes \mathbf{E}_{l,k}) = \mathbf{E}_{n_2(i-1)+k, m_2(j-1)+l}$*, which is a basis for the space* $\mathbb{R}^{n_1 n_2 \times m_1 m_2}$*.*

Two important corollaries can be derived from theorem 9.

Corollary 1 (Dimension of $U \otimes V$)**.** *If* U *and* V *have finite dimensions* $\dim(U)$*,* $\dim(V)$*, then* $\dim(U \otimes V) = \dim(U) \times \dim(V)$*.*

Corollary 2 (Decomposable tensors)**.** *The set* $\{\mathbf{u} \otimes \mathbf{v}, \mathbf{u} \in U, \mathbf{v} \in V\}$ *spans* $U \otimes V$*. Tensors that can be written as* $\mathbf{u} \otimes \mathbf{v}$ *are called decomposable, or rank-1 tensors.*

Note that the dimension of the resulting space is the product (not the sum) of the dimensions of the initial spaces. Corollary 2, in its turn, states the very important result that any tensor can be written as a sum of rank-1 tensors. However not all tensors are rank-1, and the problem of finding, if it exists, the smallest necessary number of rank-1 tensors to express a given tensor is a non-trivial problem. We will get back to this discussion in section 3.3. For now, let us see what this result implies to the Kronecker product as a tensor product.

Matrices as a sum of Kronecker products

The fact that the Kronecker product is a tensor product between two (or more) matrices – see example 3 – implies, from corollary 2, that matrices written as a Kronecker product $\mathbf{A} \boxtimes \mathbf{B}$ span the set of all matrices.

In other words, any matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$ can be expressed as a sum of Kronecker products $\mathbf{M} = \sum_i \mathbf{A}_i \boxtimes \mathbf{B}_i$ with $\mathbf{A}_i \in \mathbb{R}^{n_1 \times m_1}$, $\mathbf{B}_i \in \mathbb{R}^{n_2 \times m_2}$ and $n_1 n_2 = n$, $m_1 m_2 = m$.[1] Each term $\mathbf{A}_i \boxtimes \mathbf{B}_i$ can be seen as a rank-1 tensor in the tensor product space $\mathbb{R}^{n_1 \times m_1} \boxtimes \mathbb{R}^{n_1 \times m_1} = \mathbb{R}^{n_1 n_2 \times m_1 m_2} = \mathbb{R}^{n \times m}$. So, a Kronecker product matrix can be seen as rank-1 tensor in the space of $n \times m$ matrices.

In chapter 4 we provide another way of visualizing this same result, by exhibiting a suitable rearrangement operation that transforms a Kronecker product matrix into a rank-1 matrix (in the usual sense). The same rearrangement maps a generic matrix into a

1. Even if n and m are prime number, there is always the trivial (though useless in practice) solution of setting $n_1 = 1$, $n_2 = n$ and $m_1 = 1$, $m_2 = n$.

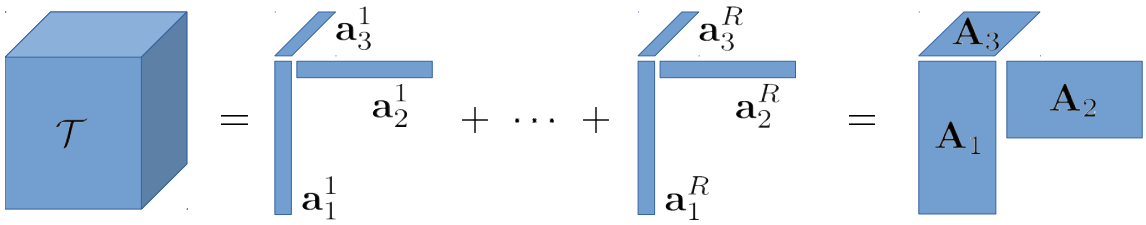


Figure 3.5 – Polyadic decomposition of a three-way array.

rank-R matrix, which is known to be written as a sum of R rank-1 matrices. Consequently, the original matrix can be written as the as a sum of R Kronecker products.

This result also holds for the Kronecker product of three or more matrices, by using the same reasoning and setting $\prod_k n_k = n$, $\prod_k m_k = m$.

3.3 Canonical Polyadic Decomposition

In 1927, [Hitchcock 1927] proposed the idea of the polyadic form of a tensor, i.e., expressing a order-N tensor as a linear combination of decomposable tensors:

$$\boldsymbol{\mathcal{T}} = \sum_{r=1}^{R} \mathbf{a}_1^r \circ \cdots \circ \mathbf{a}_N^r \tag{3.3}$$

In a more general formulation, the summing terms are written as $\mathbf{a}_1 \otimes \cdots \otimes \mathbf{a}_N$ which corresponds to the rank-1 tensors introduced in Corollary 2. Here we take the outer product as the standard tensor product between vectors. The vectors $\mathbf{a}_n^r$ may be assembled as the columns of a matrix $\mathbf{A}_n = [\mathbf{a}_n^1, \ldots, \mathbf{a}_n^R]$, which leads to the compact notation $\boldsymbol{\mathcal{T}} = [\![\mathbf{A}_1, \ldots, \mathbf{A}_N]\!]$. Figure 3.5 gives a visual representation of this decomposition in the third-order case.

Tensor rank The smallest value R for which (3.3) holds is called the tensor rank.

If a tensor is of order three or higher, the rank may depend on the field, in the sense that a real tensor of rank R may have smaller rank if we allow the decomposition to be complex. In this thesis, we will restrict ourselves to the field of real numbers.

This decomposition has later been reintroduced in the literature as CANDECOMP (canonical decomposition) by [Carroll & Chang 1970], PARAFAC (parallel factors) [Harshman 1970], not to mention other independent appearances of the concept in different

communities. If the number of rank-1 terms R is minimal, the decomposition is called canonical. For simplicity, we will refer to it in a simple abbreviate form as the CPD.[2]

From Corollary we know that any tensor can be decomposed in such a way. It doesn't imply, however, the uniqueness of the decomposition.

3.3.1 On the uniqueness of the CPD

First of all, notice that even a rank-1 tensor $\boldsymbol{\mathcal{T}}$ of order N is not uniquely represented by an outer product of N vectors since there remains a scaling and permutation indeterminacy between the factors. So, we are rather interested in the uniqueness of the rank-1 terms $\mathbf{a}_1 \circ \cdots \circ \mathbf{a}_N$. This is sometimes called *essential uniqueness*. As a simplification, we will simply write uniqueness while actually referring to essential uniqueness.

The CPD is *generally not unique*. A lot of research has been made to find special scenarios where there is uniqueness. Roughly speaking, the decomposition can be unique if R is small.

Probably the most general and well-known result on uniqueness was obtained by [Kruskal 1977] for three-way arrays and later extended by [Sidiropoulos & Bro 2000] to higher orders. It provides a sufficient condition for uniqueness by using the concept of *Kruskal rank* of a matrix $\mathbf{A}$, denoted $\kappa_{\mathbf{A}}$, which is the largest number κ such that any subset of κ columns of $\mathbf{A}$ are linearly independent[3]. For a N-mode array $\boldsymbol{\mathcal{T}}$ with decomposition $\boldsymbol{\mathcal{T}} = [\![\mathbf{A}_1, \ldots, \mathbf{A}_N]\!]$, a sufficient condition for uniqueness of the CPD is:

$$2R + N - 1 \leq \sum_{n=1}^{N} \kappa_{\mathbf{A}_n}$$

Later, [ten Berge & Sidiropoulos 2002] showed that this sufficient condition is also necessary for tensors of rank $R = 2$ and $R = 3$, but not for $R > 3$.

A necessary condition for uniqueness was provided by [Liu *et al.* 2001]:

$$R \leq \min_{n=1,\ldots,N} \left(\prod_{\substack{m=1 \\ m \neq n}}^{N} \operatorname{rank}(\mathbf{A}_m) \right)$$

Further recent results may be found, for instance, in [Domanov & De Lathauwer 2013a]

2. We call it CPD even when it is not calculated for the minimal rank, since, conveniently, it may stand for both Canonical Polyadic or CANDECOMP/PARAFAC decomposition.

3. We recall that the Kruskal rank is closely related to the spark (cf. definition 1): $\operatorname{spark}(\mathbf{A}) = \kappa_{\mathbf{A}} + 1$.

[Domanov & De Lathauwer 2013b]. They are sometimes much more powerful than Kruskal's bound.

In part II we propose an algorithm in which we are led to compute a CPD. Fortunately, we restrict ourselves to the low-rank regime, in which the CPD is known to behave better.

3.3.2 Computing the CPD

Given a tensor, there is no straightforward algorithm to determine its rank, i.e. the minimum number of rank-one components in the CP decomposition. The rank of a tensor is usually determined numerically by fitting various CP decompositions with different R.

In practice, rather than calculating exact decompositions, we are more often confronted to the problem of finding a good rank-R approximation of a given tensor. In the matrix case, the best rank-R approximation is known to be given by the SVD truncated to its first R terms [Eckart & Young 1936]. It is important to emphasize that, for higher-order arrays, there is no equivalent of one such result. As an illustration, the best rank-1 approximation of a cubic tensor might not be a factor in its best rank-2 approximation [Kolda 2001].

Now, let us focus on the case where the number of components is fixed beforehand. The goal is to compute a CPD with R components that best approximates $\boldsymbol{\mathcal{T}}$, i.e.,

$$\min_{\hat{\boldsymbol{\mathcal{T}}}} \|\boldsymbol{\mathcal{T}} - \hat{\boldsymbol{\mathcal{T}}}\|_F \quad \text{with} \quad \hat{\boldsymbol{\mathcal{T}}} = \sum_{r=1}^{R} \mathbf{a}_1^r \circ \cdots \circ \mathbf{a}_N^r = [\![\mathbf{A}_1, \ldots, \mathbf{A}_n]\!]$$

The most widespread algorithm for addressing this problem is a very simple alternating least squares (ALS) method proposed in [Carroll & Chang 1970] and [Harshman 1970]. The ALS approach iteratively updates each of the factors $\mathbf{A}_n$ by fixing all other terms, until convergence. Having fixed all but one matrix, the problem reduces to a linear least-squares problem, which can be solved with a simple Moore-Penrose pseudoinverse. We do not provide the explicit update step of the ALS procedure because it requires the notion of Katri-Rao product, which we didn't define here. The full algorithm can be found for instance in [Kolda & Bader 2009, Figure 3.3].

This algorithm may be considerably sensitive to the initialization and is not guaranteed to converge to a global minimum. Some techniques for improving the efficiency of ALS are discussed in [Comon *et al.* 2009]. We refer the reader to [Kolda & Bader 2009, Section 3.4] for a brief overview of other existing algorithms, and [Tomasi & Bro 2006] for an experimental comparison of different algorithms.

3.4 Other tensor decomposition

Let us make a connection with the SVD of a rank-R matrix $\mathbf{M}$:

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T. \tag{3.4}$$

It is characterized by two features: 1) $\mathbf{U} \in \mathbb{R}^{I\times R}$ and $\mathbf{V} \in \mathbb{R}^{J\times R}$ have orthonormal columns, and 2) $\mathbf{\Sigma} \in \mathbb{R}^{R\times R}$ is diagonal. We will see that it is not possible to directly generalize this decomposition to higher-order arrays while preserving both of these features.

Alternatively, one can express the SVD as a sum of rank-1 matrices:

$$\mathbf{M} = \sum_{r=1}^{R} \sigma_r \mathbf{u}^r \circ \mathbf{v}^r \tag{3.5}$$

where $\sigma_r = \Sigma_{r,r}$ is the r-th term in the diagonal of $\mathbf{\Sigma}$, and $\mathbf{u}^r$ and $\mathbf{v}^r$ are the r-th column of the matrices $\mathbf{U}$ and $\mathbf{V}$ respectively.

Note that the CPD of an N-way array $\boldsymbol{\mathcal{T}}$ can be expressed in a similar way by imposing the vectors $\mathbf{a}_n^r$ to be of unit-norm and introducing the scaling factors $\lambda_r \in \mathbb{R}$:

$$\boldsymbol{\mathcal{T}} = \sum_{r=1}^{R} \lambda_r \mathbf{a}_1^r \circ \cdots \circ \mathbf{a}_N^r \tag{3.6}$$

The difference between (3.5) and (3.6) is that, in the latter, there is no orthogonality restriction between the vectors $\{\mathbf{a}_n^r\}_{r=1}^R$ (i.e. matrices $\mathbf{A}_n$ are not constrained to have orthonormal columns.

It is also interesting to rewrite the CPD in a different way, to make a connection with other tensor decompositions:

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}_1 \cdots \times_N \mathbf{A}_N \quad \text{with} \quad \boldsymbol{\mathcal{G}} \text{ diagonal (i.e. } G_{i,j,k,l} \neq 0 \iff i=j=k=l) \tag{3.7}$$

where $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R\times\cdots\times R}$ is called the *core tensor*. We can compactly denote $\boldsymbol{\mathcal{T}} = [\![\boldsymbol{\mathcal{G}}; \mathbf{A}_1, \ldots, \mathbf{A}_N]\!]$. By removing the diagonality constraint on the core tensor, we obtain the more general *Tucker decomposition*.

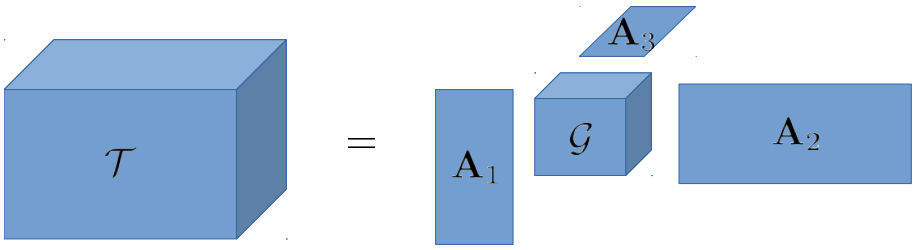


Figure 3.6 – Tucker decomposition of a three-way array.

3.4.1 Tucker decomposition

Tucker decomposition [Tucker 1966] decomposes a tensor into a core tensor multiplied (or transformed) by a matrix along each mode. For an N-way tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ it gives:

$$\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}_1 \cdots \times_N \mathbf{A}_N = \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} G_{r_1,\ldots,r_N} \mathbf{a}_1^{r_1} \circ \cdots \circ \mathbf{a}_N^{r_N} = [\![\boldsymbol{\mathcal{G}}; \mathbf{A}_1, \ldots, \mathbf{A}_N]\!] \quad (3.8)$$

where $\mathbf{A}_n \in \mathbb{R}^{I_n \times r_n}$ are the factor matrices and can be thought of as the principal components in each mode. The core tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{R_1,\ldots,R_N}$ shows the level of interaction between the different components. See Figure 3.6 for a visual representation.

Getting back to the analogy with the SVD: if we impose both the diagonality of $\boldsymbol{\mathcal{G}}$ and orthogonality between the columns of matrices $\mathbf{A}_n$, the number of free parameters becomes smaller than the number of equations, then there is generally no solution [Comon 2014].

That's why, when extending the SVD to higher orders, we have to choose one of the constraints. When we constrain $\boldsymbol{\mathcal{G}}$ to be diagonal (with $R_1 = \cdots = R_N = R$), but relax the orthogonality constraint on factor matrices, we obtain the CPD. If, on the other hand, we keep the orthonormality constraint, but allow $\boldsymbol{\mathcal{G}}$ to have nonzero entries outside its diagonal, we obtain the so-called HOSVD (Higher-order SVD) [De Lathauwer *et al.* 2000].

3.4.2 Higher-Order SVD

HOSVD is equivalent to the Tucker decomposition (3.8) when constraining the factor matrices $\mathbf{A}_n$ to have orthonormal columns.[4]

The dimensions of the core tensor $(R_1, \ldots, R_N)$ are related to the concepts of *n-rank* and *multilinear rank*.

4. Strictly speaking, they are semi-orthogonal matrices. A non-square $(n \times m)$ matrix $\mathbf{A}$ with $n > m$ is semi-orthogonal if $\mathbf{A}^T\mathbf{A} = \mathbf{I}_n$.

Definition 22 (n-rank)**.** *The mode-n rank, or simply n-rank, R_n of a tensor $\boldsymbol{\mathcal{T}}$ is the column rank of its n-unfolding matrix $\mathbf{T}_{(n)}$.*

Definition 23 (Multilinear rank)**.** *The N-uple of n-ranks $(R_1, \ldots, R_N)$ is the multilinear rank of a N-way tensor $\boldsymbol{\mathcal{T}}$.*

For an exact decomposition, the dimensions of the core tensor may be imposed to be $(R_1 \times R_2 \times R_3)$, where R_n is the n-rank of the decomposed tensor. In addition, the n-rank cannot exceed the tensor rank R nor the nth dimension I_n, i.e. $R_n < R$ and $R_n < I_n$, $\forall n$.

A typical application of Tucker and HOSVD is data compression, see for instance [Vasilescu & Terzopoulos 2003] [Wang & Ahuja 2004]. That is because, in many cases, the storage for the decomposed version of the tensor can be significantly less costly than for the original tensor.

3.4.3 More decompositions

There exist a great variety of tensor decompositions, more or less related to Tucker and CPD. We will just very briefly cite some of them without getting further into the details.

The Tucker2 decomposition [Tucker 1966] of a third-order array sets one of the factor matrices in the Tucker decomposition to be the identity matrix: $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{G}} \times_1 \mathbf{A}_1 \times_2 \mathbf{A}_2 = [\![\boldsymbol{\mathcal{G}}; \mathbf{A}_1, \mathbf{A}_2, \mathbf{I}]\!]$.

PARAFAC2 [Harshman 1972] can be seen as a relaxed variant of CP that can be applied to a collection of matrices that each have the same number of columns but a different number of rows, say $\{\mathbf{M}_k\}_{k=1}^{K}$. PARAFAC2 performs an SVD of each of the matrices $\mathbf{M}_k$ with the additional constraint that the right factor $\mathbf{V}$ is shared among all matrices: $\mathbf{M}_k = \mathbf{U}_k \boldsymbol{\Sigma}_k \mathbf{V}$. It can also be applied to three-way tensors as a particular case where all matrices have the same number of rows.

Other decompositions include: CANDELINC [Carroll *et al.* 1980], DEDICOM [Harshman 1978], and PARATUCK2 [Harshman & Lundy 1996]. We refer the reader to [Kolda & Bader 2009] for a more detailed survey.

Part II

Fast Structured Dictionaries

Chapter 4

Approximation by Kronecker-structured matrices

In this chapter, we introduce the Kronecker structure and its relation to multidimensional data. We also address the problem of approximating a general linear operator as a sum of Kronecker products.

This chapter is partly based on the article [Dantas *et al.* 2018].

4.1 Kronecker structure and multidimensional data

Multi-dimensional data arise in a large variety of applications such as telecommunications, biomedical sciences, image and video processing to name a few [Kolda & Bader 2009].

In most machine learning and data processing techniques, however, input data are assumed to be vectors. Consider, for instance, the Dictionary Learning problem introduced in section 1.5, in which input data samples are given by vectors $\mathbf{y}_i$ arranged as the columns of a data matrix $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]$. Similar formulations are commonly adopted in related learning approaches. When the targeted data is multidimensional – say, images as 2-dimensional data or color and hyperspectral images as 3-D data examples – it is usual to perform a vectorization of each data sample to conform them to the vectorial framework, as illustrated in Figure 4.1.

Explicitly accounting for this tensorial structure of the data can be more advantageous than relying on its vectorized version. It avoids losing the original neighboring relations and inter-mode correlations, besides providing a more efficient and economic representation and subsequent processing.

The Kronecker product structure arises naturally when dealing with multi-dimensional (tensorial) data, since it manages to recover the underlying tensorial nature of vectorized data samples. Indeed, when applied to a vectorized tensor, each composing factor of a

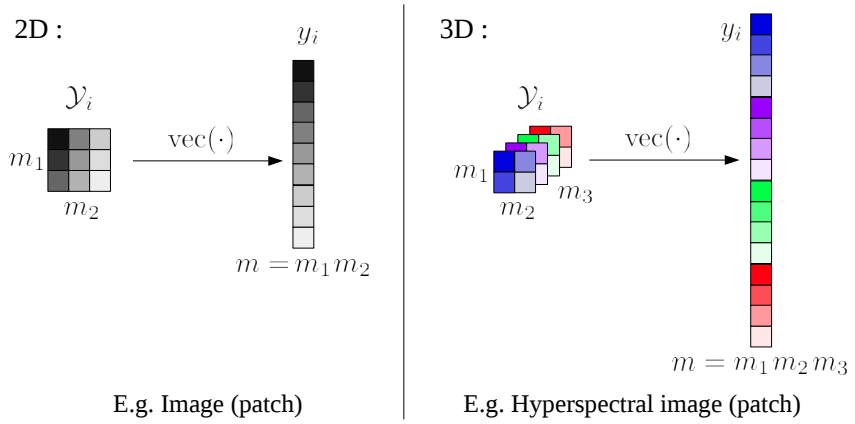


Figure 4.1 – Input data samples are usually vectorized.

Kronecker-structured linear operator acts independently on each mode of the data (cf. equation (3.1)). For this reason, such operators are commonly called *separable operators*. A widespread example is the 2D-DCT, which is defined as a composition via Kronecker product of two 1D-DCTs. The same argument applies to the 3D-DCT. Conveniently, such operators are more compact to store and can be applied much more efficiently in computational terms than their unstructured counterpart.

Nevertheless, relatively little attention has been paid to exploiting this type of structure on representation learning methods such as dictionary learning algorithms, which aim at obtaining a set of explanatory variables (the dictionary) capable of sparsely approximating an input dataset. Conventional methods impose no particular structure to the dictionary matrix learned from the vectorized input samples, therefore completely disregarding a potential multi-dimensional characteristic of the data.

Here, we propose a broader class of structured operators which has the separable structure as a special case. We generalize the Kronecker structure by allowing matrices to be written as a *sum* of Kronecker products:

$$\mathbf{D} = \sum_{r=1}^{R} \mathbf{D}_1^r \boxtimes \cdots \boxtimes \mathbf{D}_K^r = \sum_{r=1}^{R} \bigboxtimes_{k=1}^{K} \mathbf{D}_k^r, \tag{4.1}$$

which we refer to as a *rank-R K-Kronecker-structured* matrix (or simply (R, K)-KS and even K-KS when $R = 1$). In [Tsiligkaridis & Hero 2013], where a particular case of this model (with $K = 2$) was proposed (see section 4.1.2 for more details), R is referred to as the *separation rank*. As we will see in section 4.1.1, this structure may lead to significant

memory and computational savings when compared to an unstructured matrix. At the same time, because we allow sums of several Kronecker terms, we make the structure more flexible thus increasing its approximation capabilities with respect to the conventional Kronecker structure (i.e. ordinary separable operators).

Actually, provided that the rank R can be set large enough, any matrix $\mathbf{D}$ may be written as a (R,K)-KS matrix [Tsiligkaridis & Hero 2013]. This point has also been discussed in section 3.2.3: it follows from Corollary 2 that matrices written as Kronecker products ($R = 1$) can be seen as rank-1 tensors in the tensor space generated by the Kronecker product and, as such, they span the entire space of matrices of size $(\prod_k n_k \times \prod_k m_k)$.

In this chapter, we provide a novel method to approximate general linear operators as a sum of Kronecker products. To this end, we draw a parallel between the problem of approximating an arbitrary linear operator by a Kronecker-structured one and the low-rank tensor approximation problem. Later, in chapter 5, this parallel is proved useful in the dictionary learning context. That's why in this chapter we choose to denote $\mathbf{D}$ the matrices to be approximated.

4.1.1 Motivations

The quest for Kronecker-structured linear operators has various motivations besides the suitability to multi-dimensional signals:

1. reduced computational complexity;
2. diminished memory requirements and
3. smaller sample complexity on learning applications.

Computational complexity

The complexity savings on matrix-vector multiplications are explained as follows. For an $(n \times m)$ K-KS matrix $\mathbf{D} = \mathbf{D}_K \boxtimes \cdots \boxtimes \mathbf{D}_1$ with factors $\mathbf{D}_k \in \mathbb{R}^{n_k \times m_k}$, $n = \prod_{k=1}^{K} n_k$ and $m = \prod_{k=1}^{K} m_k$, the matrix-vector product $\mathbf{Dx}$ can be rewritten as

$$\mathbf{y} = (\mathbf{D}_K \boxtimes \cdots \boxtimes \mathbf{D}_1)\mathbf{x} \qquad \rightarrow \qquad \boldsymbol{\mathcal{Y}} = \boldsymbol{\mathcal{X}} \times_1 \mathbf{D}_1 \times_2 \cdots \times_K \mathbf{D}_K \tag{4.2}$$

where $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{n_1 \times \cdots \times n_K}$ and $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{m_1 \times \cdots \times m_K}$ are the tensorized versions of $\mathbf{y} \in \mathbb{R}^{n_1 \cdots n_K}$ and $\mathbf{x} \in \mathbb{R}^{m_1 \cdots m_K}$ respectively (cf. section 3.1.2) and $\times_k$ denotes the mode-k tensor matrix

product. In other words, it comes down to multiplying each factor by the corresponding mode on the tensorized version $\boldsymbol{\mathcal{X}}$ of the vector $\mathbf{x}$.

Since the composing factors $\mathbf{D}_k$ are much smaller than $\mathbf{D}$, the total complexity for computing (4.2) can be significantly smaller than the usual $\mathcal{O}(nm)$. In particular, when the factors are all square (i.e. $n_k = m_k \ \forall k$) the total number of operations is given by $(\prod_{k=1}^{K} n_k)(\sum_{k=1}^{K} n_k)$ [Tadonki & Philippe 2001] compared to $(\prod_{k=1}^{K} n_k)^2$ operations for an unstructured matrix of the same size. Note that a set of products $\prod_k n_k$ in the latter is replaced by a summation $\sum_k n_k$ in former, which is a significant complexity reduction.

For a sum of R Kronecker products, the mentioned complexity is simply multiplied by R, since we now have a sum of R terms similar to (4.2):

$$\mathbf{y} = \sum_{r=1}^{R} (\mathbf{D}_K^r \boxtimes \cdots \boxtimes \mathbf{D}_1^r)\mathbf{x} \qquad \rightarrow \qquad \boldsymbol{\mathcal{Y}} = \sum_{r=1}^{R} \boldsymbol{\mathcal{X}} \times_1 \mathbf{D}_1^r \times_2 \cdots \times_K \mathbf{D}_K^r. \tag{4.3}$$

In addition, as the summing terms are independent from each other, they can be calculated in parallel. This potential parallelization can even further accelerate this computation.

Storage cost

The total storage cost for the structured operator is proportional to $(\sum_{k=1}^{K} n_k m_k)$ instead of $(\prod_{k=1}^{K} n_k m_k)$.

Indeed, a non-structured $(n \times m)$ matrix, with $n = \prod_{k=1}^{K} n_k$ and $m = \prod_{k=1}^{K} m_k$, has a total of nm parameters. On the other hand, a (R, K)-KS matrix of the same size is determined by the smaller matrices $\mathbf{D}_k^r \in \mathbb{R}^{n_k \times m_k}$. The total number of parameters is now precisely $R(\sum_{k=1}^{K} n_k m_k)$. Once again, products are exchanged against sums, which leads to considerable reductions in memory requirements as long as R remains relatively small.

Sample complexity

Similarly, recent studies on the minimax risk for the dictionary identifiability problem showed that the necessary number of samples for reliable reconstruction, up to a given mean squared error, of a K-KS dictionary within its local neighborhood scales with $(m \sum_{k=1}^{K} n_k m_k)$ [Shakeri *et al.* 2016, Shakeri *et al.* 2017a, Shakeri *et al.* 2018] compared to $(m \prod_{k=1}^{K} n_k m_k)$ for unstructured dictionaries of the same size [Jung *et al.* 2016].

In parallel to these lower bounds on the minimax risk, similar results have been obtained on upper bounds for the sample complexity. In particular, [Shakeri *et al.* 2017b]

[Shakeri *et al.* 2017c] show that the sufficient number of samples to recover an underlying K-KS dictionary with high probability up to a given estimation error scales with $\max_{k\in\{1,\dots K\}}(n_k m_k^3)$, compared to $(\prod_{k=1}^{K} n_k m_k^3)$ for an unstructured dictionary [Gribonval *et al.* 2015a]. A comprehensive overview of sample complexity in Kronecker-structured dictionaries compared to their unstructured counterparts is provided in [Shakeri *et al.* 2019].

4.1.2 Related Work

Kronecker-structured transforms are somewhat standard in the signal processing community [Mallat 2008, section 7.7], and particularly in the Dictionary Learning literature, though typically only the rank one (separable) case is considered. The Kronecker structure was introduced in Dictionary Learning by [Hawe *et al.* 2013, Roemer *et al.* 2014] both addressing only 2-dimensional data (i.e. 2-KS dictionaries). The model was extended to the 3rd-order (3-KS dictionaries) [Zubair & Wang 2013, Peng *et al.* 2014] and even for an arbitrary tensor order [Caiafa & Cichocki 2013, Ghassemi *et al.* 2017] based on the Tucker decomposition, a model coined as Tucker Dictionary Learning. However, none of these works include a sum of Kronecker terms. Even though the formulation in [Ghassemi *et al.* 2017] would allow it, they restrict their analysis to the rank-one ($R = 1$) case.

The sum of Kronecker products model was initially explored in the covariance estimation community [Tsiligkaridis & Hero 2013, Bijma *et al.* 2005] and was recently used in [Dantas *et al.* 2017] as an extension of the existing Kronecker-structured dictionaries. Once again, these works addressed only the 2nd-order case ($K = 2$). In this thesis we extend this model to an arbitrary tensor order, thus allowing for both $R \geq 1$ and $K \geq 2$. An advantage of our approach w.r.t [Dantas *et al.* 2017] and [Ghassemi *et al.* 2017] is that here we can choose the desired number of summing terms beforehand without needing to empirically adjust a regularization parameter.

Finally, similarly to what is introduced in this manuscript, [Batselier & Wong 2017] have recently discussed factorizations strategies for (R,K)-KS matrices, but with additional orthogonality constraints which are not considered here.

Notation reminder We denote $\boxtimes$ the Kronecker Product and $\circ$ the outer product. Matrices (resp. tensors) are represented by bold uppercase (resp. caligraphic) letters and the (i, j)th entry of a matrix $\mathbf{D}$ is denoted $d(i, j)$. The vectorization operation, denoted $\text{vec}(\cdot)$, consists in stacking the columns of a matrix and $\text{unvec}_{n,m}(\cdot)$ stands for the converse operation, for a resulting $(n \times m)$ matrix.

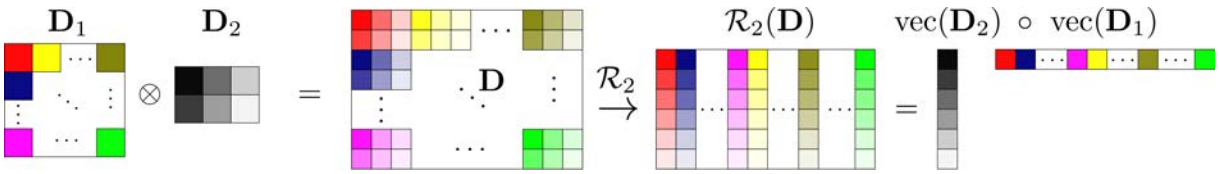


Figure 4.2 – Rearrangement operation

4.2 Rearrangement: from Kronecker to low-rank

This section introduces a rearrangement from a matrix to a multidimensional array that links the inference of the Kronecker structure for matrices to the low-rank approximation problem for multidimensional arrays *i.e.* higher-order tensors.

4.2.1 The second order case

Consider a matrix $\mathbf{D} \in \mathbb{R}^{n_1 n_2 \times m_1 m_2}$ such that $\mathbf{D} = \mathbf{D}_1 \boxtimes \mathbf{D}_2$, where $\mathbf{D}_1 \in \mathbb{R}^{n_1 \times m_1}$ and $\mathbf{D}_2 \in \mathbb{R}^{n_2 \times m_2}$. Then one can define an operator $\mathcal{R}_2 : \mathbb{R}^{n_1 n_2 \times m_1 m_2} \to \mathbb{R}^{n_2 m_2 \times n_1 m_1}$ which rearranges the elements of $\mathbf{D}$ in such way that the output $\mathbf{D}^\pi = \mathcal{R}_2(\mathbf{D})$ is a rank-1 matrix [Van Loan & Pitsianis 1993] given by

$$\mathbf{D}^\pi = \mathcal{R}_2(\mathbf{D}) = \mathrm{vec}(\mathbf{D}_2)\,\mathrm{vec}(\mathbf{D}_1)^T = \mathrm{vec}(\mathbf{D}_2) \circ \mathrm{vec}(\mathbf{D}_1) \tag{4.4}$$

The rearrangement consists in vectorizing the jth-block of size $(n_2 \times m_2)$ in $\mathbf{D}$ to form the jth-column of $\mathcal{R}(\mathbf{D})$, running through the blocks column-wise. This process is illustrated in Figure 4.2 and Example 4.

Example 4 (Rearrangement)**.** *We illustrate this result with a simple example, with* $n_1 = n_2 = m_1 = m_2 = 2$*. Consider the* (2×2) *matrices*

$$\mathbf{D}_1 = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}, \qquad \mathbf{D}_2 = \begin{bmatrix} a & c \\ b & d \end{bmatrix}.$$

The resulting Kronecker product is given by

$$\mathbf{D} = \mathbf{D}_1 \boxtimes \mathbf{D}_2 = \left[\begin{array}{c:c} d_{1,1}\mathbf{D}_2 & d_{1,2}\mathbf{D}_2 \\ \hdashline d_{2,1}\mathbf{D}_2 & d_{2,2}\mathbf{D}_2 \end{array}\right] = \left[\begin{array}{cc:cc} 1a & 1c & 3a & 3c \\ 1b & 1d & 3b & 3d \\ \hdashline 2a & 2c & 4a & 4c \\ 2b & 2d & 4b & 4d \end{array}\right]$$

and its rearranged version becomes

$$\mathcal{R}(\mathbf{D}) = \begin{bmatrix} 1a & 2a & 3a & 4a \\ 1b & 2b & 3b & 4b \\ 1c & 2c & 3c & 4c \\ 1d & 2d & 3d & 4d \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix},$$

which is a rank-1 matrix.

Since the isomorphism $\mathcal{R}_2$ is linear, when applied to a $(R,2)$-KS matrix, it outputs a matrix $\mathbf{D}^\pi = \sum_{r=1}^{R} \operatorname{vec}(\mathbf{D}_2^r) \circ \operatorname{vec}(\mathbf{D}_1^r)$ which is of rank at most R.

$$\mathbf{D}^\pi = \mathcal{R}_2\left(\sum_{r=1}^{R} \mathbf{D}_1^r \boxtimes \mathbf{D}_2^r\right) = \sum_{r=1}^{R} \mathcal{R}_2\left(\mathbf{D}_1^r \boxtimes \mathbf{D}_2^r\right) = \sum_{r=1}^{R} \operatorname{vec}(\mathbf{D}_2^r) \circ \operatorname{vec}(\mathbf{D}_1^r)$$

4.2.2 Generalizing to higher order

Now, consider a K-KS matrix $\mathbf{D} \in \mathbb{R}^{n_1 n_2 \ldots n_K \times m_1 m_2 \ldots m_K}$ given by

$$\mathbf{D} = \mathbf{D}_1 \boxtimes \cdots \boxtimes \mathbf{D}_K = \bigboxtimes_{k=1}^{K} \mathbf{D}_k \tag{4.5}$$

with $\mathbf{D}_k \in \mathbb{R}^{n_k \times m_k}$ for $k \in \{1, \ldots, K\}$.

We can generalize the rearrangement operator $\mathcal{R}_2$ (for 2-KS matrices) to an operator $\mathcal{R}_K : \mathbb{R}^{n_1 n_2 \ldots n_K \times m_1 m_2 \ldots m_K} \to \mathbb{R}^{n_K m_K \times \cdots \times n_1 m_1}$ (for K-KS matrices) in a recursive manner. For this, let us consider $\mathcal{R}_{K-1}$ known, such that

$$\mathbf{D} = \bigboxtimes_{k=2}^{K} \mathbf{D}_k \qquad \Rightarrow \qquad \underline{\boldsymbol{\mathcal{D}}}^\pi = \mathcal{R}_{K-1}(\mathbf{D}) = \operatorname{vec}(\mathbf{D}_K) \circ \cdots \circ \operatorname{vec}(\mathbf{D}_2), \tag{4.6}$$

which outputs a $(K-1)$th-order rank-1 tensor for an input $(K-1)$-KS matrix, and try to define $\mathcal{R}_K$ from it.

Note that we can rewrite $\mathbf{D} = \mathbf{D}_1 \boxtimes (\mathbf{D}_2 \boxtimes \cdots \boxtimes \mathbf{D}_K)$ which means that $\mathbf{D}$ is composed

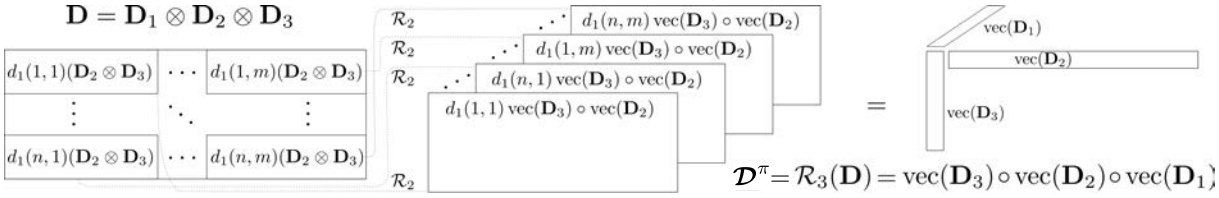


Figure 4.3 – Rearrangement operation $\mathcal{R}_K$ for $K=3$.

of n_1 by m_1 blocks given by $d_1(i,j)\,(\mathbf{D}_2 \boxtimes \cdots \boxtimes \mathbf{D}_K)$ in which we can directly apply $\mathcal{R}_{K-1}$. If we again run through all these blocks columnwise applying $\mathcal{R}_{K-1}$ and stacking the resulting $(K-1)$th-order tensors along dimension K, we will obtain a (K)th-order rank-1 tensor given by:

$$\boldsymbol{\mathcal{D}}^{\pi} = \mathcal{R}_K(\mathbf{D}) = (\mathrm{vec}(\mathbf{D}_K) \circ \cdots \circ \mathrm{vec}(\mathbf{D}_2)) \circ \mathrm{vec}(\mathbf{D}_1)$$

Therefore, we can define a recursive algorithm to calculate $\mathcal{R}_K$ which has $\mathcal{R}_2$ defined in section 4.2.1 as a base case. Actually, we can go even further and decompose $\mathcal{R}_2$ recursively in the exact same way, in which case the base case $\mathcal{R}_1$ becomes a simple vectorization operation, leading to Algorithm 7 below. We denote $\mathrm{cat}(\mathbf{A}, \mathbf{B}, K)$ the concatenation of $\mathbf{A}$ and $\mathbf{B}$ along the dimension K and if one of the entries is empty $\mathrm{cat}([\,], \mathbf{B}, K)$ simply gives the remaining entry $\mathbf{B}$.

The described procedure is illustrated in Figure 4.3 for the particular case of $K=3$.

Algorithm 7 $\mathcal{R}_K(\mathbf{D}) = \mathrm{Rearrange}(\mathbf{D}, \mathbf{n} = [n_1, \ldots, n_K], \mathbf{m} = [m_1, \ldots, m_K])$

$K = \mathrm{length}(\mathbf{n})$
if $K = 1$ **then** ▷ Base case
  return $\mathrm{vec}(\mathbf{D})$
else
  $\mathcal{R}_K(\mathbf{D}) = [\,]$; ▷ Initialize as an empty object
  $n_{\mathrm{rows}} = \prod_{k=2}^{K} n_k$
  $m_{\mathrm{cols}} = \prod_{k=2}^{K} m_k$ ▷ Size of the blocks
  for $i = 1 : m_1$ **do** ▷ Go over all blocks on the matrix.
    for $j = 1 : n_1$ **do**
      $\mathbf{D}_{\mathrm{block}} = \mathbf{D}((i-1)n_{\mathrm{rows}} + (1 : n_{\mathrm{rows}}), (j-1)m_{\mathrm{cols}} + (1 : m_{\mathrm{cols}}))$
      $\mathcal{R}_K(\mathbf{D}) = \mathrm{cat}(\mathcal{R}_K(\mathbf{D}), \mathrm{Rearrange}(\mathbf{D}_{\mathrm{block}}, \mathbf{n}(2:K), \mathbf{m}(2:K)), K)$
    end for
  end for
  return $\mathcal{R}_K(\mathbf{D})$
end if

Just like in section 4.2.1, note that for an input (R, K)-KS matrix, the rearrangement outputs a sum of R rank-1 tensors with factors $\text{vec}(\mathbf{D}_i)$ sorted in the reverse lexicographic order:

$$\mathbf{D} = \sum_{r=1}^{R} \bigboxtimes_{k=1}^{K} \mathbf{D}_k^r \qquad \Rightarrow \qquad \boldsymbol{\mathcal{D}}^\pi = \sum_{r=1}^{R} \text{vec}(\mathbf{D}_K^r) \circ \cdots \circ \text{vec}(\mathbf{D}_1^r). \tag{4.7}$$

4.2.3 Inverse rearrangement

The inverse rearrangement $\mathcal{R}_K^{-1}$ consists simply in switching the input and output indexes of the previous operator so to yield $\mathbf{D} = \mathcal{R}_K^{-1}(\boldsymbol{\mathcal{D}}^\pi)$. In practical terms, the resulting recursive algorithm consists in progressively reconstructing the blocks of $\mathbf{D}$ from their vectorized versions in $\boldsymbol{\mathcal{D}}^\pi$. The base case is now a matricization (unvec) operation.

4.3 From SVD to CPD

Consider $\mathbf{D} \in \mathbb{R}^{n \times m}$ and the following constrained approximation problem:

$$\min_{\hat{\mathbf{D}} \in \mathcal{C}_K^R} \|\hat{\mathbf{D}} - \mathbf{D}\|_F^2 \tag{4.8}$$

where $\mathbf{D} \in \mathbb{R}^{n \times m}$ and $\mathcal{C}_K^R$ denotes the set of all (R, K)-KS matrices of size $(n \times m)$.

With the help of the rearrangement operators defined in section 4.2 and for $\{n_k, m_k\}_{k=1}^K$ known[1], the task of approximating any given matrix $\mathbf{D}$ comes down to a low-rank approximation of the Kth-order rearranged tensor $\boldsymbol{\mathcal{D}}^\pi = \mathcal{R}_K(\mathbf{D})$.

When the targeted structure is a (sum of) 2-Kronecker product(s), i.e. $K = 2$, we are interested in the low-rank approximation of $\mathbf{D}^\pi = \mathcal{R}_2(\mathbf{D})$ which is still a matrix (2nd-order tensor). This is easily achieved – and also optimally, from Eckart-Young theorem [Eckart & Young 1936] – via the SVD.

The task gets harder if we want to generalize to a (sum of) K-Kronecker product(s), since a low-rank approximation of an order-K tensor $\boldsymbol{\mathcal{D}}^\pi = \mathcal{R}_K(\mathbf{D})$ is required. In this work, we propose to use a Canonical Polyadic Decomposition (CPD) [Kolda & Bader 2009]

1. Otherwise, it would necessary to also optimize wit respect to the dimensions of the factors (n_k, m_k). This is an interesting perspective that we did not explore in this thesis.

to approximate the tensor $\boldsymbol{\mathcal{D}}^\pi$ with a sum of R rank-one tensors.

$$\{\hat{\mathbf{d}}_k^r\}_{k=\{1,\dots,K\}}^{r=\{1,\dots,R\}} = \text{CPD}(\boldsymbol{\mathcal{D}}^\pi, R) \qquad \text{such that} \qquad \hat{\boldsymbol{\mathcal{D}}}^\pi = \sum_{r=1}^{R} \hat{\mathbf{d}}_K^r \circ \cdots \circ \hat{\mathbf{d}}_1^r \tag{4.9}$$

By comparing equations (4.9) and (4.7), we can see that each composing factor $\hat{\mathbf{D}}_k^r$ can be obtained from the corresponding vector $\hat{\mathbf{d}}_k^r$ through a simple matricization operation: $\hat{\mathbf{D}}_k^r = \text{unvec}_{n_k,m_k}(\hat{\mathbf{d}}_k^r)$. The resulting approximation is thus a (R, K)-KS matrix with factors $\hat{\mathbf{D}}_k^r$.

The proposed procedure to obtain $\hat{\mathbf{D}}$ and its factors $\hat{\mathbf{D}}_k^r$ is summarized in Algorithm 8, which we call HO-SuKro (**H**igher **O**rder **Su**m of **Kro**neckers) Approximation algorithm. It is parametrized by the targeted rank R and two vectors $\mathbf{n} = [n_1, \dots, n_K]$ and $\mathbf{m} = [m_1, \dots, m_K]$ with $(n_k \times m_k)$ being the dimensions of the k-th factor $\hat{\mathbf{D}}_k^r$ ($\forall r \in \{1, \dots, R\}$), such that $n = \prod_{k=1}^K n_k$ and $m = \prod_{k=1}^K m_k$.

Algorithm 8 $\left[\hat{\mathbf{D}}\ ,\ \{\hat{\mathbf{D}}_k^r\}_{k=\{1,\dots,K\}}^{r=\{1,\dots,R\}}\right] = \text{HO-SuKroApprox}(\mathbf{D}, R, \mathbf{n}, \mathbf{m})$

1: $\boldsymbol{\mathcal{D}}^\pi = \mathcal{R}_K(\mathbf{D}) = \text{Rearrange}(\mathbf{D}, \mathbf{n}, \mathbf{m})$ ▷ Rearranging input matrix
2: $\{\hat{\mathbf{d}}_k^r\}_{k=\{1,\dots,K\}}^{r=\{1,\dots,R\}} = \text{CPD}(\boldsymbol{\mathcal{D}}^\pi, R)$ ▷ CPD on rearranged tensor
3: $\{\hat{\mathbf{D}}_k^r\}_{k=\{1,\dots,K\}}^{r=\{1,\dots,R\}} = \text{unvec}_{n_k,m_k}(\hat{\mathbf{d}}_k^r)$ ▷ Recovering factors $\hat{\mathbf{D}}_k^r$
4: **return** $\left[\hat{\mathbf{D}} = \sum_{r=1}^R \hat{\mathbf{D}}_1^r \boxtimes \cdots \boxtimes \hat{\mathbf{D}}_K^r\ ,\ \{\hat{\mathbf{D}}_k^r\}_{k=\{1,\dots,K\}}^{r=\{1,\dots,R\}}\right]$

For the computation of the CPD we use a standard implementation available in the Tensorlab 3.0 toolbox [Vervliet *et al.* 2016]. It is important to emphasize that the rank approximation problem is ill-posed on tensors [de Silva & Lim 2008] and has no direct solution. One thus needs to resort to iterative approaches (as mentioned in section 3.3.2).

4.4 Proof of concept: simulations

In these experiments, we evaluate the approximation capabilities of the proposed technique (Algorithm 8), by applying it to synthetically generated matrices. We generate (R, K)-Kronecker-structured matrices and add some random Gaussian noise as follows:

$$\mathbf{D}_{\text{noisy}} = \sum_{r=1}^{R} \gamma^{-r} \mathbf{D}_1^r \boxtimes \cdots \boxtimes \mathbf{D}_K^r + \sigma \mathbf{N} = \mathbf{D} + \sigma \mathbf{N} \tag{4.10}$$

where the composing submatrices $\{\mathbf{D}_k^r\}_{k=\{1,\dots,K\}}^{r=\{1,\dots,R\}}$ as well as the noise matrix $\mathbf{N}$ are generated with random gaussian entries $\mathcal{N}(0,1)$ and the parameter σ controls the signal-to-noise ratio. We then run Algorithm 8 to obtain an approximation $\hat{\mathbf{D}} \in \mathcal{C}_K^R$ of $\mathbf{D}_{\text{noisy}}$.

We establish an exponential decay for the Kronecker summing terms, determined by the decay constant $\gamma \geq 1$. If we set $\gamma = 1$, then every summing term has the same power. The block dimensions are set to $\mathbf{n} = [5, 4, 3]$ and $\mathbf{m} = [6, 5, 4]$. We also set $K = 3$.

