Orthogonal greedy algorithms for non-negative sparse reconstruction

Thi Thanh Nguyen

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Algorithmes gloutons orthogonaux sous contrainte de positivité

THÈSE

présentée et soutenue publiquement le 18 Novembre 2019

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention automatique, traitement du signal et génie informatique)

par

Thi Thanh NGUYEN

Composition du jury

Rapporteurs :	Charles DOSSAL	Professeur à l'INSA de Toulouse
	Loïc DENIS	MdC HDR à l'Université de Saint-Etienne
Examinateurs :	Christian HEINRICH	Professeur à l'Université de Strasbourg
	Laura REBOLLO-NEIRA	Lecturer at Aston University
Encadrants :	Charles SOUSSEN	Professeur à CentraleSupelec
	Jérôme IDIER	Directeur de Recherche CNRS
	El-Hadi DJERMOUNE	MdC HDR à l'Université de Lorraine

Centre de Recherche en Automatique de Nancy
CRAN UMR 7039, Université de Lorraine, CNRS
Faculté des Sciences et Technologies, BP 70239, 54506 Vandoeuvre Cedex

Acknowledgments

This thesis was impossible without the guidance of my supervisors Charles Soussen, Jérôme Idier and El-Hadi Djermoune. I am deeply grateful for their kindness and support during the three years of my PhD. I would like to thank the members of my jury: Charles Dossal, Loïc Denis, Christian Heinrich and Laura Rebollo-Neira for their time reviewing my work. It is a great honor for me to present my work to them. Special thanks go to Mehrdad Yaghoobi for providing his Matlab implementation of FNNOLS and Hassan Mortada for sharing the figure of spectra decomposition. I would like to acknowledge to my laboratory CRAN and the Inverse Problem Group (GPI) at L2S for providing the best working condition for my thesis preparation. Finally, I would like to thank my family for their unconditional love and my friends for their endless encouragement.

Contents

Notations and abbreviations

Notations

$\boldsymbol{y}$	Data vector
H	Dictionary (observation matrix)
$\boldsymbol{x}$	Solution vector
$\boldsymbol{\epsilon}$	Error vector
S	Support of solution
$\|S\|$	Cardinality of S
$\bar{S}$	Complement of S
$\|\cdot\|_0$	ℓ_0-"norm" counting the number of nonzero entries
$\|\cdot\|_p$	ℓ_p-norm
$\|\cdot\|$	Euclidean norm
$.^t$	Transpose operator
$.^\dagger$	Moore-Penrose pseudo-inverse
$.^\perp$	Orthogonal complement
$\text{span}(H)$	Column space of H
H_S	Subdictionary indexed by S
$\boldsymbol{x}(S)$	Subvector indexed by S
$\boldsymbol{x}_S$	Subvector indexed by S (in Chapter 4 only)
$\hat{\boldsymbol{x}}_S$	ULS solution related to S
$\hat{\boldsymbol{x}}_S^+$	NNLS solution related to S
$\boldsymbol{h}_i$	Column of dictionary H (atom)
$\tilde{\boldsymbol{h}}_i^S$	Orthogonal projection of $\boldsymbol{h}_i$ onto $\text{span}(H_S)^\perp$ (projected atom)
$\tilde{\boldsymbol{g}}_i^S$	Normalization of $\tilde{\boldsymbol{h}}_i^S$ (normalized projected atom)
μ	Mutual coherence

Abbreviations

OMP	Orthogonal Matching Pursuit
OLS	Orthogonal Least Squares
Oxx	Generic acronym for OMP and OLS
ULS	Unconstrained Least Squares
NNLS	Non-Negative Least Squares
NNOG	Non-Negative Orthogonal Greedy
NNOMP	Non-Negative Orthogonal Matching Pursuit
NNOLS	Non-Negative Orthogonal Least Squares
SNNOLS	Suboptimal Non-Negative Orthogonal Least Squares

Algorithmes gloutons orthogonaux sous contrainte de positivité

Reconstruction de signaux parcimonieux positifs

De nombreux domaines applicatifs conduisent à résoudre des problèmes inverses où le signal ou l'image à reconstruire sont à la fois parcimonieux et positifs. Citons les exemples de la télédétection [Iordache et al., 2011], de l'analyse de signaux audio [Virtanen et al., 2013], de la chimiométrie [Bro and De Jond, 1997], de l'astrophysique [Högbom, 1974] ou de la reconstruction tomographique [Petra and Schnörr, 2014]. La reconstruction d'un signal parcimonieux et positif est naturellement formulée comme un problème de minimisation ℓ_0 sous contrainte de positivité, du type:

$$\min_{\boldsymbol{x}} \|\boldsymbol{y} - H\boldsymbol{x}\|^2 \quad \text{s.c.} \quad \boldsymbol{x} \geq \boldsymbol{0},\ \|\boldsymbol{x}\|_0 \leq K, \tag{1}$$

où $\|\,.\,\|$ et $\|\,.\,\|_0$ représentent respectivement la norme euclidienne et la "norme" ℓ_0, correspondant au nombre d'éléments non-nuls d'un vecteur.

Ce problème est connu pour être NP-difficile [Natarajan, 1995]. On distingue plusieurs catégories d'approches allant des méthodes de relaxation convexe (*via* les algorithmes proximaux notamment) aux méthodes d'optimisation non-convexe d'un critère continu et non-différentiable approchant [Gasso et al., 2009] ou reformulant [Soubies et al., 2015] le critère (1), et aux méthodes abordant directement l'optimisation du critère ℓ_0 non-modifié *via* une stratégie d'optimisation exacte [Bourguignon et al., 2016] ou sous-optimale. Les algorithmes gloutons font partie de cette dernière catégorie. Leur principe est de partir d'une solution nulle et de construire progressivement une solution K-parcimonieuse en sélectionnant un à un des atomes du dictionnaire H, avec une mise à jour des amplitudes associées et du résidu d'approximation $\boldsymbol{r} = \boldsymbol{y} - H\boldsymbol{x}$ au fur et à mesure que les nouveaux atomes sont sélectionnés.

Les algorithmes gloutons orthogonaux comme *Orthogonal Matching Pursuit* (OMP) et *Orthogonal Least Squares* (OLS) mettent à jour les amplitudes par un calcul de projection orthogonale. En notant S le support courant, les amplitudes associées $\boldsymbol{x}(S)$ sont celles pour lesquelles le résidu d'approximation $\boldsymbol{r}_S = \boldsymbol{y} - H_S\boldsymbol{x}(S)$ est minimum en norme ℓ_2, où H_S est la sous-matrice de H réduite à S. Leur calcul nécessite la résolution d'un problème de moindres carrés non-contraints (ULS pour *Unconstrained Least Squares*). Les implémentations efficaces des algorithmes gloutons reposent sur l'idée que la solution de ce problème de moindres carrés peut être mise à jour rapidement lorsque le support est augmenté d'un élément ($S \leftarrow S \cup \{\ell\}$), modulo la mise à jour d'une factorisation matricielle de type Gram-Schmidt ou de Cholesky [Sturm and Christensen, 2012].

Les algorithmes gloutons généralisés à la reconstruction de signaux positifs possèdent une structure proche des algorithmes gloutons standard, avec à chaque itération, la sélection d'un nouvel atome et la mise à jour des amplitudes en résolvant un problème de moindres carrés sous contrainte de positivité (NNLS pour *Non-Negative Least Squares*) [Bruckstein et al., 2008]. Cependant, leur implémentation pose une difficulté majeure liée au fait que les problèmes NNLS ne possèdent pas de solution explicite. Dans la littérature, ces algorithmes sont réputés lents et des solutions récursives approchées ont été privilégiées [Yaghoobi et al., 2015]. Dans [Nguyen et al., 2019a], nous montrons que des implémentations récursives exactes sont possibles sans nécessiter de surcoût calculatoire substantiel. Cette contribution est présentée dans les chapitres 2 et 3.

L'analyse théorique des algorithmes de reconstruction parcimonieuse vise à caractériser leur capacité à reconstruire le support d'une représentation K-parcimonieuse. Les analyses classiques

Algorithm 1: Structure générale des algorithmes gloutons orthogonaux pour minimiser (1). $\boldsymbol{h}_i$ désigne la i-ème colonne du dictionnaire H, et $\hat{\boldsymbol{x}}_S^+$ représente la solution NNLS associée au support S.

entrées: $\boldsymbol{y}, H, K$
sorties : $\boldsymbol{x}$

$\boldsymbol{x} \leftarrow \boldsymbol{0}$; $S \leftarrow \emptyset$; $\boldsymbol{r}_S \leftarrow \boldsymbol{y}$;
tant que $|S| < K$ *et* $\max_{i \notin S} \boldsymbol{h}_i^{\mathsf{t}} \boldsymbol{r}_S > 0$ **faire**
  Sélectionner un atome $\ell \notin S$;
  $S \leftarrow S \cup \{\ell\}$;
  $\boldsymbol{x} \leftarrow \hat{\boldsymbol{x}}_S^+$;
  $S \leftarrow \mathrm{supp}(\boldsymbol{x})$;
  $\boldsymbol{r}_S = \boldsymbol{y} - H\boldsymbol{x}$;
fin

d'algorithmes gloutons en K-itérations cherchent à prouver que chaque itération sélectionne un atome dans le support du vecteur parcimonieux inconnu [Tropp, 2004]. Elles ne s'étendent pas directement aux algorithmes gloutons non-négatifs, car les propriétés géométriques sur lesquelles elles s'appuient (en particulier, l'orthogonalité entre le résidu et l'espace d'approximation) ne sont plus valables. Il est donc nécessaire de repenser ces analyses. Dans [Nguyen et al., 2019d], nous proposons une analyse basée sur l'hypothèse de cohérence mutuelle faible $\mu < 1/(2K-1)$ pour analyser de façon unifiée différents algorithmes proposés dans la littérature [Bruckstein et al., 2008, Yaghoobi et al., 2015]. Les mécanismes sur lesquels cette analyse repose sont synthétisés dans le chapitre 4.

Algorithmes gloutons orthogonaux positifs

La structure générale des algorithmes gloutons orthogonaux positifs (NNOG pour *Non-Negative Orthogonal Greedy algorithms*) est résumée dans l'algorithme 1. Une itération de l'algorithme consiste en trois étapes:

1. Sélection d'un nouvel atome $\ell \notin S$ pour enrichir le support S de la représentation parcimonieuse.

2. Mise à jour des amplitudes liées aux atomes sélectionnés ($i \in S$) *via* la résolution du problème NNLS associé au support de $\boldsymbol{x}$:

$$\hat{\boldsymbol{x}}_S^+ = \underset{\boldsymbol{x} \geq \boldsymbol{0}}{\arg\min} \|\boldsymbol{y} - H\boldsymbol{x}\|^2 \ \text{ s.c. } \ \mathrm{supp}(\boldsymbol{x}) \subset S. \tag{2}$$

3. Compression du support.

La principale différence structurelle avec les algorithmes gloutons orthogonaux classiques est la troisième étape de compression du support. On remarque en effet que les coefficients de la représentation parcimonieuse obtenus après l'étape NNLS s'annulent lorsque les contraintes de positivité du problème (2) sont activées. L'étape de compression que nous avons proposée dans [Nguyen et al., 2017, Nguyen et al., 2019a] vise à supprimer les indices correspondants aux coefficients annulés, pour faire coïncider S avec le support de la solution courante $\boldsymbol{x}$. La structure de l'algorithme est donc sensiblement différente des algorithmes gloutons classiques, qui possèdent une propriété d'emboîtement entre supports.

L'algorithme 1 est conçu comme un algorithme de descente. Ses deux conditions d'arrêt garantissent que le vecteur obtenu est exactement K-parcimonieux ($|S| = K$) ou qu'aucune

décroissance du critère $\|\boldsymbol{y}-H\boldsymbol{x}\|^2$ n'est possible en ajoutant un nouvel atome au support courant. La condition $\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S > 0$ garantit en effet que l'erreur quadratique peut décroître en sélectionnant l'atome i, ce qui nous a conduit à introduire le concept de "atome descendant" dans le chapitre 2.

La structure des algorithmes NNOG dans le tableau 1 est commune à différents algorithmes comme *Non-Negative Orthogonal Matching Pursuit* (NNOMP) [Bruckstein et al., 2008], défini comme une généralisation d'OMP, *Non-Negative Orthogonal Least Squares* (NNOLS) et *Suboptimal Non-Negative Orthogonal Least Squares* (SNNOLS), qui sont deux généralisations d'OLS [Yaghoobi and Davies, 2015]. Dans leur principe, ces algorithmes diffèrent uniquement par leur règle de sélection du nouvel atome:

$$\ell^{\mathrm{NNOMP}} \in \arg\max_{i\notin S} \boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S, \tag{3}$$

$$\ell^{\mathrm{SNNOLS}} \in \arg\max_{i\notin S} \boldsymbol{b}_i^{\mathrm{t}}\boldsymbol{r}_S, \tag{4}$$

$$\ell^{\mathrm{NNOLS}} \in \arg\min_{i\notin S} \|\boldsymbol{r}^+_{S\cup\{i\}}\|^2, \tag{5}$$

où $\boldsymbol{r}^+_{S\cup\{i\}} := \boldsymbol{y} - H\hat{\boldsymbol{x}}^+_{S\cup\{i\}}$ désigne le "résidu positif" lié à la solution NNLS pour le support $S\cup\{i\}$, et $\boldsymbol{b}_i$ représente le projeté orthogonal normalisé de $\boldsymbol{h}_i$ sur le sous-espace orthogonal aux atomes $\boldsymbol{h}_j$, $j\in S$.

La règle de sélection de NNOMP est une extension directe de celle d'OMP, qui maximise $|\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S|$. De même, NNOLS suit le mécanisme d'OLS, où l'atome sélectionné est celui qui engendre une décroissance maximale du résidu d'approximation. Finalement, SNNOLS est inspiré d'une vision géométrique d'OLS en terme d'angle maximal entre le projeté orthogonal $\boldsymbol{b}_i$ de l'atome candidat et le résidu courant. Si les deux interprétations d'OLS en terme d'optimisation et de géométrie projective sont équivalentes, cette équivalence n'est plus valable dans le cas des algorithmes gloutons positifs, ce qui conduit à deux algorithmes distincts SNNOLS et NNOLS (voir chapitre 2).

NNOMP est clairement le moins coûteux des trois algorithmes, car sa règle de sélection ne repose que sur le calcul d'un produit scalaire par atome. Néanmoins, ses limites de performance pour des problèmes mal posés nous ont poussé à nous intéresser aux algorithmes inspirés d'OLS, dont la complexité calculatoire est accrue. La différence de complexité entre NNOLS et SNNOLS s'avère délicate à analyser à partir des expressions (4) et (5). Nous avons montré que (4) équivaut à (voir chapitre 2):

$$\ell^{\mathrm{SNNOLS}} \in \arg\min_{i\notin S}\left\{\min_{\boldsymbol{u},v\geq 0} \|\boldsymbol{y} - H_S\boldsymbol{u} - \boldsymbol{h}_i v\|^2\right\}. \tag{6}$$

Il apparaît alors que le test de sélection d'atome de SNNOLS est largement moins coûteux que celui de NNOLS, car (6) est un problème de moindres carrés non-contraint par rapport à $\boldsymbol{u}$, et sous contrainte de positivité par rapport à la variable scalaire v. Par opposition, le calcul de $\boldsymbol{r}^+_{S\cup\{i\}}$ dans le cas de NNOLS nécessite de résoudre un problème NNLS avec des contraintes de positivité sur $\boldsymbol{u}$ et v.

Implémentation rapide

Dans notre article de conférence [Nguyen et al., 2017], nous avons présenté une implémentation rapide de NNOMP basée sur la résolution récursive des sous-problèmes NNLS par l'algorithme des contraintes actives [Lawson and Hanson, 1974] avec un démarrage à chaud correspondant à une initialisation par l'itéré courant de NNOMP. L'algorithme des contraintes actives résout le problème NNLS de façon exacte en un nombre fini d'itérations. Comme il possède une structure gloutonne, les implémentations qui résultent de son couplage avec les algorithmes NNOG sont

pleinement récursives. Nous avons constaté empiriquement que l'algorithme des contraintes actives avec démarrage à chaud converge en un très faible nombre d'itérations.

Dans le chapitre 2, nous avons proposé une série d'améliorations supplémentaires visant à résoudre les problèmes de sélection d'atomes (3)-(5) de manière rapide. Il s'agit *(i)* de prédétecter que certains atomes candidats ne sont pas descendants et donc les écarter; *(ii)* pour les atomes restants et dans le cas de NNOLS (qui requiert la résolution répétée de problèmes NNLS), un précalcul des solutions du problème ULS pour le support candidat $S \cup \{i\}$ permet de détecter des atomes non-optimaux au sens de (5), ce qui nous évite des appels à NNLS.

À titre d'exemple, les tests effectués dans le chapitre 3 pour un problème de déconvolution impulsionnelle de taille $\boldsymbol{x} \in \mathbb{R}^{1140}$ avec un filtre passe-bas montrent que pour un support S de taille 80, il y a 30 % d'atomes candidats ($i \notin S$) non testés, 68 % d'atomes candidats pour lesquels seule la solution ULS associée à $S \cup \{i\}$ est calculée, et seulement 2% des atomes candidats pour lequel un appel à NNLS est requis.

Analyse d'algorithmes gloutons orthogonaux positifs

L'analyse théorique des algorithmes NNOG en termes de reconstruction exacte du support d'une représentation parcimonieuse $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ (avec $\boldsymbol{x} \geq \boldsymbol{0}$) est un problème ouvert. En effet, à notre connaissance, il n'existe pas d'analyse mathématique des algorithmes NNOMP, NNOLS et SNNOLS. Néanmoins, Bruckstein *et al.* [Bruckstein et al., 2008] ont conjecturé que NNOMP présente des garanties de reconstruction exacte en K itérations si la cohérence mutuelle du dictionnaire $\mu = \max_{i \neq j} |\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle|$ vérifie:

$$\mu < \frac{1}{2K-1}. \tag{7}$$

Dans le chapitre 4, nous avons prouvé cette conjecture non seulement pour NNOMP mais aussi pour les algorithmes NNOLS et SNNOLS. Notre technique de preuve s'appuie sur une analyse de préservation de signe par les algorithmes OMP et OLS. Nous avons d'abord établi le résultat suivant.

Theorem 1. *[Nguyen et al., 2019d, Cor. III.1] Supposons que $\mu < \frac{1}{2K-1}$. Soit $\boldsymbol{y} = H\boldsymbol{x}$ une représentation K-parcimonieuse avec $\boldsymbol{x} \geq \boldsymbol{0}$. Alors OMP et OLS reconstruisent correctement le support de $\boldsymbol{x}$ en K itérations. De plus, pour chaque itération $k = 1, \ldots, K$, les poids des k atomes sélectionnés sont positifs.*

La première partie de ce résultat est bien connue. En effet, (7) est une condition nécessaire [Cai et al., 2010] et suffisante [Tropp, 2004, Herzet et al., 2013] de reconstruction exacte quelles que soient les amplitudes non nulles des poids dans $\boldsymbol{x}$. La deuxième partie du théorème indique qu'en régime de reconstruction exacte, l'itéré courant d'OMP/OLS a des coordonnées positives. Comme cet itéré correspond à la solution du problème des moindres carrés (ULS) associé au support courant S, la solution NNLS correspondant au même support coïncide avec la solution ULS. La très grande proximité entre les règles de sélection d'atomes d'OMP et de NNOMP (3) nous permet de conclure que les itérés de NNOMP et OMP coïncident. De façon parallèle, ceux de NNOLS, SNNOLS et OLS coïncident, d'où le résultat suivant.

Theorem 2. *[Nguyen et al., 2019d, Cor. III.2] Supposons que $\mu < \frac{1}{2K-1}$. Soit $\boldsymbol{y} = H\boldsymbol{x}$ une représentation K-parcimonieuse avec $\boldsymbol{x} \geq \boldsymbol{0}$. Alors les itérés de NNOMP coïncident avec ceux de OMP. De plus, les itérés de NNOLS et SNNOLS coïncident avec ceux de OLS. Par conséquent, NNOMP, SNNOLS et NNOLS reconstruisent tous le support de $\boldsymbol{x}$ en K itérations.*

Notons enfin que NNLS étant des problèmes d'optimisation contraints, certaines contraintes de positivité sont susceptibles d'être activées à chaque itération des algorithmes NNOG, entraînant l'annulation des poids correspondants dans la représentation parcimonieuse. Dans ce cas, l'algorithme NNOG doit effectuer un nombre d'itérations supérieur à K pour pouvoir reconstruire une représentation réellement K-parcimonieuse. Le théorème 2 indique que le phénomène d'annulation des poids ne se produit pas en régime de reconstruction exacte, où la reconstruction exacte se produit en K itérations.

Conclusions

Dans ce document, nous présentons un panorama unifié des algorithmes gloutons orthogonaux non-négatifs (NNOG), conçus comme des algorithmes de descente pour le problème de minimisation ℓ_0 (1), ainsi qu'une analyse de leur propriété de reconstruction exacte d'un vecteur K-parcimonieux positif sous l'hypothèse de cohérence mutuelle inférieure à $\frac{1}{2K-1}$.

Du point de vue pratique, nous avons développé un logiciel Matlab en accès libre qui intègre les implémentations récursives proposées dans la section 2.4 [Nguyen et al., 2019b]. Nous avons effectué des simulations numériques extensives (voire chapitre 3) visant à comparer les algorithmes NNOMP, SNNOLS et NNOLS pour des problèmes difficiles de type déconvolution impulsionnelle avec un filtre passe-bas pour lesquels la cohérence mutuelle est proche de 1. Il ressort d'abord de ces tests que bien que les algorithmes non-négatifs sont plus coûteux que les versions standards OMP et OLS, le surcoût n'est pas rédhibitoire grâce aux implémentations rapides proposées (le ratio en temps de calcul entre les versions standards et non-négatives n'excède jamais un facteur 5 pour des problèmes de taille 1200 avec $K = 80$). NNOMP s'avère plus rapide mais nettement moins précis que SNNOLS et NNOLS pour reconstruire le vecteur parcimonieux. La précision est évaluée en termes d'erreur quadratique sur $\boldsymbol{x}$ et de nombre de vrais positifs dans le support trouvé. Ces deux algorithmes fournissent des reconstructions parcimonieuses de qualité comparable, SNNOLS étant sensiblement plus rapide. Plus précisément, on peut noter que le nombre d'itérations de NNOLS pour atteindre un support de taille K est sensiblement plus faible que celui de SNNOLS (en moyenne 120 itérations contre 150 sont nécessaires pour obtenir un support de taille $K = 80$ pour l'exemple donné dans la section 3.3) mais le coût par itération de NNOLS reste plus important. En revanche, les défauts de SNNOLS semblent être plus prononcés pour des problèmes de type "super-résolution" en utilisant des grilles fines (impliquant des dictionnaires plus gros et plus corrélés), où le nombre d'itérations de SNNOLS augmente significativement, induisant un coût calculatoire qui devient globalement supérieur à celui de NNOLS.

Orthogonal greedy algorithms for non-negative sparse reconstruction

Chapter 1

Introduction

Contents

This work is supported by the BECOSE project (2016-2019)[1] aiming to develop efficient sparse algorithms for ill-conditioned inverse problems. The BECOSE project contains three main tasks. The first task is to develop new sparsity-based algorithms dedicated to ill-posed inverse problems by exploiting additional structural information such as non-negativity or by working on continuous dictionaries. The second task is to enrich the theoretical analysis of sparse algorithms, especially the stepwise orthogonal greedy search algorithms, by taking into account prior knowledge on the signs, coefficient values, and partial support information. The third task is to assess the proposed algorithms in the context of tomographic Particle Image Velocimetry (PIV) application. This thesis addresses partly the first and second tasks of the BECOSE project.

In this introduction, we start by presenting the non-negative sparse reconstruction problem and some motivating applications. We next review the state-of-the-art sparse algorithms with a focus on greedy algorithms and recall the state-of-the-art exact recovery analyses of greedy algorithms. Then we summarize our contributions related to non-negative orthogonal greedy algorithms and exact recovery analysis. We end up with the list of resulting publications.

[1]Funded by the French National Research Agency (No. ANR-15-CE23-0021). http://becose.univ-lorraine.fr

1.1 Non-negative sparse reconstruction

Sparse approximation appears in a wide range of applications, especially in signal and image processing and compressive sensing [Elad, 2010]. Given a signal $\boldsymbol{y} \in \mathbb{R}^m$ and a redundant dictionary $H \in \mathbb{R}^{m \times n}$, one is interested in finding the smallest set of dictionary columns (also called atoms) that describes $\boldsymbol{y}$ well enough. In other words, one seeks for a vector $\boldsymbol{x} \in \mathbb{R}^n$ that gives the best approximation $\boldsymbol{y} \approx H\boldsymbol{x}$ and has the fewest non-zero coefficients (*i.e.*, the sparsest solution). A simple illustration is given in Figure 1.1. Sparse approximation often leads to solving an ℓ_0 minimization problem that takes one of the following forms:

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{x}\|_0 \quad \text{s.t.} \quad \|\boldsymbol{y} - H\boldsymbol{x}\|_2 \leq \epsilon \tag{$\ell_0 C_\epsilon$}$$

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{y} - H\boldsymbol{x}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{x}\|_0 \leq K \tag{$\ell_0 C_K$}$$

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{y} - H\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_0 \tag{$\ell_0 P$}$$

in which ϵ, K and λ are positive quantities related to the noise standard deviation, the sparsity level and the regularization strength, respectively. Let us recall that the ℓ_0-"norm" counts the number of non-zero coefficients: $\|\boldsymbol{x}\|_0 = \text{Card}\{i : x_i \neq 0\}$. According to its definition, ℓ_0 is not a proper norm since it is not homogeneous. However, it is widely used as a pseudo-norm in statistics, scientific computing and information theory. Hereafter, we call $H\boldsymbol{x}$ a K-sparse representation if $\|\boldsymbol{x}\|_0 = K$. Without loss of generality, we assume that the dictionary H is normalized (so every atom $\boldsymbol{h}_i$ has ℓ_2-norm equal to 1). The ℓ_2-norm $\|\cdot\|_2$ will be also denoted $\|\cdot\|$.

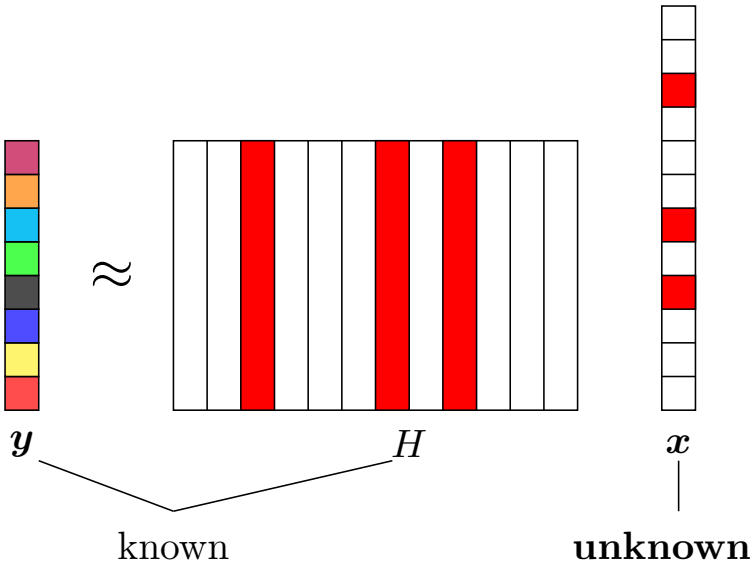


Figure 1.1: A simple illustration of sparse approximation. The red color indicates the non-zero coefficients and the corresponding dictionary atoms.

In several application fields such as geoscience and remote sensing [Iordache et al., 2011, Esser et al., 2013], audio [Virtanen et al., 2013], chemometrics [Bro and De Jond, 1997], bioinformatics [Slawski et al., 2012], astrophysics [Högbom, 1974] and computed tomography [Petra and Schnörr, 2014], the signal or image of interest is sparse, but also non-negative. In such contexts, a common practice is to regularize the inverse problem to favor both sparsity and non-negativity

of the sought signal, see, *e.g.,* [Hoyer, 2004, Petra and Schnörr, 2014, Slawski et al., 2012, Rapin et al., 2013]. This regularization leads to non-negative ℓ_0 minimization problems as follows.

$$\min_{\boldsymbol{x}\geq\mathbf{0}} \quad \|\boldsymbol{x}\|_0 \quad \text{s.t.} \quad \|\boldsymbol{y}-H\boldsymbol{x}\|_2 \leq \epsilon \qquad (\ell_0 C_\epsilon+)$$

$$\min_{\boldsymbol{x}\geq\mathbf{0}} \quad \|\boldsymbol{y}-H\boldsymbol{x}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{x}\|_0 \leq K \qquad (\ell_0 C_K+)$$

$$\min_{\boldsymbol{x}\geq\mathbf{0}} \quad \|\boldsymbol{y}-H\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_0 \qquad (\ell_0 P+)$$

To address these problems, a usual approach is to incorporate the non-negativity into existing sparse solvers. The obtained algorithms are called non-negative sparse solvers. However, the extension of sparse solvers to the non-negative setting is not always obvious. Besides, the non-negative sparse solvers may be significantly more expensive than the primary sparse solvers. We will discuss this later in Section 1.3. Now, let us take a quick tour on some applications of non-negative sparse reconstruction.

1.2 Some applications

In this section, we briefly discuss two among several applications in which both sparsity and non-negativity are required for the sought solution. The presented applications contain tomographic PIV reconstruction and spectral unmixing. More details on other applications can be found in the references cited in the previous section.

1.2.1 Tomographic PIV reconstruction

Tomographic Particle Image Velocimetry (Tomo-PIV) [Elsinga et al., 2006] is a method to reconstruct 3-D maps of the velocity field of the flows in fluids. The fluid is seeded with point-like tracer particles whose motion is used to calculate the speed and direction of the flow. A laser with an optical arrangement is used to limit the physical region illuminated. Several (at least four) cameras are placed at different angles to record simultaneous views (radiograph, *i.e.*, projection images) of the illuminated particles. The volume is then reconstructed to yield a discretized 3-D intensity field. The principle of Tomo-PIV is summarized in Figure 1.2.

Tomo-PIV contains two steps: (i) reconstruction of 3-D volume from 2-D radiographs and (ii) estimation of the vector field from reconstructed volumes at consecutive times. The reconstruction procedure is an under-determined inverse problem since the number of unknowns (the number of voxels of the 3-D volume) is usually much higher than the number of data (the number of radiograph pixels). The light intensity corresponding to a voxel is nonzero if a particle is present at that voxel and zero otherwise. Since the number of tracer particles is small compared with the number of voxels and the light intensity is non-negative, the sought solution corresponding to the volume intensity is sparse and non-negative [Barbu and Herzet, 2016]. Therefore, Tomo-PIV reconstruction leads to a non-negative sparse approximation problem.

One example of Tomo-PIV reconstruction is given in Figs. 1.3-1.4. The simulated discretized 3-D volume of dimension $79 \times 75 \times 12$ with 81 tracer particles is reconstructed from 4 quantized radiographs of size 32×32[2]. Here the data vector $\boldsymbol{y}$ is built from the 4 images whose pixels are unfolded and rearranged in lexicographical order. The ground truth vector $\boldsymbol{x}$ is built similarly from the unfolded discretized 3-D volume and the amplitudes represent the light intensities at each voxel of the discretized volume. The dictionary H is built from the projection operator [Barbu and Herzet, 2016], each column is the projection of each voxel. It should be noted that the choice of the 3-D grid is somewhat arbitrary although there are standard conventions in Tomo-PIV. The choice of a fine grid leads to a high resolution but also a more difficult problem

[2]This work is done within the scope of the collaboration with ONERA in the BECOSE project.

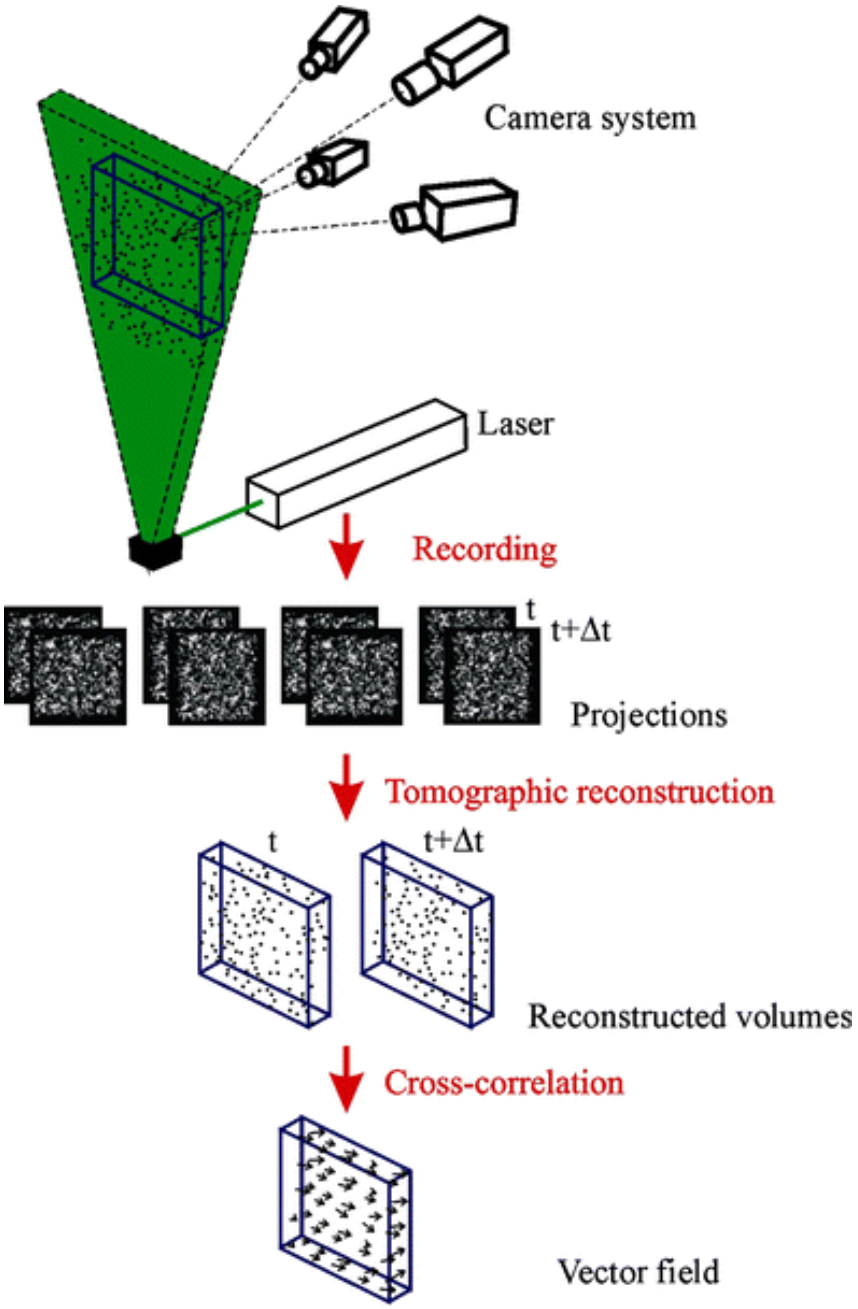


Figure 1.2: Principle of Tomo-PIV (source: [Elsinga et al., 2006])

since the number of voxels dramatically increases. It is hence important to know until which resolution one can handle (see Chapter 3 for further discussion). A recent trend is to work with a continuous dictionary (off-the-grid approach) [Ait Tilat et al., 2019, Elvira et al., 2019]. In this thesis, we will not explore the continuous approach. Instead, we will elaborate on the case of a discrete dictionary corresponding to a fine grid.

Figure 1.4 presents our reconstruction from the simulated data in Figure 1.3 using the SBR algorithm [Soussen et al., 2011]. The rate of correct detection is 68 %. It should be noted that correct detection is the detection of a particle in the same voxel as the ground truth particle. The rate 68 % might seem low but, as can be seen in Figure 1.4, the reconstruction is very accurate in the qualitative viewpoint: each detected particle is located very close to a ground truth particle.

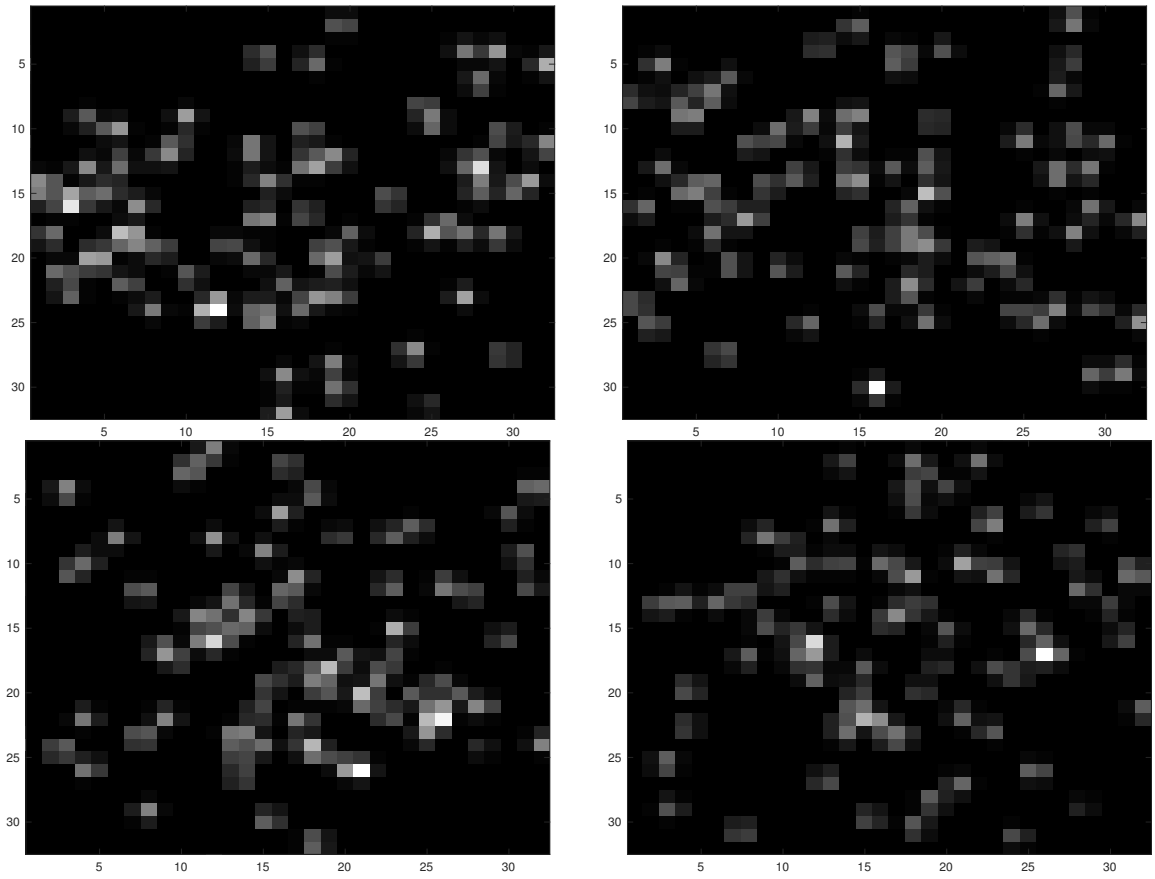


Figure 1.3: Four quantized images in a Tomo-PIV simulation provided by ONERA.

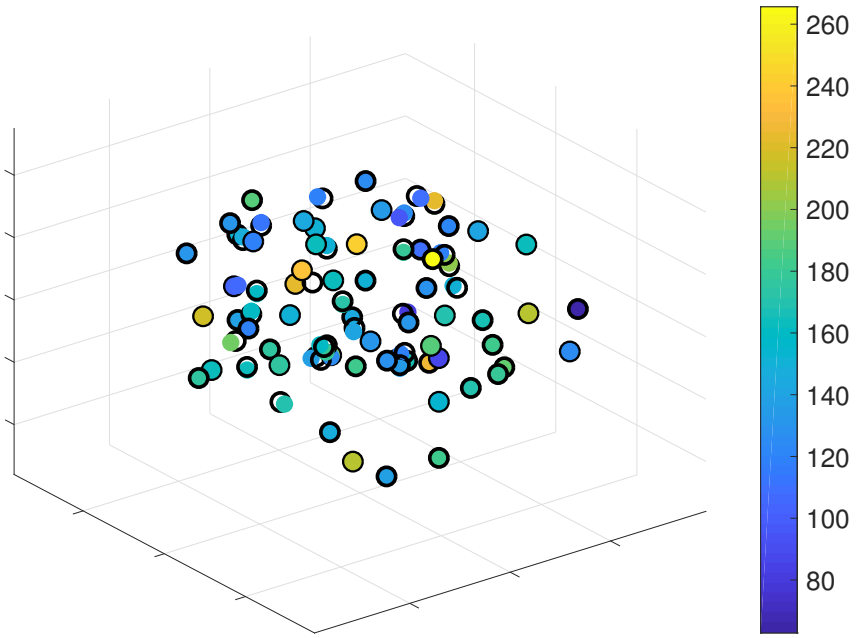


Figure 1.4: Reconstructed discretized 3-D volume of a Tomo-PIV simulation using the SBR algorithm. The black circles represent the true particles. The detected particles are represented with bullets where the color represents the light intensity.

1.2.2 Spectral unmixing

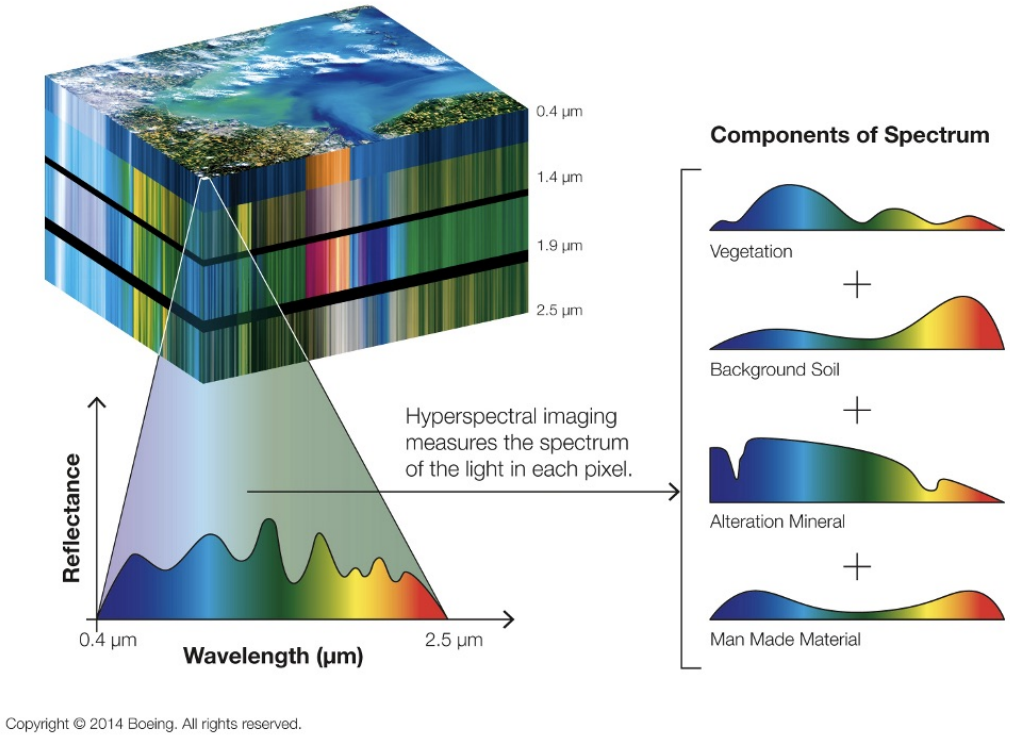


Figure 1.5: Spectral unmixing (image credit: Boeing)

Hyperspectral images are often presented as a cube with two spatial dimensions and one spectral dimension. Each hyperspectral pixel is a spectrum collecting light reflectance at different spectral bands. Very often, each spectrum of a hyperspectral image can be seen as a linear mixture of pure spectra related to several materials [Bioucas-Dias et al., 2012]. This happens when different materials are combined in the scene location or when the spatial resolution of the hyperspectral sensor is low enough for many materials to jointly occupy a single pixel. The linear mixture model assumes that the mixed pixel spectrum can be presented as a linear combination of spectra of constituent materials [Keshava and Mustard, 2002]. Therefore, decomposing the spectrum helps to detect the materials constituting the measured pixel.

Spectral unmixing is the procedure that decomposes a mixed spectrum into different spectra (endmembers) and the corresponding fractions (abundances). This decomposition is often done using a library (dictionary) of spectra [Mortada, 2018]. Since the number of constituting spectra is often very small compared with the number of spectra in the library and the corresponding fractions are non-negative, spectral unmixing leads to a non-negative sparse approximation problem [Iordache et al., 2011, Wang et al., 2018]. Using our notation, the data vector $\boldsymbol{y}$ is the mixed spectrum, the dictionary H is the library of spectra in which each column presents a pure spectrum. One looks for the vector $\boldsymbol{x}$ containing the fractions of each pure spectrum in the mixed spectrum $\boldsymbol{y}$. In Figure 1.5, the mixed spectrum (bottom left) is decomposed as a linear combination of 4 spectra (right) extracted from the large size dictionary (not shown) which may contain many hundreds or many thousands of spectra.

1.3 State-of-the-art algorithms

The aforementioned ℓ_0 minimization problems can be seen as discrete (combinatorial problem) since the main difficulty is to find the support of the sparse representation $S = \{i : x_i \neq 0\}$.

The ℓ_0 minimization problems are known to be NP-hard for general dictionaries [Natarajan, 1995, Davis et al., 1997]. This means that when the problem dimension is large, it is hopeless to look for the optimal solution since the exhaustive search might be extremely long. Therefore, one rather relies on suboptimal algorithms which run in polynomial time and return a solution (hopefully) close to the optimal. Such algorithms include greedy algorithms, convex and non-convex solvers. In this section we make an overview of the three categories with a focus on greedy algorithms and their non-negative extensions. We therefore choose to start the section with a review of greedy algorithms, then go towards non-negative greedy algorithms, then quickly present convex solvers and end up with a short discussion about non-convex solvers.

1.3.1 Greedy algorithms

Greedy algorithms are suboptimal algorithms acting on the ℓ_0-norm of solution. Originally, greedy algorithms start from the zero vector (sparsest solution) which corresponds to an empty support, and gradually change the cardinality of the support one-by-one through iterations, hence the name "greedy". According to this formulation, greedy algorithms can be interpreted as descent algorithms (the residual norm $\|\boldsymbol{y}-H\boldsymbol{x}\|$ is decreasing at each iteration) especially suited for reconstructing highly sparse solutions. In the literature, the name "greedy" also refers to another class of algorithms in which a support of fixed cardinality is produced at each iteration. Unlike the previous strategies, these algorithms are not always descent algorithms and they require some prior knowledge about the cardinality of the solution support. We next discuss some typical greedy algorithms of both strategies. These algorithms and the corresponding problem they address are shown in Table 1.1. Note that in the rest of the thesis we use the term "greedy" to refer to the one-by-one updating strategy only.

Problem	Algorithms
$(\ell_0 C_\epsilon)$	MP, OMP, OLS
$(\ell_0 C_K)$	MP, OMP, OLS, CoSaMP, SP, IHT, HTP
$(\ell_0 P)$	Bayesian MP, Bayesian OMP, SBR, CSBR

Table 1.1: Typical greedy algorithms and the corresponding problems addressed

1.3.1.1 One-by-one updating strategy

Greedy algorithms increasing the support cardinality one-by-one at each iteration contain Matching Pursuit (MP) [Mallat and Zhang, 1993], Orthogonal Matching Pursuit (OMP) [Pati et al., 1993], and Orthogonal Least Squares (OLS) [Chen et al., 1989][3], in increasing order of complexity. The greedy mechanism consists of two steps: (i) new atom selection ($\boldsymbol{h}_\ell$) yielding a new support $S \leftarrow S \cup \{\ell\}$, and (ii) update of the sparse coefficients $\boldsymbol{x}_S$. MP and OMP share the way of choosing the new atom by maximizing the inner product of the current residual and dictionary atoms

$$\ell \in \arg\max_i |\boldsymbol{r}^{\mathrm{t}}\boldsymbol{h}_i|. \tag{1.1}$$

However, they differ in the way the coefficients are updated. While MP only updates the coefficient related to the new selected atom x_ℓ, OMP updates all the coefficients x_i, $i \in S$ by

[3] In the literature, OLS is also known as Order Recursive Matching Pursuit (ORMP) [Cotter et al., 1999], Optimized Orthogonal Matching Pursuit (OOMP) [Rebollo-Neira and Lowe, 2002], and Pure Orthogonal Matching Pursuit [Foucart, 2013].

performing an orthogonal projection (hence the name "orthogonal"). As a result, the residual $\boldsymbol{r} = \boldsymbol{y} - H_S \boldsymbol{x}(S)$ is orthogonal to the subspace spanned by selected atoms span (H_S) (see Fig. 1.6). Note that the orthogonal projection $\boldsymbol{x}(S) \leftarrow H_S^\dagger \boldsymbol{y}$ is equivalent to solve the Unconstrained

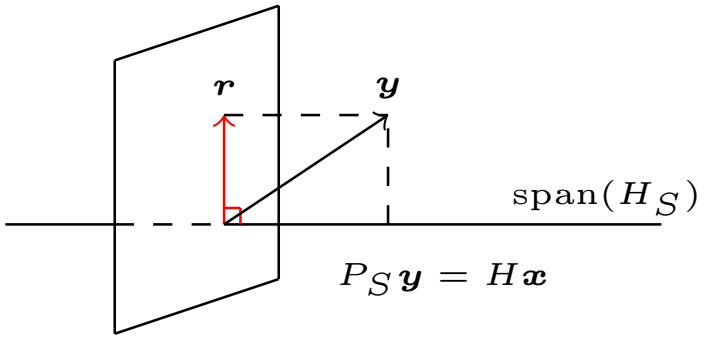


Figure 1.6: Orthogonal projection of $\boldsymbol{y}$ onto span (H_S) and the resulting residual $\boldsymbol{r}$.

