Greedy sparse recovery algorithms: from weighted generalizations to deep unrolling

Sina Mohammad-Taheri

A Thesis
in
The Department
of
Mathematics and Statistics

Presented in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy (Mathematics) at
Concordia University
Montréal, Québec, Canada

August 2025

CONCORDIA UNIVERSITY
School of Graduate Studies

This is to certify that the thesis prepared

By:	**Dr. Sina Mohammad-Taheri**
Entitled:	**Greedy sparse recovery algorithms: from weighted generalizations to deep unrolling**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Mathematics)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.
Signed by the Final Examining Committee:

Dr. Behrooz Yousefzadeh	Chair
Dr. Abbas Khalili	External Examiner
Dr. Habib Benali	Arms-Length Examiner
Dr. Jason Bramburger	Examiner
Dr. Junxi Zhang	Examiner
Dr. Simone Brugiapaglia	Supervisor

Approved by
Lea Popovic, Graduate Program Director
Department of Mathematics and Statistics

2025

Pascale Sicotte, Dean
Faculty of Arts and Sciences

Abstract

Greedy sparse recovery algorithms: from weighted generalizations to deep unrolling

Sina Mohammad-Taheri
Concordia University, 2025

Sparse recovery is a clear manifestation of Occam's razor in the mathematics of data thanks to its ability to favor the simplest explanation. A prominent example is the reconstruction of a sparse vector (i.e., one with mostly zero or negligible coefficients) from linear measurements, possibly corrupted by noise. This problem arises in numerous applications across various domains, including medical imaging and high-dimensional function approximation. Greedy sparse recovery algorithms approach this problem by recursively solving local optimization problems and have proven to be efficient alternatives to convex methods.

In this dissertation, we first develop weighted generalizations of Orthogonal Matching Pursuit (OMP)–one of the most popular greedy sparse recovery algorithms–based on two distinct weighting strategies aimed at incorporating a priori knowledge about the signal structure. These generalizations feature explicit and theoretically justified greedy index selection rules. The first strategy establishes a novel connection between greedy sparse recovery and convex relaxation methods, particularly the LASSO family, resulting in new OMP-based algorithms suited to a wide range of loss functions. Numerical results show that these greedy variants inherit key traits from their ancestor convex decoders. For the second strategy, we provide optimal recovery guarantees for rescaling-based weighted OMP under the weighted Restricted Isometry Property (wRIP), extending the classical RIP-based analysis to the weighted setting.

The second part of the thesis addresses the non-differentiability of greedy algorithms caused by the (arg)sorting operator by employing "softsorting", enabling differentiable, neural-network-compatible versions of these algorithms. We provide rigorous theoretical analysis and numerically demonstrate that these "soft" algorithms reliably approximate their original counterparts. We then link our weighted greedy solvers to neural architectures, by embedding their iterations into neural networks using the "algorithm unrolling" paradigm, treating weights as meaningful trainable parameters. This approach not only advances the development of data-driven methods for sparse recovery, but also marks a significant step toward building safer and more trustworthy neural networks. Finally, we extend this framework to deterministic compressed sensing and to learning additional parameters beyond weights.

Acknowledgments

Above all, I am grateful to the One, the omnipresent, who has been there when no one else has, and in whose remembrance hearts do find peace.

First off, I would like to express my deepest gratitude to my supervisor, Dr. Simone Brugiapaglia, for his unwavering support, trust and knowledge, without which this journey would not have been possible.

In addition to his continual guidance, Simone graciously welcomed me into his academic family, many members of whom I have been fortunate to learn from through their profound insights and thoughtful feedback. I am especially thankful to Dr. Matthew Colbrook (University of Cambridge), Dr. Ben Adcock (Simon Fraser University) and Dr. Mark Iwen (Michigan State University). Matthew generously accepted the role of my co-mentor and made my academic visit to the University of Cambridge in Winter 2024 possible. The publications [87, 88] presented in Chapter 4 and much of the work Chapter 5 were made possible through collaboration with him. I have also greatly enjoyed discussing with Ben and Mark on several occasions, and their insights resonate throughout this thesis. In particular, Chapter 4 owes its development to insightful discussions with Ben, and Section 5.3 reflects some of the early steps in collaboration with Mark.

I also thank Dr. Jason Bramburger, Dr. Abbas Khalili, Dr. Habib Benali and Dr. Junxi Zhang for generously dedicating their time to read this thesis and offering invaluable comments and discussions. I am also thankful to Dr. Behrooz Yousefzadeh for his warm presence in my committee.

My sincerest gratitude goes to my family, especially my parents and siblings, whose unconditional love, kindness and unwavering support–even from afar–have sustained me throughout. Without them, none of this would have been possible. I also extend heartfelt thanks to my friends, those who made Concordia and Montréal a second home, and those who may have been far in distance, but always close in heart. I hope I can return their love as deserved.

Dedication

Much of the time I spent developing and writing this thesis, Middle East have been in the turmoil of encroachment and aggression. I dedicate this work to my people in the Middle East, whose peace and prosperity have always been—and will remain—my deepest wish and purpose.

Contents

List of Figures

Chapter 1

Introduction

“Entia non sunt multiplicanda praeter necessitatem”
“Entities must not be multiplied beyond necessity”
— William of Ockham

The *law of parsimony* or *Occam’s razor* is an intuitive principle in problem solving attributed to William of Ockham, a Franciscan friar who lived between the 13th and 14th century in a town in Surrey, UK. It states that, when faced with several hypotheses with similar explanatory power one should choose the one that involves fewer assumptions, thus “shaving away” unnecessarily complex ones. In modeling and mathematical sciences this translates into the desire to have the most parsimonious–“sparse”–model, involving the smallest number of parameters while preserving its fidelity to reality. In addition, prior knowledge about the world’s structure biases modeling towards specific assumptions, corresponding to giving larger “weights” to certain parameters. Hypotheses rely on observations (or data) from the real-world and these observations, through the learning process, lead to knowledge. The present thesis revolves around these three axial principles of mathematical modeling, i.e., sparsity, structure and learning. These possess exciting implications in modern mathematics of data with countless applications in science and engineering, including, e.g., surrogate modeling [11], uncertainty quantification [107] and medical imaging [3].

1.1 Background

We begin by presenting some substantial materials for our study. Our starting point is the sparse recovery problem and methods to solve it, wherein we also discuss structured sparsity. We then connect sparse recovery to deep neural networks and conclude this section by presenting some mathematical tools needed in our analysis.

1.1.1 Sparse recovery problem and compressed sensing

Sparsity is the clear manifestation of Occam's razor principle in mathematics of data: a vector (signal) $x \in \mathbb{C}^N$ is s-sparse if it possesses at most s nonzero entries, i.e.,

$$\|x\|_0 := \mathrm{card}(\mathrm{supp}(x)) \leq s.$$

Sparse recovery is the art of reconstructing an s-sparse signal $x \in \mathbb{C}^N$, observed through a (linear) mapping $A : \mathbb{C}^N \to \mathbb{C}^m$ represented by the matrix $A \in \mathbb{C}^{m \times N}$, potentially corrupted by error or noise $e \in \mathbb{C}^m$, from an observation (measurement) vector $y \in \mathbb{C}^m$ such that

$$y = Ax + e. \tag{1}$$

In the case of unknown error or insufficient data ($m < N$ or even $m \ll N$), which are the more interesting scenarios from the practical viewpoint, Equation (1) is a severely ill-posed problem and this is where Occam's razor comes into play. By imposing sparsity one might find an approximate solution to Equation (1) via the optimization problem

$$\hat{x} \in \operatorname*{argmin}_{z \in \mathbb{C}^N} \|y - Az\|^2 \quad \text{s.t.} \quad \|z\|_0 \leq s, \tag{2}$$

or

$$\hat{x} \in \operatorname*{argmin}_{z \in \mathbb{C}^N} \|z\|_0 \quad \text{s.t.} \quad \|y - Az\|^2 \leq \eta, \tag{3}$$

with $\eta > 0$ depending on the noise magnitude and $\|\cdot\|$ denoting the standard Euclidean norm.

If the unknown signal x is sparse, in many applications (especially in high dimensional ones such as image processing), it seems reasonable to design sensing devices that enable compression at the measurement stage rather than measuring the signal in its full length and then throwing out unimportant elements after its reconstruction. This is the central idea behind *compressed* (or *compressive*) *sensing* [26, 45], which is one of the main areas revolving around sparse recovery. At this point, three important questions naturally arise:

(Q1) *How can we solve sparse recovery problems of Equation* (2) *and Equation* (3) *in practice?*

(Q2) *Which types of the measurement matrix A allow for recovery with theoretical guarantees?*

(Q3) *How many measurements are enough for successful recovery?*

The first question is one of the pivotal themes of this thesis and will be introduced in the subsequent subsection and studied in more detail throughout. We now provide brief answers for the other two as they are not directly the subject of our study in this thesis, but they are nonetheless important to better contextualize our contributions.

As far as it concerns the number of measurements (Q3) one might argue that to guarantee a unique solution $m \geq 2s$ should be enough for signal reconstruction and in fact there are certain constructions and algorithms supporting this [97, 116]. However, robustness to noise in the measurements and stability to approximate sparsity in the signal demand for more measurements. Therefore, regarding the second and third question, the following major settings have been proposed [55]:

(i) Fully random matrices such as the family of *subgaussian matrices*, e.g., with mean-zero identically and independently distributed (iid) Gaussian or Bernoulli entries. This group of matrices are the most well-studied and theoretically understood, resulting in optimal recovery guarantees with a number of measurements scaling like $m \gtrsim s\log(N/s)$, where $\gtrsim$ denotes the presence of an absolute constant in the right hand side. However, they have limited practicality. In fact, first, (almost) fully random constructions are not easy to realize in physical sensing devices. One exception is the single-pixel camera [48], although in terms of practical implementation is still at its infancy. Second, due to lack of structure they are highly inefficient in terms of memory storage. Yet, thanks to their simplicity, they have become a gold standard in compressive sensing simulations as a first validation benchmark and thus we will extensively use them in our numerical experiments.

(ii) Matrices related to sampling in *Bounded Orthonormal Systems* (BOS's), i.e., a space endowed with orthonormal basis $\{\Phi_j(\mathcal{D})\}_{j=1}^{\infty}$, for $\mathcal{D} \subset \mathbb{R}^d$, with basis elements uniformly bounded, i.e., for which there exists a $K \geq 0$ such that $\|\Phi_j\|_\infty \leq K$ (see, e.g., [11, 55]). BOS's provide a good balance between theory and practice as, despite their randomness, they also allow certain types of structure, with e.g., $m \gtrsim s\log^2(s)\log(N)$ sample complexity (see, e.g., [24]). A prominent example in the discrete setup is the *partial Fourier matrix* equipped with the efficient Fast Fourier Transform (FFT) (see, Section 4.5). Experiments on function approximation in Chapter 2 and signal recovery using algorithm unrolling in Chapter 4 are of such type.

(iii) *Deterministic matrices* [44, 68]: this compressive sensing setup is inherently fast as it can leverage deterministic, efficient structure in matrix operations and highly memory friendly. However, deterministic constructions lack enough theoretical support in comparison with setups (i) and (ii) and there are still many open research problems in the area. One major drawback is the quadratic bottleneck, i.e., the fact that the required number of measurements scales quadratically with sparsity, e.g., $m \gtrsim s^2\log^4(N)$. We dive into this category only toward the end of Chapter 5.

1.1.2 Sparse recovery solvers

Equation (2) is a combinatorial problem (NP-hard) and thus cannot be solved directly (assuming the common belief that $P \neq NP$) [13]. As a consequence, methods to solve the sparse recovery problem (2) fall into the following two main categories:

(a) **Convex relaxation techniques** replace the ℓ^0-norm in favor of the more manageable convex ℓ^1-norm, defined for $z \in \mathbb{C}^N$ as

$$\|z\|_1 := \sum_{j=1}^{N} |z_j|.$$

This allows one to leverage the existing plethora of algorithms from the realm of convex optimization. In particular, the following optimization programs will be of interest:

- *Quadratically Constrained Basis Pursuit* (QCBP) [30], i.e.,

$$\hat{x} \in \operatorname*{argmin}_{z \in \mathbb{C}^N} \|z\|_1 \quad \text{s.t.} \quad \|y - Az\|^2 \leq \eta, \tag{4}$$

- *Least Absolute Shrinkage and Selection Operator* (LASSO) [102, 110], defined by

$$\hat{x} \in \operatorname*{argmin}_{z \in \mathbb{C}^N} \|y - Az\|^2 + \lambda \|z\|_1, \tag{5}$$

  where $\lambda > 0$ is a tuning parameter balancing the data fidelity and ℓ^1 sparsity promoting terms. One can see the LASSO as the unconstrained form of QCBP in the sense that the solutions of the two can be directly related (see, e.g., [55, Chapter 3]). Due to its unconstrained formulation, LASSO can be solved very efficiently, mostly preferred over QCBP and thus has gained a prominent position for variable selection in statistical applications and signal processing. Also, noise-blind [17] and fault-tolerant [25] variants of the LASSO have been proposed. We will return to the LASSO and its variants in Chapter 2, where we will propose iterative sparse recovery solvers based on the weighted generalization of the LASSO.

Despite the plethora of methods and solvers for convex optimization methods (i) they are usually computationally costly, especially when the ambient dimension N is high; (ii) ℓ^1-relaxation leads to a compressible (i.e., *approximately* sparse) solutions, rather than sparse, and extra thresholding step is required to retrieve a sparse approximation; (iii) optimization programs themselves need to be solved by an algorithm, hence there is an intrinsic gap in their theoretical analysis with respect to practice. These motivations encourage one to consider the following alternative strategies.

(b) **Greedy and iterative algorithms** attempt to directly solve (2) or (3), leaning against solving a more tractable local optimization problem that recursively constructs the solution through subsequent iterations. Greedy algorithms are the most prominent group in this category. They are called "greedy" as they favor locally optimized steps that converge towards the solution iteratively rather than searching for a global solution from the beginning. Flagship algorithms in this category are *Orthogonal Matching Pursuit* (OMP) [83, 93] (see Algorithm 4.1), *Iterative Hard Thresholding* (IHT) [21, 22] (see Algorithm 4.2) and *Compressive Sampling Matching Pursuit* (CoSaMP) [91] (see Algorithm 5.1). All these algorithms bear the computational complexity of $\mathcal{O}(TmN)$, where T is the number of iterations, so with N large enough they can all be considered as linear-time. Due to the motivations stated earlier and other reasons to be clarified, they constitute the main theme of this thesis. In particular, many of the ideas and methods produced in this work rely on OMP as a case study due to its simple formulation. However, implications and extensions of these techniques for other algorithms will be discussed at different points throughout the text (see, e.g., Section 5.3.1 and Chapter 6). Specifically, IHT and CoSaMP will be studied in more detail in Chapters 4 and 5 in conjunction with *neural network unfolding* (see Section 1.1.4).

1.1.3 Structured sparsity

If one has some a priori knowledge about the structure of the signal to be reconstructed, then another interesting question arises:

(Q4) *How can a priori knowledge of the signal structure improve recovery?*

An insightful geometrical interpretation of the sparse recovery problem (2) is to search for an s-dimensional subspace $\mathcal{X}$ over the union of $\binom{N}{s}$ possible choices, i.e.,

$$\Sigma_s^N = \bigcup_{k=1}^{\binom{N}{s}} \mathcal{X}_k, \; \mathcal{X}_k = \{x : x|_{\Omega_k} \in \mathbb{C}^s, x|_{\Omega_k^c} = 0\}, \quad \text{s.t.} \quad \mathcal{X}_k \subseteq \mathbb{C}^N,$$

where $\Omega_1, \ldots, \Omega_{\binom{N}{s}}$ are all possible subsets of $[N]$ such that $|\Omega_k| = s, \; k = 1, \ldots, \binom{N}{s}$. Based on some prior information available, one might be able to reduce the search space, by rejecting/promoting some of these subspaces, i.e., to promote some structure within the reconstructed signal. This provides several benefits on both the measurement and the signal recovery side. In fact, suitably embedding the structure in the reconstruction pipeline can lead to faster recovery techniques with improved theoretical bounds [15, 56], from fewer measurements (better sample complexity) [14]. As a result thus far, many structures have been considered and studied in the sparse recovery and compressive sensing literature, including *sparsity in levels* [5, 9], *block sparsity*[47, 51], *joint sparsity*[8], *tree sparsity*[15, 82] to name but a few. In addition to these structures, there is another type of sparsity, known as *weighted sparsity*, which is not a structure itself but rather a way to promote structure [11, 98]. However, although weighted sparsity has been largely incorporated into and studied for convex optimization [56, 98], its integration into greedy and iterative algorithms remains far from being well studied. This will be studied in detail in the subsequent chapters.

We conclude by illustrating an important example of structured sparsity called "sparsity in levels" [5]. This is of interest here because the experiments of Chapters 4 and 5 rely on promoting sparsity in levels through weights.

Definition 1.1 (Sparsity in levels [5])**.** *Let $r \geq 1$, $M = (M_1, \ldots, M_r)$, where $1 \leq M_1 \leq M_2 \leq \cdots \leq M_r = N$ and $s = (s_1, \ldots, s_r)$, where $s_k \leq M_k - M_{k-1}$ for $k = 1, \ldots, r$, with $M_0 = 0$. A vector $x = (x_i)_{i=1}^N \in \mathbb{C}^N$ is (s, M)-sparse if*

$$|\mathit{supp}(x) \cap \{M_{k-1} + 1, \ldots, M_k\}| \leq s_k, \; k = 1, \ldots, r.$$

We denote $\overline{s} = s_1 + \cdots + s_r$ for the total sparsity, and we clearly have $\Sigma_{(s,M)} \subseteq \Sigma_{\overline{s}}^N$.

More intuitively, this means that the signal dimension $[N]$ is divided into a certain number of levels with a sparsity budget in each level. A prominent example of this type of structure can be found in *multiresolution analysis* of wavelets (see, e.g., [5, 82]), that serves as a suitable basis for natural images. We add in passing that OMP, IHT and CoSaMP have been previously extended to the sparsity in levels setting [9].

1.1.4 Algorithm unrolling: Deep neural networks for sparse recovery

In recent years, there has been a significant technological leap, largely driven by the advent of *artificial neural networks*–a data-driven approach based on the composition of multiple layers of nonlinear units, or "neurons", inspired by models of the human brain [119]. More formally, for L layers and given input x, they compute an output

$$y = W_L\sigma(W_{L-1}\sigma(\dots\sigma(W_1x + b_1) + \dots) + b_{L-1}) + b_L,$$

where $W_i,\ b_i,\ i = 1, \dots, L$ are trainable weights and biases, respectively, and σ is a nonlinear activation function. These networks are designed to make accurate predictions after being trained on large amounts of data through a learning process, that involves updating training parameters by optimizing a so-called *loss function* (see Chapter 4). Various neural network architectures have been proposed, such as *Multi Layer Perceptron* (MLP) [100], *Convolutional Neural Networks* (CNN) [77], *Residual Networks* (ResNet) [65], transformers [115], etc. In particular, the last decade or so has witnessed the advantage of depth (i.e., $L \gg 1$) (see, e.g., [108]) leading to deep neural networks, or *deep learning* [58]. These models have shown to serve a variety of applications from self-driven vehicles to language models [43].

However, there remain many concerns regarding the safe use of deep networks in critical sectors such as medical imaging and other areas directly related to human lives, beyond environmental concerns due to high amount of data and required computational power [35, 66]. The principal reason behind these concerns is model-blindness of these architectures due to their black-box nature, meaning that they mostly incorporate very little knowledge about the underlying physical model. This renders neural networks fundamentally unstable and untrustworthy, mathematically speaking lacking recovery guarantees. This leads us to the last fundamental question related to the three principles introduced at the beginning of this chapter:

(Q5) *How can we design a model-based data-driven recovery method for sparse recovery? Or on the flip side, how can sparse recovery algorithms be of help in building safer and more efficient neural networks?*

There has been much research in recent years trying to answer this question from various perspectives [81, 103], but one promising direction is through the lens of a paradigm called *algorithm unrolling* (or *unfolding*) [61, 89]. In algorithm unrolling, each iteration of an iterative algorithm plays the role of a layer of a neural network, and then by end-to-end training of such a network, one can optimize the performance of the algorithm with respect to some of its parameters. A remarkable benefit of algorithm unrolling is that neural networks designed in this way can admit stable and robust recovery guarantees [37]. In this dissertation, we address (Q5) through the lens of algorithm unrolling in conjunction with greedy sparse recovery algorithms. This will be the subject of Chapters 4 and 5.

1.1.5 Some mathematical tools for sparsity

We conclude this section by introducing some mathematical tools employed throughout the thesis. They possess key significance as they build the foundations of our theoretical analysis in the generalization of recovery error bounds for OMP to the weighted setting in Chapter 3. In particular, we need: first of all, a notion of signal compressibility for signals that are not completely sparse. This notion is captured by the *best s-term approximation error* in ℓ^p norm[1], i.e., for a signal $x \in \mathbb{C}^N$ and for $p > 0$,

$$\sigma_s(x)_p := \inf_{z \in \mathbb{C}^N,\ \|z\|_0 \leq s} \|x - z\|_p, \tag{6}$$

which is, informally, the ℓ^p energy of the tail of the signal. Secondly, we need a way of assessing the quality of the measurement matrix, in terms of how distorting its action is over s-sparse signals.

Definition 1.2 (Restricted Isometry Property (RIP) [26, 27]). *The Restricted Isometry Constant (RIC) $\delta_s < 1$ of a matrix $A \in \mathbb{C}^{m \times N}$ is the smallest $\delta > 0$ such that*

$$(1 - \delta)\|x\|^2 \leq \|Ax\|^2 \leq (1 + \delta)\|x\|^2, \quad \forall \textit{ s-sparse vectors } x \in \mathbb{C}^N. \tag{7}$$

A matrix satisfying Equation (7) is said to have the Restricted Isometry Property (RIP) of order s with constant δ.

One can see that the smaller δ_s is, the closer A is to acting as an isometry over s-sparse signals, thus the easier it is to create the inverse path to the solution from measurement. In particular, a direct implication of RIP is that all singular values of $A_S,\ S \subset [N],\ \mathrm{card}(S) \leq s$ lie in the interval $[\sqrt{1 - \delta_s}, \sqrt{1 + \delta_s}]$, thus A_S is injective when $\delta_s < 1$. Although hard to evaluate in practice, the RIP is of high theoretical significance in establishing recovery guarantees for compressive sensing with optimal sample complexity bounds.