As a performance measure, we use the input and output signal-to-noise ratio (SNR):

$$\text{SNR}_{\text{in}} = \frac{\|\mathbf{D}\|_F^2}{\|\mathbf{D}_{\text{noisy}} - \mathbf{D}\|_F^2}, \quad \text{SNR}_{\text{out}} = \frac{\|\mathbf{D}\|_F^2}{\|\hat{\mathbf{D}} - \mathbf{D}\|_F^2}.$$

Informed approximation rank

Table 4.1 shows the output SNR (SNR_{out}) for different decay values. The rank used in the approximation $\hat{\mathbf{D}}$ corresponds to the true rank, which is set to $R = 3$. The results are averaged over 10 independent runs.

Table 4.1 – Output SNR (in dB) for different decay factors.

	Input SNR [dB]								
	-10	-5	0	5	10	15	20	25	30
Decay $\gamma = 1$	5.20	10.82	16.04	21.08	25.97	31.07	36.14	40.81	45.97
Decay $\gamma = 2$	3.63	9.10	15.90	21.00	25.94	31.05	36.13	40.81	45.97
Decay $\gamma = 10$	3.17	7.92	13.39	19.03	24.32	28.98	34.17	40.59	45.85

We observe an improvement of about 15dB with respect to the input SNR (i.e. $\text{SNR}_{\text{out}} \simeq \text{SNR}_{\text{out}} + 15\text{dB}$), which proves the effectiveness of proposed approach. When the energy of the terms decays quickly (i.e. for big decay constants γ), the factors with less energy become harder to retrieve, especially under higher noises. At some point, the noise can surpass the energy of some terms. When there is no decay ($\gamma = 1$), there is no such problem. In this case, however, the factors might be recovered in any permuted ordering by the CPD.

An example of recovered factors is given in Figure 4.4. We set smaller factor dimensions ($\mathbf{n} = [2, 2, 3]$, $\mathbf{m} = [3, 4, 4]$) and $R = 2$ for better visualization. While the factors in the first summing term ($r = 1$) are recovered with quite good precision, some differences are easy to spot in the remaining terms (second column).

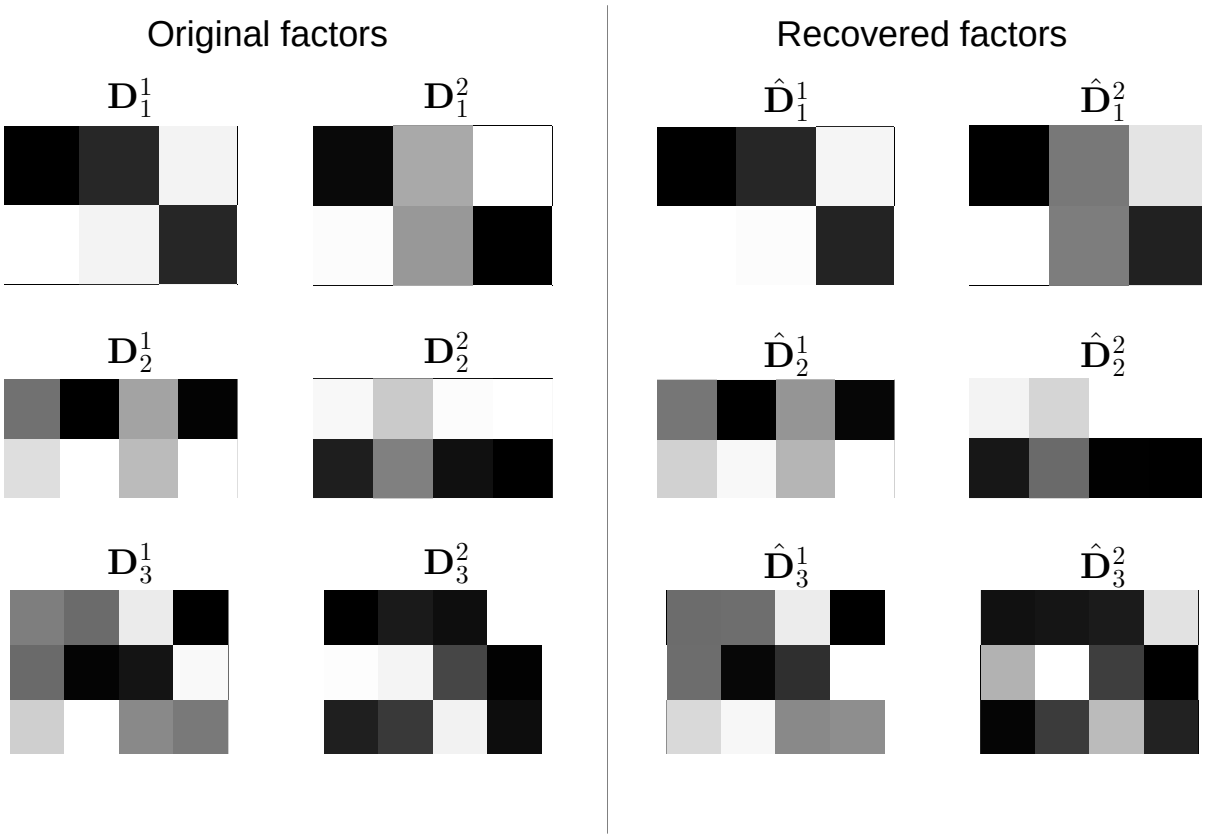


Figure 4.4 – Example of recovered factors with $\mathrm{SNR}_{\mathrm{in}} = 10\mathrm{dB}$.

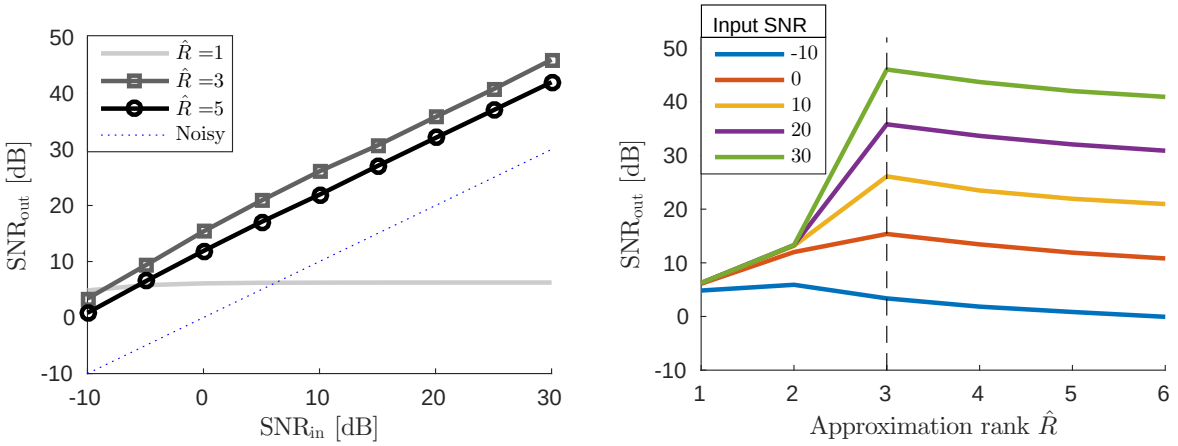


Figure 4.5 – SNR of the recovered factors as a function of the input SNR (left) and the reconstruction rank (right).

Unknown approximation rank

We now set $\gamma = 2$ and $R = 3$, but we do not suppose R to be known.

Figure 4.5 shows the output SNR as a function of the input SNR (left plot) and as a function of the approximation rank (right plot). Note that the best reconstruction occurs at the true rank $\hat{R} = R = 3$, except in very severe noise scenarios where the noise might mask some of the factors. Similar results are obtained for different block dimensions.

The results obtained in these simple experiments are reassuring, in that they provide encouraging initial evidence on the effectiveness of the proposed matrix approximation approach. In the following chapters, we will push further the ideas here introduced.

Chapter 5

Kronecker-structured Dictionary Learning

In this chapter we apply the proposed Sum-of-Kronecker model to the Dictionary Learning task. We show how the proposed HO-SuKro structure, can be used to obtain more compact and readily-applicable dictionaries than the completely unstructured ones when the targeted data is a collection of multiway arrays.

After formulating the Structured Dictionary Learning problem in a suitable way for multidimensional input data (section 5.1), we introduce two different algorithms for learning dictionaries constrained to the proposed HO-SuKro structure (sections 5.2 and 5.3).

> This chapter contains contributions published in the following conference papers: [Dantas *et al.* 2018] (first algorithm, section 5.2) and [Dantas *et al.* 2019b] (second algorithm, section 5.3).

The algorithm presented in section 5.2 is the natural extension to the dictionary learning problem of the tools developed in chapter 4 for matrix approximation. The algorithm presented in section 5.3 on the other hand, uses a different line of reasoning but leads to a considerably more efficient algorithm than the previous one.

Later, in section 5.4, we discuss the computational complexity of the proposed algorithms as well as some implementation details that have a considerable impact on both algorithmic complexity and actual speed. This leads to a discussion on how the theoretical speedups provided by the structured dictionaries translate to actual speedups in practice and which are the main bottlenecks in this regard.

5.1 Problem Statement

Let $\boldsymbol{\mathcal{Y}}$ be a collection of N tensor data $\{\boldsymbol{\mathcal{Y}}_1, \ldots, \boldsymbol{\mathcal{Y}}_N\}$ all of the same size. For the sake of simplicity let us suppose $\boldsymbol{\mathcal{Y}}_i$ are three-mode arrays in $\mathbb{R}^{n_1 \times n_2 \times n_3}$ stacked along the fourth

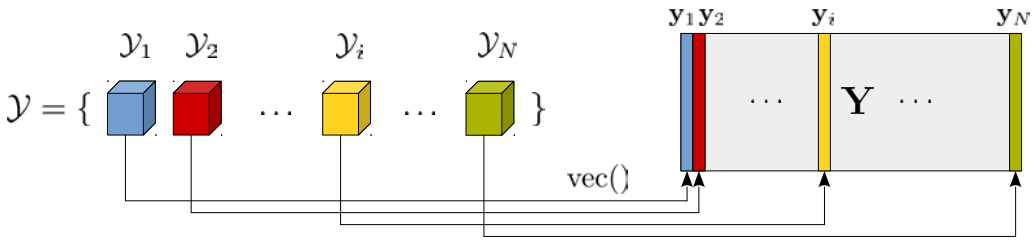


Figure 5.1 – Each input data sample $\boldsymbol{\mathcal{Y}}_i$ is a tensor (in this case, a three-mode array). To conform with the classical Dictionary Learning model, each vectorized sample corresponds to a column of the input data matrix $\mathbf{Y}$.

mode to form $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times N}$ –generalization to higher-order tensors is straightforward. We are interested in the Dictionary Learning problem, which consists in representing the input data as a linear combination of a few atoms from a certain (synthesis) dictionary $\mathbf{D}$ while simultaneously finding the dictionary itself. This problem is usually formulated for vector input data (cf. (1.9)). We recall it below along with an equivalent entrywise form:

$$\underset{\mathbf{D}\in\mathcal{S}_\mathbf{D},\mathbf{X}}{\operatorname{argmin}} \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \sum_{i=1}^{N} g(\mathbf{x}_i) \quad \Longleftrightarrow \quad \underset{\mathbf{D}\in\mathcal{S}_\mathbf{D},\mathbf{x}_i}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{y}_i - \mathbf{D}\mathbf{x}_i\|_2^2 + g(\mathbf{x}_i) \tag{5.1}$$

In the case of multi-dimensional data, a vectorized version of data samples is considered:

$$\underset{\mathbf{D}\in\mathcal{S}_\mathbf{D},\mathbf{x}_i}{\operatorname{argmin}} \sum_{i=1}^{N} \| \operatorname{vec}(\boldsymbol{\mathcal{Y}}_i) - \mathbf{D}\mathbf{x}_i\|_2^2 + g(\mathbf{x}_i) \tag{5.2}$$

where $\mathbf{D} \in \mathbb{R}^{n_1 n_2 n_3 \times m}$ is the dictionary which is often overcomplete ($m \geq n_1 n_2 n_3$) and belongs to a constraint set $\mathcal{S}_\mathbf{D}$. Function g is a sparsity inducing penalty and $\mathbf{x}_i$ is the sparse vector of coefficients describing the vectorized tensor $\boldsymbol{\mathcal{Y}}_i$ in the set of atoms $\mathbf{D}$. Most often, $\mathcal{S}_\mathbf{D}$ is the set of matrices with unit Euclidean norm columns. The equivalence between (5.2) and the previously presented formulation (5.1) is illustrated in Figure 5.1. Each column of the input data matrix $\mathbf{Y}$ is given by a vectorized sample $\mathbf{y}_i = \operatorname{vec}(\boldsymbol{\mathcal{Y}}_i)$.

$$\mathbf{Y} = [\operatorname{vec}(\boldsymbol{\mathcal{Y}}_1), \operatorname{vec}(\boldsymbol{\mathcal{Y}}_2), \ldots, \operatorname{vec}(\boldsymbol{\mathcal{Y}}_N)] \tag{5.3}$$

When tensor data is considered, two main drawbacks emerge in this formulation: i) it is completely agnostic to the original multidimensional structure of the data; ii) data sizes n_k may be relatively large (even more so the product $n_1 n_2 n_3$) and both the storage of matrix $\mathbf{D}$ and the computation of products such as $\mathbf{D}\mathbf{x}_i$ may be cumbersome.

One way to tackle the second issue is to restrict the class $\mathcal{S}_{\mathbf{D}}$ of dictionaries that are sought in problem (5.2). To also tackle the first mentioned issue, we make use of the proposed Higher-Order Sum of Kronecker model (HO-SuKro), which we have argued to be particularly suited to tensorial input data. This suitability comes primarily from the fact that the Kronecker product allows to create separable operators acting in vectorized multiway data. We recall the HO-SuKro structure:

$$\mathbf{D} = \sum_{r=1}^{R} \mathbf{D}_1^r \boxtimes \mathbf{D}_2^r \boxtimes \mathbf{D}_3^r \tag{5.4}$$

where $\boxtimes$ is the Kronecker product, $\mathbf{D}_k^r$ are matrices in $\mathbb{R}^{n_k \times m_k}$ and $m_1 m_2 m_3 = m$. The unit-norm constraint in $\mathbf{D}$ is handled as a post-processing step (see sections 5.2 and 5.3).

To summarize, we tackle the structured dictionary learning problem in which we constrain the learned dictionary to be (R, K)-Kronecker-structured. So, the constraint set $\mathcal{S}_{\mathbf{D}}$ is hereby given by the set $\mathcal{C}_K^R$ of all (R, K)-KS matrices of size $(n \times m)$ and with unit-norm columns.

To provide some extra intuition, for a fixed r we can see the terms $\mathbf{D}_k^r$ as linear operators acting independently in the k-th mode of a targeted tensorial data. For $R=1$, it boils down to the classic separable operators, which include for instance the 3-dimensional discrete cosine transform (3D-DCT).

An obvious remark about HO-SuKro is that the number of parameters to represent $\mathbf{D}$ using (5.4), which is $R(n_1 m_1 + n_2 m_2 + n_3 m_3)$, is much smaller than the number of entries $n_1 n_2 n_3 m_1 m_2 m_3$ as long as R is small. This entails several advantages w.r.t. unstructured dictionaries (as detailed in section 4.1.1): 1) matrix products with the dictionary can be computed markedly faster; 2) storage cost is significantly reduced; 3) learning process demands fewer training data – in other words, a reduced sample complexity.

By increasing the rank R, the set of attainable linear operators grows. However, the computational costs associated to the resulting dictionary also increase (the same applies to the storage cost and sample complexity). Therefore, rank value in the proposed approach may be understood as a trade-off between precision and complexity, storage cost and robustness to small training sets.

A tensor formulation

Let the coefficient vector $\mathbf{x}_i$ be rewritten as a tensor $\boldsymbol{\mathcal{X}}_i = \text{tens}_{m_1,m_2,m_3}(\mathbf{x}_i) \in \mathbb{R}^{m_1 \times m_2 \times m_3}$ and $\boldsymbol{\mathcal{D}} : \mathbb{R}^{m_1 \times m_2 \times m_3} \to \mathbb{R}^{n_1 \times n_2 \times n_3}$ as an operator such that elementwise, $\mathbf{D}\mathbf{x}_i = \text{vec}(\boldsymbol{\mathcal{D}}\boldsymbol{\mathcal{X}}_i)$,

and $m_1 m_2 m_3 = m$. Equation (5.2) may be rewritten as follows:

$$\underset{\boldsymbol{\mathcal{D}} \in \mathcal{S}_{\mathbf{D}}, \boldsymbol{\mathcal{X}}_i}{\operatorname{argmin}} \sum_{i=1}^{N} \|\boldsymbol{\mathcal{Y}}_i - \boldsymbol{\mathcal{D}} \boldsymbol{\mathcal{X}}_i\|_F^2 + g(\boldsymbol{\mathcal{X}}_i) \tag{5.5}$$

Let us denote $\otimes$ a tensor product mapping tuples of linear operators $\mathbf{D}_k^r$ to separable operators. It is such that, for any matrices $\mathbf{D}_1$ and $\mathbf{D}_2$ of legitimate sizes, $(\mathbf{D}_1 \otimes \mathbf{D}_2)\mathbf{X} = \mathbf{D}_1 \mathbf{X} \mathbf{D}_2^T$. A generalization to the third order, and further, is given by the n-mode product,

$$(\mathbf{D}_1 \otimes \mathbf{D}_2 \otimes \mathbf{D}_3)\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}} \times_1 \mathbf{D}_1^T \times_2 \mathbf{D}_2^T \times_3 \mathbf{D}_3^T \tag{5.6}$$

Note that this tensor product is deeply connected to the Kronecker product, which also maps tuples of linear operators $\mathbf{D}_k^r$ to separable operators. The difference is that the resulting operator $\boldsymbol{\mathcal{D}}$ now applies directly to the data in its original tensor form, thus skipping the vectorization step required by the Kronecker product (cf. (3.1)).

The HO-SuKro model now translates as

$$\boldsymbol{\mathcal{D}} = \sum_{r=1}^{R} \mathbf{D}_1^r \otimes \mathbf{D}_2^r \otimes \mathbf{D}_3^r. \tag{5.7}$$

Applying the usual tensor rank definition (see section 3.3) w.r.t. the tensor product $\otimes$ defined above, we denote $rank_{\otimes}(\boldsymbol{\mathcal{D}})$ the smallest number of products $\mathbf{D}_1 \otimes \mathbf{D}_2 \otimes \mathbf{D}_3$ that sum up exactly to $\boldsymbol{\mathcal{D}}$. In that sense, the HO-SuKro model (5.7) comes down to imposing $rank_{\otimes}(\boldsymbol{\mathcal{D}}) = R$. Therefore, the dictionary $\mathbf{D}$ is in fact an unfolded version of the operator $\boldsymbol{\mathcal{D}}$, which is supposed to be low-rank. Similarly, separable dictionaries such as DCT are nothing more than rank-one operators. Given the relative success of separable dictionaries (for instance, in image processing applications), it seems reasonable to restrain the set of admissible dictionaries $\mathcal{S}_{\mathbf{D}}$ to low-rank operators, that are even more general than separable ones.

Interestingly, equation (5.7) defines an operator version of the well-known CP decomposition for multiway arrays (recall that the CPD can be defined for a generic tensor product instead of the outer product, as mentioned in section 3.3). Indeed, we have seen in chapter 4 that the HO-SuKro terms $\mathbf{D}_k^r$ may be computed by performing the CPD of a transformed version $\mathcal{R}(\mathbf{D})$ of $\mathbf{D}$.

General optimization framework

Following the literature, we employ a sub-optimal alternating minimization strategy to tackle problem (5.1), which is not jointly convex (as discussed in section 1.5). Its non-convexity also implies that there are no optimality guarantees for this classical alternating approach. At each step, respectively called *dictionary update* and *sparse coding*, we optimize with respect to one variable, resp. $\mathbf{D}$ and $\mathbf{X}$, while fixing the other. In practice, each sub-problem is usually not exactly solved. The procedure is repeated N_{it} times, as shown in Algorithm 9.

Algorithm 9 $[\mathbf{D}, \mathbf{X}] = \text{HO-SuKroDL}(\mathbf{Y}, R, \mathbf{n}, \mathbf{m})$

for it $= \{1, 2, \ldots, N_{\text{it}}\}$ **do**

Sparse Coding: $\mathbf{X}_{\text{it}} \simeq \underset{\mathbf{X}}{\text{argmin}} \ \|\mathbf{Y} - \mathbf{D}_{\text{it}}\mathbf{X}\|_F^2 + g(\mathbf{X})$

Dictionary Update: $\mathbf{D}_{\text{it}} \simeq \underset{\mathbf{D} \in \mathcal{C}_K^R}{\text{argmin}} \ \|\mathbf{Y} - \mathbf{D}\mathbf{X}_{\text{it}}\|_F^2$

end for

The *sparse coding* step is unchanged by the Kronecker structure constraint since the dictionary is fixed, and can be performed by any existing sparse regression algorithm. One may use, for instance, greedy heuristics such as Orthogonal Matching Pursuit (OMP) [Pati *et al.* 1993] or convex relaxations based algorithms such as FISTA [Beck & Teboulle 2009]. The Kronecker structure can be exploited to positively impact the running time and complexity of these methods, see section 5.4 for more details.

The *dictionary update* step is tackled differently in each of the two proposed approaches, detailed in sections 5.2 and 5.3.

5.2 A Projected Gradient Approach

The difficulty in the *dictionary update* step lies in the (R,K)-KS structure constraint on $\mathbf{D}$. In fact, in chapter 4 we have shown that this constraint is equivalent to imposing a Canonical Polyadic Decomposition model on a rearranged tensor $\mathcal{R}_K(\mathbf{D})$ obtained from dictionary $\mathbf{D}$. Therefore, for this step, we propose a projected gradient algorithm, where the projection onto the set of $\mathcal{C}_K^R$ (set of matrices written as a sum of R K-Kronecker products) is given by the CPD algorithm applied to the rearranged tensor $\mathcal{R}_K(\mathbf{D})$ as shown in Algorithm 10. The projection step was previously presented in Algorithm 8.

Algorithm 10 $\mathbf{D} = \text{DictionaryUpdate}(\mathbf{Y}, \mathbf{X}, R, \mathbf{n}, \mathbf{m}, \mathbf{D}_0)$

1: Set $t = 0$
2: **while** $\|\mathbf{D}_{t+1} - \mathbf{D}_t\|_F > tol$ **do**
3: $\quad \mathbf{D}_{t+1} = \mathbf{D}_t - \gamma_t(\mathbf{D}_t\mathbf{X} - \mathbf{Y})\mathbf{X}^T$ ▷ Gradient step
4: $\quad \mathbf{D}_{t+1} = \text{HO-SuKroApprox}(\mathbf{D}_{t+1}, R, \mathbf{n}, \mathbf{m})$ ▷ Projection onto $\mathcal{C}_K^R$ (see Alg. 8)
5: **end while**
6: Normalize columns of $\mathbf{D}$

The step-size γ_t is determined with a backtracking line search. For acceleration purposes, the CPD can be initialized with the results of the previous iteration. Finally, note that the column normalization (line 5) can break the imposed structure. This is not a real concern, since the normalization coefficients can be stored separately, implying only m additional products in a matrix-vector operation. Put differently, it is equivalent to storing a dictionary in the form $\mathbf{D}\boldsymbol{\Sigma}$ where $\boldsymbol{\Sigma}$ is a diagonal matrix containing the inverse norm of each column of $\mathbf{D}$.

5.3 An Alternate Least Squares approach

In what follows, we show that a rather simple algorithm can be designed to estimate matrices $\mathbf{D}_k^r$ in the HO-SuKro Dictionary Learning model. This contrasts with sometimes intricate optimization schemes employed in the literature (see references mentioned in section 4.1.2). In particular, it is more efficient than the projected alternating algorithm proposed in section 5.2. The final gains in parameter size and memory requirements are the same for both algorithms, once the dictionary is learned. However, the time spent in the learning process can be significantly different between both algorithms.

The main distinction between the two proposed algorithms, which mostly explains the efficiency discrepancy, is that in the previous algorithm we *leave* the constraint space at each step and then project back into it. This means that we mainly deal with *unstructured* dictionaries during the learning process. In this new algorithm, on the other hand, we refine directly the factors $\mathbf{D}_k^r$ within the HO-SuKro structure.

Just as before, we employ the described alternate optimization strategy and the *sparse coding* step remains unchanged. Let us now concentrate on the *dictionary update* step, which is affected by the HO-SuKro constraint. In this algorithm, we exploit the following insight: the minimization with respect to a single block $\mathbf{D}_k^r$, with $\mathbf{X}$ and all other blocks $\{\mathbf{D}_{k'}^{r'}\}_{k' \neq k}^{r' \neq r}$ fixed, is a quadratic problem which has a closed-form solution. Let us detail this

point by tacking $K = 3$ for simplicity. The extension to a higher order is straightforward.

The partial cost function with respect to the blocks $\{\mathbf{D}_k^r\}_{k=1,\dots,3}^{r=1,\dots,R}$ may be written as follows:

$$f(\{\mathbf{D}_k^r\}_{k,r}) = \sum_{i=1}^{N} \| \operatorname{vec}(\boldsymbol{\mathcal{Y}}_i) - \sum_{r=1}^{R} (\mathbf{D}_1^r \boxtimes \mathbf{D}_2^r \boxtimes \mathbf{D}_3^r)\, \mathbf{x}_i \|_2^2 \tag{5.8}$$

Of course, one may think of multiple techniques to minimize (5.8), for instance gradient-based approaches, second order methods, or block-coordinate descent. In fact, because of the similarity of (5.8) with the Tucker Decomposition problem (defined in section 3.4.1)[1], a first approach we tried (but that revealed to be slower than the final retained solution) was to minimize (5.8) in an alternating fashion, fixing all parameters but one elementary block $\mathbf{D}_k^r$. This leads to a quadratic problem, which can be solved in closed form.

However, in the spirit of Alternating Least Squares algorithms (ALS), it is possible to gather all elementary blocks $\mathbf{D}_k^r$ for a fixed mode k and rather alternate between only three macro-blocks (one per mode) $\boldsymbol{\Delta}_k = \{\mathbf{D}_k^r\}_{r\in[1,R]}$, $1 \leq k \leq 3$, and still have a closed-form solution. Figure 5.2 illustrates the macro-blocks $\boldsymbol{\Delta}_k$.

$$\mathbf{D} = \sum_{r=1}^{R} \mathbf{D}_1^r \boxtimes \mathbf{D}_2^r \boxtimes \mathbf{D}_3^r = \begin{array}{ccccc} & \boldsymbol{\Delta}_1 & & \boldsymbol{\Delta}_2 & & \boldsymbol{\Delta}_3 \\ & \mathbf{D}_1^1 & \boxtimes & \mathbf{D}_2^1 & \boxtimes & \mathbf{D}_3^1 \\ + & \mathbf{D}_1^2 & \boxtimes & \mathbf{D}_2^2 & \boxtimes & \mathbf{D}_3^2 \\ & \vdots & & \vdots & & \vdots \\ + & \mathbf{D}_1^R & \boxtimes & \mathbf{D}_2^R & \boxtimes & \mathbf{D}_3^R \end{array}$$

Figure 5.2 – All terms $\mathbf{D}_k^r$ in a same macro-block $\boldsymbol{\Delta}_k$ are updated simultaneously.

5.3.1 Quadratic partial cost function explained

Using, for instance, the first-mode unfolding denoted $\mathbf{Y}_{(1)} \in \mathbb{R}^{n_1 \times n_2 n_3 N}$ for $\boldsymbol{\mathcal{Y}}$ and $\mathbf{X}_{(1)} \in \mathbb{R}^{m_1 \times m_2 m_3 N}$ for $\boldsymbol{\mathcal{X}}$, and identity (3.2) which allows to isolate a single block $\mathbf{D}_1^r$ from the Kronecker product, one can rewrite (5.8) as

$$f(\{\mathbf{D}_k^r\}_{k,r}) = \|\mathbf{Y}_{(1)} - \sum_{r=1}^{R} \mathbf{D}_1^r \mathbf{X}_{(1)} (\mathbf{D}_2^r \boxtimes \mathbf{D}_3^r \boxtimes \mathbf{I}_N)^T \|_F^2 \tag{5.9}$$

1. We refer the interested reader to Appendix B.1 for a more detailed parallel between this problem and the Tucker decomposition.

The identity matrix $\mathbf{I}_N$ that appears in the Kronecker structure is an implicit way to apply the dictionary to $\boldsymbol{\mathcal{X}}_i$ simultaneously for all $i \in \{1, \ldots, N\}$. Equation (5.9), in turn, can be seen as a matrix factorization problem

$$f(\{\mathbf{D}_k^r\}_{k,r}) = \|\mathbf{Y}_{(1)} - \boldsymbol{\Delta}_1 \mathbf{U}_1\|_F^2 \tag{5.10}$$

where $\boldsymbol{\Delta}_1 = \left[\mathbf{D}_1^1, \ \ldots \ , \mathbf{D}_1^R\right] \in \mathbb{R}^{n_1 \times Rm_1}$ contains the horizontally-stacked matrices $\mathbf{D}_1^r$ with $1 \leq r \leq R$, and $\mathbf{U}_1 = \left[\mathbf{U}_1^1; \ \ldots \ ; \mathbf{U}_1^R\right] \in \mathbb{R}^{Rm_1 \times n_2 n_3 N}$ the vertically-stacked right-hand terms with $\mathbf{U}_1^r = \mathbf{X}_{(1)}(\mathbf{D}_2^r \boxtimes \mathbf{D}_3^r \boxtimes \mathbf{I}_N)^T$.

Note that f is quadratic with respect to $\boldsymbol{\Delta}_1$, and an optimal solution for fixed $\mathbf{U}_1$ is given in closed form by

$$\widehat{\boldsymbol{\Delta}_1} = \underset{\boldsymbol{\Delta}}{\operatorname{argmin}} \|\mathbf{Y}_{(1)} - \boldsymbol{\Delta}_1 \mathbf{U}_1\|_F^2 = \mathbf{Y}_{(1)} \mathbf{U}_1^\dagger. \tag{5.11}$$

where $\mathbf{U}_1^\dagger = \mathbf{U}_1^T \left(\mathbf{U}_1 \mathbf{U}_1^T\right)^{-1}$ is the right pseudo-inverse of $\mathbf{U}_1$. A similar reasoning applies to the other modes with the adapted definition of $\mathbf{U}_k$.

5.3.2 ALS for training

Since we have shown that f is quadratic with respect to blocks $\boldsymbol{\Delta}_k$, let us now derive formally an algorithm to estimate these blocks $\boldsymbol{\Delta}_k$ using Alternating Least Squares, as well as $\boldsymbol{\mathcal{X}}$. ALS is reported as a baseline algorithm for unconstrained PARAFAC [Kolda & Bader 2009]. It is also a well studied algorithm for computing the Tucker decomposition, a decomposition to which our training problem reduces when $R=1$ [Caiafa & Cichocki 2013].

The proposed ALS is summarized in Algorithm 11, for the spacial case of $K = 3$. Note that in this configuration, the training data in its matrix form $\mathbf{Y} \in \mathbb{R}^{n_1 n_2 n_3 \times N}$ (cf. equation (5.3)) is obtained from the tensor $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{n_1 \times n_2 \times n_3 \times N}$ by unfolding it w.r.t. the fourth mode $\mathbf{Y}_{(4)}^T$. Factors $\boldsymbol{\Delta}_k$ are estimated in an alternating fashion using (5.11), while $\boldsymbol{\mathcal{X}}$ is computed using any sparse coding method. Little can be said about convergence of Algorithm 11, since ALS for the training problem does not have a local convergence guarantee [Kolda & Bader 2009], and the sparse coding step is not even guaranteed to decrease the cost function. Yet, no convergence problems have been observed in practice. Extension of Algorithm 11 to $K > 3$ (and even $K = 2$) is straightforward.

Special attention has to be devoted to satisfy the unit-norm constraint. Like most dictionary learning algorithms, we normalize the estimated $\mathbf{D} = \sum_r \mathbf{D}_1^r \boxtimes \mathbf{D}_2^r \boxtimes \mathbf{D}_3^r$ at

each iteration. However, at line 17 of Algorithm 11, we store both the blocks $\{\mathbf{D}_k^r\}_{k=1,\dots,3}^{r=1,\dots,R}$ and the norms $\mathbf{\Sigma} = \text{diag}\,(1/\|\mathbf{D}(:,j)\|_2)$ separately to make use of the Kronecker structure in the sparse coding step. Even if this ad-hoc normalization step may marginally increase the cost function, it was not observed to decisively impact the algorithm's performance.

Algorithm 11 Alternating Least Squares for HO-SuKro

1: **INPUTS:** Data $\boldsymbol{\mathcal{Y}}$, initial $\mathbf{D}_k^r$ $(k \in [1,2,3], r \in [1,\dots,R])$
2: Write $\mathbf{D}_4^r = \mathbf{I}_N$ for all $r \leq R$
3: **while** stopping criterion is not met **do**
4: ▷ *Sparse Coding*
5: $\mathbf{X}_{(4)} \simeq \text{argmin}_{\mathbf{X}} \|\mathbf{Y}_{(4)}^T - \mathbf{D}\mathbf{X}\|_F + g(\boldsymbol{\mathcal{X}})$ using any sparse coding algorithm.
6: ▷ *Dictionary Update*
7: **while** update on $\{\mathbf{D}_k^r\}_{k,r}$ is significant **do**
8: **for** k from 1 to 3 **do**
9: Set $\mathbf{U}_r = \mathbf{X}_{(k)} \left(\boxtimes_{l \neq k} \mathbf{D}_l^r\right)^T$ for all $r \leq R$
10: Set $\mathbf{U} = [\mathbf{U}_1; \dots; \mathbf{U}_R]$ stacked vertically
11: $\mathbf{\Delta}_k = \mathbf{Y}_{(k)}\mathbf{U}^T \left(\mathbf{U}\mathbf{U}^T\right)^{-1}$
12: Set $[\mathbf{D}_k^1, \dots, \mathbf{D}_k^R] = \mathbf{\Delta}_k$ stacked horizontally
13: **end for**
14: **end while**
15: ▷ *Dictionary Column Normalization*
16: $\mathbf{\Sigma} = \text{diag}\,(1/\|\mathbf{D}(:,j)\|_2)$, with $j \in \{1,\dots,m\}$
17: $\mathbf{D} = (\sum_{r=1}^R \mathbf{D}_1^r \boxtimes \mathbf{D}_2^r \boxtimes \mathbf{D}_3^r)\mathbf{\Sigma}$
18: **end while**
19: **OUTPUTS:** Estimated blocks $\mathbf{D}_k^r$, norms $\mathbf{\Sigma}$, sparse $\boldsymbol{\mathcal{X}}$

5.4 Complexity analysis and practical speedups

In this section we analyze the complexity costs associated to the HO-SuKro dictionaries. It is important to distinguish two scenarios in this discussion: 1) the complexity of the learning algorithm; 2) the complexity associated to HO-SuKro dictionary *once learned*.

When it comes to the second point, the advantages of HO-SuKro have been scrutinized in section 4.1.1. As a brief recall, they are threefold: i) reduced computational complexity in matrix-vector operations; ii) reduced storage cost; iii) reduced sample complexity.

Naturally, these advantages can also positively impact the learning algorithm. The third advantage (reduced sample complexity) can be very useful in practice by allowing the learning to be carried over with less training data, making it proportionally lighter

and faster. The second advantage contributes in reducing space complexity during the training process whenever the dictionary is structured[2]. Finally, in terms of computational complexity, the learning algorithms are impacted in both steps of the alternate minimization: 1) dictionary update and 2) sparse coding (detailed respectively in sections 5.4.1 and 5.4.2).

Basically, as already mentioned, every time a product between $\mathbf{D}$ and any collection of vectors $\mathbf{V} = [\mathbf{v}_1, \dots, \mathbf{v}_N] \in \mathbb{R}^{m \times N}$ is computed, one may use the following result:

$$\mathbf{DV} = \left(\sum_{r=1}^{R} \mathbf{D}_1^r \boxtimes \mathbf{D}_2^r \boxtimes \mathbf{D}_3^r \right) \mathbf{V} = \sum_{r=1}^{R} \boldsymbol{\mathcal{V}} \times_1 \mathbf{D}_1^r \times_2 \mathbf{D}_2^r \times_3 \mathbf{D}_3^r \tag{5.12}$$

where $\boldsymbol{\mathcal{V}} \in \mathbb{R}^{m_1 \times m_2 \times m_3 \times N}$ is a tensorized version of $\mathbf{V}$ and $\times_i$ is the mode-wise product that verifies $\mathbf{A} \times_i \boldsymbol{\mathcal{V}} := \mathbf{A}\mathbf{V}_{(i)}$. So, it comes down to a sequence of mode products with the smaller matrices $\mathbf{D}_k^r$ which involves approximately $R(\sum_k n_k)(\prod_k m_k)N$ term to term products instead of $(\prod_k n_k m_k)N$ for a full matrix product $\mathbf{DV}$.

The same reasoning applies to the adjoint operator, by using the simple observation that $\mathbf{D}^T = \sum_{r=1}^{R} (\mathbf{D}_1^r)^T \boxtimes (\mathbf{D}_2^r)^T \boxtimes (\mathbf{D}_3^r)^T$.

When computing a sequence of mode-product as in (5.12), the mode ordering can be chosen wisely to minimize the total complexity. Since the mode dimensions m_k are progressively replaced by n_k ($< m_k$), an efficient strategy consists in choosing modes with higher ratio m_k/n_k first, to maximize the dimensionality reduction at each step. For simplicity, we will use an upper bound for the complexity of (5.12), given by the compact expression $R(\sum_k n_k)(\prod_k m_k)N$. In this simplification, the tensor dimensions are always considered to be m_k $\forall k$, even though in practice such dimensions are progressively reduced to $n_k \leq m_k$ as the corresponding mode products are performed.

Additionally, some parallelization is possible with respect to r, since the different terms in the summation can be computed independently. This point is not considered in the complexity analysis below.

2. Note that in the projected gradient algorithm, the dictionary becomes unstructured at every iteration, before the projection step. Nevertheless, during the entire sparse coding step, the HO-SuKro structure can still be harnessed.

5.4.1 Dictionary update step

Algorithm 10

In the projected gradient algorithm, the dictionary update consists of two major steps: a gradient update (a) and a projection step (b).

(a) $\mathbf{D}_{t+1} = \mathbf{D}_t - \gamma_t(\mathbf{D}_t\mathbf{X} - \mathbf{Y})\mathbf{X}^T$, where $\mathbf{D}_t$ is structured but not $\mathbf{D}_{t+1}$.

(b) $\mathbf{D}_{t+1} = \text{HO-SuKroApprox}(\mathbf{D}_{t+1}, R, \mathbf{n}, \mathbf{m})$, where $\mathbf{D}_{t+1}$ becomes structured again.

During the gradient step, the dictionary is structured, which accelerates the matrix product $\mathbf{D}_t\mathbf{X}$. After this, the dictionary is no longer structured until the projection step is finished. The computational cost of the latter step is dominated by the CPD computation. The computational complexities in each of these steps are detailed in Table 5.1. We denote $\text{nnz}(\mathbf{X})$ the number of non-zero entries in $\mathbf{X}$.[3]

For the CPD, we considered the standard ALS algorithm discussed in section 3.3.2. Its cost is dominated by matrix-matrix products, repeated for $k \in \{1, \dots, K\}$, between the unfoldings of the rearranged dictionary and a matrix grouping all other fixed factors, respectively of sizes $(n_k m_k \times \prod_{l \neq k} n_l m_l)$ and $(\prod_{l \neq k} n_l m_l \times R)$.

Rearrangement cost Note that the rearrangement operation is apparently transparent in terms of computational complexity, since it does not involve any sum or multiplication operation, only some memory reallocation. In practice, however, such operation does have a time cost. A similar discussion emerges when implementing the successive mode products in the HO-SuKro matrix-vector operation. This point is further developed in section 5.4.3.

Table 5.1 – Algorithm 10 computational complexity: dictionary update.

	Operation	**Complexity**
Gradient step	$\mathbf{\Delta} = \mathbf{DX} - \mathbf{Y}$	$R(\sum_k n_k)(\prod_k m_k)N + (\prod_k n_k)N$
	$\mathbf{\Delta}\mathbf{X}^T$	$(\prod_k n_k)\,\text{nnz}(\mathbf{X})$
Projection step	CPD	$\mathcal{O}(KR(\prod_k n_k m_k))$

3. So, if $\bar{s}$ is the average sparsity of its columns, we have that $\text{nnz}(\mathbf{X}) = \bar{s}N$.

Algorithm 11

In the ALS algorithm, the HO-SuKro structure is particularly useful when calculating $\mathbf{U}_r$ for all $r \leq R$ (line 9). Table 5.2 lists the computational complexities for one iteration of the inner loop in the dictionary update step (lines 8-13 in Algo. 11). The listed operations are performed for each of the modes and repeated a certain number of times (outer loop, lines 7-14). Our experiments indicate that very few outer iterations suffice to provide good convergence (typically less than five).

Computational complexities for an alternative calculation ordering are detailed in Appendix B.2. Although leading to comparable theoretical complexities, this alternative implementation was empirically observed to be slower in practice.

Table 5.2 – Algorithm 11 computational complexity: dictionary update

Operation	**Complexity**
$\mathbf{U}_r,\ \forall r \leq R$	$R(\sum_{l\neq k} n_l)(\prod_l m_l)N$
$\mathbf{Y}_{(k)}\mathbf{U}^T$	$R(\prod_l n_l)m_k N$
$\mathbf{U}\mathbf{U}^T$	$R^2(\prod_{l\neq k} n_l)m_k^2 N$
$(\mathbf{U}\mathbf{U}^T)^{-1}$	$(Rm_l)^3$

Discussion

Considering that typically $N \gg \prod_k m_k > \prod_k n_k \gg R$, the first line in both Tables 5.1 5.2, scaling with $(\sum_l n_l)(\prod_l m_l)N$, are among the most costly – alongside with the CPD, which, although not scaling with N, requires several iterations of the reported complexity. Thanks to the tensor structure we manage to completely avoid complexities scaling with $(\prod_l n_l)(\prod_l m_l)N$ as it would be the case for a single product $\mathbf{DX}$ or $\mathbf{D}^T\mathbf{Y}$ with an unstructured dictionary.[4]

In practice, the projected gradient algorithm takes significantly more iterations to converge (about an order of magnitude) than the ALS algorithm. In addition, in the former algorithm, the CPD adds an extra inner loop, which also takes a considerable number of iterations. The result in practice is a gap of about an order of magnitude in

4. In Algorithm 10, we still manipulate an unstructured dictionary during the CPD step. However, no operation involving $\mathbf{Y}$ or $\mathbf{X}$ (which bring the dimension N) is performed during this step.

execution time. Obviously, the gap depends on several parameters, especially the stopping criteria. A more detailed comparison is provided in the experiments chapter (section 6.3.2).

Note that we neglected the sparsity of $\mathbf{X}$ when multiplying with a HO-SuKro dictionary. Nonetheless, the final complexity is not drastically affected by this simplification, since the sparsity is lost as soon as the first mode-product is performed. The ensuing mode-products are not affected. The first mode-product (say, mode k) would cost $n_k \operatorname{nnz}(\mathbf{X})$ instead of $n_k(\prod_k m_k)N$, but the final complexity remains of the same order: $R(\sum_k n_k)(\prod_k m_k)N$.

In terms of space complexity, the structured dictionary is more compact since only the factors $\mathbf{D}_k^r$ (and norms $\mathbf{\Sigma}$) need to be stored. However, in the projected gradient algorithm, the dictionary repeatedly becomes unstructured, which invalidates this particular advantage. Since the coefficient matrix $\mathbf{X}$ is sparse – and therefore also compact to store –, the training data matrix $\mathbf{Y}$ ends up dominating the space complexity. To effectively overcome this bottleneck and achieve actual scalability, one should consider, associated to the proposed structured dictionaries, an online leaning algorithm, for instance, inspired by the approaches proposed in [Mairal *et al.* 2009] and [Sulam *et al.* 2016].

5.4.2 Structured sparse coding

Sparse coding algorithms, either greedy (like OMP and its variants) or convex-relaxation-based (like Iterative Soft-Thresholding algorithm – ISTA – and its variants), can also benefit from the proposed structure in the dictionary. Basically, any product with the dictionary or its adjoint can be replaced by a mode-product with the smaller factors $\mathbf{D}_k^r$ as shown in eq. (5.12).

In OMP-like algorithms, the complexity of an iteration is dominated by the calculation of the inner product between the dictionary and the current residual in $\mathcal{O}(\prod_k n_k m_k)$. See Table 5.3 for some empirical evidence of this fact, which was previously pointed out in section 2.1.1. For an HO-SuKro dictionary, this operation can be accelerated using (5.12) to a complexity of $\mathcal{O}(R(\sum_k n_k)(\prod_k m_k))$. This operation is followed by a normalization using $\mathbf{\Sigma}$ with a minor computation cost of $\prod_k m_k$ multiplications.

The same reasoning applies to ISTA-like algorithms (see section 2.2.1). The iteration complexity is dominated by the gradient step $\nabla f(\mathbf{D}) = \mathbf{D}^T(\mathbf{Y} - \mathbf{D}\mathbf{X})$. The matrix products $\mathbf{D}\mathbf{X}$ and $\mathbf{D}^T(\mathbf{Y} - \mathbf{D}\mathbf{X})$, or $(\mathbf{D}^T\mathbf{D})\mathbf{X}$ and $\mathbf{D}^T\mathbf{Y}$, can be replaced by structured products.

The mentioned accelerations can be used to speedup the learning process (sparse

Table 5.3 – Time spent in operation $\mathbf{D}^T\rho$ for 10 iterations

n	[6, 6, 6]	[8, 8, 8]	[10, 10, 10]
m	[12, 12, 12]	[16, 16, 16]	[20, 20, 20]
OMP Cholesky	46 %	83 %	93 %

coding step) and also after it, once the dictionary is learned and applied repeatedly to some targeted data. In the patch-based denoising applications considered in chapter 6, the latter corresponds to the reconstruction of all image patches using the learned dictionary.

Table 5.3 shows the percentage of time spent in the matrix-vector product $\mathbf{D}^T\rho_t$ between the dictionary and the current residual, $\rho_t = \mathbf{y} - \mathbf{D}\mathbf{x}_t$ given the estimated coefficient vector $\mathbf{x}_t$ at iteration t, for a standard implementation of OMP with an unstructured dictionary. A higher time percentage implies a higher potential of speedup with the structured dictionary. Note that the percentage grows with the problem's dimensions.

Other more intricate implementations of OMP, such as Batch-OMP (see Algorithm 4 in chapter 2), which assumes the precomputation of both $\mathbf{D}^T\mathbf{Y}$ and the Gram matrix ($\mathbf{D}^T\mathbf{D}$), still admit some acceleration with the proposed structure. This, however, requires a more involved discussion, which we put aside for future works.

5.4.3 From theoretical to practical speedups

The proposed accelerations, in both dictionary update and sparse coding steps, rely on the theoretical complexity of tensor mode-products. In this section, we evaluate to which extent such theoretical gains translate into actual speedups.

As explained, a product with the HO-SuKro dictionary becomes a sequence of mode-products. In the way such operations are conventionally implemented, the data tensor is repeatedly unfolded along each of its modes. Given a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{m_1 \times m_2 \times m_3 \times N}$, the mode-$k$ unfolding is obtained by cascading two operations: (i) a permutation of the indexes of $\boldsymbol{\mathcal{X}}$ so that mode k becomes the first dimension of the permuted tensor, and (ii) a reshape operation that unfolds all modes but the first along the second dimension. Although such operations are neglected in theory (considered $\mathcal{O}(1)$), this is not the case in practice as a reorganization of the tensor entries in memory might be required.

This seemingly harmless overhead actually creates a considerable gap between theoretical and empirical speedups, especially for smaller dimensions, as exemplified in Table 5.4.

Table 5.4 – Speedups in matrix-vector products

Dimensions		Theoretical speedup | *Empirical speedup*					
n	m	$R=1$		$R=3$		$R=5$	
[6, 6, 6]	[12, 12, 12]	20.3	*0.8*	6.8	*0.4*	4.9	*0.3*
[8, 8, 8]	[16, 16, 16]	36.6	*4.5*	12.2	*2.0*	7.3	*1.4*
[10, 10, 10]	[20, 20, 20]	57.1	*13.2*	19.1	*5.7*	11.4	*3.5*

Nevertheless, some considerable acceleration is still achieved in practice, in higher dimensions. A lower-level implementation [Li *et al.* 2015] [Matthews 2018] can be considered to try and tighten this gap, which would be of great interest since there is still a large acceleration potential to be exploited. The approach proposed in [Vannieuwenhoven *et al.* 2013] that aims at eliminating explicit data reordering is also worth-considering.

Chapter 6

Experiments

In this chapter we evaluate the dictionary learning algorithms that were proposed in chapter 5 on some image denoising applications.