Least Squares (ULS) subproblem

$$\boldsymbol{x}(S) \leftarrow \arg\min_{\boldsymbol{z}} \ \|\boldsymbol{y} - H_S \boldsymbol{z}\|^2. \tag{1.2}$$

MP iteration is very fast but the number of MP iterations to reach a given residual value ϵ might be significantly larger than that of OMP. In addition, OMP is empirically more effective than MP [Pati et al., 1993]. OLS updates the coefficients the same way as OMP but differs in atom selection. While OMP selects the atom having the highest inner product (in magnitude) with the current residual, OLS considers the inner product with the projection of atoms onto the orthogonal complement of selected subspace after normalization (the normalized projected atoms). This is illustrated in Algorithms 2-3 in which S denotes the support of the current solution and $\tilde{\boldsymbol{g}}_i^S$ denotes the normalized projected atom, *i.e.*,

$$\tilde{\boldsymbol{h}}_i^S = P_S^\perp \boldsymbol{h}_i, \quad \tilde{\boldsymbol{g}}_i^S = \frac{\tilde{\boldsymbol{h}}_i^S}{\|\tilde{\boldsymbol{h}}_i^S\|}. \tag{1.3}$$

Algorithm 2: OMP [Pati et al., 1993]

Input: $\boldsymbol{y}$, H, K
Output: $\boldsymbol{x}$ solving $(\ell_0 C_K)$
1 Initialization: $\boldsymbol{x} \leftarrow \boldsymbol{0}$; $S \leftarrow \emptyset$; $\boldsymbol{r} \leftarrow \boldsymbol{y}$;
2 **while** $|S| < K$ **do**
3 $\quad \ell \in \arg\max_{i \notin S} |\boldsymbol{r}^{\mathrm{t}} \boldsymbol{h}_i|$;
4 $\quad S \leftarrow S \cup \{\ell\}$;
5 $\quad \boldsymbol{x}(S) \leftarrow H_S^\dagger \boldsymbol{y}$;
6 $\quad \boldsymbol{r} \leftarrow \boldsymbol{y} - H_S \boldsymbol{x}(S)$;
7 **end**

Algorithm 3: OLS [Chen et al., 1989]

Input: $\boldsymbol{y}$, H, K
Output: $\boldsymbol{x}$ solving $(\ell_0 C_K)$
1 Initialization: $\boldsymbol{x} \leftarrow \boldsymbol{0}$; $S \leftarrow \emptyset$; $\boldsymbol{r} \leftarrow \boldsymbol{y}$;
2 **while** $|S| < K$ **do**
3 $\quad \ell \in \arg\max_{i \notin S} |\boldsymbol{r}^{\mathrm{t}} \tilde{\boldsymbol{g}}_i^S|$;
4 $\quad S \leftarrow S \cup \{\ell\}$;
5 $\quad \boldsymbol{x}(S) \leftarrow H_S^\dagger \boldsymbol{y}$;
6 $\quad \boldsymbol{r} \leftarrow \boldsymbol{y} - H_S \boldsymbol{x}(S)$;
7 **end**

Note that from an optimization viewpoint, the selection rule of OMP (line 3 of Algorithm 2) is equivalent to

$$\ell \in \arg\min_{i \notin S} \left(\min_z \|\boldsymbol{y} - H_S \boldsymbol{x}(S) - z \boldsymbol{h}_i\|^2 \right). \tag{1.4}$$

Indeed, $\min_z \|\boldsymbol{y} - H_S\boldsymbol{x}(S) - z\boldsymbol{h}_i\|^2 = \min_z \|\boldsymbol{r} - z\boldsymbol{h}_i\|^2 = \|\boldsymbol{r}\|^2 - |\boldsymbol{r}^{\mathrm{t}}\boldsymbol{h}_i|^2$ as $\|\boldsymbol{h}_i\| = 1$. Besides, OLS selection rule (line 3 of Algorithm 3) is equivalent to [Blumensath and Davies, 2007]

$$\ell \in \arg\min_{i \notin S} \left(\min_{\boldsymbol{z}} \|\boldsymbol{y} - H_{S\cup\{i\}}\boldsymbol{z}\|^2 \right). \tag{1.5}$$

From this point of view, it is obvious that the OLS selection rule is more costly than that of OMP, although recursive[4] update schemes are available, see, *e.g.,* [Chen et al., 1989, Rebollo-Neira and Lowe, 2002]. In this thesis we are particularly interested in orthogonal greedy algorithms such as OMP and OLS since they are fast in comparison with non-greedy solvers and empirically effective.

Some greedy algorithms change the support cardinality one-by-one at each iteration but in both forward and backward directions [Miller, 2002]. This means these algorithms either add or remove one atom at each iteration. Examples include Bayesian MP (BMP), Bayesian OMP (BOMP) [Herzet and Drémeau, 2010], Single Best Replacement (SBR) [Soussen et al., 2011] and CSBR [Soussen et al., 2015]. BMP, BOMP, SBR are forward-backward extensions of MP, OMP, OLS, respectively. They are designed as descent algorithms dedicated to the penalized ℓ_0 minimization problem ($\ell_0 P$) with a particular value of the parameter λ. CSBR is the "continuous" version of SBR which solves ($\ell_0 P$) for a continuum of values of λ. Like MP, BMP is fast but the three algorithms BOMP, SBR and CSBR are empirically more effective [Herzet and Drémeau, 2010, Soussen et al., 2011]. Like OMP and OLS, the forward-backward extensions BOMP, SBR and CSBR can be efficiently implemented using recursive updating schemes [Sturm and Christensen, 2012, Soussen et al., 2011].

1.3.1.2 Fixed-cardinality updating strategy

Instead of changing the support size one-by-one, some algorithms produce a support of fixed-cardinality at every iteration. Such algorithms are specifically designed to address ($\ell_0 C_K$). They include CoSaMP [Needell and Tropp, 2009], Subspace Pursuit (SP) [Dai and Milenkovic, 2009], Iterative Hard Thresholding (IHT) [Blumensath and Davies, 2008], and Hard Thresholding Pursuit (HTP) [Foucart, 2011], to name but a few. CoSaMP and SP are very similar, as well as IHT and HTP. We hence choose to present them closely for ease of comparison (see Algorithms 4-7).

At each CoSaMP iteration, at most $2K$ atoms (reselection might occur) are selected to enter the support, then the coefficients are updated by performing an orthogonal projection followed by a thresholding. Similarly, SP selects at most K atoms to enter the support and updates the coefficients by performing an orthogonal projection followed by a thresholding and an additional orthogonal projection. Note that CoSaMP performs one orthogonal projection related to a support of size around $3K$ at each iteration while SP performs two orthogonal projections related to support sizes around $2K$ and K. In the literature, CoSaMP and SP are known to have very competitive performances [Needell and Tropp, 2009, Dai and Milenkovic, 2009].

IHT's principle is to repeat a gradient descent move followed by a thresholding [Blumensath and Davies, 2008]. Therefore, IHT is very fast at each iteration but it might need a large number of iterations to converge. To improve the convergence speed, more effective step search strategies were proposed later [Blumensath and Davies, 2010]. Hereafter, we use the name IHT to refer to the later scheme (presented in Algorithm 6). It should be noted that a version of IHT dedicated to ($\ell_0 P$) was also proposed but it was reported to be less effective than Algorithm 6 [Blumensath and Davies, 2008].

HTP [Foucart, 2011] is a variant of IHT in which an orthogonal projection is performed to update the coefficients at each iteration. Note that the orthogonal prjection at line 12 of

[4] in the sense that the current iterate is used to accelerate the computation of the next iterate.

Algorithm 7 is computed approximately using a certain number of gradient descent iterations (otherwise $\alpha = 0$ at the next iteration). Each iteration of HTP is more costly than that of IHT. However, HTP is empirically more effective [Foucart, 2011].

Since CoSaMP, SP and HTP may change many of the support elements at each iteration, they are difficult to lend themselves to recursive implementations. Besides, their coefficient update is done by orthogonal projection which might be expensive without recursive implementation. Therefore, their coefficient update step is often implemented approximately (see, *e.g.,* [Needell and Tropp, 2009]). Finally, it is worth emphasizing that, among greedy algorithms performing orthogonal projection at every stage, OMP and OLS can address both ($\ell_0 C_\epsilon$) and ($\ell_0 C_K$) with a slight modification in their stopping rules (replacing $|S| < K$ by $\|\boldsymbol{r}\| > \epsilon$). On the contrary, CoSaMP, SP, and HTP are restricted to ($\ell_0 C_K$) since they require the sparsity level K as an input argument.

Algorithm 4: CoSaMP [Needell and Tropp, 2009]

Input: $\boldsymbol{y}$, H, K
Output: $\boldsymbol{x}$ solving ($\ell_0 C_K$)
Initialization: $\boldsymbol{x} \leftarrow \boldsymbol{0}$; $S \leftarrow \emptyset$; $\boldsymbol{r} \leftarrow \boldsymbol{y}$;
while *not stop* **do**
  $T \leftarrow$ set of indices of the $2K$ largest magnitudes of $|H^{\mathrm{t}}\boldsymbol{r}|$;
  $S \leftarrow S \cup T$;
  $\boldsymbol{x}(S) \leftarrow H_S^{\dagger}\boldsymbol{y}$;
  $S \leftarrow$ set of indices of the K largest magnitudes of $\boldsymbol{x}$;
  Set $x_i = 0$ for all $i \notin S$;
  $\boldsymbol{r} \leftarrow \boldsymbol{y} - H_S\boldsymbol{x}(S)$;
end
;

Algorithm 5: SP [Dai and Milenkovic, 2009]

Input: $\boldsymbol{y}$, H, K
Output: $\boldsymbol{x}$ solving ($\ell_0 C_K$)
Initialization: $\boldsymbol{x} \leftarrow \boldsymbol{0}$; $S \leftarrow \emptyset$; $\boldsymbol{r} \leftarrow \boldsymbol{y}$;
while *not stop* **do**
  $T \leftarrow$ set of indices of the K largest magnitudes of $|H^{\mathrm{t}}\boldsymbol{r}|$;
  $S \leftarrow S \cup T$;
  $\boldsymbol{x}(S) \leftarrow H_S^{\dagger}\boldsymbol{y}$;
  $S \leftarrow$ set of indices of the K largest magnitudes of $\boldsymbol{x}$;
  Set $x_i = 0$ for all $i \notin S$;
  $\boldsymbol{x}(S) \leftarrow H_S^{\dagger}\boldsymbol{y}$;
  $\boldsymbol{r} \leftarrow \boldsymbol{y} - H_S\boldsymbol{x}(S)$;
end

Algorithm 6: IHT [Blumensath and Davies, 2010]

Input: $\boldsymbol{y}$, H, K
Output: $\boldsymbol{x}$ solving ($\ell_0 C_K$)
Initialization: $\boldsymbol{x} \leftarrow \boldsymbol{0}$; $S \leftarrow \emptyset$; $\boldsymbol{r} \leftarrow \boldsymbol{y}$;
while *not stop* **do**
  if $S = \emptyset$ **then**
    $\alpha = 1$
  else
    $\alpha \leftarrow \dfrac{\|H_S^{\mathrm{t}}\boldsymbol{r}\|^2}{\|H_S H_S^{\mathrm{t}}\boldsymbol{r}\|^2}$
  end
  $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha H^{\mathrm{t}}\boldsymbol{r}$;
  $S \leftarrow$ set of indices of the K largest magnitudes of $\boldsymbol{x}$;
  Set $x_i = 0$ for all $i \notin S$;
  $\boldsymbol{r} \leftarrow \boldsymbol{y} - H_S\boldsymbol{x}(S)$;
end

Algorithm 7: HTP [Foucart, 2011]

Input: $\boldsymbol{y}$, H, K
Output: $\boldsymbol{x}$ solving ($\ell_0 C_K$)
Initialization: $\boldsymbol{x} \leftarrow \boldsymbol{0}$; $S \leftarrow \emptyset$; $\boldsymbol{r} \leftarrow \boldsymbol{y}$;
while *not stop* **do**
  if $S = \emptyset$ **then**
    $\alpha = 1$
  else
    $\alpha \leftarrow \dfrac{\|H_S^{\mathrm{t}}\boldsymbol{r}\|^2}{\|H_S H_S^{\mathrm{t}}\boldsymbol{r}\|^2}$
  end
  $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha H^{\mathrm{t}}\boldsymbol{r}$;
  $S \leftarrow$ set of indices of the K largest magnitudes of $\boldsymbol{x}$;
  Set $x_i = 0$ for all $i \notin S$;
  $\boldsymbol{x}(S) \leftarrow H_S^{\dagger}\boldsymbol{y}$;
  $\boldsymbol{r} \leftarrow \boldsymbol{y} - H_S\boldsymbol{x}(S)$;
end

1.3.2 Non-negative greedy algorithms

Some greedy algorithms such as MP and IHT, due to the simplicity of their iterations, can be turned to non-negative setting quite simply. However, it is not the case for orthogonal greedy algorithms such as OMP and OLS. Taking non-negativity into account within OMP and OLS can raise some non-trivial issues since the structure of the algorithm is deeply modified and, more importantly, proposing efficient implementation is not obvious. Let us now explain the issues and how they are addressed in the literature.

Non-negative extensions of OMP The first non-negative extension of OMP was introduced by Bruckstein *et al.* under the name Non-negative OMP (NNOMP) [Bruckstein et al., 2008]. This extension naturally considers only positive inner products in the selection step (see Algorithm 8). In addition, the coefficient update is done by solving a Non-Negative Least Squares (NNLS) subproblem

$$\boldsymbol{x}(S) \leftarrow \underset{\boldsymbol{z}\geq \mathbf{0}}{\arg\min} \ \|\boldsymbol{y} - H_S \boldsymbol{z}\|^2 \tag{1.6}$$

instead of the ULS subproblem (1.2). Since the NNLS solution might differ from the ULS solution when some non-negativity constraints are activated, the orthogonality between the residual vector and the selected subspace might not hold anymore. Besides, unlike the ULS problem, NNLS does not have a closed-form solution, so one needs to run an iterative subroutine to compute the NNLS solution at each iteration. As a result, NNOMP and other non-negative versions of OMP and OLS are expected to be much more expensive than OMP and OLS [Bruckstein et al., 2008, Yaghoobi et al., 2015, Yaghoobi and Davies, 2015]. The same argument applies to the non-negative extensions of CoSaMP, SP and HTP. However, the non-negative extensions of OMP and OLS can still have fully recursive implementation as explained in Chapter 2 while it seems not to be the case for the non-negative extensions of CoSaMP, SP and HTP.

The fast implementation of NNOMP was not addressed in [Bruckstein et al., 2008] but later in [Yaghoobi et al., 2015, Ramamurthy et al., 2014, Kim and Haldar, 2016, Wang et al., 2018]. Yaghoobi *et al.* [Yaghoobi et al., 2015] proposed a scheme named Fast NNOMP (FNNOMP) which combines the selection and the coefficient update steps in one so that the residual vector is orthogonal to the selected subspace at every iteration. Thus, FNNOMP does not require to solve any NNLS subproblem but it might return a different output from that of NNOMP. In other words, FNNOMP can be seen as a variant of NNOMP without NNLS solving. Another variant of NNOMP was proposed by Wang *et al.* [Wang et al., 2018]. This variant, named NN-OMP, selects the new atom the same way as OMP but updates the coefficients by solving a NNLS subproblem. Thus NN-OMP and NNOMP have roughly the same costs but likely different outputs.

Non-negative extensions of OLS [Yaghoobi and Davies, 2015] firstly introduced a non-negative extension of OLS named Non-Negative OLS (NNOLS). This version (presented in Algorithm 10) replaces all the orthogonal projections (1.5) by NNLS subproblems

$$\ell \in \underset{i\notin S}{\arg\min} \left(\min_{\boldsymbol{z}\geq \mathbf{0}} \ \|\boldsymbol{y} - H_{S\cup\{i\}} \boldsymbol{z}\|^2 \right). \tag{1.7}$$

As a result, NNOLS requires to solve as many NNLS subproblems as the number of non-selected atoms at each iteration, hence it was considered to be too expensive [Yaghoobi and Davies, 2015]. Yaghoobi and Davies proposed two variants of NNOLS namely Suboptimal NNOLS (SNNOLS) and Fast NNOLS (FNNOLS) [Yaghoobi and Davies, 2015]. SNNOLS (presented in Algorithm 9) is based on OLS selection rule (see Algorithm 3) but it only considers positive inner products. Besides, SNNOLS requires to solve only one NNLS subproblem related to the augmented support

at each iteration. FNNOLS was inspired from SNNOLS the same way as FNNOMP was inspired from NNOMP. The two approximate schemes FNNOMP and FNNOLS are not presented here because we have examples showing that FNNOMP and FNNOLS do not always respect the positivity constraint [Nguyen et al., 2017].

NNOMP and SNNOLS (Algorithms 8-9) can be seen as intuitive extensions to the non-negative setting of OMP and OLS (Algorithms 2-3), respectively. Besides, NNOLS is a direct non-negative extension of OLS in optimization viewpoint. Indeed, NNOLS is designed as a descent algorithm aiming to decrease at most the residual using a single change in the support. It is noteworthy that, while the two interpretations of OLS ((1.5) and line 3-Algorithm 3) are equivalent [Blumensath and Davies, 2007], NNOLS and SNNOLS significantly differ since (1.5) and line 3-Algorithm 3 are no more equivalent in the non-negative setting. Indeed, the SNNOLS selection rule (line 3-Algorithm 9) can be interpreted as

$$\ell \in \arg\min_{i \notin S} \left(\min_{\boldsymbol{u}, v \geq 0} \|\boldsymbol{y} - H_S \boldsymbol{u} - v \boldsymbol{h}_i\|^2 \right)$$

while (1.7) can be rewritten as

$$\ell \in \arg\min_{i \notin S} \left(\min_{\boldsymbol{u} \geq \boldsymbol{0}, v \geq 0} \|\boldsymbol{y} - H_S \boldsymbol{u} - v \boldsymbol{h}_i\|^2 \right)$$

(see Chapter 2 for more details). From this point of view, we can see that SNNOLS selection rule is less costly than that of NNOLS. Finally, the three algorithms NNOLS, SNNOLS and NNOMP can address either ($\ell_0 C_\epsilon+$) or ($\ell_0 C_K+$) by a slight modification in the stopping rule (replacing $|S| < K$ by $\|\boldsymbol{r}\| > \epsilon$).

Algorithm 8: NNOMP [Bruckstein et al., 2008]

Input: $\boldsymbol{y}$, H, K
Output: $\boldsymbol{x}$ solving ($\ell_0 C_K+$)

Initialization: $\boldsymbol{x} \leftarrow \boldsymbol{0}$; $S \leftarrow \emptyset$; $\boldsymbol{r} \leftarrow \boldsymbol{y}$;
while $|S| < K$ *and* $\max_{i \notin S} \boldsymbol{r}^{\mathrm{t}} \boldsymbol{h}_i > 0$ **do**
  $\ell \in \arg\max_{i \notin S} \boldsymbol{r}^{\mathrm{t}} \boldsymbol{h}_i$;
  $S \leftarrow S \cup \{\ell\}$;
  $\boldsymbol{x}(S) \leftarrow \arg\min_{\boldsymbol{z} \geq \boldsymbol{0}} \|\boldsymbol{y} - H_S \boldsymbol{z}\|^2$;
  $\boldsymbol{r} \leftarrow \boldsymbol{y} - H_S \boldsymbol{x}(S)$;
end

Algorithm 9: SNNOLS [Yaghoobi and Davies, 2015]

Input: $\boldsymbol{y}$, H, K
Output: $\boldsymbol{x}$ solving ($\ell_0 C_K+$)

Initialization: $\boldsymbol{x} \leftarrow \boldsymbol{0}$; $S \leftarrow \emptyset$; $\boldsymbol{r} \leftarrow \boldsymbol{y}$;
while $|S| < K$ *and* $\max_{i \notin S} \boldsymbol{r}^{\mathrm{t}} \tilde{\boldsymbol{g}}_i^S > 0$ **do**
  $\ell \in \arg\max_{i \notin S} \boldsymbol{r}^{\mathrm{t}} \tilde{\boldsymbol{g}}_i^S$;
  $S \leftarrow S \cup \{\ell\}$;
  $\boldsymbol{x}(S) \leftarrow \arg\min_{\boldsymbol{z} \geq \boldsymbol{0}} \|\boldsymbol{y} - H_S \boldsymbol{z}\|^2$;
  $\boldsymbol{r} \leftarrow \boldsymbol{y} - H_S \boldsymbol{x}(S)$;
end

Algorithm 10: NNOLS [Yaghoobi and Davies, 2015]

Input: $\boldsymbol{y}$, H, K
Output: $\boldsymbol{x}$ solving ($\ell_0 C_K+$)

Initialization: $\boldsymbol{x} \leftarrow \boldsymbol{0}$; $S \leftarrow \emptyset$; $\boldsymbol{r} \leftarrow \boldsymbol{y}$;
while $|S| < K$ **do**
  $\ell \in \arg\min_{i \notin S} \left(\min_{\boldsymbol{z} \geq \boldsymbol{0}} \|\boldsymbol{y} - H_{S \cup \{i\}} \boldsymbol{z}\|^2 \right)$;
  $S \leftarrow S \cup \{\ell\}$;
  $\boldsymbol{x}(S) \leftarrow \arg\min_{\boldsymbol{z} \geq \boldsymbol{0}} \|\boldsymbol{y} - H_S \boldsymbol{z}\|^2$;
end

Non-negative extensions of CoSaMP, SP, and HTP CoSaMP, SP, and HTP were extended to the non-negative setting in [Kim and Haldar, 2016]. These extensions (named NNCoSaMP,

NNSP and NNHTP, respectively) consider only the positive inner products in atom selection and replace the orthogonal projection by NNLS solving, just as the way NNOMP [Bruckstein et al., 2008] was designed. NNCoSaMP is displayed in Algorithm 11 for an illustration. It is noteworthy that, while CoSaMP, SP and HTP produce an exactly K-sparse solution at each iteration, their non-negative extension do not. This is because the NNLS solution differs from the ULS solution (and is sparser) when some non-negative constraints are activated. More importantly, there is no obvious way to enforce the K-sparseness of the output of NNCoSaMP, NNSP and NNHTP. This is one drawback when using these algorithms in practice. In addition, NNCoSaMP, NNSP and NNHTP are restricted to address ($\ell_0 C_K+$) as they requires the prior knowledge on the sparsity level K of the solution. Finally, these algorithms are hard to lend themselves to recursive implementation since several support elements are changed at each iteration.

Algorithm 11: Non-Negative CoSaMP (NNCoSaMP) [Kim and Haldar, 2016]

Input: $\boldsymbol{y}$, H, K
Output: $\boldsymbol{x}$ solving ($\ell_0 C_K+$)
Initialization: $\boldsymbol{x} \leftarrow \mathbf{0}$; $S \leftarrow \emptyset$; $\boldsymbol{r} \leftarrow \boldsymbol{y}$;
while *not stop* **do**
  $T \leftarrow$ set of indices of the $2K$ largest positive entries of $H^{\mathrm{t}}\boldsymbol{r}$;
  $S \leftarrow S \cup T$;
  $\boldsymbol{x}(S) \leftarrow \arg\min_{\boldsymbol{z}\geq \mathbf{0}} \|\boldsymbol{y} - H_S \boldsymbol{z}\|^2$;
  $S \leftarrow$ set of indices of the K largest magnitude of $\boldsymbol{x}$;
  Set $x_i = 0$ for all $i \notin S$;
  $\boldsymbol{r} \leftarrow \boldsymbol{y} - H_S \boldsymbol{x}(S)$;
end

1.3.3 Convex solvers

Since ℓ_0 minimization is NP-hard and several local minimizers may exist, an alternative approach is to replace ℓ_0 by ℓ_1 to obtain a convex optimization problem which is easier to solve (and there exist fast solvers) and has a single minimizer. However, the ℓ_1 solution might be very far from the global minimizer of the ℓ_0 problem (except for easy problems where $\ell_0 - \ell_1$ equivalence holds). ℓ_1 minimization problem which is often referred to as LASSO [Tibshirani, 1996, Friedman et al., 2007] or Basic Pursuit [Chen et al., 2001, van den Berg and Friedlander, 2009] can take one of the following forms

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{x}\|_1 \quad \text{s.t.} \quad \|\boldsymbol{y} - H\boldsymbol{x}\|_2 \leq \epsilon \tag{$\ell_1 C_\epsilon$}$$

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{y} - H\boldsymbol{x}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{x}\|_1 \leq K \tag{$\ell_1 C_K$}$$

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{y} - H\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1 \tag{$\ell_1 P$}$$

Note that the three ℓ_1 minimization problems are equivalent in the sense that we can find a value of λ so that ($\ell_1 P$) solution identifies with that of ($\ell_1 C_\epsilon$) or ($\ell_1 C_K$). Moreover, it was shown that when λ tends to zero, the solution of ($\ell_1 P$) converges to that of ($\ell_1 C_\epsilon$) for $\epsilon = 0$ [Donoho and Tsaig, 2008]. ℓ_1 minimization problems can be solved by many methods such as interior point, gradient projection, homotopy, Lagrange multiplier, etc. Popular ℓ_1 solvers include ADMM [Boyd et al., 2011], FISTA [Beck and Teboulle, 2009], LARS [Efron et al., 2004] and Homotopy [Donoho and Tsaig, 2008]. Originally, LARS [Efron et al., 2004] was designed as a forward algorithm. LARS with the LASSO modification [Efron et al., 2004] is forward-backward and it is identical to Homotopy. While ADMM an FISTA only solve ($\ell_0 P$) approximately for a particular value of

λ, LARS and Homotopy solve it exactly under mild assumptions and produce the entire solution path for all values of λ (for undercomplete case in [Donoho and Tsaig, 2008] and overcomplete case in [Efron et al., 2004]). Interestingly, both Homotopy and LARS use greedy strategies and they are shown to be structurally close to OMP [Donoho and Tsaig, 2008]. It is noteworthy that since ℓ_1 solvers tend to produce biased coefficents [Fan and Li, 2001], usually a post-processing step is done to debias amplitudes

$$S \leftarrow \text{supp}\,(\boldsymbol{x}), \quad \boldsymbol{x}(S) \leftarrow H_S^{\dagger}\boldsymbol{y}. \tag{1.8}$$

Addressing a well-defined convex problem, ℓ_1 solvers can be extended to the non-negative setting without many difficulties. ADMM [Barbu and Herzet, 2016] and Non-negative LARS (NLARS) [Efron et al., 2004, Morup et al., 2008] are typical examples among others. Note that NLARS is the non-negative extension of modified LARS (*i.e.*, non-negative extension of Homotopy). Since we are interested in greedy structure and non-negativity, NLARS is worth to be commented. NLARS returns the solution path of non-negative LASSO

$$\min_{\boldsymbol{x}\geq\boldsymbol{0}} \quad \|\boldsymbol{y}-H\boldsymbol{x}\|_2^2+\lambda\|\boldsymbol{x}\|_1 \tag{ℓ_1P+}$$

for all values of λ. It is notable that (ℓ_1P+) is equivalent to

$$\min_{\boldsymbol{x}\geq\boldsymbol{0}} \quad \|\boldsymbol{y}-H\boldsymbol{x}\|_2^2+\lambda\boldsymbol{1}^{\mathrm{t}}\boldsymbol{x}$$

which is a standard quadratic program for a certain value of λ hence it can also be solved by active-set method [Nocedal and Wright, 2006]. The main idea of NLARS is to maintain ℓ_1 optimal condition when λ is changing. Each iteration consists of either an addition or a removal of one atom from the support and the update of sparse approximation $H\boldsymbol{x}$ along the direction equiangular to the selected atoms. The main structural difference with LARS is that NLARS considers only the positive inner products $\boldsymbol{r}^{\mathrm{t}}\boldsymbol{h}_i$ in the atom selection. In terms of complexity, NLARS requires the update of the inverse Gram matrix $(H_S^tH_S)^{-1}$ at each iteration but it does not perform orthogonal projection $H_S^{\dagger}\boldsymbol{y}$. Finally, NLARS can also be turned to obtain a K-sparse solution (see Matlab code accompanying to the paper [Morup et al., 2008]).

1.3.4 Non-convex solvers

Since ℓ_1 solvers tend to produce less sparse solutions and biased estimates for large coefficients [Fan and Li, 2001], another approach is to replace ℓ_0 by a non-convex separable function which is singular at the origin to promote sparsity and has vanished derivative for large values to be unbiased. Such non-convex functions include the smoothly clipped absolute deviation (SCAD) [Fan and Li, 2001], the ℓ_p pseudonorm ($\|\boldsymbol{x}\|_p^p$ with $0<p<1$) [Frank and Friedman, 1993] and logarithm $\sum_i \log(|x_i|+\eta)$ [Candes et al., 2008] among others. The SCAD function is piecewise smooth. It is defined in 1-D by clipping three 1-D functions with separable regularization. Intuitively, it looks like ℓ_1 norm around the origin and flat at large values. The ℓ_p pseudonorm (with $0<p<1$) and logarithm are quasi-smooth approximations of the ℓ_0 norm. To minimize the non-convex objective function, a usual technique is to locally approximate it by a convex (linear or quadratic) function then use one or many iterations of proximal gradient methods or Newton's method to solve the resulting convex optimization problem [Zou and Li, 2008, Huang et al., 2008, Saab et al., 2008]. As a result, non-convex solvers often suffer from a rather high computational complexity compared with ℓ_1 solvers and greedy algorithms. Another technique is to solve the non-convex minimization problem through a sequence of weighted ℓ_1 minimization problems [Candes et al., 2008]. This is a special case of DC programming [Thoai, 1988, Le Thi et al., 2013]. In the literature, non-negative extension of non-convex solvers is quite rare.

1.4 State-of-the-art exact recovery analyses of greedy algorithms

1.4.1 Exact recovery with greedy algorithms

OMP is guaranteed to recover the K-sparse representation of a data signal in the noiseless case if the mutual coherence

$$\mu = \max_{i \neq j} |\boldsymbol{h}_i^{\mathrm{t}} \boldsymbol{h}_j|$$

of the dictionary H satisfies [Tropp, 2004, Herzet et al., 2013]

$$\mu < \frac{1}{2K-1}. \tag{1.9}$$

This condition was proved to be sharp [Cai et al., 2010] and extensions to noisy cases were considered [Ben-Haim et al., 2010, Cai and Wang, 2011]. Besides, relaxations of (1.9) given partial information on the decomposition of the data signal were proposed [Herzet et al., 2013, Herzet et al., 2016]. Other exact recovery analyses of OMP also consider ERC condition [Tropp, 2004]

$$\max_{i \notin S^*} \|H_{S^*}^{\dagger} \boldsymbol{h}_i\|_1 < 1 \tag{1.10}$$

(where S^* presents the true support of the K-sparse representation) or the Restricted Isometry Property (RIP) condition of order K [Davenport and Wakin, 2010, Satpathi et al., 2013, Li et al., 2015, Wen et al., 2017b]

$$(1-\delta)\|\boldsymbol{x}\|^2 \leq \|H\boldsymbol{x}\|^2 \leq (1+\delta)\|\boldsymbol{x}\|^2 \tag{1.11}$$

for all K-sparse vectors $\boldsymbol{x}$ (where δ presents the RIP constant).

While the mutual coherence based condition (1.9) is intuitive and its computation is feasible, it is stronger than (*i.e.*, it implies) the ERC condition (1.10) [Tropp, 2004] and might never hold for very large size dictionaries [Foucart and Rauhut, 2013]. On the contrary, evaluating ERC condition (1.10) requires the knowledge of the true support S^* which is often unavailable and evaluating RIP condition (1.11) is computationally intractable in general [Tillmann, 2015].

Exact support recovery properties of OLS have been studied more recently. In [Herzet et al., 2013], the mutual coherence based condition (1.9) was extended to OLS in the noiseless case, while [Soussen et al., 2013] considered ERC condition and [Wen et al., 2017a, Wang and Li, 2017] used RIP assumptions.

Exact recovery analyses of CoSaMP, SP, IHT and HTP are all based on RIP condition [Needell and Tropp, 2009, Dai and Milenkovic, 2009, Blumensath and Davies, 2008, Foucart, 2011].

1.4.2 Exact recovery with non-negative greedy algorithms

It was stated in [Bruckstein et al., 2008, Theorem 3] that the mutual coherence based condition (1.9) is a sufficient condition for exact recovery of any K-sparse representation using NNOMP [Bruckstein et al., 2008] in the noiseless setting. However, the proof is not provided. Besides, Kim *et al.* generalized (1.9) for the case of multiple measurement vector (*i.e.*, data $\boldsymbol{y}$ is a matrix instead of a vector) in the noiseless setting but the authors acknowledged that their condition turns out to be very restrictive [Kim and Haldar, 2016].

In the literature, the exact recovery analysis of NNOMP in noisy cases has not been addressed yet. The theoretical guarantees of other non-negative greedy algorithms are still missing too.

1.5 Contributions

Targeting ℓ_0 minimization, we are interested in greedy algorithms since their structure is simple and fast implementation is available (for standard greedy algorithms). On the contrary, the

ℓ_1 global minimizer may be far from the ℓ_0 global minimizer and non-convex solvers are with rather high computational complexity. Orthogonal greedy algorithms such as OMP and OLS often have good performance in sparse reconstruction. Moreover, they can be easily adapted to address the constraint related to the sparsity level $\|\boldsymbol{x}\|_0 \leq K$ and/or noise level $\|\boldsymbol{y} - H\boldsymbol{x}\| \leq \epsilon$. It is not the case for the algorithms such as CoSaMP, SP and HTP which are also difficult to lend themselves to recursive implementation. However, the extension of orthogonal greedy algorithms to the non-negative setting is challenging. The issue arises since the orthogonal projection which is equivalent to ULS solving (1.2) is replaced by NNLS solving (1.6). Since the NNLS problem does not have a closed-form solution, solving NNLS is more computationally demanding than ULS solving. In addition, when some non-negativity constraints are activated, the NNLS solution differs from the ULS solution so the orthogonality between the residual vector and the selected subspace does not hold anymore. Addressing these algorithmic issues is our first main contribution.

Besides, the theoretical guarantee of OMP and OLS has always been a fruitful topic. OMP and OLS are guaranteed to have exact support recovery under several types of conditions such as mutual coherence based condition, Tropp's condition ERC, RIP condition, etc. These results strengthen the interest of OMP and OLS and make them more popular. However, equivalent results for the non-negative extensions of OMP and OLS are rare and limited. Hence our theoretical study of non-negative orthogonal greedy algorithms is a second important contribution.

More precisely, our contributions can be summarized in four main points as follows, where the first contribution is split in two: analyzing structural properties (§1.5.1) and proposing fast implementation (§1.5.2) of non-negative greedy algorithms. The second contribution appears in §1.5.3. The last contribution is about NP-hardness of the non-negative ℓ_0 minimization and can be found in §1.5.4.

1.5.1 Structural properties of non-negative greedy algorithms

We revisit the non-negative versions of OMP and OLS in a unified way. We introduce a new class of Non-Negative Orthogonal Greedy (NNOG) algorithms which covers many existing non-negative extensions of OMP and OLS such as NNOMP, NNOLS and SNNOLS, up to some modifications. Moreover, we show that NNOG algorithms share many desirable properties, which are not obvious for existing non-negative extensions of OMP and OLS, such as: the norm of residual is decreasing at each iteration and the residual vector is orthogonal to the selected atoms. This is possible because a so-called *support compression* step is included in NNOG structure to make the support set identifies to the support of the current NNLS solution at each iteration. This contribution is presented in Chapter 2.

1.5.2 Fast implementation

As the NNLS problem does not have a closed-form solution, solving it is computationally demanding. Existing implementations of non-negative greedy algorithms either solve NNLS subproblems independently (hence in an exact but time-consuming way) or avoid NNLS solving (hence yielding fast but approximate schemes). Unlike any of previous contributions, we propose a *fast and exact implementation* of NNOG algorithms. This implementation exploits the active-set algorithm [Lawson and Hanson, 1974] for NNLS solving, and makes use of warm start initialization. Warm start is a popular technique but it is especially efficient when it is combined with active-set NNLS in NNOG implementation. Moreover, we elaborate on recursive implementations of NNOG algorithms and design further types of accelerations. According to our comparisons, the computing time of NNOG algorithms using proposed implementation becomes comparable to that of orthogonal greedy algorithms. This contribution is presented in Chapter 2 with numerical results in Chapter 3.

1.5.3 Theoretical guarantees

In the literature, theoretical analyses of NNOMP are rare and somewhat discordant while the analysis of other NNOG algorithms is missing. In this context, we establish a unified *exact support recovery* analysis for non-negative extensions of OMP and OLS using mutual coherence based condition. This is a consequence of a more fundamental result about OMP and OLS which is named *sign preservation* property. More precisely, we show that under mutual coherence based condition (1.9), (i) OMP and OLS outputs identify with the true support of the K-sparse representation in K iterations, and (ii) the weights of selected atoms have the correct sign at any of the K iterations. As a result, NNOG algorithms recover exactly the true support of the K-sparse representation with non-negative weights in K iterations under the condition (1.9). It should be noted that while the very few existing analyses of NNOMP are made in the noise-free case, our analysis is made in the noisy cases and it also applies to other algorithms such as NNOLS and SNNOLS. This contribution is presented in Chapter 4.

1.5.4 NP-hardness of non-negative ℓ_0 minimization

The final contribution of the thesis is about the NP-hardness of non-negative ℓ_0 minimization problems. The reasoning is the same as that of Natarajan [Natarajan, 1995] and our analysis is a direct extension. However, we discovered that some existing analyses have errors and clarify these issues. Besides, our analysis can apply to more general cases. This contribution is presented in Chapter 5.

1.6 List of publications

Journal paper

Thanh T. Nguyen, Jerome Idier, Charles Soussen, El-Hadi Djermoune. *Non-Negative Orthogonal Greedy Algorithms*. IEEE Transactions on Signal Processing, vol. 67(21), pp. 5643-5658. 2019.

Preprints

Thanh T. Nguyen, Charles Soussen, Jerome Idier, El-Hadi Djermoune. *Sign Preservation Analysis of Orthogonal Greedy Algorithms*. January 2019. hal-01971697.

Conference papers

- **International conference with proceedings**
  - Thanh T. Nguyen, Charles Soussen, Jerome Idier, El-Hadi Djermoune. *NP-hardness of ℓ_0 minimization problems: revision and extension to the non-negative setting*. International Conference on Sampling Theory and Applications, Bordeaux, France. July 2019. hal-02112180.
- **International workshop**
  - Thanh T. Nguyen, Charles Soussen, Jerome Idier, El-Hadi Djermoune. *Exact recovery analysis of non-negative Orthogonal Matching Pursuit*. Signal Processing with Adaptive Sparse Structured Representations (SPARS) workshop, Toulouse, France, July 2019.
- **National conference with proceedings**

– Thanh T. Nguyen, Charles Soussen, Jerome Idier, El-Hadi Djermoune. *Algorithmes gloutons orthognaux pour la reconstruction de signaux parcimonieux positifs*. GRETSI, Lille, France. August 2019. hal-02149677.

– Thanh T. Nguyen, Charles Soussen, Jerome Idier, El-Hadi Djermoune. *An optimized version of non-negative OMP*. GRETSI, Juan-les-pins, France. September 2017. hal-01585253.

Open source software

Matlab implementation of Non-Negative Orthogonal Greedy Algorithms (CodeOcean), supplementary material to the paper: Thanh T. Nguyen, Jerome Idier, Charles Soussen, El-Hadi Djermoune. *Non-Negative Orthogonal Greedy Algorithms*. IEEE Transactions on Signal Processing, vol. 67(21), pp. 5643-5658. 2019.

Presentations

- Ph.D student day at L2S, Gif-sur-Yvette, France, June 28, 2018.
- Workshop on Sparsity and Applications, Bordeaux, France, May 3, 2018.
- Ph.D. student day at CRAN, Nancy, France, January 12, 2018.
- Seminar at L2S, Gif-sur-Yvette, France, December 8, 2017.
- Seminar at LS2N, Nantes, France, October 12, 2017.

Chapter 2

Unified view of non-negative orthogonal greedy algorithms[1]

Contents

In this chapter, we revisit the non-negative versions of OMP and OLS in a unified way. In particular, we introduce a new class of non-negative orthogonal greedy (NNOG) algorithms which covers many existing non-negative versions of OMP and OLS, up to some modifications. We exhibit the structural properties of NNOG algorithms and introduce a fast implementation. The proposed fast implementation of NNOG algorithm will be assessed afterward by a set of numerical results in Chapter 3.

2.1 Introduction

Greedy algorithms for sparse signal reconstruction are very popular iterative schemes. Their principle is to repeatedly *(i)* enrich the sparse support by selecting a new dictionary atom, and then *(ii)* update the sparse approximation coefficients. In orthogonal greedy algorithms, the sparse approximation signal is computed as the orthogonal projection of the data vector onto

[1] This chapter is an adaptation of the paper [Nguyen et al., 2019a]

the subspace spanned by the selected atoms. Therefore, the coefficients can be estimated by solving an Unconstrained Least Squares (ULS) problem. Popular orthogonal greedy algorithms include Orthogonal Matching Pursuit (OMP) [Pati et al., 1993] and Orthogonal Least Squares (OLS) [Chen et al., 1989]. OMP and OLS differ in the way the new atom is selected. In both cases, the atom inducing the largest decrease of the norm of the residual is selected. However, all nonzero atom weights are optimally tuned in OLS whereas only the new atom weight is considered in OMP, which amounts to selecting the atom having the largest inner product with the current residual. The computational complexity of OLS is obviously higher, since the selection rule requires to solve as many ULS problems as the number of candidate atoms. Fortunately, the ULS solutions have a closed-form expression, which can be recursively (fastly) updated when the support is enriched by a new element, see *e.g.,* [Miller, 2002]. Specifically, both OMP and OLS implementations are recursive and make use of matrix factorization, such as Gram-Schmidt orthogonalization, the Cholesky factorization or techniques utilizing the matrix inversion lemma [Sturm and Christensen, 2012].

In many applications, the signal or image of interest is sparse, but also non-negative. In such contexts, a common practice is to regularize the inverse problem in order to favor both sparsity and non-negativity. Some classical sparse algorithms can be straightforwardly adapted to deal with non-negativity constraints. This is the case of proximal splitting algorithms and the Alternating Direction Method of Multipliers (ADMM) for convex optimization [Combettes and Pesquet, 2011, Boyd et al., 2011], and of the DC algorithm (Difference of Convex functions) for nonconvex optimization [Gasso et al., 2009, Le Thi et al., 2013]. On the contrary, the non-negative extension of greedy algorithms is a challenging issue since the unconstrained least-squares subproblems are replaced by non-negative least-squares (NNLS) subproblems which do not have closed-form solutions anymore, so a subroutine solving NNLS is needed. There are different methods for NNLS solving [Björck, 1996] such as active-set [Lawson and Hanson, 1974], interior-point [Wright, 1992], and gradient-projection [Benvenuto et al., 2010]. The latter two families typically require to tune some stopping criteria empirically, resulting in *approximate* resolution of the NNLS problem. Here, we are focusing on active-set methods for NNLS solving since such methods have a greedy structure and exactly solve NNLS problems after a finite number of iterations. Although our focus will be on extensions of orthogonal greedy schemes to the non-negative case, let us mention that several other non-negative sparse methods have been elaborated on the basis of the active-set NNLS algorithm, *e.g.,* hard-thresholded NNLS [Slawski and Hein, 2011, Slawski and Hein, 2013] and Sparse NNLS or Reverse Sparse NNLS [Peharz and Pernkopf, 2012].

Several existing contributions deal with orthogonal greedy algorithms in the non-negative case. Non-Negative OMP (NNOMP) was first proposed by Bruckstein *et al.* [Bruckstein et al., 2008] as a direct generalization of OMP. At each iteration, the atom having the maximum *positive* inner product with the current residual is selected. Contrary to OMP, negative inner products are discarded. Then, the sparse approximation coefficients are updated by solving the NNLS problem related to the augmented subset. The canonical (*i.e.*, non-recursive) NNOMP implementation of Bruckstein *et al.* [Bruckstein et al., 2008] solves NNLS subproblems independently. Later, Yaghoobi *et al.* proposed an accelerated version named Fast Non-Negative OMP (FNNOMP), which avoids solving NNLS subproblems but rather recursively approximates the sought solution using QR matrix factorization [Yaghoobi et al., 2015]. Although FNNOMP is much faster than canonical NNOMP, it is an approximate version likely to deliver a different output. In [Yaghoobi and Davies, 2015], Yaghoobi *et al.* introduced a canonical version of Non-Negative OLS (NNOLS), defined as a direct non-negative extension of OLS. The principle of NNOLS is to select the atom for which the positive residual (the residual corresponding to the NNLS solution) is minimum. This selection rule appears to be time-consuming since one needs to compute as many positive residuals as the number of candidate atoms, *i.e.*, $n - k$ NNLS problems have to

be solved at iteration k, with n being the size of the dictionary. Yaghoobi *et al.* [Yaghoobi and Davies, 2015] further proposed two accelerated versions of NNOLS named Suboptimal NNOLS (SNNOLS) and Fast NNOLS (FNNOLS). SNNOLS [Yaghoobi and Davies, 2015] selects the atom that is positively correlated with the current residual whose projection forms a maximum angle with the residual. Besides, an NNLS problem must be solved at each iteration to update the sparse approximation coefficients. FNNOLS is a recursive implementation in the same spirit as FNNOMP, where no NNLS problem needs to be solved anymore. It shall be noticed that FNNOLS and SNNOLS do not necessarily deliver the same iterates, and that both can be viewed as approximate versions of NNOLS.

Generally speaking, the "orthogonal" denomination of NNOMP and NNOLS is somewhat abusive since when the support set S is updated, the related NNLS solution may not identify with the orthogonal projection of the data vector onto the span of atoms indexed by S, but rather with its projection onto their positive span. Both projected vectors differ as soon as the NNLS solution has zero entries, *i.e.*, when some non-negativity constraints become active. Therefore, in NNOMP, NNOLS and their derived versions [Bruckstein et al., 2008, Yaghoobi et al., 2015, Yaghoobi and Davies, 2015], the support of the sparse vector at the current iteration is a subset of the current support set S (which is expanded at each iteration) and may not identify with it. In turn, more than K iterations may be necessary to reach a truly K-sparse representation.

In this chapter, our contributions are twofold. First, non-negative orthogonal greedy algorithms are revisited in a unified way. The algorithms under study share four desirable properties:

1. The norm of the data residual is always decreasing when a new atom enters the solution support.

2. The algorithm does not stop while additional atom selections would make it decrease, unless an explicit stopping condition is reached.

3. A so-called compression step is included to shrink the support set by removing the atoms having zero coefficients, so the support set identifies to the support of the current NNLS solution.

4. The residual vector is orthogonal to the selected atoms. In other words, the sparse approximation vector identifies with the orthogonal projection of the data vector onto the span of the selected atoms.

These structural properties are exhibited and compared to those of existing non-negative greedy algorithms. The second contribution is a fast and exact implementation of non-negative orthogonal greedy algorithms exploiting the active-set algorithm [Lawson and Hanson, 1974] for NNLS solving, and based on warm start initialization. Moreover, we elaborate on recursive implementations and we design further types of accelerations.

The chapter is organized as follows. Section 2.2 introduces the family of so-called *Non-negative Orthogonal Greedy* (NNOG) algorithms. The different members of the family differ by the selection rule to pick a new atom at each iteration. It includes NNOLS, SNNOLS and NNOLS up to a modification of their structure, namely the compression step mentioned above. In Section 2.3, we propose a fast implementation based on recursivity and on the use of warm starts for solving the NNLS subproblems. Section 2.4 is devoted to NNOG acceleration. Finally, discussion and perspectives will be found in Section 2.5.

2.2 Non-negative greedy algorithms

2.2.1 Basic definitions and notations

Given a data vector $\boldsymbol{y} \in \mathbb{R}^m$ and a dictionary $H \in \mathbb{R}^{m \times n}$, we are interested in finding a K-sparse non-negative weight vector $\boldsymbol{x} \in \mathbb{R}^n_+$ yielding an accurate approximation $\boldsymbol{y} \approx H\boldsymbol{x}$. This can be formulated as the constrained minimization program:

$$\min_{\boldsymbol{x} \geq \boldsymbol{0}} \|\boldsymbol{y} - H\boldsymbol{x}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{x}\|_0 \leq K. \tag{ℓ_0+}$$

We have the following useful identity for any two vectors $\boldsymbol{r}$, $\boldsymbol{h}$ of same length, $\boldsymbol{h}$ being normalized:

$$\min_{v \geq 0} \|\boldsymbol{r} - \boldsymbol{h}v\|^2 = \|\boldsymbol{r}\|^2 - \left(\max\{\boldsymbol{h}^{\mathrm{t}}\boldsymbol{r}, 0\}\right)^2 \tag{2.1}$$

where, $.^{\mathrm{t}}$ stands for the transpose operator.

We denote by $S = \mathrm{supp}(\boldsymbol{x}) = \{i : \boldsymbol{x}(i) \neq 0\}$ the support of $\boldsymbol{x}$ ($\boldsymbol{x}(i)$ being the i-th entry of $\boldsymbol{x}$), $\bar{S}$ the complement of S, $|S|$ the cardinality of S, H_S and $\boldsymbol{x}(S)$ the subdictionary and subvector indexed by S, respectively. Finally, $H^\dagger$ and $\mathrm{span}(H)$ are the pseudo-inverse and the column space of H, respectively. Let $\tilde{\boldsymbol{h}}_i^S = \boldsymbol{h}_i - H_S H_S^\dagger \boldsymbol{h}_i$ stand for the orthogonal projection of $\boldsymbol{h}_i$ onto the orthogonal complement $(\mathrm{span}(H_S))^\perp$, which will be simply denoted $\tilde{\boldsymbol{h}}_i$ whenever unambiguous, and $\tilde{\boldsymbol{g}}_i = \tilde{\boldsymbol{h}}_i / \|\tilde{\boldsymbol{h}}_i\|$ denote the normalized projected atom if $\boldsymbol{h}_i \notin \mathrm{span}(H_S)$, *i.e.*, $\tilde{\boldsymbol{h}}_i \neq \boldsymbol{0}$. If $\boldsymbol{h}_i \in \mathrm{span}(H_S)$, it will be convenient to set $\tilde{\boldsymbol{g}}_i = \boldsymbol{0}$.

For any support S, let us call an unconstrained least-squares (ULS) and a nonnegative least-squares (NNLS) solution corresponding to S, any vector $\boldsymbol{x}$ in $\mathbb{R}^n$ or $\mathbb{R}^n_+$, respectively, that minimizes $\|\boldsymbol{y} - H\boldsymbol{x}\|^2$ with the constraint that $\mathrm{supp}(\boldsymbol{x}) \subset S$. Such vectors will be denoted $\hat{\boldsymbol{x}}_S$ and $\hat{\boldsymbol{x}}_S^+$, respectively. The following notations will be also useful:

$$\boldsymbol{r}_S = \boldsymbol{y} - H\hat{\boldsymbol{x}}_S,$$
$$\boldsymbol{r}_S^+ = \boldsymbol{y} - H\hat{\boldsymbol{x}}_S^+.$$

When H_S is full column rank, $\|\boldsymbol{y} - H_S\boldsymbol{z}\|^2$ is a strictly convex function of $\boldsymbol{z} \in \mathbb{R}^{|S|}$, so $\hat{\boldsymbol{x}}_S^+$ and $\hat{\boldsymbol{x}}_S$ are then uniquely defined. Throughout the chapter, we will denote by $C \subset S$ the so-called compressed support, defined as the support of $\hat{\boldsymbol{x}}_S^+$. The NNLS optimal solutions can be characterized using the Karush-Kuhn-Tucker (KKT) conditions [Lawson and Hanson, 1974, Chap. 3], which are recalled next for completeness.

Lemma 2.1. *Consider the NNLS problem related to support S:*

$$\min_{\boldsymbol{x} \geq \boldsymbol{0}} \|\boldsymbol{y} - H\boldsymbol{x}\|^2 \quad \textit{s.t.} \quad \mathrm{supp}(\boldsymbol{x}) \subset S. \tag{2.2}$$

$\hat{\boldsymbol{x}}_S^+$ is a solution to (2.2) if and only if the KKT conditions are satisfied:

$$\begin{cases} H_C^{\mathrm{t}}(\boldsymbol{y} - H\hat{\boldsymbol{x}}_S^+) &= \boldsymbol{0} \\ H_{S\setminus C}^{\mathrm{t}}(\boldsymbol{y} - H\hat{\boldsymbol{x}}_S^+) &\leq \boldsymbol{0} \end{cases} \tag{2.3}$$

where $C := \mathrm{supp}(\hat{\boldsymbol{x}}_S^+) \subset S$.