1.2 Main contributions of the thesis

This dissertation revolves around the central themes of greedy sparse recovery (parsimony), weighted sparsity (effect of domain knowledge) and neural network unrolling (learning). In the following we briefly explain our contributions while organizing them in two parts:

1.2.1 Part I: Weighted greedy sparse recovery algorithms

OMP seeks a solution to (2) in a greedy manner: it reconstructs the signal by iteratively enlarging the support one index at a time, followed by fitting data via least-squares restricted to the current support. Aiming at generalizing OMP to the weighted case, with weights to encapsulate prior information about the signal in question, we will adopt two strategies that will be discussed in Chapters 2 and 3:

(1) Chapter 2 sheds light on the connection between greedy algorithms (with OMP as a case study) and convex optimization methods, while filling the gap regarding weighted generalization of greedy algorithms to the weighted case. The results presented in this chapter are

[1] For a vector $x \in \mathbb{C}^N$, ℓ^p-norm is defined as $\|x\|_p := (\sum_j |x_j|^p)^{1/p}$, $p \geq 1$. Also, $\|x\|_\infty = \sup_{j \in [N]} |x_j|$.

published in the *Springer Journal of Sampling Theory, Signal Processing, and Data Analysis* in collaboration with the supervisor, Prof. Simone Brugiapaglia [86]. In particular,

- In [86], we view the index selection rule of OMP as a local loss-function minimization problem. Adopting this perspective, we build upon the weighted OMP strategy proposed in [2] based on a weighted ℓ^0-based LASSO formulation, to a large class of loss functions of the form

$$G(z) := F(z) + \lambda R(z),\ \forall z \in \mathbb{F}^N,\ F, R : \mathbb{F}^N \to [0, +\infty), \tag{8}$$

  where F is a data fidelity term, R a regularization term (not necessarily differentiable), and $\lambda > 0$ a tuning parameter that balances out effects of F and R. This leads to general loss-function based greedy algorithm presented in Algorithm 2.1.

- For the weighted LASSO-based family, we show theoretically that minimizing $G(z)$ leads to explicit choice of the greedy selection rule for both weighted ℓ^0 and ℓ^1 sparsity-inducing norms, i.e., when

$$R(z) = \|z\|_{1,w} = \sum_{j \in \text{supp}(z)} w_j |z_j|, \quad \text{or} \quad R(z) = \|z\|_{0,w} = \sum_{j \in \text{supp}(z)} w_j^2,$$

  with $w \in \mathbb{R}^N,\ w_j > 0$ as the weight vector. These results are presented in Theorem 2.3 in the general case, in Theorems 2.5 to 2.7 for the ℓ^1-norm and in Theorem 2.12 and Theorem 2.13 for the ℓ^0-norm.

- Our numerical experiments demonstrate that these *greedy LASSO*-type algorithms inherit desirable characteristics of their corresponding convex optimization programs, such as noise-blindness of Square-root LASSO [17] and fault-tolerance of LAD-LASSO [6, 25]. Furthermore, this regularization prevents overfitting, and consequently, improves the robustness of OMP with respect to the number of iterations and the overall performance.

(2) In Chapter 3 we follow up on one of our proposal for future work proposed in Section 2.7 of Chapter 2. Specifically, our contributions in this chapter are as follows:

- We consider a weighted generalization of OMP in Algorithm 3.1 that we call *rescaling-based WOMP*, with the greedy selection criterion term divided by weights.

- We derive a theoretical recovery guarantee for rescaling-based WOMP based on a weighted variant of the RIP under suitable assumptions. These include recovery guarantees on the signal in terms of the best weighted approximation error and noise energy (Theorem 3.8), and a bound for the residual in terms of the noise energy (Proposition 3.7).

- Our proofs theoretically justify the choice of the greedy selection criterion, that has been missing in earlier (mostly practical) applications of rescaling-based WOMP.

- On the numerical side, we compare WOMP with two weighting strategies: the rescaling-based strategy of Algorithm 3.1 and ℓ^1-LASSO-WOMP presented in Chapter 2. In addition, we demonstrate the benefits of weights, especially in undersampled regimes.

1.2.2 Part II: Unrolled greedy sparse recovery algorithms

In the second part of this thesis, which includes Chapters 4 and 5, we turn our attention to implementing iterations of (weighted) greedy sparse recovery algorithms onto layers of a neural network, under the algorithm unrolling paradigm. The results of Chapter 4 and Section 5.1 in Chapter 5 are in collaboration with Prof. Matthew J. Colbrook at the University of Cambridge and the thesis' supervisor, Prof. Simone Brugiapaglia. They have been submitted to the *SIAM journal on Mathematics of Data Science* [88] and published in a conference paper for the *2024 International Workshop on the Theory of Computational Sensing and its Applications to Radar, Multimodal Sensing and Imaging (CoSeRa)* [87], respectively. Also, the results of Section 5.3 were obtained in collaboration with Prof. Mark Iwen at Michigan State University.

The main challenge when unrolling greedy sparse recovery algorithms (and many others) is the presence of the non-differentiable (arg)sorting operator among their iterations and also the implicit connection between steps of these algorithms. These issues obstruct the gradient flow, which is essential in the neural networks' training phase, which relies on gradient-based optimization. In this part, we also consider IHT and, to a certain extent, CoSaMP. We further investigate the *Median Recovery* (MR) algorithm [34, 69] from the deterministic compressed sensing realm. All these algorithms employ argsorting in their algorithmic routine. Specifically, the greedy selection rule of OMP and CoSaMP and the hard-thresholding operator in IHT, CoSaMP and MR involve argsorting. Our contributions regarding these chapters are the following:

- We resolve the gradient blockage issue by reinterpreting these algorithms through a projection based lens, and approximating permutation matrices related to these projections by a continuous proxy of argsort's permutation matrices called "softsorting" [96]. To the best of our knowledge, this is the first time that a fully mathematical treatment of the non-differentiability of argsort-based operators in these algorithms has been proposed, rather than empirically driven deep learning tweaks.

- For all these algorithms, we theoretically analyze and experimentally demonstrate that their differentiable versions offer reliable approximations, with accuracy adjustable via the *temperature parameter* of softsort.

- In particular, for OMP and IHT we derive theoretical results in the form of recovery guarantees showing that Soft-OMP and Soft-IHT effectively approximate OMP and IHT under suitable conditions on the temperature parameter.

- As a quest to discover hidden structures within the data, we consider weights as trainable parameters and implement these differentiable algorithm onto neural networks and propose the *OMP-Net*, *IHT-Net* and *MR-Net* architectures, and show that these networks are trainable.

- We demonstrate that with weights as trainable parameters, OMP- and IHT and MR-Net remarkably outperform, respectively, OMP, IHT and MR in heavily undersampled regimes, provided that the data exhibits sufficient structure.

1.3 Outline

The remainder of the thesis is organized as follows. In Chapter 2, we study weighted generalization of OMP through a loss function-based perspective, proposing weighted greedy LASSO-type algorithms with rigorous index choices for their greedy selection criterion. Then we turn to the rescaling-based weighting strategy in Chapter 3, where we prove a weighted RIP-based recovery guarantee for weighted OMP. Chapter 4 investigates unrolling OMP and IHT iterations over neural networks, handling the nondifferentiablity issue within these algorithms and supported by solid theoretical analysis. Chapter 5 presents some complementary numerical results for Chapter 4, namely unfolding CoSaMP and MR. The latter extends the applicability of the unrolling framework to deterministic compressed sensing.

Chapter 2

LASSO-type weighted OMP

"All are equal, but some are more equal than others!"
— *Animal Farm*, George Orwell

The content of this chapter, in its current form (modulo very minor edits), is published in the *Springer Journal of Sampling Theory, Signal Processing, and Data Analysis* in collaboration with the supervisor, Prof. Simone Brugiapaglia [86].

2.1 Introduction

Sparse recovery lies at the heart of modern data science, signal processing, and statistical learning. Its goal is to reconstruct an N-dimensional s-sparse signal x (i.e., such that $\|x\|_0 := |\{j : x_j \neq 0\}| \leq s$) from m (possibly noisy) linear measurements $y = Ax + e$, where A is an $m \times N$ measurement (sensing, mixing, or dictionary) matrix and e is an m-dimensional noise vector. In this chapter, we focus in particular on the *compressed sensing* framework [26, 45], corresponding to the underdetermined regime (i.e., $m < N$). For a general treatment of sparse recovery, compressed sensing and their numerous applications in data science, signal processing, and scientific computing we refer to, e.g., the books [3, 11, 49, 50, 55, 64, 75, 117].

Sparse recovery techniques are typically divided into two main categories: convex relaxation methods and iterative algorithms. In convex relaxation methods, sparse solutions are identified by solving convex optimization programs such as those based on ℓ^1 minimization. Popular examples are *(Quadratically-Constrained) Basis Pursuit* and the *Least Absolute Shrinkage and Selection Operator* (*LASSO*). On the other hand, iterative algorithms aim at computing a sparse solution through explicit iterative algorithmic procedures that combine techniques from numerical linear algebra with sparsity-enhancing ideas. These include thresholding-based algorithms such as *Iterative Hard Thresholding* (*IHT*) and *Hard Thresholding Pursuit* (*HTP*), and greedy algorithms such as *Compressive Sampling Matching Pursuit* (*CoSaMP*) and *Orthogonal Matching Pursuit* (*OMP*)—the main object of study of this chapter. For a detailed overview of these and other sparse recovery techniques we refer readers to, e.g., [3, 55, 75, 109, 117].

Over the last few years, motivated by the need to incorporate prior knowledge about the target signal into sparse reconstruction methods, a substantial amount of research has been devoted to *weighted* sparse recovery. In a variety of applications, ranging from compressive imaging to surrogate modelling and uncertainty quantification, it has been shown both empirically and theoretically that a careful choice of weights can improve both reconstruction accuracy and sample complexity with respect to unweighted ℓ^1 minimization. A non-exhaustive list of works in this direction includes [1, 3, 4, 7, 11, 14, 28, 32, 56, 67, 72, 94, 98, 123, 126] and references therein.

Although weighted sparse recovery has been extensively investigated from the perspective of convex relaxation through weighted ℓ^1 minimization, iterative algorithms are far from being well studied in the weighted setting. To the best of our knowledge, iterative algorithms for weighted sparse recovery have only been considered in a handful of works [2, 71, 78, 121]. The main goal of our chapter is to reduce this gap. With this aim, adopting an approach that merges convex relaxation and iterative algorithms, we propose new LASSO-based weighted greedy algorithms of OMP type.

2.1.1 Main contributions

The main contributions of this chapter can be summarized as follows.

(1) Adopting a *loss function-based* perspective (see Section 2.1.2 and Section 2.3), we propose a new class of greedy algorithms able to promote weighted sparse recovery based on the OMP paradigm. They are defined via theoretically-justified greedy index selection rules based on maximal reduction of weighted LASSO-type loss functions (see Theorem 2.3, Theorem 2.5, Theorem 2.6 and Theorem 2.7). These include the weighed (unconstrained) LASSO and two of its most notable variants: the weighted *Square-Root LASSO* (*SR-LASSO*) and *Least Absolute Deviations LASSO* (*LAD-LASSO*). This loss function-based perspective allows one to adapt OMP to various structured signal models and sources of errors corrupting the data.

(2) The proposed algorithms are numerically shown to outperform standard OMP (with respect to both accuracy and computational cost) and inherit the desirable characteristics of the underlying loss functions. In particular, those based on the SR- and LAD-LASSO, have noise-blind tuning parameter selection strategies and fault-tolerance, respectively. In addition, thanks to the presence of a regularization term, our greedy algorithms prevent overfitting and, consequently, improve the robustness of OMP with respect to the number of iterations. Numerical evidence in this direction is presented in Section 2.4. These results shed new light on the connection between convex relaxation methods and iterative (specifically, greedy) algorithms.

(3) The proposed algorithms admit a reliable stopping criterion and a significant reduction in runtime, thanks to the regularization effect mentioned above.

We conclude with a remark about the novelty of our contributions in relation to an OMP variant proposed in [2]. A comprehensive literature review can be found in Section 2.1.3

Remark 2.1. *Our construction in this work builds upon a variant of OMP proposed in [2] that relies on a weighted ℓ^0-based LASSO formulation. Here, adopting a more general loss function-based perspective, we extend the work of [2] to a broader class of loss functions including ℓ^1-based*

LASSO and other variants of the LASSO family, i.e., weighted SR- and LAD-LASSO. Moreover, we extend the weighted OMP strategy proposed in [2] to the case of ℓ^0-based SR- and LAD-LASSO. See Section 2.6.

2.1.2 Summary of the main results

We now provide an overview of our main results, referring to Section 2.3 for a detailed technical discussion. Our objective is to construct a signal that minimizes a loss function of the form

$$G(z) := F(z) + \lambda R(z), \quad \forall z \in \mathbb{F}^N, \tag{9}$$

where $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$, and $F, R : \mathbb{F}^N \to [0, +\infty)$ are a data-fidelity and a regularization term, respectively, and $\lambda \geq 0$ is a tuning parameter. For weighted LASSO, SR-LASSO, and LAD-LASSO loss functions (see Equations (25) to (27), respectively) F is an ℓ^2- or ℓ^1-based data-fidelity term and R is a weighted ℓ^1 norm. We aim at minimizing G in a greedy fashion. Following the OMP paradigm, we construct the support of the signal one index at a time. Specifically, at iteration k, the support set $S^{(k)}$ of the approximation $x^{(k)}$ is updated according to the following *greedy index selection rule*:

$$S^{(k)} = S^{(k-1)} \cup \{j^{(k)}\}, \quad \text{where} \quad j^{(k)} \in \arg\max_{j \in [N]} \Delta(x^{(k-1)}, S^{(k-1)}, j),$$

where $\Delta(x^{(k-1)}, S^{(k-1)}, j)$ is the loss reduction resulting from adding a single index j to the support $S^{(k-1)}$ of $x^{(k-1)}$ and with $[N] := \{1, \dots, N\}$. Δ is implicitly defined by

$$\min_{t \in \mathbb{F}} G(z + te_j) = G(z) - \Delta(z, S, j), \quad \text{where } S := \operatorname{supp}(z), \quad \forall z \in \mathbb{F}^N. \tag{10}$$

Then, the signal is updated by solving a local optimization problem restricted to the newly constructed support $S^{(k)}$, i.e.,

$$x^{(k)} \in \arg\min_{z \in \mathbb{F}^N} F(z) \quad \text{s.t} \quad \operatorname{supp}(z) \subseteq S^{(k)}. \tag{11}$$

The corresponding loss function-based OMP algorithm is presented in Algorithm 2.1 (adopting a stopping criterion based on the number of iterations).

Remark 2.2 (Standard OMP). *The standard OMP algorithm is a special case of Algorithm 2.1 when G is the least-squares loss function, i.e., $G(z) = F(z) = \|y - Az\|_2^2$ and for $\lambda = 0$.*

To demonstrate that Algorithm 2.1 is practically implementable, we ought to show that the loss reduction factor $\Delta(x, S, j)$ is (ideally, easily) computable. The main technical contribution of the chapter is to show that this is indeed the case for the weighted LASSO, SR-LASSO, and LAD-LASSO loss functions (referred to as "$*$-LASSO" below). This is summarized in the following result, which unifies Theorem 2.5, Theorem 2.6 and Theorem 2.7.

Theorem 2.3 (Weighted $*$-LASSO-based greedy selection rules). *Let $\lambda \geq 0$, $S \subseteq [N]$,*

$$A \in \mathbb{F}^{m \times N} \quad \begin{cases} \text{with } \mathbb{F} = \mathbb{C} \text{ and } \ell^2\text{-normalized columns} & \text{(LASSO and SR-LASSO)} \\ \text{with } \mathbb{F} = \mathbb{R} \text{ and nonzero columns} & \text{(LAD-LASSO)} \end{cases}$$

Algorithm 2.1 Loss function-based OMP

```
 1: Inputs: $G : \mathbb{F}^N \to [0, +\infty)$ (loss function of the form (9)), with $\mathbb{F} = \mathbb{R}$ or $\mathbb{C}$; $A \in \mathbb{F}^{m\times N}$
    (measurement matrix); $y \in \mathbb{F}^m$ (measurement vector); $K \in [N]$ (number of iterations).
 2: Output: $x^{(K)} \in \mathbb{F}^N$ (approximate $K$-sparse solution to $Az = y$).
 3: procedure LOSS-FUNCTION BASED OMP($G$, $A$, $y$, $K$)
 4:     Let $x^{(0)} = 0$ and $S^{(0)} = \emptyset$
 5:     for $k = 1, \ldots, K$ do
 6:         Find $j^{(k)} \in \arg\max_{j\in[N]} \Delta(x^{(k-1)}, S^{(k-1)}, j)$, with $\Delta$ defined as in (10)
 7:         Define $S^{(k)} = S^{(k-1)} \cup \{j^{(k)}\}$
 8:         Compute $x^{(k)}$ by solving (11)
 9:     end for
10:     return $x^{(K)}$
11: end procedure
```

and $x \in \mathbb{F}^N$ satisfying

$$x \in \arg\min_{z\in\mathbb{F}^N} F(z) \quad \textit{s.t} \quad \operatorname{supp}(z) \subseteq S.$$

Then, the loss reduction $\Delta(x, S, j)$ defined in (10) *admits explicit formulas provided by* (29)*,* (33) *and* (40)*, respectively, for the weighted LASSO, SR-LASSO and LAD-LASSO loss functions (see* (25)*,* (26) *and* (27)*).*

2.1.3 Literature review

Weights have been employed in sparse recovery methods for various purposes. For instance, in the seminal work [28], the authors propose to solve a sequence of (re)weighted ℓ^1 minimization problems to enhance sparse signal recovery. In our context, weights can generally be thought of as a way of incorporating prior information about the signal into a sparse recovery model. In *adaptive LASSO* [67, 126], a data-driven but careful choice of weights is shown to admit near oracle properties. Adaptive LASSO is fundamentally different from our greedy ℓ^1-LASSO WOMP as adaptive LASSO solves the global convex LASSO program, while our greedy technique seeks the solution to the sparse recovery problem of Equation (2) using consecutive local optimization problems based on the LASSO loss function. Works such as [56, 123] show that replacing the ℓ^1-norm with its weighted version can improve recovery assuming that accurate (partial) support knowledge is provided. A similar result was derived in [72] from a probabilistic point of view where the signal support is assumed to be formed by two subsets with different probability of occurrence. Further studies of weighted ℓ^1 minimization and its impactful application in the context of function approximation from pointwise samples and uncertainty quantification include [1, 7, 11, 94, 98]. The notion of weighted sparsity was formalized in [98]. Weighted sparsity is related to structured sparsity (see [15]). In fact it allows one to *promote* structures (rather than being a structure itself). For example, in the context of high-dimensional function approximation (see [11] and references therein) weights are able to promote so-called *sparsity in lower sets*, which largely contributes to mitigating the curse of dimensionality in the sample complexity. Using the weighted ℓ^1 minimization to improve the sample complexity was also addressed in [14] in the signal processing context. Apart from convex ℓ^1-minimization, weights are implemented in algorithms such as weighted IHT

[71], and weighted OMP [2, 23, 78, 121] (see below).

OMP and its non-orthogonalized version, Matching Pursuit (MP), were introduced in [83, 93] for time-frequency dictionaries, and later analyzed in, e.g., [92, 112, 113]. Well-known advantages of OMP are its simple and intuitive formulation and its computational efficiency, especially for small values of sparsity. A lot of research has been devoted to improve OMP, e.g., by allowing the algorithm to select several indices at each iteration or combining it with thresholding strategies [41, 46, 91, 92], or by optimizing the greedy selection rule [99]. The loss function-based perspective adopted in our work is related to the approach in [104, 125]. However, there are at least two key differences with our setting: (i) we do not assume the loss function to be differentiable and (ii) the corresponding greedy selection criterion is not based on the gradient of the loss function. Our framework extends both the standard OMP algorithm and the weighted OMP algorithm proposed in [2], based on ℓ^0 regularization. To the best of our knowledge, the only other works that incorporate weights into OMP, but with different weighting strategies than those proposed here, are [23, 78, 121].

Let us finally consider the family of greedy coordinate descent algorithms (see, e.g., [80]). They aim to solve a given optimization problem by selecting one coordinate index at a time and minimizing the loss function with respect to the corresponding entry while freezing all the others. Although their greedy selection method coincides with the one adopted in this chapter, greedy coordinate descent algorithms differ from loss function-based OMP since their greedy index selection is not combined with the solution of a local data-fitting optimization problem of the form (11). In addition, the greedy coordinate selection in [80] is only explicitly computed for the unweighted LASSO, whereas here we derive explicit greedy index selection rules for weighted LASSO, SR-LASSO and LAD-LASSO.

2.1.4 Outline of the chapter

The rest of the chapter is organized as follows. In the next section we discuss in detail the loss function-based OMP framework summarized in Section 2.1.2 and present weighted $*$-LASSO-based greedy selection rules. Then, we illustrate the practical performance of $*$-LASSO-based OMP through numerical experiments in Section 2.4 and outline open problems and future research directions in Section 2.7. Section 2.5 contains the proofs of Theorem 2.5, Theorem 2.6 and Theorem 2.7, stated in Section 2.3. In Section 2.6, we present ℓ^0-based variants of the proposed algorithms.

2.2 LASSO-based weighted OMP

In this section we present LASSO-based weighted OMP (WOMP) algorithms. In order to theoretically justify our methodology, we first review the rationale behind greedy algorithms such as OMP, emphasizing the role played by certain (regularized) loss functions.

2.2.1 Loss function-based OMP

Greedy algorithms such as OMP are iterative procedures characterized by the following two steps:

(i) the iterative construction of signal's support by means of greedy index selection;

(ii) the computation of signal's entries on (a subset of) the constructed support by solving a "local" optimization problem.

In this section, we describe a general paradigm to perform these two operations (and, consequently, design greedy algorithms) from the perspective of loss functions. Specifically, we consider an optimization problem of the form

$$\min_{z\in\mathbb{C}^N} G(z) := \min_{z\in\mathbb{C}^N} \left(F(z) + \lambda R(z)\right), \tag{12}$$

where $G, F, R : \mathbb{C}^N \to [0, +\infty)$ and $\lambda \geq 0$. Here G is a (regularized) loss function, composed by a *data fidelity* term F and a *regularization* term R, balanced by a *tuning parameter* λ.