> This chapter gathers results presented in the following papers:
> [Dantas *et al.* 2018] (color image denoising), [Dantas *et al.* 2019b] and [Dantas *et al.* 2019a] (hyperspectral image denoising).

After briefly discussing the sparsity-based denoising framework (section 6.1), we describe the patch-based approach employed in all experiments (section 6.2). Then, in sections 6.3 and 6.4 we show some results respectively on color image [Dantas *et al.* 2018] and hyperspectral image [Dantas *et al.* 2019b] data. Finally, in section 6.5 we introduce a more advanced hyperspectral image denoising approach [Dantas *et al.* 2019a] inspired by some prior-art techniques, using a low-rank image model.

6.1 Sparsity and denoising

Removing noise from a certain targeted data is a classical problem in signal processing. Being the simplest possible inverse problem, it provides a convenient platform over which signal processing ideas and techniques can be tested and perfected. In this chapter, we will be interested on the image denoising task. There is, of course, an immense variety of techniques using different paradigms to tackle this problem, each accompanied by a vast literature. We thus have no pretension to provide a complete survey of the area. But before describing the proposed algorithms and simulation set-ups we dedicate this section to briefly overview a particular family of techniques: those using sparse and redundant representation modeling. For a more detailed survey, we refer the reader for instance to [Elad 2010, Chapter 14].

The sparse model supposes the existence of a suitable representation domain (a dictionary) in which the image can be sparsely approximated [Mairal *et al.* 2008].

Initial attempts to design dictionaries that fulfill the above model assumption included wavelets of various sorts [Mallat 2008], wedgelets [Donoho 1999], curvelets [Starck *et al.* 2002], contourlets [Do & Vetterli 2005], bandelets [Pennec & Mallat 2005], shearlets [Labate *et al.* 2005] and steerable wavelets [Simoncelli *et al.* 1992]. These analytical transforms have had great success in the image denoising task [Portilla *et al.* 2003, Eslami & Radha 2006].

All the above-mentioned methods employ a global model on the image, which requires a dictionary that sparsifies images as a whole. An appealing alternative to this global approach is a local processing of the image by operating on small patches. This led to the introduction of trained dictionaries [Elad & Aharon 2006, Mairal *et al.* 2008], which exploit the non-local self-similarity in natural images (i.e. the existence of self-repeating patterns across the image). Another related technique that exploits this same property is the Non-Local Means (NLM) algorithm [Buades *et al.* 2005] and related works [Kervrann & Boulanger 2006, Mahmoudi & Sapiro 2005].

Patch-based dictionary learning (to be detailed in section 6.2 below) has been first applied to black and white images in [Elad & Aharon 2006] and then extended to color images in [Mairal *et al.* 2008] by employing 3D-patches to also capture the correlation between the different color channels. The application of this type of approach to hyperspectral images (HSI) is still in its early stages, some examples are [Xing *et al.* 2012, Fu *et al.* 2015, Zhao & Yang 2015]. The high correlation and redundancy in the spectral domain is known to considerably enhance the denoising performance [Lam *et al.* 2012, Rasti *et al.* 2018], which motivates the use of 3D-modeling approaches. However, it remains unclear if –and how– such correlations can be duly exploited via local processing with patches. While in [Xing *et al.* 2012] and [Fu *et al.* 2015] the patches range over the entire spectral dimension, in [Zhao & Yang 2015] the patches are extracted from a reshaped version image obtained by vectorizing the spatial dimensions. Here, we adopt a straightforward extension of the approaches in [Elad & Aharon 2006] and [Mairal *et al.* 2008], by taking 3D-patches distributed all over the HSI data cube.

6.2 Patch-based denoising procedure

A common approach for dictionary-based image denoising consists in handling small image patches [Elad & Aharon 2006]. It allows to collect enough training data to learn the dictionary even if only the noisy image is used. An alternative scenario would be to

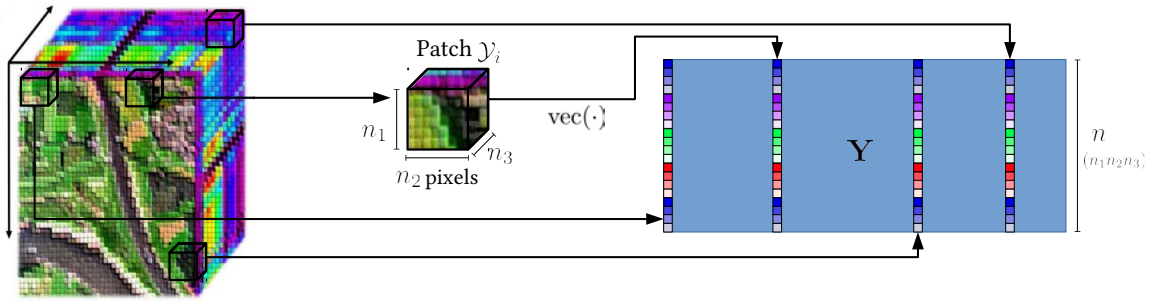


Figure 6.1 – Construction of the training data matrix $\mathbf{Y}$ by extracting 3D-patches from the noisy image.

learn the dictionary on a corpus of patches taken from a set of noiseless images. In the following experiments, we place ourselves in the former *single-image* denoising scenario. The adopted simulation set-up follows the one proposed in [Elad & Aharon 2006].

The test images are corrupted by white Gaussian noise with standard deviation σ.

Learning

The training data consists of N patches which are extracted from the noisy image. The patches are taken uniformly spaced and might partially overlap with each other, depending on N. As a pre-treatment, each patch is re-centralized to have a zero mean along its pixel values. Then, each patch is vectorized to form the columns of the training data matrix $\mathbf{Y}$.

The construction of the training data matrix is illustrated in Figure 6.1. In our experiments, we deal with 3D-data from which we extract 3D-patches of size $(n_1 \times n_2 \times n_3)$. We denote $(N_{px1} \times N_{px2} \times N_{px3})$ the dimensions of the input image. Note that for the color image experiments (section 6.3), the third dimension is only of size $N_{px3} = 3$. In this case, the patches take the entire range of pixels in this dimension (i.e. $n_3 = N_{px3} = 3$) and the patch extraction consists in sweeping only through the two spatial dimensions. This is different from the hyperspectral case (section 6.4) in which $n_3 < N_{px3}$ and the patch extraction phase also includes sweeping through the third (spectral) dimension.

A dictionary is learned from this data with N_{it} iterations of alternating optimization (dictionary update and sparse coding). Sparse coding is performed by OMP with an error threshold τ proportional to the noise level σ, i.e.: $\tau = \sigma\sqrt{\prod_k n_k}$.

Although in sections 6.3 and 6.4 te noise level is supposed known, in section 6.5 we propose a way to estimate it.

Denoising

The learned dictionary is then used to reconstruct all possible overlapping patches in the image, with a one-pixel translation from its neighbors. Therefore, the total number of patches to be reconstructed is given by $N_{\text{patches}} = (N_{px1} - n_1 + 1)(N_{px2} - n_2 + 1)(N_{px3} - n_3 + 1)$. The reconstruction is performed via sparse coding with the OMP algorithm, just like in the learning phase, but on a considerably larger amount of data ($N_{\text{patches}} \gg N$). Signal, as opposed to noise, is expected to be well approximated by few atoms. Thus, the reconstruction upon the learned dictionary is expected to reject the noise to some extent. As soon as the reconstruction error reaches the noise threshold τ, it stops. Each pixel in the final denoised image is then calculated as the weighted average of all reconstructed patches covering that pixel (each with weight one) and the pixel value in the noisy image (with weight inversely proportional to the noise level $\lambda = \alpha/\sigma$).[1] See [Elad & Aharon 2006] for details on why, in some sense, this is an optimal recovering procedure.

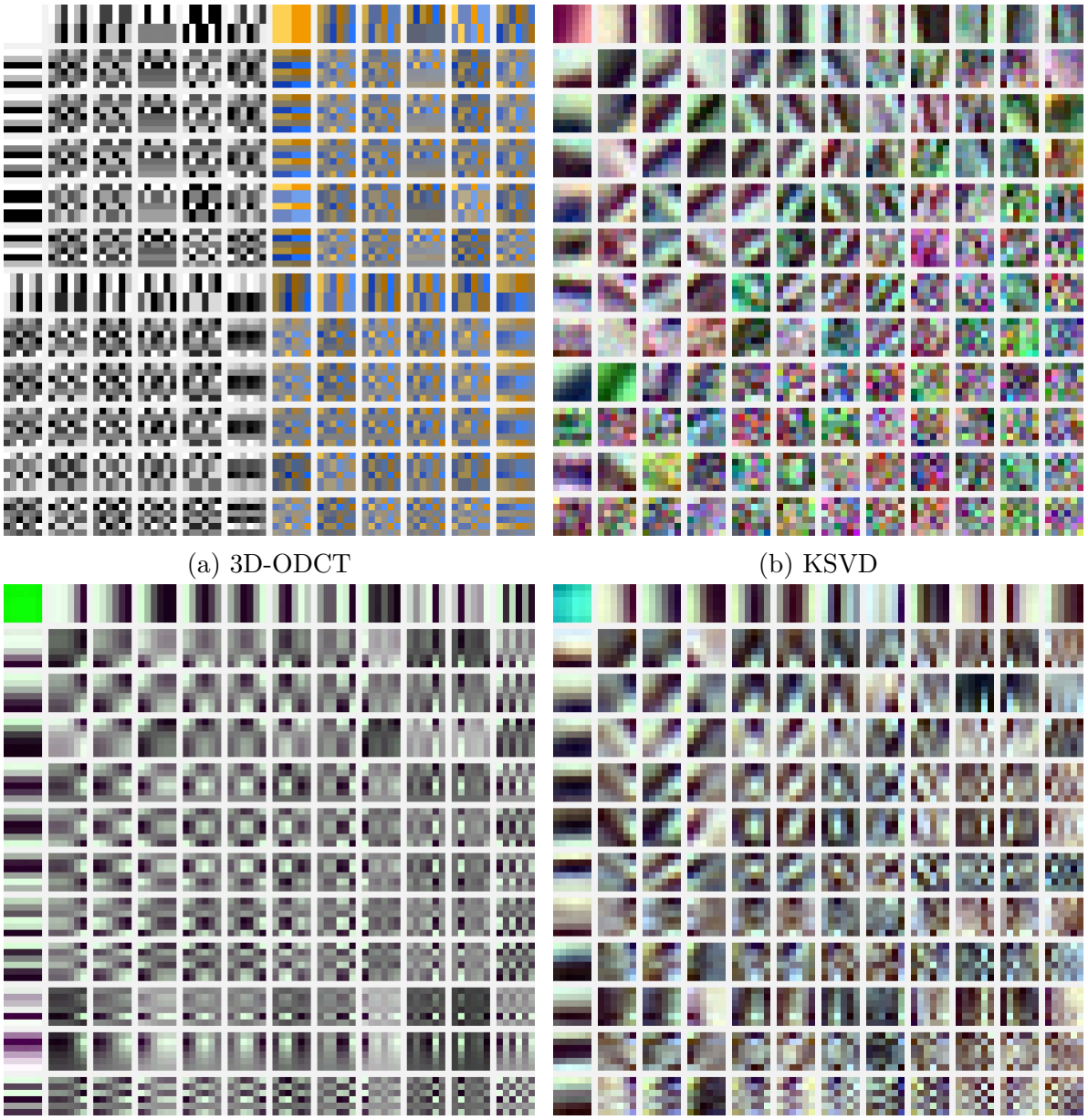
6.2.1 Visualizing the atoms

In figure 6.2 we show some examples of dictionaries learned from noisy color images. We compare a fixed DCT dictionary with some learned ones: the unstructure KSVD and the structured HO-SuKro dictionaries. The ODCT atoms, similarly to HO-SuKro of rank 1, are limited to purely vertical or horizontal patterns, due to their separable nature. The KSVD atoms, in turn, manage to capture richer patterns such as diagonal stripes in different angles. Note that HO-SuKro atoms also manages to represent such diagonal (as well as circular) patterns when R is increased.

We draw attention to the fact that such color atoms can be multiplied by negative coefficients when reconstructing an input signal, which inverts its colors. So, black turns into white (and vice versa), red becomes cyan, green becomes magenta and blue becomes yellow (basically switching from RGB to CMY colors).

In figure 6.3 and 6.4 we compare structured (HO-SuKro) and unstructured (KSVD) dictionaries in higher noise scenarios. Note that the KSVD atoms look considerably more noisy than the HO-SuKro ones, which indicates that the structure constraint may serve as a regularization which avoids overfitting (i.e. "fitting the noise"). This observation leads us to expect the structured dictionaries to achieve better results under higher noise.

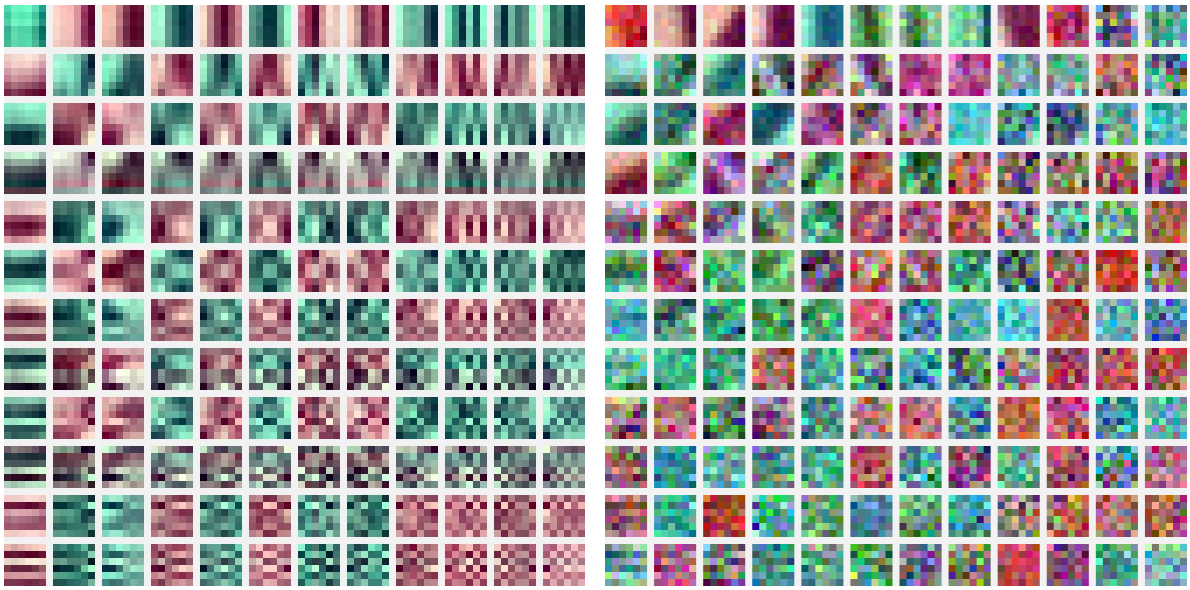
1. Unless stated otherwise, we set the proportionality constant to $\alpha = 0.1 \max_{px}$, following [Elad & Aharon 2006], where $\max_{px}$ denotes the maximum pixel value.



(a) 3D-ODCT (b) KSVD

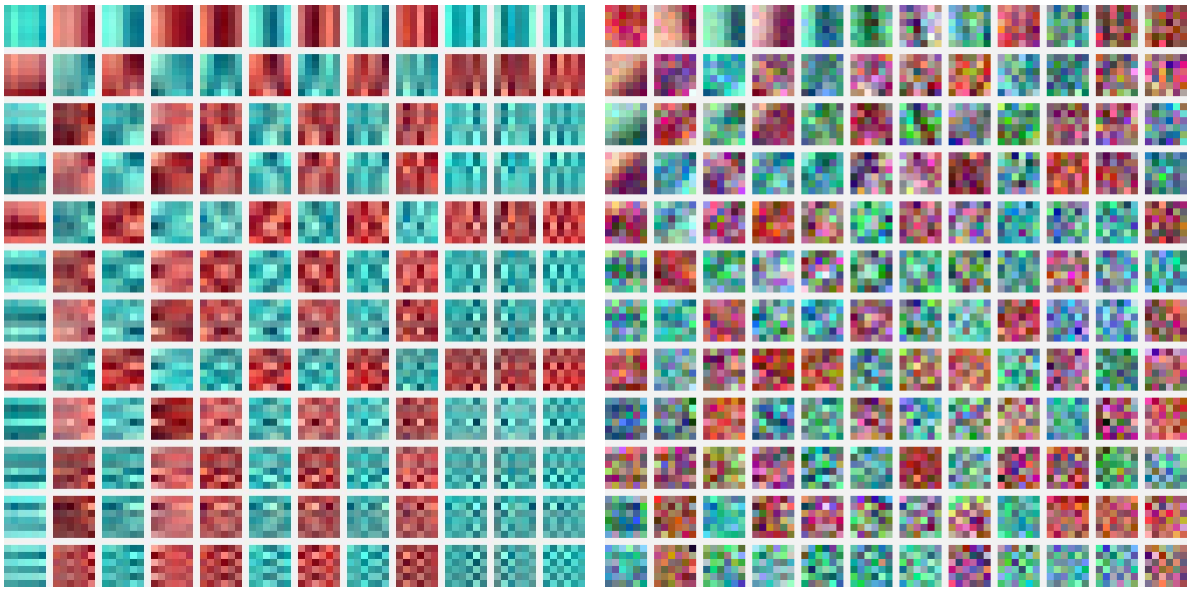
(c) HO-SuKro ($R = 1$) (d) HO-SuKro ($R = 10$)

Figure 6.2 – Examples of dictionaries learned on Lena color image, input PSNR 22.11dB



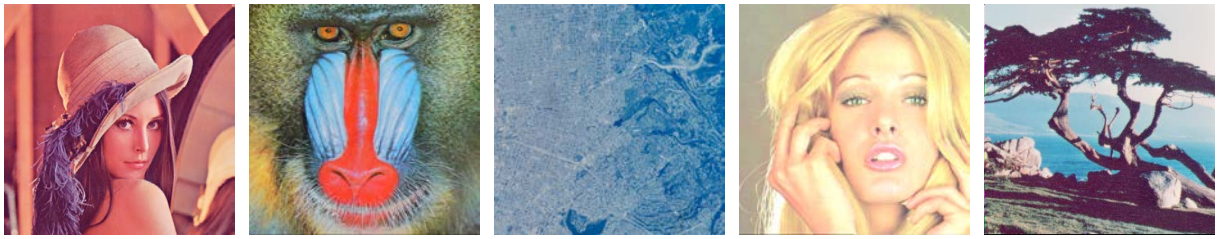
(a) HO-SuKro ($R = 3$) (b) KSVD

Figure 6.3 – Examples of dictionaries learned on Lena color image, input PSNR 10.65dB



(a) HO-SuKro ($R = 3$) (b) KSVD

Figure 6.4 – Examples of dictionaries learned on Lena color image, input PSNR 8.13dB



(a) Lena (b) Mandrill (c) Oakland (d) Tiffany (e) Tree

Figure 6.5 – Test color images from the USC-SIPI Image Database (available online at: `http://sipi.usc.edu/database/`)

6.3 Color image denoising

To evaluate and compare the proposed dictionary learning algorithms, we have performed some color image denoising experiments following the set-up described in section 6.2 (similar to the one introduced by [Elad & Aharon 2006]). The test images, listed in Figure 6.5, are color images of size $(N_{px1} \times N_{px2} \times N_{px3}) = (512 \times 512 \times 3)$, except for the Tree image which has $(256 \times 256 \times 3)$ pixels. In these experiments we thus have $K = 3$.

Table 6.1 summarizes the default simulation parameters. The dictionary is trained from a set of vectorized $(6 \times 6 \times 3)$-pixel patches. Note that a patch covers the entire third dimension of the image –which only contains three layers (red, green and blue). This means that the different patches are obtained by sliding only over the two first dimensions.

Table 6.1 – Simulation parameters

Sample dimension (n)	$n_1 \times n_2 \times n_3 = 6 \times 6 \times 3 = 108$
Number of atoms (m)	$m_1 \times m_2 \times m_3 = 12 \times 12 \times 6 = 864$
Training samples (N)	40000
Convergence tolerance (tol)	$\|\mathbf{D}\|_F \times 10^{-4}$
Iterations (N_{it})	20
Dictionary initialization ($\mathbf{D}_0$)	3D-ODCT[2]

The sparse coding step is performed by the OMP algorithm. The peak signal-to-noise

2. The 1-D $n \times m$ overcomplete DCT dictionary, as defined in [Rubinstein *et al.* 2010b], is a cropped version of the orthogonal $m \times m$ DCT matrix. The K-dimensional ODCT is the Kronecker product of K 1-D ODCT.

ratio (PSNR) of the reconstructed images is calculated as follows:

$$\text{PSNR} = \frac{255^2 N_{px}}{\left(\sum_{i=1}^{N_{px}} (\mathfrak{p}_i - \widehat{\mathfrak{p}}_i)^2\right)}$$

where 255 is the maximum pixel value, $N_{px} = N_{px1} N_{px2} N_{px3}$ is the total number of pixels on the input image, $\mathfrak{p}_i$ and $\hat{\mathfrak{p}}_i$ are respectively the i-th pixel value on the noiseless and reconstructed image. All results are averaged over five noise realizations.

In the presented results the dictionary is initialized with a 3D-ODCT, but a random initialization was tested and the denoising performance was practically equivalent.

6.3.1 Denoising performance

Impact of the separation rank (R)

Figures 6.6 to 6.9 show the denoised image PSNR (in dB) as a function of the number of Kronecker summing terms in the dictionary (i.e. the rank R of the rearranged tensor). We compare our results to an unstructured dictionary with the same size learned by the K-SVD [Aharon *et al.* 2006] algorithm and to the 3D-ODCT analytic dictionary, which is actually a 3-Kronecker dictionary as well (although not trained from the data). Compared to the fixed 3D-ODCT dictionary, our algorithm achieves considerably better denoising results, even for one single separable term ($R = 1$) which is the exact same structure as the 3D-ODCT. It also manages to overcome the unstructured K-SVD dictionary in many cases, especially under higher noise and as R grows. HO-SuKro has, in addition, the advantage of the reduced application complexity due to the Kronecker structure (see Table 6.3) when compared to K-SVD. The chosen structure proved well suited to this kind of application, since its introduction compromised very little the performance. The addition of separable terms tends to improve the performance until a certain point after which it saturates – or even deteriorates at higher noise, indicating the onset of overfitting. A bigger separable rank also implies higher learning and application complexities. Comparing both proposed algorithms for HO-SuKro, Algorithm 10 (denoted PGD for Projected Gradient Descent) and Algorithm 11 (denoted ALS), the latter shows a consistent performance superiority w.r.t. to the former, not to mention its significantly lower learning times as detailed in section 6.3.2.

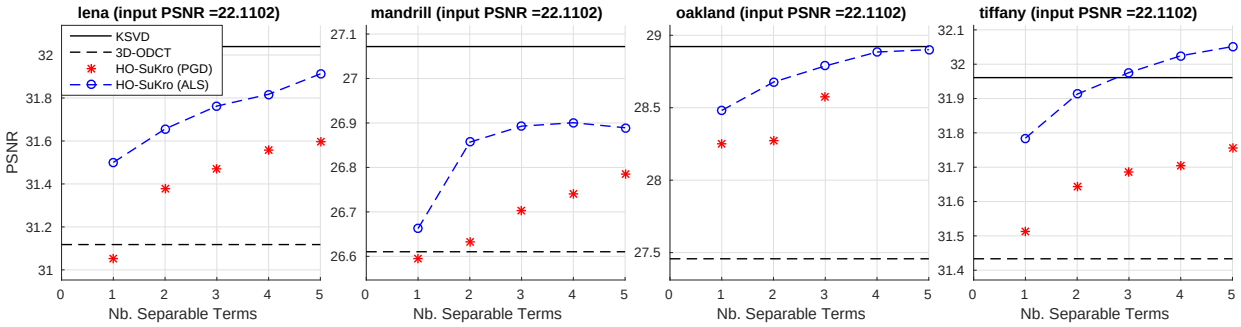


Figure 6.6 – PSNR vs. Separable rank (R) for all tested images (input PSNR = 22.11dB).

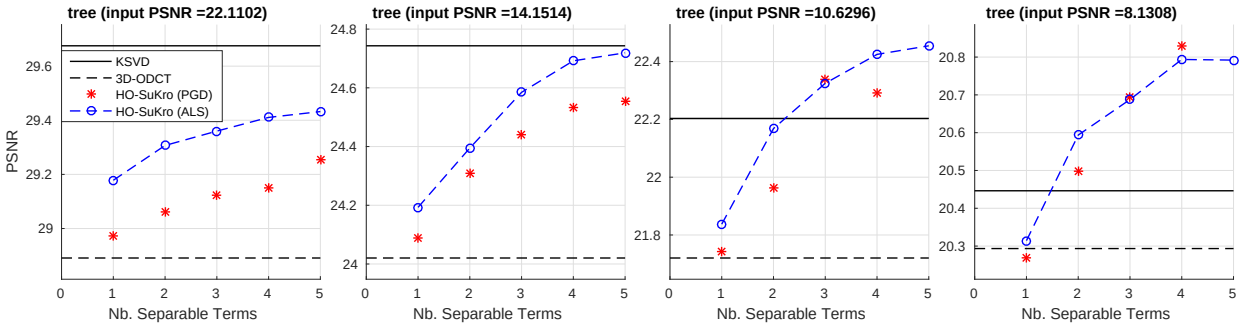


Figure 6.7 – PSNR vs. Separable rank (R) for the tree image.

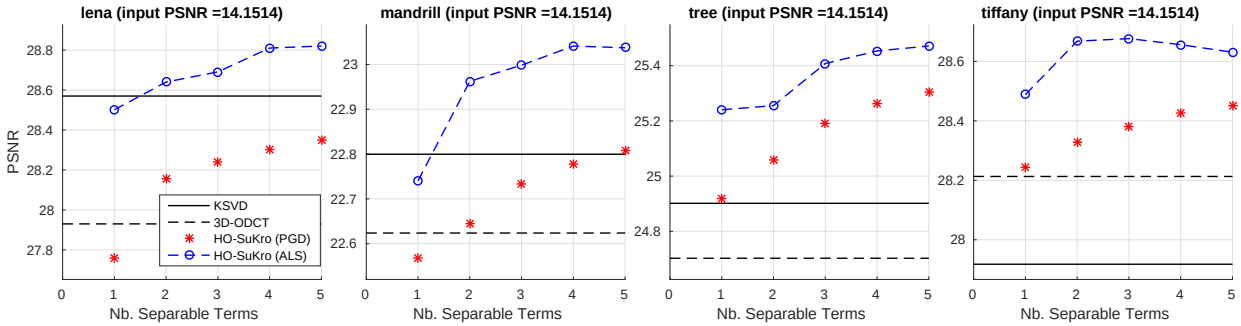


Figure 6.8 – PSNR for all tested images (input PSNR = 14.15dB) with bigger patches.

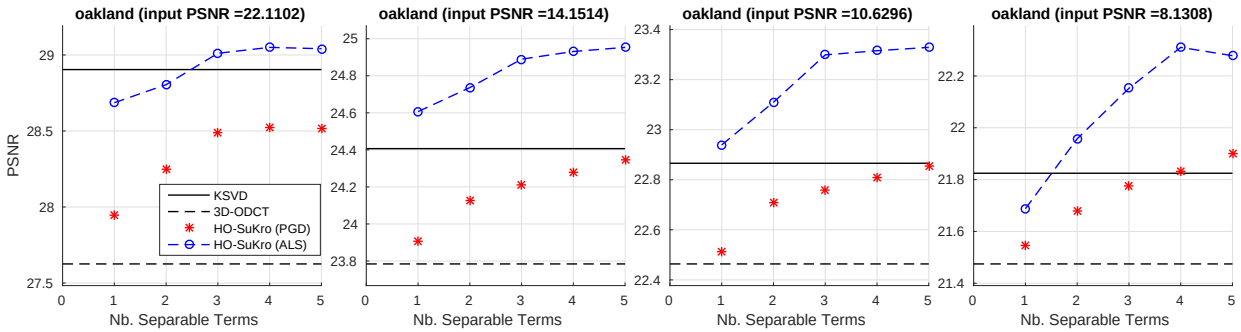


Figure 6.9 – PSNR vs. Separable rank (R) for the okland image with bigger patches.

Varying the noise level

Figures 6.7 and 6.9 show the denoising performances, respectively for images Tree and Oakland, under different noise standard deviations $\sigma = \{20, 50, 75, 100\}$ in pixel value (which corresponds to an input PSNR $= \{22.11, 14.15, 10.63, 8.13\}$ in decibel). These are standard noise values for benchmark in the literature [Elad & Aharon 2006]. The performance of the structured dictionaries are significantly enhanced as the noise level grows – not only HO-SuKro, but also ODCT. In the highest tested noise scenarios, HO-SuKro overcomes both ODCT and KSVD with as few as two separable terms. We attribute this superiority to an overfitting prevention brought by the structure constraint.

Impact of the patch size

In Table 6.2, we evaluate the impact of increasing the patch size. Besides the usual size $(n_1 \times n_2 \times n_3) = (6 \times 6 \times 3)$, we test a bigger patch configuration $(12 \times 12 \times 3)$. We show the results on Lena and Mandrill images. Similar results were obtained for the other test images (see Appendix B.3). Given the observed superiority of the ALS approach for HO-SuKro dictionaries, we report only its results in the table. We also restrict ourselves to a separable rank of $R = 3$, which represents a good complexity-performance compromise. This alternative configuration tends to increase the denoising performance, especially under higher noise: from 0.5dB to 3.5db on Lena image and from 0.1db to 1.5dB on Mandrill and Oakland. But on the down-side, using bigger patches comes with an additional cost in learning and denoising times for all the compared methods. This overhead can be alleviated by the structured dictionaries – in fact, we will see in section 6.3.2 that for bigger patches the practical speedups provided by HO-SuKro structure are more compelling.

Robustness to a reduction in the number training samples

In Figure 6.10 we evaluate the robustness of the learning algorithms to a reduction on the training set size. It shows the ΔPSNR, defined as the difference with respect to the PSNR obtained by ODCT. Note that the Kronecker-structured dictionaries become more competitive as the size of the training set decreases, to the point of consistently outperforming K-SVD for small enough training sets. This result goes in line with the theoretical results in [Shakeri *et al.* 2017a] suggesting a smaller sample complexity for KS dictionaries.

Table 6.2 – Output PSNR varying the patch size – Lena

	Algorithm	Patch size		Input SNR			
		n	m	22.11	14.15	10.63	8.13
Lena	ODCT	[6, 6, 3]	[12,12, 6]	31.12	26.82	23.68	20.82
		[12,12, 3]	[24,24, 6]	32.05	27.93	26.16	24.52
	K-SVD	[6, 6, 3]	[12,12, 6]	32.04	27.02	23.46	20.77
		[12,12, 6]	[24,24, 6]	**32.45**	28.57	26.52	24.49
	HO-SuKro ALS (R=3)	[6, 6, 3]	[12,12, 6]	31.76	27.03	23.41	21.07
		[12,12, 3]	[24,24, 6]	32.33	**28.69**	**26.75**	**24.90**
Mandrill	ODCT	[6, 6, 3]	[12,12, 6]	26.61	22.07	20.26	19.04
		[12,12, 3]	[24,24, 6]	26.83	22.62	20.93	19.84
	K-SVD	[6, 6, 3]	[12,12, 6]	27.07	22.71	20.68	19.28
		[12,12, 6]	[24,24, 6]	**27.16**	22.80	21.00	19.85
	HO-SuKro ALS (R=3)	[6, 6, 3]	[12,12, 6]	26.89	22.50	20.56	19.31
		[12,12, 3]	[24,24, 6]	27.00	**23.00**	**21.36**	**20.32**
Oakland	ODCT	[6, 6, 3]	[12,12, 6]	27.46	23.27	21.35	19.87
		[12,12, 3]	[24,24, 6]	27.63	23.78	22.46	21.47
	K-SVD	[6, 6, 3]	[12,12, 6]	28.92	24.29	22.10	20.45
		[12,12, 6]	[24,24, 6]	28.90	24.41	22.87	21.82
	HO-SuKro ALS (R=3)	[6, 6, 3]	[12,12, 6]	28.79	24.22	22.20	20.58
		[12,12, 3]	[24,24, 6]	**29.01**	**24.89**	**23.30**	**22.15**

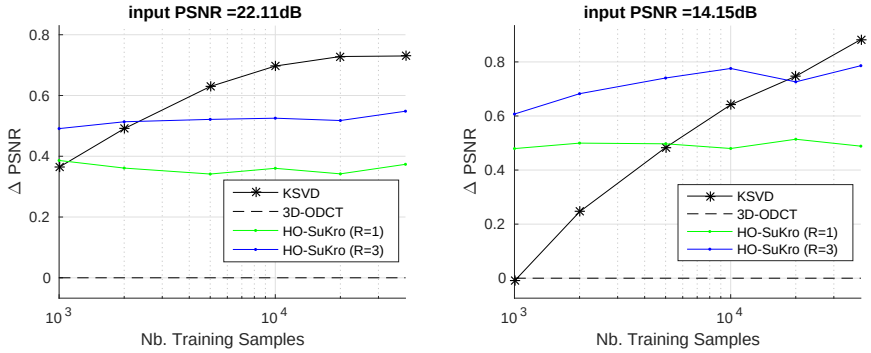


Figure 6.10 – PSNR vs. Number of training samples for the mandrill image.

Examples of denoised images

In Figure 6.11, we show some examples of denoised Lena images. It illustrates both a low noise (second row) and high noise (third row) case. Note that in the former case, the KSVD performs slightly better, while in the latter case HO-SuKro has an advantage. In both cases, the difference is very subtle visually, which testifies that the structure constraint, at least, doesn't harm the denoising.

6.3.2 Execution times

Before proceeding to actual execution times, let us first briefly examine the potential gains in this particular experimental setup. Table 6.3 shows the theoretical complexity savings provided by the Kronecker structure for matrix vector operations as well as its storage cost compared to an unstructured dictionary. The gains are around one and two orders of magnitude in this case. Obviously, there are multiple obstacles for the complexity gains to be translated into practice speedups, as discussed in section 5.4.3.

Table 6.3 – Complexity costs (for matrix-vector multiplications) and storage costs[3]

	HO-SuKro	Unstructured	Ratio (Unstructured/HO-SuKro)
Complexity (# operations)	$12960 \times R$	186624	$14.4/R$
Storage (# parameters)	$162 \times R$	93312	$576/R$

3. Complexity cost for HO-SuKro is obtained considering eq. (4.3). Storage cost is the total number of parameters in all factors: $R(\sum_k n_k m_k)$.

Original

Noisy (22.11dB) KSVD (32.04dB) HO-SuKro (31.72dB)

Noisy (8.13dB) KSVD (20.70dB) HO-SuKro (21.16dB)

Figure 6.11 – Examples of denoised images for input PSNR of 22.11dB and 8.13dB respectively at second and third rows.

Table 6.4 shows the execution times for the HO-SuKro and KSVD[4] dictionary learning algorithm. The running times are divided into two categories: training and learning times. ODCT is not included in the table because its training time is zero and its denoising time is basically equal to that of KSVD. The denoising times basically correspond to a series of runs of a sparse coding algorithm – in this case, OMP.

Note that the execution times are much higher when bigger patches are used. In this scenario, HO-SuKro structure manages to accelerate the denoising times by a factor of 2 to 4. This is not the case for smaller patches, because of the practical overheads discussed in section 5.4.3 which create a gap between the expected and the observed speedups in matrix-vector operation. As discussed then, these overheads are attenuated when the involved dimensions grow. In the reported denoising times, we used a pure Matlab implementation of OMP Cholesky – for both structured and unstructured dictionaries – in order to draw a fair comparison. Obviously, reported times can be reduced if a lower-level OMP implementation is used (for instance, a C implementation, as the one provided in the KSVD package, which manages to reduce the reported times in about one order of magnitude).

The training times, to the contrary, are not a fair comparison, since the KSVD algorithm has its core function efficiently implemented in C while both HO-SuKro learning algorithms are fully implemented in Matlab. Nevertheless, we still manages to remain roughly in the same order of magnitude as KSVD when using the ALS algorithm, which is by the way much faster than the PGD algorithm.

4. For the KSVD algorithm, we use the Matlab package provided by the authors. Available online at: `http://www.cs.technion.ac.il/~ronrubin/software.html`

Table 6.4 – Execution times for Lena image denoising with $\sigma = 20$.

Dimensions n	Dimensions m	Algorithm	Training time	Denoising time
$[6, 6, 3]$	$[12, 12, 6]$	KSVD	36 s	35 s
		HO-SuKro (R=1)	1500 s (PGD) 55 s (ALS)	38 s
		HO-SuKro (R=3)	1900 s (PGD) 180 s (ALS)	70 s
$[12, 12, 3]$	$[24, 24, 6]$	KSVD	290 s	460 s
		HO-SuKro (R=1)	$\sim$ 2 h (PGD) 245 s (ALS)	105 s
		HO-SuKro (R=3)	$\sim$ 3 h (PGD) 750 s (ALS)	190 s

6.4 Hyperspectral image denoising

Hyperspectral imaging has become a major image modality over the last years, largely thanks to the development of spectral sensors. Hyperspectral images collect reflectance spectra with significant spectral resolution ($\sim$200 wavelengths) for a large number of pixels ($\sim$1000$\times$1000 pixels). These images contain a lot of information regarding the composition of the scene, and can therefore be used in remote sensing for monitoring forest or coastal regions, or in chemometrics to study the composition of chemical compounds [Ghamisi *et al.* 2017].

However, Hyperspectral Images (HSI) are very often corrupted by various types of noise. In particular, for remotely acquired HSI, at least two kinds of noise are of importance: noise due to the spectral sensor sensitivity, and noise due to the swiping pattern of the sensors which yields stripes. Moreover, in the presence of clouds or other atmospheric perturbation, missing data may be present as well [Rasti *et al.* 2018]. In this work, we will suppose that HSI are only corrupted with anisotropic Gaussian noise in order to simplify the analysis of our denoising method, but other types of noise as well as missing data can be tackled with similar tools.

Removing noise in HSI is an important pre-processing step for any learning task such as segmentation, detection or spectral unmixing, and has therefore been studied extensively in the literature. The correlation along spectral bands in HSI have been found very useful

to improve HSI denoising, as opposed to conventional denoising techniques based solely on 2D modeling. As a result, modern HSI denoising techniques have evolved to incorporate spectral information. Our structured dictionary framework allows us to manipulated directly the 3D data and properly exploit its original structure. Patch-based approaches, like ours, have rarely been explored in the literature so far. For a more extensive survey of this domain, we refer the reader to [Rasti *et al.* 2018].

We now evaluate the proposed HO-SuKro dictionary learning approach in a Hyperspectral image (HSI) denoising task. In this section, we restrict ourselves to Algorithm 11 (ALS approach), which was demonstrated in section 6.3 to be more efficient than the projected gradient approach. We will initially compare our method to other techniques based on the sparse modeling assumption: fixed sparsifying transforms (2D and 3D [Basuhail & Kozaitis 1998] Wavelet) and patch-based unstructured learned dictionaries (K-SVD [Rubinstein *et al.* 2008]). We will see that we manage to outperform both of them. In the Hyperspectral imaging literature, another assumption besides sparsity proved to be very useful: the low-rankness of the image itself. Since, at first, we don't take this assumption into account, we don't manage to match the performance of state-of-the-art methods which combine both sparsity and low-rank assumptions. We still manage, nevertheless, to approach their performance, which is quite encouraging. The integration of the low-rank assumption in our model is done in section 6.5.

6.4.1 Simulation setup

The considered test hyperspectral images, shown in Figure 6.12, are listed below, along with their dimensions in number of pixels:

- *San Diego* ($400 \times 400 \times 158$),
- *Houston* ($349 \times 1905 \times 144$),
- *Washington DC* ($1280 \times 256 \times 191$),
- *Urban* ($307 \times 307 \times 162$).

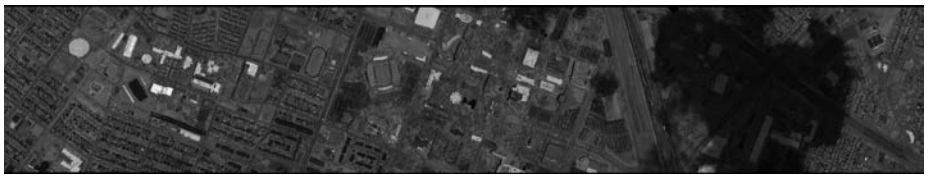
In this section, we use a cropped version of *San Diego* and *Houston* images with $256 \times 256 \times 100$ pixels, as shown in Figure 6.13. The full images are used in section 6.5.

Given a hyperspectral image $\boldsymbol{\mathcal{H}}$ of size $(N_{px1} \times N_{px2} \times N_{px3})$ corrupted with random Gaussian noise with standard deviation σ uniform over all spectral modes, we collect N 3D-patches with dimensions $\{n_1, n_2, n_3\}$ from the noisy image to form our training data

(a) San Diego

(b) Urban

(c) Washington DC



(d) Houston

Figure 6.12 – Test hyperspectral images (single spectral band visualization).

(a) San Diego

(b) Urban

Figure 6.13 – Cropped version of the test hyperspectral images.

$\mathbf{Y} \in \mathbb{R}^{n_1 n_2 n_3 \times N}$. A dictionary is learned from this data with $N_{\text{it}} = 20$ iterations of alternating optimization. The learning and denoising process follows the workflow described in section 6.2.

Reported results were obtained using 100000 training samples, which corresponds roughly to a 4-pixel step between adjacent patches. Although some performance improvement was observed when increasing this number, it does not compensate for the resulting raise in training time. Dictionaries were initialized with a 3D-ODCT. In HO-SuKro's $R > 1$ case, the remaining terms were initialized with unit-norm Gaussian random matrices, and revealed to be quite robust to the initialization. The ALS loop (line 7 in Alg. 11) was carried over until the update on the blocks $\mathbf{D}_k^r$ fell below a threshold empirically set to 10^{-1}, i.e.: $\frac{1}{R\sqrt{m}} \sum_r \|(\mathbf{D}_k^r)^{\text{old}} - (\mathbf{D}_k^r)^{\text{new}}\|_F < 10^{-1},\ \forall k$.

6.4.2 Denoising performance

In all experiments with HSI data we use the conventional signal-to-noise ratio (SNR) as a performance measure (instead of the PSNR used in the color image experiments) to comply with the usual practice in the HSI community. Denoting $\boldsymbol{\mathcal{H}}$ the noiseless image and $\hat{\boldsymbol{\mathcal{H}}}$ the denoised one, the SNR is calculated as follows:

$$\text{SNR} = \frac{\|\boldsymbol{\mathcal{H}}\|_F^2}{\|\boldsymbol{\mathcal{H}} - \hat{\boldsymbol{\mathcal{H}}}\|_F^2}$$

Table 6.5 shows the denoising performance of the proposed structured dictionaries compared to K-SVD as an unstructured counterpart. In order not to overburden the analysis, we report only the HO-SuKro results with rank $R = 3$, which we judged to represent a good performance-complexity compromise.

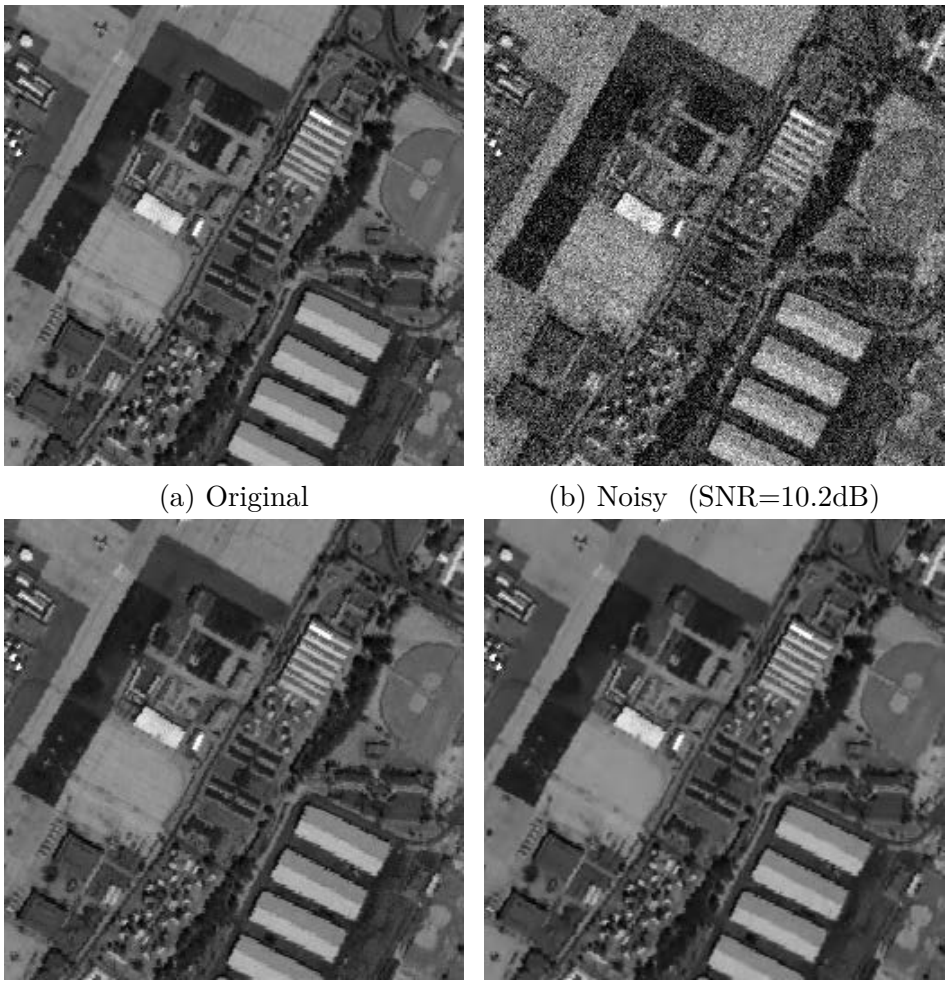
Increasing the patch size improves significantly the performance of HO-SuKro, while that of K-SVD doesn't benefit as much and may even deteriorate. This can be attributed to the growing number of parameters to estimate (as the dictionary size grows) making the number of available training data insufficient. HO-SuKro, thanks to its structured nature, avoids this issue.

More than simply increasing the patch size, what was empirically observed to drastically improve performance was increasing the patch dimension in the spectral mode, as shown in the last line of Table 6.5. Naturally, bigger patches also imply higher learning and denoising times. It is, thus, a compromise to be considered according to the available resources.

Table 6.5 – Output SNR (in dB) for various patch sizes – San Diego

Algorithm	Patch size		Input SNR [dB]			
	n	m	10	15	20	25
K-SVD	[6, 6, 6]	[12,12,12]	21.77	25.31	29.09	32.68
	[8, 8, 8]	[16,16,16]	21.51	25.19	29.2	32.95
	[10,10,10]	[20,20,20]	21.52	25.34	29.42	33.24
	[6, 6,20]	[12,12,20]	22.49	26.06	29.91	33.21
HO-SuKro ($R = 3$)	[6, 6, 6]	[12,12,12]	22.05	25.35	28.9	32.3
	[8, 8, 8]	[16,16,16]	22.73	25.82	29.22	32.64
	[10,10,10]	[20,20,20]	23.08	26.35	29.73	**33.27**
	[6, 6,20]	[12,12,20]	**24.10**	**27.07**	**30.09**	33.22

Table 6.6 compares HO-SuKro's performance to that of other techniques from the literature. By directly applying the proposed structured dictionary learning algorithm to Hyperspectral data, without any further domain-specific adaptation, we already obtain results comparable to the literature. Both HO-SuKro and K-SVD outperform the wavelet-based approaches (2D and 3D [Basuhail & Kozaitis 1998]), corroborating the interest of learning the sparsifying dictionary from data. HO-SuKro also consistently outperforms FORPDN [Rasti *et al.* 2014] which exploits the correlation within spectral bands. Naturally, we don't manage to reach state-of-the-art performance (HyRes [Rasti *et al.* 2017]) for this task, which makes use of a meaningful low-rank prior on the HSI. The performance gap is around 1.5dB and 3dB for *San Diego* and *Houston* images respectively. An example of denoised image is provided in Figure 6.14. A closer look reveals that our approach may over-smooth some details.