Proof. From the definition of C, it is clear that $\hat{\boldsymbol{x}}_S^+(C) > \boldsymbol{0}$, so the active constraints in (2.2) are indexed by $\bar{C}$. Let $\boldsymbol{\lambda} \in \mathbb{R}^n$ gather the Lagrange multipliers related to both equality and inequality constraints. The Lagrangian function induced by (2.2) is defined as $\mathcal{L}(\boldsymbol{x}; \boldsymbol{\lambda}) = \|\boldsymbol{y} - H\boldsymbol{x}\|^2 - \boldsymbol{\lambda}^{\mathrm{t}}\boldsymbol{x}$ and the KKT conditions for optimal variables $(\hat{\boldsymbol{x}}_S^+, \hat{\boldsymbol{\lambda}})$ read:

$$\begin{cases} \nabla_{\boldsymbol{x}} \mathcal{L}(\hat{\boldsymbol{x}}_S^+; \hat{\boldsymbol{\lambda}}) = 2H^{\mathrm{t}}(H\hat{\boldsymbol{x}}_S^+ - \boldsymbol{y}) - \hat{\boldsymbol{\lambda}} = \boldsymbol{0}, \\ \forall i \in S,\ \hat{\boldsymbol{\lambda}}(i)\hat{\boldsymbol{x}}_S^+(i) = 0 \text{ with } \hat{\boldsymbol{x}}_S^+(i) \geq 0,\ \hat{\boldsymbol{\lambda}}(i) \geq 0, \\ \forall i \notin S,\ \hat{\boldsymbol{x}}_S^+(i) = 0. \end{cases} \tag{2.4}$$

For quadratic programming problems involving positive semidefinite matrices, the KKT conditions are necessary and sufficient conditions of optimality [Nocedal and Wright, 2006, Chap. 16], so $\hat{\boldsymbol{x}}_S^+$ is a solution to (2.2) *if and only if*

$$\begin{cases} \hat{\boldsymbol{\lambda}} = -2H^{\mathrm{t}}(\boldsymbol{y} - H\hat{\boldsymbol{x}}_S^+) \\ \hat{\boldsymbol{\lambda}}(C) = \boldsymbol{0} \\ \hat{\boldsymbol{\lambda}}(S\backslash C) \geq \boldsymbol{0} \end{cases}$$

that is, when (2.3) is satisfied. ■

Definition 2.1. *Let us call a positive support related to the full-size NNLS problem*

$$\min_{\boldsymbol{x} \geq \boldsymbol{0}} \|\boldsymbol{y} - H\boldsymbol{x}\|^2 , \tag{2.5}$$

any index set S such that H_S is full column rank and $\hat{\boldsymbol{x}}_S^+(S) > \boldsymbol{0}$. By extension, the empty support $S = \emptyset$ will also be considered as a positive support.

Lemma 2.2. *S is a positive support if and only if H_S is full rank and $\hat{\boldsymbol{x}}_S(S) > \boldsymbol{0}$. Moreover, when S is a positive support, $\hat{\boldsymbol{x}}_S^+ = \hat{\boldsymbol{x}}_S$, $\boldsymbol{r}_S^+ = \boldsymbol{r}_S$ and $H_S^{\mathrm{t}}\boldsymbol{r}_S = \boldsymbol{0}$.*

Proof. When S is a positive support, ULS and NNLS solutions $\hat{\boldsymbol{x}}_S$ and $\hat{\boldsymbol{x}}_S^+$ are uniquely defined and coincide. The orthogonality property $H_S^{\mathrm{t}}\boldsymbol{r}_S = \boldsymbol{0}$ follows from (2.3). ■

Positive supports will play an important role in our specification of fast non-negative orthogonal greedy algorithms.

2.2.2 Non-negative orthogonal greedy algorithms

Let us define the class of *non-negative orthogonal greedy* (NNOG) algorithms, sharing the following general structure. We start from the empty support $S = \emptyset$. At each iteration, an atom is moved from $\bar{S}$ to S. A new NNLS solution $\hat{\boldsymbol{x}}_S^+$ is then computed to optimally adapt the weights to the newly extended support. The algorithm stops when the desired cardinality K is reached or when the norm of the residual cannot decrease anymore. The general structure of NNOG algorithms is given by Algorithm 12. Some aspects will be made clear later, such as the role of the test $\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S > 0$ with respect to the decrease of the norm of the residual.

Algorithm 12: General structure of a non-negative orthogonal greedy algorithm to solve (ℓ_0+).

input : $\boldsymbol{y}, H, K$
output: $\boldsymbol{x}$

$\boldsymbol{x} \leftarrow \boldsymbol{0}$; $S \leftarrow \emptyset$; $\boldsymbol{r}_S \leftarrow \boldsymbol{y}$;
while $|S| < K$ *and* $\max_{i \in \bar{S}} \boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S > 0$ **do**
  Select an index $\ell \in \bar{S}$ by a selection rule $\mathcal{S}(\boldsymbol{y}, H, S)$;
  $S \leftarrow S \cup \{\ell\}$;
  $\boldsymbol{x} \leftarrow \hat{\boldsymbol{x}}_S^+$;
  $S \leftarrow \mathrm{supp}(\boldsymbol{x})$;
  $\boldsymbol{r}_S = \boldsymbol{y} - H\boldsymbol{x}$;
end

The NNOG class is a direct adaptation of the family of orthogonal greedy algorithms from the unconstrained case to the nonnegative one. At first glance, the two families only differ by

the fact that an NNLS solution is computed rather than a ULS one to update the weights at each iteration. However, some important features differ between the two cases, which require non trivial adaptations.

In both cases, the greedy character corresponds to the fact that a unique atom is added to the current support per iteration. However, a distinct feature of NNOG algorithms is that the support size may be smaller than the current iteration index, because some components of $\hat{\boldsymbol{x}}_S^+(S)$ may vanish at each iteration due to the activation of the corresponding nonnegativity constraints. In the unconstrained case, some components of $\hat{\boldsymbol{x}}_S(S)$ may also vanish, but such events are fortuitous and do not need any specific consideration.

In the unconstrained case, $\boldsymbol{r}_S$ is orthogonal to $\operatorname{span}(H_S)$. This geometrical property does not necessarily hold for $\boldsymbol{r}_S^+$ because NNLS is an inequality constrained problem. Fortunately, provided that the indices of zero components of $\hat{\boldsymbol{x}}_S^+$ are moved to $\bar{S}$, it remains true that $\boldsymbol{y}-H\hat{\boldsymbol{x}}_S^+$ is orthogonal to $\operatorname{span}(H_S)$. This is a direct consequence of the following lemma, which states that $\hat{\boldsymbol{x}}_S^+$ reads as a ULS solution related to the compressed version of support S.

Lemma 2.3. *For any S, let $C = \operatorname{supp}(\hat{\boldsymbol{x}}_S^+)$ (where neither $\hat{\boldsymbol{x}}_S^+$ nor C are necessarily unique if H_S is not full column rank). Then we have*

$$\hat{\boldsymbol{x}}_S^+ = \hat{\boldsymbol{x}}_C = \hat{\boldsymbol{x}}_C^+. \tag{2.6}$$

Proof. Using Lemma 2.1, we have $H_C^{\mathrm{t}}\boldsymbol{r}_S^+ = \boldsymbol{0}$. Since $\boldsymbol{r}_S^+ = \boldsymbol{y} - H\hat{\boldsymbol{x}}_S^+$, we get $H_C^{\mathrm{t}}\boldsymbol{y} = H_C^{\mathrm{t}}H\hat{\boldsymbol{x}}_S^+$. Thus, $\hat{\boldsymbol{x}}_S^+$ is a ULS solution (denoted by $\hat{\boldsymbol{x}}_C$) associated to support C. We have also $\hat{\boldsymbol{x}}_C = \hat{\boldsymbol{x}}_C^+$ since $\hat{\boldsymbol{x}}_C \geq \boldsymbol{0}$. ■

According to the above definition of NNOG algorithms, distinct algorithms can only differ by the selection rule used to select an atom at each iteration. The design of a selection rule corresponds to the definition of a function $\mathcal{S}(\boldsymbol{y}, H, S)$, taking values in $\bar{S}$. It is clear that some indices $\ell \in \bar{S}$ correspond to inappropriate choices, in the sense that their selection would produce $\hat{\boldsymbol{x}}_{S\cup\{\ell\}}^+(\ell) = 0$, and hence a useless iteration, and possibly an early stopping of the algorithm. In contrast, in the unconstrained case, any selection $\ell \in \bar{S}$ yields a decrease of $\|\boldsymbol{y} - H\boldsymbol{x}\|^2$ unless $\boldsymbol{h}_\ell \in \operatorname{span}(H_S)$. The capacity of some selection rules to avoid inappropriate selections is examined in the next two subsections.

Finally, a practically important aspect is the computing cost of NNOG algorithms. It is computationally more demanding to solve an NNLS problem than the corresponding ULS problem, so one must expect a larger computing cost for NNOG algorithms compared to their unconstrained counterparts. However, NNOG algorithms lend themselves to recursive implementations akin to usual orthogonal greedy schemes, as detailed in Section 2.3.

2.2.3 Descending atoms and descent selection rules

Greedy algorithms can be interpreted as descent algorithms dedicated to the minimization of the residual norm using supports of growing size. Contrary to the unconstrained case, only the selection of some atoms in $\bar{S}$ may produce a decrease of the residual at a given iteration of an NNOG algorithm. For the rest of the atoms in $\bar{S}$, the residual norm decrease is possible only if the nonnegativity constraint is violated. Such a specificity has both formal and practical consequences on the design of NNOG algorithms, as examined in this subsection.

Definition 2.2. *For a given support S, let us define the set of indices corresponding to descending atoms as follows:*

$$\mathcal{D}_S = \left\{ i \in \{1, \ldots, n\}, \|\boldsymbol{r}_{S\cup\{i\}}^+\| < \|\boldsymbol{r}_S^+\| \right\}.$$

Obviously, we have $\mathcal{D}_S \subset \bar{S}$. In what follows, we focus on selection rules ensuring the selection of a descending atom at any iteration. The latter rules are referred to as *descent* selection rules.

Definition 2.3. *A descent selection rule is a function $\mathcal{S}(\boldsymbol{y}, H, S)$ that takes its values in $\mathcal{D}_S$.*

Clearly, NNOG algorithms relying on a descent selection rule are descent algorithms and we have the following property.

Lemma 2.4. *NNOG algorithms relying on a descent selection rule terminate after a finite number of iterations.*

Proof. The error norm decreases at each iteration, and there is a finite number of supports with a cardinality not exceeding K, and thus a finite number of solutions to visit. ■

The following proposition allows one to characterize whether an atom is descending.

Proposition 2.1. *The descending atom condition $i \in \mathcal{D}_S$ is equivalent to*

$$0 < \boldsymbol{h}_i^{\mathrm{t}} \boldsymbol{r}_S^+. \tag{2.7}$$

When S is a positive support, it is also equivalent to each condition

$$0 < \boldsymbol{h}_i^{\mathrm{t}} \boldsymbol{r}_S, \tag{2.8}$$
$$0 < \tilde{\boldsymbol{g}}_i^{\mathrm{t}} \boldsymbol{r}_S^+, \tag{2.9}$$
$$0 < \tilde{\boldsymbol{g}}_i^{\mathrm{t}} \boldsymbol{r}_S. \tag{2.10}$$

When $H_{S\cup\{i\}}$ is full column rank, it is also equivalent to

$$\hat{\boldsymbol{x}}_{S\cup\{i\}}^+(i) > 0. \tag{2.11}$$

Proof. See Appendix 2.A.1. ■

Let us remark that from the first item of Proposition 2.1, the selection rule is invoked at Line 3 of Algorithm 12 only if $\mathcal{D}_S \neq \emptyset$, otherwise the stopping condition of Line 2 is activated. Hence, we do not need to define $\mathcal{S}(\boldsymbol{y}, H, S)$ when $\mathcal{D}_S = \emptyset$.

The following three lemmas have interesting consequences for the practical specification of valid NNOG algorithms.

Lemma 2.5. *If S is a positive support and $i \in \mathcal{D}_S$, then matrix $H_{S\cup\{i\}}$ is full column rank.*

Proof. Assume that S is a positive support, so H_S is full column rank, and $\hat{\boldsymbol{x}}_S^+ = \hat{\boldsymbol{x}}_S$. Let us also assume that $H_{S\cup\{i\}}$ is not full column rank. Then $\boldsymbol{h}_i \in \mathrm{span}(H_S)$, so $\hat{\boldsymbol{x}}_S$ is a ULS solution corresponding to $S \cup \{i\}$, and also an NNLS solution corresponding to $S \cup \{i\}$ since $\hat{\boldsymbol{x}}_S \geq \boldsymbol{0}$. This implies that $i \notin \mathcal{D}_S$. ■

Lemma 2.6. *After each iteration of an NNOG algorithm relying on a descent selection rule, it holds that the support of the current solution is positive.*

Proof. The proof is immediate by recursive application of Lemmas 2.3 and 2.5, starting with the empty support. ■

Let us stress that Lemma 2.6 refers to NNOG algorithms strictly conforming to the scheme of Algorithm 12 (with an additional restriction to descent selection rules at Line 3). In particular, the *support compression* step performed at Line 6 is necessary to make Lemma 2.5 applicable. To our best knowledge, such a compression step has not been proposed in any previous contribution about nonnegative greedy schemes targeting ℓ_0 minimization [2].

According to Lemma 2.6, the restriction to a descent selection rule implies $\boldsymbol{r}_S^+ = \boldsymbol{r}_S$ at any iteration, which justifies that we have dropped the '+' sign in Algorithm 12. This simplification is adopted in the rest of the chapter. Moreover, the termination rule $\max_i \boldsymbol{h}_i^{\mathrm{t}} \boldsymbol{r}_S \leq 0$ is used at Line 2 since in this case, there are no descending atoms anymore, so the residual cannot decrease by selection of a new atom.

[2] However, this kind of operation was introduced in [Leichner et al., 1993, Morup et al., 2008].

2.2.4 Examples of descent selection rules

In what follows, selection rules are denoted $\mathcal{S}(S)$, the dependence on $\boldsymbol{y}$ and H being implicit. Let us introduce three important selection rules by their distinct ways of picking an index in $\mathcal{D}_S$ when $\mathcal{D}_S \neq \emptyset$.

- NNOMP rule [Bruckstein et al., 2008, Ramamurthy et al., 2014, Yaghoobi et al., 2015, Kim and Haldar, 2016]:
$$\mathcal{S}_1(S) \in \arg\max_{i \notin S} \boldsymbol{h}_i^{\mathrm{t}} \boldsymbol{r}_S \tag{2.12}$$
- Suboptimal NNOLS (SNNOLS, [Yaghoobi and Davies, 2015]) rule:
$$\mathcal{S}_2(S) \in \arg\max_{i \notin S} \tilde{\boldsymbol{g}}_i^{\mathrm{t}} \boldsymbol{r}_S \tag{2.13}$$
- NNOLS rule [Yaghoobi and Davies, 2015]:
$$\mathcal{S}_3(S) \in \arg\min_{i \notin S} \|\boldsymbol{r}^+_{S\cup\{i\}}\|^2 \tag{2.14}$$

(2.14) is a descent selection rule by definition. The fact that (2.12) and (2.13) are descent selection rules is deduced from recursive application of Proposition 2.1 and Lemma 2.6. As regards the latter rules, $\boldsymbol{r}_S$ is the current residual vector, *i.e.*, a readily available quantity. On the other hand, projected atoms $\tilde{\boldsymbol{g}}_i$ enter rule (2.13), so we can expect the computing cost of (2.13) to be larger than that of (2.12). Rule (2.14) needs the solution of NNLS problems on supports $S \cup \{i\}$, which is even more demanding.

Note that [Wang et al., 2018] introduced another version of non-negative OMP named NN-OMP in which the selection rule is that of OMP. Clearly, this version does not rely on a descent selection rule. On the other hand, it is unclear whether the FNNOMP and FNNOLS algorithms proposed in [Yaghoobi et al., 2015, Yaghoobi and Davies, 2015] rely on a descent selection rule. They will therefore not be further analyzed.

The following proposition makes it possible to compare the three rules (2.12)-(2.14) by relating them to the minimization of a residual norm.

Proposition 2.2. *Rules* (2.12)-(2.14) *are equivalent to*

$$\mathcal{S}_j(S) \in \arg\min_{i \in \mathcal{D}_S} \mu_j(S, i), \tag{2.15}$$

where μ_j *are specific to each rule:*

$$\mu_1(S, i) = \min_{v} \|\boldsymbol{y} - H_S \boldsymbol{x}_S - \boldsymbol{h}_i v\|^2, \tag{2.16}$$
$$\mu_2(S, i) = \min_{\boldsymbol{u}, v} \|\boldsymbol{y} - H_S \boldsymbol{u} - \boldsymbol{h}_i v\|^2, \tag{2.17}$$
$$\mu_3(S, i) = \min_{\boldsymbol{u} \geq \boldsymbol{0}, v \geq 0} \|\boldsymbol{y} - H_S \boldsymbol{u} - \boldsymbol{h}_i v\|^2. \tag{2.18}$$

Proof. Let us first emphasize that because (2.12)-(2.14) are descent selection rules, $i \notin S$ in (2.12)-(2.14) can be replaced by $i \in \mathcal{D}_S$. Clearly, (2.15) with (2.18) simply duplicate (2.14). Using identity (2.1) with $\boldsymbol{r} = \boldsymbol{r}_S$ and $\boldsymbol{h} = \boldsymbol{h}_i$, (2.12) can be rewritten as the argmin over $i \in \mathcal{D}_S$ of

$$\min_{v \geq 0} \|\boldsymbol{y} - H_S \boldsymbol{x}_S - \boldsymbol{h}_i v\|^2. \tag{2.19}$$

Likewise, applying (2.1) with $\boldsymbol{r} = \boldsymbol{r}_S$ and $\boldsymbol{h} = \tilde{\boldsymbol{g}}_i$ and using the fact that $\tilde{\boldsymbol{g}}_i$ is the normalized version of $\tilde{\boldsymbol{h}}_i$, (2.13) can be rewritten as the argmin over $i \in \mathcal{D}_S$ of

$$\min_{v \geq 0} \|\boldsymbol{r}_S - \tilde{\boldsymbol{h}}_i v\|^2 = \min_{\boldsymbol{u}, v \geq 0} \|\boldsymbol{y} - H_S \boldsymbol{u} - \boldsymbol{h}_i v\|^2. \tag{2.20}$$

(2.20) follows from explicit minimization with respect to $\boldsymbol{u}$ and from the fact that $\boldsymbol{r}_S$ and $\tilde{\boldsymbol{h}}_i$ read as the orthogonal projections of $\boldsymbol{y}$ and $\boldsymbol{h}_i$ onto $(\mathrm{span}(H_S))^\perp$. Finally, the positivity constraint on v in (2.19)-(2.20) turns out to be inactive. Indeed, according to Proposition 2.1, we have that $\boldsymbol{h}_i^\mathrm{t}\boldsymbol{r}_S > 0$ and $\tilde{\boldsymbol{h}}_i^\mathrm{t}\boldsymbol{r}_S > 0$ for $i \in \mathcal{D}_S$. It is easy to see from (2.1) that the minimum error norm in (2.19)-(2.20) is then strictly lower than $\|\boldsymbol{r}_S\|$. Hence, the optimal variable v in (2.19)-(2.20) is nonzero.

Finally, (2.15) with (2.19) and (2.15) with (2.20) rewrite as (2.15) with (2.16) and (2.15) with (2.17), respectively. ■

For $j \in \{1, 2, 3\}$, an alternate way of viewing the descending character of each rule consists in noticing that $\|\boldsymbol{r}_{S\cup\{i\}}\|^2 \leq \mu_j(S, i) < \|\boldsymbol{r}_S\|^2$ for all $i \in \mathcal{D}_S$. Figure 2.2.4 illustrates the three rules in a simple case where all results differ. It is interesting to see that, by restricting the selection to the set of descending atoms, the selection rules of NNOMP and SNNOLS rely on the same criteria (2.16)-(2.17) as those of OMP and OLS, respectively.

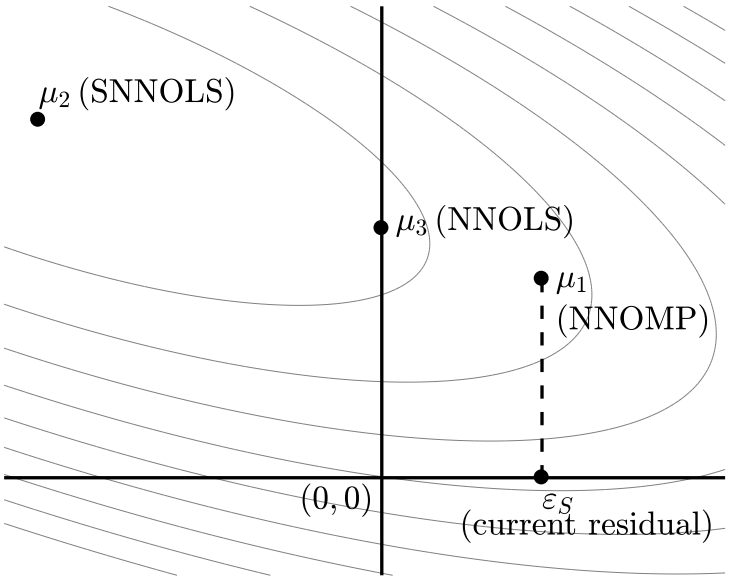


Figure 2.1: Graphical illustration of (2.16)-(2.18) in a simple case where S contains a single atom, whose weight u varies along the horizontal axis, while the weight v of the candidate atom varies along the vertical axis. The contour plot represents $\|\boldsymbol{y} - \boldsymbol{h}_S u - \boldsymbol{h}_i v\|^2$.

2.2.5 On the usefulness of support compression

As already mentioned, rules (2.12)-(2.14) were introduced in existing works devoted to (ℓ_0+) without support compression. Let us analyze the impact of removing the support compression step at Line 6 in Algorithm 12.

- The subset S then becomes an *extended* support, with $\boldsymbol{x} = \hat{\boldsymbol{x}}_S^+$ and $\mathrm{supp}(\boldsymbol{x}) \subset S$. As a consequence, when $\mathrm{supp}(\boldsymbol{x}) \neq S$, the current residual $\boldsymbol{r}_S^+ = \boldsymbol{y} - H\boldsymbol{x}$ does not identify to $\boldsymbol{r}_S$ anymore since $\boldsymbol{x}$ does not read as a ULS solution related to S.

- In the case of NNOMP and NNOLS (selection rules (2.12) and (2.14), with $\boldsymbol{r}_S$ replaced by the current residual $\boldsymbol{r}_S^+$), Proposition 2.1 can still be applied, since condition $\boldsymbol{h}_i^\mathrm{t}\boldsymbol{r}_S^+ > 0$ still characterizes descending atoms without support compression. As for SNNOLS, we cannot guarantee that $\mathcal{S}(\boldsymbol{y}, H, S)$ defined in (2.13) is a descent selection rule when S is not positive, since the equivalence (2.9) in Proposition 2.1 does not hold anymore.

- Finally, for any NNOG algorithm, Lemma 2.5 is no more valid when S is not a positive support, so H_S might become rank deficient at any subsequent iteration, $\hat{\boldsymbol{x}}_S^+$ not being well defined anymore.

We consider the last point as the main justification to apply support compression. An open question would be to determine whether the assumptions of Lemma 2.5 can be weakened, the positivity of S being replaced by the full rankness of H_S. However, we lack concrete elements in this sense. Therefore, at this stage, NNOG with compression present formal guarantees that NNOG without compression do not.

2.3 Active-set NNLS algorithms and recursivity

Let us consider the NNLS problem related to support S in (2.2). When matrix H_S is full column rank, it is a special case of a strictly convex quadratic program. The successive resolution of possibly many NNLS problems for nested supports S is a basic ingredient of NNOG algorithms. NNLS problems do not admit closed-form solutions in general. However, *active-set* NNLS (AS-NNLS) algorithms are well-known schemes that solve NNLS problems in a finite number of iterations [Lawson and Hanson, 1974, Chen et al., 2010]. Moreover, the computation of NNLS solutions can be efficiently accelerated by exploiting that warm start solutions are available and using classical recursive updates of matrix factorizations.

This section first contains a short reminder on active-set NNLS algorithms and on their efficient implementation. Then, we show how to preserve computational efficiency when NNLS subproblems are solved within an NNOG scheme. We also analyze the similarity between the structures of AS-NNLS algorithm and NNOMP, already pointed out in [Peharz and Pernkopf, 2012, Foucart and Koslicki, 2014].

2.3.1 Fast active-set algorithms

Among many numerical methods for solving (2.2), AS-NNLS algorithms correspond to greedy schemes, since the solution is found by incremental modifications of its support $V \subset S$, the *active set* being defined as the complementary set $S \backslash V$ (by reference to the active constraints). Such an incremental structure is an essential element to obtain practically fast implementations [Lawson and Hanson, 1974, Björck, 1996, Nocedal and Wright, 2006]. Whenever V is modified, the corresponding ULS solution $\hat{\boldsymbol{x}}_V$ is updated. Each iteration requires at least one ULS solution of the *selection* type, *i.e.*, $H_{V\cup\{\ell\}}^\dagger \boldsymbol{y}$. Some iterations also need to compute *deselection* type ULS solutions $H_{V\backslash\{\ell\}}^\dagger \boldsymbol{y}$. For the sake of computational efficiency, it is crucial to compute both types of solutions recursively, given that $\hat{\boldsymbol{x}}_V$ is already available. In this respect, the situation is identical to that of bi-directional greedy algorithms in the unconstrained case (*e.g.,* Bayesian OMP [Herzet and Drémeau, 2010], SBR [Soussen et al., 2011] or FoBa [Zhang, 2011]), also called stepwise algorithms [Miller, 2002, Chapter 3], for which selections and deselections are implemented recursively. Fast recursive implementations require specific computations and storage of quantities related to the Gram matrix $G_V = H_V^t H_V$. Efficient selection steps can be obtained using QR or Cholesky matrix factorizations applied to G_V, or the Matrix Inversion Lemma (MIL), with roughly the same cost [Sturm and Christensen, 2012]. MIL consists in storing and updating the inverse of G_V. It appears to be the cheapest concerning deselections, so our default choice here is based on the MIL.

2.3.2 Warm start and full recursivity

Since NNOG algorithms are based on iterated calls to an active-set scheme, we must carefully consider the way we initialize the latter. The starting point of AS-NNLS is usually defined as

the zero solution (associated with the empty support). This is the case in the Lawson-Hanson algorithm, which is the reference AS-NNLS scheme [Lawson and Hanson, 1974]. In [Nocedal and Wright, 2006, Chap. 16], Nocedal and Wright proposed an AS-NNLS algorithm with any feasible vector as a possible starting point. Algorithm 13 is a generalized version of the Lawson-Hanson algorithm to address the NNLS problem related to an augmented target set $T \supset S$, where the initial point is not restricted to be the zero vector (the rest of the scheme being unaltered). Specifically, the initial point is set as the ULS solution $\hat{\boldsymbol{x}}_S \geq \boldsymbol{0}$, S being a positive support. For this specific initial point, it can easily be checked that Algorithm 13 identifies with Nocedal and Wright's scheme.

Algorithm 13: Active-set algorithm to solve the NNLS problem related to T, starting from a positive support S. Format: $\texttt{AS\_NNLS}(\boldsymbol{y}, H, T, S, \hat{\boldsymbol{x}}_S)$

input : $\boldsymbol{y}$, H, target set T, initial support $S \subset T$, $\hat{\boldsymbol{x}}_S$
output: $V := \text{supp}(\hat{\boldsymbol{x}}_T^+)$, $\hat{\boldsymbol{x}}_T^+ := \hat{\boldsymbol{x}}_V$

$\boldsymbol{x} \leftarrow \hat{\boldsymbol{x}}_S$; $V \leftarrow S$; $\boldsymbol{r}_V \leftarrow \boldsymbol{y} - H\boldsymbol{x}$;
while $\max\{\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_V,\, i \in T\backslash V\} > 0$ **do**
  $\ell^+ \leftarrow \arg\max\{\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_V,\, i \in T\backslash V\}$;
  $V \leftarrow V \cup \{\ell^+\}$;
  Update $\hat{\boldsymbol{x}}_V$ (call Algorithm 15);
  while $\min(\hat{\boldsymbol{x}}_V) < 0$ **do**
    $\ell^- \in \underset{\{i \in V : \hat{\boldsymbol{x}}_V(i) < 0\}}{\arg\min} \boldsymbol{x}(i)/(\boldsymbol{x}(i) - \hat{\boldsymbol{x}}_V(i))$;
    $\alpha \leftarrow \boldsymbol{x}(\ell^-)/(\boldsymbol{x}(\ell^-) - \hat{\boldsymbol{x}}_V(\ell^-))$;
    $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha(\hat{\boldsymbol{x}}_V - \boldsymbol{x})$;
    $V \leftarrow V \setminus \{\ell^-\}$;
    Update $\hat{\boldsymbol{x}}_V$ (call Algorithm 15);
  end
  $\boldsymbol{x} \leftarrow \hat{\boldsymbol{x}}_V$;
  $\boldsymbol{r}_V \leftarrow \boldsymbol{y} - H\boldsymbol{x}$;
end

Algorithm 14: NNOG with active-set implementation.

input : $\boldsymbol{y}, H, K$
output: $\boldsymbol{x} := \hat{\boldsymbol{x}}_S^+$ (with S a positive support)

$\boldsymbol{x} \leftarrow \boldsymbol{0}$; $S \leftarrow \emptyset$; $\boldsymbol{r}_S \leftarrow \boldsymbol{y}$;
while $|S| < K$ *and* $\max_{i \in \bar{S}} \boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S > 0$ **do**
  Select an index $\ell \in \bar{S}$ by a selection rule $\mathcal{S}(\boldsymbol{y}, H, S)$;
  Call $[C, \boldsymbol{x}] = \texttt{AS\_NNLS}(\boldsymbol{y}, H, S \cup \{\ell\}, S, \boldsymbol{x})$;
  $S \leftarrow C$;
  $\boldsymbol{r}_S = \boldsymbol{y} - H\boldsymbol{x}$;
end

At any iteration of a valid NNOG algorithm, the current solution $\hat{\boldsymbol{x}}_S^+$ can be used as the initial point to compute $\hat{\boldsymbol{x}}_{S\cup\{\ell\}}^+$ using Algorithm 13 with $T = S \cup \{\ell\}$. In this way, the initial support is $V = S$, so $T\backslash V = \{\ell\}$ and the first iteration of AS-NNLS begins by selecting ℓ. In practice, the NNLS algorithm is expected to terminate after a single iteration (if no deselection is performed in the second part of it), or otherwise after very few iterations. Algorithm 14 is a global view of the active-set implementation of NNOG obtained by integrating the calls to

the AS-NNLS solver (Algorithm 13) in the NNOG framework (Algorithm 12). Whenever a new atom ℓ is selected, AS-NNLS starts by computing $\hat{\boldsymbol{x}}_{S\cup\{\ell\}}$. If $\hat{\boldsymbol{x}}_{S\cup\{\ell\}} \geq \mathbf{0}$, then $\hat{\boldsymbol{x}}^{+}_{S\cup\{\ell\}} = \hat{\boldsymbol{x}}_{S\cup\{\ell\}}$ and AS-NNLS stops after one support change. Otherwise, AS-NNLS deselects at least one atom from $S \cup \{\ell\}$ (Algorithm 13, Line 7) and then alternates between atom selections and series of deselections. This mechanism is illustrated by a simple example in the next subsection.

A reduced number of iterations is obtained because we use Algorithm 13 with a warm start. To further improve the overall numerical efficiency, we also need to reduce the computing cost of the ULS solution at Lines 5 and 11. These are selection and deselection-type ULS problems, respectively, that can be solved recursively provided that the inverse of the Gram matrix G_V be both an input and an output quantity of the NNLS algorithm. In Appendix 2.B, Algorithm 15 is a pseudo-code to implement ULS updates in the forward ($V \leftarrow V \cup \{\ell\}$) and backward scenarios ($V \leftarrow V\backslash\{\ell\}$). This implementation enables us to obtain a fully recursive implementation of AS-NNLS as well by updating the Gram matrix inverse at each call to ULS in Algorithm 13 (Lines 4-5 and 10-11).

2.3.3 Step-by-step illustrative example of NNOG

Fig. 2.2 displays a schematic step-by-step illustration of NNOG. NNOG iterates S_k are represented with bullets. NNOG starts with the empty support. In this example, it turns out that during the first three iterations, NNOG yields a positive support $S_{k-1} \cup \{\ell\}$ ($\hat{\boldsymbol{x}}_{S_{k-1}\cup\{\ell\}} \geq \mathbf{0}$), hence $S_k \leftarrow S_{k-1} \cup \{\ell\}$. Therefore, $S_1 \subset S_2 \subset S_3$ are of cardinalities 1, 2 and 3, respectively. At iteration 4, $S_3 \cup \{\ell\}$ is not positive so AS-NNLS performs two support changes, namely the selection of ℓ and a deselection. The next NNOG iterate reads $S_4 \leftarrow S_3 \cup \{\ell\}\backslash\{\ell_1\}$. Iteration 5 is more tricky (and unlikely). Here again, $S_4 \cup \{\ell\}$ is not a positive support. The first deselection does not yield a positive support either, so another deselection is carried out, yielding $V \leftarrow S_4 \cup \{\ell\}\backslash\{\ell_1, \ell_2\}$, V being a positive support. However, the stopping condition of AS-NNLS (Line 2 of Algorithm 13, with $T \leftarrow S_4 \cup \{\ell\}$) is not met since $\boldsymbol{h}_{\ell_1}$ is a descending atom. Therefore, ℓ_1 is reselected: $V \leftarrow V \cup \{\ell_1\}$. Since V is now a positive support and there are no descending atoms anymore in $T\backslash V$, the convergence of AS-NNLS is reached. In the last two NNOG iterations, $S_{k-1} \cup \{\ell\}$ are positive supports, so single selection moves are done within AS-NNLS.

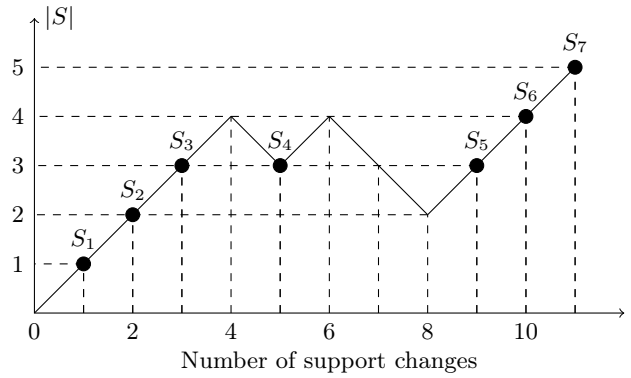


Figure 2.2: Step-by-step illustration of NNOG after each change of support. Bullets represent supports corresponding to the first seven NNOG iterates whereas other intermediate supports found during the calls to AS-NNLS are represented without bullets.

2.3.4 Connection between AS-NNLS and NNOMP

In Algorithm 14, the structure of the NNOG main iteration consists of one atom selection followed by a variable number of updates (selections or deselections) of the support S when the ULS solution $\hat{\boldsymbol{x}}_S$ has some negative entries. The first of these updates is necessarily an atom deselection. Interestingly, [Peharz and Pernkopf, 2012] and [Foucart and Koslicki, 2014] pointed out that the AS-NNLS algorithm initialized with the zero vector, has a structure similar to NNOMP: each main NNLS iteration consists of an atom selection followed by a series of atom deselections (respectively, Line 3 and Lines 6-12 of Algorithm 13). This connection led Peharz and Pernkopf to propose AS-NNLS with an early stopping rule $|S| = K$ as a stand-alone NNOG algorithm (called Sparse NNLS in [Peharz and Pernkopf, 2012]). Given the strong similarity of Sparse NNLS and NNOMP, an interesting question is to determine whether their iterates always coincide. It turns out that this is not always true. However, as long as Sparse NNLS performs only simple support changes, both algorithms yield the same iterates, according to the following proposition.

Proposition 2.3. *Let us consider Sparse NNLS, i.e., Algorithm 13 initialized with the empty support together with the early stopping rule $|S| = K$. In any case where no iteration produces two or more successive removals in Lines 6–12, the output of Sparse NNLS identifies with that of NNOMP (i.e., Algorithm 12 with rule $\mathcal{S}_1$ in (2.12)).*

Proof. See Appendix 2.A.2. ■

2.4 Acceleration of NNOG algorithms

The NNOMP, SNNOLS and NNOLS selection rules (2.12)-(2.14) all read as the optimization of a criterion with respect to the candidate index $i \notin S$. Since all three ensure the selection of an atom in $\mathcal{D}_S$, an obvious acceleration of SNNOLS and NNOLS consists of pre-selecting the descending atoms according to $\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S > 0$ (see Proposition 2.1) to carry out the optimization tasks (2.13) and (2.14) over $i \in \mathcal{D}_S$ only. This operation is referred to as *type-I pruning* of the dictionary[3]. Testing the sign of $\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S$ requires $\mathcal{O}(m)$ operations. This is much less than the (recursive) computation of the criteria $\tilde{\boldsymbol{g}}_i^{\mathrm{t}}\boldsymbol{r}_S$ and $\|\boldsymbol{r}^+_{S\cup\{i\}}\|^2$, which costs at least $\mathcal{O}(|S|^2 + km)$ operations.

2.4.1 Atom selection

The atom selection step of both NNOMP and SNNOLS can be efficiently implemented using vectorized computations. The NNOMP case is the simplest, since $H_S^{\mathrm{t}}\boldsymbol{r}_S$ directly yields the expected set of inner products $\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S$. Indeed, the selection steps of OMP and NNOMP are both based on the minimization of (2.16), so they share the same possibilities of parallel computations. Likewise, OLS and SNNOLS being both based on the minimization of (2.17), they share the same possibilities of parallel implementation. In coherence with our choice of recursive implementation of ULS solutions (see Appendix 2.B), we have adopted a MIL based solution to solve (2.17) in a vectorized way (see Matlab code in supplementary material).

In contrast, the NNOLS selection rule (2.14) does not lend itself to fully vectorized computations, since we have as many NNLS subproblems to solve as candidate atoms, with a variable number of subiterations of AS-NNLS for each of them. Fortunately, the structure of the NNOLS rule can be made closer to that of SNNOLS, with the benefit of vectorized computations for

[3]Contrary to screening techniques, see *e.g.,* [Bonnefoy et al., 2015], note that the atoms indexed by $i \notin \mathcal{D}_S$ are pruned from the dictionary for the current NNOG iteration only, but they are considered again in further iterations.

the largest part. The key point is that for each candidate atom, the initial step of AS-NNLS corresponding to Lines 1-5 of Algorithm 13 yields the same unconstrained minimizer $\hat{\boldsymbol{x}}_{S\cup\{i\}}$ as the one involved in the SNNOLS rule (2.17). Hence, these vectors can be obtained using vectorized computations that exactly identify to the main step of SNNOLS atom selection. The extra computations induced by NNOLS reside in the additional AS-NNLS iterations required for each non-positive support $S \cup \{i\}$. According to our empirical tests, only a small minority of atoms needs more than one iteration. Moreover, a lot of them can be pruned without actually performing any additional AS-NNLS iterations. Let us denote by e_{opt} the smallest residual error produced by atoms i for which $S \cup \{i\}$ is a positive support. Since $\|\boldsymbol{r}^+_{S\cup\{i\}}\| \geq \|\boldsymbol{r}_{S\cup\{i\}}\|$ for all i, one can immediately ignore the atoms i for which $\|\boldsymbol{r}_{S\cup\{i\}}\| \geq e_{\text{opt}}$. Moreover, since we sequentially visit the remaining ones, the threshold e_{opt} can be gradually lowered whenever a new atom is found to improve the smallest current residual error. This operation called *type-II pruning* is specific to NNOLS implementation.

2.4.2 Coefficient update

Once an atom i is selected, NNOG algorithms need to update the coefficients by solving an NNLS subproblem (Algorithm 12, Line 5). However, in the case of NNOLS, the update is already being performed in the selection step. In the case of NNOMP, one needs to call the AS-NNLS algorithm (Algorithm 13) from the initial set S to the target set $S \cup \{\ell\}$. For SNNOLS, a call to Algorithm 13 is needed only when $S \cup \{\ell\}$ is not positive.

2.4.3 Software

An open source Matlab implementation of the acceleration strategies detailed above is provided on CodeOcean [Nguyen et al., 2019b] (`https://doi.org/10.24433/CO.2445681.v1`). This software contains a fully recursive, vectorized version of NNOMP, NNOLS and SNNOLS together with test programs.

2.5 Conclusion

Until now, greedy algorithms dedicated to non-negative sparse signal reconstruction have been considered as slow schemes, requiring the repeated resolution of constrained least square problems. In order to accelerate the computation, approximate schemes have been proposed [Yaghoobi et al., 2015, Yaghoobi and Davies, 2015] at the price of some loss of control on the algorithmic behavior, and possibly degraded performance. Another commonly found option has been to replace the ℓ_0 "norm" by the ℓ_1-norm and to solve the resulting convex programming problem, with a possible loss in terms of performance (see [Peharz and Pernkopf, 2012] for an interesting case of sparse NMF).

In this chapter, our first contribution is to provide a unified framework to define non-negative orthogonal greedy algorithms in a formal way, ensuring well-defined iterations. The second and probably most important one in terms of practical impact, is to show that the additional cost of non-negative greedy algorithms to handle the sign constraint can be strongly reduced using three ingredients. The main one is that non-negative greedy algorithms can be made fully recursive. Moreover, several pruning strategies can be combined to reduce the number of tests at the atom selection stage. Finally, the latter step can benefit from vectorized computations.

Our contributions can be extended in several directions. A straightforward generalization can be made to deal with nonnegativity-constrained simultaneous sparse decomposition, which is useful in several applications such as hyperspectral imaging [Wang et al., 2018], dynamic PET [Lin et al., 2014], and diffusion MRI [Kim and Haldar, 2016]. On the other hand, other

greedy algorithms such as CoSaMP [Needell and Tropp, 2009], BOMP [Herzet and Drémeau, 2010] and SBR [Soussen et al., 2011] could also be extended to the non-negative setting using similar principles and using a recursive implementation.

Appendix 2.A Proof of technical results

2.A.1 Proof of Proposition 2.1

Let us first prove that $\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S^+ > 0$ implies $i \in \mathcal{D}_S$. Let $\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S^+ > 0$. According to (2.1) for $\boldsymbol{r} = \boldsymbol{r}_S^+$, we deduce that $i \in \mathcal{D}_S$ since

$$\|\boldsymbol{r}_{S\cup\{i\}}^+\|^2 \leq \min_{v\geq 0} \left\|\boldsymbol{r}_S^+ - v\boldsymbol{h}_i\right\|^2 < \|\boldsymbol{r}_S^+\|^2.$$

Conversely, let $i \in \mathcal{D}_S$. Define $f(\boldsymbol{z}) = \left\|\boldsymbol{y} - H_{S\cup\{i\}}\boldsymbol{z}\right\|^2$ where $\boldsymbol{z} \in \mathbb{R}^{|S|+1}$. Let us also define the subvectors $\boldsymbol{z}_S := \hat{\boldsymbol{x}}_S^+(S\cup\{i\})$ and $\boldsymbol{z}_{S\cup\{i\}} := \hat{\boldsymbol{x}}_{S\cup\{i\}}^+(S\cup\{i\})$, $\hat{\boldsymbol{x}}_S^+$ and $\hat{\boldsymbol{x}}_{S\cup\{i\}}^+$ being two NNLS solutions related to S and $S\cup\{i\}$. Condition $i \in \mathcal{D}_S$ reads

$$f(\boldsymbol{z}_{S\cup\{i\}}) < f(\boldsymbol{z}_S).$$

Since f is convex, one has

$$(\boldsymbol{z}_{S\cup\{i\}} - \boldsymbol{z}_S)^{\mathrm{t}}\nabla f(\boldsymbol{z}_S) \;\leq\; f(\boldsymbol{z}_{S\cup\{i\}}) - f(\boldsymbol{z}_S) \;<\; 0 \tag{2.21}$$

where the gradient of f is defined by

$$\nabla f(\boldsymbol{z}_S) = 2H_{S\cup\{i\}}^{\mathrm{t}}(H_{S\cup\{i\}}\boldsymbol{z}_S - \boldsymbol{y}) = -2H_{S\cup\{i\}}^{\mathrm{t}}\boldsymbol{r}_S^+.$$

Denoting by $C := \mathrm{supp}(\hat{\boldsymbol{x}}_S^+)$ the compressed support, we have from Lemma 2.1 that $H_{S\backslash C}^{\mathrm{t}}\boldsymbol{r}_S^+ \leq \boldsymbol{0}$ and $H_C^{\mathrm{t}}\boldsymbol{r}_S^+ = \boldsymbol{0}$. Since $\hat{\boldsymbol{x}}_S^+$ is supported by C, the latter equality implies that $\boldsymbol{z}_S^{\mathrm{t}}\nabla f(\boldsymbol{z}_S) = 0$. (2.21) yields $(\boldsymbol{z}_{S\cup\{i\}})^{\mathrm{t}}H_{S\cup\{i\}}^{\mathrm{t}}\boldsymbol{r}_S^+ > 0$, *i.e.*,

$$(\hat{\boldsymbol{x}}_{S\cup\{i\}}^+(S\cup\{i\}))^{\mathrm{t}}H_{S\cup\{i\}}^{\mathrm{t}}\boldsymbol{r}_S^+ \;>\; 0. \tag{2.22}$$

Since $H_C^{\mathrm{t}}\boldsymbol{r}_S^+ = \boldsymbol{0}$, (2.22) rereads:

$$(\hat{\boldsymbol{x}}_{S\cup\{i\}}^+(S\backslash C))^{\mathrm{t}}H_{S\backslash C}^{\mathrm{t}}\boldsymbol{r}_S^+ + (\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S^+)\,\hat{\boldsymbol{x}}_{S\cup\{i\}}^+(i) \;>\; 0 \tag{2.23}$$

and since $H_{S\backslash C}^{\mathrm{t}}\boldsymbol{r}_S^+ \leq \boldsymbol{0}$ and $\hat{\boldsymbol{x}}_{S\cup\{i\}}^+ \geq \boldsymbol{0}$, (2.23) implies that

$$(\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S^+)\,\hat{\boldsymbol{x}}_{S\cup\{i\}}^+(i) \;>\; 0 \tag{2.24}$$

and thus $\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S^+ > 0$.

Let us now assume that S is a positive support. According to Lemma 2.2, we have $\hat{\boldsymbol{x}}_S^+ = \hat{\boldsymbol{x}}_S$ and $\boldsymbol{r}_S^+ = \boldsymbol{r}_S$, so (2.7) and (2.8) are identical, as well as (2.9) and (2.10). To show that (2.7)-(2.8) are equivalent to (2.9)-(2.10), we first notice that $\boldsymbol{r}_S \in (\mathrm{span}(H_S))^{\perp}$, thus $\tilde{\boldsymbol{h}}_i^{\mathrm{t}}\boldsymbol{r}_S = \boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{r}_S$ since $\tilde{\boldsymbol{h}}_i - \boldsymbol{h}_i \in \mathrm{span}(H_S)$. Therefore, (2.8) rereads $0 < \tilde{\boldsymbol{h}}_i^{\mathrm{t}}\boldsymbol{r}_S$, which implies that $\tilde{\boldsymbol{h}}_i \neq \boldsymbol{0}$ and $\tilde{\boldsymbol{g}}_i = \tilde{\boldsymbol{h}}_i/\|\tilde{\boldsymbol{h}}_i\| \neq \boldsymbol{0}$. Hence, (2.8) rereads $0 < \tilde{\boldsymbol{g}}_i^{\mathrm{t}}\boldsymbol{r}_S$, which identifies with (2.10).

Finally, let us show that $i \in \mathcal{D}_S$ is equivalent to condition (2.11). Consider the function $f(\boldsymbol{z}) = \left\|\boldsymbol{y} - H_{S\cup\{i\}}\boldsymbol{z}\right\|^2$ and the notations $\boldsymbol{z}_S$ and $\boldsymbol{z}_{S\cup\{i\}}$ defined above. Assuming that $H_{S\cup\{i\}}$ is full column rank, we have that f is strictly convex, so f admits a unique minimizer $\boldsymbol{z}_{S\cup\{i\}}$. If $\hat{\boldsymbol{x}}_{S\cup\{i\}}^+(i) > 0$, then $\boldsymbol{z}_{S\cup\{i\}} \neq \boldsymbol{z}_S$ and

$$\|\boldsymbol{r}_{S\cup\{i\}}^+\|^2 = f(\boldsymbol{z}_{S\cup\{i\}}) < \|\boldsymbol{r}_S^+\|^2 = f(\boldsymbol{z}_S),$$

that is, $i \in \mathcal{D}_S$. Conversely, $\hat{\boldsymbol{x}}_{S\cup\{i\}}^+(i) = 0$ implies $\hat{\boldsymbol{x}}_{S\cup\{i\}}^+ = \hat{\boldsymbol{x}}_S^+$, hence $\boldsymbol{r}_{S\cup\{i\}}^+ = \boldsymbol{r}_S^+$ and $i \notin \mathcal{D}_S$.

2.A.2 Proof of Proposition 2.3

Any iteration of AS-NNLS starts with the addition of a new atom to the current support (Line 4 of Algorithm 13). Then, a variable number of atoms are removed from it one after the other (Lines 6 to 12). Let r denote the number of removals at the current AS-NNLS iteration. Let also $V \subset \{1, \dots, n\}$ and $\hat{\boldsymbol{x}}_V$ respectively stand for the current support and solution obtained at Lines 4 and 5, and let $V' \subset V$ and $\hat{\boldsymbol{x}}_{V'}$ denote the corresponding quantities after r removals. Let us first show that any AS-NNLS iteration for which $r = 0$ or $r = 1$ yields a solution of the NNLS problem restricted to the support V.