Aiming to minimize G, in Step (i) an OMP-type greedy algorithm constructs the signal support by selecting the index (or indices) leading to a maximal reduction of the loss function G—this is why this type of algorithm is called "greedy". Specifically, given a support set $S^{(k-1)}$ and an approximation $x^{(k-1)}$, at iteration k the algorithm constructs a new index set $S^{(k)}$ as follows:

$$S^{(k)} = S^{(k-1)} \cup \{j^{(k)}\}, \quad \text{where } j^{(k)} \in \arg\max_{j\in[N]} \Delta(x^{(k-1)}, S^{(k-1)}, j),$$

with $\Delta : \mathbb{C}^N \times 2^{[N]} \times [N] \to [0, +\infty)$ (where 2^X denotes the power set of X) implicitly defined by

$$\min_{t\in\mathbb{C}} G(x^{(k-1)} + te_j) := G(x^{(k-1)}) - \Delta(x^{(k-1)}, S^{(k-1)}, j). \tag{13}$$

Here $\Delta(z, S, j)$ is the loss function reduction corresponding to adding the index $j \in [N]$ to the support $S \subseteq [N]$ and given a current approximation $z \in \mathbb{C}^N$. In fact, rearranging the above relation leads to

$$\Delta(x^{(k-1)}, S^{(k-1)}, j) = \max_{t\in\mathbb{C}}[G(x^{(k-1)}) - G(x^{(k-1)} + te_j)].$$

After a suitable updated support $S^{(k)}$ is identified, in Step (ii) the approximation $x^{(k-1)}$ is updated as $x^{(k)}$ by solving a local data-fitting optimization problem. This optimization problem takes the form

$$x^{(k)} \in \arg\min_{z\in\mathbb{C}^N} F(z) \quad \text{s.t. } \operatorname{supp}(z) \subseteq S^{(k)}. \tag{14}$$

Note that this local optimization only involves the data-fidelity term F and not the regularization term R. As we will see, this will lead to theoretical benefits in order to formally certify that Δ corresponds to the maximal reduction of G. Moreover, the choice of the support $S^{(k)}$ is already regularized. Therefore, the local optimization performs only a data fitting step onto the regularized subspace $\{z \in \mathbb{C}^N : \operatorname{supp}(z) \subseteq S^{(k)}\}$. This can be summarized in the following iteration.

Loss function-based OMP iteration

$$S^{(k)} = S^{(k-1)} \cup \{j^{(k)}\}, \quad \text{with } j^{(k)} \in \arg\max_{j\in[N]} \Delta(x^{(k-1)}, S^{(k-1)}, j) \text{ and } \Delta \text{ as in (13)} \tag{15}$$

$$x^{(k)} \in \arg\min_{z\in\mathbb{C}^N} F(z) \quad \text{s.t. } \operatorname{supp}(z) \subseteq S^{(k)} \tag{16}$$

We now revisit the standard OMP algorithm and the weighted OMP algorithm of [2] in light of the above framework.

Standard OMP. With the above discussion in mind, we consider the least-squares loss function, without regularization (i.e., $\lambda = 0$),

$$G^{\mathrm{LS}}(z) = F^{\mathrm{LS}}(z) := \|y - Az\|_2^2, \quad \forall z \in \mathbb{C}^N, \tag{17}$$

where $y \in \mathbb{C}^m$ is a vector of measurements (or observations) and $A \in \mathbb{C}^{m \times N}$ is a measurement (or design) matrix with ℓ^2-normalized columns. With this choice, steps (15) and (16) correspond to the following well-known iteration of the OMP algorithm:

$$S^{(k)} = S^{(k-1)} \cup \{j^{(k)}\}, \quad \text{where } j^{(k)} \in \arg\max_{j \in [N]} \Delta^{\mathrm{LS}}, \quad \Delta^{\mathrm{LS}} = |(A^*(y - Ax^{(k-1)}))_j|, \tag{18}$$

$$x^{(k)} \in \arg\min_{z \in \mathbb{C}^N} \|y - Az\|_2^2 \quad \text{s.t.} \quad \operatorname{supp}(z) \subseteq S^{(k)}. \tag{19}$$

In OMP the index selected at each iteration maximizes the correlation between the columns of the matrix A and the residual vector $r^{(k-1)} := y - Ax^{(k-1)} \in \mathbb{C}^m$, but interestingly at the same time it also maximizes the least-squares loss reduction. In fact, it is possible to show that (see, e.g., [55, Lemma 3.3]) the problem

$$\min_{t \in \mathbb{C}} G^{\mathrm{LS}}(z + te_j) = G^{\mathrm{LS}}(z) - |(A^*(y - Az))_j|^2$$

prescribes the choice of the new greedy index.

We note that sparsity is not directly promoted by minimizing the least-squares loss function G^{LS} of OMP. In fact, in OMP the sparsity of the approximated solution is directly related with the number of iterations. Specifically, each iteration adds a single index to the support $S^{(k)}$ (thanks to the orthogonality due to the least-squares, no index is selected twice). Hence, running s iterations of OMP generates an s-sparse vector (i.e., with $\|x^{(s)}\|_0 \leq s$). Although very powerful in the case of standard sparsity, standard OMP does not directly allow one to promote other sparsity structures, such as, e.g., weighted sparsity [98].

In this chapter, we focus on algorithms that are able to promote weighted sparsity [98]. Recall that, given a vector of weights $w \in \mathbb{R}^N$ with $w > 0$, the weighted ℓ^0_w- and ℓ^1_w-norm of a vector $z \in \mathbb{C}^N$ are defined as

$$\|z\|_{0,w} := \sum_{j \in \operatorname{supp}(z)} w_j^2 \quad \text{and} \quad \|z\|_{1,w} := \sum_{j \in \operatorname{supp}(z)} w_j |x_j|, \tag{20}$$

respectively [98]. As the use of weights allows one to encourage hidden structures in the ground truth signal, it is highly application dependent. Examples of specific applications with explicit weight choices include sparse polynomial approximation [11], recovery with partial support information [56, 123] and sparse-in-levels signal reconstruction [3]. In accordance with the loss function-based perspective adopted in this section, we promote weighted sparsity through the regularized loss function G, in particular, by suitable choice of the regularization term R. This idea was recently employed in [2] in the context of sparse high-dimensional function approximation, as illustrated in the next paragraph.

ℓ^0_w-based Weighted OMP (ℓ^0_w-WOMP). Inspired by the unconstrained LASSO formulation (see Section 2.2.2) [2] suggested to adopt the ℓ^0_w-regularized least squares loss function

$$G_{\ell^0_w}(z) = \|y - Az\|_2^2 + \lambda \|z\|_{0,w} \quad \forall z \in \mathbb{C}^N. \tag{21}$$

Although the loss function $G_{\ell^0_w}$ is nonconvex and discontinuous, the corresponding loss reduction function $\Delta_{\ell^0_w}$ can be explicitly computed as

$$\Delta_{\ell^0_w}(x, S, j) = \begin{cases} \max\{|(A^*(Ax - y))_j|^2 - \lambda w_j^2, 0\} & j \notin S \\ \max\{\lambda w_j^2 - |x_j|^2, 0\} & j \in S,\ x_j \neq 0 \\ 0 & j \in S,\ x_j = 0 \end{cases}, \tag{22}$$

under the assumption that A has ℓ^2-normalized columns and that

$$x \in \arg\min_{z \in \mathbb{C}^N} \|y - Az\|_2^2 \quad \text{s.t.} \quad \operatorname{supp}(z) \subseteq S.$$

For more details and a proof of this result, we refer to [2, Proposition 1]. This leads to what we will refer to as the ℓ^0_w-weighted OMP (ℓ^0_w-WOMP) algorithm, defined by the following iteration:

$$S^{(k)} = S^{(k-1)} \cup j^{(k)}, \quad \text{where } j^{(k)} = \arg\max_{j \in [N]} \Delta_{\ell^0_w}(x^{(k-1)}, S^{(k-1)}, j), \tag{23}$$

$$x^{(k)} \in \arg\min_{z \in \mathbb{C}^N} \|y - Az\|_2^2 \quad \text{s.t} \quad \operatorname{supp}(z) \subseteq S^{(k)}. \tag{24}$$

Note that for $\lambda = 0$, the ℓ^0_w-WOMP algorithms coincides with standard OMP. On top of allowing one to incorporate weights, using a regularized loss function such as $G_{\ell^0_w}$ also improves the robustness of OMP with respect to the stopping criterion. In fact, the presence of a regularization term prevents the greedy algorithm from *overfitting* due to an excessive number of iterations (see [2] for details on numerical results).

However, there is no free lunch. The possibility of including weights and the improved robustness with respect to the number of iterations come at the cost of adding an extra parameter λ that needs to be tuned appropriately. Unfortunately, the optimal value of λ (i.e., the value that minimizes the reconstruction error) depends on characteristics of the model such as the sparsity of the ground truth signal or the magnitude of the noise corrupting the measurements. This makes λ challenging to tune in general. Luckily, some insights on how to tune λ can be found in the convex optimization literature for LASSO-type loss functions. These are described in the next subsection and constitute the foundation upon which we will design the class of LASSO-inspired greedy algorithms proposed in this chapter.

2.2.2 LASSO-type loss functions for weighted ℓ^1 minimization

In this subsection we introduce different convex optimization programs that have been extensively used for weighted sparse signal recovery. Consider a vector of weights $w \in \mathbb{R}^N$ with $w > 0$. We aim to recover a sparse vector $x \in \mathbb{C}^N$ from measurements $y = Ax + e \in \mathbb{C}^m$, where $e \in \mathbb{C}^m$ is an error or noise vector corrupting the measurements. This could include errors from various sources, such as physical noise (e.g., from measurement devices), numerical or discretization error (e.g., from numerical solvers), or sparse corruptions (e.g., from node failures in a parallel computing setting). In the context of weighted sparse recovery, an approximation to the signal x from noisy measurements y can be computed by solving one of the unconstrained weighted ℓ^1 minimization problems discussed below.[1] We organize our discussion based on the nature of the noise e corrupting the measurements.

Bounded noise: weighted LASSO and SR-LASSO. If the noise satisfies a bound of the form $\|e\|_2 \leq \eta$ for a small constant η (that might be known or unknown in advance), weighted quadratically constrained basis pursuit is one of the most popular weighted ℓ^1-minimization strategies [3, 11, 98]. However, it requires the knowledge of η and it is a constrained optimization problem—hence, it is not of the form (12). For this reason, we do not consider it further in this chapter. A popular recovery strategy of the form (12) is the (unconstrained) *weighted LASSO*, defined by the loss function

$$G_{\ell^1_w}(z) := \|y - Az\|_2^2 + \lambda\|z\|_{1,w}, \quad \forall z \in \mathbb{C}^N. \tag{25}$$

The LASSO dates back at least to the pioneering works [102, 110] and since then has become one of the most widespread optimization problems in statistics and data science. Although the LASSO eliminates the need for an explicit knowledge of η, the choice of its tuning parameter λ is not straightforward. The range of values of λ that lead to theoretical optimal recovery guarantees scales linearly in $\|e\|_2$, i.e., more specifically, $\lambda \asymp \|e\|_2/\sqrt{\|x\|_{0,w}}$, [7] (or, when e is a normal random vector, on its standard deviation; see, e.g., [18, 105]). In practice, this means that λ should often be tuned via *cross validation* (see, e.g., [63]) that, although generally accurate, is often computationally daunting.

To alleviate this issue, an alternative strategy of the form (12) is the weighted *Square-Root LASSO (SR-LASSO)*, whose loss function is defined by

$$G^{\text{SR}}_{\ell^1_w}(z) := \|y - Az\|_2 + \lambda\|z\|_{1,w}, \quad \forall z \in \mathbb{C}^N. \tag{26}$$

The (unweighted) SR-LASSO was proposed in [17]. It is a well known optimization problem in statistics (see, e.g., the book [114]), and has become increasingly popular in compressive sensing [7, 10, 11, 54, 95]. There is only a small difference between the objective functions of SR-LASSO and LASSO, i.e., the lack of the exponent 2 on the data-fidelity term of SR-LASSO. However, this slight difference gives rise to substantial changes. It has been demonstrated both theoretically and empirically that the optimal choice of λ for (weighted) SR-LASSO is no longer dependent on the noise level, i.e., $\lambda \asymp 1/\sqrt{\|x\|_{0,w}}$, which facilitates parameter tuning in the presence of unknown bounded noise [7, 17].

Sparse corruptions: weighted LAD-LASSO. When the noise corrupting the measurements is of the form

$$e = e^{\text{bounded}} + e^{\text{sparse}},$$

[1]Here we do not consider constrained programs such as quadratically constrained basis pursuit or constrained LASSO (see, e.g., [3, 11, 98]) since they do not fit our framework.

where $\|e^{\text{bounded}}\|_2$ and $\|e^{\text{sparse}}\|_0$ are bounded, but $\|e^{\text{sparse}}\|_2$ is possibly very large, the LASSO and SR-LASSO are generally not able to achieve successful sparse recovery. A simple remedy is to use a data-fidelity term based on the ℓ^1-norm, as opposed to the ℓ^2-norm, thanks to its ability to promote sparsity on the residual. This is the idea behind the (unconstrained) weighted *LAD-LASSO*, whose loss function is given by

$$G^{\text{LAD}}_{\ell^1_w}(z) := \|y - Az\|_1 + \lambda\|z\|_{1,w}, \quad \forall z \in \mathbb{C}^N. \tag{27}$$

Early works on LAD-LASSO include [76, 118]. It is a regularized version of the classical *Least Absolute Deviations* (*LAD*) problem [20, 25]. In addition to signal's weighted sparsity, in weighted LAD-LASSO, the optimal choice of the tuning parameter further depends on the sparsity of e^{sparse}, i.e., $\lambda \asymp \sqrt{\|e^{\text{sparse}}\|_0/\|x\|_{0,w}}$ [6, 7]. Nonetheless, the choice $\lambda \asymp 1$ usually works well in practice [6].

2.2.3 Two key questions

Our objective for the rest of the chapter is to study OMP-type greedy algorithms characterized by the iteration (15)-(16), based on the weighted LASSO, SR-LASSO, and LAD-LASSO loss functions defined in (25), (26), and (27), respectively. Our investigation is driven by two main questions:

(Q1) *Is the quantity Δ in* (13)*, defining the OMP-type greedy selection rule, explicitly computable for the weighted LASSO, SR-LASSO, and LAD-LASSO loss functions?*

(Q2) *Are the favorable properties of the weighted SR-LASSO, and LAD-LASSO inherited by the corresponding OMP-type greedy algorithms?*

We will provide affirmative answers to both (Q1) and (Q2). The answer to (Q1) will be accompanied by explicit formulas for Δ and rigorous loss-function reduction guarantees, discussed in Section 2.3. The answer to (Q2) will be based on numerical evidence, presented in Section 2.4. Specifically, we will show that SR-LASSO-based OMP admits a *noise robust* optimal parameter tuning strategy (i.e., the optimal value of λ is independent to the noise level) and that LAD-LASSO-based OMP is *fault tolerant*, i.e. able to correct for high-magnitude sparse corruptions.

Remark 2.4 (ℓ^0_w-based regularization)**.** *It is possible to consider ℓ^0_w-based loss functions for the SR-LASSO and the LAD-LASSO, similarly to the ℓ^0_w-regularized least squares loss defined in* (21) *and employed in [2] (which correspond to an ℓ^0_w-based LASSO formulation). However, we have observed experimentally that (Q2) does not admit an affirmative answer for the ℓ^0_w-based analogs (see Experiments I and II in Section 2.4.2). For this reason, we refrained from studying the ℓ^0_w-based formulations in detail in the present chapter. Nonetheless, we provide explicit formulas for Δ for these variants in Section 2.6.*

2.3 Greedy selection rules for weighted LASSO-type loss functions

Equipped with the general loss function-based OMP paradigm presented in Section 2.2.1, we present three weighted OMP iterations based on the LASSO, square-root LASSO, and LAD-LASSO loss functions reviewed in Section 2.2.2. The proofs of the results in this section can be found in Section 2.5.

2.3.1 LASSO-based OMP

We start by considering the LASSO loss function $G_{\ell^1_w}$ defined in (25), whose corresponding greedy selection rule is identified by the following result.

Theorem 2.5 (LASSO-based greedy selection rule)**.** *Let $\lambda \geq 0$, $S \subseteq [N]$, $A \in \mathbb{C}^{m\times N}$ with ℓ^2-normalized columns, and $x \in \mathbb{C}^N$ be such that*

$$x \in \arg\min_{z\in\mathbb{C}^N} \|y - Az\|_2^2 \quad \textit{s.t.} \quad \operatorname{supp}(z) \subseteq S. \tag{28}$$

Then, for every $j \in [N]$,

$$\min_{t\in\mathbb{C}} G_{\ell^1_w}(x + te_j) = G_{\ell^1_w}(x) - \Delta_{\ell^1_w}(x, S, j),$$

where

$$\Delta_{\ell^1_w}(x,S,j) = \begin{cases} \max\left\{|(A^*(Ax-y))_j| - \frac{\lambda}{2}w_j, 0\right\}^2 & j \notin S \\ \max\left\{|x_j|(\lambda w_j - |x_j|), \lambda w_j\left(|x_j| - \frac{\lambda w_j}{4} - \left||x_j| - \frac{\lambda w_j}{2}\right|\right), 0\right\} & j \in S \end{cases}. \tag{29}$$

This leads to the following LASSO-based OMP iteration:

$$S^{(k)} = S^{(k-1)} \cup \{j^{(k)}\} \quad \text{where} \quad j^{(k)} \in \arg\max_{j\in[N]} \Delta_{\ell^1_w}(x^{(k-1)}, S^{(k-1)}, j) \tag{30}$$

$$x^{(k)} \in \arg\min_{z\in\mathbb{C}^N} \|y - Az\|_2^2 \quad \text{s.t.}\ \operatorname{supp}(z) \subseteq S^{(k)}. \tag{31}$$

Note that here again Equation (29) distinguishes between the elements already chosen in the support and future candidates, while setting $\lambda = 0$ restores the greedy selection criterion of the standard OMP in Equation (18). From a computational point of view, this step can be computed cheaply by thresholding and searching.

2.3.2 SR-LASSO-based OMP

For the SR-LASSO loss function $G^{\text{SR}}_{\ell^1_w}$ defined in (26), we have the following result.

Theorem 2.6 (SR-LASSO-based greedy selection rule)**.** *Let $\lambda \geq 0$, $S \subseteq [N]$, $A \in \mathbb{C}^{m\times N}$ with ℓ^2-normalized columns, and $x \in \mathbb{C}^N$ satisfying*

$$x \in \arg\min_{z\in\mathbb{C}^N} \|y - Az\|_2 \quad \textit{s.t.} \quad \operatorname{supp}(z) \subseteq S. \tag{32}$$

Then, for every $j \in [N]$,

$$\min_{t \in \mathbb{C}} G^{\mathrm{SR}}_{\ell^1_w}(x + te_j) = G^{\mathrm{SR}}_{\ell^1_w}(x) - \Delta^{\mathrm{SR}}_{\ell^1_w}(x, S, j),$$

where

$$\Delta^{\mathrm{SR}}_{\ell^1_w}(x, S, j) = \begin{cases} 0 & j \notin S \textit{ and } w_j \geq 1 \\ \sqrt{(1-(\lambda w_j)^2)(\|r\|_2^2 - |\langle r, a_j\rangle|^2)} + \lambda w_j |\langle r, a_j \rangle| - \|r\|_2 & j \notin S \textit{ and } \lambda w_j < 1 \\ \|r\|_2 - \sqrt{|x_j|^2 + \|r\|_2^2} + \lambda w_j |x_j| & j \in S \textit{ and } \lambda w_j \geq 1 \\ \|r\|_2 - \sqrt{\hat{\rho}^2 + \|r\|_2^2} + \lambda w_j \left(|x_j| - ||x_j| - \hat{\rho}|\right) & j \in S \textit{ and } \lambda w_j < 1 \end{cases}, \tag{33}$$

with $r = y - Ax$ and $\hat{\rho} = \min\left\{|x_j|, \frac{\lambda w_j \|r\|_2}{\sqrt{1-(\lambda w_j)^2}}\right\}$

The corresponding SR-LASSO-based OMP iteration reads

$$S^{(k)} = S^{(k-1)} \cup \{j^{(k)}\} \quad \text{where} \quad j^{(k)} \in \arg\max_{j \in [N]} \Delta^{\mathrm{SR}}_{\ell^1_w}(x^{(k-1)}, S^{(k-1)}, j), \tag{34}$$

$$x^{(k)} \in \arg\min_{z \in \mathbb{C}^N} \|y - Az\|_2 \quad \text{s.t. } \operatorname{supp}(z) \subseteq S^{(k)}. \tag{35}$$

2.3.3 LAD-LASSO-based OMP

Finally, we consider the LAD-LASSO loss function $G^{\mathrm{LAD}}_{\ell^1_w}$ defined in (27). In this case, we restrict ourselves to the real-valued case for the sake of simplicity. In order to formulate the corresponding greedy selection rule, we need to introduce some auxiliary notation. First, we define an augmented version $\widetilde{A} \in \mathbb{R}^{(m+1)\times N}$ of the matrix $A \in \mathbb{R}^{m \times N}$ as

$$\widetilde{A} := \begin{bmatrix} A \\ \lambda w^* \end{bmatrix} \quad \text{or, equivalently,} \quad \widetilde{A}_{ij} := \begin{cases} A_{ij} & i \in [m],\ j \in [N] \\ \lambda w_j & i = m+1,\ j \in [N] \end{cases}. \tag{36}$$

In addition, given $x \in \mathbb{R}^N$, we consider N augmentations of the residual vector $r = Ax - y \in \mathbb{R}^N$ as the vectors $\widetilde{r}^j \in \mathbb{R}^{m+1}$, defined by

$$\widetilde{r}^j := \begin{bmatrix} r \\ -\lambda w_j x_j \end{bmatrix} \quad \text{or, equivalently,} \quad \widetilde{r}^j_i := \begin{cases} r_i & i \in [m] \\ -\lambda w_j x_j & i = m+1 \end{cases}, \quad \forall j \in [N]. \tag{37}$$

Let $\widetilde{a}_j$ be the jth column of $\widetilde{A}$ and $\tau_j : [\|\widetilde{a}_j\|_0] \to \operatorname{supp}(\widetilde{a}_j)$ be a bijective map defining a nondecreasing rearrangement of the vector

$$\left(\frac{\widetilde{r}^j_i}{\widetilde{A}_{ij}}\right)_{i \in \operatorname{supp}(\widetilde{a}_j)} \in \mathbb{R}^{\|\widetilde{a}_j\|_0},$$

i.e., such that

$$\frac{\widetilde{r}^j_{\tau_j(1)}}{\widetilde{A}_{\tau_j(1),j}} \leq \frac{\widetilde{r}^j_{\tau_j(2)}}{\widetilde{A}_{\tau_j(2),j}} \leq \cdots \leq \frac{\widetilde{r}^j_{\tau_j(\|\widetilde{a}_j\|_0)}}{\widetilde{A}_{\tau_j(\|\widetilde{a}_j\|_0),j}}. \tag{38}$$

We are now in a position to state our result.