(a) Original (b) Noisy (SNR=10.2dB)

(c) HyRes (25.4dB) (d) HO-SuKro (24dB)

Figure 6.14 – Example of denoised images (100^{th} spectral band)

Table 6.6 – Output SNR [dB] comparison with literature

Image	Algorithm	Input SNR [dB]			
		10	15	20	25
San Diego	Wavelet 2D	14.75	18.00	21.70	25.92
	Wavelet 3D	23.11	26.04	28.91	31.68
	FORPDN	22.23	24.17	26.42	29.00
	HyRes	**25.38**	**28.60**	**31.75**	**34.70**
	HO-SuKro	24.10	27.07	30.09	33.22
Houston	Wavelet 2D	14.22	17.67	21.43	25.80
	Wavelet 3D	22.35	25.54	28.65	31.86
	FORPDN	22.80	25.46	28.09	30.74
	HyRes	**26.00**	**29.35**	**33.24**	**37.05**
	HO-SuKro	23.29	26.63	29.93	33.20

6.5 Improved Hyperspectral denoising

In this section, we propose to more efficiently denoise hyperspectral images under two assumptions:

(i) noiseless hyperspectral images in matrix form (i.e. vectorized in the spatial dimensions) are low-rank;

(ii) image patches are sparse in a proper representation domain defined by a dictionary.

These two assumptions have already led to state-of-the-art denoising methods using fixed Wavelet transforms [Rasti *et al.* 2018]. We propose to rather learn the dictionary from hyperspectral images, as done in the previous sections. We show that the dictionary learning approach is more efficient to denoise hyperspectral images than state-of-the-art methods with fixed dictionaries, at the cost of a larger computation time.

The difference with respect to the dictionary-based denoising approaches formerly presented is precisely the first assumption. We now incorporate this low-rank assumption which is a well-established prior for hyperspectral data processing [Manolakis *et al.* 2001] [Bioucas-Dias & Nascimento 2008], with proven efficiency in the denoising task [Rasti *et al.* 2017].

Although the HO-SuKro model is itself a form of low-rank constraint on the dictionary operator, the low-rank prior that we bring up here applies *directly to the data.*

Relation to prior art Dictionary Learning (DL) is well established in HSI applications such as unmixing [Castrodad *et al.* 2011] and classification [Chen *et al.* 2011]. It has also been previously applied to HSI denoising in several occasions (see [Xing *et al.* 2012] and references therein) but rarely coupled with a low-rank decomposition. Although the low-rankness of the HSI is enforced via nuclear norm in [Zhao & Yang 2015], the dimensionality reduction promoted by an explicit low-rank factorization is not exploited at all as the full-size HSI is manipulated. Moreover, in [Zhao & Yang 2015], like in most DL methods for traditional image denoising [Elad & Aharon 2006], the noise variance is supposed known, while here we use an efficient heuristic to estimate it.

In what follows, we first formalize the denoising task and the low-rank and image sparsity assumptions in section 6.5.1. In section 6.5.2 we present our generic DL-based algorithm for HSI, and instantiate it with two DL algorithms, K-SVD [Aharon *et al.* 2006]

and SuKro [Dantas *et al.* 2017]. Finally, in section 6.5.3, we show on artificially noised HSIs that the proposed approach outperforms state-of-the-art methods for HSI denoising at the cost of increased computation time that is reasonable for offline applications.

6.5.1 Sparse and Low-rank Modeling

A hyperspetral image is composed of two spatial dimensions (image space) and one spectral dimension, and thus naturally represented as a 3 dimensional cube of data, say $\boldsymbol{\mathcal{H}} \in \mathbb{R}^{h_1 \times h_2 \times p}$, where h_1 and h_2 correspond to the spatial dimensions and p is the number of spectral bands.

Nevertheless, a hyperspectral image is also often represented in its matrix (2 dimensional) form by vectorizing its spatial dimensions at each spectral band. In an additive noise model, this leads to:

$$\mathbf{H} = \mathbf{Z} + \mathbf{N} \tag{6.1}$$

where $\mathbf{H} \in \mathbb{R}^{h \times p}$ (with $h = h_1 h_2$) containing as its j-th column the vectorized observed image at band j. $\mathbf{Z}$ and $\mathbf{N}$ are respectively the noise-free unknown signal, and the noise matrix, both $(h \times p)$ matrices.

Low-rank assumption

An HSI $\mathbf{Z}$ is commonly modeled as a low-rank matrix in the literature [Golbabaee & Vandergheynst 2012, Rasti *et al.* 2017]. The classical linear mixture model [Manolakis *et al.* 2001] is itself a low-rank model $\mathbf{Z}{=}\mathbf{A}\mathbf{S}^T$, in which the image is decomposed into a few subregions called source images (as the columns of $\mathbf{A}$) each containing a certain material with distinct spectral signature (in the columns of $\mathbf{S}$). The HSI rank R in this case is the number of materials present in the image.

The singular value decomposition (SVD) is a way of obtaining an analogous low-rank decomposition of $\mathbf{Z}$, even though it does not, in principle, promote spectral unmixing. It leads to the noisy low-rank model

$$\mathbf{H} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T + \mathbf{N} \tag{6.2}$$

where the columns of $\mathbf{U}, \mathbf{V} \in \mathbb{R}^{h \times R}$ are respectively a set of vectorized *eigen-images* and the associated spectral components, and $\boldsymbol{\Sigma} \in \mathbb{R}^{R \times R}$ is a diagonal singular value matrix.

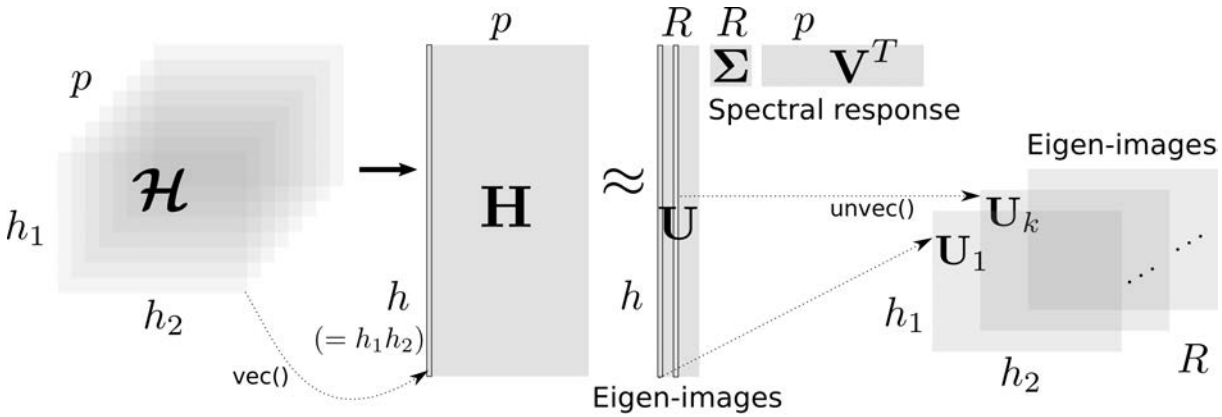


Figure 6.15 – Hyperspectral image modeling: 3D data cube (left), 2D representation with vectorized spatial dimensions (center) and low-rank factorization model (right).

Figure 6.15 illustrates the described models and notations.

Sparsity assumption

The eigen-images $\mathbf{U}_r = \text{unvec}_{h_1,h_2}(\mathbf{u}_r)$ (with $\mathbf{u}_r$ the r-th column of $\mathbf{U}$) can, then, be sparsely represented in a well-suited base –or, more generally, representation domain. In the noisy case, sparse reconstruction thus works as a form of denoising. Indeed, sparse modeling for images has been used for decades now [Bruckstein *et al.* 2009, Elad & Aharon 2006], with very good performance in denoising tasks.

State-of-the art techniques in Hyperspectral image restauration [Rasti *et al.* 2017] use a 2-D (Kronecker) orthogonal wavelet basis $\mathbf{B}$ in which the vectorized eigen-images $\mathbf{u}_r$ are represented as a sparse set of wavelet coefficients $\mathbf{w}_r$ (r-th column of sparse coefficient matrix $\mathbf{W}$): $\mathbf{U} = \mathbf{BW}$. The full model becomes:

$$\mathbf{H} = \mathbf{BW\Sigma V}^T + \mathbf{N} \tag{6.3}$$

A typical processing pipeline

In view of model (6.3), a usual denoising approach given the corrupted signal $\mathbf{H}$ –which is no longer low-rank due to the noise– consists in estimating $\hat{\mathbf{U}}$,$\hat{\mathbf{\Sigma}}$ and $\hat{\mathbf{V}}$ via SVD truncation and futher denoising $\hat{\mathbf{U}}$ via 2D-Wavelet shrinkage. The spectral component of the low-rank model $\hat{\mathbf{V}}$ can also be denoised (for instance, [Chen & Qian 2011] uses a 1D-Wavelet shrinkage).

6.5.2 Proposed Approach

A first part of our contribution consists in replacing the classic orthogonal Wavelet basis in the described pipeline by a dictionary both 1) overcomplete and 2) learned from data.

Overcomplete learned dictionary

Both the overcompleteness and the fact of learning a dictionary from a targeted type of data has been shown to improve the performance of natural image denoising tasks [Elad & Aharon 2006] when compared to the classical orthogonal Wavelet-based approach. Any existing dictionary learning approach could be used for our purpose here. We propose to use two methods: K-SVD as a standard (baseline) algorithm for learning unstructured dictionaries and the proposed Kronecker-structured dictionaries. The usual Wavelet approaches (e.g. in the state-of-the-art HyRes [Rasti *et al.* 2017]) are themselves Kronecker-structured, although not learned from data.

An important drawback of the overcompleteness (and consequently, non-orthogonality) of the dictionary can be pointed out: it significantly complicates the estimation of the sparse coefficients when compared to the orthogonal case (for instance, in HyRes [Rasti *et al.* 2017], this step comes down to a simple thresholding operation). Here, we will need to resort to the mentioned sparse coding algorithms as a sub-optimal way of solving this –now ill-posed– problem.

Consequently, computational complexity (and running time) is significantly bigger for the proposed method when compared to the literature. However, we will see in section 6.5.3 that a performance gain is obtained. Therefore, except in very constrained cases, the proposed technique is worthy.

A (slightly different) patch-based denoising approach

In order to avoid having a prohibitively large dictionary $\mathbf{D} \in \mathbb{R}^{h \times m}$ (with $m > h$), which would be very hard to train, we rely on a patch-based approach like before. An important difference, however, is that the patches are no longer extracted from the HSI in its original 3D-form ($\boldsymbol{\mathcal{H}}$) but rather from the eigen-images $\mathbf{U}_r$. Therefore, strictly speaking, we are now denoising a set of 2D-images.

Remark 2. *Now, differently from section 6.4, we are dealing with 2D-data patches, i.e. $K = 2$. The HO-SuKro model boils down to what we call the SuKro model [Dantas et al. 2017], which consists of sums of Kronecker products between two matrices.*

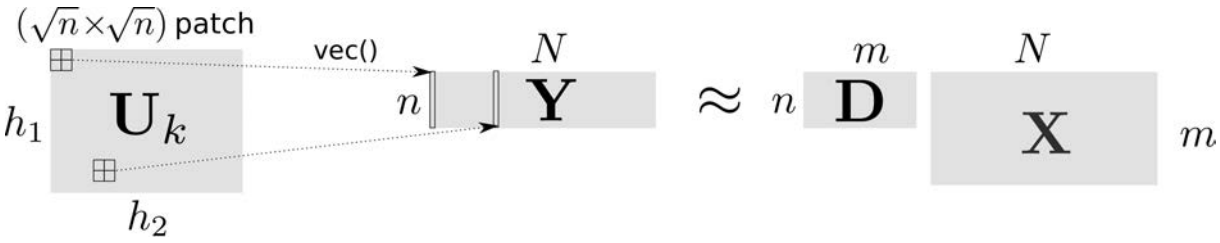


Figure 6.16 – N patches $(\sqrt{n} \times \sqrt{n})$ are extracted from a eigen-image $\mathbf{U}_r$ to form the training data matrix $\mathbf{Y}$, which is used to learn a dictionary $\mathbf{D}$ and a sparse reconstruction matrix $\mathbf{X}$.

Given an eigen-image $\hat{\mathbf{U}}_r \in \mathbb{R}^{h_1 \times h_2}$ we (sub)sample N equally-spaced training patches with dimensions $(\sqrt{n} \times \sqrt{n})$. The patches are vectorized and stacked as the columns of a training matrix $\mathbf{Y} \in \mathbb{R}^{n \times N}$., from which the dictionary is learned.

The proposed denoising approach is described in Algorithm 12. Note that a new dictionary is learned for each eigen-image $\mathbf{U}_r$. To fully characterize the algorithm, some elements are still to be defined: the rank and noise estimation, sparse coding, dictionary update and denoising functions (respectively in lines 4, 5, 13, 14 and 16) are further detailed below. Some of these steps are very similar to what have been done Although some of these steps are very similar to what has been done in the previous sections, we still go through each of them while highlighting the main differences.

Dictionary Learning Algorithms

The customary alternating minimization DL framework is used. The *sparse coding* and *dictionary update* are briefly specified below.

Sparse coding Like previously, we use the OMP algorithm with a stopping criterion based on the energy of the residual. The error threshold τ is proportional to the noise standard deviation σ. An important difference is that, here, we estimate the noise level from the data at hand.

Dictionary update This is the step which distinguishes one DL algorithm from another. The two adopted dictionary learning approaches are:

1. K-SVD: as described in section 1.5.1.

2. SuKro: to learn structured dictionaries which can be written as a sum of a few

Algorithm 12 Dictionary-based HSI denoising approach

INPUT: Noisy hyperspectral image $\mathbf{H}$
▷ Low-rank step
$[\mathbf{U}, \mathbf{\Sigma}, \mathbf{V}] = \text{SVD}(\mathbf{H})$
$R = \text{EstimateRank}(\mathbf{\Sigma})$
$[\sigma_1, \ldots, \sigma_R] = \text{EstimateNoise}(\mathbf{\Sigma})$
$\hat{\mathbf{U}} = \mathbf{U}(:, 1:R)$, $\hat{\mathbf{V}} = \mathbf{V}(:, 1:R)\mathbf{\Sigma}(1:R, 1:R)$
▷ *Sparse step*
for $r = [1, \ldots, R]$ **do**
    Extract N patches from $\hat{\mathbf{U}}_r$ to form $\mathbf{Y}$
    ▷ *Dictionary learning*
    Initialize $\mathbf{D}_0$
    for $i = [1, \ldots, N_{\text{it}}]$ **do**
        $\mathbf{X}_i = \text{SparseCoding}(\mathbf{Y}, \mathbf{D}_{i-1}, \sigma_r)$
        $\mathbf{D}_i = \text{DictionaryUpdate}(\mathbf{Y}, \mathbf{X}_i)$
    end for
    $\hat{\mathbf{U}}_r = \text{Denoise}(\hat{\mathbf{U}}_r, \mathbf{D}_{N_{\text{it}}}, \sigma_r)$
end for
$\hat{\mathbf{U}} = [\hat{\mathbf{U}}_1, \ldots, \hat{\mathbf{U}}_R]$
OUTPUT: Recovered image $\hat{\mathbf{Z}} = \hat{\mathbf{U}}\hat{\mathbf{V}}^T$

Kronecker products of smaller matrices $\mathbf{D}_k^r \in \mathbb{R}^{\sqrt{n}\times\sqrt{m}}$.

$$\mathbf{D} = \sum_{r=1}^{R} \mathbf{D}_1^r \boxtimes \mathbf{D}_2^r. \tag{6.4}$$

For the SuKro dictionary update step we use the Alternating Least-Squares (ALS) approach proposed in Algorithm 11. It was originally proposed for a higher-order generalization of SuKro, i.e. for sums of Kronecker products of any number of blocks, but here we apply it for the particular case of two blocks only (i.e. $K = 2$).

Computation of the denoised eigen-image Once the learning process is finished, the denoised version of the eigen-image $\hat{\mathbf{U}}_r$ is computed by following the procedure described in section 6.2 (same as before)[5]. Note that this learning-then-denoising procedure is repeated for each of the R eigen-images, $\mathbf{U}_r, \forall r \in \{1, \ldots, R\}$.

5. A small modification comes from the fact that the eigen-images are no ordinary images, as they stem from the low-rank decomposition of the matricized HSI. As such, they do not have a fixed range for its entries (i.e. "pixel values"). Recall that in the patch-averaging process for calculating the final denoised image, we considered the maximum pixel value with a weight λ proportional to the maximum pixel value (see section 6.2). Instead, we now set $\lambda = \alpha/\sigma$ with $\alpha = 1/\sqrt{h}$.

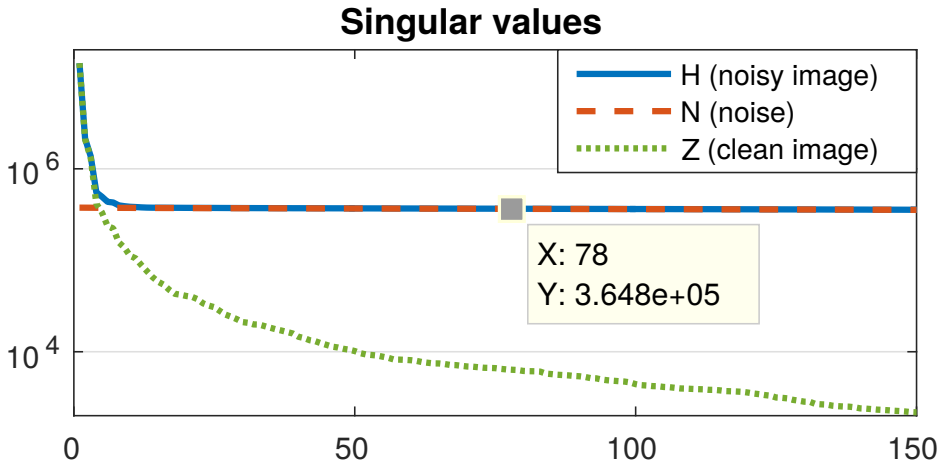


Figure 6.17 – Singular values of the San Diego HSI before (dotted green line) and after the noise addition (full blue line), as well as the noise singular values (dashed red) in logarithmic scale. Note that the noise singular values are very close to a constant $\sigma\sqrt{h}$ ($= 912 \times 400 = 3.648 \cdot 10^5$) as predicted by the theory.

Noise and Rank Estimation

In this subsection we show how the noise level can be efficiently estimated in practice under the assumption that the original image is low-rank and for a Gaussian distribution on the entries of the noise matrix $\mathbf{N}$.

First of all, as $\frac{h}{p} \gg 1$, the singular values of the noise matrix $\mathbf{N}$ are very close to a constant. Indeed, using the limit case of the Marčenko-Pastur distribution [Marčenko & Pastur 1967] when the column dimension is large in front of the row dimension, we get that all singular values of $\mathbf{N}$ are close to $\sigma\sqrt{h}$.

Moreover, as both the noiseless matrix $\mathbf{Z}$ and the noise $\mathbf{N}$ are tall matrices generated by *a priori* independent processes, they are likely to be almost orthogonal to each other. Then,

$$\mathbf{H}^T\mathbf{H} \approx \mathbf{Z}^T\mathbf{Z} + \mathbf{N}^T\mathbf{N} \approx \mathbf{Z}^T\mathbf{Z} + \sigma^2 h\mathbf{I} \tag{6.5}$$

and we get that the eigenvalues of $\mathbf{H}^T\mathbf{H}$ are the sum of those of $\mathbf{Z}^T\mathbf{Z}$ and a constant term $\sigma^2 n$.

Finally, since $\mathbf{Z}$ is (approximately) a rank R matrix, any singular value of $\mathbf{H}$ after the R-th index is (approximately) equal to $\sigma\sqrt{h}$. Therefore, the noise level can be estimated by looking at the tail value ς_{tail} of the singular values of $\mathbf{H}$. Figure 6.17 shows that we indeed observe such a singular value profile in practice when adding Gaussian noise to a

real hyperspectral image.

While $\hat{\sigma} = \varsigma_{\text{tail}}/\sqrt{h}$ estimates the noise level in $\mathbf{H}$, we are rather interested in the noise level σ_r at an eigen-image $\mathbf{U}_r$. Considering that each column $\mathbf{u}_r$ of $\mathbf{U}$ is scaled to unit-norm regardless of the associated singular value (denoted ς_r for the r-th largest singular value), we observed that σ_r is inversely proportional to ς_r and can be estimated as follows:

$$\hat{\sigma}_r = \sigma/\varsigma_r = \varsigma_{\text{tail}}/(\varsigma_r\sqrt{h}) \tag{6.6}$$

The rank estimation is based on this same observation: the singular values saturate as the noise dominates. A suitable approximate rank R is obtained by detecting this saturation. We set $\hat{R}$ equals to the first singular value ς_r such that $(\varsigma_r - \varsigma_{\text{tail}})/\varsigma_{\text{tail}} < \epsilon$.

$$\text{EstimateRank}(\mathbf{\Sigma}) = \min\{r \mid (\varsigma_r - \varsigma_{\text{tail}})/\varsigma_{\text{tail}} < \epsilon\} \tag{6.7}$$

where ϵ is a threshold empirically calibrated to $\epsilon = 3\times10^{-2}$.

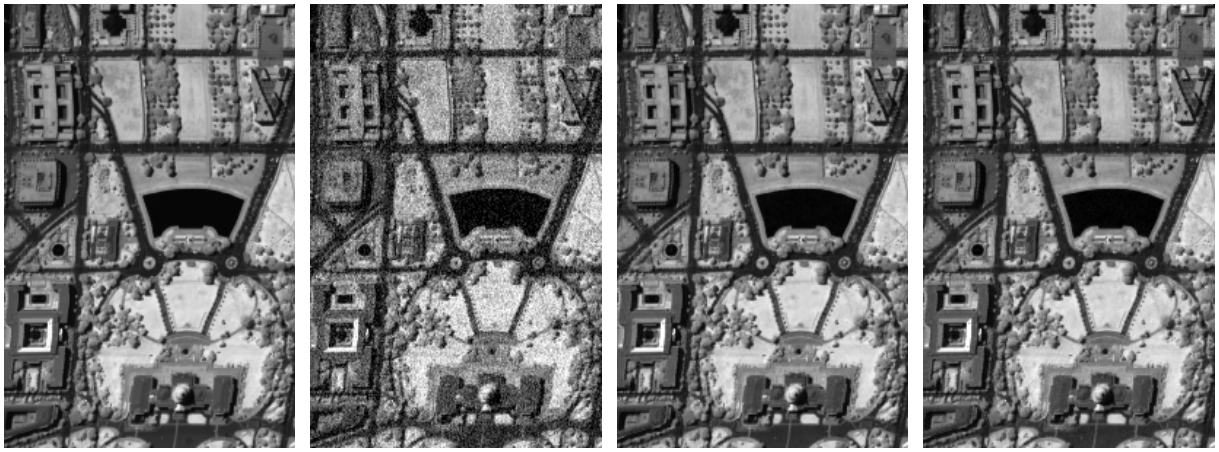
6.5.3 Experimental results

Four hyperspectral images were used in the experiments with $(h_1 \times h_2 \times p)$ pixels: *San Diego* $(400 \times 400 \times 158)$, *Houston* $(349 \times 1905 \times 144)$, *Washington DC* $(1280 \times 256 \times 191)$, *Urban* $(307 \times 307 \times 162)$. The images were corrupted with additive white Gaussian noise with standard deviation σ uniform over all spectral modes and adjusted to give the desired input signal-to-noise ratio (SNR).

For each eigen-image (columns of $\mathbf{U}_r$), we collect $N = 20000$ patches with (6×6) pixels from the noisy image to form the training data. A dictionary is learned from this data with $N_{\text{it}} = 20$ iterations of alternating optimization. Dictionaries are initialized with a 2D-ODCT. In the case of SuKro, the remaining terms are initialized with unit-norm Gaussian random matrices, with no relevant impact on its performance.

The results reported in Table 6.7 are averaged over 10 runs with independent noise realizations. Standard deviations are usually around 0.005dB (and never bigger than 0.02dB).

Three variations of the proposed method are tested, with three different types of dictionary: K-SVD unstructured learned dictionary, SuKro learned structured dictionary with $R = 3$ terms and an overcomplete but fixed ODCT dictionary (i.e. the learning process is skipped). The proposed methods are compared to: a Wavelet 3D approach [Basuhail & Kozaitis 1998] which uses only the sparsity prior by applying a 3D-wavelet basis directly in



(a) Original (b) Noisy: 10dB (c) SuKro (d) HyRes

Figure 6.18 – Zoom in Washington image (100^{th} spectral band).

the data cube $\boldsymbol{\mathcal{H}}$; and the HyRes [Rasti *et al.* 2017] method, which combines both sparse and low-rank assumptions. The latter is a state-of-the-art approach (see [Rasti *et al.* 2017] for a detailed comparison of this method with other methods in the literature).

The proposed methods with K-SVD and SuKro dictionaries consistently outperform the state-of-the-art HyRes approach by about 1.5dB. A performance gain of about 1.3dB is already obtained by replacing the orthogonal Wavelet basis in HyRes by a fixed over-complete DCT dictionary allied to the patch-based procedure. The learning process then brings an additional 0.2dB in denoising performance. A visual comparison is provided in Figure 6.18.

Execution times

Execution times are reported in Table 6.8. The literature methods HyRes and Wavelet 3D take about one and two minutes respectively, independently of the noise level. The running times for the proposed method, on the other hand, increase as the noise level decreases. That's because the OMP reconstruction threshold gets smaller, requiring more iterations. The execution times range from 120 to 760s in the ODCT case and from 260 to 1600s with a learned dictionary (SuKro being slightly faster than K-SVD). Therefore, the proposed technique with dictionary learning takes around 5 to 30 times longer than HyRes. We point out that, for the proposed dictionary-based approaches, the reported times include both learning and denoising phases.

Fewer training samples

A possible way to reduce the execution times of the proposed method consists in reducing the number of training samples N used in the dictionary learning process, at the expense of some denoising performance. In Table 6.9 we show the time and performance variation when reducing from 20000 to 5000 the number of training samples. A total time reduction of around 40 to 50% is obtained, while the denoising performance is not severely affected. Note that SuKro is more robust to the reduction in the training set size due to its structured nature (less parameters to estimate) as already previously observed in the color image experiments.

Table 6.7 – Output SNR [dB] comparison with literature

	Method	Input SNR [dB]					
		5	10	15	20	25	30
San Diego	Wav. 3D	20.19	23.36	26.51	29.48	32.33	35.08
	HyRes	23.26	26.37	29.63	32.61	35.44	37.92
	ODCT	24.79	27.77	30.79	33.77	36.57	38.88
	K-SVD	24.92	27.86	30.89	33.88	36.68	**39.00**
	SuKro	**24.95**	**27.93**	**30.94**	**33.89**	**36.69**	38.99
Houston	Wav. 3D	18.28	21.59	24.90	28.19	31.45	34.64
	HyRes	22.86	26.45	29.76	33.00	36.08	39.49
	ODCT	24.18	27.49	30.87	34.13	37.30	39.85
	K-SVD	24.38	**27.64**	**31.01**	**34.27**	**37.45**	**39.99**
	SuKro	**24.39**	27.63	30.98	34.23	37.42	39.97
Washington	Wav. 3D	18.87	21.92	25.03	28.25	31.51	34.77
	HyRes	23.25	26.54	29.78	33.30	36.35	38.76
	ODCT	24.58	27.92	31.27	34.57	37.47	39.80
	K-SVD	24.74	28.04	31.37	34.68	**37.59**	**39.89**
	SuKro	**24.76**	**28.05**	**31.38**	**34.69**	**37.59**	39.88
Urban	Wav. 3D	18.37	21.58	24.78	27.85	30.82	33.68
	HyRes	22.02	25.38	28.45	31.29	33.95	36.01
	ODCT	23.34	26.52	29.54	32.34	34.76	36.54
	K-SVD	**23.45**	**26.63**	**29.65**	**32.44**	**34.86**	**36.61**
	SuKro	**23.45**	**26.63**	**29.65**	32.43	34.83	36.58

Table 6.8 – Execution times (in seconds) for Washington image

	Method	Input SNR [dB]					
		5	10	15	20	25	30
Washington	Wav. 3D	125	120	125	125	120	124
	HyRes	55	50	50	49	48	50
	ODCT	128	201	296	408	533	761
	K-SVD	273	445	677	980	1315	1650
	SuKro	262	394	592	857	1143	1433

Table 6.9 – SNR variation (in dB) and execution time variation (in %) w.r.t. the ones displayed in Tables 6.7 and 6.8 respectively, for 5000 training samples instead of 20000.

	Δ	Method	Input SNR [dB]					
			5	10	15	20	25	30
Washington	Time	K-SVD	-49%	-50%	-50%	-49%	-50%	-46%
		SuKro	-45%	-42%	-42%	-43%	-44%	-44%
	SNR	K-SVD	-0.06	-0.05	-0.04	-0.03	-0.05	-0.04
		SuKro	-0.01	-0.02	-0.01	-0.01	-0.04	-0.02

Part III

Stable Safe Screening

Chapter 7

Accelerating ℓ_1-minimization

Sparsity-constrained linear regression has found numerous applications in signal processing and machine learning, tackling under-determined inverse problems. These appear in many forms such as image or audio inpainting [Adler *et al.* 2012], source localization [Malioutov *et al.* 2005], spike deconvolution [Dossal & Mallat 2005], or sparse variable selection for gene expression [Gui & Li 2005], to cite only a few.

As reviewed in chapter 2, there are many computational approaches to estimate the sparse coefficient vector in such settings with two main families: greedy algorithms [Mallat & Zhang 1993] [Pati *et al.* 1993], and convex optimization approaches relying on ℓ_1-norm minimization, where the regression problem is expressed as an ℓ_1 regularized optimization problem –known as basis pursuit [Chen *et al.* 1998] or lasso [Tibshirani 1996]– which solution is computed using iterative convex optimization algorithms such as ISTA or its variants [Daubechies *et al.* 2004, Beck & Teboulle 2009, Bioucas-Dias & Figueiredo 2007, Wright *et al.* 2009, Chambolle & Pock 2011].

For very high-dimensional problems, however, iterative algorithms to solve ℓ_1 minimization problems can become computationally prohibitive, which is why accelerating techniques are still an intense research topic. This part of this thesis demonstrates how to combine two such techniques:

1. Fast structured operators, which provide faster matrix-vector products (see section 1.5.2 and chapter 5);

2. Safe screening tests, which safely eliminate unused dictionary atoms (see chapter 8).

> The following chapters (from 8 to 11) are based on the journal paper [Dantas & Gribonval 2019a], which synthesizes and extends the results in [Dantas & Gribonval 2017] and [Dantas & Gribonval 2018].

In [Dantas & Gribonval 2017] we introduced a safe screening sphere test that manipulates an approximate dictionary and in [Dantas & Gribonval 2018] the GAP Safe sphere

test [Fercoq *et al.* 2015] was extended to this new setting. In [Dantas & Gribonval 2019a], we revisited the previous results under a broader formalism (presented in chapter 9) and proposed a fast algorithm for ℓ_1-minimization which combines safe screening and (potentially multiple) fast approximate dictionaries (described in chapter 10). Experimental results are given in chapter 11.

7.1 Sparsity constrained regularization

Let n and m be respectively the dimension of the observed vector and that of the unknown coefficient vector. The observed vector is denoted $\mathbf{y} \in \mathbb{R}^n$ and modeled as $\mathbf{y} \approx \mathbf{D}\mathbf{x}_0$ where $\mathbf{x}_0 \in \mathbb{R}^m$ is sparse and $\mathbf{D} = [\mathbf{d}_1, \ldots, \mathbf{d}_m] \in \mathbb{R}^{n \times m}$. In the context of linear inverse problems, $\mathbf{D}$ is the so-called measurement matrix [Foucart & Rauhut 2013], while for sparse signal representations $\mathbf{D}$ would be the dictionary matrix [Chen *et al.* 1998], and in statistics the design matrix [Tibshirani 1996]. Here, we adopt the terminology of sparse signal representations hence $\mathbf{D}$ is called a dictionary and its columns, denoted $\mathbf{d}_j \in \mathbb{R}^n$, are called atoms.

The ℓ_1-regularized least squares, referred to as lasso or basis pursuit, consists in finding a sparse coefficient vector $\mathbf{x} \in \mathbb{R}^m$, solution of the following optimization problem:

$$\mathcal{L}(\lambda, \mathbf{D}, \mathbf{y}): \qquad \mathbf{x}^\star = \underset{\mathbf{x}}{\operatorname{argmin}} \underbrace{\frac{1}{2}\|\mathbf{D}\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1}_{P(\mathbf{x}|\mathbf{D})} \tag{7.1}$$

where $P(\mathbf{x}|\mathbf{D})$ is called the primal objective and the parameter $\lambda > 0$ controls the trade-off between data fidelity and sparsity of the solution. We suppose that

$$\lambda \leq \lambda_{\max} := \|\mathbf{D}^T\mathbf{y}\|_\infty \tag{7.2}$$

since otherwise $\mathbf{x}^\star = \mathbf{0} \in \mathbb{R}^m$ is the unique solution (see section 2.3.4).

7.2 Iterative algorithms

First-order iterative algorithms, especially proximal-gradient methods (ISTA [Daubechies *et al.* 2004], FISTA [Beck & Teboulle 2009], TwIST [Bioucas-Dias & Figueiredo 2007], SpaRSA [Wright *et al.* 2009], Chambolle-Pock [Chambolle & Pock 2011]), are popular approaches for solving (7.1).

One can use an abstract notation to represent the update step for a generic iterative algorithm for the problem $\mathcal{L}(\lambda, \mathbf{D}, \mathbf{y})$: $\{\mathbf{x}_{t+1}, \boldsymbol{\alpha}_{t+1}\} \leftarrow p(\mathbf{x}_t, \mathbf{D}, \boldsymbol{\alpha}_t)$ where $\mathbf{x}_t$ is the current estimate of the primal variable and $\boldsymbol{\alpha}_t$ is a list of updated auxiliary scalars (e.g. the gradient step-size, and possibly a few previous estimates of the primal variable). For instance, the ISTA algorithm is given by

$$\mathbf{x}_{t+1} \leftarrow p(\mathbf{x}_t, \mathbf{D}, L_t) = \mathrm{ST}_{\gamma_t \lambda}\left(\mathbf{x}_t + \eta_t \mathbf{D}^T(\mathbf{y} - \mathbf{D}\mathbf{x}_t)\right) \tag{7.3}$$

where γ_t denotes the gradient step size and $\mathrm{ST}_u(z) = \mathrm{sign}(z)(|z| - u)_+$ denotes the soft-thresholding operation with threshold u, which is the proximal operator associated to (scaled versions of) the ℓ_1-norm.

7.3 Computational bottlenecks

The main bottleneck of existing iterative algorithms in terms of computational complexity is the cost of the required matrix-vector products involving the dictionary matrix, which dominate the overall iteration cost. For example, ISTA requires two matrix-vector multiplications at each iteration (or at least one if the Gram matrix $\mathbf{D}^T\mathbf{D}$ is pre-computed and stored).

7.4 Addressing the computational bottlenecks

A popular way to address this limitation is to constrain the dictionary matrix to a certain type of structure which would allow for fast matrix-vector products. Another approach is to use the so-called *screening rules* to identify a set indexing zero components of the solution $\mathbf{x}^\star$ *before even computing it.*

7.4.1 Acceleration with structured dictionaries

Different types of structure, more or less suited to a given application, can be imagined. In this kind of work, there is always a compromise between the flexibility (generality) of the structure and the provided speedup. We refer the reader to section 1.5.2 for a literature review of this area.

In practical applications, the dictionary matrix $\mathbf{D}$ is not necessarily structured. A possible strategy is to replace certain iterations $\mathbf{x}_{t+1} = p(\mathbf{x}_t, \mathbf{D}, \boldsymbol{\alpha}_t)$ with $\mathbf{x}_{t+1} = p(\mathbf{x}_t, \tilde{\mathbf{D}}, \boldsymbol{\alpha}_t)$

where $\tilde{\mathbf{D}}$ is a structured approximation of $\mathbf{D}$, i.e.,

$$\mathbf{D} = \tilde{\mathbf{D}} + \mathbf{E} \tag{7.4}$$

with controlled approximation error $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_m] \in \mathbb{R}^{n \times m}$.

7.4.2 Acceleration with screening rules

For a given instance $\mathcal{L}(\lambda, \mathbf{D}, \mathbf{y})$ of (7.1), characterized by a regularization parameter λ, a dictionary $\mathbf{D}$, and an input vector $\mathbf{y}$, the safe screening approach, introduced by [El Ghaoui *et al.* 2010], consists in identifying and removing from the dictionary a subset of atoms which are guaranteed to have zero weight in any solution $\mathbf{x}^\star$, before solving the problem. By removing these so-called *inactive atoms*, a more compact and readily solved problem is obtained, with decreased matrix-vector multiplication cost, while not affecting at all the original solution which can be obtained by simply zero-padding its restricted version.

Put differently, safe screening is a feature selection technique for a given instance of (7.1). However, unlike other previous feature selection heuristics [Fan & Lv 2008, Tibshirani *et al.* 2011] based on correlation measures between the atoms $\mathbf{d}_j$ and the input signal $\mathbf{y}$, safe screening has zero risk of false rejections. Moreover, screening techniques are transparent to the underlying ℓ_1 solver and can be readily combined with almost any existing solver.

Basically, three classes of screening rules can be distinguished: 1) Static; 2) Dynamic; 3) Sequential. The two first categories being mutually exclusive, but not the third one.

The earliest safe screening techniques [El Ghaoui *et al.* 2010] [Xiang *et al.* 2011] [Xiang & Ramadge 2012], currently classified as *static rules*, where designed to be applied once and for all before starting the optimization. In contrast, the more recent *dynamic rules*, introduced in [Bonnefoy *et al.* 2015] and followed by [Fercoq *et al.* 2015], are repeatedly applied during the iterative optimization algorithm leveraging its current solution estimate and gradually increasing the set of rejected atoms.

Sequential rules [Wang *et al.* 2015,Malti & Herzet 2016,Xiang *et al.* 2011,Liu *et al.* 2014] exploit the fact that (7.1) is commonly solved over a grid of regularization parameters λ_i and reuse the results from a previous configuration λ_{i-1} to improve the screening performance on further configurations.

7.5 Contributions

First, we introduce a formalism for defining safe screening tests which are *robust to approximation errors on the dictionary matrix* given that an error bound is provided. The resulting tests are called *stable screening* tests. The stable tests proposed here are general and apply to any context in which the atoms are known up to a certain error margin. The source of this error can be manifold, but we will focus on the case where it is a side effect of manipulating structured approximations of the true dictionary matrix. The proposed framework is also general with respect to the form of the safe region (be it a sphere, dome, etc) as well as to that of the error bound. Here, again, we stick to ℓ_p-balls, which allows us to exemplify the stable tests on some existing ℓ_2-sphere tests –one of which being the state-of-the-art GAP Safe [Fercoq *et al.* 2015]– for a general ℓ_q-ball error bound. Extension to dome tests is not particularly difficult within the proposed formalism, but is left for future work.

In a second moment (chapter 10), we exploit the proposed stable screening in a fast algorithm for ℓ_1-minimization problems, by making use of fast structured approximations of the problem's dictionary matrix. It consists in starting the iterative optimization by using a coarser (but faster) version of the true dictionary and, as approaching convergence, finer approximations are progressively adopted until eventually the original dictionary takes over. Choosing an appropriate moment to switch to a more precise dictionary is crucial in the proposed algorithm. A robust switching criterion based on both the duality gap saturation and the screening ratio is proposed.

7.6 Related Work

Apart from the aforementioned structured dictionaries and safe screening tests –as well as other preceding correlation-based feature selection heuristics [Fan & Lv 2008, Tibshirani *et al.* 2011]– some related acceleration strategies for sparsity-inducing optimization problems (or more specifically for problem (7.1)) are worth citing.

Instead of starting from the full problem and pruning the feature set, working set techniques [Kim & Park 2010, Johnson & Guestrin 2015] start with small restricted problems and progressively include more promising features. In [M. Massias & Salmon 2017], the authors combine a working set strategy with safe screening and in [Massias *et al.* 2018] they incorporate a dual extrapolation technique to further enhance the screening performance

and accelerate convergence. This idea is conceivably complementary to the techniques proposed here.

Joint screening [Herzet & Drémeau 2018] [Herzet *et al.* 2019] allows to screen many atoms which lie close together in one single test, reducing the number of required tests for a given dictionary. Interestingly, the resulting tests share many similarities and mathematical connections to the stable screening tests introduced here, despite arising from an essentially different premise.

A less closely related acceleration for ℓ_1-regularized problems is the Learned ISTA (LISTA) algorithm [Gregor & LeCun 2010] which consists in training a recurrent neural network for solving (7.1). However, the learned solver remains quite limited in scope, since: 1) the number of iterations has to be fixed beforehand, thus losing flexibility on the desired convergence precision; 2) the trained solver specializes to the specific type of data it was trained on. The techniques proposed in this paper, on the other hand, apply a wide range of general-purpose optimization algorithms for the considered problem, whatever the type of aimed data, convergence precision and regularization level.

Chapter 8

Reminders on Safe Screening

In this chapter, we briefly introduce the concept of safe screening along with some of the main screening methods proposed in the literature.

Given a set $\mathcal{A}$ of non-repeating integers, $\mathbf{D}_{[\mathcal{A}]} := [\mathbf{d}_i]_{i\in\mathcal{A}}$ denotes a sub-matrix of $\mathbf{D}$ composed of the columns indexed by the elements in $\mathcal{A}$. The notation extends to vectors: $\mathbf{x}_{[\mathcal{A}]} := [\mathbf{x}(i)]_{i\in\mathcal{A}}$, where $\mathbf{x}(i)$ denotes the i-th component of $\mathbf{x}$. Screening rules allow to identify such sets of reduced cardinality $|\mathcal{A}| < m$, so that the complexity of applying $\mathbf{D}_{[\mathcal{A}]}$ or its transpose is reduced compared to that of applying $\mathbf{D}$, while ensuring that $\mathcal{A}$ contains the support of the solution $\{i : \mathbf{x}^\star(i) \neq 0\}$. This is achieved using the dual of (7.1) (cf. section 2.3.3):

$$\boldsymbol{\theta}^\star = \underset{\boldsymbol{\theta}\in\Delta_{\mathbf{D}}}{\operatorname{argmax}} \underbrace{\frac{1}{2}\|\mathbf{y}\|_2^2 - \frac{\lambda^2}{2}\left\|\boldsymbol{\theta} - \frac{\mathbf{y}}{\lambda}\right\|_2^2}_{D(\boldsymbol{\theta})} \tag{8.1}$$

where

$$\Delta_{\mathbf{D}} = \{\boldsymbol{\theta}\in\mathbb{R}^n : \ \|\mathbf{D}^T\boldsymbol{\theta}\|_\infty \leq 1\} \tag{8.2}$$

is the *dual feasible set* and $D(\boldsymbol{\theta})$ is the *dual objective*. The dual and primal solutions ($\boldsymbol{\theta}^\star$ and $\mathbf{x}^\star$) are linked through the relation[1] $\mathbf{y} = \mathbf{D}\mathbf{x}^\star + \lambda\boldsymbol{\theta}^\star$. Optimality conditions (KKT) at the dual solution $\boldsymbol{\theta}^\star$ read (see for instance [Xiang *et al.* 2017] for more details)

$$\mathbf{d}_j^T\boldsymbol{\theta}^\star = \begin{cases} \operatorname{sign}(\mathbf{x}^\star(j)) & \text{if} \quad \mathbf{x}^\star(j) \neq 0 \\ \gamma_j \in [-1,1] & \text{if} \quad \mathbf{x}^\star(j) = 0 \end{cases}, \quad \forall j \in \{1,\dots,m\}. \tag{8.3}$$

Hence, every dictionary atom for which $|\mathbf{d}_j^T\boldsymbol{\theta}^\star| < 1$ is inactive.

1. The product $\mathbf{D}\mathbf{x}^\star$ is always unique, even when the primal solution $\mathbf{x}^\star$ is not [Tibshirani 2013].

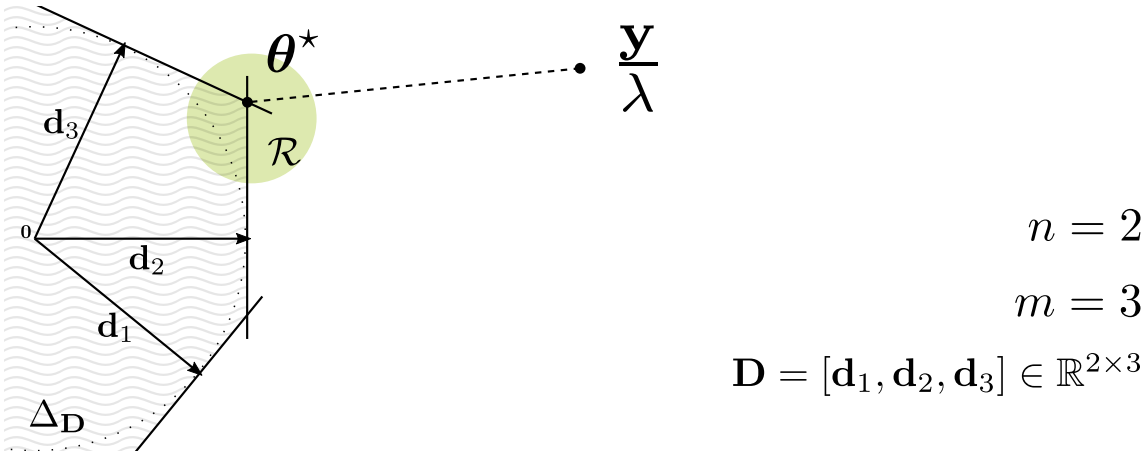


Figure 8.1 – Example of safe region $\mathcal{R}$ (in green), for $n = 2$ and $m = 3$. The dual solution $\boldsymbol{\theta}^\star$ is the projection of $\mathbf{y}/\lambda$ over the dual feasible set $\Delta_{\mathbf{D}}$. Atoms $\mathbf{d}_2$ and $\mathbf{d}_3$ are active.

8.1 Notion of safe region

Since the optimal solution $\boldsymbol{\theta}^\star$ of the dual problem (8.1) is not known beforehand, the inner products $\mathbf{d}_j^T\boldsymbol{\theta}^\star$ cannot be evaluated. Fortunately, given only $\mathbf{D}$ and $\mathbf{y}$, it is possible to identify at a moderate computational cost a region $\mathcal{R} \subset \mathbb{R}^n$, called *safe region*, which is guaranteed to contain the optimal $\boldsymbol{\theta}^\star$.

Definition 24 (Safe region)**.** *A region $\mathcal{R} \subset \mathbb{R}^n$ is safe (with respect to the dual problem* (8.1)*) if and only if it contains the dual solution $\boldsymbol{\theta}^\star$, i.e. $\boldsymbol{\theta}^\star \in \mathcal{R}$.*

Figure 8.1 illustrates the concept of safe region. We consider the simple case of $n = 2$ (i.e. input data $\mathbf{y}$ lies in a plane) and $m = 3$ (i.e. the dictionary $\mathbf{D}$ has three atoms).

8.2 Screening an atom given a safe region

Consider an atom $\mathbf{d}_j$. If the inequality $|\mathbf{d}_j^T\boldsymbol{\theta}| < 1$ holds for all $\boldsymbol{\theta} \in \mathcal{R}$ where $\mathcal{R}$ is a safe region, then the above analysis ensures that $\mathbf{d}_j$ is inactive. This gives rise to a *screening test*.

Definition 25 (Screening test for an atom)**.** *Given a region $\mathcal{R}$, a screening test for the atom $\mathbf{d} \in \mathbb{R}^n$ is given by:*

$$\mu(\mathbf{d}|\mathcal{R}) = \sup_{\boldsymbol{\theta}\in\mathcal{R}} |\mathbf{d}^T\boldsymbol{\theta}| \tag{8.4}$$

A sufficient condition for an atom $\mathbf{d}_j$ to be inactive can be expressed as follows: if $\mathcal{R}$ is safe then

$$\mu(\mathbf{d}_j|\mathcal{R}) < 1 \quad \Longrightarrow \quad \mathbf{x}^\star(j) = 0.$$

In practice, for each atom $\mathbf{d}_j$, computing the test $\mu(\mathbf{d}_j|\mathcal{R})$ allows to eliminate or not the atom.

Formally, given a safe region, the atoms can be partitioned into a preserved set $\mathcal{A}$ and its complement, the rejection set $\mathcal{A}^\mathsf{c}$, that gathers the indices of inactive atoms:

$$\begin{aligned} \mathcal{A} &= \{j \in \{1, \dots, m\} : \mu(\mathbf{d}_j|\mathcal{R}) \geq 1\}, \\ \mathcal{A}^\mathsf{c} &= \{j \in \{1, \dots, m\} : \mu(\mathbf{d}_j|\mathcal{R}) < 1\}. \end{aligned} \tag{8.5}$$

In practice, safe regions have simple parameterized shapes which parameters need to be identified with moderate computations from the only knowledge of $\mathbf{D}$, $\mathbf{y}$, and possibly the current iterate $\mathbf{x}_t$ of an iterative algorithm addressing (7.1).