If $\hat{\boldsymbol{x}}_V \geq \boldsymbol{0}$, then $r = 0$, so $V' = V$ and $\hat{\boldsymbol{x}}_{V'} = \hat{\boldsymbol{x}}_V^+ \geq \boldsymbol{0}$. Otherwise, we have $\min(\hat{\boldsymbol{x}}_V) < 0$ and $r > 0$. If $r = 1$, a single index ℓ^- is removed at Lines 6-12, so that $V' = V \setminus \{\ell^-\}$, and $\hat{\boldsymbol{x}}_{V'} \geq \boldsymbol{0}$. Let us remark that we have $\hat{\boldsymbol{x}}_V(\ell^-) < 0$ according to Line 7. Let us then prove that $\hat{\boldsymbol{x}}_{V'} = \hat{\boldsymbol{x}}_V^+$ by showing that KKT conditions are satisfied at $\hat{\boldsymbol{x}}_{V'}$ for the NNLS problem related to support V. Note that the NNLS solution is unique since the supports generated by AS-NNLS are such that H_V is full column rank, as pointed out in subsection 2.3.2. According to Lemma 2.1, the KKT conditions read:

$$H_{V'}^{\mathrm{t}}(\boldsymbol{y} - H\hat{\boldsymbol{x}}_{V'}) = \boldsymbol{0}, \tag{2.25a}$$

$$\boldsymbol{h}_{\ell^-}^{\mathrm{t}}(\boldsymbol{y} - H\hat{\boldsymbol{x}}_{V'}) \leq 0. \tag{2.25b}$$

(2.25a) is obviously satisfied. On the other hand, remark that $\hat{\boldsymbol{x}}_{V'} = \hat{\boldsymbol{x}}_{V'}^+$ since $\hat{\boldsymbol{x}}_{V'} \geq \boldsymbol{0}$ and that according to Proposition 2.1, $\hat{\boldsymbol{x}}_V(\ell^-) < 0$ implies that $\boldsymbol{h}_{\ell^-}^{\mathrm{t}} \boldsymbol{r}_{V'} \leq \boldsymbol{0}$, which identifies with (2.25b). This concludes the proof.

Appendix 2.B Recursive implementation of ULS

Algorithm 15 recalls the recursive ULS computation using MIL [Björck, 1996] for selection (forward move $V \leftarrow V \cup \{\ell\}$) and deselection (backward move $V \leftarrow V \backslash \{\ell\}$) operations. The ULS solution $\hat{\boldsymbol{x}} := \hat{\boldsymbol{x}}_V$ is updated. Moreover, $\Theta := (H_V^{\mathrm{t}} H_V)^{-1}$ refers to the inverse of the Gram matrix related to subset V. The Boolean entry `fw` is set to `true` and `false` for selection and deselection updates, respectively. Finally, e^2 stands for the squared residual error $\|\boldsymbol{r}_V\|^2$. All these factors are updated in Algorithm 15. Notation $-j$ refers to all indices except j, and $\boldsymbol{\theta}_j$ stands for the j-th column of Θ.

For reminding, MIL for block matrix can be read as [Golub and Van Loan, 1996]

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix} \tag{2.26}$$

When a new atom is added to the support, $S' = S \cup \{\ell\}$ and $H_{S'} = [H_S, \boldsymbol{h}_\ell]$. Therefore,

$$H_{S'}^{\mathrm{t}} H_{S'} = \begin{bmatrix} H_S^{\mathrm{t}} H_S & H_S^{\mathrm{t}} \boldsymbol{h}_\ell \\ \boldsymbol{h}_\ell^{\mathrm{t}} H_S & \|\boldsymbol{h}_i\|^2 \end{bmatrix}.$$

By applying (2.26), one can express the inversion $(H_{S'}^{\mathrm{t}} H_{S'})^{-1}$ by means of $(H_S^{\mathrm{t}} H_S)^{-1}$ as formulated at line 7 of Algorithm 15. When an atom is removed from the support, $S' = S \setminus \{\ell\}$. Let us firstly consider the case where $\boldsymbol{h}_\ell$ is the last column of H_S. Then $H_S = [H_{S'}, \boldsymbol{h}_\ell]$. By switching S and S' in the calculation of the case of atom addition, one can express the inversion $(H_{S'}^{\mathrm{t}} H_{S'})^{-1}$ by means of $(H_S^{\mathrm{t}} H_S)^{-1}$ as formulated at line 13 of Algorithm 15. The case where $\boldsymbol{h}_\ell$ is not the last column of H_S can be turned to the previous case by using a permutation matrix

$$P_j = [\boldsymbol{e}_1, \cdots, \boldsymbol{e}_{j-1}, \boldsymbol{e}_{|S|}, \boldsymbol{e}_j, \cdots, \boldsymbol{e}_{|S|-1}]$$

Algorithm 15: Recursive ULS [Björck, 1996].
Format: ULS($\boldsymbol{y}, H, V, \texttt{fw}, \ell, \hat{\boldsymbol{x}}, \Theta, e^2$)

1 **if** `fw` **then**
2 $\quad \boldsymbol{\phi} \leftarrow H_V^{\mathrm{t}} \boldsymbol{h}_\ell$;
3 $\quad \delta \leftarrow (1 - \boldsymbol{\phi}^{\mathrm{t}} \Theta \boldsymbol{\phi})^{-1}$;
4 $\quad \beta \leftarrow \boldsymbol{\phi}^{\mathrm{t}} \hat{\boldsymbol{x}}(V) - \boldsymbol{h}_\ell^{\mathrm{t}} \boldsymbol{y}$;
5 $\quad e^2 \leftarrow e^2 - \delta \beta^2$;
6 $\quad \hat{\boldsymbol{x}}(V \cup \{\ell\}) \leftarrow \hat{\boldsymbol{x}}(V \cup \{\ell\}) + \delta\beta \begin{bmatrix} \Theta\boldsymbol{\phi} \\ -1 \end{bmatrix}$;
7 $\quad \Theta \leftarrow \begin{bmatrix} \Theta & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} + \delta \begin{bmatrix} \Theta\boldsymbol{\phi} \\ -1 \end{bmatrix} \begin{bmatrix} \Theta\boldsymbol{\phi} \\ -1 \end{bmatrix}^{\mathrm{t}}$;
8 $\quad V \leftarrow V \cup \{\ell\}$;
9 **else**
10 $\quad j \leftarrow$ index of ℓ in V ;
11 $\quad e^2 \leftarrow e^2 + (\hat{\boldsymbol{x}}(\ell))^2 / \boldsymbol{\theta}_j(j)$;
12 $\quad \hat{\boldsymbol{x}}(V) \leftarrow \hat{\boldsymbol{x}}(V) - \hat{\boldsymbol{x}}(\ell)\, \boldsymbol{\theta}_j / \boldsymbol{\theta}_j(j)$;
13 $\quad \Theta \leftarrow \Theta(-j, -j) - \boldsymbol{\theta}_j(-j) \boldsymbol{\theta}_j(-j)^{\mathrm{t}} / \boldsymbol{\theta}_j(j)$;
14 $\quad V \leftarrow V \setminus \{\ell\}$;
15 **end**

where j is the index of ℓ in S and $\boldsymbol{e}_k$ is the vector of size $|S| \times 1$ whose k-th entry is equal to 1 and other entries are equal to 0. Note that P_j is an orthogonal matrix and its inversion is also a permutation matrix

$$P_j^{-1} = P_j^{\mathrm{t}} = [\boldsymbol{e}_1, \cdots, \boldsymbol{e}_{j-1}, \boldsymbol{e}_{j+1}, \cdots, \boldsymbol{e}_{|S|}, \boldsymbol{e}_j].$$

One can easily check that $H_S P_j$ moves the last column of H_S to the j-th column (and keeps the same order for the other columns) and $H_S P_j^{\mathrm{t}}$ moves the j-th column of H_S to the last column.

The calls to Algorithm 15 for updating ULS solutions at Lines 4-5 and 10-11 of Algorithm 13 take the respective forms:

$$\texttt{ULS}(\boldsymbol{y}, H, V, 1, \ell^+, \hat{\boldsymbol{x}}_V, \Theta, \|\boldsymbol{r}_V\|^2),$$
$$\texttt{ULS}(\boldsymbol{y}, H, V, 0, \ell^-, \hat{\boldsymbol{x}}_V, \Theta, \|\boldsymbol{r}_V\|^2).$$

Chapter 3

Numerical validation of NNOG algorithms

Contents

3.1 Introduction

In the previous chapter, we presented our unified framework of NNOG algorithms and proposed a (recursive) fast implementation. The purpose of this chapter is to assess the proposed fast implementation of NNOG algorithms by a set of numerical results. Firstly, we introduce the problems considered in our tests which include: (i) sparse deconvolution with Gaussian kernel, (ii) sparse deconvolution for super resolution and (iii) decomposition of real-world near-infrared (NIR) spectra into elementary Gaussian features. Secondly, we validate the accelerations of NNOG algorithms proposed in Section 2.4. Thirdly, we compare the performance of several NNOG algorithms. Finally, we compare NNOG algorithms with competing algorithms which include: (i) unconstrained greedy algorithms, (ii) fast approximate non-negative greedy algorithms, (iii) non-negative extensions of CoSaMP, SP, HTP and (iv) non-negative extension of LARS. The results presented in this chapter are achieved on a macOS X system with 16 GB RAM and Intel Core i7 processor at 2.7 GHz.

Before going further, let us emphasize that in applications (microscopy, tomography, etc...), the data are discrete. Therefore, two grids will be considered: (i) the sampling grid corresponding to the data domain (*e.g.,* the pixels of the radiograph in Tomo-PIV), and (ii) the grid corresponding to the sparse solution (*e.g.,* the voxels of the discretized 3-D volume in Tomo-PIV). Note that in sparse deconvolution both data and solution domain coincide (*e.g.,* the time domain). However, the practitioner may set the solution grid according to the desired resolution level.

3.2 Problems

3.2.1 Sparse deconvolution with Gaussian kernel

We consider a convolution problem with a normalized Gaussian kernel $\boldsymbol{h}$ of standard deviation σ. $\boldsymbol{h}$ is approximated by a finite impulse response of length 6σ by thresholding the smallest values. The discrete convolution $\boldsymbol{h} * \boldsymbol{x}^*$ can be rewritten as a matrix-vector product $H\boldsymbol{x}^*$ where H is a Toeplitz matrix containing all possible delayed version of the Gaussian kernel $\boldsymbol{h} = [h_1, \cdots, h_p]$, *i.e.*,

$$H = \begin{bmatrix} h_1 & 0 & \dots & 0 \\ \vdots & h_1 & \ddots & \vdots \\ h_p & \vdots & \ddots & 0 \\ 0 & h_p & \ddots & h_1 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & h_p \end{bmatrix}. \tag{3.1}$$

H is a slightly under-complete dictionary (of size $m \times n$ with $n = m - p + 1$) since we set the boundary condition so that the supports of all atoms are included in the observation window.

Simulated data are generated according to

$$\boldsymbol{y} = H\boldsymbol{x}^* + \boldsymbol{\epsilon} \tag{3.2}$$

where $\boldsymbol{x}^*$ and $\boldsymbol{\epsilon}$ stand for the ground truth coefficients and white Gaussian noise, respectively. The support S^* of $\boldsymbol{x}^*$, of cardinality K, is randomly generated with a uniform distribution whereas the non-zero coefficients of $\boldsymbol{x}^*$ are set to either a positive constant or randomly generated from a Gamma distribution. The signal-to-noise ratio is defined by

$$\mathrm{SNR} = 10 \log_{10} \left(P_{H\boldsymbol{x}^*} / P_{\boldsymbol{\epsilon}} \right) \tag{3.3}$$

where

$$P_{H\boldsymbol{x}^*} = \left\| H\boldsymbol{x}^* \right\|^2 / m \tag{3.4}$$

is the average power of the noise-free data and $P_{\boldsymbol{\epsilon}}$ is the noise variance.

A simple illustration of the data decomposition is given in Fig. 3.1. The data $\boldsymbol{y}$ is sampled with step equal to 1 and it is decomposed into three Gaussians $\boldsymbol{h}_1$, $\boldsymbol{h}_2$, $\boldsymbol{h}_3$ whose centers are located on the sampling grid. The Gaussians $\boldsymbol{h}_1$, $\boldsymbol{h}_2$, $\boldsymbol{h}_3$ are of the same width, thus each Gaussian is a delayed version of the first Gaussian. Fig. 3.2 shows an example of simulated data $\boldsymbol{y}$ corresponding to an SNR = 30 dB. The number of samples is set to $m = 300$. The Gaussian kernel is of width $\sigma = 5$. The ground truth $\boldsymbol{x}^*$ contains 10 spikes located at integer values.

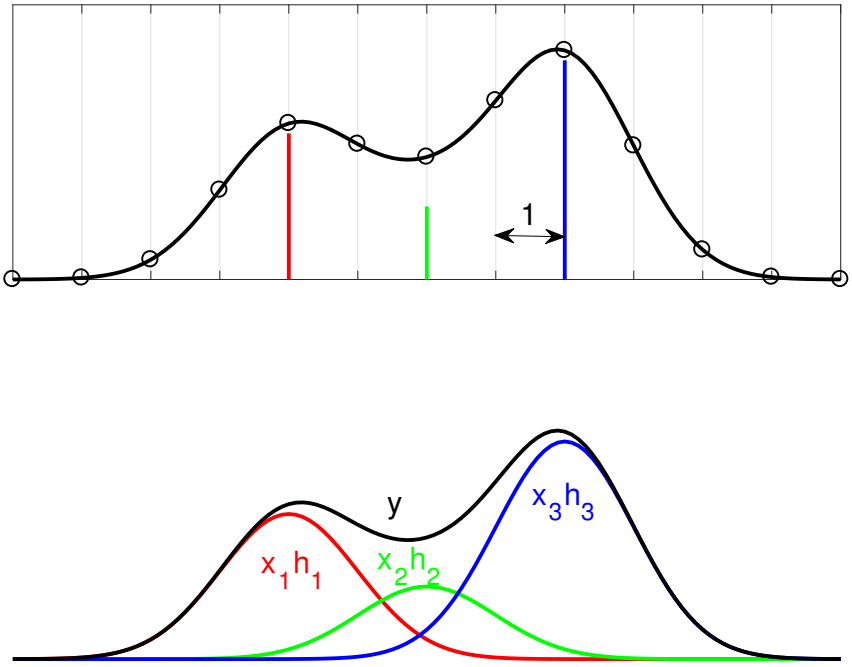


Figure 3.1: A simple illustration of sparse deconvolution with Gaussian kernel. (top) Discrete data are represented with bullets. Sampling step is set to 1. The three spikes (one in red, one in blue, one in green) corresponding to the centers of the constitute Gaussians $\boldsymbol{h}_1$, $\boldsymbol{h}_2$, $\boldsymbol{h}_3$ are located on the sampling grid. (bottom) The data $\boldsymbol{y}$ can be represented as a linear combination of three Gaussians $\boldsymbol{h}_1$, $\boldsymbol{h}_2$, $\boldsymbol{h}_3$ with corresponding weights x_1, x_2, x_3.

It should be noted that in this simulation the grid corresponding to the delays (locations of spikes) is the same as the one of the data signal $\boldsymbol{y}$ with step size equal to 1. In the generation of ground truth $\boldsymbol{x}^*$, the delays of Gaussians are set to integer values. Therefore, it is possible to get exact recovery results ($\boldsymbol{y} = H\boldsymbol{x}$) in the noise-free case ($\boldsymbol{\epsilon} = \boldsymbol{0}$) if the delays are correctly found by the algorithm.

All comparisons using this problem are based on reconstruction accuracy and CPU time. Reconstruction accuracy is quantified by three factors:

- *Support recovery:* ratio of true positives to K;
- *Coefficient inaccuracy:* relative error for the recovered coefficients

$$\|\boldsymbol{x} - \boldsymbol{x}^*\| \,/\, \|\boldsymbol{x}^*\| \,; \tag{3.5}$$

- *Residual norm:* Euclidean norm of the data residual ($\|\boldsymbol{y} - H\boldsymbol{x}\|$).

Note that here the accuracy criterion (3.5) makes sense because the true spikes are located at integer values. This criterion will be no longer applicable in the simulation of §3.2.2 since the true spikes in that simulation are located at real values.

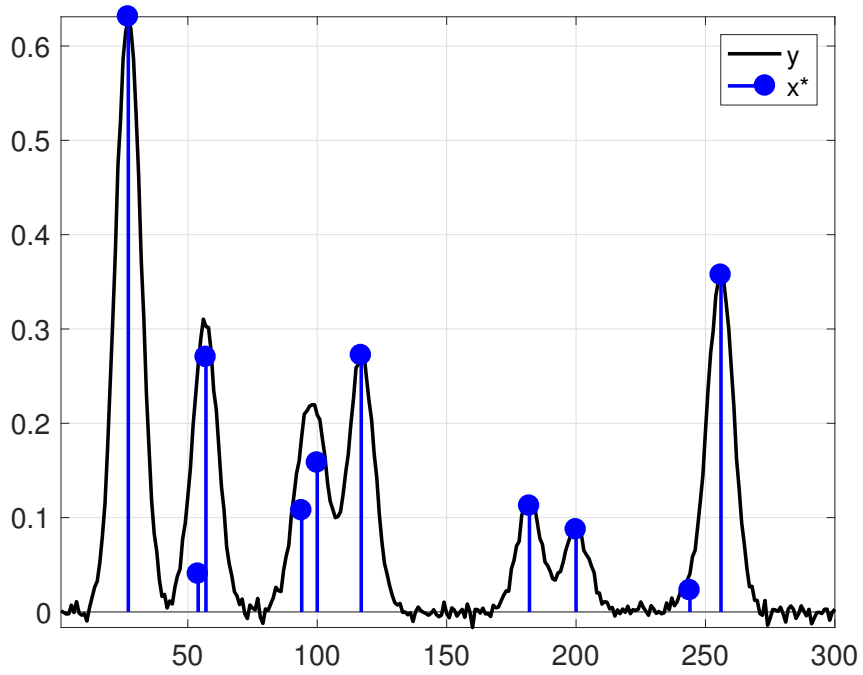


Figure 3.2: An example of simulated data and the corresponding ground truth in a sparse deconvolution problem with Gaussian kernel.

3.2.2 Sparse deconvolution for super resolution

While it is more convenient to impose that the true spike are located at integer values or on a predefined grid, this situation is not feasible in real-world applications. For instance, in Tomo-PIV application (see Chapter 1-§1.2.1), the spike locations correspond to the 3-D positions of the tracer particles in the volume. Thus, when the volume is discretized by a predefined grid, it is unrealistic to expect the spike locations to be located at the center of voxels. A recent trend is to work with a continuous dictionary (off-the-grid approach) (see, *e.g.,* [Ait Tilat et al., 2019, Elvira et al., 2019]). In the continuous approach, the dictionary contains an infinite number of atoms. Here we do not exploit the continuous approach. Instead, we elaborate on the case of a discrete dictionary corresponding to a fine grid.

In this simulation, we consider a more realistic setting where the spikes are located at real values. The continuous signal $y(\tau)$ is obtained by a convolution of a Gaussian function

$$h(\tau) = \exp\left(\frac{-\tau^2}{2\sigma^2}\right) \tag{3.6}$$

and a sparse spike train

$$x(\tau) = \sum_{j=1}^{K} x_j^* \delta(\tau - c_j^*) \tag{3.7}$$

where x_j^* are positive weights, $\delta(\tau)$ is the Dirac delta function and $c_j^* \in \mathbb{R}, j = 1, \ldots, K$ stands for the location of the K spikes. Hence, the noisy continuous signal can be expressed as

$$y(\tau) = (h * x)(\tau) + \text{noise} \tag{3.8}$$

$$= \sum_{j=1}^{K} x_j^* h(\tau - c_j^*) + \text{noise}. \tag{3.9}$$

We assume, without loss of generality, that the sampling step corresponding to the acquisition of the data $\boldsymbol{y}$ is equal to 1, so that the samples y_i correspond to the time $\tau = i$. Therefore, for

$i = 1, \cdots, m$ with m being the number of samples,

$$y_i = \sum_{j=1}^{K} x_j^* h(i - c_j^*) + \text{noise}. \tag{3.10}$$

Thus, the simulated data are generated according to

$$\boldsymbol{y} = \sum_{j=1}^{K} x_j^* \boldsymbol{h}_{c_j^*} + \boldsymbol{\epsilon} \tag{3.11}$$

where $\boldsymbol{h}_{c_j^*}$ stands for the discrete Gaussian of width σ and centered at c_j^*, and $\boldsymbol{\epsilon}$ stands for the white Gaussian noise. The true centers c_j^* and true weights x_j^* are randomly generated with a uniform distribution. Note that we restrict the true centers to

$$3\sigma \leq c_j^* \leq m - 3\sigma \tag{3.12}$$

so that the supports of all Gaussians are included in the observation window.

For sparse reconstruction, it is usual to choose a resolution grid of step size $\Delta = 1/\alpha$ where α is a positive integer. We approximate the data $\boldsymbol{y}$ in (3.11) by a linear combination of normalized discrete Gaussians $\boldsymbol{h}_{c_j}$ whose centers c_j are located on the resolution grid. It should be noted that in the previous simulation (§3.2.1), as the true centers c_j^* are located at integer values, one can expect the exact recovery (*i.e.*, $c_j = c_j^*$) in the noise-free case (*i.e.*, $\boldsymbol{\epsilon} = 0$). On the contrary, in the current simulation, as the true center c_j^* are located at real values, one can only expect that the estimated centers will be in the same cell as the true centers (*i.e.*, $|c_j - c_j^*| < \Delta/2$).

The related dictionary H contains all normalized discrete Gaussians $\boldsymbol{h}_{c_i}$ whose centers c_i are located on the resolution grid of step size $\Delta = 1/\alpha$. Thus, H has roughly αm atoms. However, since we set the boundary condition so that the supports of all atoms are included in the observation window, the number of atoms n is restricted to

$$n = \alpha \lfloor m - 6\sigma \rfloor - 1 \tag{3.13}$$

where $\lfloor \cdot \rfloor$ denotes the integer part. Note that H can be read as a block Toeplitz matrix up to reordering of its atoms. A simple illustration is shown in Fig. 3.3 where $m = 9$, $\sigma = 1$, $n = 5$, and step size $\Delta = 1/2$. H contains 5 normalized atoms:

$$H = [\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3, \boldsymbol{h}_4, \boldsymbol{h}_5] \tag{3.14}$$

with support of size $p = 7$. Each atom is a delayed version of the previous one by a step Δ. By reordering the atoms, H can be read as a block Toeplitz matrix $H = [H^{(1)}, H^{(2)}]$ where $H^{(1)} = [\boldsymbol{h}_1, \boldsymbol{h}_3, \boldsymbol{h}_5]$ and $H^{(2)} = [\boldsymbol{h}_2, \boldsymbol{h}_4]$. Since the dictionary H can be read as a block Toeplitz matrix, it can lead to fast computations [Carcreff, 2014]. It should be noted that in this simulation all atoms are of the same width. On the contrary, in the decomposition of NIR spectra (see §3.2.3), the Gaussian dictionary is built as a block Toeplitz matrix but each block is corresponding to the convolution matrix related to the Gaussian kernel of a different width.

3.2.3 Decomposition of NIR spectra into elementary Gaussian features

In the two previous problems, we have moved the simulated data from a basic framework (§3.2.1) to a more realistic one (§3.2.2). Here we deal with real-world near-infrared (NIR) data processing.

The data were obtained during Trispirabois project (2013-2016) gathering two public laboratories (CRAN and LCPME), three industrial companies (Egger, Pellenc ST and CrittBois) and funded by the french "FUI EcoIndustries 2012" program. The main objective of the project was

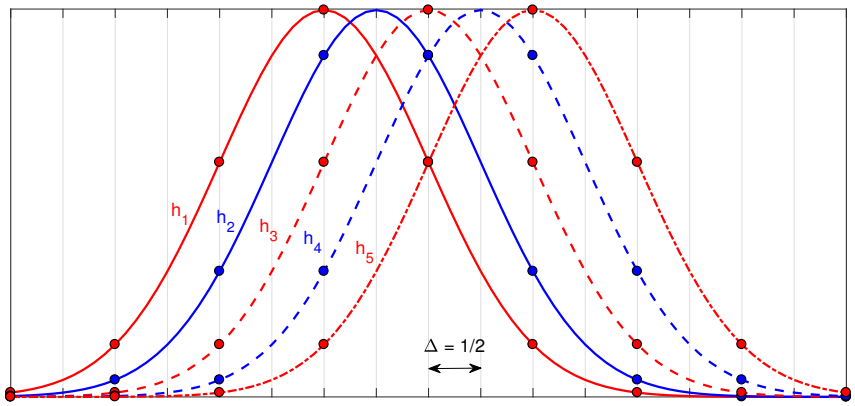


Figure 3.3: A simple illustration of the dictionary in super resolution problem. $H = [\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3, \boldsymbol{h}_4]$. Discrete atoms are represented by bullets. Sampling grid has step size 1 while the grid for sparse reconstruction has step $\Delta = 0.5$. Each atom is a delayed version of the previous atom by a step Δ.

to increase the rate of recycled wood in the production of new particleboards. In this respect, more than 300 wood samples with different composition (raw wood, plywood, particle boards, MDF-HDF) and different finition (raw, painted, varnished) were collected on a wood waste park and scanned using a Nicolet 8700 FTIR spectrometer. The resulting reflectance spectra are composed of 1647 wavenumbers covering the near-infrared (NIR) range 3600–10000 cm^{-1} (which correspond to wavelengths in the interval $[1, 2.8]$ μm). The aim is to design a binary classifier based on the identification of a subset of informative wavelengths for detecting the so-called non recyclable samples (*i.e.*, MDF-HDF) [Belmerhnia et al., 2015, Wagner et al., 2015]. As the final objective is to design an industrial (fast) sorting system, the number of selected wavelengths has to be as small as possible (typically between 16 and 32). Sparse modeling of the NIR spectra is a common approach to select the informative wavelengths. For instance, Turlach *et al.* [Turlach et al., 2005] proposed an algorithm for variable selection based on an extension of the LASSO and applied it to near infrared measurements.

The principle of wood classification by the decomposition of NIR spectra is summarized in Fig. 3.4. The different wood pieces are placed on a black background (conveyor) and scanned using a spectrometer. The resulted data can be represented as a cube with two spatial dimensions and one spectral dimension. One pixel of the data cube is a NIR spectrum composed of different wavelengths. One decompose the NIR spectrum (into elementary Gaussian features) to identify a subset of informative wavelengths in order to detect recyclable wood samples.

For convenience, let us start by introducing the sparse decomposition of one spectrum into elementary Gaussian features. Fig. 3.5 presents a simple illustration of such a decomposition. The simulated data (in black) is decomposed into three Gaussians of different widths σ ($\sigma = 10$ for the red Gaussian, 20 for the green and 30 for the blue). The noisy data is generated such that SNR = 30 dB. Fig. 3.6 presents another illustration of the decomposition with a focus on how the dictionary looks like. The dictionary is a block Toeplitz matrix

$$H = [H_1, \cdots, H_j, \cdots, H_q] \tag{3.15}$$

where each block H_j is the Toeplitz dictionary corresponding to the convolution matrix related to the Gaussian impulse response of width σ_j. In Fig. 3.6, H contains 3 blocks: red, green, blue (hence $q = 3$) where the red block corresponds to the smallest width σ and the blue block corresponds to the largest width.

Hereafter, we consider the decomposition of 50 NIR spectra, seen as data vectors $\boldsymbol{y}$ of length 1647. The data pre-processing includes baseline removal, offset correction ensuring zero lower

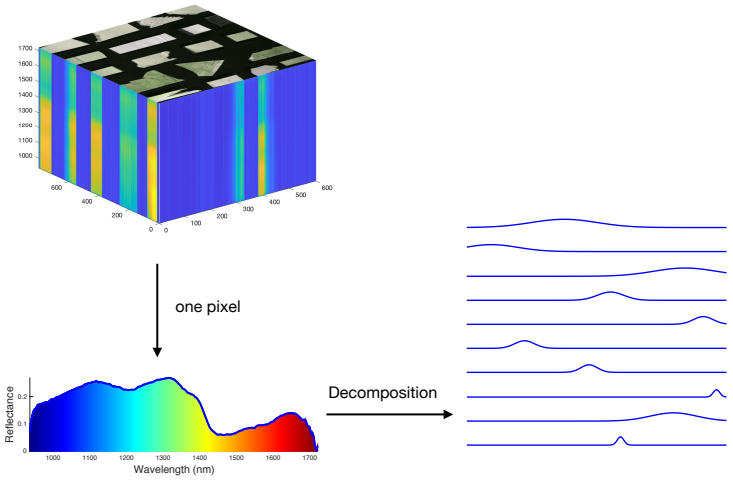


Figure 3.4: Principle of wood classification by NIR spectra decomposition.

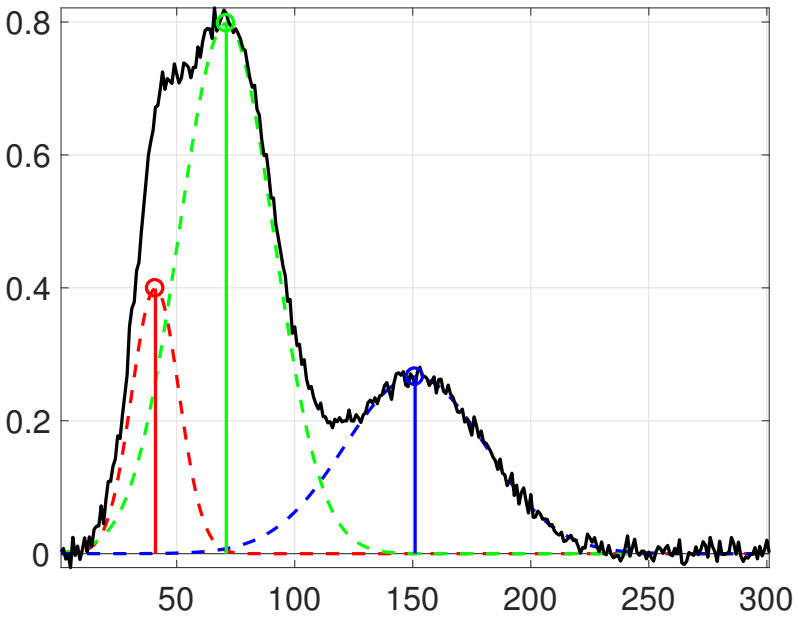


Figure 3.5: A simulation of the decomposition of a spectrum into elementary Gaussian features. The data signal (in black) is decomposed into three Gaussians (red, green and blue) of different widths (smallest in red and largest in blue).

bound, and unit energy normalization. One typical spectrum is shown in Fig. 3.7. To decompose the spectra, we build a dictionary H with Gaussian-shaped columns obtained by discretizing the parameters of a Gaussian function (centers and widths). This dictionary is formed by appending the columns of the convolution dictionaries (corresponding to a fixed width σ) used in Subsection 3.2.1 for 60 equally spaced values of $\sigma \in [10, 600]$ cm^{-1}. The generated dictionary is composed of 2998 atoms. Note that the centers of Gaussian atoms of same width σ are sampled with a step equal to σ, whereas the sampling step of the input signals $\boldsymbol{y}$ equals 4 cm^{-1}.

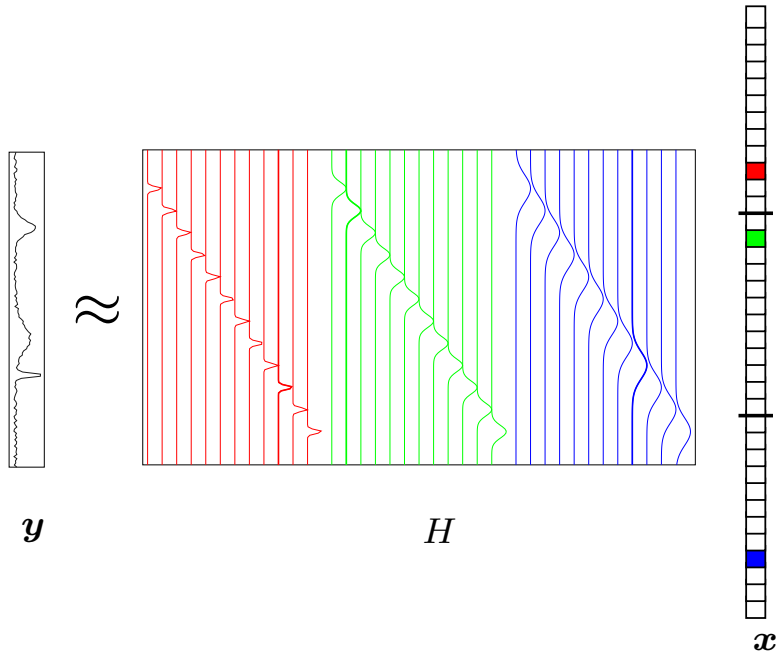


Figure 3.6: A simple illustration of sparse decomposition of one spectrum. The Gaussian dictionary consists of three Toeplitz blocks (red, green and blue) corresponding to the convolution matrices related to the Gaussian impulse response of given widths (smallest width in red and largest width in blue). The spectrum $\boldsymbol{y}$ can be decomposed into three spectra of different widths (one red, one green and one blue) with the corresponding weights in vector $\boldsymbol{x}$. The three spectral components are highlighted in bold and the non-zero weights are in color (red, green and blue). Figure was kindly shared by Dr. Hassan Mortada.

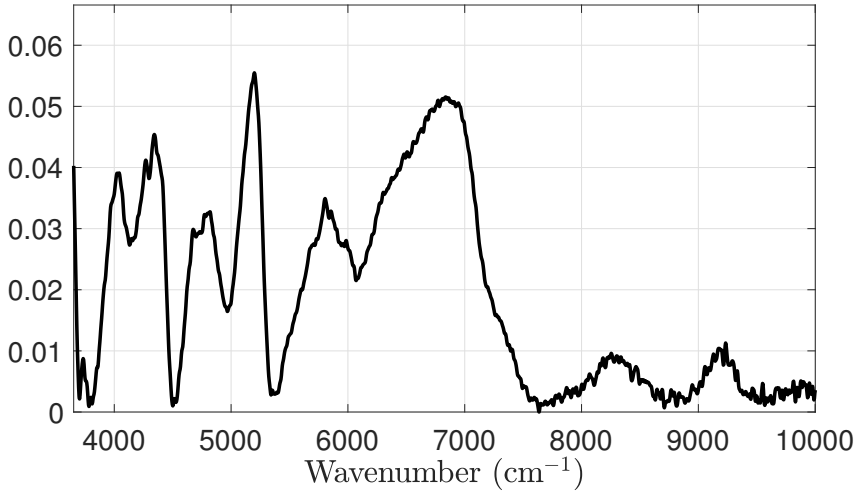


Figure 3.7: A typical NIR spectrum.

3.3 Validation of NNOG accelerations

In §2.4 we proposed accelerations for NNOG algorithms. Let us remind that NNOG algorithms can be made fully recursive by using active-set NNLS initialized with warm start. Moreover, several pruning strategies (type-I and type-II) can be combined to reduce the number of tests at the atom selection stage. Finally, the latter step can benefit from vectorized computations.

In this section we compare the accelerated implementation of NNOG algorithms with their

standard implementation in terms of computing time in a sparse deconvolution problem with Gaussian kernel (see §3.2.1 for description). Our comparison aims to illustrate the effect of the proposed accelerations in reducing the computing time of NNOG algorithms.

3.3.1 Sparse deconvolution with Gaussian kernel

We generate a convolution dictionary of size 1200×1140 (with $\sigma = 10$). For each value of $K \in \{20, 40, 60, 80\}$, 200 trials are carried out in which the support S^* is generated. The non-zero coefficients of $\boldsymbol{x}^*$ are set to 1, and the SNR is set to 30 dB. OMP, OLS, NNOMP, SNNOLS, and NNOLS are run until the support cardinality equals K. We consider the recursive implementation of OMP and OLS utilizing the matrix inversion lemma. We chose this implementation to make it consistent with our proposed NNOG implementation. Note that NNOG algorithms may need more than K iterations because of support compression. The following quantities are computed and averaged out over the number of trials:

- *Acceleration gain:* ratio, in terms of CPU time, between canonical (non-recursive) and accelerated recursive implementations of NNOG algorithms.
- *Non-negativity loss:* ratio, in terms of CPU time, between accelerated implementation of NNOG algorithms and the corresponding unconstrained versions.
- *Iterations:* average number of iterations of NNOG algorithms needed to yield a support of cardinality K. This number is larger than K if support compression occurs.

Before going further, let us clarify that the so-called "canonical implementations" refer to the following settings. The AS-NNLS algorithm is called from scratch from the initial zero solution. Moreover, obvious accelerations are taken into account such as type-I pruning, computation of the ULS solutions for candidate supports $S \cup \{i\}$ so as to avoid calling AS-NNLS when the augmented support is positive, and recursive computation of the AS-NNLS iterates. On the other hand, advanced accelerations such as type-II pruning and warm start initialization are not included. Support compression is included in such a way that both canonical and accelerated versions of a given NNOG algorithm yield the same iterates.

The scores can be found in Table 3.1. Accelerated implementations yield a gain in time by a factor greater than 2. For NNOMP, the gain increases with K. Since NNOLS needs to solve many NNLS subproblems per iteration, the gain is much larger. The time gain is intermediate for SNNOLS. We further notice that using accelerated implementations, the cost of NNOMP and SNNOLS becomes comparable with that of OMP and OLS, respectively, the non-negativity loss remaining below 1.5. Regarding NNOLS *vs* OLS, the non-negativity loss remains lower than 5 in these simulations. At last, the fact that the number of iterations is often larger than K reveals that support compression is happening quite often. Note that the number of times the support compression occurs does not identify to the gap between the number of iterations and K, since support compression may remove several atoms per iteration.

Table 3.1: Acceleration gain of NNOG algorithms for a sparse deconvolution problem with Gaussian kernel ($\sigma = 10$). Mean over 200 trials. The dictionary size is 1200×1140.

K	Acceleration gain			Non-negativity loss			Iterations		
	NNOMP	SNNOLS	NNOLS	NNOMP	SNNOLS	NNOLS	NNOMP	SNNOLS	NNOLS
20	2.7	10.5	792	1.2	1.0	1.5	20	22	21
40	4.5	11.5	1110	1.2	1.2	1.9	41	51	43
60	6.4	11.4	952	1.1	1.2	2.1	65	97	73
80	8.5	11.5	646	1.1	1.3	4.1	95	152	121

3.3.2 Computation burden of SNNOLS and NNOLS

Hereafter, the computation burden of SNNOLS and NNOLS is assessed more thoroughly so as to evaluate the accelerations proposed in Section 2.4. Four indicators are computed at each iteration:

1. $\rho_{\downarrow} = |\mathcal{D}_S|/|\bar{S}|$: rate of descending atoms. The rate of discarded atoms after type-I pruning reads $1 - \rho_{\downarrow}$.

2. ρ^{+}_{cand}: rate of descending candidate atoms for which $S \cup \{i\}$ is a positive support.

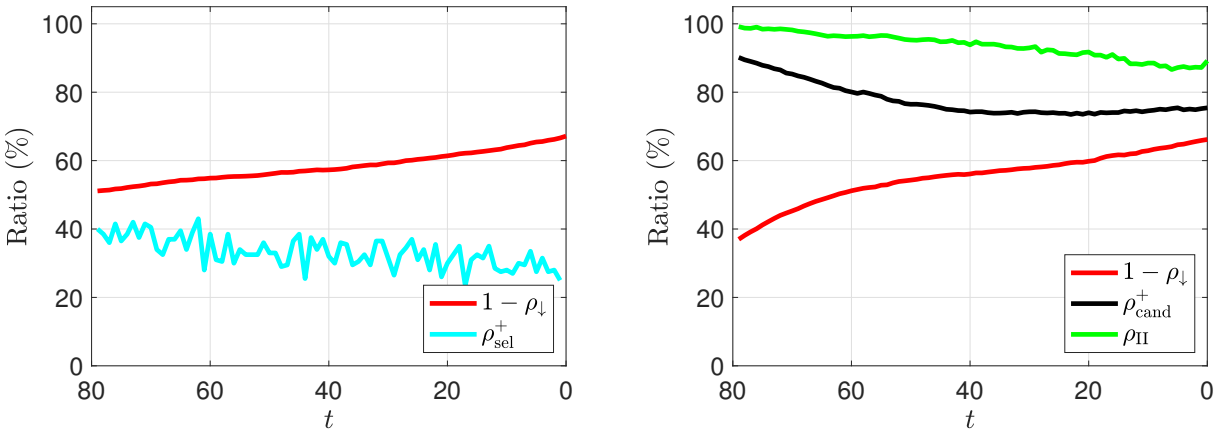
3. ρ_{II}: rate of candidate atoms discarded by type-II pruning among all atoms i for which $S \cup \{i\}$ is not a positive support.

4. ρ^{+}_{sel}: rate of *selected* atoms yielding a positive support $S \cup \{\ell\}$.

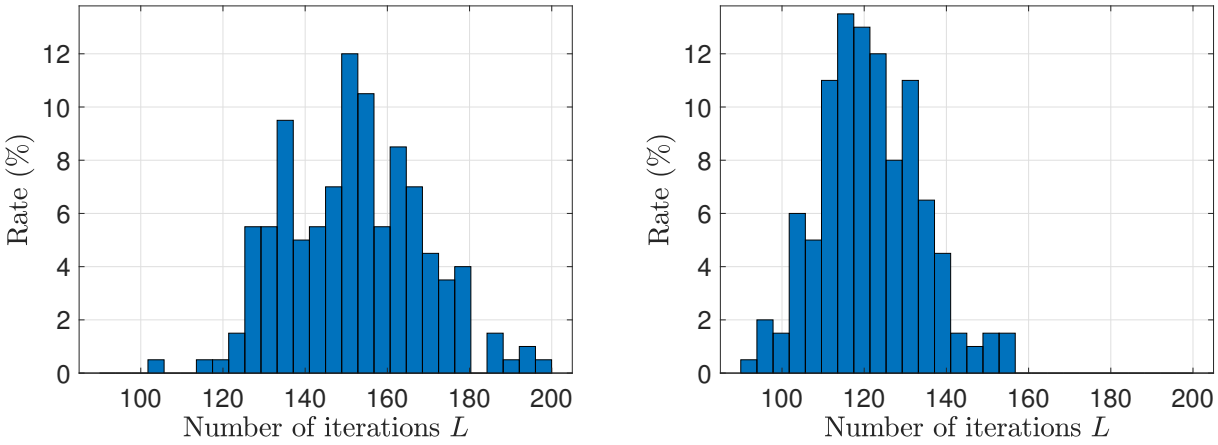
The underlying idea is that the computational cost of an NNOG iteration is closely related to the values of these ratios. Indeed, large scores indicate that the cost of testing candidate atoms is dramatically reduced. Specifically, $\rho_{\downarrow}$ is the rate of candidate that are truly considered in the selection rule (2.15) of NNOG algorithms. Both ratios ρ^{+}_{cand} and ρ_{II} quantify the computational burden of the NNOLS selection step: large values of ρ^{+}_{cand} and ρ_{II} indicate that AS-NNLS has to be run for a few candidate atoms only, since other atoms are either pruned or yield a non-negative ULS solution. ρ^{+}_{sel} is defined similar to ρ^{+}_{cand}. However, ρ^{+}_{sel} does not apply to the candidate atoms, but to the selected atoms, in order to bring information on the computational burden of the coefficient update stage of SNNOLS.

Using the dictionary of size 1200×1140 and the settings $K = 80$ and SNR $= 30$ dB, the computational burden of SNNOLS and NNOLS is assessed in Fig. 3.8. It is noticeable that the number of iterations L required to reach a support of cardinality K is larger than K because of support compression. Specifically, the histograms of Fig. 3.8(b) show that on average, L is larger than K for NNOLS and even larger for SNNOLS (the average values are given in the last lines and columns of Table 3.1). This is consistent with the fact that the NNOLS selection rule is more involved but more reliable. Moreover, the standard deviation of L corresponding to the histograms of Fig. 3.8(b) is 17 and 12 for SNNOLS and NNOLS, respectively, which indicates that the size of the support found after k iterations may significantly vary between trials. In order to get meaningful evaluations, we choose to compute the average values of each indicator over the last t iterations, with $t \in \{0, \dots, 79\}$. When $t = 0$, only the last iteration is taken into account, so the current support is of size K. For larger values of t, the supports found during the last t iterations have varying sizes, but the averaging operation remains meaningful, especially for the last iterations, which are the most costly. The curves displaying the average of each indicator over 200 trials and over the last t iterations are shown in Fig. 3.8(a). One can observe from the curve $(1-\rho_{\downarrow})$ that descending atoms are numerous at early iterations, and then their rate gradually decreases for both SNNOLS and NNOLS. Therefore, type-I pruning is more effective at late iterations, where it discards about half of the atoms.

The effect of type-II pruning in NNOLS is measured by ρ^{+}_{cand} and ρ_{II} indicators in Fig. 3.8(a). Large values of ρ^{+}_{cand} and ρ_{II} are obtained, which indicates that the computing cost of NNOLS is dramatically reduced. Recall that a large value of ρ_{II} implies that type-II pruning eliminates most atoms in $\mathcal{D}_S$ that do not result in a positive support. The decrease of ρ^{+}_{cand} implies that the rate of descending atoms not resulting in a positive support gets larger as the number of iteration increases. However, type-II pruning remains truly effective. For instance, in the last iterations, around 25% of descending atoms do not yield a positive support and type-II pruning eliminates 90% of them, that is, 22% of descending atoms. As a result, only 3% of descending atoms require to proceed a complete NNLS solving. Let us stress that the extra cost of NNOLS



(a) Complexity indicators of SNNOLS (left) and NNOLS (right)



(b) Histograms of SNNOLS (left) and NNOLS (right)

Figure 3.8: Complexity analysis of SNNOLS and NNOLS for a simulated sparse deconvolution problem with Gaussian kernel ($\sigma = 10$) and a SNR of 30 dB. The dictionary size is 1200×1140, $K = 80$ and 200 trials are performed. (a) Evolution of complexity factors during the last t iterations (from $L - t$ to L). $\rho_{\downarrow}$: ratio of descending atoms; ρ^{+}_{cand}: ratio of atoms resulting in a positive support in NNOLS; ρ_{II}: ratio of atoms discarded by type-II pruning in NNOLS; ρ^{+}_{sel}: rate of positive supports found by SNNOLS. (b) Histogram of the average number of iterations L required to reach a support of cardinality $K = 80$.

as compared with SNNOLS essentially comes from the number of atoms related to a complete NNLS solving, so it directly depends on the efficiency of type-II pruning.

Besides, the decrease of the indicators ρ^{+}_{sel}, ρ^{+}_{cand} and ρ_{II} highlights that both SNNOLS and NNOLS call NNLS more often at the late iterations. A reason for this behavior is that the correlation between atoms becomes larger when the set of considered atoms enlarges.

3.3.3 Conclusion

The accelerated implementation of NNOG algorithms are significantly faster than the standard implementation and becomes comparable with that of unconstrained greedy algorithms. Therefore it is reasonable to use NNOG algorithms in any non-negative sparse problem instead of the traditional OMP and OLS.

3.4 Comparison of three NNOG algorithms

In this section we compare the computing time and performance of the three NNOG algorithms (NNOMP, NNOLS and SNNOLS, see Chapter 2 for their descriptions) in two simulated problems: (i) sparse deconvolution with Gaussian kernel, and (ii) sparse deconvolution for super resolution. The readers are referred to §3.2.1-§3.2.2 for the description of these problems.

3.4.1 Sparse deconvolution with Gaussian kernel

The curves shown in Fig. 3.9 are obtained for a convolution dictionary of size 2400×2388, with $\sigma = 2$ and for various settings of $K \in [2, 80]$. For each K, 400 trials are carried out in which the support S^* is drawn according to the uniform distribution and the non-zero coefficients of $\boldsymbol{x}^*$ are drawn from a Gamma distribution with shape and scale parameters set to 1 and 2, respectively. Note that for increasing K, the density of spikes increases, and hence the difficulty of the problem. Working at a given level of SNR would have even increased the difficulty at large values of K, so we have preferred to keep the noise level $P_{\boldsymbol{\epsilon}}$ constant; it is set to $P_{\boldsymbol{\epsilon}} = 10^{-2}$. In this simulation condition, it is noticeable in Fig. 3.9(b) that the coefficient inaccuracy $\|\boldsymbol{x} - \boldsymbol{x}^*\|$ becomes somewhat large when the spike density is high. For easier problems (lower value of σ and lower density of spikes), many algorithms would provide good results, so it would be difficult to see differences in terms of estimation quality.

In this test, NNOMP is faster but less accurate than NNOLS and SNNOLS. SNNOLS has roughly the same performance as NNOLS but with lower cost which is close to that of NNOMP. Illustration on one trial is shown in Fig. 3.10.

3.4.2 Super resolution problem

The noiseless data of size $m = 300$ are obtained by a linear combination of $K = 40$ discrete Gaussian features of width $\sigma = 2$ and centered at continuous locations c_j^*. The true centers c_j^* and weights x_j^* are randomly generated with uniform distribution. Then the data signal $\boldsymbol{y}$ is obtained by adding a white Gaussian noise corresponding to a SNR=30 dB. The resolution grid is chosen with step size $\Delta = 2^{-N}$ where N varies from 1 to 7. For each value of N (hence each value of Δ), the corresponding dictionary H is built from all possible delayed versions of the Gaussian of width σ. The centers of atoms are located on the grid of chosen step Δ.

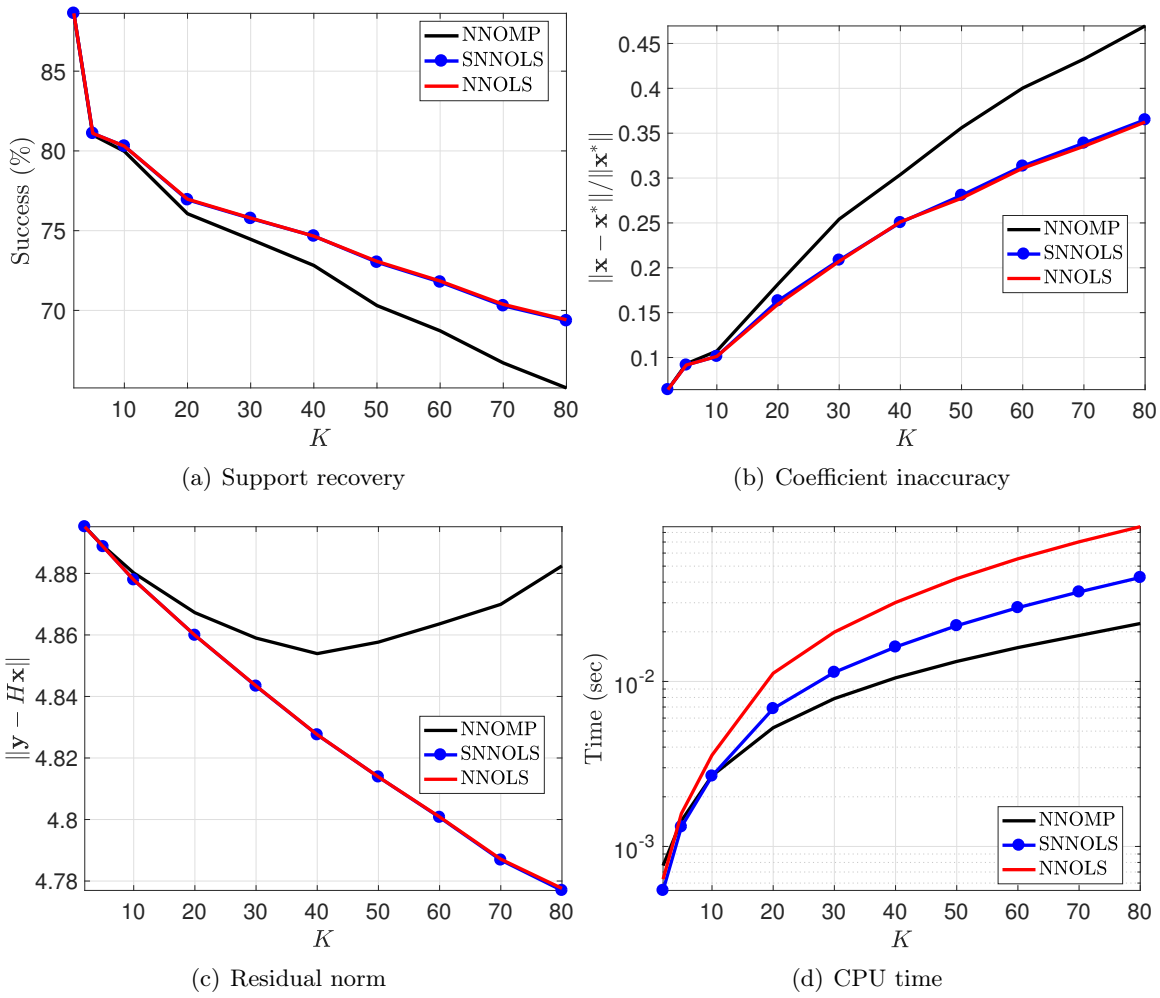
NNOMP, NNOLS and SNNOLS are run until a support of cardinality K is reached. We compare the three algorithms in terms of CPU time, the number of iterations and the reconstruction accuracy which is characterized by three factors:

- *Support recovery*: ratio of true positives to K
- *Relative residual norm*: $\|\boldsymbol{y} - H\boldsymbol{x}\|/\|\boldsymbol{y}\|$
- *Normalized support distance*:

$$\left(\sum_i \min_j |c_i^* - c_j| + \sum_j \min_i |c_i^* - c_j| \right) /(2K) \tag{3.16}$$

  where c_j are estimated locations of the true centers c_j^*.