Theorem 2.7 (LAD-LASSO-based greedy selection rule)**.** *Let $\lambda \geq 0$, $S \subseteq [N]$, $A \in \mathbb{R}^{m\times N}$ with nonzero columns, and $x \in \mathbb{R}^N$ satisfying*

$$x \in \arg\min_{z\in\mathbb{R}^N} \|y - Az\|_1 \quad s.t. \quad \mathrm{supp}(z) \subseteq S. \tag{39}$$

Then, for every $j \in [N]$,

$$\min_{t\in\mathbb{R}} G^{\mathrm{LAD}}_{\ell^1_w}(x + te_j) = G^{\mathrm{LAD}}_{\ell^1_w}(x) - \Delta^{\mathrm{LAD}}_{\ell^1_w}(x, S, j),$$

where

$$\Delta^{\mathrm{LAD}}_{\ell^1_w}(x, S, j) = \lambda w_j|x_j| + \|r\|_1 - \left\| \widetilde{r}^j - \frac{\widetilde{r}^j_{\hat{i}(j)}}{\widetilde{A}_{\hat{i}(j),j}} \widetilde{a}_j \right\|_1, \tag{40}$$

with $\hat{i}(j) := \tau_j(\hat{k}(j))$ and

$$\hat{k}(j) := \min\left\{ k \in [\|\widetilde{a}_j\|_0] : \sum_{i=1}^{k} \frac{|\widetilde{A}_{\tau_j(i),j}|}{\|\widetilde{a}_j\|_1} \geq \frac{1}{2} \right\}, \tag{41}$$

and where $\widetilde{A}$, $\widetilde{r}^j$, and τ_j are defined as in (36), (37), and (38), respectively.

This proposition leads to the LAD-LASSO-based OMP iteration

$$S^{(k+1)} = S^{(k)} \cup j^{(k)} \quad \text{where} \quad j^{(k)} = \arg\max_j \Delta^{\mathrm{LASSO}}_{\lambda}(x^{(k)}, S^{(k)}, j) \tag{42}$$

$$x^{(k+1)} \in \arg\min_{z\in\mathbb{C}^N} \|y - Az\|_1 \quad \text{s.t} \quad \mathrm{supp}(z) \subseteq S^{(k)}. \tag{43}$$

Some remarks are in order.

Remark 2.8 (On terminology)**.** *The least-squares projection step of LASSO- and SR-LASSO-based OMP ensures orthogonality between the residual and the span of selected columns at each iteration. This property is no longer valid in the LAD-LASSO case because (43) does not define an orthogonal projection. With a slight abuse of terminology, we will refer to the method defined by (42)–(43) as a variant of OMP, despite the lack of orthogonality.*

Remark 2.9 (Solving LAD problems)**.** *Unlike the least-squares projection step of LASSO- and SR-LASSO-based WOMP, the LAD problem (43) does not admit an explicit solution in general. However, one can take advantage of efficient convex optimization algorithms to approximately solve it. We note in passing that, for small values of k, the corresponding LAD problems over $\mathbb{R}^k$ are much cheaper to solve than an ℓ^1 minimization problem over $\mathbb{R}^N$. In this chapter, we numerically solve LAD problems using the MATLAB CVX package [59, 60] with MOSEK solver [90].*

Remark 2.10 (An alternative strategy)**.** *An alternative LAD-LASSO-based OMP iteration can be derived by relaxing the LAD-LASSO to an augmented LASSO or SR-LASSO problem. Notably, this strategy works naturally in the complex case. Recall that our objective is to minimize $G^{\mathrm{LAD}}_{\ell^1_w}$*

defined in (27) *over* $\mathbb{C}^N$. *Now, for any* $z \in \mathbb{C}^N$ *let* $c = y - Az \in \mathbb{C}^m$ *or, equivalently,* $y = Bt$, *where*

$$B := \begin{bmatrix} A & I \end{bmatrix} \in \mathbb{C}^{m\times(N+m)} \quad \textit{and} \quad t := \begin{bmatrix} z \\ c \end{bmatrix} \in \mathbb{C}^{N+m}.$$

With this change of variable, a minimizer $\hat{z}$ *of* $G^{\mathrm{LAD}}_{\ell^1_w}$ *over* $\mathbb{C}^N$ *satisfies*

$$\begin{bmatrix} \hat{z} \\ \hat{c} \end{bmatrix} \in \arg \min_{t\in\mathbb{C}^{N+m}} \|t\|_{1,v} \quad s.t. \quad Bt = y, \quad \textit{where } v = \begin{bmatrix} \lambda w \\ \mathbf{1} \end{bmatrix}$$

for some $\hat{c} \in \mathbb{C}^m$, *and where* $\mathbf{1} \in \mathbb{C}^m$ *is the vector of ones (see* [6, 79] *and references therein). This basis pursuit problem can be relaxed to a quadratically-constrained basis pursuit problem*

$$\min_{t\in\mathbb{C}^{N+m}} \|t\|_{1,v} \quad s.t. \quad \|y - Bt\|_2 \le \eta,$$

with $\eta > 0$. *Now, one could consider either a LASSO or SR-LASSO reformulation of this problem. For example, in the LASSO case one would consider a loss function of the form*

$$G(t) = \|y - Bt\|_2^2 + \mu\|t\|_{1,v}, \quad t \in \mathbb{C}^{m+N} \tag{44}$$

with $\mu > 0$, *which leads to a LASSO-based OMP method. It is worth observing that a possible disadvantage of this strategy is the introduction of an extra tuning parameter* μ.

2.4 Numerical experiments

In this section we present numerical results for the proposed LASSO-based WOMP algorithms. All the numerical experiments were performed in MATLAB 2017b 64-bit on a laptop equipped with a 2.4 GHz Intel Core i5 processor and 8 GB of DDR3 RAM. In some experiments, we compare our proposed algorithms with convex optimization-based recovery strategies. In these cases, we use the MATLAB CVX package [59, 60] with MOSEK solver [90] and set `cvx_precision best`. For the sake of convenience, we sometimes use MATLAB's vector notation. For example, $10.\hat{}(1 : 2 : 5)$ denotes the vector $(10^1, 10^3, 10^5)$. The source code needed to reproduce our numerical experiments can be found on the GitHub repository http://github.com/sina-taheri/Greedy_LASSO_WOMP.

The section is organized as follows. In Section 2.4.1, we start by presenting three settings used to validate and test the proposed algorithms. In Section 2.4.2, we carry out a first set of experiments aimed at studying the effect of the tuning parameter on the recovery error for different levels of noise or corruption, and for different weights' values. Section 2.4.3 is dedicated to investigating the connection between the iteration number of the proposed greedy methods and the recovery error. We conclude by illustrating experiments on algorithms' runtime, loss function decay and a discussion on the stopping criteria in Section 2.4.4.

2.4.1 Description of the numerical settings

The three numerical settings employed in our experiments are illustrated below.

(i) Sparse random Gaussian setting (sparse and unweighted). First, we generate an s-sparse random Gaussian vector $x \in \mathbb{R}^N$ as follows. S, the support of x, is generated by randomly and uniformly drawing a subset of $[N]$ of size s (this avoids repeated indices). Within the support, the entries x are independently sampled from a Gaussian distribution with zero mean and unit variance, i.e., $x_i \sim \mathcal{N}(0,1)$, for every $i \in S$. This vector is measured by a sensing matrix $A \in \mathbb{R}^{m\times N}$ obtained after an ℓ^2-normalization of the columns of a random Gaussian matrix $G \in \mathbb{R}^{m\times N}$ with independent entries $G_{i,j} \sim \mathcal{N}(0,1)$ for every $i \in [m],\ j \in [N]$. The objective is to recover the synthetically-generated signal x from corrupted measurements, i.e.,

$$y = Ax + e^{\text{bounded}} + e^{\text{unbounded}} \in \mathbb{R}^m, \quad \text{where} \quad \|e^{\text{bounded}}\|_2 = \eta,\ \|e^{\text{unbounded}}\|_0 \le K. \tag{45}$$

Here, $e^{\text{bounded}} = \eta e'/\|e'\|_2 \in \mathbb{R}^m$ is a ℓ^2-normalized random Gaussian vector with independent entries, i.e., $e'_i \sim \mathcal{N}(0,1)$ for every $i \in [m]$ and $e^{\text{unbounded}} \in \mathbb{R}^m$ is a K-sparse vector generated by randomly and independently drawing K integers uniformly from $[m]$ and filling the corresponding entries with independent random samples from $\mathcal{N}(0,M^2)$ for some $M > 0$. When we have unbounded noise in our measurements, we choose M to be very large, while in other cases we simply set it to zero. In this setting we consider unweighted recovery, i.e., $w = \mathbf{1}$, the vector of ones.

(ii) Sparse random Gaussian setting with oracle (sparse and weighted). Using the same model as in the previous setting, we acquire noisy measurements $y = Ax + e^{\text{bounded}} + e^{\text{unbounded}} \in \mathbb{R}^m$ of a random s-sparse vector $x \in \mathbb{R}^N$. In this second setting, we assume to have some *a priori* knowledge of the support of x and incorporate this knowledge through weights in order to improve reconstruction. More precisely, we assume to know a set S^{oracle} that partially approximates (i.e., that has nontrivial intersection with) the support of x. Then, we define the weight vector $w \in \mathbb{R}^N$ as

$$w_j := \begin{cases} w_0 & j \in S^{\text{oracle}} \\ 1 & j \notin S^{\text{oracle}} \end{cases}, \tag{46}$$

for a suitable $w_0 \in [0,1]$. Note that if w_0 is chosen to be small, the contribution of signal coefficients weighted by w_0 is attenuated in the LASSO-type loss function. Consequently, activation of the corresponding indices is promoted in the greedy index selection stage of WOMP.

(iii) Function approximation (compressible and weighted). In the third setting, the goal is to approximate a multivariate function

$$f : D \to \mathbb{R}, \quad D = [-1,1]^d,$$

with $d \gg 1$, from pointwise evaluations $f(t_1), f(t_2), \ldots, f(t_m)$, where $t_1, \ldots, t_m$ are independently and identically sampled from a probability distribution ϱ over D. Here we briefly summarize how to perform this task efficiently via compressed sensing and refer the reader to the book [11] for a comprehensive treatment of the topic. This problem is mainly motivated by the study of quantity of interests in parametric models such as parametric differential equations, with applications to uncertainty quantification [106]. Considering a basis of orthogonal polynomials $\{\Psi_\nu\}_{\nu\in\mathbb{N}_0^d}$ for $L^2_\varrho(D)$ (i.e., the Hilbert space of square-integrable functions over D weighted by the probability measure ϱ). We aim to compute an approximation of the form

$$f_\Lambda := \sum_{j\in[N]} x_{\nu_j}\Psi_{\nu_j} \approx f, \quad \text{where } \underbrace{\Lambda := \{\nu_j\}_{j\in[N]}}_{\text{truncation set}} \subset \mathbb{N}_0^d \quad \text{and} \quad N \gg m,$$

and where $x = (x_{\nu_j})_{j\in[N]} \in \mathbb{R}^N$. This can be reformulated as a linear system in the coefficients x, namely,

$$y = Ax + e, \tag{47}$$

where the measurement matrix $A \in \mathbb{R}^{m\times N}$ and the measurement vector $y \in \mathbb{C}^m$ are defined as

$$A_{ij} := \frac{1}{\sqrt{m}}\Psi_{\nu_j}(t_i), \quad y_i := \frac{1}{\sqrt{m}} f(t_i), \quad \forall i \in [m],\ \forall j \in [N],$$

and where $e \in \mathbb{C}^m$ is the noise vector, including the inherent truncation error (depending on Λ) and, possibly, other types of error (e.g., numerical, model, or physical error). Under suitable smoothness conditions on f, such as holomorphy, the vector of coefficients x is approximately sparse or compressible (see [11, Chapter 3]). Therefore, the problem of approximating the function f is recast as finding a compressible solution x to the linear system (47). As a test function, we consider the so-called *iso-exponential* function, defined as

$$f(t) = \exp\left(-\sum_{i=1}^{d} t_i/(2d)\right), \quad \forall t \in D, \tag{48}$$

which can be shown to be well approximated by sparse polynomial expansions (see [11, §A.1.1]). Recovering x using LASSO-based WOMP algorithms, we set weights as

$$w_j := \|\Psi_{\nu_j}\|_{L^\infty(D)} = \sup_{t\in D}|\Psi_{\nu_j}(t)|, \quad \forall j \in [N], \tag{49}$$

known as *intrinsic weights*. Note that these weights admit explicit formulas, e.g., Legendre and Chebyshev orthogonal polynomials (see [11, Remark 2.15]). In this chapter, we will employ just Legendre polynomials.

2.4.2 Recovery error versus tuning parameter

The aim of Experiments I, II, and III presented in this section is to investigate the interplay between the tuning parameter λ and the recovery accuracy of LASSO-type WOMP algorithms in the three settings described in Section 2.4.1, for different noise levels and weight values. We recover x for a range of values of the tuning parameter λ, at a fixed iteration number of the LASSO-type WOMP algorithms. We measure accuracy via the relative ℓ^2-error

$$E_\lambda = \frac{\|\hat{x}_\lambda - x\|_2}{\|x\|_2},$$

where $\hat{x}_\lambda$ denotes the computed approximation to x when the tuning parameter is set to λ. Hence, we plot the recovery error as a function of λ. We repeat this experiment N_{trial} number of times for

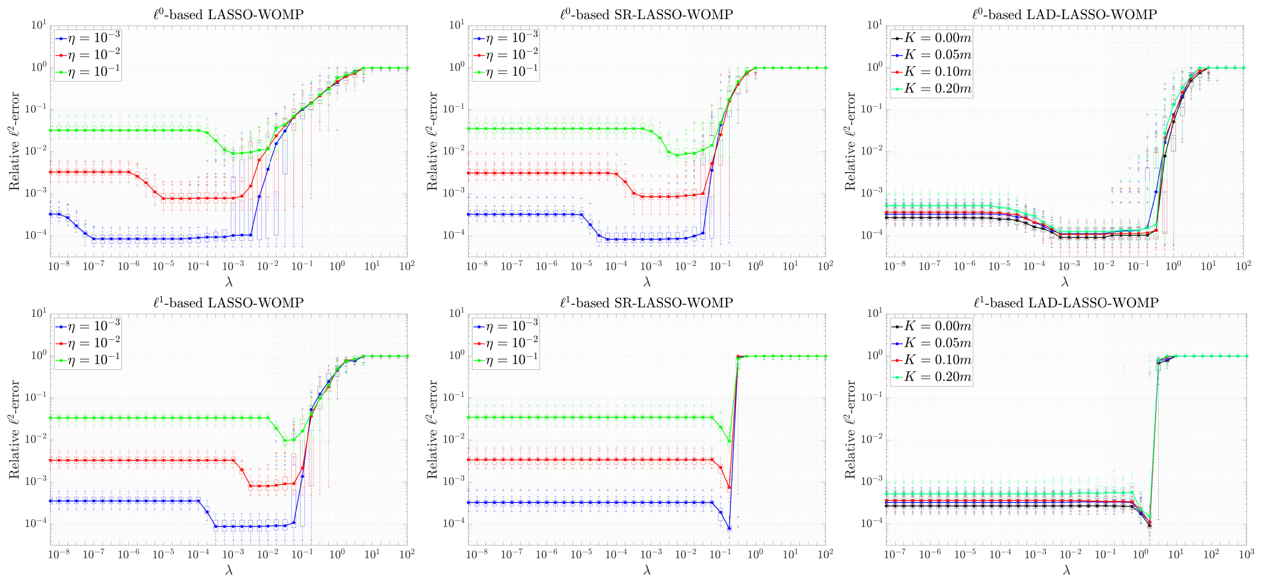

Figure 2.1: Relative error as a function of the tuning parameter (Experiment I, sparse random Gaussian setting). We compare the recovery accuracy of ℓ^0- and ℓ^1-based WOMP algorithms for different noise or corruption levels, as in (45).

different levels of noise and corruptions (in Experiments I and II), or different weight values (in Experiment III). The results of these statistical simulations are visualized using *boxplots*, whose median values are linked by solid curves.

In Experiments I and II we also consider ℓ^0-based variants of LASSO-type WOMP algorithms. The ℓ^0-based variant of LASSO WOMP was proposed in [2] and the greedy selection rules for ℓ^0-based SR- and LAD-LASSO WOMP are derived in Section 2.6. They constitute natural alternatives to the loss functions presented in Section 2.3, and we study their performance to justify our choice of ℓ^1-based loss functions in this chapter.

Experiment I (sparse random Gaussian setting). We begin with the sparse random Gaussian setting. Figure 2.1 shows results for recovery performed via ℓ^0- and ℓ^1-based WOMP algorithms and for measurements corrupted by different levels of noise. In the LASSO and SR-LASSO WOMP cases, we let

$$N = 300,\ m = 150,\ s = 10,\ \eta = \|e^{\text{bounded}}\|_2 \in 10\,\hat{.}\,(-3:-1),\ M = 0.$$

For LAD-LASSO WOMP, we fix

$$N = 300,\ m = 150,\ s = 10,\ \eta = 10^{-3},\ M = 100,\ K \in \{0, 0.05m, 0.1m, 0.2m\}.$$

Both the ℓ^0- and ℓ^1-based algorithms are able to reach a relative ℓ^2-error below the noise level for appropriate choices of the tuning parameter λ. We note that every experiment has optimal values of λ for which the recovery error associated with a certain noise level is minimized. These optimal values are independent of the noise level for ℓ^1-based SR-LASSO and of the corruption level for both ℓ^0- and ℓ^1-based LAD-LASSO WOMP. An analogous phenomenon can be observed for the

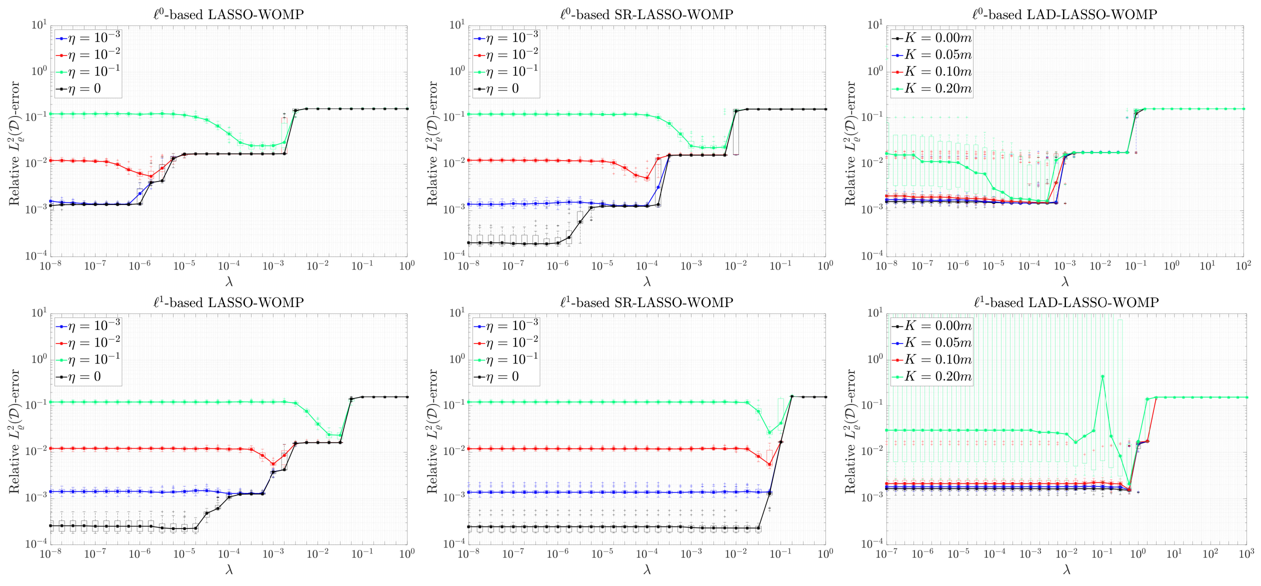

Figure 2.2: Relative error as a function of the tuning parameter (Experiment II, function approximation). We compare the recovery accuracy of ℓ^0- and ℓ^1-based WOMP algorithms for different noise or corruption levels, as in (45).

corresponding ℓ^1 minimization programs [7]. Finally, it is worth noting that the optimal values of λ depend on the noise level for the ℓ^0-based SR-LASSO formulation.

Experiment II (function approximation). Next we consider the high-dimensional function approximation setting. We approximate the high-dimensional function defined in (48) with $d = 5$, where

$$N = |\Lambda| = 426, \ m = 200,$$

and M, η and K as before. Specifically, the truncation set $\Lambda = \Lambda_n^{\text{HC}}$ is a hyperbolic cross of order $n \in \mathbb{N}_0$, defined as

$$\Lambda_n^{\text{HC}} := \left\{ \nu = (\nu_k)_{k=1}^d \in \mathbb{N}_0^d : \prod_{k=1}^d (\nu_k + 1) \le n + 1 \right\}, \tag{50}$$

In this experiment, we let $n = 18$. Note that in the function approximation setting, even when $\eta = 0$, samples are intrinsically corrupted by noise. This is due to the truncation error introduced by Λ (see [11, Chapter 7]). Moreover, we recall that in this experiment we use weights $w \in \mathbb{R}^N$ defined as in (49).