The two most common shapes of safe regions in the literature are spheres and domes (i.e. intersection between a sphere and one or more half-planes).

Let us focus on sphere tests. Assume we are given a safe region $\mathcal{R}$ which is a closed ℓ_p-ball with center $\mathbf{c}$ and radius R, denoted $B_p(\mathbf{c}, R) = \{\mathbf{z} : \|\mathbf{z} - \mathbf{c}\|_p \leq R\}$. The screening test for this region has a closed form (a proof is given in Appendix C.1)

$$\mu(\mathbf{d}|B_p(\mathbf{c}, R)) = |\mathbf{d}^T\mathbf{c}| + R\|\mathbf{d}\|_{p^*} \tag{8.6}$$

where $\|\cdot\|_{p^*}$ denotes the dual norm associated to the p-norm, with $\frac{1}{p} + \frac{1}{p^*} = 1$. For simplicity, we omit subscripts for the ℓ_2-ball ($p = p^* = 2$) denoted $B(\mathbf{c}, R)$.

8.3 Construction of safe spherical regions

A safe region should be as small as possible (to maximize the screening effect) while requiring as little computational overhead as possible. In light of (8.6), as $\|\mathbf{d}\|_{p^*}$ can be precomputed for each atom, the computational overhead of a screening test with a spherical region is governed by the cost of computing the radius R and the inner products $\mathbf{d}^T\mathbf{c}$ for all atoms that have not been screened out yet. This calls for techniques where the choice of R and $\mathbf{c}$ allows to reuse either quantities that have already been computed

along previous iterations of the optimization algorithm.

The construction of the first (static) safe region, obtained in [El Ghaoui *et al.* 2010], follows from the simple observation that the solution $\boldsymbol{\theta}^\star$ of the dual problem (8.1) is the Euclidean projection of $\mathbf{y}/\lambda$ on the feasible set $\Delta_\mathbf{D}$. As a result, if some feasible point $\boldsymbol{\theta}_F \in \Delta_\mathbf{D}$ is known, then $\boldsymbol{\theta}^\star$ can't be further away from $\mathbf{y}/\lambda$ than $\boldsymbol{\theta}_F$ in the ℓ_2 sense, i.e. $\|\boldsymbol{\theta}^\star - \mathbf{y}/\lambda\|_2 \leq \|\boldsymbol{\theta}_F - \mathbf{y}/\lambda\|_2 \; \forall \boldsymbol{\theta}_F \in \Delta_\mathbf{D}$.

This leads to an ℓ_2-spherical safe region ($p{=}p^*{=}2$),

$$\mathcal{R} = B(\mathbf{c} = \mathbf{y}/\lambda, R = \|\boldsymbol{\theta}_F - \mathbf{y}/\lambda\|_2), \tag{8.7}$$

which full specification requires a feasible point $\boldsymbol{\theta}_F \in \Delta_\mathbf{D}$.

To generate a feasible point $\boldsymbol{\theta}_F \in \Delta_\mathbf{D}$, one could compute the Euclidean projection of any given point $\mathbf{z} \in \mathbb{R}^n$ over the closed convex set $\Delta_\mathbf{D}$. As this is too computationally demanding, simpler scaling approaches are preferred: given $\mathbf{z} \in \mathbb{R}^n$ we determine a scaling factor α so that $\Theta(\mathbf{z}|\mathbf{D}) := \alpha\mathbf{z} \in \Delta_\mathbf{D}$.

By definition of $\Delta_\mathbf{D}$, if $\|\mathbf{D}^T\mathbf{z}\|_\infty \leq 1$ no scaling is needed ($\alpha = 1$ works), otherwise one only needs to divide $\mathbf{z}$ by $\|\mathbf{D}^T\mathbf{z}\|_\infty$. This yields

$$\Theta(\mathbf{z}|\mathbf{D}) := \frac{\mathbf{z}}{\max\left(1, \|\mathbf{D}^T\mathbf{z}\|_\infty\right)}. \tag{8.8}$$

While the scaling in (8.8) selects a feasible point to build a safe sphere (8.7) given an arbitrary $\mathbf{z}$, it neglects the fact that one would like to minimize the *radius* of this sphere to maximize the effect of screening. This leads to the *dual scaling* [El Ghaoui *et al.* 2010] formalized as follows (a proof is given in [Bonnefoy *et al.* 2015, Lemma 8])

Proposition 2. *Denoting* $\text{proj}_\mathbf{z}\dfrac{\mathbf{y}}{\lambda} = \dfrac{\mathbf{y}^T\mathbf{z}}{\lambda\|\mathbf{z}\|_2^2}\mathbf{z}$ *the Euclidean projection of* $\mathbf{y}/\lambda$ *onto the direction of* $\mathbf{z}$*, we have*

$$\Theta\left(\text{proj}_\mathbf{z}\frac{\mathbf{y}}{\lambda} \mid \mathbf{D}\right) = \underset{\mathbf{z}'=\alpha\mathbf{z}}{\text{argmin}} \|\mathbf{z}' - \mathbf{y}/\lambda\|_2 \quad \textit{s.t.} \; \|\mathbf{D}^T\mathbf{z}'\|_\infty \leq 1$$

i.e. among all points which are both dual feasible and proportional to $\mathbf{z}$, $\Theta\left(\text{proj}_\mathbf{z}\dfrac{\mathbf{y}}{\lambda} \mid \mathbf{D}\right)$ *is the one closest to* $\mathbf{y}/\lambda$.

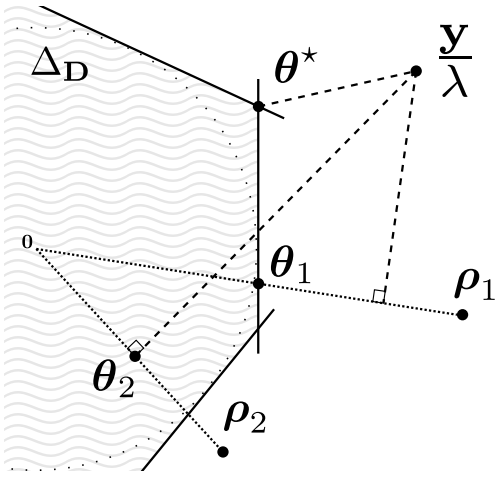


Figure 8.2 – Examples of dual feasible points calculated as in equation (8.9). The dual point $\boldsymbol{\theta}_t$ is obtained by projecting $\mathbf{y}/\lambda$ over the direction of the current residual $\boldsymbol{\rho}_t$ and then, if necessary, scaling the result. Note that $\boldsymbol{\theta}_1$ is obtained by scaling the residual $\boldsymbol{\rho}_1$, since the projection of $\mathbf{y}/\lambda$ is not feasible; while $\boldsymbol{\theta}_2$ corresponds directly to projection of $\mathbf{y}/\lambda$, which is already feasible.

To obtain dynamic spherical regions, we need dual feasible points that change over iterations ($\boldsymbol{\theta}_F = \boldsymbol{\theta}_t$ at iteration t) and leverage the current state of the optimization algorithm. Following [Bonnefoy *et al.* 2015], this can be achieved by defining $\boldsymbol{\theta}_t$ proportional to the current residual $\boldsymbol{\rho}_t := \mathbf{y} - \mathbf{D}\mathbf{x}_t$, since the dual solution $\boldsymbol{\theta}^\star$ is proportional to $\mathbf{y} - \mathbf{D}\mathbf{x}^\star$. Taking $\mathbf{z} = \boldsymbol{\rho}_t$ and $\boldsymbol{\theta}_t = \Theta\left(\text{proj}_{\boldsymbol{\rho}_t}\frac{\mathbf{y}}{\lambda} \mid \mathbf{D}\right)$ yields

$$\boldsymbol{\theta}_t = \left[\frac{\mathbf{y}^T\boldsymbol{\rho}_t}{\lambda\|\boldsymbol{\rho}_t\|_2^2}\right]_{-\alpha_t}^{\alpha_t}\boldsymbol{\rho}_t, \qquad \text{with } \alpha_t = (\|\mathbf{D}^T\boldsymbol{\rho}_t\|_\infty)^{-1} \tag{8.9}$$

where $[z]_a^b := \min(\max(z, a), b)$ denotes the projection of the scalar z onto segment $[a, b]$.

An illustration of the procedure described in equation (8.9) is given in Figure 8.2.

The above principles yield existing safe spherical regions, some of which we list below.

8.3.1 Static Safe sphere [El Ghaoui *et al.* 2010]

By setting $\mathbf{z} = \mathbf{y}/\lambda$ in (8.8) one obtains $\theta_F = \Theta(\mathbf{y}/\lambda|\mathbf{D}) = \mathbf{y}/\lambda_{\max}$, and (8.7) reads $\mathcal{R} = B(\mathbf{c}, R)$ with

$$\text{Static Safe:} \quad \mathbf{c} = \mathbf{y}/\lambda, R = |\tfrac{1}{\lambda_{\max}} - \tfrac{1}{\lambda}| \cdot \|\mathbf{y}\|_2 \tag{8.10}$$

where we recall that $\lambda_{\max} := \|\mathbf{D}^T\mathbf{y}\|_\infty$ is defined in (7.2).

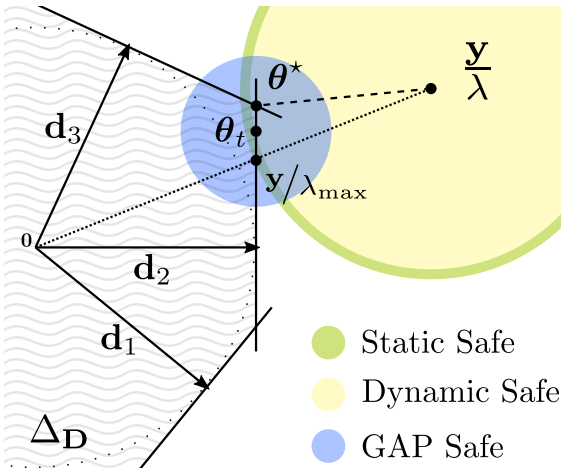


Figure 8.3 – Examples of safe spherical regions. The static sphere is centered at $\mathbf{y}/\lambda$ and extends until reaching the dual feasible point $\mathbf{y}/\lambda_{\max}$. The dynamic sphere is also centered at $\mathbf{y}/\lambda$ but extends until reaching the dual feasible point $\boldsymbol{\theta}_t$ – updated at each iteration t using, for instance, equation (8.9). The GAP Safe sphere, on the other hand, is centered at $\boldsymbol{\theta}_t$ and has a radius proportional to the duality gap at the current iteration.

8.3.2 Dynamic Safe sphere [Bonnefoy *et al.* 2015]

With the dual feasible point defined in (8.9), safe region (8.7) yields $\mathcal{R} = B(\mathbf{c}, R)$ with

$$\text{Dynamic Safe:} \quad \mathbf{c} = \mathbf{y}/\lambda, R = \|\boldsymbol{\theta}_t - \mathbf{y}/\lambda\|_2. \tag{8.11}$$

8.3.3 GAP Safe sphere [Fercoq *et al.* 2015]

Given any primal-dual feasible pair $(\mathbf{x}_t, \boldsymbol{\theta}_t) \in \mathbb{R}^m \times \Delta_\mathbf{D}$ (say the solution estimations at iteration t) and denoting

$$G(\mathbf{x}_t, \boldsymbol{\theta}_t|\mathbf{D}) := P(\mathbf{x}_t|\mathbf{D}) - D(\boldsymbol{\theta}_t) \tag{8.12}$$

the corresponding duality gap (section 2.3.2), the GAP Safe sphere is $\mathcal{R} = B(\mathbf{c}, R)$ with

$$\text{GAP Safe:} \quad \mathbf{c} = \boldsymbol{\theta}_t, R = \frac{1}{\lambda}\sqrt{2G(\mathbf{x}_t, \boldsymbol{\theta}_t|\mathbf{D})}. \tag{8.13}$$

This region is provably safe for any dual feasible point $\boldsymbol{\theta}_t \in \Delta_\mathbf{D}$ [Fercoq *et al.* 2015, Theorem 2]. The authors, following [Bonnefoy *et al.* 2015], propose to use $\boldsymbol{\theta}_t$ defined by (8.9).

Figure 8.3 exemplifies the three described safe regions (subsections 8.3.1 to 8.3.3). Note that the cited safe regions reuse quantities computed along the iterations of classical proximal algorithms such as the residual $\boldsymbol{\rho}_t$, the inner products $\mathbf{D}^T\boldsymbol{\rho}_t$, or the duality gap (which typically serves as a stopping criterion). This will be detailed in chapter 10.

8.4 Supplementary discussion

Let us use this section to draw the relation between some remarkable subsets of the set of all atoms of the dictionary. We first recall some definitions.

Solution support An important set is the support of a primal solution $\mathbf{x}^\star$, given by

$$\operatorname{supp}(\mathbf{x}^\star) := \{j \in \{1, \dots, m\} : \mathbf{x}^\star(j) \neq 0\}\,.$$

Equicorrelation set Another related (but distinct) subset of atoms is the equicorrelation set, denoted $\mathcal{E}$, which can be defined as follows (cf. definition 3)

$$\mathcal{E} := \left\{j \in \{1, \dots, m\} : |\mathbf{d}_j^T \boldsymbol{\theta}^\star| = 1\right\}.$$

It contains the support of all possible primal solutions [Tibshirani 2013].

Inactive atoms We also define the set of inactive atoms as follows:

$$\left\{j \in \{1, \dots, m\} : |\mathbf{d}_j^T \boldsymbol{\theta}^\star| < 1\right\}.$$

Note that our definition of *inactive atoms* corresponds precisely to the complement of the equicorrelation set, since from the optimality condition in (8.3) we know that $|\mathbf{d}_j^T \boldsymbol{\theta}^\star| \leq 1$.

Preserved set we denote $\mathcal{A}_t$ the preserved set determined by a safe dynamic screening rule at iteration t, and its complement $\mathcal{A}_t^\complement$ the rejection set (cf. equation (8.5), and note that in a dynamic rule we take into account the previous preserved set $\mathcal{A}_{t-1}$, which gives a nested structure to the sequence of preserved sets $\mathcal{A}_t$).

$$\mathcal{A}_t \;= \{j \in \mathcal{A}_{t-1} : \mu(\mathbf{d}_j|\mathcal{R}) \geq 1\}. \tag{8.14}$$

From the definition of the screening test (cf. definition 25), one can see that, as the radius of the safe region vanishes ($R \to 0$), we have that $\mu(\mathbf{d}_j|\mathcal{R}) \to |\mathbf{d}_j^T \boldsymbol{\theta}^\star|$ and the preserved set tends to the equicorrelation set (not the support!), i.e. $\mathcal{A} \to \mathcal{E}$.

The relations between the mentioned sets is illustrated in Figure 8.4 and summarized in the property below.

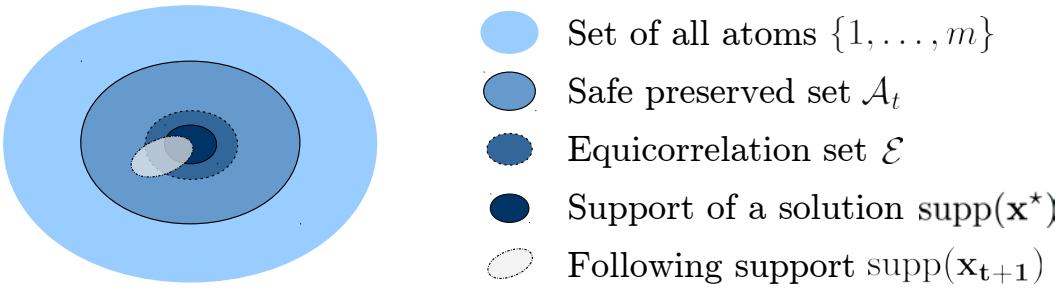


Figure 8.4 – Links between some remarkable sets, see Property 13.

Property 13. *For a given lasso instance* $\mathcal{L}(\lambda, \mathbf{D}, \mathbf{y})$ *with* $\mathbf{D} \in \mathbb{R}^{n \times m}$*, denoting* $\mathbf{x}^\star$ *a primal solution,* $\mathcal{A}_t$ *a safe preserved set at iteration* t *and* $\mathcal{E}$ *the equicorrelation set, we have:*

$$\mathrm{supp}(\mathbf{x}^\star) \subseteq \mathcal{E} \subseteq \mathcal{A}_t \subseteq [1, \ldots, m].$$

Also, when using a dynamic screening rule, we have:

$$\mathrm{supp}(\mathbf{x}_{t+1}) \subseteq \mathcal{A}_t$$

with $\mathbf{x}_{t+1}$ *being a solution estimate at iteration* $t+1$*, when the set* $\mathcal{A}_t$ *is taken into account.*

Proof. The first inclusion is a direct consequence of the definition of $\mathcal{E}$ and the optimality condition in (8.3). Indeed, $|\mathbf{d}_j^T \boldsymbol{\theta}^\star| = 1$ is a necessary (but not sufficient) condition for $\mathbf{x}^\star(j) \neq 0$.
Since the screening tests identify a subset of the inactive atoms (which, in turn, is equivalent to $\mathcal{E}^\mathsf{c}$), we obtain $\mathcal{A}_t^\mathsf{c} \subseteq \mathcal{E}^\mathsf{c}$ which implies the second inclusion.
The third inclusion is trivial.
The inclusion in the second expression is a direct consequence of using a screening rule, as the entries in the rejection set are definitively set to zero. □

A dynamic safe rule is said to be *converging* [Ndiaye *et al.* 2017, Definition 12] if it defines a sequence of safe regions with diameter converging to zero. A converging safe rule eventually recovers the equicorrelation set. The GAP Safe rule can be shown to be converging [Ndiaye *et al.* 2017, Section 4.5] when paired with a converging lasso solver. This result is actually easy to grasp by observing that the radius of the GAP Safe sphere is proportional to the duality gap, which vanishes as the optimization algorithm converges (as explained in section 2.3.2). Figure 8.5 illustrates a typical behavior of the preserved

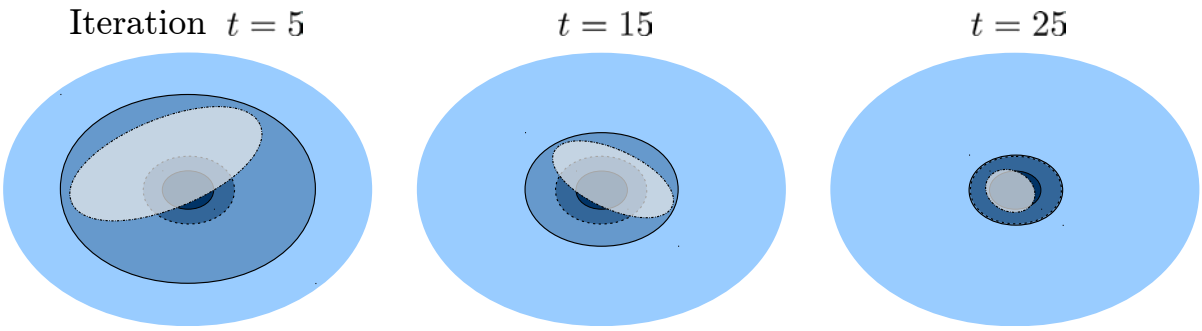


Figure 8.5 – Evolution of the preserved set and current support over the iterations.

set and the current support over the iterations of a lasso solver. In this example, the active set converged to the equicorrelation set (rightmost diagram in Figure 8.5).

As discussed in Section 1.4.3 the primal solution might not be unique. However, the very mild assumption that the the entries of $\mathbf{D}$ can take continuous values is sufficient to guarantee uniqueness, which is actually the case in most practical scenarios –and, in particular, in all the performed experiments. It is important to emphasize that the screening rules remain safe in the non-unique case, since the rejected atoms are those not belonging to the equicorrelation set and, thus, those that will never appear in *any* solution.

Chapter 9

Stable Safe Screening

To further accelerate the ℓ_1-minimization, our idea is to combine the screening rules described in the previous chapter with the use of fast structured dictionaries. To do so, we propose to perform the initial iterations using a fast but approximate version $\tilde{\mathbf{D}}$ of the dictionary $\mathbf{D}$. We would like, however, to perform safe screening, despite manipulating an approximate version of the dictionary.

When iterating with $\tilde{\mathbf{D}}$ instead of $\mathbf{D}$, the computational overhead of accessing inner products $\mathbf{d}^T\mathbf{c}$ is no longer moderate, hence the screening techniques reviewed in chapter 8 cannot be directly applied in this context. A possibility would of course be to recompute residuals / inner products / duality gaps associated to $\mathbf{D}$ (instead of $\tilde{\mathbf{D}}$) to implement the above screening tests, but the whole purpose of re-using quantities already computed with a structured (computationally efficient) matrix $\tilde{\mathbf{D}}$ would be lost. Also, in some cases, $\mathbf{D}$ might not even be accessible, which eliminates the previously evoked possibility.

In this chapter, we propose an alternative solution by deriving screening rules which remain safe w.r.t the original problem (7.1) *even when manipulating an approximate version of the dictionary*. These *stable* screening rules only require some knowledge on the magnitude of certain approximation errors between $\tilde{\mathbf{D}}$ and $\mathbf{D}$.

9.1 Screening a *zone* given a safe region

In conventional screening tests (Definition 25) the dual solution $\boldsymbol{\theta}^\star$ is assumed to belong to a known safe region $\mathcal{R}$. This allows to screen (or not) an atom $\mathbf{d}$ that is perfectly known, (re)using computations of inner products *with this atom*.

When iterating with $\tilde{\mathbf{D}}$, efficient screening tests are only entitled to reuse inner products with approximated atoms, $\tilde{\mathbf{D}}^T\mathbf{c}$. Yet, they should be able to screen certain atoms $\mathbf{d}$, using the only knowledge that these are "close" to the corresponding approximate atoms $\tilde{\mathbf{d}}$.

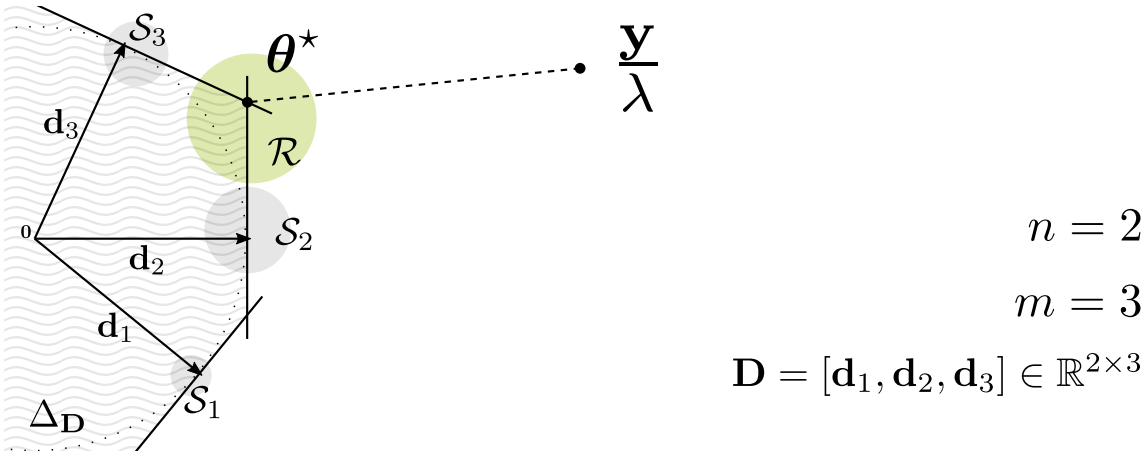


Figure 9.1 – Example of safe region $\mathcal{R}$ and stability zones $\mathcal{S}_j$ in a plane ($n = 2$).

One way to capture this knowledge is to consider $\mathcal{S} \subset \mathbb{R}^n$ some neighborhood of $\mathbf{d}$, assumed to be known. We will call $\mathcal{S}$ a *zone*. This gives rise to screening tests for zones.

Definition 26 (Screening test for a zone)**.** *Given a safe region $\mathcal{R}$, a screening test for the zone $\mathcal{S} \subset \mathbb{R}^n$ is given by:*

$$\mu(\mathcal{S}|\mathcal{R}) := \sup_{\mathbf{d}\in\mathcal{S}} \mu(\mathbf{d}|\mathcal{R}) = \sup_{\mathbf{d}\in\mathcal{S}} \sup_{\boldsymbol{\theta}\in\mathcal{R}} |\mathbf{d}^T\boldsymbol{\theta}| \tag{9.1}$$

A sufficient condition for an atom $\mathbf{d}$ to be inactive can be expressed as follows: if $\mathcal{R}$ is safe and if $\mathbf{d} \in \mathcal{S}$, then

$$\mu(\mathcal{S}|\mathcal{R}) < 1 \implies \mathbf{x}^\star(j) = 0.$$

Figure 9.1 illustrates the relation between the introduced concepts: the safe region $\mathcal{R}$ and the stability zones (a zone $\mathcal{S}_j$ for each atom $\mathbf{d}_j$).

9.1.1 *Stable* screening tests

Given an approximate dictionary $\tilde{\mathbf{D}}$, error bounds ϵ_j such that $\|\tilde{\mathbf{d}}_j - \mathbf{d}_j\|_q \leq \epsilon_j$, $1 \leq j \leq m$, can be pre-computed for some choice of $q \in [1, \infty]$, yielding spherical zones $\mathcal{S}_j := B_q(\tilde{\mathbf{d}}_j, \epsilon_j)$ known to contain the atoms: $\mathbf{d}_j \in \mathcal{S}_j$. Using (8.6), a spherical zone

$$\mathcal{S} := B_q(\tilde{\mathbf{d}}, \epsilon) \tag{9.2}$$

gives rise to the following test (see proof in Appendix C.2.):

$$\mu(\mathcal{S}|B_p(\mathbf{c},R)) = |\tilde{\mathbf{d}}^T\mathbf{c}| + \epsilon\|\mathbf{c}\|_{q^*} + R\|\tilde{\mathbf{d}}\|_{p^*} + CR\epsilon. \tag{9.3}$$

where $C = C_{p,q} := n^{\left(1-\frac{1}{p}-\frac{1}{q}\right)_+}$ and $q^* \in [1,\infty]$ is such that $\frac{1}{q} + \frac{1}{q^*} = 1$.

In practice, the norms $a_j = \|\mathbf{d}_j\|_{p^*}$ can be pre-computed and stored with negligible overhead. This motivates the use of restricted spherical zones which take into account this new information (i.e. the knowledge of the atoms' norms)

$$\mathcal{S}' := \mathcal{S} \cap \{\mathbf{d} : \|\mathbf{d}\|_{p^*} = a\}. \tag{9.4}$$

For such zones we get (see Appendix C.2)

$$\mu(\mathcal{S}'|B_p(\mathbf{c},R)) = |\tilde{\mathbf{d}}^T\mathbf{c}| + \epsilon\|\mathbf{c}\|_{q^*} + Ra. \tag{9.5}$$

Lemma 5. *Consider a dictionary* $\mathbf{D}$ *and* $\mathcal{R} = B_p(\mathbf{c},R)$ *a safe region with respect to the dual problem* (8.1). *Let* $\tilde{\mathbf{D}}$ *be an approximate dictionary,* $1 \leq j \leq m$, *and* $\mathcal{S}_j$, $\mathcal{S}'_j$ $\mu(\mathcal{S}_j|B_p(\mathbf{c},R))$, $\mu(\mathcal{S}'_j|B_p(\mathbf{c},R))$ *be defined by* (9.2)-(9.5) *with* $\tilde{\mathbf{d}} = \tilde{\mathbf{d}}_j$, $\epsilon \geq \|\tilde{\mathbf{d}}_j - \mathbf{d}_j\|_q$, *and* $a = \|\mathbf{d}_j\|_{p^*}$.

- *If* $\mu(\mathcal{S}_j|B_p(\mathbf{c},R)) < 1$ *then* $\mathbf{x}^\star_j = 0$;
- *If* $\mu(\mathcal{S}'_j|B_p(\mathbf{c},R)) < 1$ *then* $\mathbf{x}^\star_j = 0$.

Remark 3. *In [Herzet et al. 2019], Definition 26 is used in a different context. No approximate atoms are considered and the idea is rather to simultaneously test multiple atoms which eventually lie within a same zone* $\mathcal{S}$, *reducing the total number of tests performed.*

9.2 Construction of safe regions using approximate dictionaries

As in classical screening, we have all ingredients to screen provided we can build a safe region using moderate computational overhead. The object of this section is precisely to adapt the constructions of safe regions from Section 8.3, which depend on $\mathbf{D}$, to define new safe regions reusing computations done during the iterations *with* $\tilde{\mathbf{D}}$ *instead of* $\mathbf{D}$.

Remark 4. *Let us emphasize that a safe region is a region $\mathcal{R}$ containing the dual solution $\boldsymbol{\theta}^\star$ of the* original *dual problem* (8.1) *and* not necessarily *the dual solution of its approximate version (i.e. with $\Delta_{\tilde{\mathbf{D}}}$ instead of $\Delta_{\mathbf{D}}$). Indeed, although $\mathbf{D}$ is approximated to speed up computations, variable elimination needs to be guaranteed with respect to the* original *problem* (7.1).

The safe regions from Section 8.3 are built by determining a feasible dual point $\boldsymbol{\theta}_t \in \Delta_{\mathbf{D}}$. Given an arbitrary $\mathbf{z} \in \mathbb{R}^n$, the function $\Theta(\mathbf{z}|\mathbf{D})$ from (8.8) would provide such a feasible point, however it cannot be used as its computation requires multiplication by $\mathbf{D}^T$. A naive alternative could be to compute $\Theta(\mathbf{z}|\tilde{\mathbf{D}})$, however while this always belongs to $\Delta_{\tilde{\mathbf{D}}}$ (the feasible set for $\tilde{\mathbf{D}}$) it does not necessarily belong to the desired feasible set $\Delta_{\mathbf{D}}$. This can be fixed using a modified dual scaling approach that we propose to call *stable* dual scaling.

Considering error bounds $\boldsymbol{\epsilon} = (\epsilon_j)_{j=1}^m \in \mathbb{R}_+^m$ such that $\|\tilde{\mathbf{d}}_j - \mathbf{d}_j\|_q \leq \epsilon_j$, we define

$$\Theta'(\mathbf{z}|\tilde{\mathbf{D}}, \boldsymbol{\epsilon}) := \frac{\mathbf{z}}{\max\left(1, \max_{1\leq j\leq m}(|\tilde{\mathbf{d}}_j^T\mathbf{z}| + \epsilon_j\|\mathbf{z}\|_{q^*})\right)} \tag{9.6}$$

For $\boldsymbol{\epsilon} = \mathbf{0}$ we recover $\Theta(\mathbf{z}|\mathbf{D}) = \Theta'(\mathbf{z}|\mathbf{D}, \mathbf{0})$.

Lemma 6. *Assume that $\mathbf{d}_j \in \mathcal{S}_j = B_q(\tilde{\mathbf{d}}_j, \epsilon_j)$ for $1 \leq j \leq m$. Then, for any $\mathbf{z} \in \mathbb{R}^n$, $\Theta'(\mathbf{z}|\tilde{\mathbf{D}}, \boldsymbol{\epsilon}) \in \Delta_{\mathbf{D}} \cap \Delta_{\tilde{\mathbf{D}}}$ is a feasible point w.r.t. both the original dual problem* (8.1) *and its modified version with $\tilde{\mathbf{D}}$ instead of $\mathbf{D}$.*

The proof is in Appendix C.3. Analogously to (8.9), we can now define a dual feasible point proportional to the residual $\tilde{\boldsymbol{\rho}}_t := \mathbf{y} - \tilde{\mathbf{D}}\mathbf{x}_t$ at iteration t, $\boldsymbol{\theta}'_t = \Theta'\left(\mathrm{proj}_{\tilde{\boldsymbol{\rho}}_t}\frac{\mathbf{y}}{\lambda} \mid \tilde{\mathbf{D}}, \boldsymbol{\epsilon}\right)$, that is to say more explicitly

$$\begin{aligned}
&\boldsymbol{\theta}'_t = \left[\frac{\mathbf{y}^T\tilde{\boldsymbol{\rho}}_t}{\lambda\|\tilde{\boldsymbol{\rho}}_t\|_2^2}\right]_{-\alpha'_t}^{\alpha'_t}\tilde{\boldsymbol{\rho}}_t, \\
&\text{with } \alpha'_t = \left(\max_{1\leq j\leq m}\left(|\tilde{\mathbf{d}}_j^T\tilde{\boldsymbol{\rho}}_t| + \epsilon_j\|\tilde{\boldsymbol{\rho}}_t\|_{q^*}\right)\right)^{-1}.
\end{aligned} \tag{9.7}$$

Recalling that $[z]_a^b := \min(\max(z, a), b)$ is the projection of a scalar z onto segment $[a, b]$.

These principles yield new *stable* safe ℓ_2-spherical regions.

9.2.1 Stable Static Safe sphere

The static safe sphere (8.10) depends on $\lambda_{\max} = \max_j |\mathbf{d}_j^T \mathbf{y}|$. To perform only computations with $\tilde{\mathbf{D}}$, we can reuse (8.10) with

$$\lambda'_{\max} := \max_j \left(|\tilde{\mathbf{d}}_j^T \mathbf{y}| + \epsilon_j \|\mathbf{y}\|_{q^*} \right) \geq \lambda_{\max}, \tag{9.8}$$

leading to $\mathcal{R} = B(\mathbf{c}', R')$ with

$$\text{Stable Static Safe:} \quad \mathbf{c}' = \mathbf{y}/\lambda, R' = |\tfrac{1}{\lambda'_{\max}} - \tfrac{1}{\lambda}| \cdot \|\mathbf{y}\|_2. \tag{9.9}$$

9.2.2 Stable Dynamic Safe sphere

We adapt the Dynamic Safe sphere in (8.11) to our approximate setting by replacing the dual feasible point $\boldsymbol{\theta}_t$ (8.9) by $\boldsymbol{\theta}'_t$ (9.7), leading to $\mathcal{R} = B(\mathbf{c}', R')$ with

$$\text{Stable Dynamic Safe:} \quad \mathbf{c}' = \mathbf{y}/\lambda, R' = \|\boldsymbol{\theta}'_t - \mathbf{y}/\lambda\|_2. \tag{9.10}$$

The fact that $\boldsymbol{\theta}'_t \in \Delta_{\mathbf{D}}$ directly implies that this sphere is safe by definition of the basic ℓ_2-spherical bound (8.7).

9.2.3 Stable GAP Safe sphere

To build a safe sphere $B(\mathbf{c}', R')$ with center $\mathbf{c}' = \boldsymbol{\theta}'_t$ (9.7) instead of $\mathbf{c} = \boldsymbol{\theta}_t$ (8.9) we need to determine a safe radius R' depending only on $\tilde{\mathbf{D}}$. For this reason, instead of using the (inaccessible) duality gap $G(\mathbf{x}_t, \boldsymbol{\theta}'_t|\mathbf{D}) = P(\mathbf{x}_t|\mathbf{D}) - D(\boldsymbol{\theta}'_t)$ we will use $G(\mathbf{x}_t, \boldsymbol{\theta}'_t|\tilde{\mathbf{D}}) = P(\mathbf{x}_t|\tilde{\mathbf{D}}) - D(\boldsymbol{\theta}'_t)$ and add a security margin ensuring that $B(\boldsymbol{\theta}'_t, R')$ is safe. Recall the standard notation $\|\mathbf{M}\|_{p\to q} := \sup_{\|u\|_p \leq 1} \|\mathbf{M}\mathbf{u}\|_q$.

Theorem 10 (Stable GAP Safe sphere)**.** *Consider*

$$E \geq \|\mathbf{D} - \tilde{\mathbf{D}}\|_{r\to 2}$$

where $1 \leq r \leq \infty$. *For any*[1] $(\mathbf{x}, \boldsymbol{\theta}) \in \mathbb{R}^m \times \Delta_{\mathbf{D}}$, *we have*

$$\boldsymbol{\theta}^\star \in B\left(\mathbf{c}' = \boldsymbol{\theta}, R' = \frac{1}{\lambda}\sqrt{2G(\mathbf{x}, \boldsymbol{\theta}|\tilde{\mathbf{D}}) + 2\delta(\mathbf{x})}\right) \tag{9.11}$$

$$\textit{with} \qquad \delta(\mathbf{x}) := \|\mathbf{y} - \tilde{\mathbf{D}}\mathbf{x}\|_2 E\|\mathbf{x}\|_r + \frac{1}{2}E^2\|\mathbf{x}\|_r^2 \tag{9.12}$$

In other words $B(\boldsymbol{\theta}, R')$ *is safe.*

Proof. The proof, which was first derived in [Dantas & Gribonval 2018], is provided in Appendix C.4. □

By applying Theorem (10) to $\mathbf{x}_t$ and $\boldsymbol{\theta}'_t$ in (9.7) we obtain a stable safe region $\mathcal{R} = B(\mathbf{c}', R')$ with

$$\text{Stable GAP Safe:} \quad \mathbf{c}' = \boldsymbol{\theta}'_t, R' = \frac{1}{\lambda}\sqrt{2G(\mathbf{x}_t, \boldsymbol{\theta}'_t|\tilde{\mathbf{D}}) + 2\delta(\mathbf{x_t})}. \tag{9.13}$$

The first columns of Table 9.1 summarize the obtained safe regions alongside with their previously existing analogue.

Combining stable safe sphere regions with stable safe screening tests (cf Section 9.1.1) yields screening rules which remain safe despite manipulating an approximate dictionary (last column of Table 9.1). As before, these new rules also reuse the calculations performed by the optimization algorithm with the approximate dictionary $\tilde{\mathbf{D}}$, which is crucial to make them suitable in practice. This point will be detailed in chapter 10.

1. In this theorem, $\theta \in \Delta_{\mathbf{D}}$ can be arbitrary. In particular it is not necessarily expressed as in (9.7), hence it does not need to belong to $\Delta_{\tilde{\mathbf{D}}}$.

2. Since the center $\mathbf{c} = \mathbf{y}/\lambda$ is independent of the iteration count t, the resulting test can be simplified provided that $\mathbf{D}^T\mathbf{y}$ can be pre-computed.

Sphere test	**Center**	**Radius**	**Test expression** - $\mu(\mathbf{d}\|B(\mathbf{c},R))$ or $\mu(\mathcal{S}'\|B(\mathbf{c}',R'))$
Static Safe	$\mathbf{y}/\lambda$	$\|1/\lambda_{\max} - 1/\lambda\| \cdot \|\mathbf{y}\|_2$	$\|\mathbf{d}^T\mathbf{y}\|/\lambda + \|1/\lambda_{\max} - 1/\lambda\| \cdot \|\mathbf{y}\|_2 \cdot \|\mathbf{d}\|_2$
Stable Static Safe	$\mathbf{y}/\lambda$	$\|1/\lambda'_{\max} - 1/\lambda\| \cdot \|\mathbf{y}\|_2$	$\|\tilde{\mathbf{d}}^T\mathbf{y}\|/\lambda + \epsilon\|\mathbf{y}\|_{q^*}/\lambda + \|1/\lambda'_{\max} - 1/\lambda\| \cdot \|\mathbf{y}\|_2 \cdot \|\mathbf{d}\|_2$
Dynamic Safe	$\mathbf{y}/\lambda$	$\|\boldsymbol{\theta}_t - \mathbf{y}/\lambda\|_2$	$\|\mathbf{d}^T\mathbf{y}\|/\lambda + \|\boldsymbol{\theta}_t - \mathbf{y}/\lambda\|_2 \cdot \|\mathbf{d}\|_2$
Stable Dynamic Safe	$\mathbf{y}/\lambda$	$\|\boldsymbol{\theta}'_t - \mathbf{y}/\lambda\|_2$	$\|\mathbf{d}^T\mathbf{y}\|/\lambda + \|\boldsymbol{\theta}'_t - \mathbf{y}/\lambda\|_2 \cdot \|\mathbf{d}\|_2$, if $\mathbf{d}^T\mathbf{y}$ is given[2]
			$\|\tilde{\mathbf{d}}^T\mathbf{y}\|/\lambda + \epsilon\|\mathbf{y}\|_{q^*}/\lambda + \|\boldsymbol{\theta}'_t - \mathbf{y}/\lambda\|_2 \cdot \|\mathbf{d}\|_2$, otherwise
GAP Safe	$\boldsymbol{\theta}_t$	$\frac{1}{\lambda}\sqrt{2G(\mathbf{x}_t, \boldsymbol{\theta}_t\|\mathbf{D})}$	$\|\mathbf{d}^T\boldsymbol{\theta}_t\| + \frac{1}{\lambda}\sqrt{2G(\mathbf{x}_t, \boldsymbol{\theta}_t\|\mathbf{D})} \cdot \|\mathbf{d}\|_2$
Stable GAP Safe	$\boldsymbol{\theta}'_t$	$\frac{1}{\lambda}\sqrt{2G(\mathbf{x}_t, \boldsymbol{\theta}'_t\|\tilde{\mathbf{D}}) + 2\delta(\mathbf{x}_t)}$	$\|\tilde{\mathbf{d}}^T\boldsymbol{\theta}'_t\| + \epsilon\|\boldsymbol{\theta}'_t\|_{q^*} + \frac{1}{\lambda}\sqrt{2G(\mathbf{x}_t, \boldsymbol{\theta}'_t\|\tilde{\mathbf{D}}) + 2\delta(\mathbf{x}_t)} \cdot \|\mathbf{d}\|_2$

Table 9.1 – Sphere center $\mathbf{c}$ and radius R for Static Safe, Dynamic Safe and GAP Safe ($\mathbf{c}'$ and R' for the stable versions). See definitions of $\lambda_{\max}$ in (7.2), of $\lambda'_{\max}$ in (9.8), of $\boldsymbol{\theta}_t$ in (8.9), of $\boldsymbol{\theta}'_t$ in (9.7), and of $\delta(\mathbf{x}_t)$ in (9.12) with $\mathbf{x} = \mathbf{x}_t$. Test expressions for $\mu(\mathbf{d}|B(\mathbf{c},R))$ – see (8.6) – and $\mu(\mathcal{S}'|B(\mathbf{c}',R'))$ – see (9.5) where $\mathcal{S}'$ is defined in (9.4) with $a = \|\mathbf{d}\|_2$.

3. Since $\mathbf{c} = \mathbf{y}/\lambda$ is independent of the iteration count t, the resulting test can be simplified provided that $\mathbf{D}^T\mathbf{y}$ can be pre-computed.

Chapter 10

A Fast Algorithm for ℓ_1 regularization

Having defined in chapter 9 safe screening rules which are robust to approximation errors on the dictionary matrix, we now have the tools to define an algorithm combining safe screening and the use of fast (but approximate) structured dictionaries.

10.1 Proposed Algorithm

The proposed Algorithm 13 consists in incorporating a stable dynamic screening rule to the iterations of a proximal algorithm. Besides the vector $\mathbf{y}$ and the regularization parameter λ, the input of the algorithm consists in a sequence $\{\tilde{\mathbf{D}}^i\}_{i=0}^I$ of approximate dictionaries with $\tilde{\mathbf{D}}^I = \mathbf{D}$, and a corresponding sequence of error bounds $\{\boldsymbol{\epsilon}^i\}_{i=0}^I$, where $\boldsymbol{\epsilon}^i = (\epsilon_j^i)_{j=1}^m \in \mathbb{R}_+^m$ contains ℓ_2 error bounds on the atoms of the i-th dictionary $\tilde{\mathbf{D}}^i$, and of course $\boldsymbol{\epsilon}^I = \mathbf{0}$.

In general terms, the proposed strategy consists in gradually switching to more accurate dictionary approximations as the optimization algorithm approaches the solution. Note that any specific iterative optimization technique can be used by replacing the generic update function $p(\mathbf{x}_t, \mathbf{D}, \boldsymbol{\alpha}_t)$ by the corresponding specific update step, for instance equation (7.3) for ISTA. A possible variation consists in performing the screening at regular intervals instead of every iteration.

To fully describe this algorithm we need to specify the stopping and switching criteria, which involve a parameter Γ. In the following section, we present a switching criterion which assumes the approximations $\{\tilde{\mathbf{D}}\}_{i=0}^I$ to be arranged in increasing order of precision. A classical stopping criterion is to stop when the duality gap falls below a threshold, recalling that strong duality holds for problem (7.1), i.e. $G(\mathbf{x}^\star, \boldsymbol{\theta}^\star | \mathbf{D}) = 0$ (see Property 7 in chapter 2). The simplest way of computing the duality gap is to wait until the algorithm reaches $\tilde{\mathbf{D}} = \tilde{\mathbf{D}}^I = \mathbf{D}$. Alternatively one could upper bound the duality gap using

Algorithm 13 $\mathbf{x}^\star = \texttt{FastL1}(\{\tilde{\mathbf{D}}^i\}_{i=0}^I, \{\boldsymbol{\epsilon}^i\}_{i=0}^I, \mathbf{y}, \lambda, \Gamma)$

Initialize: $t = 0$, $\mathbf{x}_0 = \mathbf{0}$, $i = 0$, $\mathcal{A}_0 = \{1, \dots, m\}$,
$\boldsymbol{\alpha}_0$ according to the considered proximal algorithm
while stopping criterion not met **do**
    —– *Restrict to preserved set* —–
    $\tilde{\mathbf{D}} \leftarrow \tilde{\mathbf{D}}^i_{[\mathcal{A}_t]}, \quad \mathbf{x}_t \leftarrow (\mathbf{x}_t)_{[\mathcal{A}_t]}$
    —– *Update preserved coefficients* —–
    $\{\mathbf{x}_{t+1}, \boldsymbol{\alpha}_{t+1}\} \leftarrow p(\mathbf{x}_t, \tilde{\mathbf{D}}, \boldsymbol{\alpha}_t)$
    —– *Dynamic Screening* —–
    Set $\boldsymbol{\theta}'_t$ using (9.7) and $\mathbf{c}'$, R' according to Table 9.1
    $\mathcal{A}_{t+1} \leftarrow \{j \in \mathcal{A}_t : \mu(\mathcal{S}'_j | B(\mathbf{c}', R')) \geq 1\}$
    $t \leftarrow t + 1$
    —– *Switching criterion* —–
    Compute γ_t (cf. (10.1)) and K_t (cf. (10.11))
    $i = \texttt{SwitchDictionary}(i, I, \gamma_t, \Gamma, K_t)$
end while
Return: $\mathbf{x}^\star \leftarrow \mathbf{x}_t$ zero-padded in $\mathcal{A}_t^\complement$.

approximate dictionaries: $G(\mathbf{x}, \boldsymbol{\theta}|\tilde{\mathbf{D}}) + \delta(\mathbf{x}) \geq G(\mathbf{x}, \boldsymbol{\theta}|\mathbf{D})$ with $\delta(\mathbf{x})$ defined as in (9.12) (this bound is obtained as part of our proof of Theorem 10 concerning the stable GAP Safe sphere and given in Appendix C.4).

10.2 Switching criterion

If an approximate dictionary $\tilde{\mathbf{D}}$ was kept until convergence, the solution of a different problem (the one defined by $\tilde{\mathbf{D}}$) would be obtained. It is, therefore, necessary to switch back to the original dictionary at some point.

In general, a switching criterion has two main motivations:

- **Convergence:** to avoid diverging from the solution of the original problem (7.1). The higher the error bounds $\boldsymbol{\epsilon}^i$, the further away we move from the true solution, therefore the need to switch earlier in such cases.

- **Speed:** once screened enough, the original dictionary can become faster to apply than the approximate ones. This is crucial at high regularization regimes ($\lambda/\lambda_{\max} \approx 1$) in which screening typically occurs very quickly.

10.2.1 Convergence criterion

Given a solution estimate $\mathbf{x}_t$ at iteration t, a natural convergence measure for the problem $\mathcal{L}(\lambda, \mathbf{D}, \mathbf{y})$ is the duality gap $G(\mathbf{x}_t, \boldsymbol{\theta}_t|\mathbf{D})$ (8.12) with $\boldsymbol{\theta}_t$ calculated as in (8.9). However, when manipulating approximate dictionaries, this quantity is not accessible without unwanted additional computations.