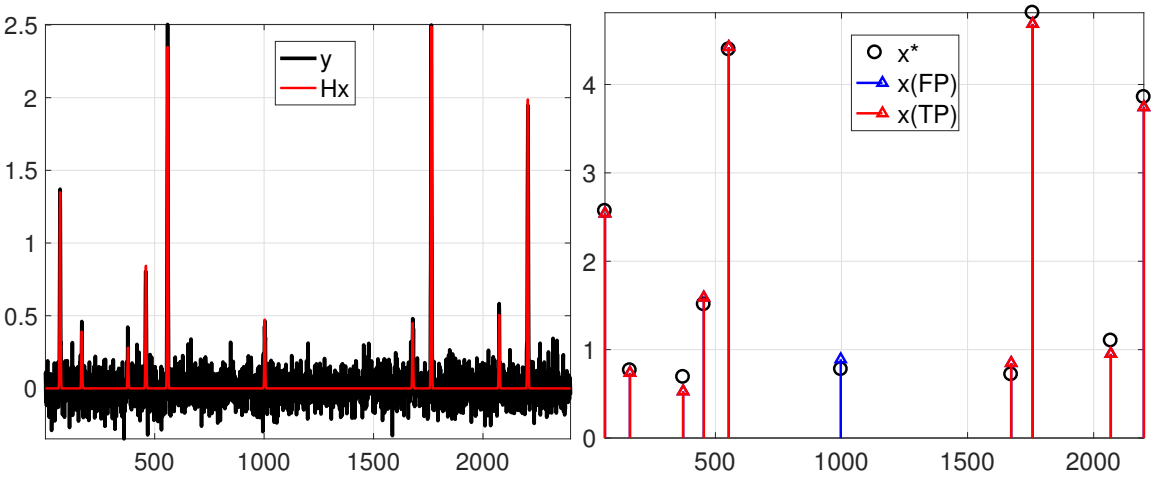
Note that (3.16) is a symmetric measure of the minimum distance (in average) between the location of one true spike and the corresponding estimated location. Its value is always non-zero since the true spike locations c_i^* are set to real values, hence, they are almost surely located off the resolution grid containing the estimated locations c_i.



(a) Support recovery (b) Coefficient inaccuracy

(c) Residual norm (d) CPU time

Figure 3.9: Comparison of NNOG algorithms. Performances are averaged over 400 trials in a simulated sparse deconvolution problem with Gaussian kernel with $\sigma = 2$, $P_{\boldsymbol{n}} = 10^{-2}$, and a dictionary of size 2400×2388.



(a) Data and approximation (b) Ground truth and recovery

Figure 3.10: One trial of data $\boldsymbol{y}$ and ground truth $\boldsymbol{x}^*$ when $K = 10$ and the reconstruction using NNOMP. NNOMP detects correctly 9 spikes (in red) and yields one false detection (in blue).

One example of simulated data and the average performance of the three algorithms over 100 trials are shown in Figure 3.11. NNOMP yields the weakest performance while NNOLS and SNNOLS perform competitively. We observed that, for larger value of N, NNOLS yields a lower support error in comparison with SNNOLS. This means that NNOLS has a better capacity in locating true spikes even when the dictionary is highly correlated (*i.e.*,, when the grid step size is small, the consecutive atoms are very similar hence it is more difficult to distinguish). Besides, when N is larger than 5, SNNOLS becomes more costly than NNOLS as it requires many more iterations.

3.4.3 Conclusion

Among the three NNOG algorithms, NNOMP is the fastest but yields the weakest performance. NNOLS has the best capacity in locating the true spikes, specially when dealing with highly correlated dictionaries. SNNOLS has the best trade of between computing time and reconstruction accuracy.

3.5 Comparison with unconstrained greedy algorithms

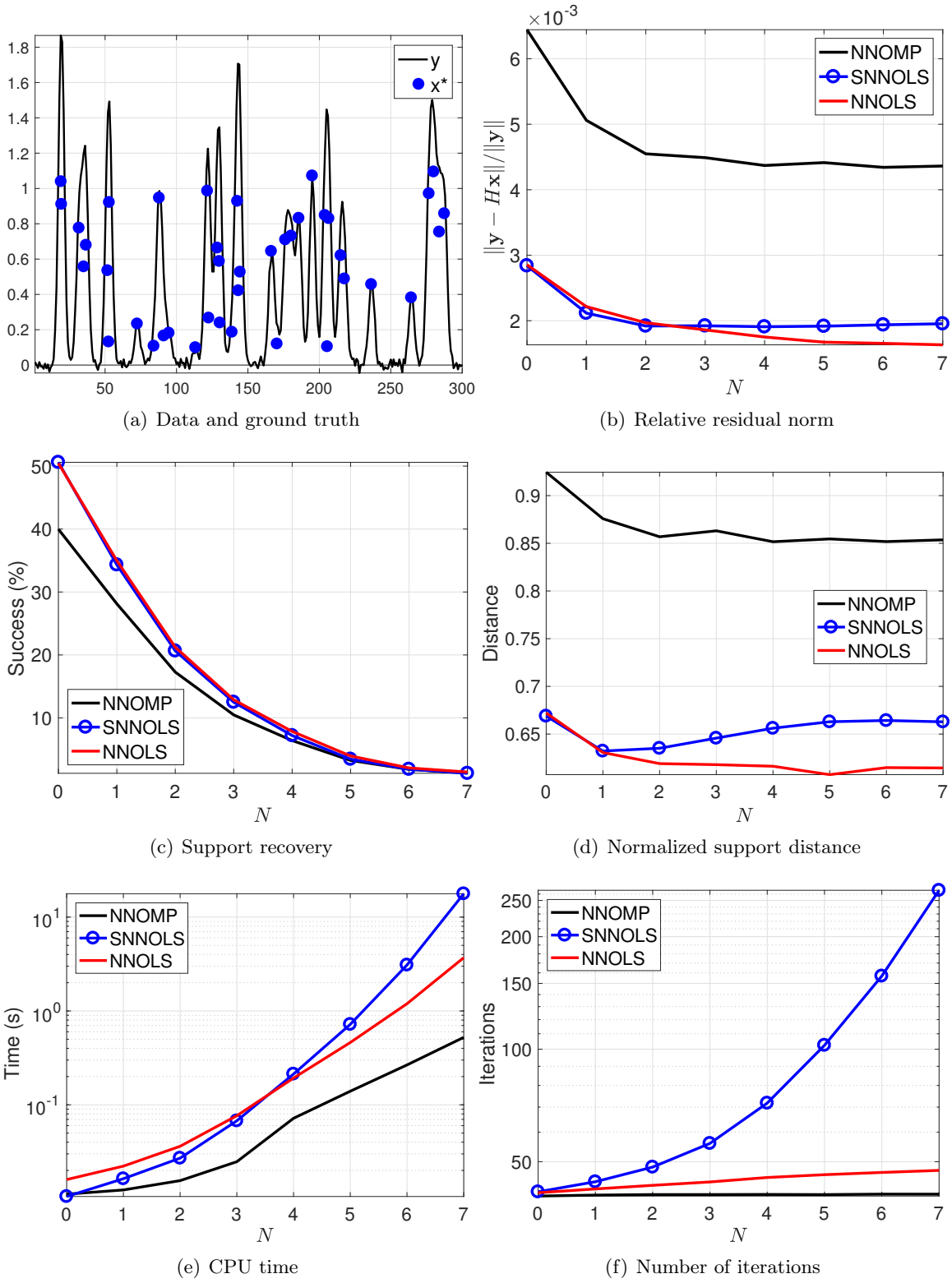
In this section we perform comparisons between NNOG algorithms and (unconstrained) OMP and OLS to demonstrate the usefulness of working with NNOG algorithms. The comparisons are done in a simulated sparse deconvolution problem with Gaussian kernel (described in §3.2.1) and in the decomposition of real-world NIR spectra into elementary Gaussian features (described in §3.2.3). In order to obtain non-negative coefficients from the OMP (respectively, OLS) outputs, a simple possibility is to apply a post-processing step using the solution support yielded after K iterations of OMP (resp., OLS). This post-processing consists of solving the NNLS problem related to the support found by OMP/OLS, with possible annealing of coefficients when non-negativity constraints are activated. The resulting schemes are denoted by OMP+ and OLS+.

3.5.1 Sparse deconvolution with Gaussian kernel

We use the same setting as in §3.4.1. Figure 3.12 includes a comparison of NNOMP and NNOLS with OMP, OLS, OMP+ and OLS+. The performance of OMP, OLS, OMP+ and OLS+ in terms of support recovery and coefficient accuracy is weaker than those of NNOG algorithms. In Figure 3.12(a), the scores of OMP+ (OLS+) are identical to those of OMP (OLS). This is because the support recovery measure considers the rate of true positives, which is unchanged since the post-processing step of OMP+ (OLS+) essentially removes false positives. On the contrary, the coefficient inaccuracy ratio is improved due to the latter removal. In Figure 3.12(c-d), the time and error scores of OMP (resp., OLS) are both lower than those of NNOMP (resp., SNNOLS and NNOLS). This is not a surprise since unconstrained algorithms are simpler, and the obtained solutions are expected to reach lower values of the residual since they do not satisfy the non-negativity constraint. The simple post-processing in OMP+/OLS+ does not allow to yield a residual norm as small as the one obtained using NNOG. This shows that NNOG algorithms do improve the performance of (unconstrained) OMP and OLS.

3.5.2 Decomposition of NIR spectra

For each NIR spectrum, NNOMP, NNOLS, OMP, OLS are run until a support of cardinality $K = 20$ is reached. The simple OMP+ and OLS+ algorithms are considered as well. For each competing algorithm, the CPU time and the relative residual norm $\|\boldsymbol{y} - H\boldsymbol{x}\|/\|\boldsymbol{y}\|$, averaged over 50 spectra, are displayed in Tab. 3.2. First, the time and approximation errors of OMP (resp., OLS) are both lower than those of NNOMP (resp., NNOLS). This is not a surprise since



(a) Data and ground truth

(b) Relative residual norm

(c) Support recovery

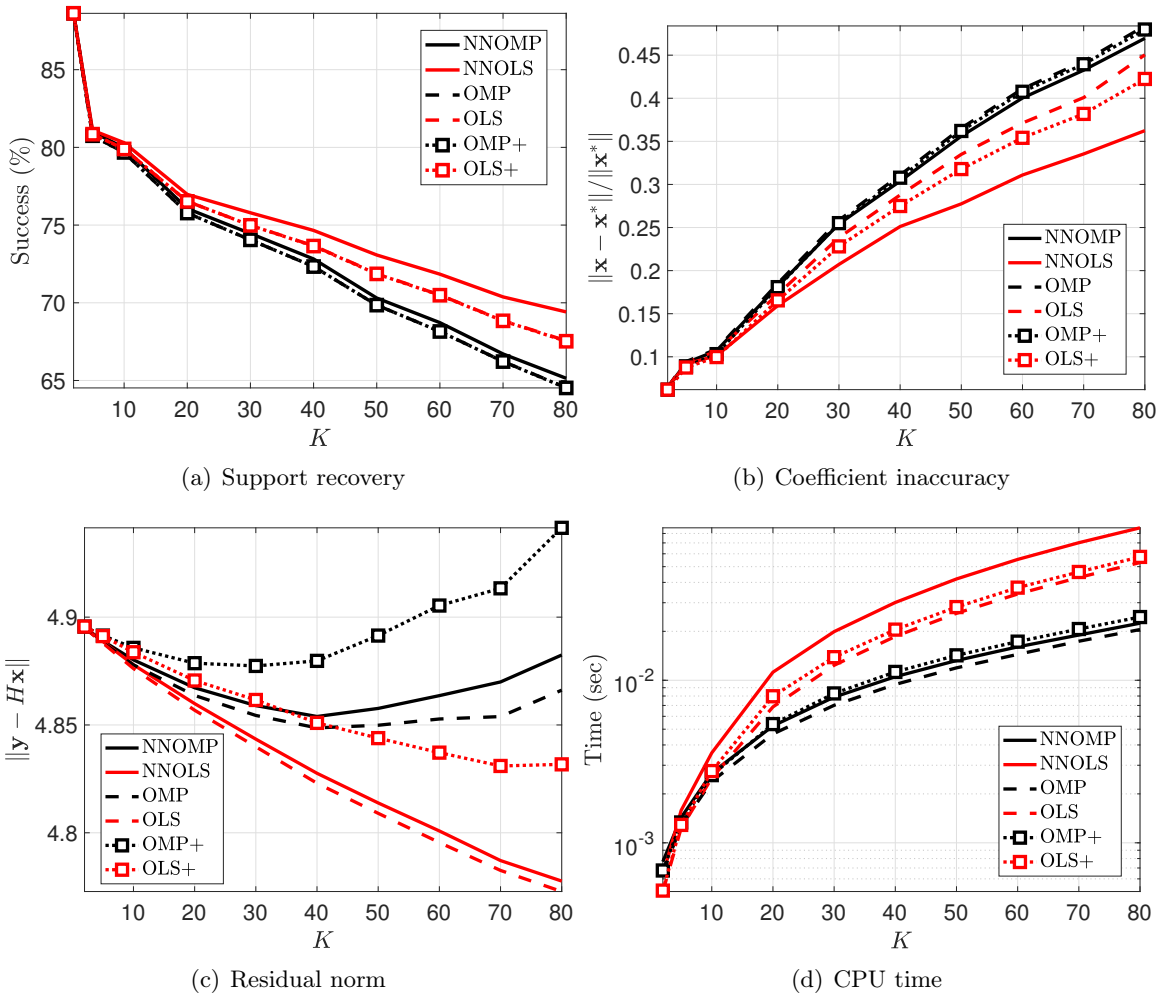
(d) Normalized support distance

(e) CPU time

(f) Number of iterations

Figure 3.11: Comparison between NNOMP, NNOLS and SNNOLS in a super resolution problem where resolution grid step is set to $\Delta = 2^{-N}$ for N varying between 1 and 7.

unconstrained algorithms are expected to reach lower criterion values. When one performs a post-processing by removing these negative spikes such as with OMP+/OLS+, the error scores become larger than those of NNOG algorithms, and the sparse approximation accuracy is weaker. This confirms the weak capacity of OMP and OLS to reconstruct correct supports from non-



(a) Support recovery

(b) Coefficient inaccuracy

(c) Residual norm

(d) CPU time

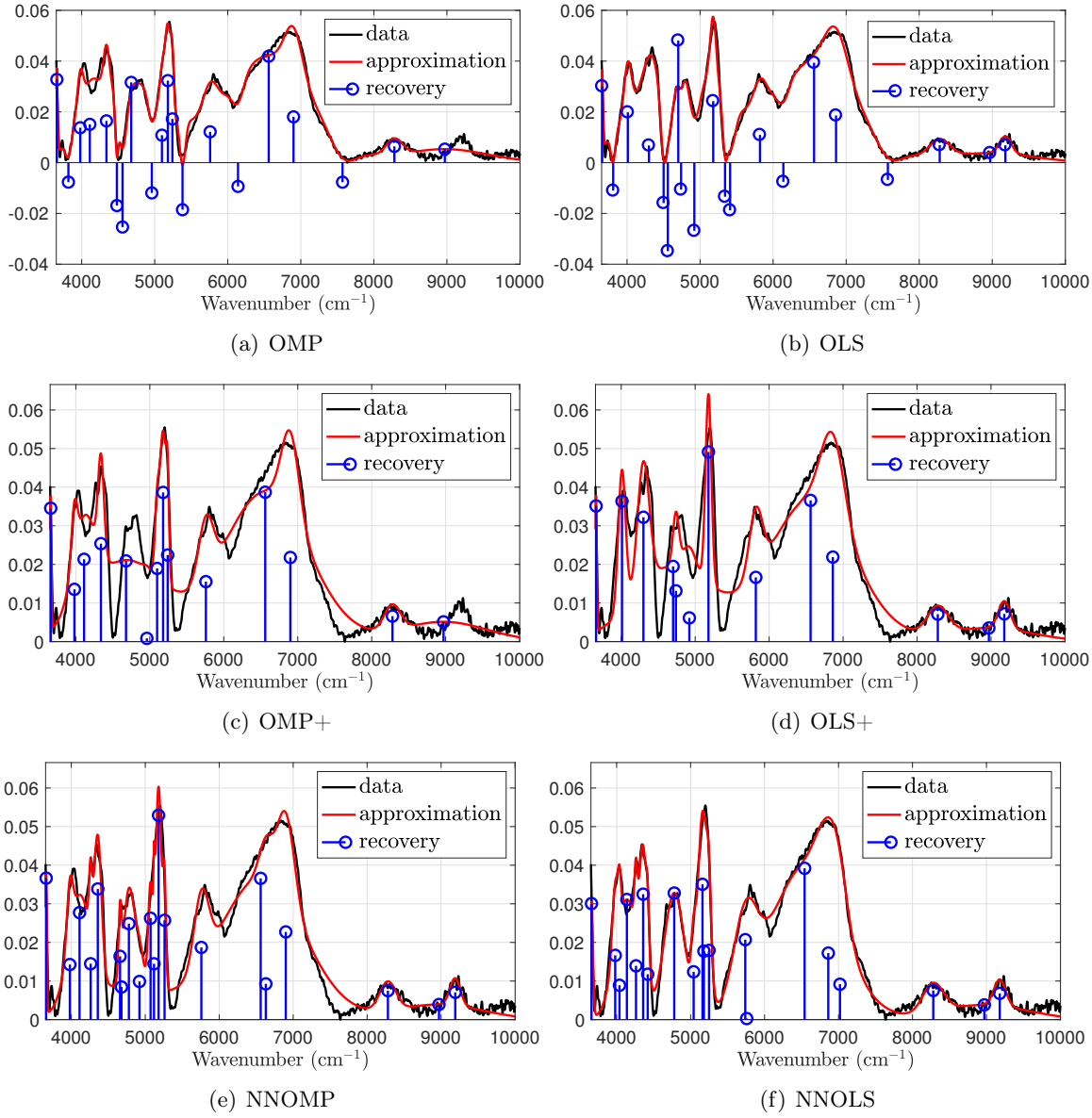
Figure 3.12: Comparison of NNOG algorithms with unconstrained greedy algorithms. Performances are averaged over 400 trials in a simulated sparse deconvolution problem with Gaussian kernel with $\sigma = 2$, $P_{\boldsymbol{n}} = 10^{-2}$, and a dictionary of size 2400×2388.

negative sparse representations. Finally, the CPU times of NNOG algorithms *vs* their related unconstrained versions given in Tab. 3.2 are consistent with the non-negativity losses gathered in Tab. 3.1.

Table 3.2: CPU time and normalized approximation error of NNOG algorithms and unconstrained greedy algorithms for sparse decomposition of NIR spectra. Average over 50 spectra. Symbol ** indicates that the considered algorithm does not enforce the non-negative constraint.

Algorithm	**OMP	OMP+	NNOMP	**OLS	OLS+	NNOLS
Time (ms)	23	24	28	25	26	40
$\|\boldsymbol{y}-H\boldsymbol{x}\|/\|\boldsymbol{y}\| \times 10^2$	6.6	16.4	9.6	5.7	16.6	6.4

These results are further illustrated for a specific spectrum in Fig. 3.13, where approximations and sparse recoveries are displayed for all competing algorithms. OMP and OLS yield 7 and 9 negative peaks, respectively. Besides, the OMP+ and OLS+ approximations around 4500–7000 cm^{-1} and 5000 cm^{-1}, respectively, are rather poor. NNOLS outperforms NNOMP, in particular, around 4200 and 7000 cm^{-1}.



(a) OMP (b) OLS

(c) OMP+ (d) OLS+

(e) NNOMP (f) NNOLS

Figure 3.13: Data approximation and sparse recovery of an NIR spectrum using NNOG algorithms and unconstrained greedy algorithms. The Gaussian dictionary contains 2998 atoms and the sparsity level is set to $K = 20$.

3.5.3 Conclusion

OMP and OLS have weak capacities of reconstructing non-negative ground truth. The simple post-processing by removing negative coefficients is not sufficient to convert OMP and OLS to non-negative sparse solvers. NNOMP and NNOLS significant improve the performance of OMP and OLS in the reconstruction of non-negative ground truth. Moreover, the computing time of NNOMP and NNOLS becomes comparable to that of OMP and OLS. Therefore, it is reasonable to use NNOMP and NNOLS in any non-negative sparse problem instead of OMP and OLS.

3.6 Comparison with fast approximate non-negative greedy algorithms

In this section we compare NNOG algorithms with FNNOMP [Yaghoobi et al., 2015] and FNNOLS [Yaghoobi and Davies, 2015]. Let us remind that FNNOMP was introduced in [Yaghoobi et al., 2015] as a fast (but approximative) implementation of NNOMP. Similarly, FNNOLS was introduced in [Yaghoobi and Davies, 2015] as a fast implementation of NNOLS. Our comparisons are done in a simulated sparse deconvolution problem with Gaussian kernel (described in §3.2.1) and in the decomposition of real-world NIR spectra into elementary Gaussian features (described in §3.2.3).

3.6.1 Sparse deconvolution with Gaussian kernel

We use the same setting as in §3.4.1. Figure 3.14 presents a comparison with the FNNOMP and FNNOLS implementations[1] of Yaghoobi *et al.* [Yaghoobi et al., 2015, Yaghoobi and Davies, 2015]. FNNOLS turns out to be much slower and less accurate than our fast implementations of SNNOLS and NNOLS. On the other hand, the statistical performances of NNOMP and FNNOMP are very close both in terms of computational time and accuracy. However, FNNOMP may return some negative coefficients in the reconstructed sparse vector. In contrast, our fast implementation of NNOMP is exact and the non-negativity constraint is always satisfied.

3.6.2 Decomposition of NIR spectra

For each NIR spectrum, NNOG algorithms, FNNOMP and FNNOLS are run until a support of cardinality $K = 20$ is reached. The average computing time and normalized approximation error over 50 spectra are shown in Tab. 3.3. As can be seen, NNOMP and FNNOMP have roughly same cost and performance. FNNOLS is more costly than NNOLS and SNNOLS with a slightly weaker performance. Illustration on a spectrum is shown in Fig. 3.15.

Table 3.3: CPU time and normalized approximation error of different implementations of NNOG algorithms for sparse decomposition of NIR spectra. Average over 50 spectra.

Algorithm	NNOMP	FNNOMP	SNNOLS	NNOLS	FNNOLS
Time (ms)	28	25	30	40	2720
$\|\boldsymbol{y} - H\boldsymbol{x}\|/\|\boldsymbol{y}\| \times 10^2$	9.6	9.6	6.4	6.4	7.0

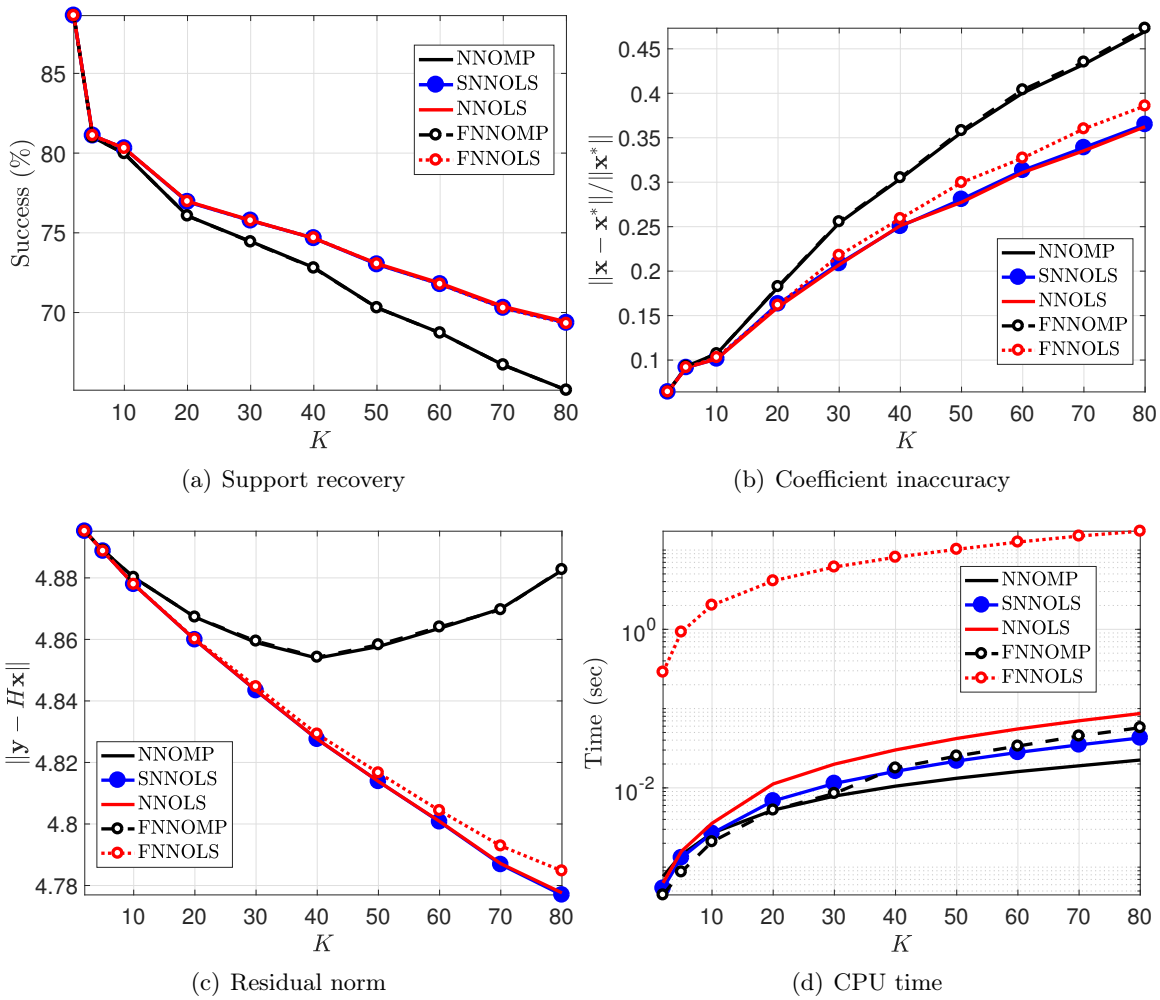
3.6.3 Conclusion

FNNOMP has roughly the same cost and performance as our exact implementation of NNOMP. However, FNNOMP might return some negative coefficients. Besides, FNNOLS is much slower than NNOLS and SNNOLS with slightly weaker performance. Therefore, we conclude that it is not necessary to work on approximative schemes to derive efficient implementations.

3.7 Comparison with non-negative extensions of CoSaMP, SP and HTP

In this section we compare NNOG algorithms with non-negative extensions of CoSaMP [Needell and Tropp, 2009], SP [Dai and Milenkovic, 2009] and HTP [Foucart, 2011] which are named NNCoSaMP, NNSP and NNHTP [Kim and Haldar, 2016], respectively. Our comparisons are

[1] kindly provided by Dr. Mehrdad Yaghoobi.



(a) Support recovery

(b) Coefficient inaccuracy

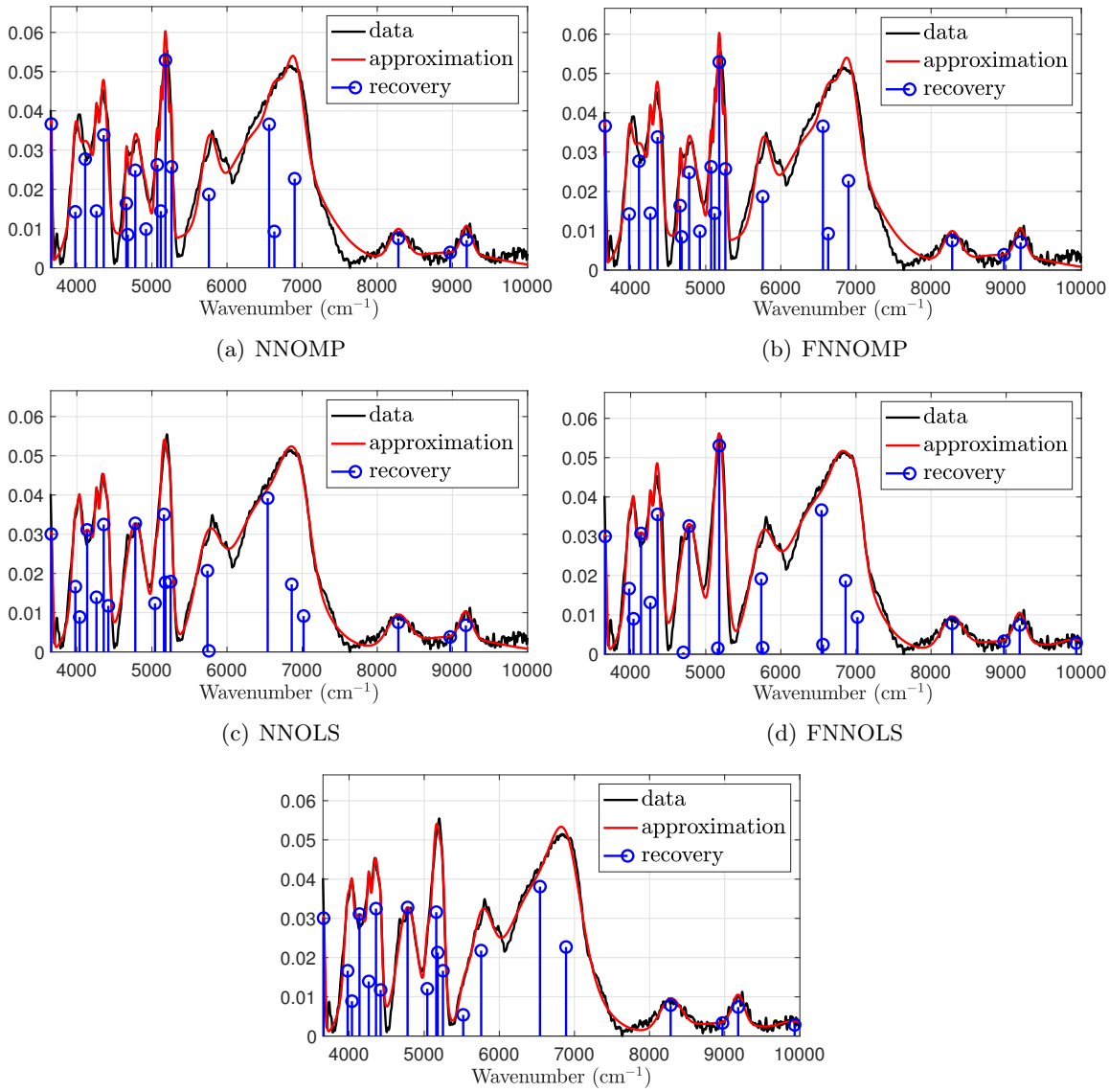
(c) Residual norm

(d) CPU time

Figure 3.14: Comparison of NNOG algorithms with with approximative fast non-negative greedy algorithms. Performances are averaged over 400 trials in a simulated sparse deconvolution problem with Gaussian kernel with $\sigma = 2$, $P_{\boldsymbol{n}} = 10^{-2}$, and a dictionary of size 2400×2388.

done in a simulated sparse deconvolution problem with Gaussian kernel (described in §3.2.1) and in the decomposition of real-world NIR spectra into elementary Gaussian features (described in §3.2.3).

Let us remind that the principle of NNCoSaMP and NNSP [Kim and Haldar, 2016] is to maintain an estimated support of size K. To do so, three kinds of operations are performed at each iteration including support merging, NNLS estimation of coefficients and support pruning. In NNCoSaMP, the current support of size K is merged with a support of size $2K$, and an NNLS problem is solved with this augmented support. Then, thresholding is performed by keeping the K coefficients having the largest magnitudes. The structure of NNSP is similar, but two NNLS problems (of size roughly $2K$ and K) are solved before and after thresholding. The NNSP output thus contains at most K nonzero elements depending on the activation of nonnegativity constraints. NNHTP [Kim and Haldar, 2016] maintains an estimated support of size K by keeping the K coefficients having the largest magnitudes from a gradient descend move then solving the NNLS problem related to the maintained support. Therefore, NNHTP output also contains at most K nonzero elements.



(a) NNOMP (b) FNNOMP

(c) NNOLS (d) FNNOLS

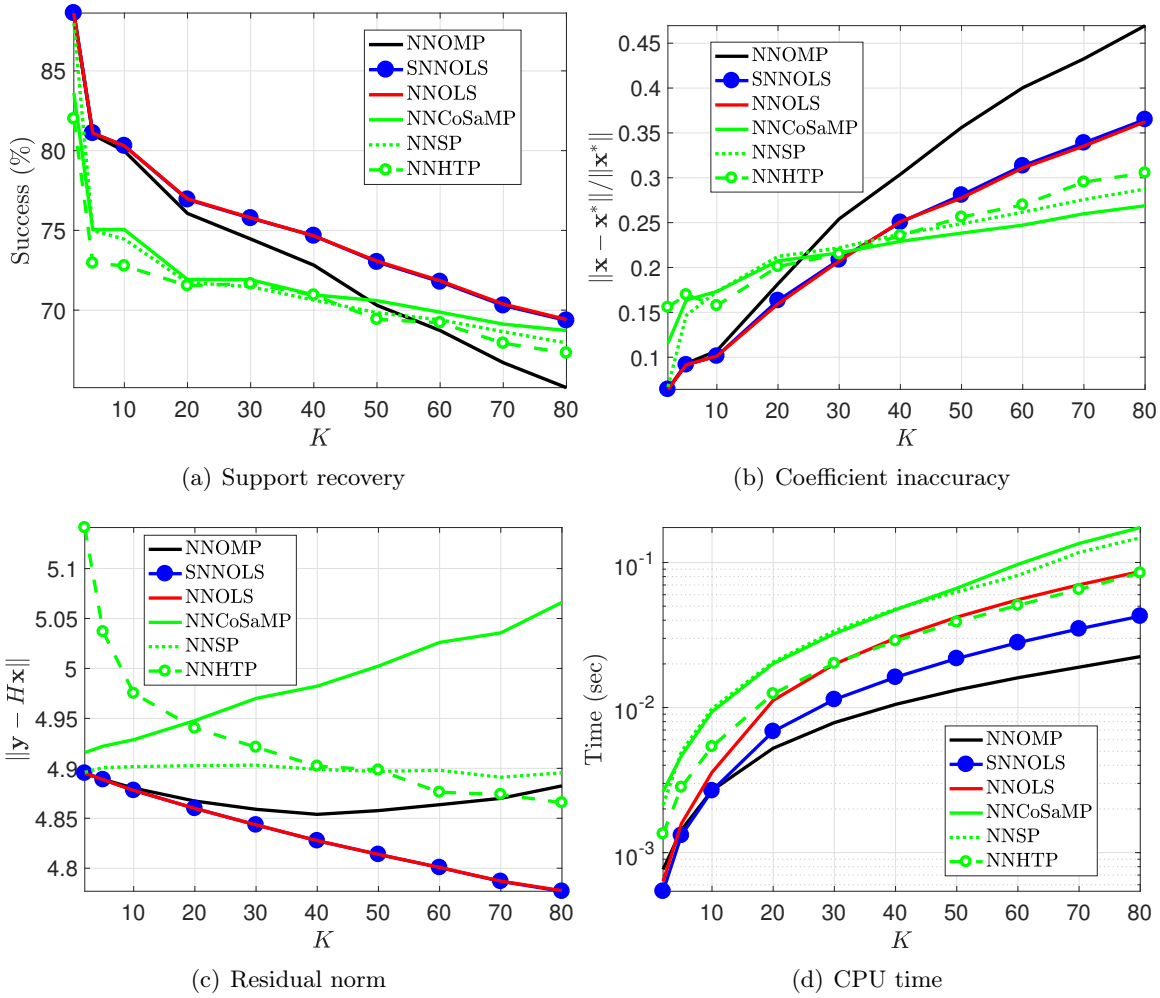
(e) SNNOLS

Figure 3.15: Data approximation and sparse recovery of an NIR spectrum using NNOG algorithms and approximative fast versions. The Gaussian dictionary contains 2998 atoms and the sparsity level is set to $K = 20$.

3.7.1 Sparse deconvolution with Gaussian kernel

We use the same setting as in §3.4.1 and the result is shown in Figure 3.16. As can be seen, the NNOG algorithms yield competitive accuracy performance compared to NNCoSaMP and NNSP while NNHTP yields the weakest performance. In particular, NNOG algorithms have a better ability to find a low value of the residual norm for a given cardinality K. This said, no algorithm outperforms all competitors in terms of coefficient accuracy for all scenarios. Nonetheless, the times of computation of NNOG with our recursive implementations are always lower than those of NNCoSaMP, NNSP and NNHTP. Let us stress that the structure of the latter algorithms does not easily lend itself to recursive implementations. Indeed, there is no nested property between the supports found at consecutive iterations. Consequently, solving the NNLS problems in a very efficient manner does not seem obvious anymore. For instance, NNCoSaMP needs to solve NNLS problems for augmented supports of size roughly $3K$. The current K-sparse vector obtained after thresholding may be used as a warm start (although it is usually not an NNLS solution), but

the number of inner NNLS iterations is expected to be much larger than for NNOG algorithms.



(a) Support recovery

(b) Coefficient inaccuracy

(c) Residual norm

(d) CPU time

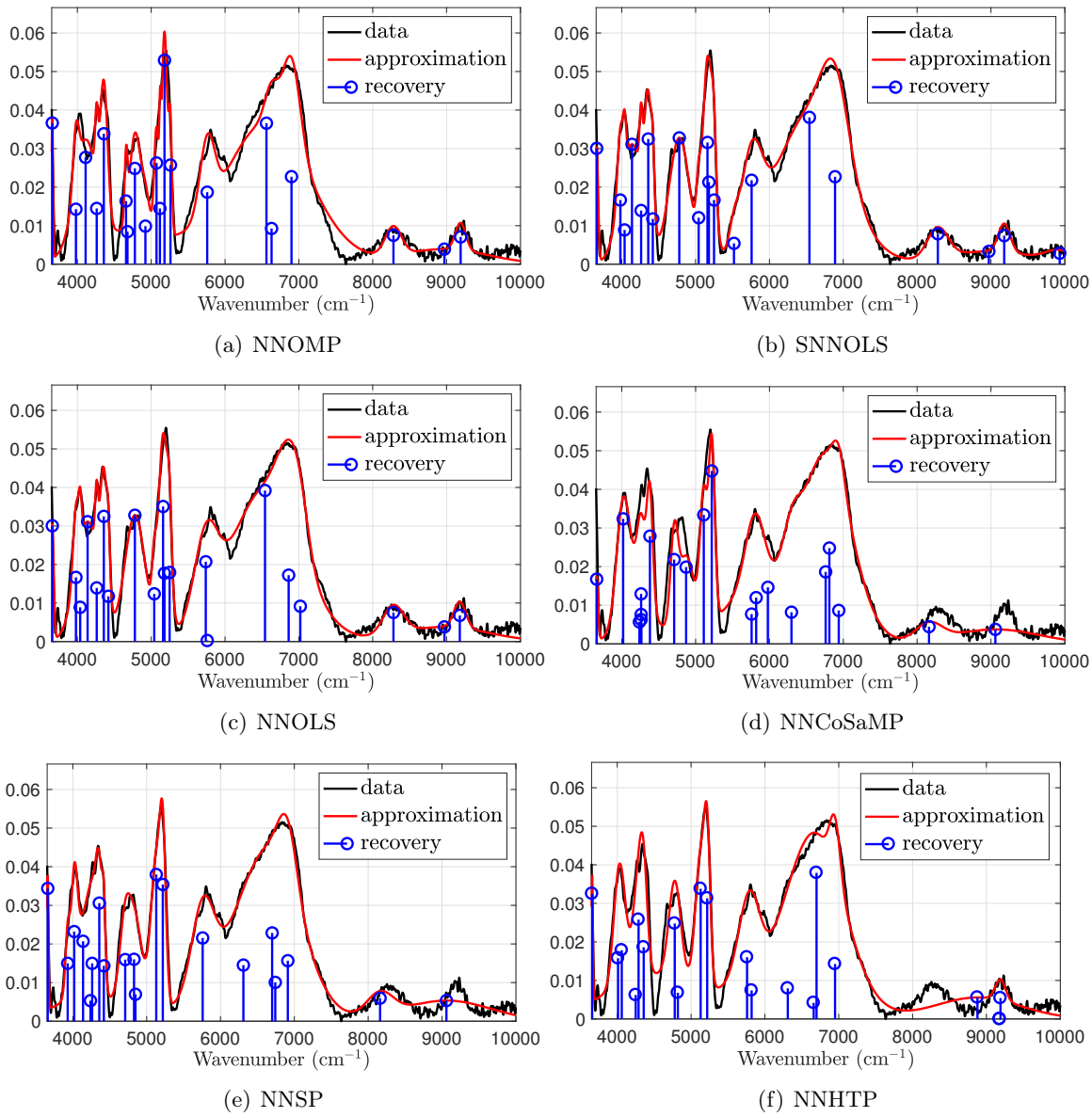
Figure 3.16: Comparison of NNOG algorithms with non-negative versions of CoSaMP, SP, HTP. Performances are averaged over 400 trials in a simulated sparse deconvolution problem with Gaussian kernel with $\sigma = 2$, $P_{\boldsymbol{n}} = 10^{-2}$, and a dictionary of size 2400×2388.

3.7.2 Decomposition of NIR spectra

For each NIR spectrum, NNOG algorithms, NNCoSaMP, NNSP and NNHTP are run until a support of cardinality $K = 20$ is reached. The average computing time and normalized approximation error over 50 spectra are shown in Tab. 3.4. NNCoSaMP and NNSP are slower than NNOG algorithms and yield higher errors. NNHTP is at the same cost as NNOLS but with a weaker performance. This result is further illustrated in Fig. 3.17. In this example, SNNOLS and NNOLS outperform NNCoSaMP and NNSP around 9000 cm^{-1} while NNHTP performance is rather poor.

Table 3.4: CPU time and normalized approximation error of several non-negative greedy algorithms for sparse decomposition of NIR spectra. Average over 50 spectra.

Algorithm	NNOMP	SNNOLS	NNOLS	NNCoSaMP	NNSP	NNHTP
Time (ms)	28	30	40	45	57	40
$\|\boldsymbol{y}-H\boldsymbol{x}\|/\|\boldsymbol{y}\| \times 10^2$	9.6	6.4	6.4	10.1	8.2	9.7



(a) NNOMP (b) SNNOLS

(c) NNOLS (d) NNCoSaMP

(e) NNSP (f) NNHTP

Figure 3.17: Data approximation and sparse recovery of an NIR spectrum using non-negative greedy algorithms. The Gaussian dictionary contains 2998 atoms and the sparsity level is set to $K = 20$.

3.7.3 Conclusion

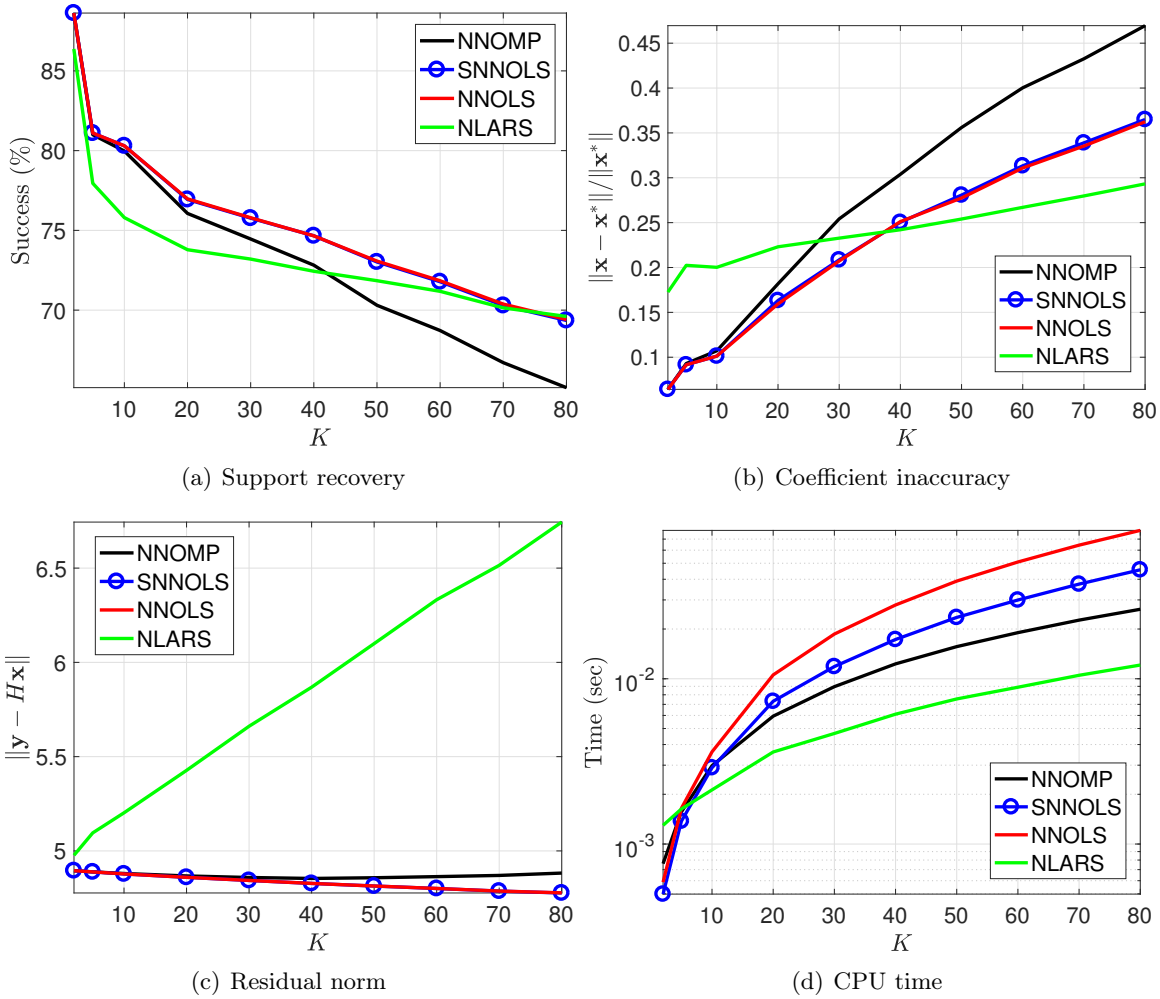
NNOG algorithms yields competitive performances to those of NNCoSaMP, NNSP and NNHTP. However, NNOG algorithms can easily lend themselves to recursive implementation while it is not the case for NNCoSaMP, NNSP and NNHTP.

3.8 Comparison with non-negative extension of LARS

In this section we compare NNOG algorithms with non-negative extensions of LARS [Efron et al., 2004] named NLARS [Morup et al., 2008]. Note that LARS is an ℓ_1 solver with a greedy structure. Each iteration of NLARS consists of either a forward move or a backward move and the update of amplitudes along a computed direction. Our comparisons are done in a simulated sparse deconvolution problem with Gaussian kernel (described in §3.2.1) and in the decomposition of real-world NIR spectra into elementary Gaussian features (described in §3.2.3).

3.8.1 Sparse deconvolution with Gaussian kernel

We use the same setting as in §3.4.1 and the result is shown in Figure 3.18. The NNOG algorithms and NLARS yield competitive performances in terms of support recovery and coefficient accuracy. However, NNOG algorithms have a better ability to find a low value of the residual for a given cardinality K while NLARS is faster.



(a) Support recovery

(b) Coefficient inaccuracy

(c) Residual norm

(d) CPU time

Figure 3.18: Comparison of NNOG algorithms with NLARS. Performances are averaged over 400 trials in a simulated sparse deconvolution problem with Gaussian kernel with $\sigma = 2$, $P_{\boldsymbol{n}} = 10^{-2}$, and a dictionary of size 2400×2388.

3.8.2 Decomposition of NIR spectra

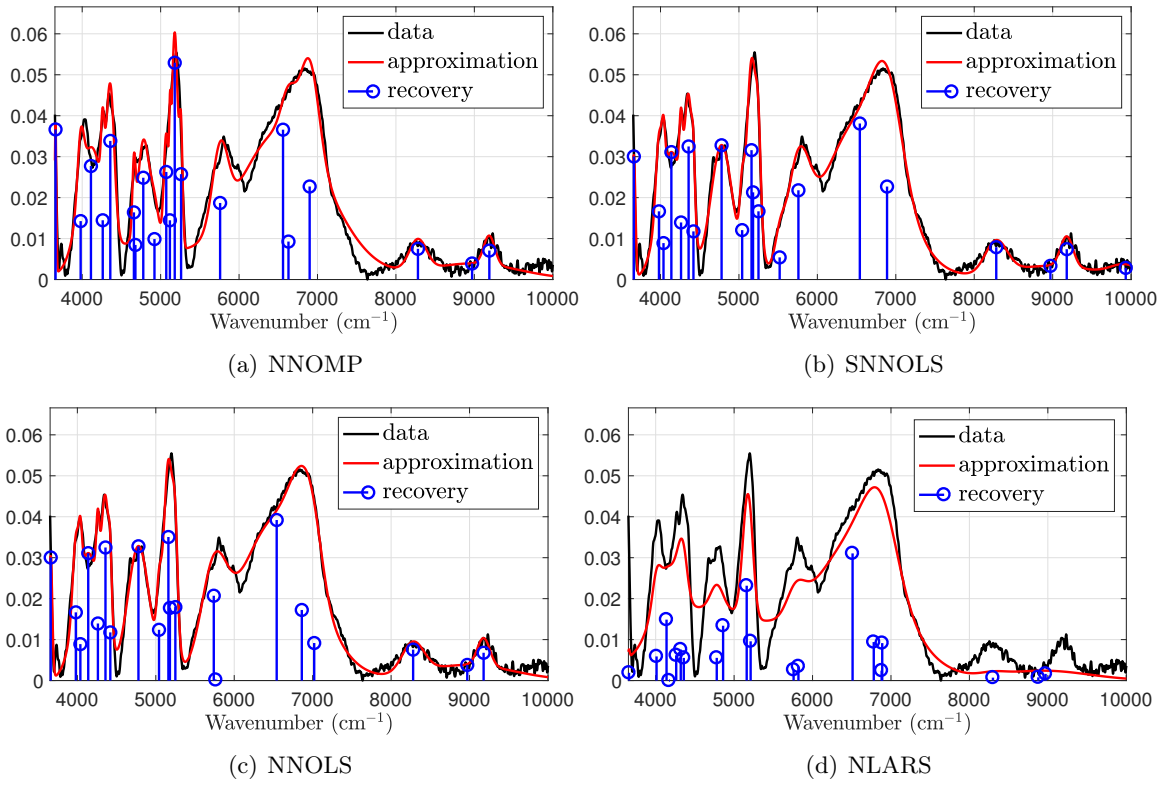
For each NIR spectrum, NNOG algorithms and NLARS are run until a support of cardinality $K = 20$ is reached. The average computing time and normalized approximation error over 50 spectra are shown in Tab. 3.5. Similar to the previous comparison (§3.8.1), NLARS yields a larger approximation error than those of NNOG algorithms. However, in this comparison, NLARS is slower than NNOG algorithms. We noticed that the number of iterations of NLARS is much higher than those of NNOG algorithms (50 iterations in average for NLARS in comparison with 21 for NNOG algorithms). This might come from the fact that the dictionary in this problem is over-complete (with roughly twice columns as rows, hence highly correlated) while the one in §3.8.1 is slightly under-complete. Since, at each iteration of NLARS, the residual is not orthogonal to the selected subspace, NLARS might need several iterations to select and deselect

the same atoms before finding a good candidate to append to the current support. Therefore, NLARS can be more costly than NNOG algorithms even though each NLARS iteration might be cheaper than that of NNOG algorithm.