Figure 2.2 shows the results of this experiment. Note that in this setting the relative $L^2_\varrho(D)$-error and the relative ℓ^2-error coincide because of orthonormality of the polynomial basis $\{\Psi_\nu\}_{\nu \in \mathbb{N}_0^d}$. Observations analogous to those made in Experiment I hold in this case as well, with some differences. First, Figure 2.2 shows even more clearly than Figure 2.1 the superiority of the ℓ^1-based SR-LASSO approach with respect to its ℓ^0-based counterpart. From it, we can see that only for ℓ^1-based SR-LASSO WOMP the optimal values of λ are vertically aligned and thus independent of the noise level. Second, when the corruption level is large ($K = 0.2m$), ℓ^0-based LAD-LASSO

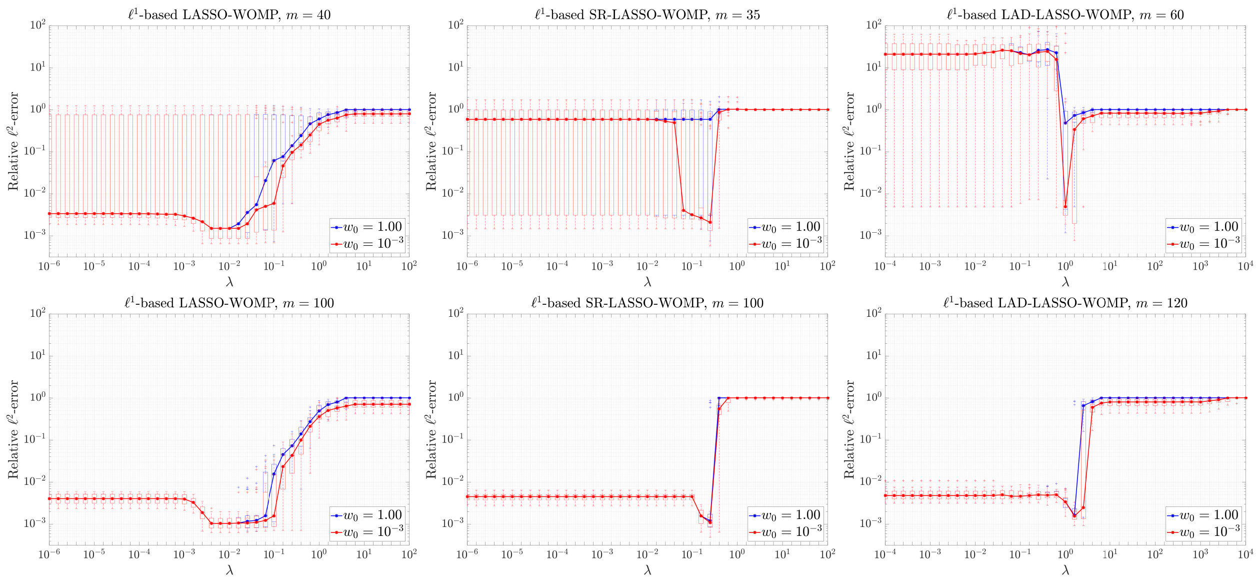

Figure 2.3: Relative error as a function of the tuning parameter (Experiment III, sparse random Gaussian setting with oracle). Different ℓ^1-based WOMP algorithms are tested for a fixed noise level, different choices of weights depending on the parameter w_0 (see (46)), and for low (top row) and high (bottom row) values of m.

WOMP is more robust to the choice of λ than its ℓ^1-based counterpart.

Experiment III (sparse random Gaussian setting with oracle). In the final experiment of this section we consider the sparse random Gaussian setting with oracle, and we illustrate the benefits provided by weights for signal recovery via WOMP. We employ the same parameter settings as Experiment I, with the difference that this time $N = 500$ and we do not consider ℓ^0-based WOMP variants, we fix the noise level, and test different choices of weights. We set the noise level to $\eta = 10^{-3}$ for LASSO and SR-LASSO, and corruptions with $K = 0.1m$ for LAD-LASSO. As mentioned earlier, the prior knowledge from S^{oracle} is incorporated into the weight vector $w \in \mathbb{R}^N$. Here we assume the oracle to have *a priori* knowledge of just half of the support of x. In order to create S^{oracle}, half of the support entries are randomly chosen and are used to generate the weight vector $w \in \mathbb{R}^N$ as in (46) with $w_0 = 10^{-3}$.

The results of this experiments are shown in Figure 2.3. Recovery is performed for different numbers of measurements, namely, $m = 40, 100$ for LASSO, $m = 35, 100$ for SR-LASSO WOMP and $m = 60, 120$ for LAD-LASSO WOMP. We observe that weights are able to improve reconstruction in all settings. This phenomenon has been previously known in the literature (see, e.g., [14, 56]), and in this experiment is particularly evident in the SR-LASSO and LAD-LASSO cases (second and third column in Figure 2.3).

2.4.3 Recovery error versus iteration number

In the last two experiments of this subsection (IV and V), we study the recovery error as a function of the number of iterations of the proposed greedy algorithms. This will highlight the benefits due

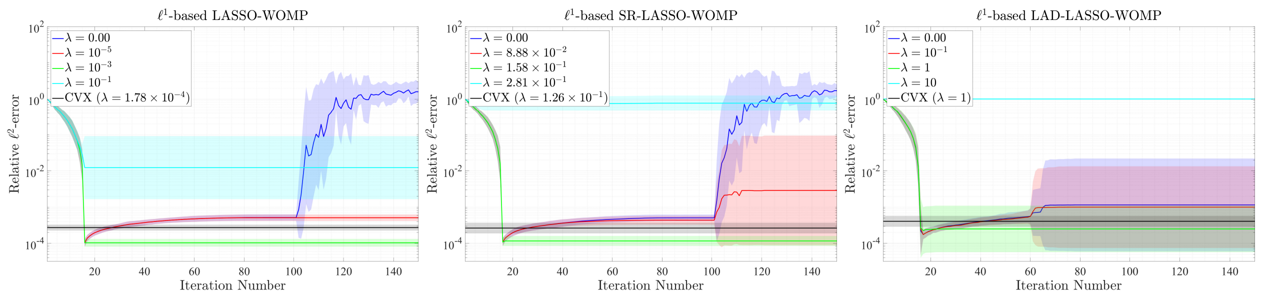

Figure 2.4: Relative error as a function of the iteration number (Experiment IV, sparse random Gaussian setting). The proposed ℓ^1-based WOMP formulations are tested for different values of the tuning parameter λ. The black curve corresponds to recovery via convex optimization of the corresponding loss function.

to the presence of a regularization term in the loss function. We compute the relative ℓ^2-error at iteration k and for a specific value of λ as

$$E_\lambda^{(k)} = \frac{\|\hat{x}^{(k)} - x\|_2}{\|x\|_2}, \quad k \in [N_{\text{iter}}],\ \lambda \in \mathcal{L},$$

where N_{iter} is the maximum number of iterations and $\mathcal{L}$ is a suitable set of tuning parameters. We repeat the above process for N_{trial} random trials. The setup for Experiments IV and V is detailed below.

Experiment IV (sparse random Gaussian setting). For LASSO and SR-LASSO WOMP, we fix

$$N = 200,\ m = 100,\ s = 15,\ N_{\text{iter}} = 150,\ \eta = 10^{-3},\ M = 0.$$

For LAD-LASSO WOMP, we let

$$N = 200,\ m = 100,\ s = 15,\ N_{\text{iter}} = 150,\ \eta = 10^{-3},\ M = 100,\ K = 0.05m.$$

Experiment V (function approximation). We choose

$$d = 10,\ n = 8,\ N = |\Lambda| = 471,\ m = 200,\ N_{\text{iter}} = 250$$

where n is the order of hyperbolic cross set defined in (50), and M, η and K as in Experiment III.

Figure 2.4 and Figure 2.5 show the relative recovery ℓ^2-error of ℓ^1-based WOMP algorithm for Experiments IV and V, respectively. For better visualization, we use *shaded plots*. The solid curves represent the mean relative error as a function of iteration number. The upper and lower boundaries of the shaded areas are designated by plotting the discrete points $(k, 10^{\mu_\lambda^k + \sigma_\lambda^k})$ and $(k, 10^{\mu_\lambda^k - \sigma_\lambda^k})$, $k \in [N_{\text{iter}}]$, $\lambda \in \mathcal{L}$. Here μ_λ^k and σ_λ^k denote, respectively, the sample mean and the

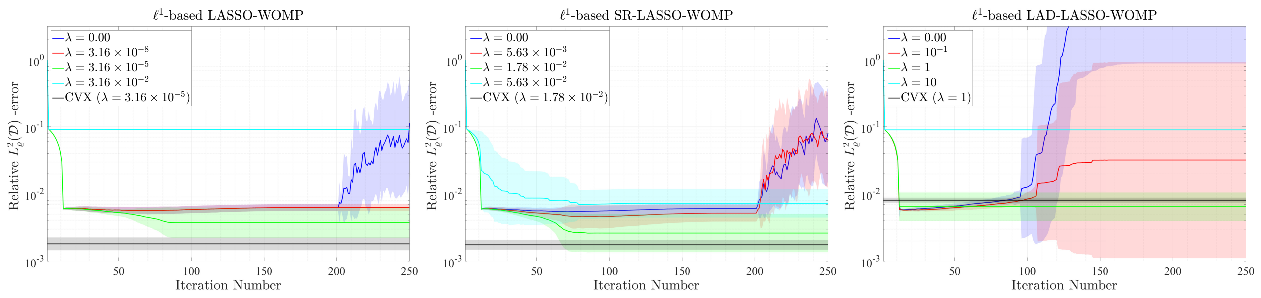

Figure 2.5: Relative error as a function of the iteration number (Experiment V, function approximation). The proposed ℓ^1-based WOMP formulations are tested for different values of the tuning parameter λ. The black curve corresponds to recovery via convex optimization of the corresponding loss function.

sample standard deviation of the $\log_{10}$-transformed relative ℓ^2-error at iteration k, i.e.,

$$\mu_\lambda^k = \frac{1}{N_{\text{trial}}} \sum_{i=1}^{N_{\text{trial}}} \log((E_\lambda^{(k)})_i) \quad \text{and} \quad \sigma_\lambda^k = \sqrt{\frac{1}{N_{\text{trial}} - 1} \sum_{i=1}^{N_{\text{trial}}} \Big(\log((E_\lambda^{(k)})_i) - (\mu_\lambda^k)_i\Big)^2}.$$

(See also [11, §A.1.3] for more details.) The set $\mathcal{L}$ of tuning parameters always consists of $\lambda = 0$ (in blue), as well as the best empirical λ (in green), an underestimated λ (in red), and an overestimated one (in cyan). By the best empirical λ, we mean the λ that achieves the smallest empirical relative error $E_\lambda^{(k)}$, over a wide range of explored values and on average for N_{trial} random trials. Moreover, we compare each ℓ^1-based WOMP formulation, with the corresponding convex optimization problem, with optimally tuned λ (in black). This experiment confirms once again that when λ is tuned appropriately, ℓ^1-based WOMP algorithms can effectively perform sparse recovery from compressive measurements. In particular, for suitably chosen values of λ, WOMP is robust with respect to the iteration number. On this note, we observe that standard OMP (corresponding to $\lambda = 0$ for LASSO and SR-LASSO) begins to severely overfit when the iteration number is larger than m. The reason behind this phenomenon is that in standard OMP there is no regularization mechanism that prevents the greedy selection from adding more indices to the support than number of measurements. Therefore, after m iterations the least-squares fitting in standard OMP leads to severe overfitting and the algorithm starts diverging. A similar phenomenon is observed in [2] for ℓ^0-based LASSO WOMP.

2.4.4 Runtime, loss reduction and stopping criterion

In this subsection we further demonstrate the robustness of the proposed algorithms with respect to the number of iterations. To do so, we consider additional numerical experiments aimed at illustrating the benefits of regularization for the computational efficiency of the algorithms and their stopping criteria. As already mentioned, this robustness is due to the fact that the support stops increasing once the sparsity of the reconstructed signal reaches a saturation point. This removes the need to perform a data-fitting step in the subsequent iterations, which, in turn, leads to significant runtime savings. In addition, this feature allows one to consider a reliable stopping

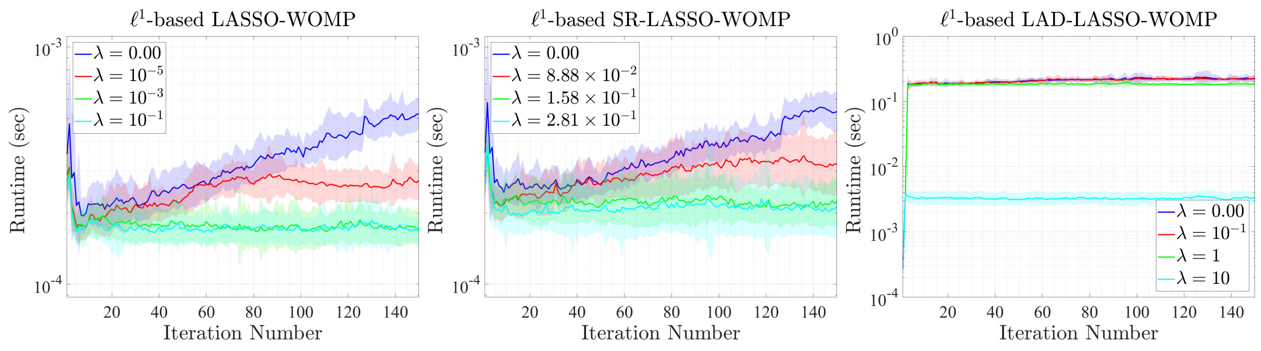

Figure 2.6: Algorithms' runtime as a function of the iteration number (same data as Experiment VI, sparse random Gaussian setting). The proposed ℓ^1-based WOMP formulations are tested for different values of the tuning parameter λ. Colors are consistent with the ones in Figure 2.4.

criterion, i.e., halting the algorithm when the loss function reaches a steady state, or equivalently when the greedy selection leads to an index that already belongs to the current support.

Experiment VI (runtime). Thanks to the mechanism discussed above, the increase in runtime overhead of $*$-LASSO WOMP algorithms' iterations becomes negligible after a certain point. To show this, we consider the same numerical setting as in Experiment IV for sparse random Gaussian signals, but this time we measure algorithms' runtime as a function of the iteration number. Figure 2.6 clearly illustrates that for an appropriate choice of λ, WOMP imposes no significant increase in runtime overhead after a certain number of iteration corresponding to the signal's sparsity. This is not the case for standard OMP (associated with $\lambda = 0$ in LASSO and SR-LASSO WOMP) as it has to solve increasingly large least-squares problems as the iterations proceed. A similar phenomenon is also observed in the context of function approximation. In higher dimensions (greater values of m), this advantage becomes even more pronounced, as solving the least-squares becomes more computationally expensive (for the sake of conciseness, we omit these plots from the chapter).

Next we revisit a point mentioned in passing in Remark 2.9. As mentioned, with N large and s small, reducing the dimensionality of a high-dimensional problem over $\mathbb{R}^{m\times N}$ into smaller consecutive problems over $\mathbb{R}^{m\times k}$, where k is the iteration number, can offer computational advantages. This phenomenon is well captured by Figure 2.7, where we plot the runtime of CVX and LAD-LASSO WOMP as a function of the solution sparsity, as well as a "dartboard" plot of relative ℓ^2-error with respect to runtime of these methods for different values of sparsity. For an ambient space dimension of $N = 4000$ and $m = 120$ measurements, we plot the runtime as a function of s over 15 trials for the left and 5 trials for the right figure. For LAD-LASSO WOMP, we always fix the number of iterations to $2s$. These plots vividly reveal the existence of a phase transition. For small enough values of s, running LAD-LASSO-WOMP is cheaper than solving a LAD-LASSO problem with CVX. After a critical value of s (in this case, $s = 10$), CVX converges faster than LAD-LASSO WOMP. Nevertheless, it is worth noting that CVX is unable to attain a reliable solution for $s \geq 14$ (the relative error is close to 1), whereas LAD-LASSO WOMP is able to compute a more accurate solution for these values of s.

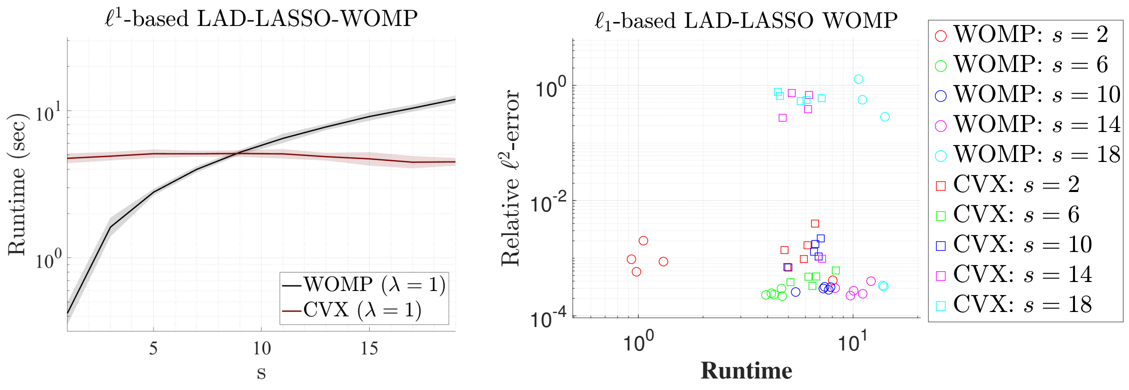

Figure 2.7: On the left: Runtime of LAD-LASSO WOMP and CVX as a function of solution sparsity s. On the right: Relative ℓ^2-error and runtime of LAD-LASSO and CVX for different values of sparsity. Both plots are generated using the same numerical settings.

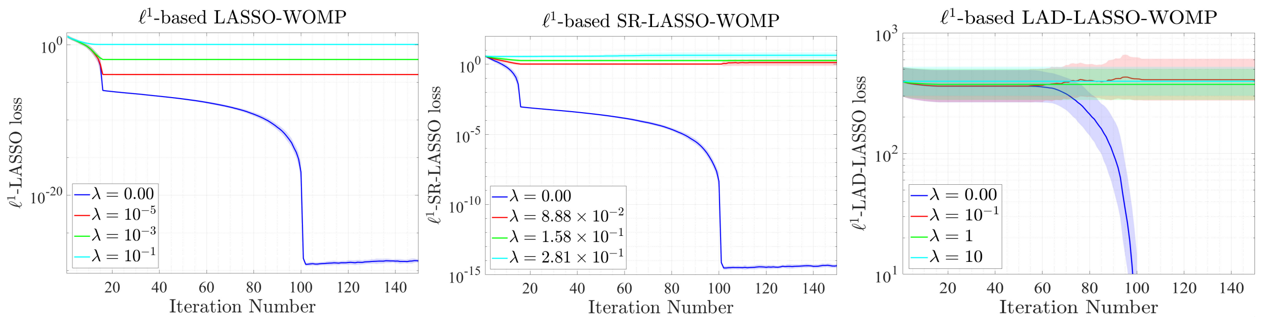

Figure 2.8: Loss value of algorithms as a function of the iteration number (same data as Experiment VI, sparse random Gaussian setting). The proposed ℓ^1-based WOMP formulations are tested for different values of the tuning parameter λ. Colors are consistent with the ones in Figure 2.8.

Experiment VII (loss reduction). In Figure 2.8 we plot the loss value, i.e., values of Equations (25) to (27), respectively, against the iteration number. We note again, in accordance with the results of Experiment VI, that for an appropriate choice of the tuning parameter, regularization also stabilizes the loss value after a certain iteration number.

Discussion on stopping criteria. The results of the previous experiments suggest that thanks to regularization of the loss functions in $*$-LASSO WOMP algorithms, there are several options for the stopping criterion. In case an estimate of the sparsity s of the signal is available, one can run a number $K > s$ iterations of $*$-LASSO WOMP and still ensure that for an appropriate choice of λ the algorithm does not overfit, in light of Figure 2.4 and Figure 2.5 (note also that running more iterations than s would not imposes too much runtime overhead in light of Figure 2.6). If the signal's sparsity cannot be estimated in advance, one can monitor the loss value over iterations and halt the algorithm when the loss reaches a steady state, although this might be challenging when the loss reduction is very small. Finally, one can halt the algorithm when the greedy selection leads to an index that already belongs to the current support. Based on our experience, the latter seems to be the most reliable option.

2.5 Proof of the main results

In this section we prove Theorem 2.5, Theorem 2.6 and Theorem 2.7.

2.5.1 Proof of Theorem 2.5

Let $G = G_{\ell^1_w}$, the weighted LASSO loss function defined in (25). The argument is organized into two cases: $j \notin S$ and $j \in S$.

Case 1: $j \notin S$**.** In this case, $j \notin \operatorname{supp}(x) \subseteq S$ and, recalling that the columns of A are ℓ^2-normalized, for any $t \in \mathbb{R}$, we can write

$$\begin{aligned} G(x + te_j) &= \|y - A(x + te_j)\|_2^2 + \lambda\|x + te_j\|_{1,w} \\ &= \|y - Ax\|_2^2 + |t|^2 - 2\operatorname{Re}(\bar{t}\langle y - Ax, Ae_j\rangle) + \lambda(\|x\|_{1,w} + |t|w_j). \end{aligned}$$

Our goal is to minimize the above quantity over $t \in \mathbb{C}$. For any $t \in \mathbb{C}$ we let $t = \rho e^{i\theta}$, with $\rho \geq 0$ and $0 \leq \theta < 2\pi$. Using the expression above we obtain

$$\begin{aligned} G(x + te_j) &= \|y - Ax\|_2^2 + \rho^2 - 2\operatorname{Re}(\rho e^{-i\theta}(A^*(y - Ax))_j) + \lambda\|x\|_{1,w} + \lambda\rho w_j \\ &\geq \|y - Ax\|_2^2 + \rho^2 - 2\rho|(A^*(y - Ax))_j| + \lambda\|x\|_{1,w} + \lambda\rho w_j, \end{aligned} \tag{51}$$

where the inequality holds as an equality for some $0 \leq \theta < 2\pi$ depending on the phase of $(A^*(y - Ax))_j$ and due to Euler's formula. An explicit computation shows that (51) is minimized at $\rho = |(A^*(y - Ax))_j| - \frac{\lambda}{2}w_j$, if $|(A^*(y - Ax))_j| \geq \frac{\lambda}{2}w_j$, and at $\rho = 0$ otherwise. Plugging this value into (51) we obtain

$$\min_{t \in \mathbb{C}} G(x + te_j) = G(x) - \max\left\{|(A^*(y - Ax))_j| - \frac{\lambda}{2}w_j, 0\right\}^2,$$

as desired.