As a surrogate, we propose to use a ratio

$$\gamma_t = \frac{G(\mathbf{x}_t, \tilde{\boldsymbol{\theta}}_t|\tilde{\mathbf{D}})}{G(\mathbf{x}_t, \boldsymbol{\theta}'_t|\tilde{\mathbf{D}})} \tag{10.1}$$

between two computable gaps arising from two different dual points $\boldsymbol{\theta}'_t$ and $\tilde{\boldsymbol{\theta}}_t$ that are respectively feasible for problems $\mathcal{L}(\lambda, \mathbf{D}, \mathbf{y})$ and $\mathcal{L}(\lambda, \tilde{\mathbf{D}}, \mathbf{y})$:

$$\boldsymbol{\theta}'_t \in \Delta_{\mathbf{D}} \cap \Delta_{\tilde{\mathbf{D}}} \longrightarrow G(\mathbf{x}_t, \boldsymbol{\theta}'_t|\tilde{\mathbf{D}}) = P(\mathbf{x}_t|\tilde{\mathbf{D}}) - D(\boldsymbol{\theta}'_t) \tag{10.2}$$

$$\tilde{\boldsymbol{\theta}}_t \in \Delta_{\tilde{\mathbf{D}}} \longrightarrow G(\mathbf{x}_t, \tilde{\boldsymbol{\theta}}_t|\tilde{\mathbf{D}}) = P(\mathbf{x}_t|\tilde{\mathbf{D}}) - D(\tilde{\boldsymbol{\theta}}_t) \tag{10.3}$$

where $\boldsymbol{\theta}'_t \in \Delta_{\mathbf{D}} \cap \Delta_{\tilde{\mathbf{D}}}$ has been previously calculated for the stable screening with (9.7), while $\tilde{\boldsymbol{\theta}}_t \in \Delta_{\tilde{\mathbf{D}}}$ is the conventional dual point (8.9) but calculated with $\tilde{\mathbf{D}}$. Then again, apart from minor extra memory requirements, the computation of the quantities in (10.3) reuses most of the calculations[1] in (10.2).

If $\tilde{\mathbf{D}} = \tilde{\mathbf{D}}^i$ was kept until convergence, we would have

$$\tilde{\mathbf{x}}_t \to \tilde{\mathbf{x}}^\star \qquad \text{and} \qquad \tilde{\boldsymbol{\theta}}_t \to \tilde{\boldsymbol{\theta}}^\star. \tag{10.4}$$

with $\tilde{\mathbf{x}}^\star$ a solution of $\mathcal{L}(\lambda, \tilde{\mathbf{D}}, \mathbf{y})$ and $\tilde{\boldsymbol{\theta}}^\star$ a solution of the corresponding dual problem. Then, as illustrated in Fig. 10.1 (left plot), the second quantity $G(\mathbf{x}_t, \tilde{\boldsymbol{\theta}}_t|\tilde{\mathbf{D}})$ –which is precisely the duality gap w.r.t. the problem $\mathcal{L}(\lambda, \tilde{\mathbf{D}}, \mathbf{y})$– would tend to zero, while the first quantity $G(\mathbf{x}_t, \boldsymbol{\theta}'_t|\tilde{\mathbf{D}})$ would typically saturate at a nonzero value since $\boldsymbol{\theta}'_t \nrightarrow \tilde{\boldsymbol{\theta}}^\star$. As a result, the ratio γ_t would shrink with the iterations as illustrated in Fig. 10.1 (right plot).

With that in mind, we propose to switch dictionaries when

$$\gamma_t \leq \Gamma \tag{10.5}$$

where the higher $\Gamma \in [0, 1]$, the greater the sensibility to the gap saturation.

1. Only n extra products for computing $\|\tilde{\boldsymbol{\theta}}\|$ and $|\mathcal{A}_t|$ comparisons for $\|\tilde{\mathbf{D}}^T\boldsymbol{\rho}_t\|_\infty$ are required.

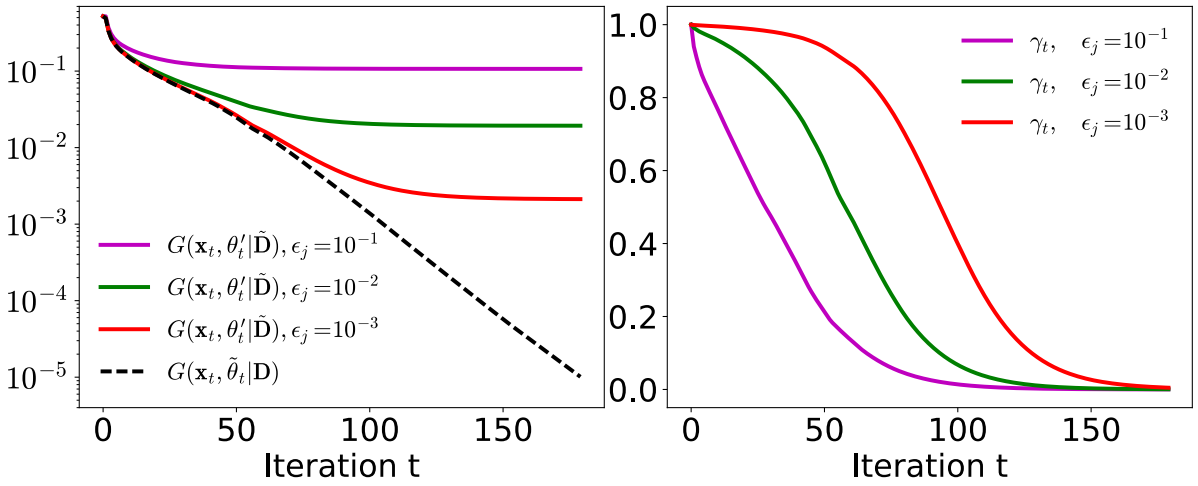


Figure 10.1 – Typical behaviour of duality gaps (left) and ratio γ_t (right) as a function of the iteration number and error bounds.

10.2.2 Speed criterion

In terms of speed, the general idea is to exploit tradeoffs between approximation and computational efficiency.

Denoting $\mathcal{C}(\mathbf{D}, \mathcal{A})$ the computational cost of multiplying matrix $\mathbf{D}_{[\mathcal{A}]}$ by a vector, it is reasonable to assume that $\mathcal{C}(\mathbf{D}, \mathcal{A}) \leq \mathcal{C}(\mathbf{D}, \mathcal{A}')$ if $\mathcal{A} \subset \mathcal{A}'$. As a result, given the nested structure of the sequence of preserved sets $\mathcal{A}_t$ (by construction, see line 10 in Alg. 13), the sequence $\mathcal{C}(\mathbf{D}, \mathcal{A}_t)$, $t \geq 0$ is non-increasing.

Consider two approximate dictionaries $\tilde{\mathbf{D}}^i$ and $\tilde{\mathbf{D}}^j$ and assume the initial "unscreened" computational costs satisfy $\mathcal{C}(\tilde{\mathbf{D}}^i, \mathcal{A}_0) < \mathcal{C}(\tilde{\mathbf{D}}^j, \mathcal{A}_0)$. Denote $\mathbf{E}^i$, $\mathbf{E}^j$ the corresponding approximation errors (cf (7.4)), and assume that $\mathbf{E}^i$ is "larger" than $\mathbf{E}^j$ (let us denote this $\mathbf{E}^i \succ \mathbf{E}^j$) in the sense that for each column the approximation error is larger. An irrevocable switching point would be an iteration T in which the coarser approximation $\tilde{\mathbf{D}}^i$ becomes more complex than $\tilde{\mathbf{D}}^j$

$$\forall t > T, \quad \mathcal{C}(\tilde{\mathbf{D}}^i, \mathcal{A}_t) \geq \mathcal{C}(\tilde{\mathbf{D}}^j, \mathcal{A}_t). \tag{10.6}$$

Note that this only happens if $\mathcal{C}(\tilde{\mathbf{D}}^i, \mathcal{A}_t)$ decreases faster than $\mathcal{C}(\tilde{\mathbf{D}}^j, \mathcal{A}_t)$ as screening progresses (i.e. as $|\mathcal{A}_t|$ gets smaller).

To use the general idea just described, one needs to specify a complexity model $\mathcal{C}(\mathbf{D}, \mathcal{A}_t)$ depending on the structure of the corresponding matrix. For a generic unstruc-

tured dictionary matrix $\mathbf{D} \in \mathbb{R}^{n \times m}$, a simple model is:

$$\mathcal{C}(\mathbf{D}, \mathcal{A}_t) = n|\mathcal{A}_t| \tag{10.7}$$

To quantify the complexity reduction on matrix-vector products entailed by an approximate dictionary we will use the concept of *Relative Complexity* (RC) [Magoarou & Gribonval 2016], such that

$$\mathcal{C}(\tilde{\mathbf{D}}, \mathcal{A}_0) = \mathrm{RC}(\tilde{\mathbf{D}}) \times nm. \tag{10.8}$$

In a worst-case scenario, screening does not further reduce the cost and $\mathcal{C}(\tilde{\mathbf{D}}, \mathcal{A}_t) = \mathrm{RC}(\tilde{\mathbf{D}}) \times nm$. In more optimistic scenarios, some fast approximate dictionary structures might still benefit from speedups upon column removal.

Consider approximate dictionaries $\{\tilde{\mathbf{D}}^0, \ldots, \tilde{\mathbf{D}}^I\}$, with $\tilde{\mathbf{D}}^I = \mathbf{D}$, of increasing precision and decreasing complexity

$$\mathbf{E}^1 \succ \ldots \succ \mathbf{E}^I = \mathbf{0} \quad \text{and} \quad \mathrm{RC}(\tilde{\mathbf{D}}^0) < \ldots < \mathrm{RC}(\tilde{\mathbf{D}}^I) = 1. \tag{10.9}$$

The above observations suggest to switch from $\tilde{\mathbf{D}}^i$ to $\mathbf{D}$ if $\mathcal{C}(\tilde{\mathbf{D}}^i, \mathcal{A}_t) \geq \mathcal{C}(\mathbf{D}, \mathcal{A}_t)$, i.e. $\mathrm{RC}(\tilde{\mathbf{D}}^i) \times nm \geq n|\mathcal{A}_t|$, that is to say if

$$|\mathcal{A}_t| \leq \mathrm{RC}(\tilde{\mathbf{D}}^i)\ m. \tag{10.10}$$

Also, as a consequence of the complexity model (10.8) adopted for the approximate dictionaries $\tilde{\mathbf{D}}$, this criterion never causes a switching from $\tilde{\mathbf{D}}^i$ to $\tilde{\mathbf{D}}^j$ with $i < j \neq I$. In other words, this criterion triggers a switching directly to the true dictionary $\mathbf{D}$.

We will see, however, that this criterion is not directly applicable in practice, which leads to the heuristic defined below.

10.2.3 Heuristic look-ahead speed criterion

The preserved set size $|\mathcal{A}_t|$ used in (10.10) should ideally be the one associated to the dictionary *after* switching ($\mathbf{D}$ in this case), while in practice only the screening rate associated to the *current* approximation $\tilde{\mathbf{D}}^i$ is available. This motivates the introduction of a heuristic (that preserves the safety of screening) to anticipate the potentially smaller size of the preserved set $|\mathcal{A}_t|$ as soon as we switch to a more precise approximation (or to

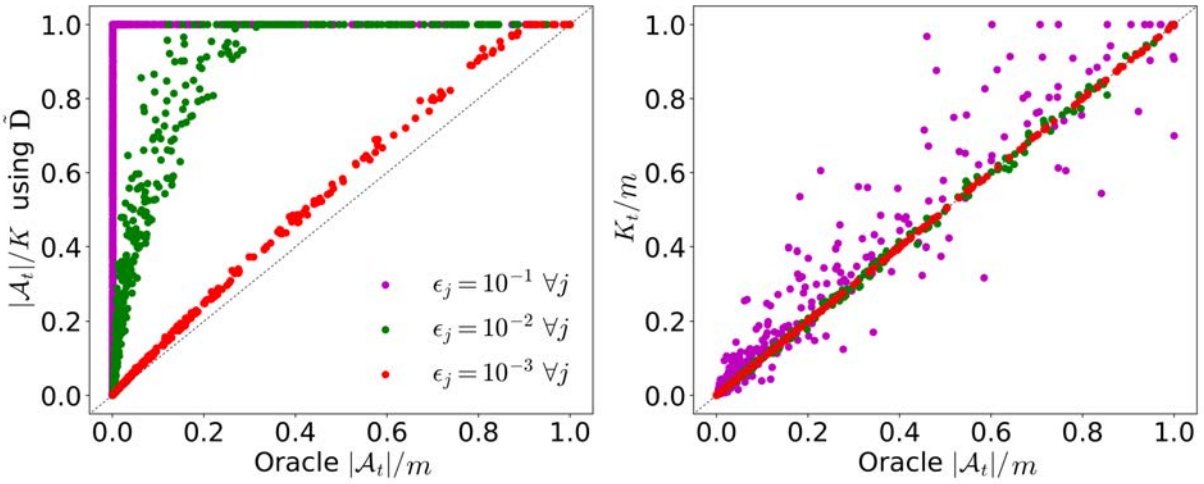


Figure 10.2 – Number of preserved atoms $|\mathcal{A}|$ using oracle conventional screening (x-axis) *vs* stable screening (y-axis, left plot), or *vs* proposed heuristic (y-axis, right plot). .

the original dictionary).

Indeed, stable screening generally leads to less atom eliminations due to the extra security margins (and more so for higher approximation errors). This is illustrated in Figure 10.2 (left) with a scatterplot in which the x-axis correspond to the "oracle" number of preserved atoms $|\mathcal{A}|$ obtained with a conventional Dynamic Safe screening mechanism having access to $\mathbf{D}$. This is the quantity we are interested in estimating for the speed criterion (10.10). The y-axis corresponds to the number of preserved atoms obtained by the corresponding stable test for a given approximation $\tilde{\mathbf{D}}$. Each plotted point compares the two mentioned screening ratios at a given iteration. The procedure was repeated a large number of times in order to give a full picture of how the two quantities correlate to one another. The main observation is that all points lie above the identity line which means that the stable test always overestimates (sometimes significantly) the value that $|\mathcal{A}_t|$ will reach after switching. This phenomenon is intensified for higher approximation errors.

As a heuristic, we propose to estimate $|\mathcal{A}_t|$ using the *conventional* test on the approximate atoms, i.e. $\mu(\tilde{\mathbf{d}}_j|\mathcal{R})$.

$$K_t = |\{j \in \{1, \dots, m\} : \mu(\tilde{\mathbf{d}}_j|\mathcal{R}) \geq 1\}| \tag{10.11}$$

As shown in Fig. 10.2 (right graph), this correlates much better with the oracle value of $|\mathcal{A}_t|$, since it doesn't have the additional security margins used in the stable test. Even

though the test used to compute K_t is unsafe with respect to the original problem (7.1), it has no impact on the safety of screening itself, which is still performed with the stable test. Moreover, the additional test is virtually costless since the calculations of the first test are reused.

10.2.4 Summary and example

The resulting switching strategy, shown in Algorithm 14, plays two roles:

(I) Decide when to stop using the current dictionary.

(II) Choose the next dictionary.

Algorithm 14 $j = \texttt{SwitchDictionary}(i, I, \gamma_t, \Gamma, K_t)$

if $K_t \leq \mathrm{RC}(\tilde{\mathbf{D}}^i)\ m$ **then** ▷ Speed criterion
    $j \leftarrow I$
else if $\gamma_t \leq \Gamma$ **then** ▷ Convergence criterion
    $j \leftarrow i + 1$
else
    $j \leftarrow i$
end if

The speed criterion (section 10.2.2) triggers a switching directly to the original dictionary $\tilde{\mathbf{D}}^I = \mathbf{D}$. We suppose the Relative Complexity associated to the i-th approximation $\mathrm{RC}(\tilde{\mathbf{D}}^i)$ (cf. (10.8)) to be known and stored in memory. The convergence criterion (section 10.2.1) switches to the next available approximation $\tilde{\mathbf{D}}^{i+1}$. Note that if $\mathbf{D}$ is already adopted, i.e. input $i = I$, then $\mathbf{D}$ is kept for all remaining iterations.

Example Figure 10.3 shows an example of the proposed algorithm, given a list of approximations $\{\tilde{\mathbf{D}}^i\}_{i=0}^I$ with $I = 4$, decreasing approximation error $\boldsymbol{\epsilon}^i$ and increasing relative complexity $\mathrm{RC}(\tilde{\mathbf{D}}^i) = 0.15(i+1)$, $0 \leq i < I$. As usual, $\tilde{\mathbf{D}}^I = \mathbf{D}$.

Figure 10.3a shows the duality gap evolution over the iterations with $\Gamma = 0.2$. As soon as the gap $G(\mathbf{x}_t, \boldsymbol{\theta}'_t | \tilde{\mathbf{D}}^i)$ saturates, a more precise approximation is adopted (the next on the list), avoiding to excessively delay the convergence compared to the conventional case where screening is performed with $\mathbf{D}$ from the beginning (dotted curve). Although some delay is introduced in terms of iterations until convergence, it is largely compensated by

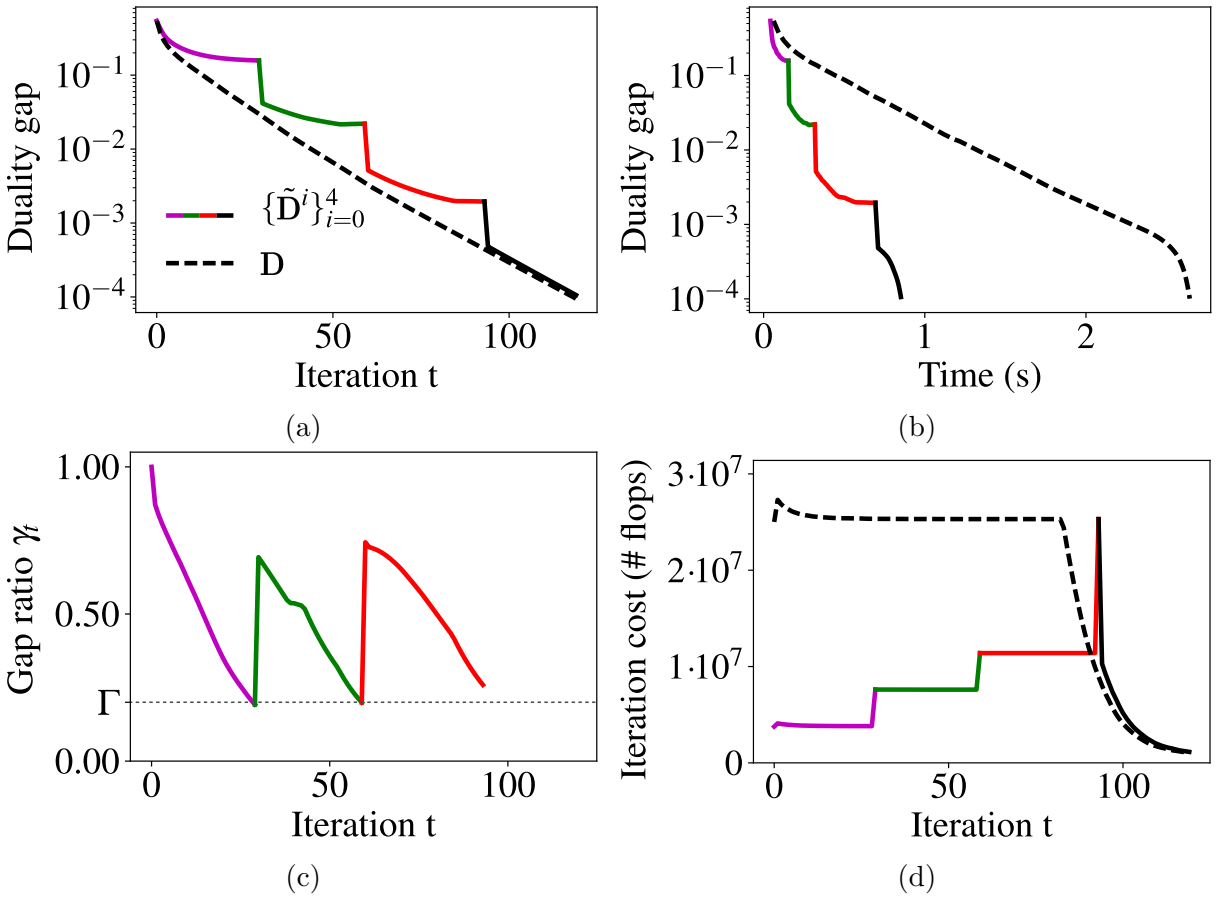


Figure 10.3 – Application example of the proposed algorithm. (a) and (b) Duality gap over the iterations and time respectively. (c) Gap ratio γ_t (cf. (10.1)) for the convergence switching criterion, with threshold $\Gamma = 0.2$. (d) Computational complexity per iteration (worst-case theoretical number of flops, cf. section 10.2.2). Parameters: $n = 2500$, $m = 10000$, $\lambda/\lambda_{max} = 0.2$, convergence threshold on duality gap $tol = 10^{-4}$.

the fact that the initial iterations with the approximate dictionaries are much faster. This is illustrated in Fig. 10.3b in which the duality gap is plotted as a function of the execution time.

The theoretical complexity per iteration is shown in Fig. 10.3d. As better approximations are adopted, the iteration complexity rises proportionally to the corresponding RC. Note that as soon as the conventional dictionary becomes faster than the current approximation, it is promptly adopted (by the speed criterion). In this example, the last approximation $\tilde{\mathbf{D}}^{i=3}$ was never used, since we switched directly from $\tilde{\mathbf{D}}^{i=2}$ to $\mathbf{D}$.

10.3 Complexity Analysis

Existing screening tests introduce only a minor computational overhead because they primarily reuse matrix-vector multiplications either already performed in the optimization algorithm update $p(\mathbf{x_t}, \mathbf{D}, \boldsymbol{\alpha_t})$ (typically the product $\mathbf{D}^T\boldsymbol{\rho}$) or that can be precalculated once for all ($\mathbf{D}^T\mathbf{y}$). The same holds for the stable tests proposed in this article. We now derive the involved number of floating point operations (flops). Since static screening rules represent only a fixed computational overhead, we concentrate on dynamic rules which could potentially (if not properly designed) lead to a significant overhead.

10.3.1 Screening cost

The most expensive computations associated to a dynamic screening rule – the ones potentially of the order of a matrix vector product $\mathcal{O}(nm)$ – are:

- Computation of a dual feasible point $\boldsymbol{\theta}_t$ in (8.9) (resp. $\boldsymbol{\theta}'_t$ (9.7)): requires the product $\mathbf{D}^T\boldsymbol{\rho}$ (resp. $\tilde{\mathbf{D}}^T\tilde{\boldsymbol{\rho}}$) reused from the optimization algorithm.

- Sphere test $\mu(\mathbf{d}|\mathcal{R})$ (resp. $\mu(\mathcal{S}'_j|\mathcal{R})$): requires the product $\mathbf{d}_j^T\mathbf{c}$ (resp. $\tilde{\mathbf{d}}_j^T\mathbf{c}$) for all preserved atoms $j \in \mathcal{A}$, which comes down to the matrix-vector product $\mathbf{D}^T\mathbf{c}$ (resp. $\tilde{\mathbf{D}}^T\mathbf{c}$). Practical sphere regions have been designed to limit this potential overhead. While in the Dynamic Safe sphere it reduces to the precalculated product $\mathbf{D}^T\mathbf{y}$, in the GAP Safe sphere it reduces to $\mathbf{D}^T\boldsymbol{\rho}$ (resp. $\tilde{\mathbf{D}}^T\tilde{\boldsymbol{\rho}}$) calculated in the optimization iteration.

The other required calculations are detailed in Appendix C.5. In short, the screening represents a rather low overhead $\mathcal{O}(n + |\mathcal{A}|)$ –even its stable version– compared to the optimization update: $\mathcal{O}(n|\mathcal{A}|)$ with screening or $\mathcal{O}(nm)$ without it, due to matrix-vector products.

10.3.2 Full iteration cost

Table 11.2 shows the number of operations of a complete iteration in Algorithm 13 (ISTA update + screening), following [Bonnefoy *et al.* 2015] and adopting the complexity models in equations (10.7) and (10.8) for matrix vector multiplications. We denote $\text{flops}_{\mathbf{D}}(t)$ the cost of iteration t with the conventional screening and $\text{flops}_{\tilde{\mathbf{D}}}(t)$ with the stable screening.

As a benchmark, we use the complexity of an ISTA iteration without screening, denoted $\text{flops}_N(t)$.

$\text{flops}_N(t)$	$(m + \|\mathbf{x}_t\|_0)n + 4m + n$
$\text{flops}_{\mathbf{D}}(t)$	$(\|\mathcal{A}\| + \|\mathbf{x}_t\|_0)n + 6\|\mathcal{A}\| + 5n$
$\text{flops}_{\tilde{\mathbf{D}}}(t)$	$(\text{RC}\times m + \|\mathbf{x}_t\|_0)n + 8\|\mathcal{A}\| + 7n$

Table 10.1 – Complete iteration complexity

To obtain the total complexity of the algorithm, simply add up all iteration costs calculated with the corresponding active set size $|\mathcal{A}_t|$ and sparsity of the solution estimate $\|\mathbf{x}_t\|_0$ at iteration t.

Chapter 11

Experiments

In this chapter we demonstrate the potential of the proposed Algorithm 13 in terms of complexity reduction and time saving for ℓ_1-minimization problems. This is done in a wide set of simulation scenarios, summarized in Table 11.1, to evaluate the influence of the main parameters involved.

Source code for reproducible research is available online [Dantas & Gribonval 2019b].

Problem parameters	**Values**
Regularization λ/λ_{max}	$[10^{-2}, 1]$ (logarithmically-spaced)
Convergence tolerance	$G(\mathbf{x}_t, \boldsymbol{\theta}_t \vert \mathbf{D}) < tol \in [10^{-6}, 10^{-3}]$
Data distribution $\mathbf{y}$	$\mathbf{Dx}$ with $\mathbf{x}$ Bernoulli-Gaussian
Algorithm parameters	**Values**
Optimization algorithm	ISTA, FISTA
Screening Test	(stable) Dynamic, (stable) GAP Safe
Switching threshold Γ	$[0.01, 0.8]$ (logarithmically-spaced)
Speedup-Error tradeoff $\tilde{\mathbf{D}}^i$	3 scenarios (see Section 11.1)

Table 11.1 – Pool of parameters explored in the simulations.

The dimension of the observed vector $\mathbf{y}$ is $n = 2500$ and that of the coefficient vector $\mathbf{x}$ (or, equivalently, the number of atoms in the dictionary) is $m = 10000$. Unit-norm input data samples $\mathbf{y} = \mathbf{Dx}$ are generated from a sparse vector $\mathbf{x}$ with support determined by a Bernoulli distribution with probability $p = 0.02$ and zero-mean standard Gaussian entries. The reported results are the average on 25 independent and identically distributed realizations of the input $\mathbf{y}$. The dictionary matrix $\mathbf{D} \in \mathbb{R}^{n \times m}$ is generated in such a way that it is more or less efficiently approximated by a fast structured matrix according to three representative scenarios (more details are given in Section 11.1).

Although other problem dimensions (n and m) and data distributions were explored,

we decided to keep these parameters fixed on the reported experiments since they were observed not to decisively influence the analyzed results.

In all figures, the GAP Safe and Dynamic Safe tests are denoted respectively by the acronyms **GAP** and **DST**.

11.1 Specifying the Fast Structured dictionaries

The performance of the proposed algorithm is directly associated to the *quality* of the available approximations $\tilde{\mathbf{D}}^i$. A good approximation would have both a small application complexity (RC) and small approximation error ($\boldsymbol{\epsilon}$). There is a compromise, however, since each of these features usually come to the price of the other.

In the experiments, we use a particular kind of fast structured dictionaries: the SuKro dictionaries discussed in part II of this thesis,which can be written as a sum of Kronecker products $\tilde{\mathbf{D}} = \sum_{r=1}^{n_{\text{kron}}} \mathbf{B}_r \boxtimes \mathbf{C}_r$. Its reduced multiplication cost comes from the fact that the sub-matrices $\mathbf{B}_r$, $\mathbf{C}_r \in \mathbb{R}^{\sqrt{n} \times \sqrt{m}}$ are much smaller than $\mathbf{D} \in \mathbb{R}^{n \times m}$. The choice of the SuKro structure is justified by the fact that it directly provides a range of speedup-error compromises by tweaking the number of Kronecker terms (n_{kron}) in the sum. A higher n_{kron} provides a more precise approximation although implying a higher RC.

We define three representative simulation scenarios – hard, moderate and easy scenarios – in which the dictionary matrix is poorly, moderately and efficiently approximated by the structured dictionary respectively. In practical terms, this translates to different approximation error decay profiles as a function of the number of Kronecker terms (n_{kron}) on the SuKro structure, as shown in Figure 11.1. The easier the scenario, the faster the approximation error decays as a function of n_{kron}.

Although computational complexity associated to a certain SuKro operator can be calculated analytically, we measured the actual time speedup obtained in practice in order to have a more realistic value. Interestingly, the measured RC is lower than the theoretical prediction.[1]

1. Note that we do not observe the same overhead as in Table 5.3, section 5.4.2. This is because here we use a 2D-SuKro (i.e. with 2 blocks in the Kronecker product) and in this scenario the mode-products translate directly to usual matrix-matrix products: $(\mathbf{B} \boxtimes \mathbf{C})\mathbf{v} = \text{vec}(\mathbf{V} \times_1 \mathbf{C} \times_2 \mathbf{B}) = \text{vec}(\mathbf{C}\mathbf{V}\mathbf{B}^T)$, with $\mathbf{v} = \text{vec}(\mathbf{V})$. Therefore, the successive rearrangement of the data in memory are no longer required. Now, the superiority w.r.t. the theoretical predictions can be explained by the fact that some parallelization is introduced (in the summing terms) and matrix-matrix products are faster than the naive cubic complexity.

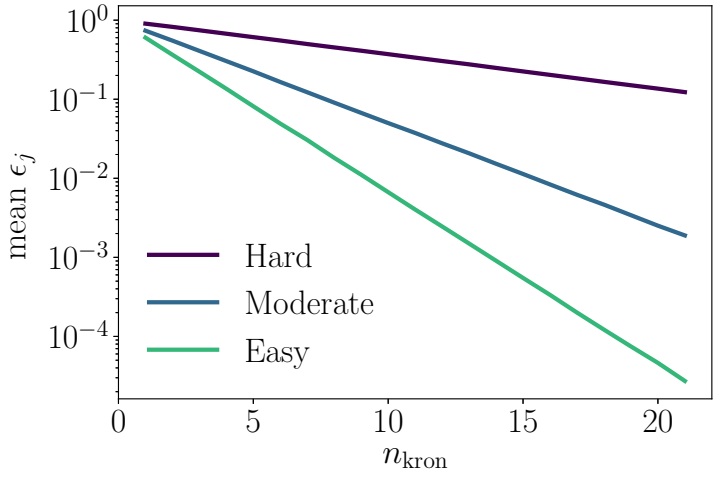


Figure 11.1 – Simulated scenarios: different trade-offs between approximation error and speedup of the structured dictionaries.

n_{kron}	**5**	**10**	**15**	**20**
Measured RC	0.08	0.15	0.21	0.28
Theoretical RC	0.15	0.30	0.45	0.60

Table 11.2 – Relative Complexities (RC)

11.2 Computational complexity and Time results

Let us denote F_N, $F_{\mathbf{D}}$ and $F_{\tilde{\mathbf{D}}}$ respectively the total computational complexity (in number of flops) of the optimization algorithm without screening, with the conventional screening and with the proposed approach, such that

$$F_X = \sum_{t=1}^{n_{\text{it}}} \text{flops}_X(t), \quad X \in \{N, \mathbf{D}, \tilde{\mathbf{D}}\} \tag{11.1}$$

where n_{it} is the number of iterations. When calculating $F_{\tilde{\mathbf{D}}}$, the current approximation at iteration t must be used in the expression of $\text{flops}_{\tilde{\mathbf{D}}}(t)$. Likewise, we denote T_N, $T_{\mathbf{D}}$ and $T_{\tilde{\mathbf{D}}}$ the measured running times.

As a main figure of merit, we evaluate the normalized number of flops ($F_{\mathbf{D}}/F_N$ and $F_{\tilde{\mathbf{D}}}/F_N$) and normalized running times ($T_{\mathbf{D}}/T_N$ and $T_{\tilde{\mathbf{D}}}/T_N$).[2]

In all simulated scenarios, the observed time reductions match closely the theoret-

2. Time results were obtained in an Intel® Core™ i7-5600U CPU @ 2.60GHz, 16GB RAM. But since mostly time ratios are reported, the results here should be relatively consistent with other machine specifications.

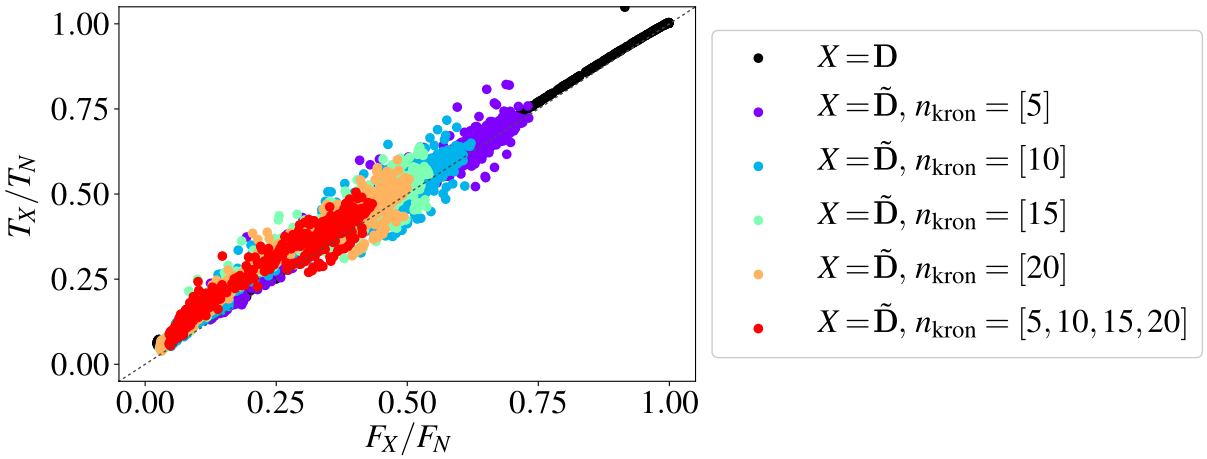


Figure 11.2 – Normalized number of flops vs. normalized running times for ISTA ($tol = 10^{-5}$, $\Gamma = 0.5$). Each point on the graph corresponds to an independent run at a given regularization $\lambda/\lambda_{\max} \in [10^{-2}, 1]$. Results for the usual GAP Safe test are shown in black and those using its stable version (as in Algorithm 13) are shown in colors – the red dots correspond to the multiple approximations case. The closer the dots lie to the identity line, the better the theoretical speedups are met in practice.

ical speedups predicted in terms of computational complexity, as illustrated in Figure 11.2 for the GAP Safe rule. It shows the correlation between theoretical complexities and measured running times in multiple independent runs. The fact that the points are well concentrated around the identity line is an empirical evidence that the predicted speedups really translate into practical accelerations. Similar results are obtained for the Dynamic Safe rule, for other convergence tolerances and switching thresholds. Given this observation, in the remainder of the chapter we report only running time results, which are more relevant in practice.

11.3 Choosing the switching threshold

The convergence-based switching criterion proposed in Section 10.2.1 relies on a hyper-parameter: the threshold Γ. As discussed then, this parameter determines how long the approximate dictionaries are kept.

We empirically observed that the choice of Γ is mostly dependent on the quality of the approximations $\tilde{\mathbf{D}}^i$ (represented here by the three simulation scenarios defined in Section 11.1). In Figure 11.3 we show the normalized times as a function of $\Gamma \in [10^{-2}, 0.8]$ for each one of the three scenarios. Each line corresponds to a different regularization level λ.

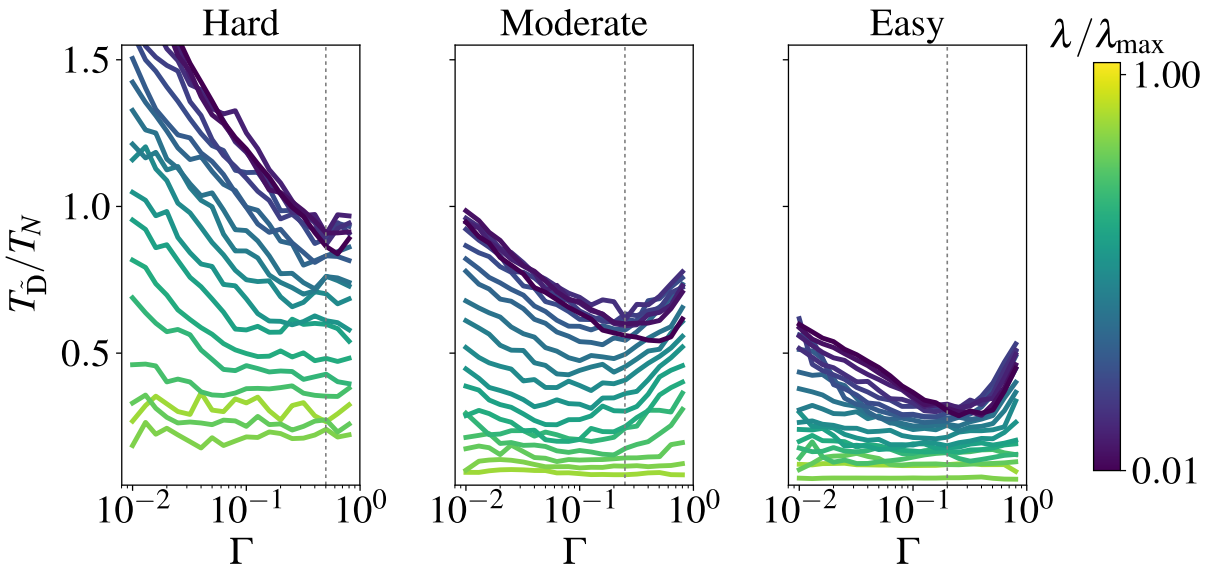


Figure 11.3 – Impact of the switching threshold Γ (x-axis) on the time results (y-axis) of Algorithm 13 using FISTA and GAP Safe. Each line corresponds to a regularization $\lambda/\lambda_{\max} \in [10^{-2}, 1]$.

For each of these lines, we are interested in the Γ value that minimizes the running times. We can see that the low regularization configurations (dark blue curves) are more sensitive to the choice of Γ, especially in the Hard scenario (left plot) in which a bad choice of Γ can even lead to normalized running times greater than one –so, actually, a slowdown. This happens when a too small Γ is used, which means that the approximate dictionaries are kept longer than they should, causing an important detour in the convergence path and thus delaying convergence.

There is no reason for the optimal Γ to be the same for every regularization level. However, a common behavior is observed regardless of the regularization and, for a given simulation scenario, a single Γ value can be chosen to obtain close to optimal execution times for any $\lambda/\lambda_{\max}$ in the tested range. Slightly different Γ values can be chosen to better adapt to each simulation scenario: $\Gamma = 0.5, 0.25, 0.2$ (indicated by the dotted vertical lines in the figure) respectively on the hard, moderate and easy scenarios. In general terms, the *worse* the available approximations are (i.e. the harder the scenario) the higher the Γ to be chosen. This is intuitive, since it implies being more conservative in switching earlier when the available approximations are of lower quality.

If the aimed speedup-error compromise is not fixed, a good general compromise is picking $\Gamma = 0.5$, for instance.

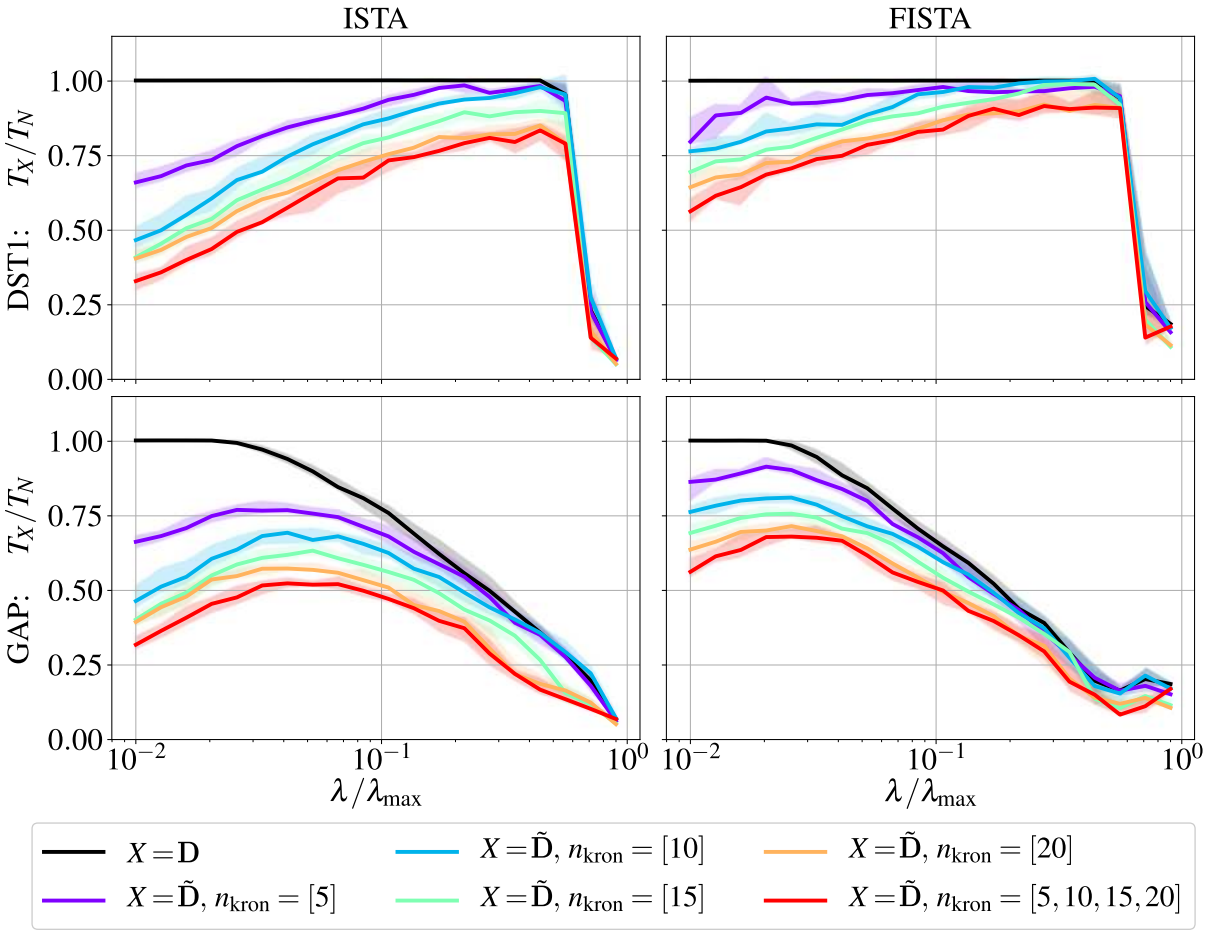


Figure 11.4 – Running times normalized w.r.t. T_N (ISTA or FISTA algorithms without screening) as a function of the regularization level. Simulations are done independently for each λ. Conventional and Stable screening results respectively in black and colored lines. Left: ISTA. Right: FISTA. Top: Dynamic Safe screening. Bottom: GAP Safe. Moderate scenario, $tol=10^{-5}$, $\Gamma=0.5$.

11.4 Single vs. Multiple Approximations

In our experiments, using multiple dictionary approximations proved to be always advantageous when compared to using a single approximation, as soon as the switching parameter Γ is well-tuned (see Section 11.3).

Figure 11.4 shows the normalized running times for an entire range of regularizations and corroborates the previous statement for both GAP and Dynamic Safe tests as well as for both ISTA and FISTA algorithms. The medians among 25 runs are plotted and the shaded area contains the 25%-to-75% percentiles. Although the single-approximation version of the proposed algorithm (as introduced in [Dantas & Gribonval 2017, Dantas

& Gribonval 2018]) already provides noticeable speedups to the tested gradient-based solvers, even with respect to their screening-based implementation (black curves), the use of multiple approximations (red curves) consistently leads to even better speedups. This ratifies the relevance of the generalization to multiple dictionaries introduced in [Dantas & Gribonval 2019a]. Similar results, which are omitted here, were obtained for other convergence tolerances and speedup-error compromise (these parameters are further explored in Sections 11.5 and 11.6).

A smaller speedup is provided to the FISTA algorithm when compared to ISTA, which is expected since the former is already faster than the latter[3]. A similar argument applies to the GAP Safe rule, which has stronger screening capabilities than Dynamic Safe: our method still manages to provide some additional acceleration when combined to these already quite efficient techniques, especially in low-regularized scenarios.

For simplicity, only the results concerning multiple dictionary approximations will be reported from this point on.

11.5 Varying speedup-error compromises

It is reasonable to expect the quality of the approximations $\tilde{\mathbf{D}}^i$ to be decisive on the performance of the proposed algorithm. Its success depends on the possibility of providing fast yet precise approximations of the dictionary $\mathbf{D}$. Let us now evaluate the influence of this speedup-error compromise on the simulation time results.

Figure 11.5 shows the same type of normalized running times as in Figure 11.4, but it now compares the results of the proposed algorithm under three different scenarios in terms of speedup-error profiles (see Figure 11.1 for details on the approximation errors in each scenario). The impact on the running times is indeed significant. However, even on the Hard scenario, the proposed approach manages to provide non-negligible acceleration.

11.6 Varying convergence tolerances

As previously highlighted, screening is known to work better on highly-regularized scenarios ($\lambda/\lambda_{\max} \approx 1$). The speedup provided by the screening is also more pronounced when a more precise convergence is required, because the final iterations are often less costly

3. The results reported in the right graphs are already normalized w.r.t the FISTA time results, which are typically smaller than those of ISTA.

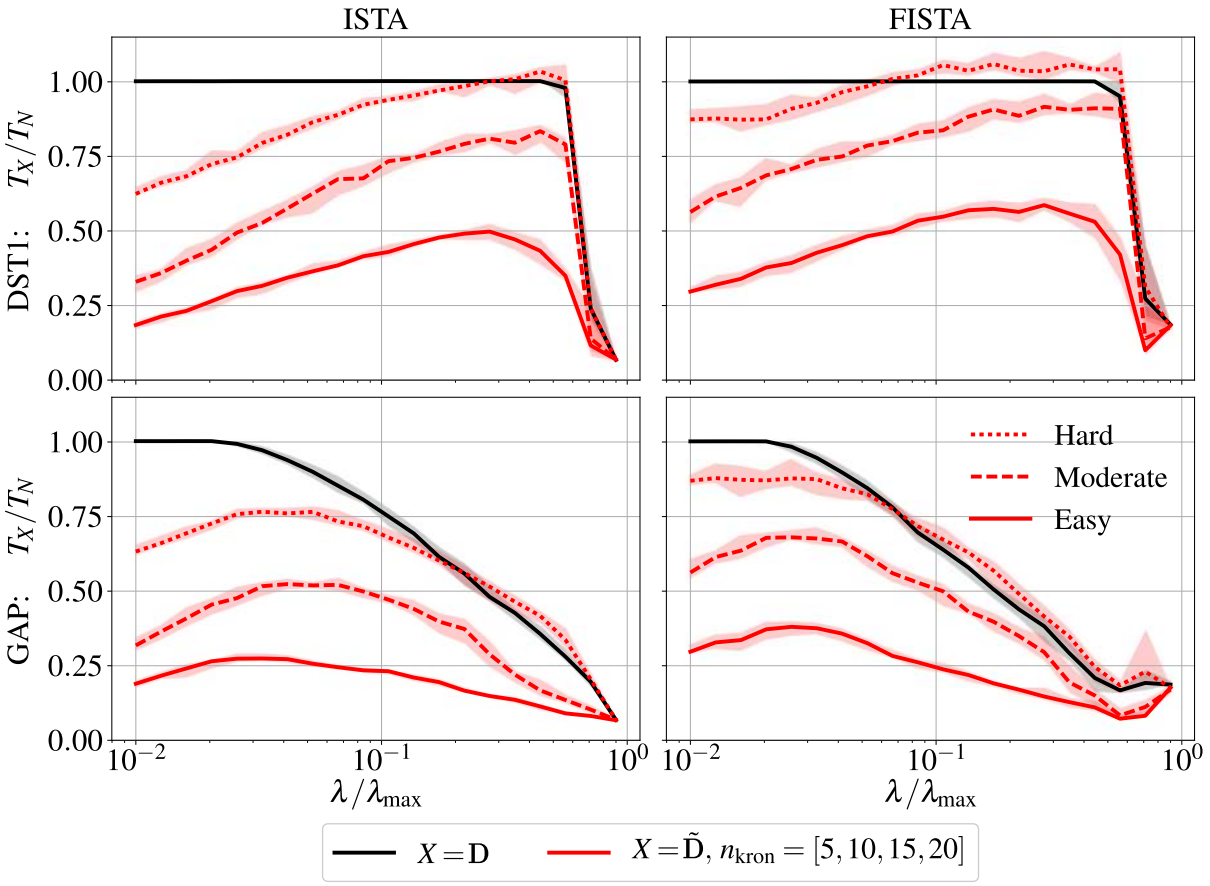
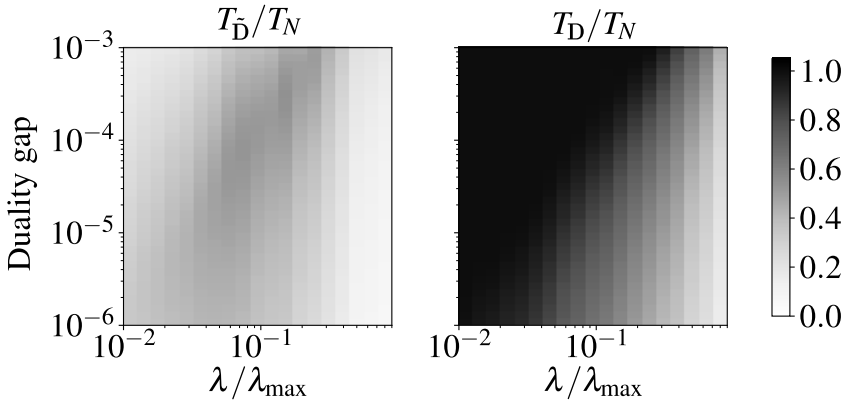
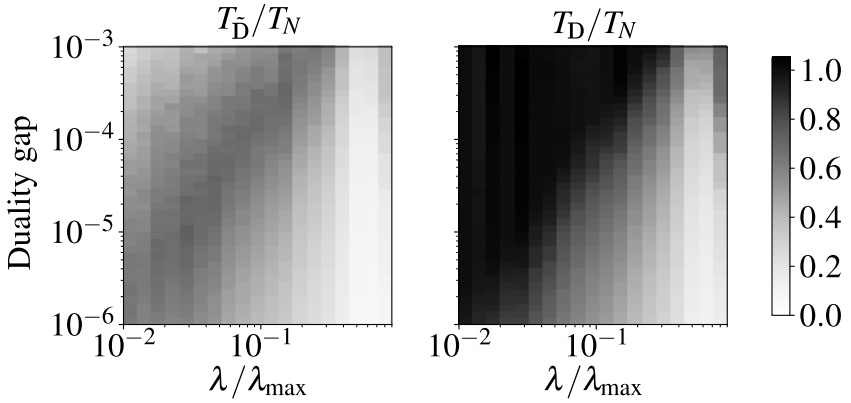


Figure 11.5 – Normalized running times in 3 different scenarios of speedup-error tradeoffs. $tol = 10^{-5}$, $\Gamma = 0.5$.