This comparison is further illustrated in Fig. 3.19. As can be seen, the approximation resulted by NLARS is much poorer than those of NNOG algorithms.

Table 3.5: CPU time and normalized approximation error of NLARS and NNOG algorithms for sparse decomposition of NIR spectra. Average over 50 spectra.

Algorithm	NNOMP	SNNOLS	NNOLS	NLARS
Time (ms)	28	30	40	77
$\|\boldsymbol{y} - H\boldsymbol{x}\|/\|\boldsymbol{y}\| \times 10^2$	9.6	6.4	6.4	21.1



(a) NNOMP (b) SNNOLS

(c) NNOLS (d) NLARS

Figure 3.19: Data approximation and sparse recovery of an NIR spectrum using NLARS and NNOG algorithms. The Gaussian dictionary contains 2998 atoms and the sparsity level is set to $K = 20$.

3.8.3 Conclusion

NNOG algorithms and NLARS yield competitive performances in terms of support recovery and coefficient accuracy. However, NNOG algorithms yield a lower residual value for a given cardinality K. Besides, NLARS tends to be slower when dealing with highly correlated dictionaries.

3.9 Conclusion

According to our numerical comparisons, we can conclude that the computing time of NNOMP then becomes comparable to that of OMP, so that the additional cost should not prevent users from skipping from OMP to NNOMP in any nonnegative sparse problem. The performances of

NNOMP and FNNOMP are very close both in terms of computational time and accuracy. Since FNNOMP was introduced as a fast but approximate version of NNOMP, we conclude that it is not necessary to work on approximate schemes to derive efficient implementations. On the other hand, we have obtained a dramatic acceleration of NNOLS compared to both canonical and fast versions proposed in [Yaghoobi and Davies, 2015]. The computing cost of NNOLS remains larger than that of OLS. However, whenever OLS can be used, our exact implementation of NNOLS is a realistic option to handle nonnegativity constraints, given that the potential of performance gain between NNOMP and NNOLS is comparable to the one between OMP and OLS in the unconstrained case. We have also studied SNNOLS, which is a suboptimal version of NNOLS originally introduced in [Yaghoobi and Davies, 2015]. Our conclusion is that SNNOLS represents a good trade-off between NNOMP and NNOLS: it is structurally simpler than NNOLS and hence significantly faster, with very similar performance in terms of estimation error. However, NNOLS is more robust in dealing with high conditioning.

Our comparisons with NNSP and NNCoSaMP show that NNOG algorithms are very competitive in terms of accuracy, although no algorithm outperforms the others in all scenarios. The proposed recursive implementations yield a substantial reduction of computation compared to NNSP and NNCoSaMP. An attractive feature of NNOG algorithms is their versatility. Indeed, the implementations presented in this chapter can be used to address both forms of (1) problems with constraints $\|\boldsymbol{x}\|_0 \leq K$ (as addressed in this chapter) or $\|\boldsymbol{y} - H\boldsymbol{x}\|_2^2 \leq \varepsilon^2$, where ε^2 is related to the noise variance. In the latter case, one simply needs to replace the stopping condition $|S| = K$ in Algorithm 12 by $\|\boldsymbol{r}_S\|_2^2 \leq \varepsilon^2$. NNSP, NNCoSaMP, and NNHTP are apparently less versatile, although the proposal of novel versions dedicated to the constraint $\|\boldsymbol{y} - H\boldsymbol{x}\|_2^2 \leq \varepsilon^2$ could be addressed as a perspective to the work in [Kim and Haldar, 2016].

Finally, our comparison shows that NNOG algorithms and NLARS are very competitive in performance. However, NNOG algorithms have a better ability to find a low value of the residual for a given cardinality K.

Chapter 4

Exact support recovery with NNOG algorithms[1]

Contents

In Chapters 2-3, we addressed the algorithmic issues of non-negative greedy algorithms. In particular, we introduced the class of NNOG algorithms, exhibited their structural properties and proposed a fast implementation of NNOG algorithms which was assessed by several numerical comparisons. In this chapter we address the theoretical guarantee of NNOG algorithms.

4.1 Introduction

Orthogonal Matching Pursuit (OMP) [Pati et al., 1993] and Orthogonal Least Squares (OLS) [Chen et al., 1989] are well known greedy, iterative algorithms for sparse decomposition. Their common principle is to sequentially select atoms from a given dictionary, that produce a maximal decrease of the residual error in the least square sense. Finding the K atoms that jointly minimize the residual error at a given sparsity level K is an NP-hard subset selection problem [Natarajan, 1995]. Greedy algorithms such as OMP and OLS form a family of approximate schemes, with a relatively low computing cost compared to exact methods [Bourguignon et al., 2016], while convex relaxation yields another important branch of approximate methods [Tibshirani, 1996, Donoho and Tsaig, 2008].

[1] This chapter is an adaptation of the paper [Nguyen et al., 2019d]

Orthogonal greedy algorithms are able to achieve exact support recovery under certain conditions. The Mutual Incoherence Property (MIP) condition states that OMP is guaranteed to recover the K-sparse representation of a data signal in the noiseless case if the mutual coherence of the dictionary is less than $\frac{1}{2K-1}$ [Tropp, 2004, Herzet et al., 2013]. In the last years, much attention was paid to variations over such a condition. In particular, extensions to noisy cases were considered, see, *e.g.,* [Ben-Haim et al., 2010, Cai and Wang, 2011], and relaxations given partial information on the decomposition of the data signal were proposed [Herzet et al., 2013, Herzet et al., 2016]. Other studies about the theoretical assessment of OMP also consider the Restricted Isometry Property (RIP) condition [Davenport and Wakin, 2010, Satpathi et al., 2013, Li et al., 2015, Wen et al., 2017b]. Exact support recovery properties of OLS have been studied more recently. The MIP condition was extended to OLS in the noiseless case [Herzet et al., 2013] as well as noisy case [Herzet et al., 2016], whereas the exact recovery analyses in [Wen et al., 2017a, Wang and Li, 2017] are based on RIP assumptions. In [Soussen et al., 2013, Herzet et al., 2013, Herzet et al., 2016], OMP and OLS were treated as two instances of a unique orthogonal greedy scheme, and the generic acronym Oxx was used to refer to both. We will adopt the same convention in this chapter.

In the traditional K-step analysis, exact support recovery holds when each iteration of Oxx necessarily selects an atom in the true support S^*, so that S^* is recovered after $K = |S^*|$ iterations. Our main contribution in this chapter is to unveil an exact *sign* recovery property regarding the weights of the selected atoms in the best current approximation under the MIP condition and when the noise is bounded in ℓ_2 norm: at any iteration k between 1 and K, the k coefficients found by Oxx have the same sign as the k corresponding ones in the true decomposition of the data signal.

As an immediate consequence of this sign preservation property, we can theoretically assess exact support recovery properties for the non-negative version of OMP and other non-negative greedy schemes. Non-negative OMP (NNOMP) was first introduced by Bruckstein *et al.* [Bruckstein et al., 2008]. It relies on the repeated maximization of the positive inner product between the residual error and the dictionary atoms, followed by the resolution of a non negative least square problem. For such a sign constrained version of OMP, existing exact support recovery analyses are rare, and somewhat discordant. On the one hand, Bruckstein *et al.* conjectured that a MIP type property holds for NNOMP [Bruckstein et al., 2008], and that the proof is similar to the one given in [Tropp, 2004, Donoho et al., 2006] for OMP. Specifically, [Bruckstein et al., 2008, Theorem 3] states that $\mu < \frac{1}{2K-1}$ is a sufficient condition for exact recovery of any K-sparse representation using NNOMP. On the other hand, Kim *et al.* elaborate a unified MIP analysis of NNOMP and its generalized version in the multiple measurement vector setting [Kim and Haldar, 2016, Theorem 1]. In the specific case of NNOMP, that is, for single measurement vectors, the authors acknowledged that their MIP condition turns out to be very restrictive. Indeed, μ is required to be lower than $\frac{1}{K-1} - \frac{1}{2}$, which can occur only when $K = 1$ or 2. In addition, both analyses in [Bruckstein et al., 2008] and [Kim and Haldar, 2016] are made for the noiseless case. To the best of our knowledge, the exact support recovery of non-negative OLS has not been addressed yet and neither does the extension of related analyses to noisy cases.

We have been unable to prove [Bruckstein et al., 2008, Theorem 3] as prescribed by the authors (as a direct extension of the derivations in [Tropp, 2004, Donoho et al., 2006]). The major obstacle is that the NNOMP selection rule performs comparisons between *signed* inner products, whereas a small mutual coherence condition yields a bound on the unsigned magnitude of inner products (see Section 4.2.3 for further details). Fortunately, the sign preservation property of OMP implies that OMP and NNOMP coincide in the exact support recovery regime, so [Bruckstein et al., 2008, Theorem 3] becomes a trivial byproduct of our sign preservation analysis. Under the same conditions, OLS and NNOLS [Yaghoobi and Davies, 2015] coincide, which allows us to extend [Bruckstein et al., 2008, Theorem 3] to NNOLS. Similarly, the same applies

to SNNOLS [Yaghoobi and Davies, 2015]. Furthermore, our analysis does not restrict to the noiseless case but can apply to noisy cases in which the noise is ℓ_2-bounded.

The chapter is organized as follows. Section 4.2 introduces useful notations and briefly recalls known results about orthogonal greedy algorithms and their non-negative versions. Section 4.3 contains our main results on sign preservation of Oxx, as well as their consequences on sign-constrained greedy algorithms. The key result, stated as Theorem 4.1, is proved in the same section, most technical steps being postponed in Appendix. Section 4.4 contains a limited set of simulations to illustrate the average behavior of the studied algorithms outside the exact support recovery regime. Finally, conclusion can be found in Section 4.5.

4.2 Some notations and background

4.2.1 Notations

In this chapter, for convenience, we denote by $\boldsymbol{x}_S$ the subvector indexed by S (and x_i being the i-th entry of $\boldsymbol{x}$). Besides, we denote by $P_S^\perp = I_m - H_S H_S^\dagger$ the orthogonal projection onto $\mathrm{span}(H_S)^\perp$, where I_m stands for the identity matrix of size $m \times m$. We place the convention that $P_S^\perp = I_m$ when $S = \emptyset$. When H_S is full column rank, one has a further explicit formulation

$$H_S^\dagger = (H_S^\mathrm{t} H_S)^{-1} H_S^\mathrm{t}. \tag{4.1}$$

Let us recall that

$$\tilde{\boldsymbol{h}}_i^S = P_S^\perp \boldsymbol{h}_i \tag{4.2}$$

and

$$\tilde{\boldsymbol{g}}_i^S = \tilde{\boldsymbol{h}}_i^S / \|\tilde{\boldsymbol{h}}_i^S\|, \tag{4.3}$$

Later, we also use the generic notation

$$\tilde{\boldsymbol{c}}_i^S = \begin{cases} \tilde{\boldsymbol{h}}_i^S & \text{(OMP case)}, \\ \tilde{\boldsymbol{g}}_i^S & \text{(OLS case)} \end{cases} \tag{4.4}$$

to refer to either $\tilde{\boldsymbol{h}}_i^S$ or $\tilde{\boldsymbol{g}}_i^S$ depending on the context. We will denote by $\tilde{H}^S$ (resp., $\tilde{G}^S$ and $\tilde{C}^S$) the matrix of size $m \times n$ formed by gathering all projected atoms $\tilde{\boldsymbol{h}}_i^S$ (resp., $\tilde{\boldsymbol{g}}_i^S$ and $\tilde{\boldsymbol{c}}_i^S$).

For convenience, the residual vector and the support found by orthogonal greedy algorithms at iteration k will be denoted $\boldsymbol{r}_k$ and S_k, respectively, with $|S_k| = k$. By extension, we will denote $\boldsymbol{r}_0 = \boldsymbol{y}$ and $S_0 = \emptyset$. Whenever unambiguous, we will use the simpler notations $\boldsymbol{r}$ and S.

In our analysis, we make use of the mutual coherence of the dictionary, defined by

$$\mu(H) = \max_{i \neq j} |\boldsymbol{h}_i^\mathrm{t} \boldsymbol{h}_j|. \tag{4.5}$$

This quantity tells us how much the dictionary atoms look alike.

4.2.2 Exact recovery analysis of OMP and OLS

As seen in Chapter 1, OMP and OLS address the following minimization problem:

$$\min_{\|\boldsymbol{x}\|_0 \leq K} \|\boldsymbol{y} - H\boldsymbol{x}\|^2. \tag{4.6}$$

Both algorithms start with a zero vector solution corresponding to the empty support. Then, at each iteration, a new atom ℓ is selected and added to the support. This process is repeated

Algorithm 16: Oxx in K steps.

Input: $\boldsymbol{y}$, H
Output: $\boldsymbol{x}$ with $\|\boldsymbol{x}\|_0 = K$

Initialization: $\boldsymbol{x} \leftarrow \boldsymbol{0}$; $S \leftarrow \emptyset$;
repeat
  $\boldsymbol{r} \leftarrow \boldsymbol{y} - H_S \boldsymbol{x}_S$;
  Choose an atom ℓ according to either (4.7) or (4.8);
  $S \leftarrow S \cup \{\ell\}$;
  $\boldsymbol{x}_S \leftarrow H_S^\dagger \boldsymbol{y}$;
until $|S| = K$;

until the support of cardinality K is reached (see Algorithm 16). OMP and OLS share the same coefficient update and only differ by their selection rule:

$$\text{OMP:} \quad \ell \in \arg\max_{i \notin S} |\boldsymbol{r}^{\mathrm{t}} \boldsymbol{h}_i|, \tag{4.7}$$

$$\text{OLS:} \quad \ell \in \arg\max_{i \notin S} |\boldsymbol{r}^{\mathrm{t}} \tilde{\boldsymbol{g}}_i^S| \tag{4.8}$$

where $\boldsymbol{r} = P_S^\perp \boldsymbol{y}$ denotes the current residual vector. Note that (4.7) can also be written as

$$\ell \in \arg\max_{i \notin S} |\boldsymbol{r}^{\mathrm{t}} \tilde{\boldsymbol{h}}_i^S| \tag{4.9}$$

since $\boldsymbol{r}$ is orthogonal to $\mathrm{span}(H_S)$. These inner product expressions come from the geometrical interpretation of OMP and OLS [Blumensath and Davies, 2007]. Besides, from an optimization viewpoint, the selection rule of OMP is based on the minimization of $\|\boldsymbol{r} - z_i \boldsymbol{h}_i\|^2$ w.r.t. scalar z_i, whereas OLS relies on the minimization of $\|\boldsymbol{y} - H_{S\cup\{i\}} \boldsymbol{z}\|^2$ w.r.t. vector $\boldsymbol{z}$ of length $|S| + 1$ [Blumensath and Davies, 2007]. From this point of view, it is obvious that the OLS selection rule is more costly than that of OMP, although recursive update schemes are available, see, *e.g.,* [Chen et al., 1989, Rebollo-Neira and Lowe, 2002].

Let us state the MIP condition that holds for both OMP and OLS in the noiseless case. Note that under the assumption of Lemma 4.1, the K-sparse representation is unique [Donoho and Elad, 2003, Theorem 7].

Lemma 4.1. *[Tropp, 2004, Theorem 3.1-3.5], [Herzet et al., 2013, Theorem 2]. Assume that* $\mu(H) < \frac{1}{2K-1}$. *Let* $\boldsymbol{y} = H\boldsymbol{x}^*$ *be a* K*-sparse representation. Then Oxx recover the support of* $\boldsymbol{x}^*$ *in* K *iterations.*

The MIP condition in Lemma 4.1 was extended to noisy cases in which the noise is bounded in ℓ_2 norm in [Cai and Wang, 2011, Herzet et al., 2016]. We combine these results in a unique statement in Lemma 4.2. One can find back the result in [Cai and Wang, 2011, Theorem 1] when replacing $\|\boldsymbol{\epsilon}\|_2$ in (4.11) by any upper bound $b \geq \|\boldsymbol{\epsilon}\|_2$. When the non-zero coefficients of $\boldsymbol{x}^*$ are sorted in a descending order, the result in [Herzet et al., 2016, Theorem 4] is obtained when replacing (4.11) by

$$\|\boldsymbol{\epsilon}\|_2 < \frac{1 - (2K - i)\mu(H)}{2} |x_i^*|, \ \forall 1 \leq i \leq K. \tag{4.10}$$

Lemma 4.2. *[Cai and Wang, 2011, Theorem 1], [Herzet et al., 2016, Theorem 4]. Assume that* $\mu(H) < \frac{1}{2K-1}$. *Let* $\boldsymbol{y} = H\boldsymbol{x}^* + \boldsymbol{\epsilon}$ *where* $\boldsymbol{x}^*$ *is* K*-sparse and*

$$\|\boldsymbol{\epsilon}\|_2 < \frac{1 - (2K - 1)\mu(H)}{2} \min\{|x_i^*|, x_i^* \neq 0\}. \tag{4.11}$$

Then Oxx recover the support of $\boldsymbol{x}^*$ *in* K *iterations.*

Obviously, Lemma 4.1 is a special case of Lemma 4.2 when $\boldsymbol{\epsilon} = \mathbf{0}$. Also, let us emphasize that the mutual coherence in the previous simulations in Chapter 3 does not satisfy the MIP condition hence the Lemmas 4.1- 4.2 do not apply. For instance, in the simulation of §3.4.1, $\mu(H) = 0.94$ hence the MIP condition $\mu(H) < \frac{1}{2K-1}$ only holds for $K = 1$ and the inequality (4.11) only holds for the noise level $P_{\boldsymbol{\epsilon}} < 10^{-7} \min\{|x_i^*|^2, x_i^* \neq 0\}$.

Here and throughout this chapter, we consider that in the special case where the Oxx selection rules (4.7)-(4.8) yield several solutions (*i.e.*, two atoms i_1 and i_2 for which $|\boldsymbol{r}^{\mathrm{t}}\tilde{\boldsymbol{c}}_{i_1}^S| = |\boldsymbol{r}^{\mathrm{t}}\tilde{\boldsymbol{c}}_{i_2}^S|$) including an atom that does not belong to the support of $\boldsymbol{x}^*$, Oxx makes the wrong decision and hence K-step exact recovery does not occur.

4.2.3 Exact recovery analysis in the non-negative setting

4.2.3.1 Extension of K-step exact recovery analysis of OMP

Bruckstein *et al.* [Bruckstein et al., 2008] claimed that the K-step exact recovery analysis of NNOMP can be carried out as a straightforward extension of the classical K-step exact recovery analyses of OMP [Tropp, 2004, Donoho et al., 2006]. Here, we argue that this extension does not actually appear to be trivial.

Tropp's reasoning in [Tropp, 2004] consists in lower bounding the absolute value of the inner product $|\boldsymbol{r}^{\mathrm{t}}\boldsymbol{h}_i|$ between the residual and the correct dictionary atoms based on induced matrix norm identities. Unfortunately, similar lower bounds cannot be directly extended when absolute values are dropped. Donoho *et al.*'s recursive proof [Donoho et al., 2006] exploits that for noiseless sparse inputs, the data residual lays in the subspace spanned by the true atoms. So the result at the first iteration (guaranteed selection of a true atom) can be exploited again in the next iterations by replacing the original input signal by the residual. To generalize this reasoning to the non-negative case, one would need to prove that for non-negative sparse decompositions $\boldsymbol{y} = H\boldsymbol{x}^*$ (*i.e.*, inputs laying in the positive span of the true atoms), the residual lays in the positive span of the dictionary atoms as well. This conjecture turns out to be false; if true atoms are selected until iteration k, the residual $\boldsymbol{r} = H(\boldsymbol{x}^* - \hat{\boldsymbol{x}}^{(k)})$ lays in the subspace spanned by the true atoms, where $\hat{\boldsymbol{x}}^{(k)}$ denotes the NNOMP iterate. Provided that the latter atoms are linearly independent, it is clear that $\boldsymbol{r}$ lays in their positive span if and only if $\hat{\boldsymbol{x}}^{(k)} \leq \boldsymbol{x}^*$. Simple numerical tests show that even for toy problems, $\hat{\boldsymbol{x}}^{(k)} \leq \boldsymbol{x}^*$ may not hold when $\boldsymbol{x}^* \geq \mathbf{0}$ and $\mu(H) < \frac{1}{2K-1}$: see Fig.. 4.1 for a simple illustration, also § 4.4.2 and Fig. 4.3(a). Therefore, sign-preservation does not apply to the residual vector, hence Donoho *et al.*'s reasoning cannot be directly extended to the non-negative case.

4.2.3.2 ℓ_1 analysis with non-negativity constraints

The theoretical analysis of non-negative versions of Basis Pursuit

$$\mathrm{BP} : \min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 \text{ s.t. } \boldsymbol{y} = H\boldsymbol{x} \tag{4.12}$$

and Basis Pursuit Denoising

$$\mathrm{BPDN} : \min_{\boldsymbol{x}} \|\boldsymbol{y} - H\boldsymbol{x}\|^2 + \lambda\|\boldsymbol{x}\|_1 \tag{4.13}$$

turns out to be far simpler than the analysis of orthogonal greedy algorithms. Indeed, it is well-known that contrary to greedy algorithms, the exact recovery analysis of Basis Pursuit heavily depends on the sign pattern. Fuchs [Fuchs, 2004] proved that when $\boldsymbol{y} = H\boldsymbol{x}^*$, BP and BPDN (for sufficiently small λ) both have a unique solution under the MIP assumption, and that this solution identifies with $\boldsymbol{x}^*$ as long as H_{S^*} is full rank and

$$\forall j \notin S^*, \ |\langle \boldsymbol{\sigma}^*, H_{S^*}^{\dagger}\boldsymbol{h}_j \rangle| < 1 \tag{4.14}$$

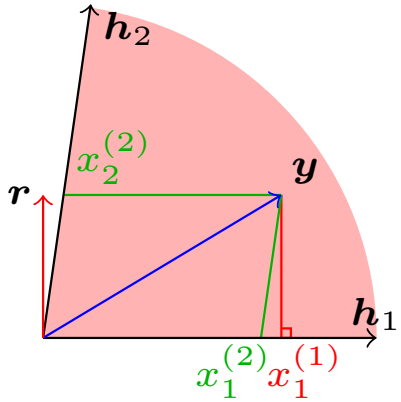


Figure 4.1: The positive span assumption does not propagate to the residual.

where $S^* := \mathrm{supp}(\boldsymbol{x}^*)$ and $\boldsymbol{\sigma}^* := \mathrm{sign}(\boldsymbol{x}^*_{S^*}) \in \{-1, 1\}^K$ denotes the sign pattern. It is noticeable that the latter condition[2] only depends on the sign pattern, but not on the magnitudes of coefficients x_i^*. Denoting by $\mathbf{1}$ the "all-ones" vector of size K, condition (4.14) with $\boldsymbol{\sigma}^* \leftarrow \mathbf{1}$ thus guarantees exact recovery of $\boldsymbol{x}^*$ for any $\boldsymbol{x}^* \geq \mathbf{0}$ supported by S^*. It follows that $\boldsymbol{x} = \boldsymbol{x}^*$ is not only the unique solution of BP/BPDN but also of their non-negative counterpart

$$\min_{\boldsymbol{x} \geq \mathbf{0}} \|\boldsymbol{y} - H\boldsymbol{x}\|^2 + \lambda \|\boldsymbol{x}\|_1, \tag{4.15}$$

often referred to as the non-negative Garrote in statistics [Miller, 2002]. Indeed, the minimum value of $\|\boldsymbol{y} - H\boldsymbol{x}\|^2 + \lambda\|\boldsymbol{x}\|_1$ over $\boldsymbol{x} \geq \mathbf{0}$ can be reached only for $\boldsymbol{x} = \boldsymbol{x}^*$. Summarizing, the sign preservation property of BP is guaranteed as long as Fuchs condition (4.14) is met.

4.2.3.3 Extension of K-step exact recovery analysis of ℓ_1 homotopy

Homotopy is a popular greedy algorithm dedicated to ℓ_1 minimization. It was originally proposed in [Efron et al., 2004] in the undercomplete setting, and named modified Least Angle Regression (LARS). It was later renamed homotopy by Donoho and Tsaig, and further analyzed in the overcomplete setting under the MIP assumption in the same paper [Donoho and Tsaig, 2008]. Homotopy aims to solve the BPDN problem (4.13) for a continuum of λ's. The principle of homotopy is to reconstruct the regularization path (defined as the set of solutions for all λ) by solving the ℓ_1 problem for gradually decreasing λ's starting from $\lambda = +\infty$, with the corresponding solution $\boldsymbol{x} = \mathbf{0}$. Homotopy has a stepwise mechanism with an atom selection or deselection at each iteration. It turns out [Donoho and Tsaig, 2008, Theorem 1] that when $\mu(H) < \frac{1}{2K-1}$, K-step recovery of any K-sparse vector $\boldsymbol{x}^*$ from noise-free observations $\boldsymbol{y} = H\boldsymbol{x}^*$ is guaranteed. Not only the support but also the magnitudes of $\boldsymbol{x}^*$ are found after K iterations (no deselection occurs).

Although homotopy has a greedy structure, a major difference with Oxx algorithms is that homotopy solves an ℓ_1-penalized least square problem at each iteration. Therefore, the related exact recovery analysis significantly differs from that of Oxx. The exact recovery analysis in [Donoho and Tsaig, 2008, Theorem 1] is based on two ingredients: the correct selection of atoms indexed by $\mathrm{supp}(\boldsymbol{x}^*)$ and the so-called *sign agreement* property. The latter ensures that elements from

[2] referred to as Fuchs corollary condition in [Plumbley, 2007] as opposed to the sharp (but more involved) Fuchs condition in [Plumbley, 2007, Theorem 2]. The latter condition solely depends on the sign pattern as well.

the solution support are never removed, so the deselection event never occurs. Sign agreement is defined as follows: *at every iteration k, the homotopy iterate $\boldsymbol{x}$ satisfies* $\operatorname{sign}(\boldsymbol{x}_S) = \operatorname{sign}(H_S^{\mathrm{t}}\boldsymbol{r})$, *where* $S = \operatorname{supp}(\boldsymbol{x})$. Donoho and Tsaig showed that when $\mu(H) < \frac{1}{2K-1}$, the magnitudes of the active atoms always increase while λ is decreasing. Since the correct magnitudes are exactly found at iteration K ($\boldsymbol{x} = \boldsymbol{x}^*$), it follows that their sign is unchanged throughout the iterations. So, a sign preservation result in the ℓ_1 case is truly obtained as a byproduct of [Donoho and Tsaig, 2008, Theorem 1]. In the Oxx setting, the sign preservation result will be obtained here using far less straightforward reasoning. Moreover, the sign agreement property "$\operatorname{sign}(\boldsymbol{x}_S) = \operatorname{sign}(H_S^{\mathrm{t}}\boldsymbol{r})$" does not make sense anymore since $H_S^{\mathrm{t}}\boldsymbol{r} = \boldsymbol{0}$, due to the orthogonality between the residual vector and the selected atoms.

4.3 Sign preservation and exact recovery

This section contains our main results concerning sign preservation of Oxx when the exact recovery condition in terms of mutual coherence is met. The cornerstone of our study is Theorem 4.1, while the other results of Subsection 4.3.1 are rather direct consequences. In Subsection 4.3.2, we have decomposed the proof of Theorem 4.1 into distinct steps, most technical elements being postponed in Appendix.

4.3.1 Main results

While the dictionary H satisfies $\mu(H) < \frac{1}{2K-1}$, let us define the constant

$$M_{\boldsymbol{x}^*} := \frac{1 - (2K-1)\mu(H)}{2} \min\{|x_i^*|, x_i^* \neq 0\} \tag{4.16}$$

for any K-sparse vector $\boldsymbol{x}^*$. The sign preservation property for arbitrary noisy K-sparse representations is stated now.

Theorem 4.1. *Assume that $\mu(H) < \frac{1}{2K-1}$. Let $\boldsymbol{y} = H\boldsymbol{x}^* + \boldsymbol{\epsilon}$ be a noisy K-sparse representation with $\|\boldsymbol{\epsilon}\|_2 < M_{\boldsymbol{x}^*}$. Then Oxx recovers the support of $\boldsymbol{x}^*$ in K iterations, and at each iteration, the weights of selected atoms are of the same sign as the corresponding magnitudes of $\boldsymbol{x}^*$.*

An obvious corollary can be stated in the special case of noisy K-sparse representations with non-negative weights.

Corollary 4.1. *Assume that $\mu(H) < \frac{1}{2K-1}$. Let $\boldsymbol{y} = H\boldsymbol{x}^* + \boldsymbol{\epsilon}$ be a noisy K-sparse representation with $\boldsymbol{x}^* \geq \boldsymbol{0}$ and $\|\boldsymbol{\epsilon}\|_2 < M_{\boldsymbol{x}^*}$. Then, Oxx recovers the support of $\boldsymbol{x}^*$ in K iterations, and at each iteration, the weights of selected atoms are positive.*

In the non-negative setting, Corollary 4.1 has interesting implications concerning non-negative versions of Oxx. Let us start with the following lemma.

Lemma 4.3. *Let $\boldsymbol{y} \in \mathbb{R}^m$ and assume that any combination of K dictionary columns is linearly independent.*

- *Assume that at every iteration $k = 1, \ldots, K$ of OMP, the selection rule yields a unique optimal index ℓ and that the weights of selected atoms are all positive. Then NNOMP provides the same iterates as OMP (i.e., with the same support and the same weights at every iteration).*
- *The same applies if OMP and NNOMP are replaced by OLS and SNNOLS, respectively.*
- *It also applies if OMP and NNOMP are replaced by OLS and NNOLS, respectively.*

This result is intuitive since at each iteration of Oxx, the weights of selected atoms form a vector defined as a least squares solution. Clearly, if an unconstrained least squares solution is positive, then it is also the solution of the corresponding NNLS problem.

Proof. See Appendix 4.B.2. ■

From Corollary 4.1 and Lemma 4.3, we can deduce the following result.

Corollary 4.2. *Assume that* $\mu(H) < \frac{1}{2K-1}$. *Let* $\boldsymbol{y} = H\boldsymbol{x}^* + \boldsymbol{\epsilon}$ *be a noisy* K*-sparse representation with* $\boldsymbol{x}^* \geq \boldsymbol{0}$ *and* $\|\boldsymbol{\epsilon}\|_2 < M_{\boldsymbol{x}^*}$. *Then NNOMP identifies with OMP whereas both NNOLS and SNNOLS identify with OLS. Thus, NNOMP, NNOLS and SNNOLS all recover the support of* $\boldsymbol{x}^*$ *in* K *iterations.*

By Proposition 2.3 in Chapter 2 one can deduce that Sparse NNLS [Peharz and Pernkopf, 2012] identifies with NNOMP and OMP under the assumptions of Corollary 4.2. Therefore, we have the following result regarding the exact recovery of Sparse NNLS.

Corollary 4.3. *Assume that* $\mu(H) < \frac{1}{2K-1}$. *Let* $\boldsymbol{y} = H\boldsymbol{x}^* + \boldsymbol{\epsilon}$ *be a noisy* K*-sparse representation with* $\boldsymbol{x}^* \geq \boldsymbol{0}$ *and* $\|\boldsymbol{\epsilon}\|_2 < M_{\boldsymbol{x}^*}$. *Then Sparse NNLS identifies with OMP and recovers the support of* $\boldsymbol{x}^*$ *in* K *iterations.*

It is known that the MIP condition $\mu(H) < \frac{1}{2K-1}$ is not only sufficient but also necessary for *uniform* (*i.e.*, irrespective of the magnitudes of nonzero coefficients in the sparse representation and of the choice of the dictionary) K-step exact support recovery [Cai et al., 2010, Herzet et al., 2013] by Oxx. Cai *et al.* [Cai et al., 2010] indeed exhibited an equiangular dictionary whose mutual coherence equals $\mu(H) = \frac{1}{2K-1}$ and a vector $\boldsymbol{y}$ having two K-sparse representations $H\boldsymbol{x} = H\boldsymbol{z}$ with distinct supports. K-step exact support recovery does not make sense anymore in this situation, since either the support of $\boldsymbol{x}$ or $\boldsymbol{z}$ cannot be reconstructed in K steps. The same analysis can be made concerning non-negative extensions of Oxx.

Corollary 4.4. *The MIP condition* $\mu(H) < \frac{1}{2K-1}$ *is necessary for* K*-step exact recovery of any non-negative* K*-sparse vector by non-negative orthogonal greedy algorithms, since there exists a dictionary* H *with* $\mu(H) = \frac{1}{2K-1}$ *and a vector* $\boldsymbol{y}$ *having two non-negative* K*-sparse representations with distinct supports.*

Proof. Consider the dictionary $H \in \mathbb{R}^{m\times n}$ with $\mu(H) = \frac{1}{2K-1}$ and the vector $\boldsymbol{y} \in \mathbb{R}^m$ proposed in [Cai et al., 2010, Section III], the latter vector having two K-sparse representations $H\boldsymbol{x} = H\boldsymbol{z}$ with distinct supports. Since both supports are distinct, one can define the subrogate dictionary $H' \in \mathbb{R}^{m\times n}$ as:

$$\boldsymbol{h}'_i = \begin{cases} \operatorname{sign}(x_i)\,\boldsymbol{h}_i & \text{if } i \in \operatorname{supp}(\boldsymbol{x}), \\ \operatorname{sign}(z_i)\,\boldsymbol{h}_i & \text{if } i \in \operatorname{supp}(\boldsymbol{z}), \\ \boldsymbol{h}_i & \text{otherwise.} \end{cases} \tag{4.17}$$

Moreover, let $|\boldsymbol{x}|$ and $|\boldsymbol{z}| \in \mathbb{R}^n$ denote the non-negative vectors whose entries are equal to $|x_i|$ and $|z_i|$, respectively. Obviously, $H'|\boldsymbol{x}| = H'|\boldsymbol{z}|$, and $\mu(H') = \mu(H) = \frac{1}{2K-1}$. ■

4.3.2 Proof of Theorem 4.1

Let us remark that it is sufficient to consider the case of non-negative weights. Indeed, an arbitrary K-sparse representation can be reduced to the case with non-negative weights, by an obvious transformation where each negative weight is replaced by its opposite value, the corresponding atom being also replaced by the opposite one. Moreover, it is straightforward to check that the list of atoms selected by Oxx is invariant through such a transformation. We

therefore simply need to prove the result for non-negative weights, which corresponds to the setting of Corollary 4.1.

Before going further, let us denote by

$$\mathcal{C}^+_{K,\epsilon} := \{H\boldsymbol{x}^* + \boldsymbol{\epsilon},\ \boldsymbol{x}^* \geq \mathbf{0},\ \|\boldsymbol{x}^*\|_0 = K \text{ and} \|\boldsymbol{\epsilon}\|_2 < M_{\boldsymbol{x}^*}\} \tag{4.18}$$

the set of noisy K-sparse representations with non-negative weights and ℓ_2-bounded noise. Now, consider $\boldsymbol{y} = H\boldsymbol{x}^* + \boldsymbol{\epsilon} \in \mathcal{C}^+_{K,\epsilon}$. From Lemma 4.2, Oxx recovers the support S^* of $\boldsymbol{x}^*$ in K iterations. At any iteration $k \leq K$, the support S_k of the current solution $\hat{\boldsymbol{x}}^{(k)}$ is therefore a subset of S^*. Recall that $\hat{\boldsymbol{x}}^{(k)}_{S_k}$ is the unconstrained least squares solution related to support S_k, see Algorithm 16, line 6. Let $\boldsymbol{r}_k = P^{\perp}_{S_k}\boldsymbol{y}$ denote the related residual vector, with $\boldsymbol{r}_0 = \boldsymbol{y}$ corresponding to $S_0 = \emptyset$.

We proceed in two distinct steps to prove that $\hat{\boldsymbol{x}}^{(k)}_{S_k} > \mathbf{0}$ for all $k \in \{1, \ldots, K\}$. First, we prove that the weight of each newly selected atom $\hat{x}^{(k)}_{S_k \setminus S_{k-1}}$ is positive for any $k \leq K$. Then, we show that the updated coefficients $\hat{\boldsymbol{x}}^{(k)}_{S_{k-1}}$ remain positive. Let us remark that Theorem 4.1 states a trivial fact at iteration K, since $\hat{\boldsymbol{x}}^{(K)} = \boldsymbol{x}^*$ according to Lemma 4.1.

Let us first characterize the last $k - j$ coefficients of $\hat{\boldsymbol{x}}^{(k)}_{S_k}$, $j < k$ being an arbitrary index in the following lemma. Note that this lemma applies to any real vector $\boldsymbol{y}$.

Lemma 4.4. *Let $\boldsymbol{y} \in \mathbb{R}^m$, and let j and k be two iteration indices such that $0 \leq j < k$. Assume that H_{S_k} is full column rank. Then, the k-th iterate of Oxx satisfies*

$$\hat{\boldsymbol{x}}^{(k)}_{S_k \setminus S_j} = \left[\tilde{H}^{S_j}_{S_k \setminus S_j}\right]^{\dagger} \boldsymbol{r}_j. \tag{4.19}$$

Proof. See Appendix 4.B.1. ■

Then, the two main steps of the proof of Theorem 4.1 correspond to the following lemmas.

Lemma 4.5 (non-negativity of new coefficient)**.** *Assume that $\mu(H) < \frac{1}{2K-1}$ and let $\boldsymbol{y} \in \mathcal{C}^+_{K,\epsilon}$. For all $k \in \{1, \ldots, K\}$, $\hat{x}^{(k)}_{S_k \setminus S_{k-1}} > 0$.*

Proof. See Appendix 4.B.3. ■

Lemma 4.6 (non-negativity of updated coefficients)**.** *Assume that $\mu(H) < \frac{1}{2K-1}$ and let $\boldsymbol{y} \in \mathcal{C}^+_{K,\epsilon}$. For all $k \in \{2, \ldots, K\}$, $\hat{\boldsymbol{x}}^{(k)}_{S_{k-1}} > \mathbf{0}$.*

Proof. See Appendix 4.B.4. ■

4.4 Numerical study

4.4.1 Comparison of Oxx and their sign-aware versions

The previous section shows that in some specifically favorable situations, greedy algorithms such as OMP have not only the capacity to recover the support of the true solution, but also to recover the correct weight signs. In such conditions, according to Lemma 4.3, implementing non-negative (or more generally, sign-aware) versions of such greedy algorithms is useless. However, one cannot generalize such a conclusion to more realistic scenarios. On the contrary, one can empirically observe that sign-aware greedy algorithms tend to reach superior performance, which is in agreement with the fact that they exploit more information than usual greedy algorithms. To illustrate this fact, let us consider a dictionary H with 22 regularly spaced, discretized Gaussian-shaped atoms, with a constant standard deviation $\sigma = 0.5$, and a mutual coherence $\mu(H) = 0.37$. We randomly choose $K = 10$ atoms in H, whose location in the dictionary are distributed

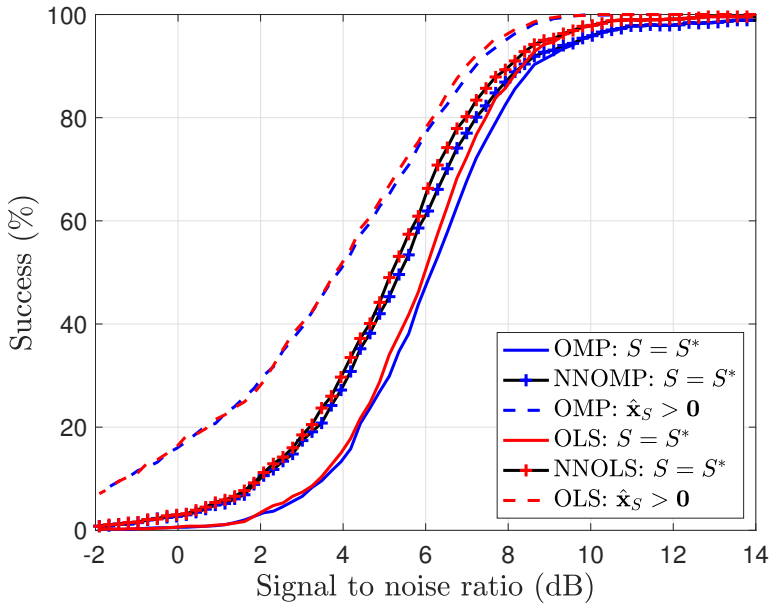
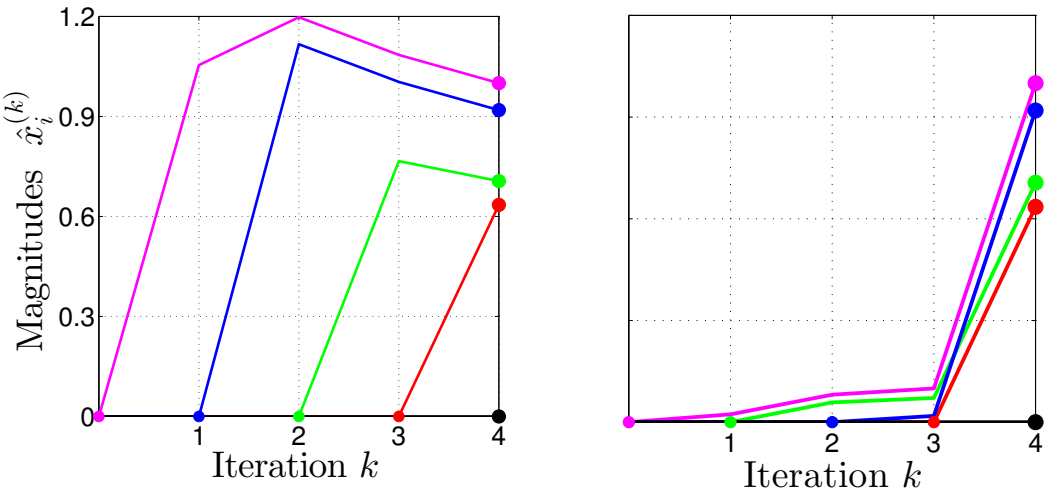


Figure 4.2: Rate of exact support recovery ($S_K = S^*$) and non-negativity at all iterations ($\forall k$, $\hat{\boldsymbol{x}}^{(k)} \geq \mathbf{0}$) w.r.t. the signal-to-noise ratio for a simulated data experiment involving Gaussian-shaped atoms. SNNOLS (not shown here) yields the exact same curve as NNOLS.

according to the uniform distribution. The atoms are all equally weighted with a unit weight, and we generate data $\boldsymbol{y} = H\boldsymbol{x}^* + \boldsymbol{\epsilon}$ with some additive Gaussian noise $\boldsymbol{\epsilon}$. Note that $\mu(H) > \frac{1}{2K-1}$, which implies that exact support recovery is not mathematically guaranteed even at low noise. Within a certain range of signal-to-noise ratio (SNR, defined by $10\log_{10}(\|H\boldsymbol{x}^*\|^2/\|\boldsymbol{\epsilon}\|^2)$), we have generated average performance for OMP, NNOMP, OLS, SNNOLS and NNOLS in terms of rate of exact support recovery, the stopping condition being that the cardinality of the estimated support should be 10. This experiment has been repeated 1000 times to obtain the average results shown in Fig. 4.2. Several empirical conclusions can be drawn. Some of them are already acknowledged facts. For instance, greedy algorithms keep some exact recovery capacities far beyond the "zero defect" area. In contrast, in the low SNR regime, the exact recovery capacity almost surely vanishes. The intermediate zone is the most interesting. Specifically, one can notice that there is a significant difference of performance between the usual greedy algorithms and their non-negative extensions. We have also performed a sign-preservation test for OMP and OLS that simply consists in checking whether at all iterations, all estimated weights are positive. Fig. 4.2 shows that such a sign-preservation property is rather robust, and that empirically, sign-preservation is always reached with Oxx whenever exact support recovery is found.

4.4.2 Non-monotony of the magnitude variations

As argued in § 4.2.3.3, ℓ_1 homotopy is a stepwise greedy algorithm for which the sign preservation guarantee holds whenever $\mu(H) < \frac{1}{2K-1}$. In [Donoho and Tsaig, 2008], Donoho and Tsaig proved a stronger result under the mutual coherence condition: the magnitudes $\hat{x}_i^{(k)}$ of the selected atoms keep increasing while k is increasing. In contrast, such monotony property does not hold for Oxx algorithms, since the magnitudes may either increase or decrease during the iterations. This claim can be proven analytically at iteration 2, by using the fact that $\hat{x}_{S_1}^{(1)} > 0$ and $\hat{\boldsymbol{x}}_{S_2}^{(2)} > \mathbf{0}$ according to Corollary 4.1. Since $\hat{\boldsymbol{x}}^{(1)}$ and $\hat{\boldsymbol{x}}^{(2)}$ have closed-form expressions, a simple calculation (skipped for brevity reasons) shows that the first magnitude is decreasing, *i.e.*, $\hat{x}_{S_1}^{(1)} - \hat{x}_{S_1}^{(2)} > 0$ if and only if the inner product between the atoms selected in the first two



(a) OMP/OLS (b) ℓ_1-homotopy

Figure 4.3: Behavior of Oxx and ℓ_1-homotopy in the case of a toy problem ($m = n = 5$) corresponding to a 4-sparse noiseless representation. (a) The OMP and OLS iterates are identical and yield magnitudes $\hat{x}_i^{(k)}$ with non-monotonous variations throughout the iterations k. (b) On the contrary, the magnitudes of the ℓ_1-homotopy iterates are always increasing. The ground truth magnitudes x_i^* are represented with bullets for $k = 4$, and are exactly recovered after $K = 4$ iterations ($\hat{\boldsymbol{x}}^{(k)} = \boldsymbol{x}^*$).

iterations is positive. So, the magnitude of $\hat{x}_{S_1}$ may either increase or decrease depending on the sign of the inner product. We further compare ℓ_1-homotopy with the Oxx algorithms for a toy problem of dimension $(m, n) = (5, 5)$, with an equiangular dictionary H such that $\mu(H) = \frac{0.9}{2K-1}$. The columns of H satisfy $\langle \boldsymbol{h}_i, \boldsymbol{h}_j \rangle = \pm\mu(H)$, where the sign of the inner product is randomly chosen. The ground truth vector $\boldsymbol{x}^*$ is K-sparse with $K = 4$, with nonzero magnitudes drawn from the uniform distribution $\mathcal{U}([0.6, 1])$. Since $\mu(H) < \frac{1}{2K-1}$, K-step exact recovery holds for all considered algorithms. In Fig. 4.3, the variation of each entry $\hat{x}_i^{(k)}$ with respect to k is represented with a specific color. As expected (since exact support recovery holds), the black magnitude, which corresponds to the wrong atom $i \notin S^*$, is equal to 0 throughout the iterations. ℓ_1-homotopy yields magnitudes that are indeed increasing with k, which is consistent with the theoretical result in [Donoho and Tsaig, 2008]. On the contrary, the OMP and OLS iterates (which are identical here; the same indices are selected at each iteration) are non-monotonous.

4.5 Conclusion

We have established that OMP and OLS satisfy the sign preservation property when dealing with sufficiently incoherent dictionaries: in the exact recovery regime $\mu(H) < \frac{1}{2K-1}$, the Oxx estimates are guaranteed to keep the same sign as the ground-truth sparse vector, at all iterations. This interesting property enables us to establish the first K-step recovery analysis of three non-negative greedy algorithms proposed previously, namely NNOMP, NNOLS and SNNOLS, under the MIP condition $\mu(H) < \frac{1}{2K-1}$. This exact recovery condition turns out to be identical for both Oxx algorithms and their non-negative extensions. Moreover, it is not only a sufficient but also a (worst-case) necessary condition of exact recovery. Therefore, one cannot distinguish the performance of Oxx and their nonnegative counterparts from this theoretical viewpoint. There is still room for improvement though in cases where the dictionary atoms are known to be non-negative valued, since then the MIP condition is not guaranteed to be necessary anymore. This setting covers many application fields ranging from sparse deconvolution [Bendory, 2017]

to tomographic image reconstruction [Petra and Schnörr, 2014].

Empirically, we observed that the sign preservation property holds for noiseless scenarios when $\mu(H)$ is substantially greater than $\frac{1}{2K-1}$. However, for truly coherent dictionaries and for noisy observations, Oxx algorithms do yield iterates with negative entries, thus sign-aware versions are worth being considered. Furthermore, the latter versions yield improvement of empirical exact recovery performance for random dictionaries when $\mu(H) > \frac{1}{2K-1}$ (see Chapter 3).

A perspective of this work is to elaborate on exact recovery conditions when $\mu(H) \geq \frac{1}{2K-1}$. We argued that the MIP condition $\mu(H) < \frac{1}{2K-1}$ is both necessary and sufficient for *uniform* recovery of K-sparse signals, that is, irrespective of the magnitudes of the nonzero coefficients in the K-sparse representation. It is well-known that such worst case analysis is pessimistic since practically, algorithms may succeed far beyond the exact recovery regime $\mu(H) < \frac{1}{2K-1}$. It was shown in [Herzet et al., 2016] that exact recovery guarantees can be obtained for Oxx when $\mu(H) \in [\frac{1}{2K-1}, \frac{1}{K})$ provided that the decay of the magnitudes of the nonzero coefficients is fast enough (the minimum rate of decay required to ensure exact support recovery depends on $\mu(H)$). It would then be interesting to generalize our sign preservation analysis to the case $\mu(H) \in [\frac{1}{2K-1}, \frac{1}{K})$. However, this extension does not appear to be obvious and is left for future work. With the same idea of elaborating on sign preservation under weaker K-step support recovery conditions, Tropp's exact recovery condition (ERC) is worth being considered. Indeed, it is both necessary and sufficient for uniform K-step exact recovery of all representations having a given support S^* with both OMP and OLS [Tropp, 2004,Soussen et al., 2013]. In all our empirical tests, we found that sign preservation holds whenever the ERC is met. However, proving this conjecture would necessitate more involved theoretical analysis than those we derived in this document.