Case 2: $j \in S$**.** Letting $r = y - Ax$, we see that $(A^*r)_j = 0$ since, by assumption, Ax is the orthogonal projection of y onto the span of the columns of A indexed by S. Thus we can write

$$\begin{aligned} G(x + te_j) &= \|y - A(x + te_j)\|_2^2 + \lambda\|x + te_j\|_{1,w} \\ &= \|y - A(x + te_j)\|_2^2 + \lambda\|x - x_je_j\|_{1,w} + \lambda|x_j + t|w_j \\ &= \|r\|_2^2 + \underbrace{|t|^2 + \lambda w_j|x_j + t|}_{=:l(t)} + \lambda\|x - x_je_j\|_{1,w}. \end{aligned} \tag{52}$$

Now, we want to minimize $l(t)$ over $t \in \mathbb{C}$. Let $\rho = |t|$. By the triangle inequality we have

$$l(t) = |t|^2 + \lambda w_j|x_j + t| \geq |t|^2 + \lambda w_j||x_j| - |t|| = \rho^2 + \lambda w_j||x_j| - \rho| =: \hat{l}(\rho),$$

where the first inequality holds as an equality only if $t = \alpha x_j$ for some $\alpha \in \mathbb{R}$ with $\alpha \leq 0$. Therefore, $\min_{t \in \mathbb{C}} l(t) = \min_{\rho \in [0,+\infty)} \hat{l}(\rho)$ (since given a minimizer $\hat{\rho}$ of $\hat{l}$, then $\hat{t} = -\hat{\rho}x_j/|x_j|$ is a minimizer of l) and it is sufficient to minimize $\hat{l}$. If $\rho \geq |x_j|$, then $\hat{l}(\rho) = \rho^2 + \lambda w_j \rho - \lambda w_j |x_j|$, which is minimized at $\rho = |x_j|$. Otherwise, if $0 \leq \rho \leq |x_j|$, we have $\hat{l}(\rho) = \rho^2 - \lambda w_j \rho + \lambda w_j |x_j|$. In this case, a direct computation shows that $\hat{l}(\rho)$ is minimized at $\rho = \lambda w_j/2$ if $\lambda w_j/2 < |x_j|$, or at $\rho = |x_j|$ otherwise. Summarizing the above discussion, we have

$$\min_{t \in \mathbb{C}} l(t) = \min_{\rho \in [0,+\infty)} \hat{l}(\rho) = \min \left\{ \hat{l}\left(|x_j|\right), \hat{l}\left(\frac{\lambda w_j}{2}\right) \right\} = \min \left\{ |x_j|^2, \left(\frac{\lambda w_j}{2}\right)^2 + \lambda w_j \left| \frac{\lambda w_j}{2} - |x_j| \right| \right\}.$$

Therefore, recalling (52), we see that

$$\begin{aligned}
\min_{t \in \mathbb{C}} G(x + te_j) &= \|r\|_2^2 + \lambda \|x - x_j e_j\|_{1,w} + \min \left\{ |x_j|^2, \left(\frac{\lambda w_j}{2}\right)^2 + \lambda w_j \left| \frac{\lambda w_j}{2} - |x_j| \right| \right\} \\
&= \|r\|_2^2 + \lambda \|x - x_j e_j\|_{1,w} + \lambda w_j |x_j| - \lambda w_j |x_j| + \min \left\{ |x_j|^2, \left(\frac{\lambda w_j}{2}\right)^2 + \lambda w_j \left| \frac{\lambda w_j}{2} - |x_j| \right| \right\} \\
&= G(x) + \min \left\{ |x_j|^2 - \lambda w_j |x_j|, \left(\frac{\lambda w_j}{2}\right)^2 + \lambda w_j \left| \frac{\lambda w_j}{2} - |x_j| \right| - \lambda w_j |x_j| \right\} \\
&= G(x) - \max \left\{ -|x_j|^2 + \lambda w_j |x_j|, -\left(\frac{\lambda w_j}{2}\right)^2 - \lambda w_j \left| |x_j| - \frac{\lambda w_j}{2} \right| + \lambda w_j |x_j| \right\},
\end{aligned}$$

which concludes the proof.

□

2.5.2 Proof of Theorem 2.6

Let $G = G^{\mathrm{SR}}_{\ell^1_w}$ be the weighted SR-LASSO loss function defined in (26) and recall that $r = y - Ax$. The proof strategy is analogous to that of Theorem 2.5 and is organized into two cases.

Case 1: $j \notin S$**.** Letting $t = \rho e^{i\theta} \in \mathbb{C}$, we have

$$\begin{aligned}
G(x + te_j) &= \|r - ta_j\|_2 + \lambda \|x\|_{1,w} + \lambda \rho w_j \\
&= \sqrt{\|r\|_2^2 + \rho^2 - 2\rho \operatorname{Re}(e^{-i\theta} \langle r, a_j \rangle)} + \lambda \|x\|_{1,w} + \lambda \rho w_j \\
&\geq \underbrace{\sqrt{\|r\|_2^2 + \rho^2 - 2\rho |\langle r, a_j \rangle|} + \lambda \|x\|_{1,w} + \lambda \rho w_j}_{=: h(\rho)},
\end{aligned} \tag{53}$$

where the last inequality holds as an equality for some $0 \leq \theta < 2\pi$. In order to minimize the right-hand side, we compute

$$h'(\rho) = \frac{\rho - |\langle r, a_j \rangle|}{\sqrt{\|r\|_2^2 + \rho^2 - 2\rho |\langle r, a_j \rangle|}} + \lambda w_j.$$

If $w_j\lambda \geq 1$, the equation $h'(\rho) = 0$ does not have any solution over $[0, +\infty)$ (since $|\langle r, a_j\rangle| \leq \|r\|_2$ due to the Cauchy-Schwarz inequality). In particular, $h'(\rho) > 0$ for every $\rho \geq 0$. Hence, in that case h is minimized at $\rho = 0$. On the other hand, if $w_j\lambda < 1$, the equation $h'(\rho) = 0$ has the unique solution

$$\rho = |\langle r, a_j\rangle| - \sqrt{\frac{(\lambda w_j)^2(\|r\|_2^2 - |\langle r, a_j\rangle|^2)}{1-(\lambda w_j)^2}}.$$

In addition, h' is nonpositive and nonnegative on the left and right side of this point, respectively. Therefore, the minimizer of h on $[0, +\infty)$ is

$$\widetilde{\rho} = \begin{cases} 0 & \lambda w_j \geq 1 \\ |\langle r, a_j\rangle| - \sqrt{\frac{(\lambda w_j)^2(\|r\|_2^2 - |\langle r, a_j\rangle|^2)}{1-(\lambda w_j)^2}} & \lambda w_j < 1 \end{cases}.$$

Hence,

$$\begin{aligned}
\min_{t\in\mathbb{C}} G(x + te_j) &= \min_{\rho\geq 0} h(\rho) = h(\widetilde{\rho}) \\
&= \begin{cases} \sqrt{(1-(\lambda w_j)^2)(\|r\|_2^2 - |\langle r, a_j\rangle|^2)} + \lambda w_j|\langle r, a_j\rangle| + \lambda\|x\|_{1,w} & \lambda w_j < 1 \\ \|r\|_2 + \lambda\|x\|_{1,w} & \lambda w_j \geq 1 \end{cases} \\
&= \|r\|_2 + \lambda\|x\|_{1,w} + \begin{cases} \sqrt{(1-(\lambda w_j)^2)(\|r\|_2^2 - |\langle r, a_j\rangle|^2)} + \lambda w_j|\langle r, a_j\rangle| - \|r\|_2 & \lambda w_j < 1 \\ 0 & \lambda w_j \geq 1 \end{cases},
\end{aligned}$$

which concludes the case $j \notin S$.

Case 2: $j \in S$**.** In this situation, $|(A^*r)_j| = 0$ since x solves a least-squares problem. Thus we can write

$$\begin{aligned}
G(x + te_j) &= \|y - A(x + te_j)\|_2 + \lambda\|x + te_j\|_{1,w} \\
&= \sqrt{\|y - A(x + te_j)\|_2^2} + \lambda \sum_{i\in S\setminus\{j\}} w_i|x_i| + \lambda|x_j + t|w_j \\
&= \underbrace{\sqrt{|t|^2 + \|r\|_2^2} + \lambda w_j|x_j + t|}_{=:l(t)} + \lambda\|x - x_je_j\|_{1,w}.
\end{aligned}$$

We continue by minimizing l over $\mathbb{C}$. Letting $t = \rho e^{i\theta}$, by the triangle inequality we have

$$l(t) = \sqrt{|t|^2 + \|r\|_2^2} + \lambda w_j|x_j + t| \geq \sqrt{\rho^2 + \|r\|_2^2} + \lambda w_j||x_j| - \rho| =: \hat{l}(\rho),$$

where the inequality holds as an equality when $t = \alpha x_j$ for some $\alpha \in \mathbb{R}$ with $\alpha \leq 0$. If $\rho \geq |x_j|$ we have $\hat{l}(\rho) = \sqrt{\rho^2 + \|r\|_2^2} + \lambda w_j\rho - \lambda w_j|x_j|$, which is minimized at $\rho = |x_j|$. Otherwise, if $0 \leq \rho < |x_j|$, we have $\hat{l}(\rho) = \sqrt{\rho^2 + \|r\|_2^2} - \lambda w_j\rho + \lambda w_j|x_j|$, and a direct computation shows that, when $\lambda w_j \geq 1$,

$$\hat{l}'(\rho) = \frac{\rho}{\sqrt{\rho^2 + \|r\|_2^2}} - \lambda w_j \leq 1 - 1 = 0,$$

and the equation $\hat{l}'(\rho) = 0$ is either solved for all $0 \leq \rho \leq |x_j|$ (if $r = 0$ and $\lambda w_j = 1$) or for no values of $0 \leq \rho \leq |x_j|$ (otherwise). Hence, when $\lambda w_j \geq 1$, $\rho = |x_j|$ is a minimizer of $\hat{l}$ over $[0, |x_j|]$. Conversely, when $\lambda w_j < 1$, the equation $\hat{l}'(\rho) = 0$ is uniquely solved by

$$\rho = \frac{\lambda w_j \|r\|_2}{\sqrt{1 - (\lambda w_j)^2}}.$$

Hence, in summary, $\hat{l}$ is minimized at

$$\widetilde{\rho} := \begin{cases} |x_j| & \lambda w_j \geq 1 \\ \min\left\{|x_j|, \frac{\lambda w_j \|r\|_2}{\sqrt{1-(\lambda w_j)^2}}\right\} & \lambda w_j < 1 \end{cases}.$$

This leads to

$$\begin{aligned}\min_{t \in \mathbb{C}} G(x + te_j) &= \min_{t \in \mathbb{C}} l(t) + \lambda \|x - x_j e_j\|_{1,w} \\ &= \hat{l}(\widetilde{\rho}) + \|r\|_2 + \lambda \|x\|_{1,w} - \|r\|_2 - \lambda w_j |x_j| \\ &= G(x) - \left(\|r\|_2 + \lambda w_j |x_j| - \hat{l}(\widetilde{\rho})\right),\end{aligned}$$

which concludes the proof. □

2.5.3 Proof of Theorem 2.7

In order to prove Theorem 2.7, we need the minimum for the LAD problem in the one-dimensional setting. To this purpose, we present the following lemma based on the arguments from [20, Lemmas 1 & 2], that we include here for the sake of completeness.

Lemma 2.11 (Explicit solution of univariate LAD)**.** *Let $y, a \in \mathbb{R}^N$ and $L : \mathbb{R} \to [0, +\infty)$, defined by*

$$L(t) := \|y - ta\|_1 = \sum_{i=1}^{N} |y_i - ta_i|, \quad \forall t \in \mathbb{R}.$$

Then a minimizer of L over $\mathbb{R}$ is

$$t^* = \left(\frac{y_{\hat{i}}}{a_{\hat{i}}}\right),$$

with $\hat{i} = \tau(\hat{k})$ where

$$\hat{k} = \min\left\{k \in [\|a\|_0] : \sum_{i=1}^{k} \frac{|a_{\tau(i)}|}{\|a\|_1} \geq \frac{1}{2}\right\},$$

and where $\tau : [\|a\|_0] \to \operatorname{supp}(a)$ is a bijective map defining a nondeacreasing rearrangement of *the vector*

$$\left(\frac{y_i}{a_i}\right)_{i\in\operatorname{supp}(a)} \in \mathbb{R}^{\|a\|_0},$$

i.e., such that

$$\frac{y_{\tau(1)}}{a_{\tau(1)}} \leq \frac{y_{\tau(2)}}{a_{\tau(2)}} \leq \cdots \leq \frac{y_{\tau(\|a\|_0)}}{a_{\tau(\|a\|_0)}}.$$

Proof. We define $t_k := y_{\tau(k)}/a_{\tau(k)}$ and the open intervals $I_0 := (-\infty, t_1),\ I_k := (t_k, t_{k+1})$ for $k \in [\|a\|_0 - 1]$, and $I_{\|a\|_0} := (t_{\|a\|_0}, +\infty)$ (if $t_k = t_{k+1}$, we simply set $I_k = \emptyset$).

Now, let $t \in I_k^j$ for some $k \in [\|\widetilde{a}_j\|_0]$. Then,

$$\begin{aligned} L(t) &= \sum_{i\notin\operatorname{supp}(a)} |y_i| + \sum_{i=1}^{\|a\|_0} |a_{\tau(i)}||t_i - t| \\ &= \sum_{i\notin\operatorname{supp}(a)} |y_i| + \sum_{i=1}^{k} |a_{\tau(i)}|(t - t_i) - \sum_{i=k+1}^{\|a\|_0} |a_{\tau(i)}|(t - t_i). \end{aligned}$$

Differentiating with respect to t, we obtain

$$L'(t) = \sum_{i=1}^{k} |a_{\tau(i)}| - \sum_{i=k+1}^{\|a\|_0} |a_{\tau(i)}| =: d_k, \quad \forall t \in I_k.$$

Hence we see that, for any $k \in [\|a\|_0 - 1]$,

$$d_{k+1} = \sum_{i=1}^{k+1} |a_{\tau(i)}| - \sum_{i=k+2}^{\|a\|_0} |a_{\tau(i)}| = \sum_{i=1}^{k} |a_{\tau(i)}| - \sum_{i=k+1}^{\|a\|_0} |a_{\tau(i)}| + 2|a_{\tau(k+1)}| = d_k + 2|a_{\tau(k+1)}| > d_k,$$

implying that $L'(t)$ is increasing with respect to t (wherever it is well defined). In summary, L is a positive and piecewise linear function with increasing derivative. Hence,

$$\min_{t\in\mathbb{R}} h(t) = h\left(t_{\hat{k}}\right),$$

where

$$\begin{aligned} \hat{k} &:= \min\left\{k \in [\|a\|_0] : d_k \geq 0\right\} && (54) \\ &= \min\left\{k \in [\|a\|_0] : \sum_{i=1}^{k} |a_{\tau(i)}| \geq \sum_{i=k+1}^{\|a\|_0} |a_{\tau(i)}|\right\} && (55) \\ &= \min\left\{k \in [\|a\|_0] : 2\sum_{i=1}^{k} |a_{\tau(i)}| \geq \sum_{i=1}^{\|a\|_0} |a_{\tau(i)}|\right\} && (56) \\ &= \min\left\{k \in [\|a\|_0] : \sum_{i=1}^{k} \frac{|a_{\tau(i)}|}{\|a\|_1} \geq \frac{1}{2}\right\}. && (57) \end{aligned}$$

(Note that $\|a\|_1 \neq 0$ since we are assuming a to be nonzero.) Finally, we let $\hat{i} := \tau(\hat{k})$, which concludes the proof. $\square$

We are now in a position to prove Theorem 2.7.

Proof of Theorem 2.7. We let $G = G^{\mathrm{LAD}}_{\ell^1_w}$, the weighted LAD-LASSO objective defined in (27). The proof structure is analogous to that of Theorem 2.5 and Theorem 2.6.

Case 1: $j \notin S$**.** We start by observing that

$$G(x+te_j) = \|y - A(x+te_j)\|_1 + \lambda\|x+te_j\|_{1,w} = \underbrace{\|y - A(x+te_j)\|_1 + \lambda w_j|t|}_{=:h(t)} + \lambda\|x\|_{1,w}.$$

We continue by minimizing $h(t)$ over $t \in \mathbb{R}$. Let $A_{ij},\ i \in [m],\ j \in [N]$ be the entries of the matrix A and recall that $\widetilde{A} \in \mathbb{R}^{(m+1)\times N}$ and $\widetilde{r}^j \in \mathbb{R}^{m+1}$ are augmented versions of A and r, defined by (36) and (37), respectively. Moreover, let $\widetilde{a}_j \in \mathbb{R}^{m+1}$ be the jth column of $\widetilde{A}$. Then we see that

$$h(t) = \sum_{i=1}^{m} |r_i - tA_{ij}| + \lambda w_j|t| = \sum_{i=1}^{m+1} \left|\widetilde{r}^j_i - t\widetilde{A}_{ij}\right|.$$

Thanks to Lemma 2.11,

$$\min_{t\in\mathbb{R}} h(t) = h\left(\frac{\widetilde{r}^j_{\hat{i}(j)}}{\widetilde{A}_{\hat{i}(j),j}}\right),$$

with $\hat{i}(j) = \tau_j(\hat{k}(j))$, where $\hat{k}(j)$ is defined as in (41). Hence, we compute

$$\begin{aligned}
\min_{t\in\mathbb{R}} G(x+te_j) &= \min_{t\in\mathbb{R}} h(t) + \lambda\|x\|_{1,w}\\
&= \left\|\widetilde{r}^j - \frac{\widetilde{r}^j_{\hat{i}(j)}}{\widetilde{A}_{\hat{i}(j),j}}\widetilde{a}_j\right\|_1 + \lambda\|x\|_{1,w} + \|r\|_1 - \|r\|_1\\
&= G(x) - \left(\|r\|_1 - \left\|\widetilde{r}^j - \frac{\widetilde{r}^j_{\hat{i}(j)}}{\widetilde{A}_{\hat{i}(j),j}}\widetilde{a}_j\right\|_1\right),
\end{aligned}$$

which concludes the case $j \notin S$.

Case 2: $j \in S$**.** The proof is similar to Case 1. We start by writing

$$G(x+te_j) = \|y - A(x+te_j)\|_1 + \lambda\|x+te_j\|_{1,w} = \underbrace{\|r - ta_j\|_1 + \lambda w_j|x_j+t|}_{=:l(t)} + \lambda\|x - x_je_j\|_{1,w},$$

where a_j denotes the jth column of A. We want to minimize $l(t)$ over $t \in \mathbb{R}$. Hence, we compute

$$l(t) = \sum_{i=1}^{m} |r_i - tA_{ij}| + \lambda w_j|x_j+t| = \sum_{i=1}^{m+1} \left|\widetilde{r}^j_i - t\widetilde{A}_{ij}\right|$$

and, thanks to Lemma 2.11, $l(t)$ is minimized at $t = \widetilde{r}^j_{\hat{i}(j)} / \widetilde{A}_{\hat{i}(j),j}$, where $\hat{i}(j) = \tau(\hat{k}(j))$ and $\hat{k}(j)$ is defined as in (41). Therefore,

$$\begin{aligned}
\min_{t\in\mathbb{R}} G(x+te_j) &= \min_{t\in\mathbb{R}} l(t) + \lambda\|x - x_j e_j\|_{1,w} \\
&= \left\| \widetilde{r}^j - \frac{\widetilde{r}^j_{\hat{i}(j)}}{\widetilde{A}_{\hat{i}(j),j}} \widetilde{a}_j \right\|_1 + \lambda\|x - x_j e_j\|_{1,w} \\
&= \left\| \widetilde{r}^j - \frac{\widetilde{r}^j_{\hat{i}(j)}}{\widetilde{A}_{\hat{i}(j),j}} \widetilde{a}_j \right\|_1 + \lambda\|x - x_j e_j\|_{1,w} + \lambda w_j |x_j| - \lambda w_j |x_j| + \|r\|_1 - \|r\|_1 \\
&= G(x) - \left(\|r\|_1 + \lambda w_j |x_j| - \left\| \widetilde{r}^j - \frac{\widetilde{r}^j_{\hat{i}(j)}}{\widetilde{A}_{\hat{i}(j),j}} \widetilde{a}_j \right\|_1 \right),
\end{aligned}$$

as desired. $\square$

2.6 Greedy selection rules for ℓ^0_w-based loss functions

In this section we show how to derive greedy selection rules for ℓ^0_w-regularized loss functions. Specifically, we derive greedy selection rules for ℓ^0_w-based variants of the SR-LASSO (Section 2.6.1) and LAD-LASSO (Section 2.6.2), extending the ℓ^0_w-based LASSO setting considered in [2]. The corresponding weighted OMP algorithms are numerically tested in Section 2.4, Experiments I and II.

2.6.1 ℓ^0_w-based SR-LASSO

We start with the ℓ^0_w-based SR-LASSO. Recall that the ℓ^0_w-norm $\|\cdot\|_{0,w}$ is defined as in (20).

Theorem 2.12 (Greedy selection rule for ℓ^0_w-based SR-LASSO). *Let $\lambda \geq 0$, $S \subseteq [N]$, $A \in \mathbb{C}^{m\times N}$ with ℓ^2-normalized columns, and $x \in \mathbb{C}^N$ satisfying*

$$x \in \arg\min_{z\in\mathbb{C}^N} \|y - Az\|_2 \quad s.t. \quad \operatorname{supp}(z) \subseteq S. \tag{58}$$

Consider the ℓ^0_w-based SR-LASSO loss function

$$G^{\mathrm{SR}}_{\ell^0_w}(z) := \|y - Az\|_2 + \lambda\|z\|_{0,w}, \quad \forall z \in \mathbb{C}^N. \tag{59}$$

Then, for every $j \in [N]$,

$$\min_{t\in\mathbb{C}} G^{\mathrm{SR}}_{\ell^0_w}(x+te_j) = G^{\mathrm{SR}}_{\ell^0_w}(x) - \Delta^{\mathrm{SR}}_{\ell^0_w}(x,S,j),$$

where

$$\Delta^{\text{SR}}_{\ell^0_w}(x,S,j) = \begin{cases} \max\left\{\|r\|_2 - \sqrt{\|r\|_2^2 - |(A^*r)_j|^2 - \lambda w_j^2}, 0\right\} & j \notin S \\ \max\left\{\|r\|_2 + \lambda w_j^2 - \sqrt{\|r\|_2^2 + |x_j|^2}, 0\right\} & j \in S,\ x_j \neq 0 \\ 0 & j \in S,\ x_j = 0 \end{cases}$$

and $r = y - Ax$.