(a) ISTA



(b) FISTA

Figure 11.6 – Normalized execution times in grayscale (darker means slower) to reach a certain duality gap for a range of regularizations. Left: proposed technique (w/ stable screening). Right: Conventional screening.

as most atoms have been screened out. These two aspects create the triangular profiles observed in Figure 11.6 (right plots). This figure shows the normalized execution times in grayscale (the darker, the slower) as a function of both the regularization (x-axis) and the convergence precision in terms of the duality gap (y-axis).

Note that the proposed method (left plots) efficiently complements the screening tests in its main weaknesses, namely: low regularizations and reduced convergence requirements. In weakly regularized scenarios, while screening tests struggle to eliminate atoms especially in initial iterations, the fast approximate dictionaries come at rescue by making those iterations faster. Besides that, when higher duality gaps are targeted, less iterations using the slow dictionary $\mathbf{D}$ are necessary. Thus, proportionally, the accelerated part of the algorithm (with fast dictionaries) is more significant.

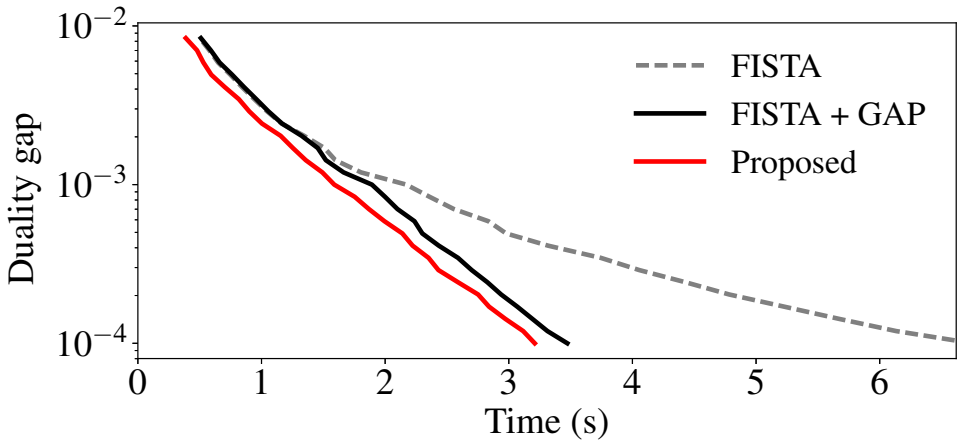


Figure 11.7 – Source localization using an MEG forward operator. Duality gap as a function of time for $\lambda/\lambda_{\max} = 10^{-1}$.

11.7 Behavior on MEG source localization

In this experiment, we consider an MEG (magnetoencephalography) source localization problem, which consists in retrieving a limited set of active brain foci associated to a given MEG measured signal. A common way to achieve such a task is to solve a convex sparse inverse problem [Matsuura & Okabe 1995, Gramfort *et al.* 2012].

We consider an MEG gain matrix (forward operator) $\mathbf{D} \in \mathbb{R}^{204\times 8193}$ obtained with the MNE software [Gramfort *et al.* 2014]. This operator can be efficiently approximated as a product of a few sparse matrices, as proposed in [Magoarou & Gribonval 2016]. The speedups and corresponding approximation errors achieved by the so-called *FAμST* dictionaries adopted here are summarized in Table 11.3. The obtained speedup-error compromise corresponds to a hard scenario.

The experiment consists in picking (uniformly) at random eight active sources with standard gaussian weights giving 8-sparse coefficient vectors $\mathbf{x} \in \mathbb{R}^{8193}$ to be recovered from the input signals $\mathbf{y} = \mathbf{D}\mathbf{x} \in \mathbb{R}^{204}$ by solving problem (7.1).

Mean ϵ_j	$2.1\cdot10^{-1}$	$1.2\cdot10^{-1}$	$6.6\cdot10^{-2}$	$5.9\cdot10^{-2}$
Measured RC	0.22	0.33	0.45	0.63

Table 11.3 – Approximation errors and Relative Complexities for the FAuST approximations of the MEG matrix.

Figure 11.7 shows the required time to reach a certain duality gap precision (averaged over 50 independent runs) for a fixed regularization $\lambda/\lambda_{\max} = 10^{-1}$, around which the

correct number of sources is recovered. The proposed method is consistently advantageous, providing some speedup with respect to both the FISTA algorithm (grey dotted line) and its enhanced version with GAP Safe screening (black line). Although somewhat modest when compared to the latter, the provided gain is proportionally more pronounced at lower precision requirements in the duality gap. More significant speedups would require *better* approximations of MEG gain matrix (in the sense of speedup-error compromise).

It is also important to emphasize that neither of the two considered acceleration techniques affects the algorithms results in terms of source localization accuracy, since they solve the exact same problem at the same convergence precision. That is why task-specific performance measures are not discussed (see [Gramfort *et al.* 2012] for more details on the performance of ℓ_1-minimization techniques for brain source localization).

Conclusion and perspectives

In this thesis, we explored techniques for accelerating sparsity-constrained inverse problems. The two main lines of work can be briefly summarized as follows: 1) Kronecker-structured dictionaries for tensor data; 2) Safe screening techniques robust to approximation error.
This last chapter summarizes the central contributions and presents some possible future research directions.

Conclusion

After reviewing some useful tools and related literature, the contributions of this thesis were presented in parts II and III.

Part II discussed efficient structured dictionaries. To improve on storage, robustness to sample size and computational complexity, a new dictionary model was introduced where the dictionary is constrained as a sum of R Kronecker products of K terms, the so-called HO-SuKro model. In chapter 4, we argued how such a constraint arises naturally when the targeted data are given by Kth-order tensors, such as collections of color or hyperspectral images. We have drawn a parallel between this sum-of-Kroneckers constraint and the tensor Canonical Polyadic Decomposition, the latter being used as a projection algorithm for imposing the structure constraint to a generic finite-dimensional linear operator.

In chapter 5 two novel dictionary learning algorithms were proposed. First, the tools developed in the preceding chapter for matrix approximation were extended to the dictionary learning problem, leading to a projected gradient approach for learning HO-SuKro dictionaries. Later, a considerably more efficient algorithm for learning the same class of dictionaries was derived. It alternatively updates the Kronecker blocks within the proposed structure, implying that the dictionary never leaves the constraint space. We also analyzed the computational complexity of both learning algorithms, discussed how the proposed structure can be exploited to accelerate traditional sparse coding algorithms and made some considerations on the actual speedups that can be attained in practice. A

considerable gap was observed between the theoretical and practical speedups, but some acceleration can still be achieved in practice and several possibilities exist to try and tighten this gap (detailed in the Perspectives section below).

Experimental results in color image and hyperspectral image denoising were reported in chapter 6. Encouraging results were obtained in both applications. The tensor structure constraint on the dictionary proved to be beneficial in terms of denoising performance compared to a completely unstructured dictionary, especially in higher noise scenarios. Besides serving as a regularization to avoid overfitting in such noisy scenarios, the structure constraint was also observed to improve robustness to smaller training datasets. Specifically for the Hyperspectral data, an improved denoising approach was proposed by integrating a low-rank prior on the images, which led to state-of-the-art denoising performance.

Part III was dedicated to safe screening techniques. Aiming to combine safe screening tests and fast structured operators in an effort to accelerate the solution of ℓ_1-minimization problems, we proposed a methodology for defining safe screening tests despite an inaccurate knowledge of the dictionary atoms.

The proposed methodology was exemplified on some existing screening tests (static and dynamic) in chapter 9. The resulting *stable screening tests* were then employed in a fast algorithm in chapter 10, which exploits a series of fast approximations of the dictionary matrix and can be combined to almost any existing first-order ℓ_1-solver. The proposed switching criterion consistently prevents the proposed method from leading to an overhead in terms of convergence time.

Simulation results in chapter 11 demonstrated the two combined strategies to be quite complementary, justifying the effort to conciliate them. The structured dictionaries contribute to accelerate the initial phase of the optimization process, which is precisely the main weakness of the screening techniques especially in weakly regularized scenarios – in such cases, screening may take several iterations to start eliminating atoms.

Perspectives

In this final section, we elaborate on some research possibilities that arise as natural follow-ups to the topics discussed in this thesis.

Envisioned further works range from improvements on the proposed algorithms to the extension of the proposed HO-SuKro model to other problems and applications.

Hyperparameter tuning The proposed methods rely on several hyperparameters: those related to the algorithms' flow (step-sizes and convergence tolerances for the CPD, the ALS loop or the sparse coding, for instance); and those related to the model specification (separable rank R, number of Kronecker blocks K and factor dimensions $\{n_k, m_k\}$). More in-depth studies on the first group of parameters may lead to improvements on the performance and convergence speed of the proposed algorithms. Although some heuristic or empirically-inspired criteria have been proposed to set decent values, as for now, the influence of these parameters is not fully characterized as well as the robustness of the proposed algorithms to their choice.

Some of the model-related parameters may be fixed by the aimed application. For instance, in the patch-based denoising applications explored in chapter 6, the number of Kronecker blocks K corresponds to the number of modes of the tensor input data and the dimensions $\{n_k\}$ correspond to the patch dimensions. However, the choice of the separable rank R remains arbitrary (although restricted to small values for complexity reasons) as well as the choice of $\{m_k\}$ which defines the number of atoms. One could also consider the case where the factor sizes $\{n_k, m_k\}$ are not known. Their estimation thus becomes an interesting problem that comes down to identifying the underlying structure of the data.

Further theoretical studies Some particular points on the thesis still lack a more rigorous investigation, of which the following two can be highlighted:

1. Study the convergence properties of the proposed algorithms to provide further theoretical convergence guarantees. The non-convex nature of the dictionary learning problem considerably complicates this task. However, several works exist on providing dictionary learning algorithms with provable guarantees [Spielman *et al.* 2012, Agarwal *et al.* 2016, Schwab *et al.* 2019] and they could be a starting point.
2. Analyze the effects of the dictionary columns normalization, both in terms of the algorithm's convergence and the quality of the obtained dictionary. This study could lead to more rigorous ways to integrate this unit-norm constraint into the optimization process, as opposed to the current arbitrary choice of handling it as a post-processing step.

Bridging the performance gap Even if the proposed structure led to interesting accelerations w.r.t. to unstructured dictionaries in matrix-vector operations, a considerable gap was observed between the theoretical complexities and the practical speedups. This gap is mainly due to inefficiencies on the computation of tensor operations (more precisely, sequences of mode products) when compared to conventional matrix algebra. Nevertheless, there is active research addressing this issue in different ways, for instance through lower-level optimization in cache reuse and vectorization [Matthews 2018] or by avoiding excessive data reordering [Vannieuwenhoven *et al.* 2013, Li *et al.* 2015].

In a related topic, still aiming at practically efficient algorithms, we envision developing online (or batch-based) versions of the proposed algorithms to achieve real scalability in terms of memory complexities.

Extension to other problems and applications HO-SuKro dictionaries are not restricted to denoising applications and can be straightforwardly applied to other dictionary-based applications such as inpainting, deblurring and image fusion, or even to other types of data like medical, seismic, audio and video. In a broader sense, the proposed structure need not be restricted to the dictionary learning domain. It can, for example, be used for covariance matrix approximation, extending the work in [Tsiligkaridis & Hero 2013]. Another potential application is in the graph signal processing domain, to obtain efficient approximations of the graph Laplacian matrix, as an alternative to the recent approach in [Le Magoarou *et al.* 2018].

Several perspectives exist in the hyperspectral denoising application, to cite some: learning the dictionary on clean data extracted from a corpus of similar images; replacing the SVD by a more meaningful decomposition using NMF or more advanced unmixing approaches; or using the proposed denoising technique as a pre-processing for other downstream applications such as classification and segmentation.

Stable safe screening

The proposed screening framework also leads to several promising perspectives, detailed in the following.

Other sources of approximation error In a broader sense, the proposed stable tests can be seen as a robust screening tool for inaccurately-defined ℓ_1-minimization problems

in which the imprecision need not be due to the use of fast approximations –for instance, due to intrinsic imprecisions on the measurement matrix of a given inverse problem.

Providing better criteria for structured approximations The proposed methodology retrospectively provides a better understanding on which atom approximation error measures are more significant to the stable screening tests, potentially leading to more suited data fidelity criteria in structured dictionary approximation techniques. It could also provide information on how critical each atom is (statistically, for a certain class of input signals) to adjust the approximation precision accordingly.

Variations and extensions In principle, the proposed framework can also be applied to other types of safe regions and error bounds than the ones explored in this thesis, such as dome regions [Xiang *et al.* 2017]. The proposed framework could also be extended to other sparsity-inducing inverse problems such as the Group-Lasso or the regularized logistic regression. Finally, an interesting perspective is to extend these zone-based screening tests to off-the-grid generalizations of ℓ_1-regularized problems [Bredies & Pikkarainen 2013]. In such problems, the ability to screen individual atoms is completely useless, since there is an infinite number of them. The proposed stable screening, which allows to screen an entire *zone*, appears to be the appropriate tool to overcome this technical difficulty.

Connections to neural networks Iterative proximal gradient algorithms for the lasso problem can be reinterpreted in the context of neural network: the gradient step is seen as the linear layer and the proximal step (soft-thresholding operation) as the non-linearity. In the LISTA (Learned ISTA) method [Gregor & LeCun 2010], the classical ISTA algorithm is mapped into a recurrent neural network which is then trained via backpropagation through time. An interesting perspective is to incorporate a screening procedure to this kind of approach to provide further acceleration. Although the linear layer in the network is initialized as the conventional gradient operation, it is modified during the training phase. This provides evidence that ideas similar to the ones behind stable screening could be useful in making this adaptation possible.

Appendices

Appendix A

Complements to Part I

A.1 Derivation of the lasso dual function

Notice that the lasso dual function in (2.33) can be rewritten as two separate minimization problems (with respect to $\mathbf{z}$ and $\mathbf{x}$)

$$D(\nu) = \underbrace{\min_{\mathbf{z}\in\mathbb{R}^n} \left(\tfrac{1}{2}\|\mathbf{y}-\mathbf{z}\|_2^2 + \boldsymbol{\nu}^T\mathbf{z}\right)}_{(II)} + \underbrace{\min_{\mathbf{x}\in\mathbb{R}^m} \lambda\|\mathbf{x}\|_1 - \boldsymbol{\nu}^T\mathbf{D}\mathbf{x}}_{(I)} \tag{A.1}$$

Let us tackle each minimization problem (I) and (II) separately.

Solving problem (I)

Problem (I) can be rewritten as

$$\min_{\mathbf{x}} \lambda \left(\|\mathbf{x}\|_1 - \mathbf{x}^T\frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda}\right) \tag{A.2}$$

At this stage, let us introduce the definition of the dual norm.

Definition 27. *The dual norm $\|\cdot\|_*$ associated to $\|\cdot\|$ is defined as*

$$\|\boldsymbol{v}\|_* = \sup_{\|\boldsymbol{u}\|\leq 1} \boldsymbol{u}^T\boldsymbol{v} \tag{A.3}$$

This is useful since we know the dual norm of the $\|\cdot\|_1$ norm.

Property 14. *The dual norm of the ℓ_1-norm $\|\cdot\|_1$ is the ℓ_∞-norm.*

Let us distinguish two cases.

First case If $\|\frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda}\|_\infty > 1$, then $\exists \boldsymbol{u}\in\mathbb{R}^m$ with $\|\boldsymbol{u}\|_1 \leq 1$ and $\boldsymbol{u}^T\frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda} > 1$.

Proof. by taking $\boldsymbol{v} = \frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda}$ in (A.3) we have $\|\frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda}\|_\infty = \sup_{\|\boldsymbol{u}\|\leq 1} \boldsymbol{u}^T \frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda} > 1$. □

So, we can take $\mathbf{x} = \alpha\boldsymbol{u}$ in (A.2), which yields

$$\|\mathbf{x}\|_1 - \mathbf{x}^T \frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda} = \alpha \underbrace{\left(\|\boldsymbol{u}\|_1 - \boldsymbol{u}^T \frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda}\right)}_{<0}. \tag{A.4}$$

Since α can be made as large as desired, and the second term is negative, we can make (I) as small as desired. So, (I) has solution $-\infty$ in this case.

Second case If $\|\frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda}\|_\infty \leq 1$, then $\mathbf{x}^T \frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda} \leq \|\mathbf{x}\|_1$.

Proof. $\mathbf{x}^T \frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda} = \|\mathbf{x}\|_1 \frac{\mathbf{x}^T}{\|\mathbf{x}\|_1} \frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda} \leq \|\mathbf{x}\|_1 \sup_{\|\boldsymbol{u}\|\leq 1} \boldsymbol{u}^T \frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda} = \|\mathbf{x}\|_1 \underbrace{\|\frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda}\|_\infty}_{\leq 1} \leq \|\mathbf{x}\|_1$. □

Therefore $\left(\|\mathbf{x}\|_1 - \mathbf{x}^T \frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda}\right) \geq 0$ and, since $\mathbf{x} = \mathbf{0}_m$ reaches the lower bound, we have that $(I) = \min_{\mathbf{x}} \lambda \left(\|\mathbf{x}\|_1 - \mathbf{x}^T \frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda}\right) = 0$, in this case.

Partial conclusion We deduce that

$$D(\boldsymbol{\nu}) = \begin{cases} (II) = \min_{\mathbf{z}\in\mathbb{R}^n} \underbrace{\frac{1}{2}\|\mathbf{y}-\mathbf{z}\|_2^2 + \boldsymbol{\nu}^T\mathbf{z}}_{\triangleq \phi(\mathbf{z})} & \text{if } \|\frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda}\|_\infty \leq 1 \\ -\infty & \text{otherwise.} \end{cases} \tag{A.5}$$

Solving problem (II)

Let us denote ϕ the function associated with problem (II). As a convex smooth function, it can be minimized by setting the gradient to zero:

$$\nabla_{\mathbf{z}}\phi(\mathbf{z}) = -\mathbf{y} + \mathbf{z} + \boldsymbol{\nu} = 0. \tag{A.6}$$

Therefore, replacing $\mathbf{z}$ by $\mathbf{y} - \boldsymbol{\nu}$ in ϕ, we obtain

$$\min_{\mathbf{z}} \ \phi(\mathbf{z}) = -\tfrac{1}{2}\|\boldsymbol{\nu}\|_2^2 + \boldsymbol{\nu}^T\mathbf{y} = \tfrac{1}{2}\left(\|\mathbf{y}\|_2^2 - \|\mathbf{y}-\boldsymbol{\nu}\|_2^2\right). \tag{A.7}$$

Finally, the closed form solution is obtained for the Dual function:

$$D(\boldsymbol{\nu}) = \begin{cases} \frac{1}{2}\left(\|\mathbf{y}\|_2^2 - \|\mathbf{y} - \boldsymbol{\nu}\|_2^2\right) \text{ if } \|\frac{\mathbf{D}^T\boldsymbol{\nu}}{\lambda}\|_\infty \leq 1 \\ -\infty \end{cases} \tag{A.8}$$

Appendix B

Complements to Part II

B.1 Tucker Dictionary Learning

Here we discuss how the Dictionary Learning problem with a (R, K)-Kronecker-structure constraint relates to the Tucker decomposition. More precisely, we show that, in the particular case of $R = 1$, the dictionary update step is equivalent to a Tucker decomposition with a sparse core given by $\mathbf{X}$.

A $(1, K)$-KS dictionary is given by

$$\mathbf{D} = \mathbf{D}_K \boxtimes \cdots \boxtimes \mathbf{D}_1,$$

and the dictionary update cost function considering the desired structure becomes:

$$f(\mathbf{D}_j) = \|\mathbf{Y} - (\mathbf{D}_K \boxtimes \cdots \boxtimes \mathbf{D}_1)\mathbf{X}\|_F^2$$

Now, this type of structure becomes interesting when the targeted data is multidimensional. So, let us consider that each data sample is a tensor $\boldsymbol{\mathcal{Y}}_i$ such that $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N] = [\mathrm{vec}(\boldsymbol{\mathcal{Y}}_1), \ldots, \mathrm{vec}(\boldsymbol{\mathcal{Y}}_N)]$. Similarly, we suppose a tensor structure for the sparse coefficients $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N] = [\mathrm{vec}(\boldsymbol{\mathcal{X}}_1), \ldots, \mathrm{vec}(\boldsymbol{\mathcal{X}}_N)]$. By stacking samples $\boldsymbol{\mathcal{Y}}_i$ (resp. $\boldsymbol{\mathcal{X}}_i$) in the $(K+1)$-th mode, we obtain the analogue to the matrix $\mathbf{Y}$ (resp. $\mathbf{X}$) without vectorization pre-processing step: the data tensor $\boldsymbol{\mathcal{Y}}$ (resp. $\boldsymbol{\mathcal{X}}$). The dictionary update can be now written as follows:

$$f(\mathbf{D}_j) = \sum_{i=1}^{N} \| \mathrm{vec}(\boldsymbol{\mathcal{Y}}_i) - (\mathbf{D}_K \boxtimes \cdots \boxtimes \mathbf{D}_1)\, \mathrm{vec}(\boldsymbol{\mathcal{X}}_i)\|_2^2$$

By applying the well-known Kronecker product identity (3.1), we obtain:

$$f(\mathbf{D}_j) = \sum_{i=1}^{N} \|\boldsymbol{\mathcal{Y}}_i - \boldsymbol{\mathcal{X}}_i \times_1 \mathbf{D}_1 \cdots \times_K \mathbf{D}_K\|_F^2$$

A simple trick allows us to absorb the summation: adding an identity operator acting on mode $K+1$, along which the different samples are stacked.

$$f(\mathbf{D}_j) = \|\boldsymbol{\mathcal{Y}} - \boldsymbol{\mathcal{X}} \times_1 \mathbf{D}_1 \cdots \times_K \mathbf{D}_K \times_{K+1} \mathbf{I}_N\|_F^2 \tag{B.1}$$

In (B.1) we are trying to approximate a tensor $\boldsymbol{\mathcal{Y}}$ as a series of mode products on a core tensor $\boldsymbol{\mathcal{X}}$. This is precisely the Tucker decomposition (defined in equation (3.8)), with one of the factors fixed (an identity matrix in the mode $K+1$).

B.2 Other computational complexities

Table B.1 shows the computational complexity for an alternative calculation ordering in the ALS algorithm 11.

1. For $\mathbf{Y}_{(j)}\mathbf{U}_q^T = \mathbf{Y}_{(j)}\left(\boxtimes_{l\neq j}\mathbf{D}_{l,q}\right)\mathbf{X}_{(j)}^T$: calculating first $\mathbf{M}_q = \mathbf{Y}_{(j)}\left(\boxtimes_{l\neq j}\mathbf{D}_{l,q}\right)$ as a structured product, then $\mathbf{M}_q\mathbf{X}_{(j)}^T$ (instead of structured product $\mathbf{U}_q = \mathbf{X}_{(j)}\left(\boxtimes_{l\neq j}\mathbf{D}_{l,q}\right)^T$, then $\mathbf{Y}_{(j)}\mathbf{U}_q^T$). This is more costly, since $m_j \geq n_j \ \forall j$.

2. For $\mathbf{U}\mathbf{U}^T$: calculate $\mathbf{G}_{q,s} = \left(\boxtimes_{l\neq j}\mathbf{D}_{l,q}^T\mathbf{D}_{l,s}\right) \ \forall q \leq r, \ s \leq r$ (since $\mathbf{G}_{q,s} = \mathbf{G}_{s,q}^T$, only $(r+1)r/2$ such products are necessary) concatenate into matrix $\mathbf{G}$, followed by the structured product $\mathbf{X}_{(j)}\mathbf{G}$ (or $\mathbf{G}\mathbf{X}_{(j)}^T$) and a dense product with $\mathbf{X}_{(j)}^T$ (or $\mathbf{X}_{(j)}$).

Table B.1 – Dictionary update complexities in ALS Alg. 11 (alternative calculation order)

Operation	**Complexity**	**Considering sparsity**
$\mathbf{M}_q = \mathbf{Y}_{(j)}\left(\boxtimes_{l\neq j}\mathbf{D}_{l,q}\right), \ \forall q \leq r$	$r(\sum_{l\neq j} m_l)(\prod_l m_l)N$	–
$\mathbf{M}_q\mathbf{X}_{(j)}^T$	$rn_j(\prod_l m_l)N$	$rn_j \operatorname{nnz}(\mathbf{X})$
$\mathbf{G}_{q,s} = \left(\boxtimes_{l\neq j}\mathbf{D}_{l,q}^T\mathbf{D}_{l,s}\right) \ \forall q \leq r, \ s \leq r$	$r^2(\prod_{l\neq j} n_l m_l^2)$	–
$\mathbf{X}_{(j)}\mathbf{G}\mathbf{X}_{(j)}^T$	$r^2(\sum_l m_l)(\prod_l m_l)N$	$r^2(\sum_l m_l)\operatorname{nnz}(\mathbf{X})$
$(\mathbf{X}_{(j)}\mathbf{G}\mathbf{X}_{(j)}^T)^{-1}$	$(rm_l)^3$	–

Overall, the resulting complexities are more or less of the same order of the ones presented in Table 5.2. If on the one hand some n_j are replaced by the bigger dimension m_j here, on the other hand the sparsity of $\mathbf{X}$ is better exploited.[1] In theory, there is no clear hierarchy between the two implementations, the comparison depending on the parameter's relative values. In practice, however, the implementation considered in Table 5.2 was faster in the considered scenarios. One of the reasons is that, in practice, sparse data, similarly to the tensor operations, does not fully achieve the theoretical speedups in practice.

B.3 Color image denoising

Results for images Oakland, Tiffany and Tree are shown in Tables B.2.

1. In this alternative calculation order, the Kronecker terms are never multiplied directly to $\mathbf{X}$. This type of operation suboptimally exploits the sparsity of $\mathbf{X}$ because the sparsity is lost after first mode product and does not intervene in the ensuing mode products.

Table B.2 – Output PSNR varying the patch size – Tiffany

	Algorithm	Patch size		Input SNR			
		n	m	22.11	14.15	10.63	8.13
Tiffany	ODCT	[6, 6, 3]	[12,12, 6]	31.43	26.80	23.50	20.82
		[12,12, 3]	[24,24, 6]	32.01	28.21	26.38	24.48
	K-SVD	[6, 6, 3]	[12,12, 6]	31.96	26.45	23.14	20.85
		[12,12, 6]	[24,24, 6]	32.15	27.92	25.84	24.01
	HO-SuKro ALS (R=3)	[6, 6, 3]	[12,12, 6]	31.98	26.48	23.51	21.26
		[12,12, 3]	[24,24, 6]	**32.49**	**28.68**	**26.59**	**24.56**
Tree	ODCT	[6, 6, 3]	[12,12, 6]	28.89	24.02	21.72	20.29
		[12,12, 3]	[24,24, 6]	28.97	24.70	22.66	21.14
	K-SVD	[6, 6, 3]	[12,12, 6]	**29.68**	24.74	22.20	20.45
		[12,12, 6]	[24,24, 6]	29.44	24.90	22.70	20.95
	HO-SuKro ALS (R=3)	[6, 6, 3]	[12,12, 6]	29.36	24.58	22.32	20.69
		[12,12, 3]	[24,24, 6]	29.35	**25.41**	**23.45**	**21.98**

Appendix C

Complements to Part III

C.1 Proof of Equation (8.6)

Screening test for a ℓ_p-ball, i.e. $\mathcal{R} = B_p(\mathbf{c}, R)$:

$$\begin{aligned}\sup_{\boldsymbol{\theta}\in B_p(\mathbf{c},R)} |\mathbf{d}^T\boldsymbol{\theta}| &= \sup_{\mathbf{u}\in B_p(\mathbf{0},1)} |\mathbf{d}^T(\mathbf{c}+R\mathbf{u})| \\ &= |\mathbf{d}^T\mathbf{c}| + R \sup_{\mathbf{u}\in B_p(\mathbf{0},1)} |\mathbf{d}^T\mathbf{u}| \\ &= |\mathbf{d}^T\mathbf{c}| + R\|\mathbf{d}\|_{p^*} \end{aligned} \tag{C.1}$$

using the definition of dual norm $\|\mathbf{d}\|_{p^*} := \sup_{\mathbf{u}\in B_p(\mathbf{0},1)} |\mathbf{d}^T\mathbf{u}|$.

C.2 Proof of Equations (9.3) and (9.5)

From Definition 26 and the conventional sphere test in eq. (8.6), the stable sphere test is given by

$$\begin{aligned}\mu(\mathcal{S}|B_p(\mathbf{c},R)) &= \sup_{\mathbf{d}\in\mathcal{S}} \left(|\mathbf{d}^T\mathbf{c}| + R\|\mathbf{d}\|_{p^*}\right) \\ &= \sup_{\mathbf{d}\in\mathcal{S}} |\mathbf{d}^T\mathbf{c}| + R \sup_{\mathbf{d}\in\mathcal{S}} \|\mathbf{d}\|_{p^*}. \end{aligned} \tag{C.2}$$

The second equality comes from the fact that $\mathbf{c}$ is arbitrary here. Therefore, we can always assume $\mathbf{c}$ to be aligned with the $\mathbf{d}$ that maximizes the second term, which makes the first term to be also maximized for the same $\mathbf{d}$.

Taking $\mathcal{S} = B_q(\tilde{\mathbf{d}}, \epsilon)$, the first term gives (similarly to (C.1))

$$\sup_{\mathbf{d}\in\mathcal{S}} |\mathbf{d}^T\mathbf{c}| = \sup_{\mathbf{u}\in B_q(\mathbf{0},1)} |(\tilde{\mathbf{d}}+\epsilon\mathbf{u})^T\mathbf{c}| = |\tilde{\mathbf{d}}^T\mathbf{c}| + \epsilon\|\mathbf{c}\|_{q^*} \tag{C.3}$$

and the last term gives

$$\sup_{\mathbf{d}\in B_q(\tilde{\mathbf{d}},\epsilon)} \|\mathbf{d}\|_{p^*} = \sup_{\mathbf{u}\in B_q(\mathbf{0},1)} \|(\tilde{\mathbf{d}}+\epsilon\mathbf{u})\|_{p^*}$$
$$= \|\tilde{\mathbf{d}}\|_{p^*} + C\epsilon \tag{C.4}$$

with a constant $C = \sup_{\mathbf{u}\in B_q(\mathbf{0},1)} \|\mathbf{u}\|_{p^*} = n^{\left(\frac{1}{p^*}-\frac{1}{q}\right)_+}$, which results from the Holder's inequality. Substituting (C.4) and (C.3) in (C.2) gives the bound in equation (9.3).

Now, supposing $\|\mathbf{d}\|_{p^*} = a$ to be known, the last term in (C.2) simplifies to Ra and the same derivation in (C.3) applies to the first term, which gives equation (9.5).

$$\mu(\mathcal{S}'|B_p(\mathbf{c},R)) = \sup_{\mathbf{d}\in\mathcal{S}'} \Big(|\mathbf{d}^T\mathbf{c}| + Ra\Big) = |\tilde{\mathbf{d}}^T\mathbf{c}| + \epsilon\|\mathbf{c}\|_{q^*} + Ra \tag{C.5}$$

C.3 Proof of Lemma 6

By definition we have $\boldsymbol{\theta}_F := \Theta'(\mathbf{z}|\tilde{\mathbf{D}},\boldsymbol{\epsilon}) = \alpha\mathbf{z}$ where

$$0 \le \alpha \le \frac{1}{\max_j \left(|\tilde{\mathbf{d}}_j^T\tilde{\mathbf{z}}| + \epsilon_j\|\mathbf{z}\|_{q^*}\right)}$$

As a result for any $1 \le j \le m$ we have

$$|\mathbf{d}_j^T\boldsymbol{\theta}_F| = |(\tilde{\mathbf{d}}_j + \mathbf{e}_j)^T\alpha\mathbf{z}| \le |\alpha|\left(|\tilde{\mathbf{d}}_j^T\mathbf{z}| + \epsilon_j\|\mathbf{z}\|_{q^*}\right) \le 1.$$

This implies that the dual point $\boldsymbol{\theta}_F$ is feasible, i.e. $\boldsymbol{\theta}_F \in \Delta_{\mathbf{D}}$. The fact that $\boldsymbol{\theta}_F \in \Delta_{\tilde{\mathbf{D}}}$ follows similarly (since $\epsilon_j\|\mathbf{z}\|_{q^*} > 0$).

C.4 Proof of Theorem 10

[Fercoq *et al.* 2015, Theorem 2] implies, for any $(\mathbf{x},\boldsymbol{\theta}) \in \mathbb{R}^m \times \Delta_{\mathbf{D}}$, that $B\left(\boldsymbol{\theta}, \frac{1}{\lambda}\sqrt{2G(\mathbf{x},\boldsymbol{\theta}|\mathbf{D})}\right)$ is safe. Therefore, a sufficient condition for $B\left(\boldsymbol{\theta}, \frac{1}{\lambda}\sqrt{2G(\mathbf{x},\boldsymbol{\theta}|\tilde{\mathbf{D}}) + 2\delta}\right)$ to be safe is to show that:

$$G(\mathbf{x},\boldsymbol{\theta}|\tilde{\mathbf{D}}) + \delta \ge G(\mathbf{x},\boldsymbol{\theta}|\mathbf{D}) \tag{C.6}$$

Substituting (7.4) in the primal objective (7.1) and denoting $\tilde{\boldsymbol{\rho}} = \mathbf{y} - \tilde{\mathbf{D}}\mathbf{x}$ and $\mathbf{E} = \mathbf{D} - \tilde{\mathbf{D}}$ yields

$$\begin{aligned} P(\mathbf{x}|\mathbf{D}) &= \tfrac{1}{2}\|(\tilde{\mathbf{D}} + \mathbf{E})\mathbf{x} - \mathbf{y}\|_2^2 + \lambda\|\mathbf{x}\|_1 \\ &= P(\mathbf{x}|\tilde{\mathbf{D}}) + \tfrac{1}{2}\|\mathbf{E}\mathbf{x}\|_2^2 - \tilde{\boldsymbol{\rho}}^T(\mathbf{E}\mathbf{x}), \end{aligned}$$

which implies that

$$G(\mathbf{x}, \boldsymbol{\theta}|\mathbf{D}) = G(\mathbf{x}, \boldsymbol{\theta}|\tilde{\mathbf{D}}) \underbrace{+\frac{1}{2}\|\mathbf{E}\mathbf{x}\|_2^2 - \tilde{\boldsymbol{\rho}}^T(\mathbf{E}\mathbf{x})}_{\delta'}. \tag{C.7}$$

In a way δ' is a security margin which, when added to the $G(\mathbf{x}, \boldsymbol{\theta}|\tilde{\mathbf{D}})$, makes it equal to $G(\mathbf{x}, \boldsymbol{\theta}|\mathbf{D})$ and, thus, safe.

In practice, however, the calculation of δ', if even possible, is too computationally demanding since it requires to recalculate the matrix-vector product $\mathbf{E}\mathbf{x}$ at every iteration. To avoid it, we adopt the following margin instead:

$$\delta = \|\tilde{\boldsymbol{\rho}}\|_2 \mathcal{E}\|\mathbf{x}\|_r + \frac{1}{2}\mathcal{E}^2\|\mathbf{x}\|_r^2 \geq \delta' \tag{C.8}$$

for any $\mathcal{E} \geq \|\mathbf{E}\|_{r\to 2}$ and using the fact that $\|\mathbf{E}\mathbf{x}\|_2 \leq \|\mathbf{E}\|_{r\to 2}\|\mathbf{x}\|_r \leq \mathcal{E}\|\mathbf{x}\|_r$ (by the definition of $\mathcal{E}$) and $\mathcal{E}$ can be precalculated once for all.

Since $\delta \geq \delta'$, then $G(\mathbf{x}, \boldsymbol{\theta}|\tilde{\mathbf{D}}) + \delta \geq G(\mathbf{x}, \boldsymbol{\theta}|\mathbf{D})$.

C.5 Detailed complexities

We suppose that the p^*-norms of the atoms $\|\mathbf{d}_j\|_{p^*}$ and q-norms of the atom approximation error $\|\mathbf{e}_j\|_q$ are precalculated and stored and memory. The total screening complexity is summarized in Table 11.2 and detailed in the following.

Feasible point $\boldsymbol{\theta}_t$

A total of at most $3N + |\mathcal{A}_t|$ operations for the point in eq. (8.9), distributed as follows:

n dot product $\mathbf{y}^T\boldsymbol{\rho}_t$.

n norm $\|\boldsymbol{\rho}_t\|_2^2$ (if primal objective not calculated as a convergence criterion).

Screening	$\boldsymbol{\theta}_t$	$\mathcal{R}$	Test	**Total**
Dynamic Safe	$3n + \lvert\mathcal{A}\rvert$	n	$\lvert\mathcal{A}\rvert$	$4n + 2\lvert\mathcal{A}\rvert$
Stable Dynamic Safe	$4n + 2\lvert\mathcal{A}\rvert$	n	$\lvert\mathcal{A}\rvert$	$5n + 3\lvert\mathcal{A}\rvert$
GAP Safe	$3n + \lvert\mathcal{A}\rvert$	n	$\lvert\mathcal{A}\rvert$	$4n + 2\lvert\mathcal{A}\rvert$
Stable GAP Safe	$4n + 2\lvert\mathcal{A}\rvert$	n	$n + 2\lvert\mathcal{A}\rvert$	$6n + 4\lvert\mathcal{A}\rvert$

Table C.1 – Screening: number of floating-point operations.

n product $\alpha\boldsymbol{\rho}_t$.

$|\mathcal{A}_t|$ comparisons for the infinity norm $\|\mathbf{D}^T\boldsymbol{\rho}\|_\infty$.

At most $n+|\mathcal{A}_t|$ extra operations for the point in eq. (9.7):

n norm $\|\tilde{\boldsymbol{\rho}}_t\|_{q^*}$ (only if $q^* \neq 2$).

$|\mathcal{A}_t|$ products $\epsilon_j\|\tilde{\boldsymbol{\rho}}_t\|_{q^*}$ on the calculation of α'_t.

Safe region $\mathcal{R}$

At most n operations for computing $\|\boldsymbol{\theta}_t - \mathbf{y}/\lambda\|_2$ (if the dual objective $D(\boldsymbol{\theta}_t)$ is not already calculated as a convergence criterion) for the radius of either Dynamic Safe or GAP Safe sphere.

Screening test

Other operations (aside from the mentioned $\mathbf{D}^T\mathbf{c}$) sum up to $|\mathcal{A}_t|$ operations for the product $R\|\mathbf{d}_j\|_{p^*}\ \forall\ j \in \mathcal{A}_t$. The proposed stable screening tests require at most an extra $n + |\mathcal{A}_t|$ operations:

n norm $\|\mathbf{c}\|_{q^*}$ (if $q^* \neq 2$ and $\mathbf{c}$ varies with t).

$|\mathcal{A}_t|$ products $\epsilon_j\|\mathbf{c}\|_{q^*}\ \forall\ j \in \mathcal{A}_t$ (if $\mathbf{c}$ varies with t).

Bibliography

[Adler *et al.* 2012] Amir Adler, Valentin Emiya, Maria G. Jafari, Michael Elad, Rémi Gribonval and Mark D. Plumbley. *Audio Inpainting*. IEEE Transactions on Audio, Speech and Language Processing, vol. 20, no. 3, pages 922–932, Mar 2012. Cited on page 161.

[Agarwal *et al.* 2016] A. Agarwal, A. Anandkumar, P. Jain and P. Netrapalli. *Learning Sparsely Used Overcomplete Dictionaries via Alternating Minimization*. SIAM Journal on Optimization, vol. 26, no. 4, pages 2775–2799, 2016. Cited on page 209.

[Aharon & Elad 2008] Michal Aharon and Michael Elad. *Sparse and Redundant Modeling of Image Content Using an Image-Signature-Dictionary*. SIAM Journal on Imaging Sciences, vol. 1, no. 3, pages 228–247, 2008. Cited on page 39.

[Aharon *et al.* 2006] M. Aharon, M. Elad and A. Bruckstein. *K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation*. IEEE Transactions on Signal Processing, vol. 54, no. 11, pages 4311–4322, Nov 2006. Cited on pages 38, 132, and 146.

[Baker 2016] Alan Baker. *Simplicity*. In Edward N. Zalta, editor, The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University, winter 2016 édition, 2016. Cited on page 17.

[Barthelemy *et al.* 2012] Q. Barthelemy, A. Larue, A. Mayoue, D. Mercier and J. I. Mars. *Shift 2D Rotation Invariant Sparse Coding for Multivariate Signals*. IEEE Transactions on Signal Processing, vol. 60, no. 4, pages 1597–1611, April 2012. Cited on page 40.

[Basuhail & Kozaitis 1998] Abdullah A Basuhail and Samuel Peter Kozaitis. *Wavelet-based noise reduction in multispectral imagery*. In Algorithms for multispectral and hyperspectral imagery IV, volume 3372, pages 234–241. International Society for Optics and Photonics, 1998. Cited on pages 140, 143, and 153.

[Batselier & Wong 2017] Kim Batselier and Ngai Wong. *A constructive arbitrary-degree Kronecker product decomposition of tensors*. Numerical Linear Algebra with Applications, vol. 24, no. 5, page e2097, 2017. Cited on page 99.

[Beck & Teboulle 2009] Amir Beck and Marc Teboulle. *A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems*. SIAM Journal on Imaging Sciences, vol. 2, no. 1, pages 183–202, Jan 2009. Cited on pages 19, 29, 52, 55, 56, 113, 161, and 162.

[Beck 2017] A. Beck. First-order methods in optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2017. Cited on pages 52, 53, and 54.

[Bijma *et al.* 2005] Fetsje Bijma, Jan De Munck and Rob M Heethaar. *The spatiotemporal MEG covariance matrix modeled as a sum of Kronecker products*. vol. 27, pages 402–15, 09 2005. Cited on page 99.

[Bioucas-Dias & Figueiredo 2007] J. M. Bioucas-Dias and M. A. T. Figueiredo. *A New TwIST: Two-Step Iterative Shrinkage/Thresholding Algorithms for Image Restoration*. IEEE Transactions on Image Processing, vol. 16, no. 12, pages 2992–3004, Dec 2007. Cited on pages 19, 56, 161, and 162.

[Bioucas-Dias & Nascimento 2008] J. M. Bioucas-Dias and J. M. P. Nascimento. *Hyperspectral Subspace Identification*. IEEE Transactions on Geoscience and Remote Sensing, vol. 46, no. 8, pages 2435–2445, Aug 2008. Cited on page 146.

[Bishop 2006] Christopher M. Bishop. Pattern recognition and machine learning (information science and statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. Cited on page 36.

[Bonnefoy *et al.* 2015] Antoine Bonnefoy, Valentin Emiya, Liva Ralaivola and Rémi Gribonval. *Dynamic screening: Accelerating first-order algorithms for the lasso and group-lasso*. IEEE Transactions on Signal Processing, vol. 63, no. 19, pages 5121–5132, Oct 2015. Cited on pages 9, 14, 22, 56, 164, 170, 171, 172, and 193.

[Boyd & Vandenberghe 2004] Stephen Boyd and Lieven Vandenberghe. Convex optimization. Cambridge University Press, New York, NY, USA, 2004. Cited on page 62.

[Bredies & Pikkarainen 2013] Kristian Bredies and Hanna Katriina Pikkarainen. *Inverse problems in spaces of measures.* ESAIM: Control, Optimisation and Calculus of Variations, vol. 19, no. 1, pages 190–218, Mar 2013. Cited on pages 10 and 211.

[Bristow *et al.* 2013] H. Bristow, A. Eriksson and S. Lucey. *Fast Convolutional Sparse Coding.* In 2013 IEEE Conference on Computer Vision and Pattern Recognition, pages 391–398, June 2013. Cited on page 40.

[Bro 1998] R. Bro. *Multi-way analysis in the food industry: models, algorithms, and applications.* PhD thesis, Københavns UniversitetKøbenhavns Universitet, Det Biovidenskabelige Fakultet for Fødevarer, VeteFaculty of Life Sciences, Institut for FødevarevidenskabDepartment of Food Science, Kvalitet og TeknologiQuality & Technology, 1998. Cited on page 74.

[Bruckstein *et al.* 2009] Alfred M. Bruckstein, David L. Donoho and Michael Elad. *From Sparse Solutions of Systems of Equations to Sparse Modeling of Signals and Images.* SIAM Rev., vol. 51, no. 1, pages 34–81, Feb 2009. Cited on pages 32, 33, and 148.

[Buades *et al.* 2005] A. Buades, B. Coll and J. . Morel. *A non-local algorithm for image denoising.* In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 2, pages 60–65 vol. 2, June 2005. Cited on page 126.

[Caiafa & Cichocki 2013] Cesar F. Caiafa and Andrzej Cichocki. *Multidimensional compressed sensing and their applications.* Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 3, no. 6, pages 355–380, 2013. Cited on pages 41, 99, and 116.

[Candes & Tao 2006] E. J. Candes and T. Tao. *Near-Optimal Signal Recovery From Random Projections: Universal Encoding Strategies?* IEEE Transactions on Information Theory, vol. 52, no. 12, pages 5406–5425, Dec 2006. Cited on page 50.

[Carroll & Chang 1970] J. Douglas Carroll and Jih-Jie Chang. *Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition.* Psychometrika, vol. 35, no. 3, pages 283–319, Sep 1970. Cited on pages 87 and 89.

[Carroll *et al.* 1980] J.D. Carroll, S. Pruzansky and J.B. Kruskal. *Candelinc: A general approach to multidimensional analysis of many-way arrays with linear constraints on parameters.* Psychometrika, vol. 45, no. 1, pages 3–24, Mar 1980. Cited on page 92.

[Castrodad *et al.* 2011] A. Castrodad, Z. Xing, J. B. Greer, E. Bosch, L. Carin and G. Sapiro. *Learning Discriminative Sparse Representations for Modeling, Source Separation, and Mapping of Hyperspectral Imagery.* IEEE Transactions on Geoscience and Remote Sensing, vol. 49, no. 11, pages 4263–4281, Nov 2011. Cited on page 146.