Appendix 4.A Useful lemmas

Let us recall some useful lemmas. Lemma 4.7 provides an upper bound on the ℓ_1-norm of the Gram matrix columns by means of mutual coherence. Lemma 4.8 provides lower and upper bounds on the inner product of projected atoms. Lemma 4.9 is related to the full rankness of the matrix of projected atoms. Lemma 4.10 is a simple algebraic manipulation related to the pseudo-inverse.

Lemma 4.7. *[Foucart and Rauhut, 2013, Theorem 5.3], [Tropp, 2004, Proposition 2.1, Theorem 3.5]. If B is a column-normalized matrix with k columns and $\mu(B) < \frac{1}{k-1}$ then B is full column rank and*

$$\|(B^{\mathrm{t}}B)^{-1}\|_{1,1} \leq \frac{1}{1-(k-1)\mu(B)} \tag{4.20}$$

wherein $\|\cdot\|_{1,1}$ equals the maximum absolute column sum of its argument.

Lemma 4.8. *[Herzet et al., 2016, Lemmas 2-3]. If $\mu(H) \leq \frac{1}{k+1}$ with $k = |S|$ then*

$$\forall i \notin S,\ \|\tilde{\boldsymbol{h}}_i^S\|^2 \geq \beta_k, \tag{4.21}$$

$$\forall p \neq q,\ |(\tilde{\boldsymbol{c}}_p^S)^{\mathrm{t}}\tilde{\boldsymbol{h}}_q^S| \leq \mu_k\eta_k, \tag{4.22}$$

where

$$\beta_k = \frac{(\mu(H)+1)(1-k\mu(H))}{1-(k-1)\mu(H)} \tag{4.23}$$

$$\mu_k = \frac{\mu(H)}{1-k\mu(H)} \tag{4.24}$$

$$\eta_k = \begin{cases} \beta_k & \text{for OMP,} \\ \sqrt{\beta_k} & \text{for OLS.} \end{cases} \tag{4.25}$$

Lemma 4.9. *[Soussen et al., 2013, Lemma 8] If $S \cap S' = \emptyset$ and $H_{S\cup S'}$ is full column rank, then matrices $\tilde{H}^S_{S'}$ and $\tilde{G}^S_{S'}$ are full column rank.*

Lemma 4.10. *Consider a full column rank matrix $H = \left[\boldsymbol{h}_1, H_2\right] \in \mathbb{R}^{p\times q}$ where $\boldsymbol{h}_1 \in \mathbb{R}^p$ and $H_2 \in \mathbb{R}^{p\times(q-1)}$ denotes the submatrix formed of the last $q-1$ column vectors. Then,*

$$\forall \boldsymbol{r} \in \mathbb{R}^p,\ \left(H^\dagger \boldsymbol{r}\right)_1 = \frac{\langle \boldsymbol{r}, P^\perp_{S_2}\boldsymbol{h}_1\rangle}{\|P^\perp_{S_2}\boldsymbol{h}_1\|^2}. \tag{4.26}$$

where the index set $S_2 = \{2,\dots,q\}$ corresponds to the columns of H_2.

Proof. Let $\boldsymbol{r} \in \mathbb{R}^p$. Since H is full column rank, $\boldsymbol{r}$ can be uniquely decomposed as

$$\boldsymbol{r} = \boldsymbol{p}_H + \boldsymbol{p}_{H^\perp} = (H^\dagger \boldsymbol{r})_1 \boldsymbol{h}_1 + \boldsymbol{r}_{H_2} + \boldsymbol{p}_{H^\perp} \tag{4.27}$$

where $\boldsymbol{p}_H$ and $\boldsymbol{p}_{H^\perp}$ are the orthogonal projections of $\boldsymbol{r}$ onto $\operatorname{span}(H)$ and $\operatorname{span}(H)^\perp$, respectively, and $\boldsymbol{r}_{H_2} \in \operatorname{span}(H_2)$. Note though that the decomposition $\boldsymbol{p}_H = (H^\dagger \boldsymbol{r})_1 \boldsymbol{h}_1 + \boldsymbol{r}_{H_2}$ is not orthogonal. Rewriting $\boldsymbol{h}_1$ as $P_{S_2}\boldsymbol{h}_1 + P^\perp_{S_2}\boldsymbol{h}_1$ yields the orthogonal decomposition:

$$\boldsymbol{r} = \left(H^\dagger \boldsymbol{r}\right)_1 P^\perp_{S_2}\boldsymbol{h}_1 + (\boldsymbol{r}_{H_2} + P_{S_2}\boldsymbol{h}_1) + \boldsymbol{p}_{H^\perp}. \tag{4.28}$$

(4.26) is obtained directly from (4.28) by writing the inner product $\langle \boldsymbol{r}, P^\perp_{S_2}\boldsymbol{h}_1\rangle$ because the latter decomposition is orthogonal. The denominator in (4.26) is nonzero because of the full rankness of H. ■

Appendix 4.B Technical proofs

4.B.1 Proof of Lemma 4.4

Let us now start with the proof of Lemma 4.4, since this Lemma will be used later in the proofs of Lemmas 4.3, 4.5, and 4.6.

Since $\hat{\boldsymbol{x}}^{(k)}_{S_k}$ is the unconstrained least squares solution related to subset S_k, we have for $j < k$,

$$\begin{aligned}\hat{\boldsymbol{x}}^{(k)}_{S_k} &= \arg\min_{\boldsymbol{z}\in\mathbb{R}^k} \|\boldsymbol{y} - H_{S_k}\boldsymbol{z}\|^2 \\ &= \arg\min_{\boldsymbol{v},\boldsymbol{w}} \|\boldsymbol{y} - H_{S_j}\boldsymbol{v} - H_{S_k\setminus S_j}\boldsymbol{w}\|^2.\end{aligned} \tag{4.29}$$

In addition,

$$\begin{aligned}\min_{\boldsymbol{v},\boldsymbol{w}} \|\boldsymbol{y} - H_{S_j}\boldsymbol{v} - H_{S_k\setminus S_j}\boldsymbol{w}\|^2 &= \min_{\boldsymbol{w}}\big(\min_{\boldsymbol{v}} \|(\boldsymbol{y} - H_{S_k\setminus S_j}\boldsymbol{w}) - H_{S_j}\boldsymbol{v}\|^2\big) \\ &= \min_{\boldsymbol{w}} \|P^\perp_{S_j}(\boldsymbol{y} - H_{S_k\setminus S_j}\boldsymbol{w})\|^2 \\ &= \min_{\boldsymbol{w}} \|\boldsymbol{r}_j - \tilde{H}^{S_j}_{S_k\setminus S_j}\boldsymbol{w}\|^2.\end{aligned} \tag{4.30}$$

Since H_{S_k} is full column rank, $\tilde{H}^{S_j}_{S_k\setminus S_j}$ is full column rank as well according to Lemma 4.9. The minimum corresponding to (4.30) is reached for

$$\boldsymbol{w} = \left[\tilde{H}^{S_j}_{S_k\setminus S_j}\right]^\dagger \boldsymbol{r}_j \tag{4.31}$$

which identifies with $\hat{\boldsymbol{x}}^{(k)}_{S_k\setminus S_j}$ according to (4.29).

4.B.2 Proof of Lemma 4.3

We prove by induction that the supports found by OMP and NNOMP (respectively, by OLS and NNOLS/SNNOLS) coincide at each iteration.

4.B.2.1 NNOMP *vs* OMP

Let us first prove the claim at the first iteration. Let us denote by ℓ the index of the atom selected by OMP. The first OMP iterate is the one-sparse vector supported by $\{\ell\}$ such that $\hat{x}_\ell^{(1)} = \boldsymbol{h}_\ell^{\mathrm{t}}\boldsymbol{y}$. By assumption, $\hat{x}_\ell^{(1)} > 0$. Since ℓ is the unique solution to (4.7) with $\boldsymbol{r}_0 = \boldsymbol{y}$, we have:

$$\{\ell\} = \arg\max_i \{\boldsymbol{h}_i^{\mathrm{t}}\boldsymbol{y}\}, \tag{4.32}$$

which implies that ℓ is also selected by NNOMP. Since $\boldsymbol{h}_\ell^{\mathrm{t}}\boldsymbol{y} > 0$, $\hat{\boldsymbol{x}}^{(1)} \geq \boldsymbol{0}$ corresponds to both the unconstrained and non-negative solutions related to subset $\{\ell\}$, that is, to both NNOMP and OMP very first iterates.

Let us assume that OMP and NNOMP deliver the same support S_{k-1} and solution after $k-1$ iterations. The residual vectors $\boldsymbol{r}_{k-1}$ corresponding to OMP and NNOMP therefore coincide. Let $S_k = S_{k-1} \cup \{\ell\}$ denote the support found by OMP at iteration k with ℓ the index of the atom selected at iteration k, and let $\hat{\boldsymbol{x}}^{(k)}$ denote the OMP iterate. Lemma 4.4 with $j \leftarrow k-1$ implies that:

$$\hat{x}_\ell^{(k)} = \left[\tilde{\boldsymbol{h}}_\ell^{S_{k-1}}\right]^\dagger \boldsymbol{r}_{k-1} = \frac{\boldsymbol{r}_{k-1}^{\mathrm{t}}\tilde{\boldsymbol{h}}_\ell^{S_{k-1}}}{\|\tilde{\boldsymbol{h}}_\ell^{S_{k-1}}\|^2} \tag{4.33}$$

$$= \frac{\boldsymbol{r}_{k-1}^{\mathrm{t}}\boldsymbol{h}_\ell}{\|\tilde{\boldsymbol{h}}_\ell^{S_{k-1}}\|^2}. \tag{4.34}$$

$\tilde{\boldsymbol{h}}_\ell^{S_{k-1}} \neq \boldsymbol{0}$ in (4.33) follows from Lemma 4.9 and the full rankness assumption of H_{S_k}. (4.34) holds because $\tilde{\boldsymbol{h}}_\ell^{S_{k-1}} - \boldsymbol{h}_\ell$ lays in $\mathrm{span}(H_{S_{k-1}})$, and the OMP residual $\boldsymbol{r}_{k-1}$ is orthogonal to $\mathrm{span}(H_{S_{k-1}})$. By assumption, $\hat{\boldsymbol{x}}^{(k)} \geq \boldsymbol{0}$, so (4.34) implies that $\boldsymbol{r}_{k-1}^{\mathrm{t}}\boldsymbol{h}_\ell \geq 0$. Since ℓ is the unique solution to (4.7), we have:

$$\{\ell\} = \arg\max_{i \notin S_{k-1}} \{\boldsymbol{r}_{k-1}^{\mathrm{t}}\boldsymbol{h}_i\}. \tag{4.35}$$

So, ℓ is also selected by NNOMP at iteration k, leading to the same subset S_k as for OMP. Because the OMP iterate is non-negative, it is also the NNLS solution corresponding to S_k. Hence, NNOMP yields the same iterate as OMP.

4.B.2.2 SNNOLS *vs* OLS

We can make a similar argument as in the previous case by replacing the dictionary atoms $\boldsymbol{h}_i$ by their normalized projections $\tilde{\boldsymbol{g}}_i^S$.

At the first iteration, $\tilde{\boldsymbol{g}}_i^\emptyset = \tilde{\boldsymbol{h}}_i^\emptyset = \boldsymbol{h}_i$, so the very first iterates of SNNOLS and OLS respectively identify with those of NNOMP and OMP. They coincide according to § 4.B.2.1. At iteration k, the proof of § 4.B.2.1 can be repeated, where $\hat{\boldsymbol{x}}^{(k)}$ now denotes the OLS iterate. The right-hand side of (4.33) rereads $\left(\boldsymbol{r}_{k-1}^{\mathrm{t}}\tilde{\boldsymbol{g}}_\ell^{S_{k-1}}\right)/\|\tilde{\boldsymbol{h}}_\ell^{S_{k-1}}\| \geq 0$, so $\boldsymbol{r}_{k-1}^{\mathrm{t}}\tilde{\boldsymbol{g}}_\ell^{S_{k-1}} \geq 0$. Finally, since ℓ is the only maximizer of the OLS selection rule (4.8), we have

$$\{\ell\} = \arg\max_{i \notin S_{k-1}} \left\{\boldsymbol{r}_{k-1}^{\mathrm{t}}\tilde{\boldsymbol{g}}_i^{S_{k-1}}\right\}. \tag{4.36}$$

So, ℓ is also selected by SNNOLS. Similar to the NNOMP *vs* OMP case, we conclude that NNOLS and SNNOLS yield the same iterate at iteration k.

4.B.2.3 NNOLS *vs* OLS

The very first iterates of OLS and NNOLS identify to those of OMP and NNOMP, respectively. We have proved above that they coincide.

Assume that OLS and NNOLS deliver the same support S_{k-1} after $k-1$ iterations, and the same iterate $\hat{\boldsymbol{x}}^{(k)}_{S_{k-1}} = H^{\dagger}_{S_{k-1}}\boldsymbol{y} > \boldsymbol{0}$. Let ℓ denote the atom selected by OLS at iteration k, with $S_k = S_{k-1} \cup \{\ell\}$. Since OLS selects the atom yielding the least squared error and the optimal index ℓ is unique, we have

$$\forall i \notin S_k,\ \min_{\boldsymbol{z}} \|\boldsymbol{y} - H_{S_k}\boldsymbol{z}\|^2 < \min_{\boldsymbol{z}} \|\boldsymbol{y} - H_{S_{k-1}\cup\{i\}}\boldsymbol{z}\|^2. \tag{4.37}$$

Clearly,

$$\min_{\boldsymbol{z}} \|\boldsymbol{y} - H_{S_{k-1}\cup\{i\}}\boldsymbol{z}\|^2 \leq \min_{\boldsymbol{z}\geq\boldsymbol{0}} \|\boldsymbol{y} - H_{S_{k-1}\cup\{i\}}\boldsymbol{z}\|^2. \tag{4.38}$$

Moreover, since the OLS iterate at iteration k is non-negative, it is also an NNLS solution related to S_k. Therefore,

$$\min_{\boldsymbol{z}} \|\boldsymbol{y} - H_{S_k}\boldsymbol{z}\|^2 = \min_{\boldsymbol{z}\geq\boldsymbol{0}} \|\boldsymbol{y} - H_{S_k}\boldsymbol{z}\|^2. \tag{4.39}$$

From (4.37)-(4.39), we get

$$\forall i \notin S_k,\ \min_{\boldsymbol{z}\geq\boldsymbol{0}} \|\boldsymbol{y} - H_{S_k}\boldsymbol{z}\|^2 < \min_{\boldsymbol{z}\geq\boldsymbol{0}} \|\boldsymbol{y} - H_{S_{k-1}\cup\{i\}}\boldsymbol{z}\|^2,$$

which implies that ℓ is also selected by NNOLS. By assumption, H_{S_k} is full column rank, so the unconstrained and non-negative least-squares solutions related to S_k are unique. (4.39) implies that they coincide. Hence, OLS and NNOLS yield the same iterate at iteration k.

4.B.3 Proof of Lemma 4.5

In this proof, the abridged notations $\tilde{\boldsymbol{y}}$, $\tilde{\boldsymbol{h}}_i$, $\tilde{\boldsymbol{g}}_i$, $\tilde{\boldsymbol{c}}_i$, $\tilde{\boldsymbol{\epsilon}}$ correspond to projected vectors onto $\mathrm{span}(H_{S_{k-1}})^{\perp}$. In a somewhat counter-intuitive manner, our proof of Lemma 4.5 is not recursive: to prove that $\hat{x}^{(k)}_{S_k\setminus S_{k-1}} > 0$, we exploit that good atoms have been selected at previous iterations, but the current weight signs are not taken into consideration. We denote by ℓ the atom selected at iteration k, so that $S_k\setminus S_{k-1} = \{\ell\}$.

Let us first consider the case $K = 1$. Without loss of generality, we assume that $\boldsymbol{y} = x\boldsymbol{h}_1 + \boldsymbol{\epsilon}$ where $x > 0$. From Lemma 4.2, $\boldsymbol{h}_1$ is selected at the first iteration. Thus we have

$$\forall j > 1,\ |\boldsymbol{h}_1^{\mathrm{t}}\boldsymbol{y}| \geq |\boldsymbol{h}_j^{\mathrm{t}}\boldsymbol{y}|. \tag{4.40}$$

To prove that $\boldsymbol{h}_1^{\mathrm{t}}\boldsymbol{y} > 0$, we are going to exhibit an index $p > 1$ such that

$$\boldsymbol{h}_1^{\mathrm{t}}\boldsymbol{y} > -|\boldsymbol{h}_p^{\mathrm{t}}\boldsymbol{y}|. \tag{4.41}$$

(4.41) and (4.40) with $j \leftarrow p$ indeed imply that $\boldsymbol{h}_1^{\mathrm{t}}\boldsymbol{y} > 0$. From $\boldsymbol{y} = x\boldsymbol{h}_1 + \boldsymbol{\epsilon}$ we have

$$\begin{aligned}
\boldsymbol{h}_1^{\mathrm{t}}\boldsymbol{y} + \boldsymbol{h}_p^{\mathrm{t}}\boldsymbol{y} &= x + x\boldsymbol{h}_1^{\mathrm{t}}\boldsymbol{h}_p + \boldsymbol{\epsilon}^{\mathrm{t}}\boldsymbol{h}_1 + \boldsymbol{\epsilon}^{\mathrm{t}}\boldsymbol{h}_p && (4.42)\\
&\geq x - \mu(H)x - |\boldsymbol{\epsilon}^{\mathrm{t}}\boldsymbol{h}_1| - |\boldsymbol{\epsilon}^{\mathrm{t}}\boldsymbol{h}_p| && (4.43)\\
&\geq x(1-\mu(H)) - 2\|\boldsymbol{\epsilon}\| && (4.44)\\
&> 0 && (4.45)
\end{aligned}$$

wherein (4.45) is by $\|\boldsymbol{\epsilon}\| < \dfrac{1-\mu(H)}{2}x$ and (4.44) is by Cauchy-Schwarz inequality

$$|\boldsymbol{\epsilon}^{\mathrm{t}}\boldsymbol{h}_i| \leq \|\boldsymbol{\epsilon}\|\,\|\boldsymbol{h}_i\| \leq \|\boldsymbol{\epsilon}\|, \tag{4.46}$$

for $i=1$ and $i=p$. Now $\boldsymbol{h}_1^{\mathrm{t}}\boldsymbol{y}+\boldsymbol{h}_p^{\mathrm{t}}\boldsymbol{y}>0$ implies (4.41) so the case $K=1$ is proved.

Hereafter we consider the case where $K\geq 2$. Also, it should be noticed that $\mu(H)<\frac{1}{2K-1}$ implies that H_{S^*} is full column rank according to Lemma 4.7. Applying Lemma 4.4 for $j\leftarrow k-1$, we have

$$\hat{x}^{(k)}_{S_k\backslash S_{k-1}}=\tilde{\boldsymbol{h}}_\ell^{\mathrm{t}}\boldsymbol{r}_{k-1}. \tag{4.47}$$

Therefore, $\hat{x}^{(k)}_{S_k\backslash S_{k-1}}$ has the same sign as $\tilde{\boldsymbol{c}}_\ell^{\mathrm{t}}\boldsymbol{r}_{k-1}$ since $\tilde{\boldsymbol{c}}_\ell=\tilde{\boldsymbol{h}}_\ell$ or $\tilde{\boldsymbol{h}}_\ell/\|\tilde{\boldsymbol{h}}_\ell\|$ ($\tilde{\boldsymbol{h}}_\ell\neq\boldsymbol{0}$ according to Lemma 4.9). The remaining part of the proof consists in showing that $\tilde{\boldsymbol{c}}_\ell^{\mathrm{t}}\boldsymbol{r}_{k-1}>0$.

Since $\boldsymbol{h}_\ell$ is selected at the k-th iteration, we have

$$\forall j\notin S_k,\ |\tilde{\boldsymbol{c}}_\ell^{\mathrm{t}}\boldsymbol{r}_{k-1}|\geq|\tilde{\boldsymbol{c}}_j^{\mathrm{t}}\boldsymbol{r}_{k-1}|. \tag{4.48}$$

To prove that $\tilde{\boldsymbol{c}}_\ell^{\mathrm{t}}\boldsymbol{r}_{k-1}>0$, we are going to exhibit an index $p\notin S_k$ such that

$$\tilde{\boldsymbol{c}}_\ell^{\mathrm{t}}\boldsymbol{r}_{k-1}>-|\tilde{\boldsymbol{c}}_p^{\mathrm{t}}\boldsymbol{r}_{k-1}|. \tag{4.49}$$

(4.49) and (4.48) with $j\leftarrow p$ indeed imply that $\tilde{\boldsymbol{c}}_\ell^{\mathrm{t}}\boldsymbol{r}_{k-1}>0$.

Let us introduce the decomposition

$$\boldsymbol{r}_{k-1}=P^{\perp}_{S_{k-1}}\boldsymbol{y}=\sum_{i\notin S_{k-1}}x_i^*\tilde{\boldsymbol{h}}_i+\tilde{\boldsymbol{\epsilon}}. \tag{4.50}$$

For any $j\in S^*\backslash S_{k-1}$, we deduce

$$\tilde{\boldsymbol{c}}_j^{\mathrm{t}}\boldsymbol{r}_{k-1}=x_j^*\tilde{\boldsymbol{h}}_j^{\mathrm{t}}\tilde{\boldsymbol{c}}_j+\sum_{i\notin S_{k-1}\cup\{j\}}x_i^*\tilde{\boldsymbol{h}}_i^{\mathrm{t}}\tilde{\boldsymbol{c}}_j+\tilde{\boldsymbol{\epsilon}}^{\mathrm{t}}\tilde{\boldsymbol{c}}_j. \tag{4.51}$$

Clearly,

$$\tilde{\boldsymbol{h}}_j^{\mathrm{t}}\tilde{\boldsymbol{c}}_j=\begin{cases}\|\tilde{\boldsymbol{h}}_j\|^2 & \text{for OMP,}\\ \|\tilde{\boldsymbol{h}}_j\| & \text{for OLS.}\end{cases} \tag{4.52}$$

Since for $K\geq 2$, $\mu(H)<\frac{1}{2K-1}\leq\frac{1}{k+1}$, Lemma 4.8 yields:

$$\tilde{\boldsymbol{h}}_j^{\mathrm{t}}\tilde{\boldsymbol{c}}_j\geq\eta_{k-1}, \tag{4.53}$$

$$|\tilde{\boldsymbol{h}}_i^{\mathrm{t}}\tilde{\boldsymbol{c}}_j|\leq\mu_{k-1}\eta_{k-1} \tag{4.54}$$

with μ_{k-1} and η_{k-1} defined in (4.24)-(4.25). By Cauchy-Schwarz inequality,

$$|\tilde{\boldsymbol{\epsilon}}^{\mathrm{t}}\tilde{\boldsymbol{c}}_j|\leq\|\tilde{\boldsymbol{\epsilon}}\|\|\tilde{\boldsymbol{c}}_j\|\leq\|\tilde{\boldsymbol{\epsilon}}\| \tag{4.55}$$

as $\|\tilde{\boldsymbol{c}}_j\|\leq 1$. From (4.51), we get for $j\in S^*\backslash S_{k-1}$,

$$\tilde{\boldsymbol{c}}_j^{\mathrm{t}}\boldsymbol{r}_{k-1}\geq\eta_{k-1}\Big(x_j^*-\mu_{k-1}\sum_{i\notin S_{k-1}\cup\{j\}}x_i^*\Big)-\|\tilde{\boldsymbol{\epsilon}}\|. \tag{4.56}$$

Notice that there are $K-k$ nonzero terms in the sum in (4.56) since $\boldsymbol{x}^*$ is K-sparse. The latter sum can then be upper bounded as:

$$\sum_{i\notin S_{k-1}\cup\{j\}}x_i^*\leq\begin{cases}x_\ell^*+(K-k-1)x_p^* & \text{if } j\neq\ell,\\ (K-k)x_p^* & \text{if } j=\ell\end{cases} \tag{4.57}$$

with p defined as:

$$p\in\operatorname*{arg\,max}_{i\in S^*\backslash S_k}x_i^*. \tag{4.58}$$

Apply (4.56) twice with $j \leftarrow p$ and $j \leftarrow \ell$. We get

$$\tilde{\boldsymbol{c}}_p^{\mathrm{t}} \boldsymbol{r}_{k-1} \geq \eta_{k-1}\big((1-(K-k-1)\mu_{k-1})x_p^* - \mu_{k-1}x_\ell^*\big) - \|\tilde{\boldsymbol{\epsilon}}\|,$$
$$\tilde{\boldsymbol{c}}_\ell^{\mathrm{t}} \boldsymbol{r}_{k-1} \geq \eta_{k-1}\big(x_\ell^* - (K-k)\mu_{k-1}x_p^*\big) - \|\tilde{\boldsymbol{\epsilon}}\|,$$

from which we deduce

$$\tilde{\boldsymbol{c}}_p^{\mathrm{t}} \boldsymbol{r}_{k-1} + \tilde{\boldsymbol{c}}_\ell^{\mathrm{t}} \boldsymbol{r}_{k-1} \geq \eta_{k-1}(\tau_1 x_\ell^* + \tau_2 x_p^*) - 2\|\tilde{\boldsymbol{\epsilon}}\|, \tag{4.59}$$

with

$$\tau_1 := 1 - \mu_{k-1}, \tag{4.60}$$
$$\tau_2 := 1 - (2K-2k-1)\mu_{k-1}. \tag{4.61}$$

According to (4.24), $\tau_1 > 0$ if $\mu(H) < \frac{1}{k}$ and $\tau_2 > 0$ if $(2K-k-2)\mu(H) < 1$. Since $\mu(H) < \frac{1}{2K-1}$, it holds true that $\tau_1 > 0$ and $\tau_2 > 0$. Let us define

$$q \in \operatorname*{arg\,min}_{i \in S^*} x_i^*. \tag{4.62}$$

Then

$$\eta_{k-1}(\tau_1 x_\ell^* + \tau_2 x_p^*) - 2\|\tilde{\boldsymbol{\epsilon}}\| \geq \eta_{k-1}(\tau_1 + \tau_2)x_q^* - 2\|\tilde{\boldsymbol{\epsilon}}\|. \tag{4.63}$$

According to (4.21), $\beta_{k-1} \leq 1$ which implies $\beta_{k-1} \leq \sqrt{\beta_{k-1}} \leq 1$ hence $\eta_{k-1} \geq \beta_{k-1}$ by definition. Besides,

$$\|\tilde{\boldsymbol{\epsilon}}\| \leq \|\boldsymbol{\epsilon}\| < \frac{1-(2K-1)\mu(H)}{2} x_q^*. \tag{4.64}$$

Therefore,

$$\eta_{k-1}(\tau_1+\tau_2)x_q^* - 2\|\tilde{\boldsymbol{\epsilon}}\| > \beta_{k-1}(\tau_1+\tau_2)x_q^* - (1-(2K-1)\mu(H))\, x_q^*. \tag{4.65}$$

According to (4.60)-(4.61) and (4.23)-(4.24),

$$\beta_{k-1}(\tau_1+\tau_2) = 2\beta_{k-1}(1-(K-k)\mu_{k-1}) \tag{4.66}$$
$$= \frac{2(1+\mu(H))(1-(k-1)\mu(H))}{1-(k-2)\mu(H)} \left(1 - \frac{(K-k)\mu(H)}{1-(k-1)\mu(H)}\right) \tag{4.67}$$
$$= \frac{2(1+\mu(H))(1-(k-1)\mu(H))}{1-(k-2)\mu(H)} \frac{1-(K-1)\mu(H)}{1-(k-1)\mu(H)} \tag{4.68}$$
$$= \frac{2(1+\mu(H))(1-(K-1)\mu(H))}{1-(k-2)\mu(H)}. \tag{4.69}$$

As $k \geq 1$, we have $1-(k-2)\mu(H) \leq 1+\mu(H)$ so

$$\frac{2(1+\mu(H))(1-(K-1)\mu(H))}{1-(k-2)\mu(H)} \geq 2(1-(K-1)\mu(H)). \tag{4.70}$$

As a result,

$$\beta_{k-1}(\tau_1+\tau_2) \geq 2(1-(K-1)\mu(H)). \tag{4.71}$$

From (4.65) and (4.71) we have

$$\eta_{k-1}(\tau_1+\tau_2)x_q^* - 2\|\tilde{\boldsymbol{\epsilon}}\| > 2(1-(K-1)\mu(H))x_q^* - (1-(2K-1)\mu(H))x_q^* \tag{4.72}$$
$$= (1+\mu(H))x_q^* \tag{4.73}$$
$$> 0. \tag{4.74}$$

Thus, from (4.63) we get $\eta_{k-1}(\tau_1 x_\ell^* + \tau_2 x_p^*) - 2\|\tilde{\boldsymbol{\epsilon}}\| > 0$.

Hence, from (4.59) we get $\tilde{\boldsymbol{c}}_p^{\mathrm{t}} \boldsymbol{r}_{k-1} + \tilde{\boldsymbol{c}}_\ell^{\mathrm{t}} \boldsymbol{r}_{k-1} > 0$, which implies (4.49). We thus conclude that $\tilde{\boldsymbol{c}}_\ell^{\mathrm{t}} \boldsymbol{r}_{k-1}$ and $\hat{x}^{(k)}_{S_k \setminus S_{k-1}}$ are positive quantities.

4.B.4 Proof of Lemma 4.6

In this proof, the abridged notations $\tilde{\boldsymbol{h}}_i$, $\tilde{\boldsymbol{g}}_i$, $\tilde{\boldsymbol{c}}_i$ correspond to projected vectors onto $\mathrm{span}(H_{S_{j-1}})^\perp$ (the latter space being different from that of Subsection 4.B.3). Similarly, we will use the abridged notations $\tilde{H}$, $\tilde{G}$, $\tilde{C}$ to refer to $\tilde{H}^{S_{j-1}}$, $\tilde{G}^{S_{j-1}}$ and $\tilde{C}^{S_{j-1}}$, respectively. For any subset S, the Gram matrices of $\tilde{H}_S$ and $\tilde{G}_S$ will be respectively denoted by $Gr_S^{\mathrm{OMP}} := \tilde{H}_S^{\mathrm{t}}\tilde{H}_S$ and $Gr_S^{\mathrm{OLS}} := \tilde{G}_S^{\mathrm{t}}\tilde{G}_S$. For statements that are common to both OMP and OLS, we will use the simpler generic notation $Gr_S := \tilde{C}_S^{\mathrm{t}}\tilde{C}_S$.

The proof of Lemma 4.6 is not recursive. However, in order to prove that $\hat{\boldsymbol{x}}^{(k)}_{S_{k-1}} > \mathbf{0}$, we use the fact that $\hat{x}^{(j)}_{S_j\backslash S_{j-1}} > 0$ for $j < k$, which holds according to Lemma 4.5.

Let $\ell \in S_{k-1}$ and denote by $j < k$ the iteration at which the atom indexed by ℓ has been selected by Oxx, so that $S_j\backslash S_{j-1} = \{\ell\}$. According to Lemma 4.4 and since ℓ is the first entry in the ordered set $S_k\backslash S_{j-1}$, we have

$$\hat{x}_\ell^{(k)} = \big(\tilde{H}^\dagger_{S_k\backslash S_{j-1}}\boldsymbol{r}_{j-1}\big)_1. \tag{4.75}$$

In order to exploit the Oxx selection rule which is based on the atoms $\tilde{c}_i$ defined in (4.4), let us rewrite (4.75) with respect to matrix $\tilde{C}$. Obviously, one can rewrite $\tilde{H}_{S_k\backslash S_{j-1}} = \tilde{C}_{S_k\backslash S_{j-1}}\Delta$, where $\Delta > \mathbf{0}$ is a square diagonal matrix whose diagonal elements are either equal to 1 (OMP, $\tilde{C} \leftarrow \tilde{H}$) or to $\|\tilde{\boldsymbol{h}}_i\| > 0$, $i \in S_k\backslash S_{j-1}$ (OLS, $\tilde{C} \leftarrow \tilde{G}$). The positivity property is related to the full rankness of $\tilde{C}_{S_k\backslash S_{j-1}}$, which deduces from that of H_{S_k} according to Lemma 4.9. From standard properties of pseudo-inverses, we have $\tilde{H}^\dagger_{S_k\backslash S_{j-1}} = \Delta^{-1}\tilde{C}^\dagger_{S_k\backslash S_{j-1}}$. (4.75) yields

$$\hat{x}_\ell^{(k)} \;\propto^+\; \big(\tilde{C}^\dagger_{S_k\backslash S_{j-1}}\boldsymbol{r}_{j-1}\big)_1 \;=\; \Big(\big[\tilde{\boldsymbol{c}}_\ell, \tilde{C}_{S_k\backslash S_j}\big]^\dagger \boldsymbol{r}_{j-1}\Big)_1 \tag{4.76}$$

$$= \frac{\big\langle \boldsymbol{r}_{j-1}, \tilde{P}^\perp_{S_k\backslash S_j}\tilde{\boldsymbol{c}}_\ell\big\rangle}{\big\|\tilde{P}^\perp_{S_k\backslash S_j}\tilde{\boldsymbol{c}}_\ell\big\|^2} \tag{4.77}$$

where $\propto^+$ indicates proportionality up to a positive factor, $\tilde{P}^\perp_{S_k\backslash S_j}$ denotes the orthogonal projection onto the orthogonal complement of $\mathrm{span}(\tilde{C}_{S_k\backslash S_j})$, and (4.77) deduces from Lemma 4.10.

(4.77) implies that:

$$\hat{x}_\ell^{(k)} > 0 \iff \langle \tilde{P}^\perp_{S_k\backslash S_j}\boldsymbol{r}_{j-1}, \tilde{\boldsymbol{c}}_\ell\rangle > 0 \tag{4.78}$$

$$\iff \langle \tilde{C}_{S_k\backslash S_j}\tilde{C}^\dagger_{S_k\backslash S_j}\boldsymbol{r}_{j-1}, \tilde{\boldsymbol{c}}_\ell\rangle < \langle \boldsymbol{r}_{j-1}, \tilde{\boldsymbol{c}}_\ell\rangle \tag{4.79}$$

$$\iff \langle \tilde{C}^\dagger_{S_k\backslash S_j}\boldsymbol{r}_{j-1}, \tilde{C}^{\mathrm{t}}_{S_k\backslash S_j}\tilde{\boldsymbol{c}}_\ell\rangle < \langle \boldsymbol{r}_{j-1}, \tilde{\boldsymbol{c}}_\ell\rangle. \tag{4.80}$$

By Hölder's inequality and since $\tilde{C}^\dagger_{S_k\backslash S_j} = Gr^{-1}_{S_k\backslash S_j}\tilde{C}^{\mathrm{t}}_{S_k\backslash S_j}$, the left-hand side (LHS) of (4.80) is upper bounded by

$$\begin{aligned}\|\tilde{C}^{\mathrm{t}}_{S_k\backslash S_j}\tilde{\boldsymbol{c}}_\ell\|_\infty\|\tilde{C}^\dagger_{S_k\backslash S_j}\boldsymbol{r}_{j-1}\|_1 &\le \|\tilde{C}^{\mathrm{t}}_{S_k\backslash S_j}\tilde{\boldsymbol{c}}_\ell\|_\infty\|Gr^{-1}_{S_k\backslash S_j}\|_{1,1}\|\tilde{C}^{\mathrm{t}}_{S_k\backslash S_j}\boldsymbol{r}_{j-1}\|_1\\ &\le \mu(\tilde{C})\,\|Gr^{-1}_{S_k\backslash S_j}\|_{1,1}\|\tilde{C}^{\mathrm{t}}_{S_k\backslash S_j}\boldsymbol{r}_{j-1}\|_1\\ &\le \mu(\tilde{C})\,\|Gr^{-1}_{S_k\backslash S_j}\|_{1,1}\,(k-j)\,|\langle \boldsymbol{r}_{j-1}, \tilde{\boldsymbol{c}}_\ell\rangle|.\end{aligned} \tag{4.81}$$

To obtain the last inequality, we exploit that $\boldsymbol{h}_\ell$ has been selected at the j-th iteration of Oxx, therefore:

$$\forall i \in S_k\backslash S_j,\ |\langle \boldsymbol{r}_{j-1}, \tilde{\boldsymbol{c}}_i\rangle| \le |\langle \boldsymbol{r}_{j-1}, \tilde{\boldsymbol{c}}_\ell\rangle|. \tag{4.82}$$

From Lemma 4.5, we have $\hat{x}_\ell^{(j)} > 0$, and as already remarked in the proof of Lemma 4.5, $\tilde{\boldsymbol{c}}_\ell^{\mathrm{t}} \boldsymbol{r}_{j-1}$ is of same sign as $\hat{x}_\ell^{(j)}$, see (4.47). The upper bound in (4.81) thus rewrites

$$(k-j)\,\mu(\tilde{C})\,\|Gr^{-1}_{S_k\backslash S_j}\|_{1,1}\,\langle \boldsymbol{r}_{j-1}, \tilde{\boldsymbol{c}}_\ell\rangle \tag{4.83}$$

and we deduce from (4.80) that

$$\big((k-j)\mu(\tilde{C})\|Gr^{-1}_{S_k\backslash S_j}\|_{1,1} < 1\big) \Longrightarrow \big(\hat{x}_\ell^{(k)} > 0\big). \tag{4.84}$$

Let us now provide some upper bounds on $\mu(\tilde{C})$ and $\|Gr^{-1}_{S_k\backslash S_j}\|_{1,1}$ in order to show that the LHS of (4.84) holds true.

4.B.4.1 Upper bound on $\mu(\tilde{C})$

Since $\mu(H) < \frac{1}{2K-1} < \frac{1}{j}$, Lemma 4.8 yields:

$$\forall i \notin S_{j-1},\ \|\tilde{\boldsymbol{h}}_i\|^2 \geq \beta_{j-1}, \tag{4.85}$$

$$\forall p \neq q,\ |\tilde{\boldsymbol{h}}_p^{\mathrm{t}} \tilde{\boldsymbol{h}}_q| \leq \mu_{j-1}\beta_{j-1}, \tag{4.86}$$

with μ_{j-1} and β_{j-1} defined in (4.23)-(4.24), from which we can easily deduce that

$$\mu(\tilde{C}) = \begin{cases} \mu(\tilde{H}) \ \leq \ \mu_{j-1}\beta_{j-1} & \text{(OMP case)}, \\ \mu(\tilde{G}) \ \leq \mu_{j-1} & \text{(OLS case)}. \end{cases} \tag{4.87}$$

4.B.4.2 Upper bound on $\|Gr^{-1}_{S_k\backslash S_j}\|_{1,1}$ in the OLS case

We have shown that for $i \in S_k\backslash S_j$, $\tilde{\boldsymbol{h}}_i \neq \boldsymbol{0}$, so matrix $\tilde{G}_{S_k\backslash S_j}$ is column-normalized, hence $Gr^{\mathrm{OLS}}_{S_k\backslash S_j}$ has a unit diagonal. Since $\mu(H) < \frac{1}{2K-1}$, (4.87) and (4.24) imply that:

$$\mu(\tilde{G}) \leq \mu_{j-1} \leq \frac{1}{2K-j} \leq \frac{1}{k-j}. \tag{4.88}$$

Lemma 4.7 then applies to matrix $\tilde{G}_{S_k\backslash S_j} \in \mathbb{R}^{m\times(k-j)}$, so the latter is full column rank, and

$$\big\|\big[Gr^{\mathrm{OLS}}_{S_k\backslash S_j}\big]^{-1}\big\|_{1,1} \leq \frac{1}{1-(k-j-1)\mu(\tilde{G}_{S_k\backslash S_j})} \leq \frac{1}{1-(k-j-1)\mu(\tilde{G})}. \tag{4.89}$$

It follows from (4.87) that

$$\begin{aligned} \mu(\tilde{G})\big\|\big[Gr^{\mathrm{OLS}}_{S_k\backslash S_j}\big]^{-1}\big\|_{1,1} &\leq \mu_{j-1}\big\|\big[Gr^{\mathrm{OLS}}_{S_k\backslash S_j}\big]^{-1}\big\|_{1,1} \\ &\leq \frac{\mu_{j-1}}{1-(k-j-1)\mu(\tilde{G})}, \\ &\leq \frac{1}{2K-k+1} \qquad (4.90) \\ &< \frac{1}{k-j} \qquad (4.91) \end{aligned}$$

where (4.90) follows from the second upper bound in (4.88). (4.91) thus implies that the LHS of (4.84) is true in the OLS case.

4.B.4.3 Upper bound on $\|Gr^{-1}_{S_k\setminus S_j}\|_{1,1}$ in the OMP case

Contrary to the OLS case, the diagonal elements of $Gr^{\mathrm{OMP}}_{S_k\setminus S_j}$ ($\|\tilde{\boldsymbol{h}}_i\|^2$, $i \in S_k\setminus S_j$) are not equal to 1, so Lemma 4.7 does not apply. Let Δ be the square diagonal matrix with the elements $\|\tilde{\boldsymbol{h}}_i\|$, $i \in S_k\setminus S_j$ on its diagonal. Clearly, $\tilde{H}_{S_k\setminus S_j} = \tilde{G}_{S_k\setminus S_j}\Delta$, hence

$$\left[Gr^{\mathrm{OMP}}_{S_k\setminus S_j}\right]^{-1} = \Delta^{-1}\left[Gr^{\mathrm{OLS}}_{S_k\setminus S_j}\right]^{-1}\Delta^{-1}, \tag{4.92}$$

and thus

$$\left\|\left[Gr^{\mathrm{OMP}}_{S_k\setminus S_j}\right]^{-1}\right\|_{1,1} \leq \left\|\left[Gr^{\mathrm{OLS}}_{S_k\setminus S_j}\right]^{-1}\right\|_{1,1} \|\Delta^{-1}\|^2_{1,1}. \tag{4.93}$$

Moreover,

$$\|\Delta^{-1}\|^2_{1,1} = \frac{1}{\min_{i\in S_k\setminus S_j}\|\tilde{\boldsymbol{h}}_i\|^2} \leq \frac{1}{\beta_{j-1}} \tag{4.94}$$

by (4.85). We have thus

$$\mu(\tilde{H})\left\|\left[Gr^{\mathrm{OMP}}_{S_k\setminus S_j}\right]^{-1}\right\|_{1,1} \leq \frac{\mu(\tilde{H})}{\beta_{j-1}}\left\|\left[Gr^{\mathrm{OLS}}_{S_k\setminus S_j}\right]^{-1}\right\|_{1,1} \leq \mu_{j-1}\left\|\left[Gr^{\mathrm{OLS}}_{S_k\setminus S_j}\right]^{-1}\right\|_{1,1}$$

according to (4.87). From (4.91), we conclude that the LHS of (4.84) is also true in the OMP case.

Chapter 5

NP-hardness of ℓ_0 minimization problems[1]

Contents

In chapters 2-4, we addressed the algorithmic issues and theoretical analysis of NNOG algorithms. In particular, we exhibited structural properties and proposed a fast implementation of NNOG algorithms. Moreover, we established an exact support recovery analysis of NNOG algorithms under the standard mutual coherence based condition. Note that NNOG algorithms are designed to address the constrained non-negative ℓ_0 minimization problems. In this chapter, we discuss about NP-hardness of non-negative ℓ_0 minimization problems.

5.1 Introduction

Sparse approximation appears in a wide range of applications, especially in signal processing, image processing and compressed sensing [Elad, 2010]. Given a signal data $\boldsymbol{y} \in \mathbb{R}^m$ and a dictionary H of size $m \times n$, the aim is to find a signal $\boldsymbol{x} \in \mathbb{R}^n$ that gives the best approximation $\boldsymbol{y} \approx H\boldsymbol{x}$ and has the fewest non-zero coefficients (*i.e.*,, sparsest solution). This task leads to solving one of the following constrained or penalized ℓ_0 minimization problems:

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{x}\|_0 \text{ s.t. } \|\boldsymbol{y} - H\boldsymbol{x}\|_2 \leq \epsilon \qquad (\ell_0 C_\epsilon)$$

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{y} - H\boldsymbol{x}\|_2^2 \text{ s.t. } \|\boldsymbol{x}\|_0 \leq K \qquad (\ell_0 C_K)$$

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{y} - H\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_0 \qquad (\ell_0 P)$$

in which ϵ, K and λ are positive quantities related to the noise standard deviation, the sparsity level and regularization strength, respectively. Letters C and P respectively indicate that the problem is constrained or penalized. Depending on application, the appropriate statement will be addressed. It is noteworthy that n and K often depend on m when one considers the size of problem. $(\ell_0 C_\epsilon)$ and $(\ell_0 C_K)$ are well known to be NP-hard [Natarajan, 1995, Davis et al., 1997]. The NP-hardness of $(\ell_0 P)$ was claimed to be a particular case of more general complexity analyses in [Chen et al., 2014, Huo and Chen, 2010]. However, we point out hereafter that these complexity analyses do not rigorously apply to $(\ell_0 P)$ as claimed. Then we provide a new proof for the NP-hardness of $(\ell_0 P)$ adapted from Natarajan's construction [Natarajan, 1995].

[1] This chapter is an adaptation of the paper [Nguyen et al., 2019c]

In several applications the signal or image of interest is both sparse and non-negative. Therefore, many researchers are interested in non-negative ℓ_0 minimization problems as follows:

$$\min_{\boldsymbol{x}\geq\mathbf{0}} \quad \|\boldsymbol{x}\|_0 \text{ s.t. } \|\boldsymbol{y}-H\boldsymbol{x}\|_2 \leq \epsilon \qquad (\ell_0 C_\epsilon+)$$

$$\min_{\boldsymbol{x}\geq\mathbf{0}} \quad \|\boldsymbol{y}-H\boldsymbol{x}\|_2^2 \text{ s.t. } \|\boldsymbol{x}\|_0 \leq K \qquad (\ell_0 C_K+)$$

$$\min_{\boldsymbol{x}\geq\mathbf{0}} \quad \|\boldsymbol{y}-H\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_0 \qquad (\ell_0 P+)$$

Several papers address non-negative ℓ_0 minimization problems in the literature (see, *e.g.*,, [Bruckstein et al., 2008, Peharz and Pernkopf, 2012, Yaghoobi et al., 2015, Wang et al., 2018]). However, to the best of our knowledge, the complexity of these problems has not been addressed yet, the question of their NP-hardness being still open. Here we show that these problems are NP-hard and the proof can be derived from the NP-hardness of ℓ_0 minimization problems.

5.2 Background on NP-hardness and constrained ℓ_0 minimization problems

Let us recall that an NP-complete problem is a problem in NP to which any other problem in NP can be reduced in polynomial time. Thus NP-complete problems are identified as the hardest problems in NP. An NP-complete problem is strongly NP-complete if it remains NP-complete when all of its numerical parameters are bounded by a polynomial in the length of the input. NP-hard problems are at least as hard as NP-complete problems. However, NP-hard problems do not need to be in NP and do not need to be decision problems. Formally, a problem is NP-hard (respectively, strongly NP-hard) if a NP-complete (respectively, strongly NP-complete) problem can be reduced in polynomial time to it. The reader is referred to [Garey and Johnson, 1979, Leeuwen, 1990] for more information on this topic.

In the literature, problem ($\ell_0 C_\epsilon$), called SAS in [Natarajan, 1995], is well known to be NP-hard [Natarajan, 1995, Theorem 1]. The NP-hardness of ($\ell_0 C_\epsilon$) is a valuable extension of an earlier result: the problem of minimum weight solution to linear equations (equivalent to ($\ell_0 C_\epsilon$) with $\epsilon = 0$) is NP-hard [Garey and Johnson, 1979, p. 246]. Davis *et al.* proved that ($\ell_0 C_K$), called M-optimal approximation in [Davis et al., 1997], is NP-hard for any $K < m$ [Davis et al., 1997, Theorem 2.1]. Both analyses of Natarajan and Davis were made by a polynomial time reduction from the "exact cover by 3-sets" problem[2] which is known to be NP-complete [Garey and Johnson, 1979, p. 221].

5.3 Existing analyses on penalized ℓ_0 minimization

In [Chen et al., 2014, Huo and Chen, 2010], the NP-hardness of ($\ell_0 P$) is deduced as a particular case of more general complexity analyses. However, it turns out that the latter do not apply to ($\ell_0 P$), as explained hereafter. Chen *et al.* [Chen et al., 2014] address the unconstrained ℓ_q-ℓ_p minimization problem, defined by:

$$\min_{\boldsymbol{x}} \|\boldsymbol{y}-H\boldsymbol{x}\|_q^q + \lambda\|\boldsymbol{x}\|_p^p \qquad (\ell_q\text{-}\ell_p)$$

where $\lambda > 0$, $q \geq 1$ and $0 \leq p < 1$. The authors showed that problem (ℓ_q-ℓ_p) is NP-hard with any $\lambda > 0$, $q \geq 1$ and $0 \leq p < 1$ [Chen et al., 2014, Theorem 3]. Obviously, ($\ell_0 P$) is the case

[2]The latter problem, denoted by X3C in [Natarajan, 1995, Garey and Johnson, 1979], is stated as follows: Given a set S and a collection C of 3-element subsets of S (called triplets), is there a subcollection of disjoint triplets that exactly covers S?

where $q = 2$ and $p = 0$. The proof was done by i) introducing an invertible transformation which scales any instance of problem (ℓ_q-ℓ_p) to the problem (ℓ_q-ℓ_p) with $\lambda = 1/2$, and ii) establishing a polynomial time reduction from the partition problem which is known to be NP-complete [Garey and Johnson, 1979] to the problem (ℓ_q-ℓ_p) with $\lambda = 1/2$. In other words, they showed that problem (ℓ_q-ℓ_p) with $\lambda = 1/2$ is NP-hard and, because there exists an invertible transformation from any problem (ℓ_q-ℓ_p) to the one with $\lambda = 1/2$, every problem (ℓ_q-ℓ_p) is NP-hard. Similarly, they showed that (ℓ_q-ℓ_p) is strongly NP-hard [Chen et al., 2014, Theorem 5] by a reduction from the 3-partition problem which is known to be strongly NP-hard [Garey and Johnson, 1979]. The invertible transform used in [Chen et al., 2014] is defined by:

$$\tilde{\boldsymbol{x}} = (2\lambda)^{1/p}\boldsymbol{x}, \quad \tilde{H} = (2\lambda)^{-1/p}H. \tag{5.1}$$

Unfortunately, (5.1) is not well-defined when $p = 0$. Therefore, [Chen et al., 2014, Theorems 3 and 5] do not apply to ($\ell_0 P$) when $\lambda \neq 1/2$.

Using a different approach, Huo and Chen's paper [Huo and Chen, 2010] addresses the penalized least-squares problem defined by:

$$\min_{\boldsymbol{x}} \|\boldsymbol{y} - H\boldsymbol{x}\|_2^2 + \lambda \sum_{i=1}^{n} \phi(|x_i|), \tag{PLS}$$

where ϕ is a penalty function mapping non-negative values to non-negative values. The authors showed that (PLS) is NP-hard if the penalty function ϕ satisfies the following four conditions [Huo and Chen, 2010, Theorem 3.1]:

C1. $\phi(0) = 0$ and $\forall\, 0 \leq \tau_1 < \tau_2$, $\phi(\tau_1) \leq \phi(\tau_2)$.