Proof. Let $G = G^{\text{SR}}_{\ell^0_w}$, the weighted SR-LASSO objective defined in (59), and recall that $r = y - Ax$.

Case 1: $j \notin S$**.** In this case, we can write

$$\begin{aligned} G(x+te_j) &= \|y - A(x+te_j)\|_2 + \lambda\|x+te_j\|_{0,w} + \|r\|_2 - \|r\|_2 \\ &= G(x) + \underbrace{\|y - A(x+te_j)\|_2 - \|r\|_2 + \lambda w_j^2 \mathbb{1}_{\{t\neq 0\}}}_{=:h(t)}, \end{aligned}$$

where $\mathbb{1}_E$ denotes the indicator function of the event E. Recalling that the columns of A have unit ℓ^2 norm, we have

$$h(t) = \begin{cases} 0 & t = 0 \\ \sqrt{|t|^2 + \|r\|_2^2 - 2\mathrm{Re}(\bar{t}(A^*r)_j)} - \|r\|_2 + \lambda w_j^2 & t \in \mathbb{C}\backslash\{0\}. \end{cases}$$

Now, letting $t = \rho e^{i\theta}$ with $\rho \geq 0$ and $\theta \in [0, 2\pi)$ we have

$$\sqrt{\rho^2 + \|r\|_2^2 - 2\mathrm{Re}(\rho e^{-i\theta}(A^*r)_j)} - \|r\|_2 + \lambda w_j^2 \geq \sqrt{\|r\|_2^2 + \rho^2 - 2\rho|(A^*r)_j|} - \|r\|_2 + \lambda w_j^2,$$

where the inequality holds as an equality for some θ and the right-hand side is minimized at $\rho = |(A^*r)_j|$. Therefore, in summary,

$$\min_{t\in\mathbb{C}} h(t) = \min\left\{-\|r\|_2 + \sqrt{\|r\|_2^2 + |(A^*r)_j|^2} + \lambda w_j^2, 0\right\},$$

which concludes the case $j \notin S$.

Case 2: $j \in S$**.** In this situation $|(A^*r)_j| = 0$. So we have

$$\begin{aligned} G(x+te_j) &= \|y - A(x+te_j)\|_2 + \lambda\|x+te_j\|_{0,w} \\ &= \underbrace{\sqrt{|t|^2 + \|r\|_2^2} + \lambda w_j^2 \mathbb{1}_{\{t\neq -x_j\}}}_{=:l(t)} + \lambda\|x - x_j e_j\|_{0,w}. \end{aligned}$$

We proceed by minimizing $l(t)$. When $t = -x_j$ we simply have $l(t) = \sqrt{|x_j|^2 + \|r\|_2^2}$. Otherwise when $t \neq -x_j$, the term $\sqrt{\|r\|_2^2 + |t|^2} + \lambda w_j^2$ is minimized at $t = 0$. As a result,

$$\min_{t\in\mathbb{C}} l(t) = \min\left\{\sqrt{|x_j|^2 + \|r\|_2^2}, \|r\|_2 + \lambda w_j^2\right\}.$$

Therefore, we see that

$$\begin{aligned}\min_{t\in\mathbb{C}} G(x + te_j) &= \min_{t\in\mathbb{C}} l(t) + \lambda\|x - x_je_j\|_{0,w}\\ &= \min\left\{\sqrt{|x_j|^2 + \|r\|_2^2}, \|r\|_2 + \lambda w_j^2\right\} + \lambda\|x - x_je_j\|_{0,w} + \|r\|_2 - \|r\|_2\\ &= \|r\|_2 + \|x\|_{0,w} - \lambda w_j^2\mathbb{1}_{\{x_j\neq 0\}} + \min\left\{\sqrt{|x_j|^2 + \|r\|_2^2}, \|r\|_2 + \lambda w_j^2\right\} - \|r\|_2\\ &= G(x) - \lambda w_j^2\mathbb{1}_{\{x_j\neq 0\}} + \min\left\{\sqrt{|x_j|^2 + \|r\|_2^2}, \|r\|_2 + \lambda w_j^2\right\} - \|r\|_2.\end{aligned}$$

Simplifying this expression in the cases $x_j = 0$ and $x_j \neq 0$ concludes the proof. □

2.6.2 ℓ^0-based LAD-LASSO

We conclude by deriving the greedy selection rule for ℓ^0_w-based LAD-LASSO. Like in the case of ℓ^1_w-based LAD-LASSO, we restrict ourselves to the real-valued case.

Theorem 2.13 (Greedy selection rule for ℓ^0_w-based LAD-LASSO). *Let $\lambda \geq 0$, $S \subseteq [N]$, $A \in \mathbb{R}^{m\times N}$ with nonzero columns $a_1, \dots, a_N$, and $x \in \mathbb{R}^N$ satisfying*

$$x \in \arg\min_{z\in\mathbb{R}^N} \|y - Az\|_1 \quad s.t. \quad \operatorname{supp}(z) \subseteq S. \tag{60}$$

Consider the ℓ^0_w-based LAD-LASSO loss function

$$G^{\mathrm{LAD}}_{\ell^0_w}(z) := \|y - Az\|_1 + \lambda\|z\|_{0,w}. \tag{61}$$

Then, for every $j \in [N]$,

$$\min_{t\in\mathbb{R}} G^{\mathrm{LAD}}_{\ell^0_w}(x + te_j) = G^{\mathrm{LAD}}_{\ell^0_w}(x) - \Delta^{\mathrm{LAD}}_{\ell^0_w}(x, S, j),$$

where

$$\Delta^{\mathrm{LAD}}_{\ell^0_w}(x, S, j) = \begin{cases}\max\left\{\|r\|_1 - \left\|r - \frac{r_{\hat{\imath}(j)}}{A_{\hat{\imath}(j),j}}a_j\right\|_1 - \lambda w_j^2, 0\right\} & j \notin S\\ \max\left\{\|r\|_1 - \left\|r - \frac{r_{\hat{\imath}(j)}}{A_{\hat{\imath}(j),j}}a_j\right\|_1, \|r\|_1 - \|r + x_ja_j\|_1 + \lambda w_j^2\right\} & j \in S,\ x_j \neq 0\\ \max\left\{\|r\|_1 - \left\|r - \frac{r_{\hat{\imath}(j)}}{A_{\hat{\imath}(j),j}}a_j\right\|_1 - \lambda w_j^2, 0\right\} & j \in S,\ x_j = 0\end{cases},$$

with $\hat{\imath}(j) = \tau_j(\hat{k}(j))$,

$$\hat{k}(j) = \min\left\{ k \in [\|a_j\|_0] : \sum_{k=1}^{\|a_j\|_0} \frac{|A_{\tau_j(k),j}|}{\|a_j\|_1} \geq \frac{1}{2} \right\},$$

and where $\tau_j : [\|a_j\|_0] \to \operatorname{supp}(a_j)$ *defines a nondecreasing rearrangement of the sequence* $(r_i/A_{ij})_{i\in\operatorname{supp}(a_j)}$*, i.e., is such that* $r_{\tau(k)}/A_{\tau(k),j} \leq r_{\tau(k+1)}/A_{\tau(k+1),j}$ *for every* $k \in [\|a_j\|_0 - 1]$.

Proof. Let $G = G^{\mathrm{LAD}}_{\ell^0_w}$ and recall that $r = y - Ax$.

Case 1: $j \notin S$**.** We have

$$G(x + te_j) = \|y - A(x + te_j)\|_1 + \lambda\|x + te_j\|_{0,w} = \underbrace{\|y - A(x + te_j)\|_1 + \lambda w_j^2 \mathbb{1}_{\{t\neq 0\}}}_{=:h(t)} + \lambda\|x\|_{0,w}.$$

We continue by minimizing $h(t)$. If $t = 0$, we simply have $G(x + te_j) = G(x)$. Otherwise,

$$h(t) = \|y - A(x + te_j)\|_1 + \lambda w_j^2 = \sum_{i=1}^m |r_i - tA_{ij}| + \lambda w_j^2, \quad \forall t \neq 0.$$

Thanks to Lemma 2.11, the right-hand side is minimized at $t = r_{\hat{\imath}(j)}/A_{\hat{\imath}(j),j}$ (note that when $r_{\hat{\imath}(j)} = 0$, the minimum of $h(t)$ over $\mathbb{R}$ is attained at $t = 0$). In summary, we have

$$\begin{aligned}
\min_{t\in\mathbb{R}} G(x + te_j) &= \min_{t\in\mathbb{R}} h(t) + \lambda\|x\|_{0,w} \\
&= \min\left\{ \sum_{i=1}^m \left| r_i - \frac{r_{\hat{\imath}(j)}}{A_{\hat{\imath}(j),j}} A_{i,j} \right| + \lambda w_j^2 + \lambda\|x\|_{0,w} + \|r\|_1 - \|r\|_1, G(x) \right\} \\
&= G(x) - \max\left\{ \|r\|_1 - \sum_{i=1}^m \left| r_i - \frac{r_{\hat{\imath}(j)}}{A_{\hat{\imath}(j),j}} A_{i,j} \right| - \lambda w_j^2, 0 \right\},
\end{aligned}$$

which concludes the case $j \notin S$.

Case 2: $j \notin S$**.** In this case, we can write

$$G(x + te_j) = \|y - A(x + te_j)\|_1 + \lambda\|x + te_j\|_{0,w} = \underbrace{\|r - ta_j\|_1 + \lambda w_j^2 \mathbb{1}_{\{t\neq -x_j\}}}_{l(t)} + \lambda\|x - x_j e_j\|_{0,w}.$$

We proceed by minimizing $l(t)$. If $t = -x_j$, we simply have $l(t) = \|r + x_j a_j\|_1$. Otherwise,

$$l(t) = \|r - ta_j\|_1 + \lambda w_j^2, \quad \forall t \neq -x_j.$$

Similarly to the case $j \notin S$, this is minimized at $t = r_{\hat{i}(j)}/A_{\hat{i}(j),j}$. Therefore, in summary

$$\min_{t\in\mathbb{R}} l(t) = \min\left\{\left\|r - \frac{r_{\hat{i}(j)}}{A_{\hat{i}(j),j}}a_j\right\|_1 + \lambda w_j^2, \|r + x_j a_j\|_1\right\}.$$

As a result,

$$\begin{aligned}\min_{t\in\mathbb{R}} G(x + te_j) &= \min_{t\in\mathbb{R}} l(t) + \lambda\|x - x_j e_j\|_{0,w} \\ &= \min\left\{\left\|r - \frac{r_{\hat{i}(j)}}{A_{\hat{i}(j),j}}a_j\right\|_1 + \lambda w_j^2, \|r + x_j a_j\|_1\right\} + \lambda\|x - x_j e_j\|_{0,w}.\end{aligned}$$

Simplifying this formula for $x_j = 0$ and $x_j \neq 0$ yields the desired result. □

2.7 Conclusions and future research

Adopting a loss-function perspective, we proposed new generalizations of OMP for weighted sparse recovery based on ℓ^0 and ℓ^1 versions of the weighted LASSO, SR-LASSO and LAD-LASSO. Moreover, we showed that the corresponding greedy index selection rules admit explicit formulas (see Theorem 2.5, Theorem 2.6, Theorem 2.7 and Section 2.6). Through numerical illustrations, in Section 2.4 we observed that these algorithms inherit desirable characteristics from some of the associated loss functions, i.e., independence of the tuning parameter to noise level for SR-LASSO, and robustness to sparse corruptions for LAD-LASSO. There are many research pathways still to be pursued. We conclude by discussing some of them.

Although we focused on LASSO-type loss functions, many other regularizers and loss functions remain to be investigated, depending on the context and specific application of interest. This includes regularization based on total variation, nuclear norm, $\ell^p - \ell^q$ norms, and group (or joint) sparsity. One might also attempt to accelerate OMP's convergence by sorting indices based on the greedy selection rules derived in this chapter and selecting more indices at each iteration. This procedure is employed in algorithms such as CoSaMP [91], which can thus be easily generalized to the loss function-based framework. The same holds for a recently proposed sublinear-time variant of CoSaMP [**?**]. It is also worth noting that a different method to incorporate weights into OMP is based on greedy index selection rules of the form

$$j^{(k)} \in \arg\max_{j\in[N]} \left|(A^*(y - Ax^{(k)}))_j/w_j\right|$$

(see [23, 78, 121]). The comparison, both empirical and theoretical, of rules of this form with the loss function-based criteria considered here deserves further investigation, which will be partially conducted in Chapter 3.

Regarding future theoretical developments, Theorem 2.5–Theorem 2.7 demonstrate that the greedy index selection rules considered here achieve maximal loss-function reduction at each iteration. However, these theorems do not provide recovery guarantees for loss-function-based OMP. The development of rigorous recovery theorems based on the Restricted Isometry Property (RIP) or the coherence of the matrix A is an important open problem. An interesting related question is whether the theoretical recipes for the optimal choice of λ available for convex optimization decoders (see Section 2.2.2) remain valid in the greedy setting.

Finally, although in this chapter we focused on high-dimensional function approximation, there are many more applications where loss-function based OMP could be tested. A particularly promising one is video reconstruction from compressive measurements that arises in contexts such as dynamic MRI, where one can incorporate information on the ambient signal of the previously reconstructed frames through weights in order to improve the reconstruction quality of subsequent frames (see, e.g., [56]). Exploring the benefits of loss-function based OMP in this and other applications will be the object of future research work.

Chapter 3

Rescaling-based weighted OMP

In Chapter 2 we proposed a weighted generalization of OMP in conjunction with LASSO-type optimization programs. However, as noted in Section 2.7 this is not the only strategy to generalize OMP to the weighted setting. A natural alternative is to incorporate weights directly into the projected residual without modifying the loss function. In this section we introduce the second strategy, leading to an algorithm that we call *rescaling-based WOMP* to distinguish it from the WLASSO-OMP algorithms of Chapter 2. Throughout the remainder of this dissertation, we use "WOMP" to refer to rescaling-based WOMP, and we refer to algorithms of Chapter 2 as (W)Greedy LASSO. WOMP has previously appeared in the literature in parametric Partial Differential Equations (PDE) [23] and signal processing [78, 121], but an RIP-based theoretical guarantee was lacking. This gap is addressed by our main results: Proposition 3.7 and Theorem 3.8.

This chapter is organized as follows. Section 3.1 provides an introduction to the WOMP algorithm and presents our theoretical results after discussing essential mathematical tools required for their proof. Section 3.2 provides a comparison between the two weighting strategies, by numerical experiments in undersampled regime. Proofs of the main results are given in Section 3.3. In particular within the proofs we also formally justify the rescaling-based weighting strategy.

3.1 Rescaling-based WOMP

Algorithm 3.1 (shown below) puts forward rescaling-based WOMP, where the weights are directly incorporated into the greedy selection criterion. As before, we interpret weights as encoding prior knowledge about the underlying signal, guiding OMP to wisely select the next index. However, rather than penalizing signal coefficients through the loss function—as in the LASSO-based approach–WOMP applies weights directly to the projected residual to prioritize some of the indices over others, and thus pushing the argmax to choose the right index.

Smaller weights encourage the inclusion of corresponding indices in the support, while larger weights tend to suppress them. In principle, whether one divides or multiplies by the weights is largely a matter of convenience, since the two can be interchanged via the change of variable $w_j \mapsto 1/w_j$ and have no fundamental difference in practice. However, for theoretical purposes, division by weights aligns more naturally with the structure of the proofs. In the next chapter, where we implement OMP in the context of neural networks, we will adopt the multiplicative formulation.

Algorithm 3.1 Rescaling-based Weighted Orthogonal Matching Pursuit (WOMP).

Input: desired sparsity k, mixing matrix $A \in \mathbb{C}^{m \times N}$ with ℓ^2-normalized columns, observation vector $y \in \mathbb{C}^m$, vector of weights $w \in \mathbb{R}^N$.

```
Let $x^{(0)} = 0$ and $S^{(0)} = \emptyset$.
For $n = 0, \dots, k-1$ repeat:
    $S^{(n+1)} = S^{(n)} \cup \{j^{(n+1)}\}, \quad j^{(n+1)} \in \underset{j \in [N]}{\operatorname{argmax}} \left\{ \left| \left( A^*(y - Ax^{(n)}) \right)_j / w_j \right| \right\},$    (OMP.1)
    $x^{(n+1)} \in \underset{z \in \mathbb{C}^N}{\operatorname{argmin}} \left\{ \|y - Az\|^2 \quad \text{s.t.} \quad \operatorname{supp}(z) \subseteq S^{(n+1)} \right\}.$    (OMP.2)
```

Output: k-sparse vector $\hat{x} = x^{(k)} \in \mathbb{C}^N$.

3.1.1 Mathematical tools for weighted sparsity

In this section we present mathematical tools necessary for the theoretical analysis of WOMP. Our presentation follows [11, 98].

Definition 3.1 (Weighted ℓ^p_w-norm)**.** *For a vector $x \in \mathbb{C}^N$ and weights $w \in \mathbb{R}^N$ with $w \geq 0$ and $1 \leq p \leq 2$, the weighted ℓ^p_w-norm is defined as*

$$\|x\|_{p,w} = \left(\sum_{j=1}^{N} w_j^{2-p} |x_j|^p \right)^{1/p}. \tag{62}$$

Remark 3.2 (Weighted ℓ^p spaces)**.** *In the infinite-dimensional setting, equipped with these norms we can define weighted ℓ^p spaces as the space of signals that have finite weighted ℓ^p_w-norm. However, since we work in a finite-dimensional setting with ambient dimension N, all signals considered throughout this work naturally belong to the weighted ℓ^p spaces (see, e.g., [11]).*

Remark 3.3. *The reason behind the $2-p$ exponent in Equation (62) is primarily convenience as it vanishes at $p = 2$ and thus many of the standard sparsity result are directly extended to the new setting (see, e.g., Theorems 3.5 and 3.6).*

We can also define the weighted ℓ^0(-quasi)-norm as

$$\|x\|_{0,w} = \sum_{j=1}^{N} w_j^2,$$

which also motivates a notion of weighted cardinality of a set, i.e., for a set S we define its weighted cardinality as $|S|_w = \sum_{j \in S} w_j^2$. Now we are in a position to define sparsity in the weighted setting.

Definition 3.4 (Weighted sparsity)**.** *A vector $x \in \mathbb{C}^N$ belonging to the ℓ^p_w-space with $w \in \mathbb{R}^N$ is weighted (k, w)-sparse if $|\operatorname{supp}(x)|_w = \sum_{j \in \operatorname{supp}(x)} w_j^2 \leq k$.*

To complete our toolbox we need two extra tools: weighted versions of the best s-term approximation error in Equation (6) and restricted isometry property (see Equation (7)).

Definition 3.5 (Weighted best (k, w)-term approximation error)**.** *Let* $0 < p \leq 2$*,* $w \geq 0$*,* $x \in \mathbb{C}^N$ *and* $k \geq 0$*. The* ℓ_w^p*-norm weighted best* (k, w)*-term approximation error of* x *is*

$$\sigma_k(x)_{p,w} = \inf_{z \in \mathbb{C}^N} \{\|x - z\|_{p,w} : \, |\operatorname{supp}(z)|_w \leq k\}.$$

Since the weighted best (k, w)-term approximation error cannot be easily computed as in the unweighted case, we further introduce a more practical proxy that can be easily evaluated using sorting and thresholding. Let $\pi : [N] \to [N]$ be a bijection defining a nonincreasing rearrangement such that $w_{\pi(1)}^{2-p}|x_{\pi(1)}|^p \geq w_{\pi(2)}^{2-p}|x_{\pi(2)}|^p \geq \cdots \geq w_{\pi(N)}^{2-p}|x_{\pi(N)}|^p$ and we define $S = \{\pi(1), \pi(2), \ldots, \pi(M)\}$ with M as the maximal integer satisfying $|S|_w \leq k$. Then the quasi-best weighted (k, w)-approximation error is

$$\tilde{\sigma}_k(x)_{p,w} = \|x - x_S\|_{p,w}. \tag{63}$$

Fortunately, there is the following relationship between the quasi-best and best weighted (k, w)-term errors that facilitates switching between them conveniently [11, Proposition 2.13].

$$\tilde{\sigma}_{k+\|w\|_\infty^2}(x)_{p,w} \leq \sigma_k(x)_{p,w} \leq \tilde{\sigma}_k(x)_{p,w}. \tag{64}$$

Now we introduce our final ingredient: a near-isometry property for the mapping A when applied to weighted (k, w)-sparse signals. This characterization enables the derivation of recovery guarantees under appropriate conditions [98].

Definition 3.6 (Weighted restricted isometry property)**.** *The weighted Restricted Isometry Constant (wRIC)* $\delta(k, w) < 1$ *of a matrix* $A \in \mathbb{C}^{m \times N}$ *is the smallest* $\delta > 0$ *such that*

$$(1 - \delta)\|x\|^2 \leq \|Ax\|^2 \leq (1 + \delta)\|x\|^2, \quad \forall (k, w)\text{-sparse vectors } x \in \mathbb{C}^N. \tag{65}$$

A matrix satisfying Equation (65) *is said to have the weighted Restricted Isometry Property (wRIP) of order* (k, w) *with constant* δ*.*

3.1.2 Theoretical analysis

Perhaps counterintuitively, an RIP-based recovery guarantee for standard (unweighted) OMP that allows reconstruction of any s-sparse signal in s iterations is proven to be impossible; as there exist simple counterexamples that demonstrate failure in such cases [85]. Recovery can become feasible if the algorithm is allowed to run more than s iterations, typically a constant multiple of s, though the corresponding proof in the general case remains convoluted. The first such RIP-based guarantee for unweighted OMP was established in [125], and was later simplified in [36, 55]. An alternative proof, applicable to signals that are "flat" over their support, was introduced in [53]. It is this latter approach that we adopt to derive an RIP-based recovery guarantee for WOMP.

We begin with the following proposition, which shows that the least-squares loss associated with a weighted (k, w)-sparse signal is reduced to the noise level after a certain number of WOMP iterations, related to the signal's "weighted flatness" and its weighted cardinality, provided that the weighted restricted isometry constant is sufficiently small.