[Chabiron *et al.* 2015] Olivier Chabiron, Francois Malgouyres, Jean-Yves Tourneret and Nicolas Dobigeon. *Toward fast transform learning.* International Journal of Computer Vision, vol. 114, no. 2, pages 195–216, Sep 2015. Cited on page 40.

[Chambolle & Dossal 2015] Antonin Chambolle and Charles Dossal. *On the convergence of the iterates of "FISTA".* Journal of Optimization Theory and Applications, vol. Volume 166, no. Issue 3, page 25, #Aug# 2015. Cited on page 53.

[Chambolle & Pock 2011] Antonin Chambolle and Thomas Pock. *A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging.* Journal of Mathematical Imaging and Vision, vol. 40, no. 1, pages 120–145, May 2011. Cited on pages 19, 56, 161, and 162.

[Chen & Qian 2011] G. Chen and S. Qian. *Denoising of Hyperspectral Imagery Using Principal Component Analysis and Wavelet Shrinkage.* IEEE Transactions on Geoscience and Remote Sensing, vol. 49, no. 3, pages 973–980, Mar 2011. Cited on page 148.

[Chen *et al.* 1998] Scott Shaobing Chen, David L. Donoho and Michael A. Saunders. *Atomic Decomposition by Basis Pursuit.* SIAM Journal on Scientific Computing, vol. 20, no. 1, pages 33–61, Jan 1998. Cited on pages 28, 29, 35, 56, 57, 161, and 162.

[Chen *et al.* 2011] Y. Chen, N. M. Nasrabadi and T. D. Tran. *Hyperspectral Image Classification Using Dictionary-Based Sparse Representation.* IEEE Transactions on Geoscience and Remote Sensing, vol. 49, no. 10, pages 3973–3985, Oct 2011. Cited on page 146.

[Combettes & Wajs 2005] P. Combettes and V. Wajs. *Signal Recovery by Proximal Forward-Backward Splitting*. Multiscale Modeling & Simulation, vol. 4, no. 4, pages 1168–1200, 2005. Cited on page 52.

[Comon *et al.* 2009] P. Comon, X. Luciani and A. L. F. de Almeida. *Tensor decompositions, alternating least squares and other tales*. Journal of Chemometrics, vol. 23, no. 7-8, pages 393–405, 2009. Cited on page 89.

[Comon 2014] P. Comon. *Tensors : A brief introduction*. IEEE Signal Processing Magazine, vol. 31, no. 3, pages 44–53, May 2014. Cited on page 91.

[Dai & Milenkovic 2009] W. Dai and O. Milenkovic. *Subspace Pursuit for Compressive Sensing Signal Reconstruction*. IEEE Transactions on Information Theory, vol. 55, no. 5, pages 2230–2249, May 2009. Cited on page 51.

[Dantas & Gribonval 2017] Cassio F. Dantas and Rémi Gribonval. *Dynamic Screening with Approximate Dictionaries*. In XXVIème colloque GRETSI, Juan-les-Pins, France, Sep 2017. Cited on pages 23, 161, and 200.

[Dantas & Gribonval 2018] Cassio F. Dantas and Rémi Gribonval. *Faster and still safe: combining screening techniques and structured dictionaries to accelerate the Lasso*. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4069–4073, Calgary, AB, Canada, Apr 2018. Cited on pages 23, 161, 182, and 200.

[Dantas & Gribonval 2019a] C. F. Dantas and R. Gribonval. *Stable Safe Screening and Structured Dictionaries for Faster ℓ_1 Regularization*. IEEE Transactions on Signal Processing, vol. 67, no. 14, pages 3756–3769, July 2019. Cited on pages 23, 161, 162, and 201.

[Dantas & Gribonval 2019b] Cassio F. Dantas and Rémi Gribonval. *Stable Screening - Python code*. (hal-02129219), available at `https://github.com/cassiofragadantas/Screening_ADST`, May 2019. Cited on page 195.

[Dantas *et al.* 2017] C. F. Dantas, M. N. da Costa and R. R. Lopes. *Learning Dictionaries as a Sum of Kronecker Products*. IEEE Signal Processing Letters, vol. 24, no. 5, pages 559–563, May 2017. Cited on pages 40, 99, 147, and 149.

[Dantas *et al.* 2018] Cassio F. Dantas, Jérémy E Cohen and Rémi Gribonval. *Learning fast dictionaries for sparse representations using low-rank tensor decompositions.* In International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA), pages 456–466, Guildford, United Kingdom, Jul 2018. Cited on pages 23, 95, 109, and 125.

[Dantas *et al.* 2019a] Cassio F. Dantas, Jérémy E. Cohen and Rémi Gribonval. *Hyperspectral Image Denoising using Dictionary Learning.* In 10th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), pages 1–5, Amsterdam, The Netherlands, Sep 2019. Cited on pages 23 and 125.

[Dantas *et al.* 2019b] Cassio F. Dantas, Jérémy E. Cohen and Rémi Gribonval. *Learning Tensor-structured Dictionaries with Application to Hyperspectral Image Denoising.* In 27th European Signal Processing Conference (EUSIPCO), pages 1–5, A Coruña, Spain, Sep 2019. Cited on pages 23, 109, and 125.

[Daubechies *et al.* 2004] I. Daubechies, M. Defrise and C. De Mol. *An iterative thresholding algorithm for linear inverse problems with a sparsity constraint.* Communications on Pure and Applied Mathematics, vol. 57, no. 11, pages 1413–1457, 2004. Cited on pages 19, 54, 161, and 162.

[De Lathauwer *et al.* 2000] L. De Lathauwer, B. De Moor and J. Vandewalle. *A Multilinear Singular Value Decomposition.* SIAM Journal on Matrix Analysis and Applications, vol. 21, no. 4, pages 1253–1278, 2000. Cited on page 91.

[de Silva & Lim 2008] V. de Silva and L. Lim. *Tensor Rank and the Ill-Posedness of the Best Low-Rank Approximation Problem.* SIAM Journal on Matrix Analysis and Applications, vol. 30, no. 3, pages 1084–1127, 2008. Cited on page 104.

[Do & Vetterli 2005] M. N. Do and M. Vetterli. *The contourlet transform: an efficient directional multiresolution image representation.* IEEE Transactions on Image Processing, vol. 14, no. 12, pages 2091–2106, Dec 2005. Cited on page 126.

[Domanov & De Lathauwer 2013a] I. Domanov and L. De Lathauwer. *On the Uniqueness of the Canonical Polyadic Decomposition of Third-Order Tensors—Part I: Basic Results and Uniqueness of One Factor Matrix.* SIAM Journal on Matrix Analysis and Applications, vol. 34, no. 3, pages 855–875, 2013. Cited on page 88.

[Domanov & De Lathauwer 2013b] I. Domanov and L. De Lathauwer. *On the Uniqueness of the Canonical Polyadic Decomposition of Third-Order Tensors—Part II: Uniqueness of the Overall Decomposition*. SIAM Journal on Matrix Analysis and Applications, vol. 34, no. 3, pages 876–903, 2013. Cited on page 89.

[Donoho & Elad 2003] David L. Donoho and Michael Elad. *Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization*. Proceedings of the National Academy of Sciences, vol. 100, no. 5, pages 2197–2202, 2003. Cited on pages 31, 32, and 33.

[Donoho *et al.* 2012] D. L. Donoho, Y. Tsaig, I. Drori and J. Starck. *Sparse Solution of Underdetermined Systems of Linear Equations by Stagewise Orthogonal Matching Pursuit*. IEEE Transactions on Information Theory, vol. 58, no. 2, pages 1094–1121, Feb 2012. Cited on pages 19 and 51.

[Donoho 1999] David L. Donoho. *Wedgelets: nearly minimax estimation of edges*. Ann. Statist., vol. 27, no. 3, pages 859–897, 06 1999. Cited on page 126.

[Dossal & Mallat 2005] Charles Dossal and Stéphane Mallat. *Sparse spike deconvolution with minimum scale*. In Signal Processing with Adaptive Sparse Structured Representations (SPARS workshop), pages 123–126, Rennes, France, Nov 2005. Cited on page 161.

[Dumitrescu & Irofti 2018] Bogdan Dumitrescu and Paul Irofti. Structured dictionaries, pages 167–208. Springer International Publishing, Cham, 2018. Cited on page 39.

[Eckart & Young 1936] C. Eckart and G. Young. *The approximation of one matrix by another of lower rank*. Psychometrika, vol. 1, no. 3, pages 211–218, Sep 1936. Cited on pages 39, 89, and 103.

[Efron *et al.* 2004] Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani. *Least angle regression*. Annals of Statistics, vol. 32, pages 407–499, 2004. Cited on page 58.

[El Ghaoui *et al.* 2010] Laurent El Ghaoui, Vivian Viallon and Tarek Rabbani. *Safe feature elimination in sparse supervised learning*. EECS Department, University of California, Berkeley, Tech. Rep, Sep 2010. Cited on pages 9, 14, 20, 22, 164, 170, and 171.

[Elad & Aharon 2006] M. Elad and M. Aharon. *Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries*. IEEE Transactions on Image Processing, vol. 15, no. 12, pages 3736–3745, Dec 2006. Cited on pages 126, 127, 128, 131, 134, 146, 148, and 149.

[Elad 2010] Michael Elad. *Prologue*. In Sparse and Redundant Representations, pages 3–15. Springer, 2010. Cited on pages 29, 30, and 125.

[Engan *et al.* 1999] K. Engan, S. O. Aase and J. Hakon Husoy. *Method of optimal directions for frame design*. In Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on, volume 5, pages 2443–2446 vol.5, 1999. Cited on page 38.

[Eslami & Radha 2006] R. Eslami and H. Radha. *Translation-Invariant Contourlet Transform and Its Application to Image Denoising*. IEEE Transactions on Image Processing, vol. 15, no. 11, pages 3362–3374, Nov 2006. Cited on page 126.

[Fan & Lv 2008] Jianqing Fan and Jinchi Lv. *Sure independence screening for ultrahigh dimensional feature space*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 70, no. 5, pages 849–911, Nov 2008. Cited on pages 164 and 165.

[Fercoq & Richtárik 2015] O. Fercoq and P. Richtárik. *Accelerated, Parallel, and Proximal Coordinate Descent*. SIAM Journal on Optimization, vol. 25, no. 4, pages 1997–2023, 2015. Cited on page 59.

[Fercoq *et al.* 2015] O. Fercoq, A. Gramfort and J. Salmon. *Mind the duality gap: safer rules for the Lasso*. In International Conference on Machine Learning, volume 37, pages 333–342, July 2015. Cited on pages 9, 14, 22, 162, 164, 165, 172, and 222.

[Foucart & Rauhut 2013] Simon Foucart and Holger Rauhut. A mathematical introduction to compressive sensing. 2013. Cited on pages 18 and 162.

[Friedman *et al.* 2007] Jerome Friedman, Trevor Hastie, Holger Höfling and Robert Tibshirani. *Pathwise coordinate optimization*. Ann. Appl. Stat., vol. 1, no. 2, pages 302–332, 12 2007. Cited on page 59.

[Fu *et al.* 2015] Y. Fu, A. Lam, I. Sato and Y. Sato. *Adaptive Spatial-Spectral Dictionary Learning for Hyperspectral Image Denoising.* In 2015 IEEE International Conference on Computer Vision (ICCV), pages 343–351, Dec 2015. Cited on page 126.

[Fu 1998] Wenjiang J. Fu. *Penalized Regressions: The Bridge versus the Lasso.* Journal of Computational and Graphical Statistics, vol. 7, no. 3, pages 397–416, 1998. Cited on page 59.

[Garcia-Cardona & Wohlberg 2018] C. Garcia-Cardona and B. Wohlberg. *Convolutional Dictionary Learning: A Comparative Review and New Algorithms.* IEEE Transactions on Computational Imaging, vol. 4, no. 3, pages 366–381, Sep. 2018. Cited on page 40.

[Ghamisi *et al.* 2017] P. Ghamisi, N. Yokoya, J. Li, W. Liao, S. Liu, J. Plaza, B. Rasti and A. Plaza. *Advances in Hyperspectral Image and Signal Processing: A Comprehensive Overview of the State of the Art.* IEEE Geoscience and Remote Sensing Magazine, vol. 5, no. 4, pages 37–78, Dec 2017. Cited on page 139.

[Ghassemi *et al.* 2017] M. Ghassemi, Z. Shakeri, A. D. Sarwate and W. U. Bajwa. *STARK: Structured Dictionary Learning Through Rank-one Tensor Recovery.* In IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), December 2017. Cited on pages 41 and 99.

[Golbabaee & Vandergheynst 2012] M. Golbabaee and P. Vandergheynst. *Hyperspectral image compressed sensing via low-rank and joint-sparse matrix recovery.* In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2741–2744, March 2012. Cited on page 147.

[Gorodnitsky & Rao 1997] I. F. Gorodnitsky and B. D. Rao. *Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm.* IEEE Transactions on Signal Processing, vol. 45, no. 3, pages 600–616, March 1997. Cited on page 31.

[Gramfort *et al.* 2012] Alexandre Gramfort, Matthieu Kowalski and Matti Hämäläinen. *Mixed-norm estimates for the M/EEG inverse problem using accelerated gradient methods.* Physics in Medicine and Biology, vol. 57, no. 7, pages 1937–1961, Mar 2012. Cited on pages 204 and 205.

[Gramfort *et al.* 2014] Alexandre Gramfort, Martin Luessi, Eric Larson, Denis A Engemann, Daniel Strohmeier, Christian Brodbeck, Lauri Parkkonen and Matti S Hämäläinen. *MNE software for processing MEG and EEG data.* Neuroimage, vol. 86, pages 446–460, Feb 2014. Cited on page 204.

[Gregor & LeCun 2010] Karol Gregor and Yann LeCun. *Learning Fast Approximations of Sparse Coding.* In Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10, pages 399–406, USA, 2010. Omnipress. Cited on pages 166 and 211.

[Gribonval & Nielsen 2003] R. Gribonval and M. Nielsen. *Sparse representations in unions of bases.* IEEE Transactions on Information Theory, vol. 49, no. 12, pages 3320–3325, Dec 2003. Cited on page 33.

[Gribonval & Nielsen 2007] R. Gribonval and M. Nielsen. *Highly sparse representations from dictionaries are unique and independent of the sparseness measure.* Applied and Computational Harmonic Analysis, vol. 22, no. 3, pages 335 – 355, 2007. Cited on page 29.

[Gribonval *et al.* 2015a] R. Gribonval, R. Jenatton and F. Bach. *Sparse and Spurious: Dictionary Learning With Noise and Outliers.* IEEE Transactions on Information Theory, vol. 61, no. 11, pages 6298–6319, Nov 2015. Cited on page 99.

[Gribonval *et al.* 2015b] R. Gribonval, R. Jenatton, F. Bach, M. Kleinsteuber and M. Seibert. *Sample Complexity of Dictionary Learning and Other Matrix Factorizations.* IEEE Transactions on Information Theory, vol. 61, no. 6, pages 3469–3486, June 2015. Cited on page 39.

[Gui & Li 2005] Jiang Gui and Hongzhe Li. *Penalized Cox Regression Analysis in the High-dimensional and Low-sample Size Settings, with Applications to Microarray Gene Expression Data.* Bioinformatics, vol. 21, no. 13, pages 3001–3008, #jul# 2005. Cited on page 161.

[Hackbusch 2012] W. Hackbusch. Tensor spaces and numerical tensor calculus. Springer–Verlag, Berlin, 2012. Cited on page 74.

[Harshman & Lundy 1996] Richard A. Harshman and Margaret E. Lundy. *Uniqueness proof for a family of models sharing features of Tucker's three-mode factor analysis*

and PARAFAC/candecomp. Psychometrika, vol. 61, no. 1, pages 133–154, Mar 1996. Cited on page 92.

[Harshman 1970] Richard Harshman. *Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis*. UCLA Working Papers in Phonetics, vol. 16, 1970. Cited on pages 87 and 89.

[Harshman 1972] R. A. Harshman. *PARAFAC2: Mathematical and technical notes*. UCLA Working Papers in Phonetics, vol. 22, pages 30–44, 1972. Cited on page 92.

[Harshman 1978] Richard A Harshman. *Models for analysis of asymmetrical relationships among N objects or stimuli*. In First Joint Meeting of the Psychometric Society and the Society of Mathematical Psychology, Hamilton, Ontario, 1978, 1978. Cited on page 92.

[Hawe *et al.* 2013] Simon Hawe, Matthias Seibert and Martin Kleinsteuber. *Separable dictionary learning*. In IEEE Conference on Computer Vision and Pattern Recognition, pages 438–445, June 2013. Cited on pages 40 and 99.

[Henderson & Searle 1981] Harold V. Henderson and S. R. Searle. *The vec-permutation matrix, the vec operator and Kronecker products: a review*. Linear and Multilinear Algebra, vol. 9, no. 4, pages 271–288, 1981. Cited on page 81.

[Herzet & Drémeau 2018] Cedric Herzet and Angélique Drémeau. *Joint Screening Tests for LASSO*. In IEEE International Conference on Acoustic, Speech and Signal Processing, pages 4084–4088, Calgary, AB, Canada, Apr 2018. Cited on page 166.

[Herzet *et al.* 2019] C. Herzet, C. Dorffer and A. Drémeau. *Gather and Conquer: Region-Based Strategies to Accelerate Safe Screening Tests*. IEEE Transactions on Signal Processing, vol. 67, no. 12, pages 3300–3315, June 2019. Cited on pages 166 and 179.

[Hitchcock 1927] Frank L. Hitchcock. *The Expression of a Tensor or a Polyadic as a Sum of Products*. Journal of Mathematics and Physics, vol. 6, no. 1-4, pages 164–189, 1927. Cited on page 87.

[Horn & Johnson 2012] Roger A. Horn and Charles R. Johnson. Matrix analysis. Cambridge University Press, New York, NY, USA, 2nd édition, 2012. Cited on page 81.

[Jain *et al.* 2011] Prateek Jain, Ambuj Tewari and Inderjit S. Dhillon. *Orthogonal Matching Pursuit with Replacement*. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 24, pages 1215–1223. Curran Associates, Inc., 2011. Cited on page 51.

[Johnson & Guestrin 2015] Tyler Johnson and Carlos Guestrin. *Blitz: A principled meta-algorithm for scaling sparse optimization*. In International Conference on Machine Learning, pages 1171–1179, 2015. Cited on pages 58 and 165.

[Jost *et al.* 2006] P. Jost, P. Vandergheynst, S. Lesage and R. Gribonval. *MoTIF: An Efficient Algorithm for Learning Translation Invariant Dictionaries*. In 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, volume 5, pages V–V, May 2006. Cited on page 40.

[Jung *et al.* 2016] A. Jung, Y. C. Eldar and N. Görtz. *On the Minimax Risk of Dictionary Learning*. IEEE Transactions on Information Theory, vol. 62, no. 3, pages 1501–1515, March 2016. Cited on page 98.

[Kervrann & Boulanger 2006] C. Kervrann and J. Boulanger. *Optimal Spatial Adaptation for Patch-Based Image Denoising*. IEEE Transactions on Image Processing, vol. 15, no. 10, pages 2866–2878, Oct 2006. Cited on page 126.

[Kim & Park 2010] Jingu Kim and Haesun Park. *Fast active-set-type algorithms for l1-regularized linear regression*. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, pages 397–404, Sardinia, Italy, May 2010. Cited on pages 58 and 165.

[Kim *et al.* 2007] Seung-Jean Kim, Kwangmoo Koh, Michael Lustig, Stephen Boyd and Dimitry Gorinevsky. *An interior-point method for large-scale l 1-regularized least squares*. Selected Topics in Signal Processing, IEEE Journal of, vol. 1, no. 4, pages 606–617, 2007. Cited on page 57.

[Kolda & Bader 2009] Tamara G. Kolda and Brett W. Bader. *Tensor Decompositions and Applications*. SIAM REVIEW, vol. 51, no. 3, pages 455–500, 2009. Cited on pages 74, 75, 76, 89, 92, 95, 103, and 116.

[Kolda 2001] Tamara G. Kolda. *Orthogonal Tensor Decompositions*. SIAM Journal on Matrix Analysis and Applications, vol. 23, no. 1, pages 243–255, July 2001. Cited on page 89.

[Kolda 2006] Tamara G. Kolda. *Multilinear Operators for Higher-order Decompositions*. Technical report SAND2006-2081, Sandia National Laboratories, April 2006. Cited on page 76.

[Kowalski *et al.* 2011] Matthieu Kowalski, Pierre Weiss, Alexandre Gramfort and Sandrine Anthoine. *Accelerating ISTA with an active set strategy*. In OPT 2011: 4th International Workshop on Optimization for Machine Learning, page 7, Sierra Nevada, Spain, Dec 2011. Cited on page 58.

[Kruskal 1977] Joseph B. Kruskal. *Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics*. Linear Algebra and its Applications, vol. 18, no. 2, pages 95 – 138, 1977. Cited on pages 31 and 88.

[Labate *et al.* 2005] Demetrio Labate, Wang-Q Lim, Gitta Kutyniok and Guido Weiss. *Sparse multidimensional representation using shearlets*. In Optics & Photonics 2005, pages 59140U–59140U. International Society for Optics and Photonics, 2005. Cited on page 126.

[Lam *et al.* 2012] A. Lam, I. Sato and Y. Sato. *Denoising hyperspectral images using spectral domain statistics*. In Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012), pages 477–480, Nov 2012. Cited on page 126.

[Landsberg 2012] Joseph M. Landsberg. Tensors: geometry and applications, volume 128. American Mathematical Soc., 2012. Cited on page 73.

[Le Magoarou *et al.* 2018] L. Le Magoarou, R. Gribonval and N. Tremblay. *Approximate Fast Graph Fourier Transforms via Multilayer Sparse Approximations*. IEEE Transactions on Signal and Information Processing over Networks, vol. 4, no. 2, pages 407–420, June 2018. Cited on page 210.

[Lesage *et al.* 2005] Sylvain Lesage, Rémi Gribonval, Frédéric Bimbot and Laurent Benaroya. *Learning unions of orthonormal bases with thresholded singular value decomposition.* In Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on, volume 5, pages v–293. IEEE, 2005. Cited on page 39.

[Li *et al.* 2013] Liangyue Li, Sheng Li and Yun Fu. *Discriminative dictionary learning with low-rank regularization for face recognition.* In Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on, pages 1–6, April 2013. Cited on page 40.

[Li *et al.* 2015] J. Li, C. Battaglino, I. Perros, J. Sun and R. Vuduc. *An input-adaptive and in-place approach to dense tensor-times-matrix multiply.* In SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pages 1–12, Nov 2015. Cited on pages 123 and 210.

[Liu *et al.* 2001] Xiangqian Liu, Nicholas D. Sidiropoulos and Senior Member. *Cramér-Rao Lower Bounds for Low-Rank Decomposition of Multidimensional Arrays.* IEEE Trans. on Signal Processing, vol. 49, pages 2074–2086, 2001. Cited on page 88.

[Liu *et al.* 2014] Jun Liu, Zheng Zhao, Jie Wang and Jieping Ye. *Safe Screening with Variational Inequalities and Its Application to Lasso.* In Proceedings of the 31st International Conference on Machine Learning, volume 32(2) of *ICML'14*, pages 289–297. JMLR.org, June 2014. Cited on page 164.

[Loth 2011] Manuel Loth. *Active Set Algorithms for the LASSO.* Theses, Université des Sciences et Technologie de Lille - Lille I, #Jul# 2011. Cited on page 58.

[M. Massias & Salmon 2017] A. Gramfort M. Massias and J. Salmon. *From safe screening rules to working sets for faster Lasso-type solvers.* In NIPS Workshop on Optimization for Machine Learning, Long Beach, USA, Dec 2017. Cited on page 165.

[Ma *et al.* 2012] L. Ma, C. Wang, B. Xiao and W. Zhou. *Sparse representation for face recognition based on discriminative low-rank dictionary learning.* In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 2586–2593, June 2012. Cited on page 40.

[Magoarou & Gribonval 2015] L. Le Magoarou and R. Gribonval. *Chasing butterflies: In search of efficient dictionaries.* In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3287–3291, April 2015. Cited on page 40.

[Magoarou & Gribonval 2016] L. Le Magoarou and R. Gribonval. *Flexible Multilayer Sparse Approximations of Matrices and Applications.* IEEE Journal of Selected Topics in Signal Processing, vol. 10, no. 4, pages 688–700, June 2016. Cited on pages 189 and 204.

[Mahmoudi & Sapiro 2005] M. Mahmoudi and G. Sapiro. *Fast image and video denoising via nonlocal means of similar neighborhoods.* IEEE Signal Processing Letters, vol. 12, no. 12, pages 839–842, Dec 2005. Cited on page 126.

[Mailhé *et al.* 2008] Boris Mailhé, Sylvain Lesage, Rémi Gribonval, Frédéric Bimbot and Pierre Vandergheynst. *Shift-invariant dictionary learning for sparse representations: extending K-SVD.* In Signal Processing Conference, 2008 16th European, pages 1–5. IEEE, 2008. Cited on page 40.

[Mairal *et al.* 2008] J. Mairal, M. Elad and G. Sapiro. *Sparse Representation for Color Image Restoration.* IEEE Transactions on Image Processing, vol. 17, no. 1, pages 53–69, Jan 2008. Cited on pages 125 and 126.

[Mairal *et al.* 2009] Julien Mairal, Francis Bach, Jean Ponce and Guillermo Sapiro. *Online dictionary learning for sparse coding.* In Proceedings of the 26th Annual International Conference on Machine Learning, pages 689–696. ACM, 2009. Cited on page 121.

[Malioutov *et al.* 2005] D. Malioutov, M. Cetin and A. S. Willsky. *A sparse signal reconstruction perspective for source localization with sensor arrays.* IEEE Transactions on Signal Processing, vol. 53, no. 8, pages 3010–3022, Aug 2005. Cited on page 161.

[Mallat & Zhang 1993] Stéphane G Mallat and Zhifeng Zhang. *Matching pursuits with time-frequency dictionaries.* IEEE Transactions on Signal Processing, vol. 41, no. 12, pages 3397–3415, Dec 1993. Cited on pages 19, 32, 35, and 161.

[Mallat 2008] Stephane Mallat. A wavelet tour of signal processing, third edition: The sparse way. Academic Press, 3rd édition, 2008. Cited on pages 99 and 126.

[Malti & Herzet 2016] A. Malti and C. Herzet. *Safe screening tests for LASSO based on firmly non-expansiveness*. In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, Mar 2016. Cited on page 164.

[Manolakis *et al.* 2001] D. Manolakis, C. Siracusa and G. Shaw. *Hyperspectral subpixel target detection using the linear mixing model*. IEEE Transactions on Geoscience and Remote Sensing, vol. 39, no. 7, pages 1392–1409, Jul 2001. Cited on pages 146 and 147.

[Marčenko & Pastur 1967] V. A. Marčenko and L. A. Pastur. *DISTRIBUTION OF EIGENVALUES FOR SOME SETS OF RANDOM MATRICES*. Mathematics of the USSR-Sbornik, vol. 1, no. 4, pages 457–483, Apr 1967. Cited on page 152.

[Massias *et al.* 2018] Mathurin Massias, Joseph Salmon and Alexandre Gramfort. *Celer: a fast solver for the lasso with dual extrapolation*. In International Conference on Machine Learning, pages 3321–3330, Stockholm, Sweden, Jul 2018. Cited on page 165.

[Matsuura & Okabe 1995] K. Matsuura and Y. Okabe. *Selective minimum-norm solution of the biomagnetic inverse problem*. IEEE Transactions on Biomedical Engineering, vol. 42, no. 6, pages 608–615, June 1995. Cited on page 204.

[Matthews 2018] Devin A. Matthews. *High-Performance Tensor Contraction without Transposition*. SIAM J. Scientific Computing, vol. 40, no. 1, 2018. Cited on pages 123 and 210.

[Moreau 1962] Jean Jacques Moreau. *Fonctions convexes duales et points proximaux dans un espace hilbertien*. Comptes rendus hebdomadaires des séances de l'Académie des sciences, vol. 255, pages 2897–2899, 1962. Cited on page 52.

[Natarajan 1995] B. Natarajan. *Sparse Approximate Solutions to Linear Systems*. SIAM Journal on Computing, vol. 24, no. 2, pages 227–234, 1995. Cited on page 28.

[Ndiaye *et al.* 2017] Eugene Ndiaye, Olivier Fercoq, Alexandre Gramfort and Joseph Salmon. *Gap safe screening rules for sparsity enforcing penalties*. Journal of Machine Learning Research, vol. 18, no. 128, pages 1–33, Nov 2017. Cited on page 174.

[Needell & Tropp 2009] Deanna Needell and Joel A Tropp. *CoSaMP: Iterative signal recovery from incomplete and inaccurate samples.* Applied and Computational Harmonic Analysis, vol. 26, no. 3, pages 301–321, 2009. Cited on pages 19 and 51.

[Needell & Vershynin 2009] Deanna Needell and Roman Vershynin. *Uniform Uncertainty Principle and Signal Recovery via Regularized Orthogonal Matching Pursuit.* Foundations of Computational Mathematics, vol. 9, no. 3, pages 317–334, Jun 2009. Cited on page 51.

[Needell & Vershynin 2010] D. Needell and R. Vershynin. *Signal Recovery From Incomplete and Inaccurate Measurements Via Regularized Orthogonal Matching Pursuit.* IEEE Journal of Selected Topics in Signal Processing, vol. 4, no. 2, pages 310–316, April 2010. Cited on page 51.

[Nemirovsky & Yudin 1983] A.S. Nemirovsky and D.B. Yudin. Problem complexity and method efficiency in optimization. A Wiley-Interscience publication. Wiley, 1983. Cited on page 53.

[Nesterov 1983] Yu. E. Nesterov. *A method for solving the convex programming problem with convergence rate* $O(1/k^2)$. Proceedings of the USSR Academy of Sciences, vol. 269, pages 543–547, 1983. Cited on page 53.

[Nutini *et al.* 2015] Julie Nutini, Mark Schmidt, Issam Laradji, Michael Friedlander and Hoyt Koepke. *Coordinate Descent Converges Faster with the Gauss-Southwell Rule Than Random Selection.* In Francis Bach and David Blei, editors, Proceedings of the 32nd International Conference on Machine Learning, volume 37 of *Proceedings of Machine Learning Research*, pages 1632–1641, Lille, France, 07–09 Jul 2015. PMLR. Cited on page 59.

[Osborne *et al.* 2000] MR Osborne, B Presnell and BA Turlach. *A new approach to variable selection in least squares problems.* IMA Journal of Numerical Analysis, vol. 20, no. 3, pages 389–403, 07 2000. Cited on page 57.

[Papyan *et al.* 2017a] Vardan Papyan, Yaniv Romano and Michael Elad. *Convolutional Neural Networks Analyzed via Convolutional Sparse Coding.* Journal of Machine Learning Research, vol. 18, no. 83, pages 1–52, 2017. Cited on page 40.

[Papyan *et al.* 2017b] Vardan Papyan, Yaniv Romano, Michael Elad and Jeremias Sulam. *Convolutional Dictionary Learning via Local Processing.* In IEEE International Conference on Computer Vision (ICCV), pages 5306–5314, Oct 2017. Cited on page 40.

[Papyan *et al.* 2018] V. Papyan, Y. Romano, J. Sulam and M. Elad. *Theoretical Foundations of Deep Learning via Sparse Representations: A Multilayer Sparse Model and Its Connection to Convolutional Neural Networks.* IEEE Signal Processing Magazine, vol. 35, no. 4, pages 72–89, July 2018. Cited on page 40.

[Pati *et al.* 1993] Y. C. Pati, R. Rezaiifar and P. S. Krishnaprasad. *Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition.* In Proceedings of 27th Asilomar Conference on Signals, Systems and Computers, pages 40–44. IEEE, 1993. Cited on pages 19, 113, and 161.

[Peng *et al.* 2014] Y. Peng, D. Meng, Z. Xu, C. Gao, Y. Yang and B. Zhang. *Decomposable Nonlocal Tensor Dictionary Learning for Multispectral Image Denoising.* In IEEE Conference on Computer Vision and Pattern Recognition, pages 2949–2956, June 2014. Cited on pages 41 and 99.

[Pennec & Mallat 2005] E. Le Pennec and S. Mallat. *Sparse geometric image representations with bandelets.* IEEE Transactions on Image Processing, vol. 14, no. 4, pages 423–438, April 2005. Cited on page 126.

[Pope *et al.* 2013] G. Pope, C. Aubel and C. Studer. *Learning phase-invariant dictionaries.* In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pages 5979–5983, May 2013. Cited on page 40.

[Portilla *et al.* 2003] J. Portilla, V. Strela, M. J. Wainwright and E. P. Simoncelli. *Image denoising using scale mixtures of Gaussians in the wavelet domain.* IEEE Transactions on Image Processing, vol. 12, no. 11, pages 1338–1351, Nov 2003. Cited on page 126.

[Rasti *et al.* 2014] B. Rasti, J. R. Sveinsson, M. O. Ulfarsson and J. A. Benediktsson. *Hyperspectral Image Denoising Using First Order Spectral Roughness Penalty in Wavelet Domain.* IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 7, no. 6, pages 2458–2467, June 2014. Cited on page 143.

[Rasti *et al.* 2017] B. Rasti, M. O. Ulfarsson and P. Ghamisi. *Automatic Hyperspectral Image Restoration Using Sparse and Low-Rank Modeling.* IEEE Geoscience and Remote Sensing Letters, vol. 14, no. 12, pages 2335–2339, Dec 2017. Cited on pages 143, 146, 147, 148, 149, and 154.

[Rasti *et al.* 2018] Behnood Rasti, Paul Scheunders, Pedram Ghamisi, Giorgio Licciardi and Jocelyn Chanussot. *Noise Reduction in Hyperspectral Imagery: Overview and Application.* Remote Sensing, vol. 10, no. 3, 2018. Cited on pages 126, 139, 140, and 146.

[Rockafellar 1970] R. Tyrrell Rockafellar. Convex analysis. Princeton Mathematical Series. Princeton University Press, Princeton, N. J., 1970. Cited on page 51.

[Roemer *et al.* 2014] Florian Roemer, Giovanni Del Galdo and Martin Haardt. *Tensor-based algorithms for learning multidimensional separable dictionaries.* In Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, pages 3963–3967. IEEE, 2014. Cited on pages 40 and 99.

[Rubinstein *et al.* 2008] Ron Rubinstein, Michael Zibulevsky and Michael Elad. *Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit.* CS Technion, vol. 40, no. 8, pages 1–15, 2008. Cited on pages 47 and 140.

[Rubinstein *et al.* 2010a] Ron Rubinstein, Alfred M Bruckstein and Michael Elad. *Dictionaries for sparse representation modeling.* Proceedings of the IEEE, vol. 98, no. 6, pages 1045–1057, 2010. Cited on page 37.

[Rubinstein *et al.* 2010b] Ron Rubinstein, Michael Zibulevsky and Michael Elad. *Double sparsity: Learning sparse dictionaries for sparse signal approximation.* IEEE Transactions on Signal Processing, vol. 58, no. 3, pages 1553–1564, Mar 2010. Cited on pages 40 and 131.

[Rusu *et al.* 2014] C. Rusu, B. Dumitrescu and S. A. Tsaftaris. *Explicit Shift-Invariant Dictionary Learning.* IEEE Signal Processing Letters, vol. 21, no. 1, pages 6–9, Jan 2014. Cited on page 40.

[Sardy *et al.* 2000] Sylvain Sardy, Andrew G. Bruce and Paul Tseng. *Block Coordinate Relaxation Methods for Nonparametric Wavelet Denoising.* Journal of Computational and Graphical Statistics, vol. 9, no. 2, pages 361–379, 2000. Cited on page 39.

[Schwab *et al.* 2019] Evan Schwab, Benjamin D. Haeffele, René Vidal and Nicolas Charon. *Global Optimality in Separable Dictionary Learning with Applications to the Analysis of Diffusion MRI*, 2019. Cited on page 209.

[Schwartz 1975] L. Schwartz. Les tenseurs. Actualités scientifiques et industrielles. Editions Hermann, 1975. Cited on pages 74, 83, 84, and 85.

[Shakeri *et al.* 2016] Z. Shakeri, W. U. Bajwa and A. D. Sarwate. *Minimax Lower Bounds for Kronecker-Structured Dictionary Learning.* In Proceedings of the 2016 IEEE International Symposium on Information Theory, pages 1148–1152, Barcelona, Spain, Jul 2016. Cited on page 98.

[Shakeri *et al.* 2017a] Z. Shakeri, W. U. Bajwa and A. D. Sarwate. *Sample complexity bounds for dictionary learning of tensor data.* In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4501–4505, March 2017. Cited on pages 98 and 134.

[Shakeri *et al.* 2017b] Z. Shakeri, A. D. Sarwate and W. U. Bajwa. *Identifiability of Kronecker-structured Dictionaries for Tensor Data.* ArXiv e-prints, #dec# 2017. Cited on page 98.

[Shakeri *et al.* 2017c] Zahra Shakeri, Anand D Sarwate and Waheed U Bajwa. *Identification of Kronecker-structured Dictionaries: An Asymptotic Analysis.* In Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2017 IEEE 7th International Workshop on, December 2017. Cited on page 99.

[Shakeri *et al.* 2018] Z. Shakeri, W. U. Bajwa and A. D. Sarwate. *Minimax Lower Bounds on Dictionary Learning for Tensor Data.* IEEE Transactions on Information Theory, vol. 64, no. 4, pages 2706–2726, April 2018. Cited on page 98.

[Shakeri *et al.* 2019] Z. Shakeri, A. D. Sarwate and W. U. Bajwa. *Sample complexity bounds for dictionary learning from vector- and tensor-valued data.* In M. Rodrigues and Y. Eldar, editors, Information Theoretic Methods in Data Science, chapter 5. Cambridge University Press, Cambridge, UK, 2019. Cited on pages 39 and 99.

[Shalev-Shwartz & Zhang 2016] Shai Shalev-Shwartz and Tong Zhang. *Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization.* Mathe-

matical Programming, vol. 155, no. 1, pages 105–145, Jan 2016. Cited on page 59.

[Sidiropoulos & Bro 2000] Nicholas D. Sidiropoulos and Rasmus Bro. *On the uniqueness of multilinear decomposition of N-way arrays*. Journal of Chemometrics, vol. 14, no. 3, pages 229–239, 2000. Cited on page 88.

[Simoncelli *et al.* 1992] E. P. Simoncelli, W. T. Freeman, E. H. Adelson and D. J. Heeger. *Shiftable multiscale transforms*. IEEE Transactions on Information Theory, vol. 38, no. 2, pages 587–607, March 1992. Cited on page 126.

[Southwell 1940] R.V. Southwell. Relaxation methods in engineering science: A treatise on approximate computation. Oxford engineering science series. Clarendon Press, 1940. Cited on page 59.

[Spielman *et al.* 2012] Daniel A. Spielman, Huan Wang and John Wright. *Exact Recovery of Sparsely-Used Dictionaries*. In Shie Mannor, Nathan Srebro and Robert C. Williamson, editors, Proceedings of the 25th Annual Conference on Learning Theory, volume 23 of *Proceedings of Machine Learning Research*, pages 37.1–37.18, Edinburgh, Scotland, 25–27 Jun 2012. PMLR. Cited on page 209.

[Starck *et al.* 2002] Jean-Luc Starck, Emmanuel J Candès and David L Donoho. *The curvelet transform for image denoising*. IEEE Transactions on image processing, vol. 11, no. 6, pages 670–684, 2002. Cited on page 126.

[Sulam *et al.* 2016] J. Sulam, B. Ophir, M. Zibulevsky and M. Elad. *Trainlets: Dictionary Learning in High Dimensions*. IEEE Transactions on Signal Processing, vol. 64, no. 12, pages 3180–3193, June 2016. Cited on pages 40 and 121.

[Tadonki & Philippe 2001] Claude Tadonki and Bernard Philippe. *Parallel Numerical Linear Algebra*. chapter Parallel Multiplication of a Vector by a Kronecker Product of Matrices, pages 71–89. Nova Science Publishers, Inc., Commack, NY, USA, 2001. Cited on page 98.

[ten Berge & Sidiropoulos 2002] Jos M. F. ten Berge and Nikolaos D. Sidiropoulos. *On uniqueness in candecomp/parafac*. Psychometrika, vol. 67, no. 3, pages 399–409, Sep 2002. Cited on page 88.

[Thiagarajan *et al.* 2008] J. J. Thiagarajan, K. N. Ramamurthy and A. Spanias. *Shift-invariant sparse representation of images using learned dictionaries.* In 2008 IEEE Workshop on Machine Learning for Signal Processing, pages 145–150, Oct 2008. Cited on page 40.

[Tibshirani *et al.* 2011] Robert Tibshirani, Jacob Bien, Jerome Friedman, Trevor Hastie, Noah Simon, Jonathan Taylor and Ryan J. Tibshirani. *Strong rules for discarding predictors in lasso-type problems.* Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 74, no. 2, pages 245–266, Nov 2011. Cited on pages 164 and 165.

[Tibshirani 1996] Robert Tibshirani. *Regression Shrinkage and Selection via the Lasso.* Journal of the Royal Statistical Society. Series B (Methodological), vol. 58, no. 1, pages 267–288, 1996. Cited on pages 18, 28, 161, and 162.

[Tibshirani 2013] Ryan J. Tibshirani. *The lasso problem and uniqueness.* Electron. J. Statist., vol. 7, pages 1456–1490, 2013. Cited on pages 33, 34, 167, and 173.

[Tomasi & Bro 2006] Giorgio Tomasi and Rasmus Bro. *A comparison of algorithms for fitting the PARAFAC model.* Computational Statistics & Data Analysis, vol. 50, pages 1700–1734, 2006. Cited on page 89.

[Tosic & Frossard 2011] I. Tosic and P. Frossard. *Dictionary Learning.* IEEE Signal Processing Magazine, vol. 28, no. 2, pages 27–38, March 2011. Cited on page 37.

[Tropp 2004] J. A. Tropp. *Greed is good: algorithmic results for sparse approximation.* IEEE Transactions on Information Theory, vol. 50, no. 10, pages 2231–2242, Oct 2004. Cited on pages 34 and 35.

[Tseng & Yun 2008] P. Tseng and S. Yun. *Block-Coordinate Gradient Descent Method for Linearly Constrained Nonsmooth Separable Optimization.* Journal of Optimization Theory and Applications, vol. 140, no. 3, page 513, Sep 2008. Cited on page 59.

[Tseng 2001] P. Tseng. *Convergence of a Block Coordinate Descent Method for Nondifferentiable Minimization.* Journal of Optimization Theory and Applications, vol. 109, no. 3, pages 475–494, Jun 2001. Cited on page 59.

[Tsiligkaridis & Hero 2013] T. Tsiligkaridis and A. O. Hero. *Covariance Estimation in High Dimensions Via Kronecker Product Expansions.* IEEE Transactions on Signal

Processing, vol. 61, no. 21, pages 5347–5360, Nov 2013. Cited on pages 96, 97, 99, and 210.

[Tucker 1966] Ledyard R. Tucker. *Some mathematical notes on three-mode factor analysis*. Psychometrika, vol. 31, no. 3, pages 279–311, Sep 1966. Cited on pages 91 and 92.

[Van Loan & Pitsianis 1993] Charles F Van Loan and Nikos Pitsianis. *Approximation with Kronecker products*. In Linear algebra for large scale and real-time applications, pages 293–314. Springer, 1993. Cited on page 100.

[Vannieuwenhoven *et al.* 2013] Nick Vannieuwenhoven, Nick Vanbaelen, Karl Meerbergen and Raf Vandebril. *The dense multiple-vector tensor-vector product: An initial study*, 2013. Cited on pages 123 and 210.

[Vasilescu & Terzopoulos 2003] M. A. O. Vasilescu and D. Terzopoulos. *Multilinear subspace analysis of image ensembles*. In 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., volume 2, pages II–93, June 2003. Cited on page 92.

[Vervliet *et al.* 2016] N. Vervliet, O. Debals, L. Sorber, M. Van Barel and L. De Lathauwer. *Tensorlab 3.0*, Mar. 2016. Available online. Cited on page 104.

[Wang & Ahuja 2004] H. Wang and N. Ahuja. *Compact representation of multidimensional data using tensor rank-one decomposition*. In Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., volume 1, pages 44–47 Vol.1, Aug 2004. Cited on page 92.

[Wang *et al.* 2015] Jie Wang, Peter Wonka and Jieping Ye. *Lasso Screening Rules via Dual Polytope Projection*. Journal of Machine Learning Research, vol. 16, no. 1, pages 1063–1101, May 2015. Cited on page 164.

[Welch 1974] L. Welch. *Lower bounds on the maximum cross correlation of signals*. IEEE Transactions on Information Theory, vol. 20, no. 3, pages 397–399, May 1974. Cited on page 33.

[Wright *et al.* 2009] S. J. Wright, R. D. Nowak and M. A. T. Figueiredo. *Sparse Reconstruction by Separable Approximation*. IEEE Transactions on Signal Processing, vol. 57, no. 7, pages 2479–2493, Jul 2009. Cited on pages 19, 56, 161, and 162.

[Wu & Lange 2008] Tong Tong Wu and Kenneth Lange. *Coordinate descent algorithms for lasso penalized regression.* Ann. Appl. Stat., vol. 2, no. 1, pages 224–244, 03 2008. Cited on page 59.

[Xiang & Ramadge 2012] Z. J. Xiang and P. J. Ramadge. *Fast lasso screening tests based on correlations.* In 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2137–2140, March 2012. Cited on page 164.

[Xiang *et al.* 2011] Zhen James Xiang, Hao Xu and Peter J Ramadge. *Learning Sparse Representations of High Dimensional Data on Large Scale Dictionaries.* In Advances in Neural Information Processing Systems (NIPS), volume 24, pages 900–908, Granada, Spain, Dec 2011. Cited on page 164.

[Xiang *et al.* 2017] Z. J. Xiang, Y. Wang and P. J. Ramadge. *Screening Tests for Lasso Problems.* IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 5, pages 1008–1027, May 2017. Cited on pages 167 and 211.

[Xing *et al.* 2012] Z. Xing, M. Zhou, A. Castrodad, G. Sapiro and L. Carin. *Dictionary Learning for Noisy and Incomplete Hyperspectral Images.* SIAM Journal on Imaging Sciences, vol. 5, no. 1, pages 33–56, 2012. Cited on pages 126 and 146.

[Zhang *et al.* 2013] Y. Zhang, Z. Jiang and L. S. Davis. *Learning Structured Low-Rank Representations for Image Classification.* In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 676–683, June 2013. Cited on page 40.

[Zhao & Yang 2015] Y. Zhao and J. Yang. *Hyperspectral Image Denoising via Sparse Representation and Low-Rank Constraint.* IEEE Transactions on Geoscience and Remote Sensing, vol. 53, no. 1, pages 296–308, Jan 2015. Cited on pages 126 and 146.

[Zheng *et al.* 2016] Chun-Hou Zheng, Yi-Fu Hou and Jun Zhang. *Improved sparse representation with low-rank representation for robust face recognition.* Neurocomputing, pages –, 2016. Cited on page 40.

[Zubair & Wang 2013] S. Zubair and Wenwu Wang. *Tensor dictionary learning with sparse TUCKER decomposition.* In 2013 18th International Conference on Digital Signal Processing (DSP), pages 1–6, July 2013. Cited on pages 41 and 99.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre: **Approximations structurées pour l'accélération de problèmes inverses parcimonieux**

Mot clés : Optimisation convexe, Apprentissage automatique, Problèmes inverses, Représentation parcimonieuse, Algèbre tensorielle

Resumé : En raison de la vertigineuse croissance des données disponibles, la complexité computationnelle des algorithmes traitant les problèmes inverses parcimonieux peut vite devenir un goulot d'étranglement. Dans cette thèse, nous explorons deux stratégies principales pour accélérer de tels algorithmes. D'abord, nous étudions l'utilisation de dictionnaires structurés rapides à manipuler. Une famille de dictionnaires écrits comme une somme de produits Kronecker est proposée. Ensuite, nous développons des tests d'élagage sûrs, capables d'identifier et éliminer des atomes inutiles (colonnes de la matrice dictionnaire ne correspondant pas au support de la solution), malgré l'utilisation de dictionnaires approchés.

Title: **Accelerating sparse inverse problems using structured approximations**

Keywords : Convex optimization, Machine learning, Inverse problems, Sparse representations, Tensor algebra

Abstract : As the quantity and size of available data grow, the existing algorithms for solving sparse inverse problems can become computationally intractable. In this work, we explore two main strategies for accelerating such algorithms. First, we study the use of structured dictionaries which are fast to operate with. A particular family of dictionaries, written as a sum of Kronecker products, is proposed. Then, we develop *stable* screening tests, which can safely identify and discard useless atoms (columns of the dictionary matrix which do not correspond to the solution support), despite manipulating approximate dictionaries.