C2. There exists $\tau_0 > 0$ and a constant $d > 0$ such that

$$\phi(\tau) \geq \phi(\tau_0) - d(\tau_0 - \tau)^2$$

for every $0 \leq \tau < \tau_0$.

C3. For the aforementioned τ_0, if $\tau_1, \tau_2 < \tau_0$ then

$$\phi(\tau_1) + \phi(\tau_2) \geq \phi(\tau_1 + \tau_2).$$

C4. For every $0 \leq \tau < \tau_0$,

$$\phi(\tau) + \phi(\tau_0 - \tau) > \phi(\tau_0). \tag{5.2}$$

The proof of [Huo and Chen, 2010, Theorem 3.1] is by a reduction from the NP-complete problem X3C to the decision version of (PLS); this leads to the NP-completeness of the decision version of (PLS) and so the NP-hardness of (PLS) [Huo and Chen, 2010, Appendix 1]. The authors claimed that the ℓ_0 penalty function satisfies conditions C1-C4 for $\tau_0 = d = 1$. Therefore, the (PLS) problem with the ℓ_0 penalty function is NP-hard [Huo and Chen, 2010, Corollary 3.2]. Unfortunately, it turns out that the ℓ_0 penalty does not fulfill condition C4 as claimed. Indeed, for $\tau = 0$ the strict inequality (5.2) becomes $\phi(0) > 0$. Besides, in the proof [Huo and Chen, 2010, Appendix 1], the inputs of the decision problem are not guaranteed to have rational values. This might also violate the polynomiality of the reduction. Therefore, [Huo and Chen, 2010, Theorem 3.1] does not apply to ($\ell_0 P$).

In [Huo and Chen, 2010], the authors also mention an alternate proof of NP-hardness of ($\ell_0 P$) from Huo and Ni's earlier paper [Huo and Ni, 2007] as a special case of their results. In this proof [Huo and Ni, 2007, Appendix A.1], the relation between ($\ell_0 P$) and ($\ell_0 C_\epsilon$) is established using the principle of Lagrange multiplier. More precisely, the authors introduce an instance of ($\ell_0 C_\epsilon$)

in which ϵ is defined from the minimizer of $(\ell_0 P)$ and argue that solving $(\ell_0 P)$ is equivalent to solving the mentioned instance of $(\ell_0 C_\epsilon)$, which is known to be NP-hard [Natarajan, 1995]. There are a number of issues in the NP-hardness proof in [Huo and Ni, 2007]. For instance, the proposed transformation between $(\ell_0 P)$ and $(\ell_0 C_\epsilon)$ is not a polynomial time reduction. Besides, it is well known that $(\ell_0 P)$ and $(\ell_0 C_\epsilon)$ are not equivalent [Nikolova, 2013].

5.4 New analysis on penalized ℓ_0 minimization

To prove that a problem T is NP-hard, one must establish a polynomial time reduction (briefly called reduction hereafter) from some known NP-hard or NP-complete problem to T [Leeuwen, 1990]. Roughly speaking, the reduction from a problem T1 to another problem T2 implies that T1 is not harder than T2. Therefore, if there exists a reduction from T1 to T2 and if T1 is NP-hard, T2 must be NP-hard too. The NP-hardness proofs in [Natarajan, 1995] and [Davis et al., 1997] use this principle. As an adaptation of Natarajan's construction, we prove the NP-hardness of $(\ell_0 P)$ using the same principle as follows.

Theorem 5.1. *Problem $(\ell_0 P)$ is NP-hard.*

The proof is by a reduction from the known NP-complete problem X3C to $(\ell_0 P)$. The proof contains three steps: (1) Construct an instance of $(\ell_0 P)$ from a given instance of X3C; (2) Construct a solution of $(\ell_0 P)$ from a solution of X3C; (3) Construct a solution of X3C from a solution of $(\ell_0 P)$.

We only need to consider the case where $0 < \lambda < 3$. For the case where $\lambda \geq 3$, we can always scale[3] the problem to the case where $\lambda < 3$ by respectively replacing $\boldsymbol{y}$, H and λ by $\frac{1}{\sqrt{t}}\boldsymbol{y}$, $\frac{1}{\sqrt{t}}H$ and $\frac{\lambda}{t}$ where $t > 1$ is a positive integer satisfying $3 \leq \lambda < 3t$.

Construction of an instance of $(\ell_0 P)$ from a given instance of X3C

Given an instance of X3C: $S = \{s_1, s_2, ..., s_m\}$ is a set of m elements. C is a collection of n triplets c_j, $1 \leq j \leq n$. Without loss of generality we can assume that m is a multiple of 3 since otherwise there is trivially no exact cover so no solution of X3C.

We now construct an instance of $(\ell_0 P)$. Let $\boldsymbol{y} = [1, 1, ..., 1]^T \in \mathbb{R}^m$. Let $H = (h_{ij})_{1 \leq i \leq m, 1 \leq j \leq n}$ where $h_{ij} = 1$ if $s_i \in c_j$ and $h_{ij} = 0$ otherwise. Let $\lambda \in \mathbb{Q}$, $0 < \lambda < 3$. Let

$$F(\boldsymbol{x}) := \|\boldsymbol{y} - H\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_0. \tag{5.3}$$

Construction of a solution of $(\ell_0 P)$ from a solution of X3C

Assume that there is a subcollection of disjoint triplets $\hat{C}$ which exactly covers S. Let $\boldsymbol{x}^* = [x_1^*, x_2^*, ..., x_n^*]^T$ where $x_j^* = 1$ if $c_j \in \hat{C}$ and $x_j^* = 0$ otherwise. We will prove that $\boldsymbol{x}^*$ is a solution of $(\ell_0 P)$.

Since $\hat{C}$ exactly covers S, $|\hat{C}| = m/3$ and $\boldsymbol{y} = H\boldsymbol{x}^*$. Thus, $\|\boldsymbol{x}^*\|_0 = m/3$ and

$$F(\boldsymbol{x}^*) = 0 + \lambda \frac{m}{3} = \lambda \frac{m}{3}.$$

Suppose that there exists $\bar{\boldsymbol{x}}$ such that

$$F(\bar{\boldsymbol{x}}) < F(\boldsymbol{x}^*) = \lambda \frac{m}{3}. \tag{5.4}$$

Let us show that this leads to a contradiction.

[3] This transformation was kindly suggested by Andreas M. Tillmann at the SampTA conference in July 2019.

Since $F(\bar{\boldsymbol{x}}) \geq \lambda\|\bar{\boldsymbol{x}}\|_0$, from (5.4) we have $\|\bar{\boldsymbol{x}}\|_0 < m/3$. Therefore, we can rewrite $\|\bar{\boldsymbol{x}}\|_0 = m/3 - q$ for some $q \in \mathbb{N}$, $1 \leq q < m/3$. Note that $H\bar{\boldsymbol{x}}$ has m entries. Since the number of non-zero entries of $H\bar{\boldsymbol{x}}$ identifies with the number of elements s_i recovered by the subcollection corresponding to $\bar{\boldsymbol{x}}$, this number cannot exceed $3\|\bar{\boldsymbol{x}}\|_0 = m - 3q$. As a result, the number of zero entries of $H\bar{\boldsymbol{x}}$ must be between $3q$ and m. Since $\boldsymbol{y}$ is the all-one vector, $\boldsymbol{y} - H\bar{\boldsymbol{x}}$ has at least $3q$ entries valued 1, which implies

$$\|\boldsymbol{y} - H\bar{\boldsymbol{x}}\|_2^2 \geq 3q. \tag{5.5}$$

Hence,

$$F(\bar{\boldsymbol{x}}) \geq 3q + \lambda\left(\frac{m}{3} - q\right) = \lambda\frac{m}{3} + (3-\lambda)q > \lambda\frac{m}{3}, \tag{5.6}$$

which contradicts (5.4). Therefore, $\boldsymbol{x}^*$ is a solution of ($\ell_0 P$).

Construction of a solution of X3C from a solution of ($\ell_0 P$)

Assume that $\boldsymbol{x}^*$ is a solution of ($\ell_0 P$). We will consider four cases as follows.

Case $\|\boldsymbol{x}^*\|_0 > m/3$ We deduce that X3C has no solution. Indeed, assume that $\hat{C}$ is an exact cover for S. Define $\boldsymbol{x} = [x_1, x_2, ..., x_n]^T$ where $x_j = 1$ if $c_j \in \hat{C}$ and $x_i = 0$ otherwise. Then we have

$$F(\boldsymbol{x}) = \lambda\frac{m}{3} < \lambda\|\boldsymbol{x}^*\|_0 \leq F(\boldsymbol{x}^*)$$

which contradicts the fact that $\boldsymbol{x}^*$ is a solution of ($\ell_0 P$).

Case $\|\boldsymbol{x}^*\|_0 < m/3$ We deduce that X3C has no solution. Indeed, assume that $\hat{C}$ is an exact cover for S. Let $\boldsymbol{x} = [x_1, x_2, ..., x_n]^T$ where $x_j = 1$ if $c_j \in \hat{C}$ and $x_i = 0$ otherwise. Then we have $F(\boldsymbol{x}) = \lambda\frac{m}{3}$. Since $\|\boldsymbol{x}^*\|_0 < m/3$, we can write $\|\boldsymbol{x}^*\|_0 = m/3 - q$ for some $q \in \mathbb{N}$ and $1 \leq q < m/3$. Similar to (5.6), we have $F(\boldsymbol{x}^*) > \lambda\frac{m}{3}$. Since $F(\boldsymbol{x}) = \lambda\frac{m}{3}$, we obtain $F(\boldsymbol{x}^*) > F(\boldsymbol{x})$ which contradicts the fact that $\boldsymbol{x}^*$ is a solution of ($\ell_0 P$).

Case where $\|\boldsymbol{x}^*\|_0 = m/3$ **and** $\boldsymbol{y} \neq H\boldsymbol{x}^*$ We deduce that X3C has no solution. Indeed, assume that $\hat{C}$ is an exact cover for S. Define $\boldsymbol{x} = [x_1, x_2, ..., x_n]^T$ where $x_j = 1$ if $c_j \in \hat{C}$ and $x_i = 0$ otherwise. Then we have

$$F(\boldsymbol{x}) = \lambda\frac{m}{3} < \|\boldsymbol{y} - H\boldsymbol{x}^*\|_2^2 + \lambda\|\boldsymbol{x}^*\|_0 = F(\boldsymbol{x}^*)$$

which contradicts the fact that $\boldsymbol{x}^*$ is a solution of ($\ell_0 P$).

Case where $\|\boldsymbol{x}^*\|_0 = m/3$ **and** $\boldsymbol{y} = H\boldsymbol{x}^*$ Let $\hat{C}$ be the collection of triplets c_j such that the j^{th} entry of $\boldsymbol{x}^*$ is non-zero. Obviously, $\hat{C}$ is an exact cover for S so a solution of X3C.

Thus Theorem 5.1 is proved.

It is notable that the proof above is also valid when $F(\boldsymbol{x}) := \|\boldsymbol{y} - H\boldsymbol{x}\|_p^p + \lambda\|\boldsymbol{x}\|_0$ for any $p \geq 1$. Indeed, one only needs to check whether (5.5) still holds when the ℓ_2 norm is replaced by the ℓ_p norm with $p \geq 1$. This is the case since $\boldsymbol{y} - H\bar{\boldsymbol{x}}$ has at least $3q$ entries equal to 1. Therefore, we have the following generalization of Theorem 5.1.

Theorem 5.2. *Problem* $\min_{\boldsymbol{x}} \|\boldsymbol{y} - H\boldsymbol{x}\|_p^p + \lambda\|\boldsymbol{x}\|_0$ *is NP-hard for* $p \geq 1$.

5.5 Hardness of non-negative ℓ_0 minimization problems

The NP-hardness of non-negative ℓ_0 minimization problems is a consequence of NP-hard proofs of $(\ell_0 C_\epsilon)$ [Natarajan, 1995], $(\ell_0 C_K)$ [Davis et al., 1997] and $(\ell_0 P)$ (Theorem 5.1). Indeed, all these proofs consist in a reduction from X3C and the solution that established equivalence is binary. Therefore, the additional non-negativity constraints do not change the validity of these proofs. In other words, one can repeat the same proofs as in [Natarajan, 1995, Davis et al., 1997] and that of Theorem 5.1 for the corresponding non-negative ℓ_0 minimization problems $(\ell_0 C_\epsilon+)$, $(\ell_0 C_K+)$ and $(\ell_0 P+)$. Another way to prove the NP-hardness of non-negative ℓ_0 minimization problems is by a reduction from the corresponding ℓ_0 minimization problems which are known to be NP-hard. In this reduction, the instance of non-negative problems is defined by

$$\tilde{\boldsymbol{y}} = \boldsymbol{y}, \quad \tilde{H} = [H, -H], \quad \tilde{\boldsymbol{x}} = \begin{bmatrix} \boldsymbol{x}^+ \\ \boldsymbol{x}^- \end{bmatrix}$$

where $\boldsymbol{x}^+ = \max\{\boldsymbol{x}, \boldsymbol{0}\}$, $\boldsymbol{x}^- = \max\{-\boldsymbol{x}, \boldsymbol{0}\}$. Naturally, by this construction, one gets $\tilde{\boldsymbol{x}} \geq \boldsymbol{0}$, $\|\tilde{\boldsymbol{x}}\|_0 = \|\boldsymbol{x}\|_0$ and $\tilde{H}\tilde{\boldsymbol{x}} = H\boldsymbol{x}$. The proofs (skipped for brevity) contain three steps similar to that of Theorem 5.1.

Therefore, we can state the following theorem without proof.

Theorem 5.3. *$(\ell_0 C_\epsilon+)$, $(\ell_0 C_K+)$ and $(\ell_0 P+)$ are NP-hard.*

In the same spirit and using the same argument as at the end of Section 5.4 one can directly extend Theorem 5.2 to the non-negative setting.

Theorem 5.4. *Problem $\min_{\boldsymbol{x}\geq\boldsymbol{0}} \|\boldsymbol{y} - H\boldsymbol{x}\|_p^p + \lambda\|\boldsymbol{x}\|_0$ is NP-hard for $p \geq 1$.*

5.6 Conclusion

NP-hardness of penalized ℓ_0 minimization problems cannot be deduced from previous complexity analyses, as stated in [Chen et al., 2014, Huo and Chen, 2010]. Here, we introduced a new proof of NP-hardness of penalized ℓ_0 minimization problems by an adaptation of Natarajan's construction [Natarajan, 1995] . Besides, we showed that the ℓ_0 minimization problems with non-negative constraints are also NP-hard. Therefore it is of interest to propose effective heuristic algorithms such as NNOG algorithms (see Chapter 2).

This work can be extended in several directions. For instance, researchers interested in what makes NP-hard problems even harder might be interested in the strong NP-hardness of the aforementioned optimization problems. As it is widely believed that X3C is strongly NP-complete, one might easily deduce the strong NP-hardness of $(\ell_0 C_\epsilon)$, $(\ell_0 C_K)$ and other problems which are reduced from X3C. However, to the best of our knowledge, X3C is only proved to be NP-complete [Garey and Johnson, 1979, pp. 53, 221] and the strong NP-completeness has not been rigorously shown yet. Therefore, we believe that the question of strong NP-hardness of (non-negative) ℓ_0 minimization problems is not trivial and needs more work in future.

Besides, as (non-negative) ℓ_0 minimization problems are NP-hard, it would be interesting to know if the associated decision problems are in NP (so being NP-complete). Let us consider the decision problem associated with $(\ell_0 C_\epsilon)$: given $\boldsymbol{y} \in \mathbb{Q}^m$, $H \in \mathbb{Q}^{m\times n}$, a positive rational number ϵ and a positive integer K, does there exist $\boldsymbol{x} \in \mathbb{R}^n$ such that $\|\boldsymbol{y} - H\boldsymbol{x}\|_2 \leq \epsilon$ and $\|\boldsymbol{x}\|_0 \leq K$? This decision problem should be in NP since if one can guess a rational solution $\boldsymbol{x}$, it can be verified in polynomial time if $\|\boldsymbol{y} - H\boldsymbol{x}\|_2 \leq \epsilon$ and $\|\boldsymbol{x}\|_0 \leq K$. Similarly, we conjecture that the decision version of other optimization problems mentioned in the chapter are in NP as well.

Another perspective is the approximability of aforementioned NP-hard problems. The hardness of approximating $(\ell_0 C_\epsilon)$ was discussed in [Amaldi and Kann, 1998, Tillmann, 2015]. It was

shown that approximating ($\ell_0 C_\epsilon$) to within a factor of $(1-\alpha)\ln(n)$, $0 < \alpha < 1$ is NP-hard [Tillmann, 2015]. Examining whether similar results can be obtained on other NP-hard problems presented in the chapter would require more involved theoretical analysis, which is left for future work.

Chapter 6

Conclusion and perspectives

Contents

The thesis addresses non-negative sparse approximation by greedy approach. The non-negative extension of orthogonal greedy algorithms is investigated to target the ℓ_0 minimization problem under non-negativity constraints. The thesis contributions cover both algorithmic aspect and theoretical guarantee of non-negative orthogonal greedy (NNOG) algorithms.

6.1 Summary of the contributions

In Chapter 2 we presented a unified framework of NNOG algorithms. We introduced the concepts of descending atom and descent selection rule and integrated the support compression step into the structure of standard NNOG algorithms. The resulted NNOG algorithms have simple structure but more advantageous structural properties in comparison with existing approximate schemes. More importantly, the proposed NNOG algorithms can naturally lend themselves to a recursive (so fast) implementation that existing approximate schemes can hardly do without some loss in the algorithmic behavior. The proposed recursive implementation uses active-set method with warm start initialization to solve the non-negative least-square (NNLS) subproblems. We also elaborated on this implementation and suggested further accelerations using several pruning strategies and vectorized computation.

The proposed implementation is validated through a rich set of simulations in Chapter 3. The proposed NNOG implementations are compared with the corresponding unconstrained versions and with existing approximate schemes, in terms of computing time and reconstruction accuracy. According to our comparison, the proposed implementation significantly reduces the cost of NNOG algorithms and appears to be more advantageous than existing approximate schemes. We also compared NNOG algorithms with other sparse solvers in their non-negative setting such as NNCoSaMP, NNSP, NNHTP and NLARS. It turns out that NNOG algorithms are quite competitive with other non-negative sparse solver in terms of reconstruction accuracy. However, NNOG algorithms have the advantage of a fast implementation that the competitors hardly apply.

While Chapters 2-3 are dedicated to algorithmic development of NNOG algorithms, Chapter 4 discusses the theoretical guarantee issue. We presented for the first time a unified K-step exact support recovery analysis of NNOG algorithms when the mutual coherence of the dictionary is lower than $1/(2K-1)$. Note that the mutual coherence value considered in exact recovery analysis is different from that in the simulations in Chapter 3, so the conclusion of Chapter 4 differs from the empirical observation in Chapter 3. More precisely, exact recovery analysis is derived from easy problem with low mutual coherence so that all NNOG algorithms perform the same. On the contrary, the mutual coherence value considered in simulations is close to 1.

This yields a highly ill-posed problem for which the performances of NNOG algorithms differ. It is usually difficult to carry out exact recovery analyses when the mutual coherence is high but, even when the mutual coherence is low, the exact recovery analysis in the non-negative setting was not done before. Therefore, our analysis is strongly novel and it can be investigated in the context of higher mutual coherence as a perspective.

Finally we showed in Chapter 5 that non-negative ℓ_0 minimization problem is actually NP-hard. Therefore, it is of interest to develop efficient heuristic algorithms. The study of NNOG algorithms and their fast implementation in Chapters 2-4 is hence meaningful.

6.2 Perspectives

6.2.1 Algorithmic aspect

The proposed framework of NNOG algorithms can be extended in several directions. A straightforward generalization can be made to deal with nonnegativity-constrained simultaneous sparse decomposition, which is useful in several applications such as hyperspectral imaging [Wang et al., 2018], dynamic PET [Lin et al., 2014], and diffusion MRI [Kim and Haldar, 2016]. On the other hand, forward-backward greedy algorithms such as BOMP [Herzet and Drémeau, 2010] and SBR [Soussen et al., 2011] could also be extended to the non-negative setting using similar principles and using a recursive implementation. Besides, the combination between NNOG algorithms and ℓ_1 solvers might generate a hybrid algorithm with higher performance than both greedy algorithms and ℓ_1 solvers as suggested in [Wen et al., 2010].

6.2.2 Theoretical guarantee

Our contribution related to K-step exact support recovery analysis of NNOG algorithms when the mutual coherence of the dictionary is lower than $1/(2K-1)$ can have different extensions. For instance, the mutual coherence bound $1/(2K-1)$ might be weakened by taking into account prior information on the sought signal such as the decay of the magnitudes of the nonzero coefficients [Herzet et al., 2016]. With the same idea of deriving exact recovery analysis under weaker conditions, Tropp's exact recovery condition (ERC) [Tropp, 2004] is worth being considered, especially when this condition has a strong connection with mutual coherence value. Finally it is also interesting to check if the usual analysis of OMP and OLS based on RIP condition [Davenport and Wakin, 2010] can be extended to the non-negative setting in a similar way.

Bibliography

[Ait Tilat et al., 2019] Ait Tilat, S., Champagnat, F., and Herzet, C. (2019). A new sparsity based particle image reconstruction approach for particle detection. In *ISPIV*, Munich, Germany. (Cited on pages 4 and 40.)

[Amaldi and Kann, 1998] Amaldi, E. and Kann, V. (1998). On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1):237 – 260. (Cited on page 88.)

[Barbu and Herzet, 2016] Barbu, I. and Herzet, C. (2016). A new approach for volume reconstruction in tomoPIV with the alternating direction method of multipliers. *Measurement Science and Technology*, 27(10):104002. (Cited on pages 3 and 14.)

[Beck and Teboulle, 2009] Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202. (Cited on page 13.)

[Belmerhnia et al., 2015] Belmerhnia, L., Djermoune, E.-H., Carteret, C., and Brie, D. (2015). Simultaneous regularized sparse approximation for wood wastes NIR spectra features selection. In *Proc. CAMSAP*, Cancun, Mexico. (Cited on page 42.)

[Ben-Haim et al., 2010] Ben-Haim, Z., Eldar, Y. C., and Elad, M. (2010). Coherence-based performance guarantees for estimating a sparse vector under random noise. *IEEE Trans. Signal Process.*, 58(10):5030–5043. (Cited on pages 15 and 64.)

[Bendory, 2017] Bendory, T. (2017). Robust recovery of positive stream of pulses. *IEEE Trans. Signal Process.*, 65(8):2114–2122. (Cited on page 73.)

[Benvenuto et al., 2010] Benvenuto, F., Zanella, R., Zanni, L., and Bertero, M. (2010). Nonnegative least-squares image deblurring: improved gradient projection approaches. *Inverse Probl.*, 26(2):1–18. (Cited on page 20.)

[Bioucas-Dias et al., 2012] Bioucas-Dias, J. M., Plaza, A., Dobigeon, N., Parente, M., Du, Q., Gader, P., and Chanussot, J. (2012). Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 5(2):354–379. (Cited on page 6.)

[Björck, 1996] Björck, A. (1996). *Numerical Methods for Least Squares Problems*. Society for Industrial and Applied Mathematics, Philadephia, PA. (Cited on pages 20, 28, 34 and 35.)

[Blumensath and Davies, 2007] Blumensath, T. and Davies, M. E. (2007). On the difference between Orthogonal Matching Pursuit and Orthogonal Least Squares. Technical report, University of Edinburgh. (Cited on pages 9, 12 and 66.)

[Blumensath and Davies, 2008] Blumensath, T. and Davies, M. E. (2008). Iterative thresholding for sparse approximations. *J. Fourier Anal. Appl.*, 14(5):629–654. (Cited on pages 9 and 15.)

[Blumensath and Davies, 2010] Blumensath, T. and Davies, M. E. (2010). Normalized iterative hard thresholding: Guaranteed stability and performance. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):298–309. (Cited on pages 9 and 10.)

[Bonnefoy et al., 2015] Bonnefoy, A., Emiya, V., Ralaivola, L., and Gribonval, R. (2015). Dynamic screening: Accelerating first-order algorithms for the lasso and group-lasso. *IEEE Trans. Signal Process.*, 63(19):5121–5132. (Cited on page 31.)

[Bourguignon et al., 2016] Bourguignon, S., Ninin, J., Carfantan, H., and Mongeau, M. (2016). Exact sparse approximation problems via mixed-integer programming: Formulations and computational performance. *IEEE Trans. Signal Process.*, 64(6):1405–1419. (Cited on pages xi and 63.)

[Boyd et al., 2011] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning *via* the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122. (Cited on pages 13 and 20.)

[Bro and De Jond, 1997] Bro, R. and De Jond, S. (1997). A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11(5):393–401. (Cited on pages xi and 2.)

[Bruckstein et al., 2008] Bruckstein, A. M., Elad, M., and Zibulevsky, M. (2008). On the uniqueness of nonnegative sparse solutions to underdetermined systems of equation. *IEEE Trans. Inf. Theory*, 54(11):4813–4820. (Cited on pages xi, xii, xiii, xiv, 11, 12, 13, 15, 20, 21, 26, 64, 67 and 84.)

[Cai and Wang, 2011] Cai, T. T. and Wang, L. (2011). Orthogonal matching pursuit for sparse signal recovery with noise. *IEEE Trans. Inf. Theory*, 57(7):4680–4688. (Cited on pages 15, 64 and 66.)

[Cai et al., 2010] Cai, T. T., Wang, L., and Xu, G. (2010). Stable recovery of sparse signals using an oracle inequality. *IEEE Trans. Inf. Theory*, 56(7):3516–3522. (Cited on pages xiv, 15 and 70.)

[Candes et al., 2008] Candes, E. J., Wakin, M. B., and Boyd, S. P. (2008). Enhancing sparsity by reweighted l1 minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905. (Cited on page 14.)

[Carcreff, 2014] Carcreff, E. (2014). *Deconvolution adaptative pour le controle non destructif par ultrasons*. PhD thesis, Universite du Maine,. (Cited on page 41.)

[Chen et al., 1989] Chen, S., Billings, S. A., and Luo, W. (1989). Orthogonal least squares methods and their application to non-linear system identification. *Int. J. Control*, 50(5):1873–1896. (Cited on pages 7, 8, 9, 20, 63 and 66.)

[Chen et al., 2001] Chen, S., Donoho, D., and Saunders, M. (2001). Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159. (Cited on page 13.)

[Chen et al., 2014] Chen, X., Ge, D., Wang, Z., and Ye, Y. (2014). Complexity of unconstrained l2-lp minimization. *Mathematical Programming*, (143):371–383. (Cited on pages 83, 84, 85 and 88.)

[Chen et al., 2010] Chen, X., Xu, F., and Ye, Y. (2010). Lower bound theory of nonzero entries in solutions of ℓ_2-ℓ_p minimization. *SIAM J. Sci. Comput.*, 32(5):2832–2852. (Cited on page 28.)

[Combettes and Pesquet, 2011] Combettes, P. L. and Pesquet, J.-C. (2011). *Proximal splitting methods in signal processing*, chapter 10, pages 185–212. Springer-Verlag, New York. (Cited on page 20.)

[Cotter et al., 1999] Cotter, S. F., Adler, J., Rao, B. D., and Kreutz-Delgado, K. (1999). Forward sequential algorithms for best basis selection. *IEE Proc. Vision, Image and Signal Processing*, 146(5):235–244. (Cited on page 7.)

[Dai and Milenkovic, 2009] Dai, W. and Milenkovic, O. (2009). Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory*, 55(5):2230–2249. (Cited on pages 9, 10, 15 and 54.)

[Davenport and Wakin, 2010] Davenport, M. A. and Wakin, M. B. (2010). Analysis of orthogonal matching pursuit using the restricted isometry property. *IEEE Trans. Inf. Theory*, 56(9):4395–4401. (Cited on pages 15, 64 and 92.)

[Davis et al., 1997] Davis, G., Mallat, S., and Avellaneda, M. (1997). Adaptive greedy approximations. *Constructive Approximation*, 13(1):57–98. (Cited on pages 7, 83, 84, 86 and 88.)

[Donoho and Elad, 2003] Donoho, D. L. and Elad, M. (2003). Optimally sparse representation in general (non-orthogonal) dictionaries via ℓ^1 minimization. *Proc. Natl. Acad. Sci.* USA, 100(5):2197–2202. (Cited on page 66.)

[Donoho et al., 2006] Donoho, D. L., Elad, M., and Temlyakov, V. N. (2006). Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Trans. Inf. Theory*, 52(1):6–18. (Cited on pages 64 and 67.)

[Donoho and Tsaig, 2008] Donoho, D. L. and Tsaig, Y. (2008). Fast solution of ℓ_1-norm minimization problems when the solution may be sparse. *IEEE Trans. Inf. Theory*, 54(11):4789–4812. (Cited on pages 13, 14, 63, 68, 69, 72 and 73.)

[Efron et al., 2004] Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *Ann. Statist.*, 32(2):407–499. (Cited on pages 13, 14, 58 and 68.)

[Elad, 2010] Elad, M. (2010). *Sparse and redundant representations: From theory to applications in signal and image processing*. Springer, New York. (Cited on pages 2 and 83.)

[Elsinga et al., 2006] Elsinga, G. E., Scarano, F., Wieneke, B., and van Oudheusden, B. W. (2006). Tomographic particle image velocimetry. *Experiments in Fluids*, 41(6):933–947. (Cited on pages 3 and 4.)

[Elvira et al., 2019] Elvira, C., Gribonval, R., Soussen, C., and Herzet, C. (2019). OMP and continuous dictionaries: is k-step recovery possible? In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5546–5550, Brighton, United Kingdom. IEEE, IEEE. (Cited on pages 4 and 40.)

[Esser et al., 2013] Esser, E., Lou, Y., and Xin, J. (2013). A method for finding structured sparse solutions to nonnegative least squares problems with applications. *SIAM J. Imaging Sci.*, 6(4):2010–2046. (Cited on page 2.)

[Fan and Li, 2001] Fan, J. and Li, R. (2001). Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360. (Cited on page 14.)

[Foucart, 2011] Foucart, S. (2011). Hard thresholding pursuit: An algorithm for compressive sensing. *SIAM Journal on Numerical Analysis*, 49(6):2543–2563. (Cited on pages 9, 10, 15 and 54.)

[Foucart, 2013] Foucart, S. (2013). Stability and robustness of weak orthogonal matching pursuits. In Bilyk, D., De Carli, L., Petukhov, A., Stokolos, A. M., and Wick, B. D., editors, *Recent advances in harmonic analysis and applications*, volume 25, pages 395–405. Springer Proceedings in Mathematics & Statistics. (Cited on page 7.)

[Foucart and Koslicki, 2014] Foucart, S. and Koslicki, D. (2014). Sparse recovery by means of nonnegative least squares. *IEEE Signal Process. Lett.*, 21(4):498–502. (Cited on pages 28 and 31.)

[Foucart and Rauhut, 2013] Foucart, S. and Rauhut, H. (2013). *A mathematical introduction to compressive sensing*. Applied and Numerical Harmonic Analysis. Birkhäuser, Basel. (Cited on pages 15 and 74.)

[Frank and Friedman, 1993] Frank, I. and Friedman, J. (1993). A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–135. (Cited on page 14.)

[Friedman et al., 2007] Friedman, J., Hastie, T., and Tibshirani, R. (2007). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441. (Cited on page 13.)

[Fuchs, 2004] Fuchs, J.-J. (2004). On sparse representations in arbitrary redundant bases. *IEEE Trans. Inf. Theory*, 50(6):1341–1344. (Cited on page 67.)

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York. (Cited on pages 84, 85 and 88.)

[Gasso et al., 2009] Gasso, G., Rakotomamonjy, A., and Canu, S. (2009). Recovering sparse signals with a certain family of nonconvex penalties and DC programming. *IEEE Trans. Signal Process.*, 57(12):4686–4698. (Cited on pages xi and 20.)

[Golub and Van Loan, 1996] Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations (3rd Ed.)*. Johns Hopkins University Press, Baltimore, MD, USA. (Cited on page 34.)

[Herzet and Drémeau, 2010] Herzet, C. and Drémeau, A. (2010). Bayesian pursuit algorithms. In *Proc. Eur. Sig. Proc. Conf.*, pages 1474–1478, Aalborg, Denmark. (Cited on pages 9, 28, 33 and 92.)

[Herzet et al., 2016] Herzet, C., Drémeau, A., and Soussen, C. (2016). Relaxed recovery conditions for OMP/OLS by exploiting both coherence and decay. *IEEE Trans. Inf. Theory*, 62(1):459–470. (Cited on pages 15, 64, 66, 74 and 92.)

[Herzet et al., 2013] Herzet, C., Soussen, C., Idier, J., and Gribonval, R. (2013). Exact recovery conditions for sparse representations with partial support information. *IEEE Trans. Inf. Theory*, 59(11):7509–7524. (Cited on pages xiv, 15, 64, 66 and 70.)

[Högbom, 1974] Högbom, J. A. (1974). Aperture synthesis with a non-regular distribution of interferometer baselines. *Astron. Astrophys. Suppl.*, 15:417–426. (Cited on pages xi and 2.)

[Hoyer, 2004] Hoyer, P. O. (2004). Non-negative matrix factorization with sparseness constraints. *J. Mach. Learning Res.*, 5:1457–1469. (Cited on page 3.)

[Huang et al., 2008] Huang, J., Horowitz, J. L., and Ma, S. (2008). Asymptotic properties of bridge estimators in sparse high-dimensional regression models. *Ann. Statist.*, 36(2):587–613. (Cited on page 14.)

[Huo and Chen, 2010] Huo, X. and Chen, J. (2010). Complexity of penalized likelihood estimation. *J. Statist. Comput. Simul.*, (80:7):747–759. (Cited on pages 83, 84, 85 and 88.)

[Huo and Ni, 2007] Huo, X. and Ni, X. (2007). When do stepwise algorithms meet subset selection criteria? *Annals Statist.*, 35(2):870–887. (Cited on pages 85 and 86.)

[Iordache et al., 2011] Iordache, M.-D., Bioucas-Dias, J. M., and Plaza, A. (2011). Sparse unmixing of hyperspectral data. *IEEE Trans. Geosci. Remote Sensing*, 49(6):2014–2039. (Cited on pages xi, 2 and 6.)

[Keshava and Mustard, 2002] Keshava, N. and Mustard, J. F. (2002). Spectral unmixing. *IEEE Signal Processing Magazine*, 19(1):44–57. (Cited on page 6.)

[Kim and Haldar, 2016] Kim, D. and Haldar, J. P. (2016). Greedy algorithms for nonnegativity-constrained simultaneous sparse recovery. *Signal Process.*, 125:274–289. (Cited on pages 11, 12, 13, 15, 26, 32, 54, 55, 61, 64 and 92.)

[Lawson and Hanson, 1974] Lawson, C. L. and Hanson, R. J. (1974). *Solving least squares problems*, pages 149–199. Society for Industrial and Applied Mathematics. (Cited on pages xiii, 16, 20, 21, 22, 28 and 29.)

[Le Thi et al., 2013] Le Thi, H. A., Nguyen Thi, B. T., and Le, H. M. (2013). Sparse signal recovery by difference of convex functions algorithms. In Selamat, A., Nguyen, N. T., and Haron, H., editors, *Intelligent Information and Database Systems*, volume 7803 of *Lect. Notes Comput. Sci.*, pages 387–397, Berlin. Springer Verlag. (Cited on pages 14 and 20.)

[Leeuwen, 1990] Leeuwen, J. v. (1990). *Handbook of theoretical computer science (vol. A): algorithms and complexity*. Elsevier Science Publishers. (Cited on pages 84 and 86.)

[Leichner et al., 1993] Leichner, S., Dantzig, G., and Davis, J. (1993). A strictly improving linear programming Phase I algorithm. *Ann. Oper. Res.*, 47:409–430. (Cited on page 25.)

[Li et al., 2015] Li, B., Shen, Y., Wu, Z., and Li, J. (2015). Sufficient conditions for generalized orthogonal matching pursuit in noisy case. *Signal Process.*, 108:111–123. (Cited on pages 15 and 64.)

[Lin et al., 2014] Lin, Y., Haldar, J. P., Li, Q., Conti, P., and Leahy, R. M. (2014). Sparsity constrained mixture modeling for the estimation of kinetic parameters in dynamic PET. *IEEE Trans. Med. Imag.*, 33(1):173–185. (Cited on pages 32 and 92.)

[Mallat and Zhang, 1993] Mallat, S. G. and Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415. (Cited on page 7.)

[Miller, 2002] Miller, A. J. (2002). *Subset selection in regression*. Chapman and Hall, London, UK, 2nd edition. (Cited on pages 9, 20, 28 and 68.)

[Mortada, 2018] Mortada, H. (2018). *Separation of parameterized and delayed sources : application to spectroscopic and multispectral data*. PhD thesis, Universite de Strasbourg. (Cited on page 6.)

[Morup et al., 2008] Morup, M., Madsen, K. H., and Hansen, L. K. (2008). Approximate l0 constrained non-negative matrix and tensor factorization. In *2008 IEEE International Symposium on Circuits and Systems*, pages 1328–1331. (Cited on pages 14, 25 and 58.)

[Natarajan, 1995] Natarajan, B. K. (1995). Sparse approximate solutions to linear systems. *SIAM J. Comput.*, 24(2):227–234. (Cited on pages xi, 7, 17, 63, 83, 84, 86 and 88.)

[Needell and Tropp, 2009] Needell, D. and Tropp, J. A. (2009). CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Appl. Comp. Harmonic Anal.*, 26(3):301–321. (Cited on pages 9, 10, 15, 33 and 54.)

[Nguyen et al., 2017] Nguyen, T., Soussen, C., Idier, J., and Djermoune, E.-H. (2017). An optimized version of non-negative omp. In *GRETSI*. (Cited on pages xii, xiii and 12.)

[Nguyen et al., 2019a] Nguyen, T. T., Idier, J., Soussen, C., and Djermoune, E.-H. (2019a). Non-negative orthogonal greedy algorithms. Technical report, Université de Lorraine, LS2N, CentraleSupélec, Nancy, France. (Cited on pages xi, xii and 19.)

[Nguyen et al., 2019b] Nguyen, T. T., Soussen, C., Idier, J., and Djermoune, E.-H. (2019b). Non-negative greedy algorithms: Matlab implementation. CodeOcean: doi.org/10.24433/CO.2445681.v1. (Cited on pages xv and 32.)

[Nguyen et al., 2019c] Nguyen, T. T., Soussen, C., Idier, J., and Djermoune, E.-H. (2019c). NP-hardness of $\ell 0$ minimization problems: revision and extension to the non-negative setting. In *13th International Conference on Sampling Theory and Applications, SampTA 2019*, Bordeaux, France. (Cited on page 83.)

[Nguyen et al., 2019d] Nguyen, T. T., Soussen, C., Idier, J., and Djermoune, E.-H. (2019d). Sign preservation analysis of orthogonal greedy algorithms. Tech. rep., Univ. Lorraine, CentraleSupélec, LS2N, Nancy, France. (Cited on pages xii, xiv and 63.)

[Nikolova, 2013] Nikolova, M. (2013). Description of the minimizers of least squares regularized with ℓ_0 norm. Uniqueness of the global minimizer. *SIAM J. Imaging Sci.*, 6(2):904–937. (Cited on page 86.)

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). *Numerical optimization*. Springer texts in Operations Research and Financial Engineering. Springer Verlag, New York, 2nd edition. (Cited on pages 14, 23, 28 and 29.)

[Pati et al., 1993] Pati, Y. C., Rezaiifar, R., and Krishnaprasad, P. S. (1993). Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proc. 27th Asilomar Conf. on Signals, Systems and Computers*, volume 1, pages 40–44. (Cited on pages 7, 8, 20 and 63.)

[Peharz and Pernkopf, 2012] Peharz, R. and Pernkopf, F. (2012). Sparse nonnegative matrix factorization with l0 constraints. *Neurocomputing*, 80:38–46. (Cited on pages 20, 28, 31, 32, 70 and 84.)

[Petra and Schnörr, 2014] Petra, S. and Schnörr, C. (2014). Average case recovery analysis of tomographic compressive sensing. *Linear Alg. Appl.*, 441:168–198. (Cited on pages xi, 2, 3 and 74.)

[Plumbley, 2007] Plumbley, M. D. (2007). On polar polytopes and the recovery of sparse representations. *IEEE Trans. Inf. Theory*, 53(9):3188–3195. (Cited on page 68.)

[Ramamurthy et al., 2014] Ramamurthy, K. N., Thiagarajan, J. J., and Spanias, A. (2014). Recovering non-negative and combined sparse representations. *Digital Signal Process.*, 26(1):21–35. (Cited on pages 11 and 26.)

[Rapin et al., 2013] Rapin, J., Bobin, J., Larue, A., and Starck, J.-L. (2013). Sparse and non-negative BSS for noisy data. *IEEE Trans. Signal Process.*, 61(22):5620–5632. (Cited on page 3.)

[Rebollo-Neira and Lowe, 2002] Rebollo-Neira, L. and Lowe, D. (2002). Optimized orthogonal matching pursuit approach. *IEEE Signal Process. Lett.*, 9(4):137–140. (Cited on pages 7, 9 and 66.)

[Saab et al., 2008] Saab, R., Chartrand, R., and Yilmaz, O. (2008). Stable sparse approximations via nonconvex optimization. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3885–3888. (Cited on page 14.)

[Satpathi et al., 2013] Satpathi, S., Lochan Das, R., and Chakraborty, M. (2013). Improving the bound on the RIP constant in generalized orthogonal matching pursuit. *IEEE Signal Process. Lett.*, 20(11):1074–1077. (Cited on pages 15 and 64.)

[Slawski and Hein, 2011] Slawski, M. and Hein, M. (2011). Sparse recovery by thresholded non-negative least squares. *Adv. Neural Inf. Process. Syst.*, 24:1926–1934. (Cited on page 20.)

[Slawski and Hein, 2013] Slawski, M. and Hein, M. (2013). Non-negative least squares for high-dimensional linear models: Consistency and sparse recovery without regularization. *Electron. J. Stat.*, 7:3004–3056. (Cited on page 20.)

[Slawski et al., 2012] Slawski, M., Hussong, R., Tholey, A., Jakoby, T., Gregorius, B., Hildebrandt, A., and Hein, M. (2012). Isotope pattern deconvolution for peptide mass spectrometry by non-negative least squares/least absolute deviation template matching. *BMC Bioinformatics*, 13(291):1–18. (Cited on pages 2 and 3.)

[Soubies et al., 2015] Soubies, E., Blanc-Féraud, L., and Aubert, G. (2015). A continuous exact l0 penalty (cel0) for least squares regularized problem. *SIAM J. Imaging Sci*, 8(3):1607–1639. (Cited on page xi.)

[Soussen et al., 2013] Soussen, C., Gribonval, R., Idier, J., and Herzet, C. (2013). Joint k-step analysis of orthogonal matching pursuit and orthogonal least squares. *IEEE Trans. Inf. Theory*, 59(5):3158–3174. (Cited on pages 15, 64, 74 and 75.)

[Soussen et al., 2011] Soussen, C., Idier, J., Brie, D., and Duan, J. (2011). From Bernoulli-Gaussian deconvolution to sparse signal restoration. *IEEE Trans. Signal Process.*, 59(10):4572–4584. (Cited on pages 4, 9, 28, 33 and 92.)

[Soussen et al., 2015] Soussen, C., Idier, J., Duan, J., and Brie, D. (2015). Homotopy based algorithms forℓ_0-regularized least-squares. *IEEE Transactions on Signal Processing*, 63(13):3301–3316. (Cited on page 9.)

[Sturm and Christensen, 2012] Sturm, B. L. and Christensen, M. G. (2012). Comparison of orthogonal matching pursuit implementations. In *Proc. Eur. Sig. Proc. Conf.*, pages 220–224, Bucharest, Romania. (Cited on pages xi, 9, 20 and 28.)

[Thoai, 1988] Thoai, N. (1988). A modified version of tuy's method for solving d.c. programing problem. *Optimization*, 19(5):665–674. (Cited on page 14.)

[Tibshirani, 1996] Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *J. R. Statist. Soc. B*, 58(1):267–288. (Cited on pages 13 and 63.)

[Tillmann, 2015] Tillmann, A. M. (2015). On the computational intractability of exact and approximate dictionary learning. *IEEE Signal Processing Letters*, 22(1):45–49. (Cited on pages 15, 88 and 89.)

[Tropp, 2004] Tropp, J. A. (2004). Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Inf. Theory*, 50(10):2231–2242. (Cited on pages xii, xiv, 15, 64, 66, 67, 74 and 92.)

[Turlach et al., 2005] Turlach, B. A., Venables, W. N., and Wright, S. J. (2005). Simultaneous variable selection. *Technometrics*, 47(3):349–363. (Cited on page 42.)

[van den Berg and Friedlander, 2009] van den Berg, E. and Friedlander, M. (2009). Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912. (Cited on page 13.)

[Virtanen et al., 2013] Virtanen, T., Gemmeke, J. F., and Raj, B. (2013). Active-set Newton algorithm for overcomplete non-negative representations of audio. *IEEE Trans. Audio, Speech, Language Process.*, 21(11):2277–2289. (Cited on pages xi and 2.)

[Wagner et al., 2015] Wagner, K., Schnabel, T., Barbu, M.-C., and Petutschnigg, A. (2015). Analysis of selected properties of fibreboard panels manufactured from wood and leather using the near infrared spectroscopy. *Int J. Spectrosc.*, 2015:691796. (Cited on page 42.)

[Wang and Li, 2017] Wang, J. and Li, P. (2017). Recovery of sparse signals using multiple orthogonal least squares. *IEEE Trans. Signal Process.*, 65(8):2049–2062. (Cited on pages 15 and 64.)

[Wang et al., 2018] Wang, Z., Zhu, R., Fukui, K., and Xue, J. (2018). Cone-based joint sparse modelling for hyperspectral image classification. *Signal Processing*, 144:417–429. (Cited on pages 6, 11, 26, 32, 84 and 92.)

[Wen et al., 2017a] Wen, J., Wang, J., and Zhang, Q. (2017a). Nearly optimal bounds for orthogonal least squares. *IEEE Trans. Signal Process.*, 65(20):5347–5356. (Cited on pages 15 and 64.)

[Wen et al., 2017b] Wen, J., Zhou, Z., Wang, J., Tang, X., and Mo, Q. (2017b). A sharp condition for exact support recovery with orthogonal matching pursuit. *IEEE Trans. Signal Process.*, 65(6):1370–1382. (Cited on pages 15 and 64.)

[Wen et al., 2010] Wen, Z., Yin, W., Goldfarb, D., and Zhang, Y. (2010). A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization, and continuation. *SIAM Journal on Scientific Computing*, 32(4):1832–1857. (Cited on page 92.)

[Wright, 1992] Wright, M. H. (1992). Interior methods for constrained optimization. *Acta Numerica*, 1:341–407. (Cited on page 20.)

[Yaghoobi and Davies, 2015] Yaghoobi, M. and Davies, M. E. (2015). Fast non-negative orthogonal least squares. In *Proc. Eur. Sig. Proc. Conf.*, pages 479–483, Nice, France. (Cited on pages xiii, 11, 12, 20, 21, 26, 32, 54, 61, 64 and 65.)

[Yaghoobi et al., 2015] Yaghoobi, M., Wu, D., and Davies, M. E. (2015). Fast non-negative orthogonal matching pursuit. *IEEE Signal Process. Lett.*, 22(9):1229–1233. (Cited on pages xi, xii, 11, 20, 21, 26, 32, 54 and 84.)

[Zhang, 2011] Zhang, T. (2011). Sparse recovery with orthogonal matching pursuit under RIP. *IEEE Trans. Inf. Theory*, 57(9):6215–6221. (Cited on page 28.)

[Zou and Li, 2008] Zou, H. and Li, R. (2008). One-step sparse estimates in nonconcave penalized likelihood models. *The Annals of Statistics*, 36(4):1509–1533. (Cited on page 14.)

Résumé

De nombreux domaines applicatifs conduisent à résoudre des problèmes inverses où le signal ou l'image à reconstruire est à la fois parcimonieux et positif. Si la structure de certains algorithmes de reconstruction parcimonieuse s'adapte directement pour traiter les contraintes de positivité, il n'en va pas de même des algorithmes gloutons orthogonaux comme OMP et OLS. Leur extension positive pose des problèmes d'implémentation car les sous-problèmes de moindres carrés positifs à résoudre ne possèdent pas de solution explicite. Dans la littérature, les algorithmes gloutons positifs (NNOG, pour *non-negative orthogonal greedy algorithms*) sont souvent considérés comme lents, et les implémentations récemment proposées exploitent des schémas récursifs approchés pour compenser cette lenteur.

Dans ce manuscrit, les algorithmes NNOG sont vus comme des heuristiques pour résoudre le problème de minimisation ℓ_0 sous contrainte de positivité. La première contribution est de montrer que ce problème est NP-difficile. Deuxièmement, nous dressons un panorama unifié des algorithmes NNOG et proposons une implémentation exacte et rapide basée sur la méthode des contraintes actives avec démarrage à chaud pour résoudre les sous-problèmes de moindres carrés positifs. Cette implémentation réduit considérablement le coût des algorithmes NNOG et s'avère avantageuse par rapport aux schémas approximatifs existants. La troisième contribution consiste en une analyse de reconstruction exacte en K étapes du support d'une représentation K-parcimonieuse par les algorithmes NNOG lorsque la cohérence mutuelle du dictionnaire est inférieure à $1/(2K-1)$. C'est la première analyse de ce type.

Mots clefs : algorithmes gloutons orthogonaux, reconstruction parcimonieux, contrainte de positivité, moindres carrés positif, l'algorithme des contraintes actives, reconstruction de support exact, cohérence mutuelle, NP-difficile, problème minimisé ℓ_0.

Abstract

Non-negative sparse approximation arises in many applications fields such as biomedical engineering, fluid mechanics, astrophysics, and remote sensing. Some classical sparse algorithms can be straightforwardly adapted to deal with non-negativity constraints. On the contrary, the non-negative extension of orthogonal greedy algorithms is a challenging issue since the unconstrained least square subproblems are replaced by non-negative least squares subproblems which do not have closed-form solutions. In the literature, non-negative orthogonal greedy (NNOG) algorithms are often considered to be slow. Moreover, some recent works exploit approximate schemes to derive efficient recursive implementations.

In this thesis, NNOG algorithms are introduced as heuristic solvers dedicated to ℓ_0 minimization under non-negativity constraints. It is first shown that the latter ℓ_0 minimization problem is NP-hard. The second contribution is to propose a unified framework on NNOG algorithms together with an exact and fast implementation, where the non-negative least-square subproblems are solved using the active-set algorithm with warm start initialization. The proposed implementation significantly reduces the cost of NNOG algorithms and appears to be more advantageous than existing approximate schemes. The third contribution consists of a unified K-step exact support recovery analysis of NNOG algorithms when the mutual coherence of the dictionary is lower than $1/(2K-1)$. This is the first analysis of this kind.

Keywords : orthogonal greedy algorithms, sparse reconstruction, non-negativity, non-negative least squares, active-set algorithms, exact recovery condition, mutual coherence, NP-hardness, ℓ_0 minimization.