Proposition 3.7. *Let $x \in \mathbb{R}^N$ be a (k, w)-sparse vector, i.e., $\|x\|_{0,w} \leq k$ where $w \in \mathbb{R}^N$ is a weight vector satisfying $w > 0$ and $\gamma \geq 1$ such that $\max_{j\in\mathrm{supp}(x)} |x_j|/w_j \leq \gamma \min_{j\in\mathrm{supp}(x)} |x_j|/w_j$. If A with ℓ^2-normalized columns has weighted restricted isometry constant $\delta_{w,r+k} \leq 1/5$ with $r := 12\gamma^2 k$, then for all $e \in \mathbb{R}^m$, $x^{(t)}$ the output of running t iterations of Algorithm 3.1 on $y = Ax + e$ with t any integer such that $\|x^{(t)}\|_{0,w} \geq r$, satisfies*

$$\|y - Ax^{(t)}\| \leq 4\|e\|.$$

Proof. See, Section 3.3.1. Note that in the first stages of the proof, we justify the choice of the greedy index in derivation of inequality (66). □

We now arrive at the main theorem of this chapter for WOMP in the form of a standard compressive sensing recovery guarantee for a signal $x \in \mathbb{C}^N$. This theorem essentially shows that the reconstruction map of OMP, i.e., $y \mapsto x^{(t)}$ admits weighted mixed-$(\ell^2, \ell^1)_w$ instance optimality of order $(k - \|w\|_\infty^2, w)$ (see, e.g., [55, Chapter 11] for the unweighted case).

Theorem 3.8. *Let $x \in \mathbb{R}^N$ associated with the weight vector $w \in \mathbb{R}^N$ with $\|w\|_\infty^2 \leq k$ and $w > 0$. Also, let $\gamma \geq 1$ such that $\max_{j\in\mathrm{supp}(x)} |x_j|/w_j \leq \gamma \min_{j\in\mathrm{supp}(x)} |x_j|/w_j$. If A with ℓ^2-normalized columns has weighted restricted isometry constant $\delta_{w,r+2k} \leq \delta_* \leq 1/5$ with $r := 24\gamma^2 k$, then for all $e \in \mathbb{R}^m$, $x^{(t)}$ the output of running t iterations of WOMP algorithm on $y = Ax + e$ for any integer t such that $\|x^{(t)}\|_{0,w} \geq r$, satisfies*

$$\|x - x^{(t)}\| \leq C\left(\frac{\|w\|_\infty^2}{\min_{j\in[N]} w_j^2} + 1\right)\frac{\sigma_{k-\|w\|_\infty^2}(x)_{1,w}}{\sqrt{k}} + D\|e\|,$$

where C and D are constants only depending on δ_* and k.

Proof (sketch). We first introduce a set T, as an equivalent version of the signal support in the weighted case for a generic signal (not necessarily sparse). This allows us to distinguish between the dominant signal contents expected to be discovered by the algorithm, and the signal tail. In light of $\|x - x^{(t)}\| \leq \|x_T - x^{(t)}\| + \|x_{\overline{T}}\|$, we relate the first term to the residual bound of Proposition 3.7 and the second term to the best $(2k, w)$-term approximation error. For a complete proof, see Section 3.3.2. □

3.2 Numerical experiments

In this section we numerically compare WOMP algorithms with two weighting strategies, namely the one proposed in Algorithm 3.1 and ℓ^1-based LASSO-WOMP of Chapter 2 (see also Theorem 2.5). To this purpose, we follow the numerical setup in Experiment III of Section 2.4, i.e., with moderate choice of m and in the undersampled regime (m very small), where weights are proven to be more effective [14]. For a set $T \subset [N]$ containing oracle knowledge and $w_0 > 1$, we

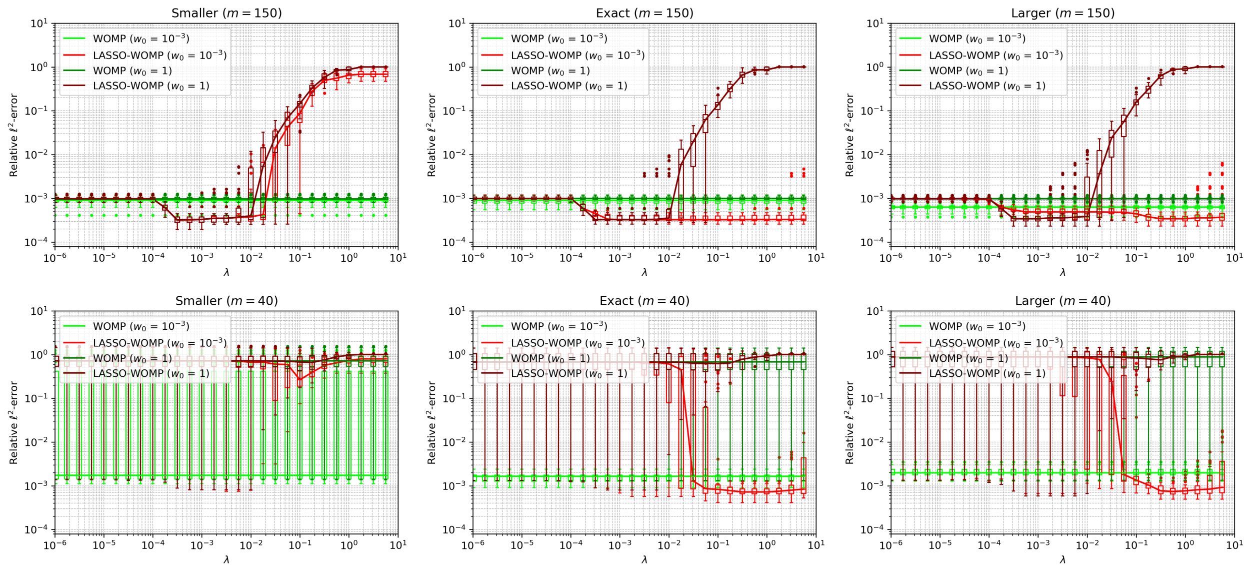

Figure 3.1: Relative error as a function of the tuning parameter. ℓ^1-based LASSO-WOMP is compared with rescaling-based WOMP (referred to as WOMP in the figures) for different weight levels and different oracle information type.

create the weight vector $w \in \mathbb{R}^N$ as

$$w_j = \begin{cases} w_0 & j \in T, \\ 1 & \text{otherwise} \end{cases}.$$

For each s-sparse signal $x \in \mathbb{C}^N$ generated with $S := \operatorname{supp}(x),\ \operatorname{card}(S) = s$, we consider oracle knowledge with

- partial support information (smaller), i.e., $T \subset S,\ \operatorname{card}(T) = \lceil s/2 \rceil$.
- exact support information (exact), i.e., $T = S,\ \operatorname{card}(T) = s$.
- redundant support information (larger), i.e., $T \supset S,\ \operatorname{card}(T) = 2s$.

We then reconstruct the signal using N_{iter} iterations of LASSO-WOMP and rescaling-based WOMP and compute the relative ℓ^2-error. We repeat this process N_{repeat} times and plot boxplots of relative ℓ^2-error over a range of tuning parameter λ, the tuning parameter of LASSO-WOMP (see Theorem 2.5). We run the following parameter setting:

$$N = 300,\ s = 16,\ m = 40 \text{ and } 150,\ w_0 \in \{10^{-3}, 1\},\ \lambda \in 10^{\{-6:0.25:1\}},\ N_{\text{repeat}} = 30,\ N_{\text{iter}} = 2s.$$

When $w_0 = 1$, no support information is provided and when $w_0 = 10^{-3}$ the indices in T are highly promoted. Figure 3.1 shows the result of this experiment. It is evident that almost in all cases there is a range of λ for which LASSO-WOMP outperforms rescaling-based WOMP, but if redundant or exact support information is provided rescaling-based WOMP could be a more reliable choice as it does not involve the tuning parameter.

3.3 Proofs

In this section we present proofs of our main results. We begin by Proposition 3.7 as it will be used later in the proof of Theorem 3.8.

3.3.1 Proof of Proposition 3.7

Proof. We begin by observing that $A^* : \mathbb{C}^m \to \mathbb{C}^N$ maps the residual back into the signal space. Accordingly, we examine how closely the projected residual aligns with the true signal. Specifically, we aim to upper bound the quantity $\langle x, A^*(y - Ax^{(n)})\rangle$ over the set $S \backslash S^{(n)}$, which represents the portion of the true support that remains unrecovered at iteration n. Interestingly, this analysis also offers a justification for the greedy selection criterion used in WOMP. We start by estimating

$$\begin{aligned}
\sum_{j\in S\backslash S^{(n)}} x_j \left(A^*(y - Ax^{(n)})\right)_j &\leq \sum_{j\in S\backslash S^{(n)}} \frac{w_j}{w_j} |x_j| \left|\left(A^*(y - Ax^{(n)})\right)_j\right| \\
&\leq \max_{j\in S\backslash S^{(n)}} \left|\frac{\left(A^*(y - Ax^{(n)})\right)_j}{w_j}\right| \sum_{j\in S\backslash S^{(n)}} w_j |x_j| \\
&\leq \|x_{S\backslash S^{(n)}}\|_{1,w} \left|\frac{\left(A^*(y - Ax^{n})\right)_{j^{(n+1)}}}{w_{j^{(n+1)}}}\right|. \qquad (66)
\end{aligned}$$

We continue by lower bounding this term. Knowing that $x^{(n)}_{S\backslash S^{(n)}} = 0$ and $\left(A^*(y - Ax^{(n)})\right)_{S^{(n)}} = 0$, as a consequence of the least-squares since residual is orthogonal to the span of chosen columns, we see that

$$\begin{aligned}
\sum_{j\in S\backslash S^{(n)}} x_j \left(A^*(y - Ax^{(n)})\right)_j &= \sum_{j\in S\backslash S^{(n)}} (x - x^{(n)})_j \left(A^*(y - Ax^{(n)})\right)_j \\
&\quad + \sum_{j\in S^{(n)}} (x - x^{(n)})_j \left(A^*(y - Ax^{(n)})\right)_j \\
&= \sum_{j\in S\cup S^{(n)}} (x - x^{(n)})_j \left(A^*(y - Ax^{(n)})\right)_j \\
&= \langle x - x^{(n)}, A^*(y - Ax^{(n)})\rangle = \langle A(x - x^{(n)}), y - Ax^{(n)}\rangle \\
&= \langle A(x - x^{(n)}), y - Ax - A(x^{(n)} - x)\rangle \\
&= \langle A(x - x^{(n)}), A(x - x^{(n)}) + e\rangle \\
&= \|A(x - x^{(n)})\|^2 + \langle A(x - x^{(n)}), e\rangle \\
&\geq \|A(x - x^{(n)})\|^2 - \|A(x - x^{(n)})\|\|e\|, \qquad (67)
\end{aligned}$$

where in the last step we used the Cauchy-Schwartz inequality. Combining Equation (66) and Equation (67) we have

$$\|x_{S\backslash S^{(n)}}\|_{1,w} \left|\frac{\left(A^*(y - Ax^{n})\right)_{j^{(n+1)}}}{w_{j^{(n+1)}}}\right| \geq \|A(x - x^{(n)})\|^2 - \|A(x - x^{(n)})\|\|e\|. \qquad (68)$$

Now if for some $n < t$, $\|A(x - x^{(n)})\| \le \|e\|/(1-c)$, then we have

$$\|y - Ax^{(t)}\| \le \|y - Ax^{(n)}\| \le \|A(x - x^{(n)})\| + \|e\| \le \left(1 + \frac{1}{1-c}\right)\|e\| \le 4\|e\|,$$

for $c = \sqrt{5}/4$ and we are done. Moreover, if $S \subseteq S^{(n)}$ for some $n < t$ then

$$\|y - Ax^{(t)}\| \le \|y - Ax^{(n)}\| \le \|y - Ax\| = \|e\|,$$

which implies $\|y - Ax^{(t)}\| \le 4\|e\|$ and we are done again. So we continue assuming, for all $n < t$, that

$$\|A(x - x^{(n)})\| \ge \|e\|/(1-c) \tag{69}$$

$$\|x_{S\backslash S^{(n)}}\|_{1,w} > 0 \quad (\text{because} \quad S\backslash S^{(n)} \ne \emptyset). \tag{70}$$

So replacing $\|e\|$ in Equation (68) by its upper bound in Equation (69) we get

$$\begin{aligned}\|y - Ax^{(n+1)}\|^2 &\le \|y - Ax^{(n)}\|^2 - \left|(A^*(y - Ax^n))_{j^{(n+1)}}\right|^2 \\ &\le \|y - Ax^{(n)}\|^2 - c^2 w_{j^{(n+1)}}^2 \frac{\|A(x - x^{(n)})\|^4}{\|x_{S\backslash S^{(n)}}\|_{1,w}^2},\end{aligned} \tag{71}$$

where the first inequality is due to loss reduction in step (WOMP.2) of Algorithm 3.1 at each iteration (see, e.g., [55, Lemma 3.3]). Also as x is supported on S and $x^{(n)}$ on $S^{(n)}$, we can derive

$$\|x - x^{(n)}\|^2 \ge \|x_{S\backslash S^{(n)}}\|^2 = \sum_{j \in S\backslash S^{(n)}} |x_j|^2 \ge \Bigg(\underbrace{\min_{j \in S\backslash S^{(n)}} \frac{|x_j|}{w_j}}_{\alpha}\Bigg)^2 \sum_{j \in S\backslash S^{(n)}} w_j^2 \ge \frac{\beta^2}{\gamma^2}|S\backslash S^{(n)}|_w, \tag{72}$$

with $\beta = \max_{j \in S} \frac{|x_j|}{w_j}$ and the factor γ is such that $\beta \le \gamma\alpha$ by assumption. We further have

$$\|x_{S\backslash S^{(n)}}\|_{1,w} = \sum_{j \in S\backslash S^{(n)}} w_j|x_j| \le \beta \sum_{j \in S\backslash S^{(n)}} w_j^2 = \beta|S\backslash S^{(n)}|_w. \tag{73}$$

Also,

$$\|x\|^2 = \sum_{j \in S} w_j^2 \frac{|x_j|^2}{w_j^2} \le \max_{j \in S} \frac{|x_j|^2}{w_j^2} \sum_{j \in S} w_j^2 \le \beta^2 k \quad \Rightarrow \quad \beta^2 \ge \frac{\|x\|^2}{k}. \tag{74}$$

Since A satisfies wRIP with constant $\delta_{w,r+k}$, then

$$(1 - \delta_{w,r+k})\|z\|^2 \le \|Az\|^2 \le (1 + \delta_{w,r+k})\|z\|^2, \quad \forall\, z\ (r+k, w)\text{-sparse}. \tag{75}$$

Putting together Equations (72) to (74) and (71) and using Equation (75), and observing that $x - x^{(n)}$ is (k, w)-sparse while noting that $\delta_{w,k_1} \leq \delta_{w,k_2}$, for any $k_1 \leq k_2$, we obtain

$$\begin{aligned}
\|y - Ax^{(n+1)}\|^2 &\leq \|y - Ax^{(n)}\|^2 - c^2 w_{j^{(n+1)}}^2 \frac{\|A(x - x^{(n)})\|^4}{\|x_{S\backslash S^{(n)}}\|_{1,w}^2} \\
&\leq \|y - Ax^{(n)}\|^2 - c^2 w_{j^{(n+1)}}^2 (1 - \delta_{w,r+k})^2 \frac{\|x - x^{(n)}\|^4}{\|x_{S\backslash S^{(n)}}\|_{1,w}^2} \\
&\leq \|y - Ax^{(n)}\|^2 - c^2 w_{j^{(n+1)}}^2 (1 - \delta_{w,r+k})^2 \frac{\beta^4 |S\backslash S^{(n)}|_w^2}{\gamma^4 \beta^2 |S\backslash S^{(n)}|_w^2} \\
&\leq \|y - Ax^{(n)}\|^2 - c^2 w_{j^{(n+1)}}^2 (1 - \delta_{w,r+k})^2 \frac{\beta^2}{\gamma^4} \\
&\leq \|y - Ax^{(n)}\|^2 - \frac{c^2 w_{j^{(n+1)}}^2 (1 - \delta_{w,r+k})^2}{\gamma^4} \frac{\|x\|^2}{k}.
\end{aligned}$$

By induction we get

$$\begin{aligned}
\|y - Ax^{(t)}\|^2 &\leq \|y - Ax^{(t-1)}\|^2 - \frac{c^2(1 - \delta_{w,r+k})^2}{\gamma^4} \frac{\|x\|^2}{k} w_{j^{(t)}}^2 \\
&\leq \|y\|^2 - \frac{c^2(1 - \delta_{w,r+k})^2}{\gamma^4} \frac{\|x\|^2}{k} \sum_{n=1}^{t} w_{j^{(n)}}^2 \\
&\leq 2\|Ax\|^2 + 2\|e\|^2 - \frac{c^2(1 - \delta_{w,r+k})^2}{\gamma^4} \frac{\|x\|^2}{k} \sum_{n=1}^{t} w_{j^{(n)}}^2 \\
&\leq \left(2(1 + \delta_{w,r+k}) - \frac{c^2(1 - \delta_{w,r+k})^2}{k\gamma^4} \sum_{n=1}^{t} w_{j^{(n)}}^2 \right) \|x\|^2 + 2\|e\|^2.
\end{aligned}$$

Now, we need the t large enough such that $\left(2(1 + \delta_{w,r+k}) - \frac{c^2(1-\delta_{w,r+k})^2}{k\gamma^4} \sum_{n=1}^{t} w_{j^{(n)}}^2\right) \leq 0$ so as to establish

$$\|y - Ax^{(t)}\|^2 \leq 2\|e\|^2.$$

Recalling Theorem 3.4, this means that t needs to be large enough so that

$$\|x\|_{0,w} \geq \frac{2(1 + \delta_{w,r+k} k\gamma^4)}{c^2(1 - \delta_{w,r+k})^2}.$$

Using $\delta_{w,r+k} \leq 1/5$ and $c = \sqrt{5}/4$, the result of the statement is satisfied when choosing t large enough such that $\|x^{(t)}\|_{0,w} \geq 12k\gamma^4$, which concludes the proof. □

3.3.2 Proof of Theorem 3.8

Before presenting the proof, we first introduce two technical lemmas. The first is a weighted version of the well-known Stechkin inequality (see, e.g., [55, Theorem 2.5] for the unweighted case), which characterizes the decay of signal coefficients. This inequality enables us to relate the best (k, w)-term approximation error measured in the ℓ^q-norm to the overall signal energy measured in ℓ^p-norm.

Lemma 3.9 (Weighted Stechkin inequality [11, Lemma 3.12])**.** *Let* $0 < p \leq q \leq 2$, $k > 0$ *and* $w = (w_j)_{j\in[N]}$ *with* $w > 0$. *Then*

$$\sigma_k(x)_{q,w} \leq \|x\|_p k^{-(\frac{1}{p}-\frac{1}{q})}.$$

In particular,

$$\sigma_k(x)_{2,w} \leq \frac{\|x\|_1}{\sqrt{k}}.$$

The second lemma is [55, Lemma 6.10] that let us to switch between ℓ^1 and ℓ^2 norm of two signals with ordered magnitudes. We extend this result to the weighted setting, enabling its application in our analysis.

Lemma 3.10 (Weighted generalization of [55, Lemma 6.10])**.** *Let* $x \in \mathbb{R}^s$, $z \in \mathbb{R}^t$ *be vectors such* $\|x\|_{0,w} \leq k_x$ *and* $\|z\|_{u,0} \leq k_z$ *and* $\max_{i\in[s]} \frac{|x_i|}{w_i} \leq \min_{j\in[t]} \frac{|z_j|}{u_j}$ *with* w, u *associated weight vectors. Then*

$$\frac{\|x\|}{\sqrt{k_x}} \leq \frac{\|z\|_{1,u}}{k_z}. \tag{76}$$

In particular, if $s = t$ *and* $w = u$ *(thus* $k_x = k_z =: k$*), then*

$$\|x\| \leq \frac{\|z\|_{1,w}}{\sqrt{k}}. \tag{77}$$

Proof. We note that

$$\|z\|_{1,u} = \sum_{j\in[t]} u_j|z_j| = \sum_{j\in[t]} u_j^2\frac{|z_j|}{u_j} \geq k_z \min_{j\in[t]} \frac{|z_j|}{u_j},$$
$$\|x\| = \sqrt{\sum_{i\in[s]} \frac{w_i^2}{w_i^2}x_i^2} \leq \sqrt{k_x} \max_{i\in[s]} \frac{|x_i|}{w_i},$$

and the result follows. □

Now we arrive at the proof of the main theorem.

Proof of Theorem 3.8. We first need an alternative to the signal support in the weighted case for a generic signal (not necessarily sparse). To this end, let $T = \{x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(M)}\}$ with M the maximal integer satisfying $\sum_{j\in T} w_{\pi(j)}^2 \leq 2k$ and $\pi : [N] \to [N]$ a bijection applying a nonincreasing rearrangement on $\{w_j|x_j|\}_{j=1}^N$, such that $w_{\pi(1)}|x_{\pi(1)}| \geq w_{\pi(2)}|x_{\pi(2)}| \geq \cdots \geq w_{\pi(N)}|x_{\pi(N)}|$. Moreover, let $S_0 = \{x_{\pi(1)}, x_{\pi(2)}, \dots, x_{\pi(M_0)}\}$ with $M_0 \leq M$ the maximal integer satisfying $\sum_{j\in S_0} w_{\pi(j)}^2 \leq k$ and $S_1 = T\backslash S_0$. We write $y = Ax + e = Ax_T + Ax_{\overline{T}} + e = Ax_T + e'$

where $e' = Ax_{\overline{T}} + e$. Then

$$\begin{aligned}
\|x - x^{(t)}\| &\le \|x_T - x^{(t)}\| + \|x_{\overline{T}}\| \\
&= \|x_T - x^{(t)}\| + \|x_{\overline{S_0}} - x_{S_1}\| \\
&= \|x_T - x^{(t)}\| + \sigma_k(x_{\overline{S_0}})_{2,w} \\
&\le \|x_T - x^{(t)}\| + \frac{\|x_{\overline{S_0}}\|_{1,w}}{\sqrt{k}} = \|x_T - x^{(t)}\| + \frac{\tilde{\sigma}_k(x)_{1,w}}{\sqrt{k}},
\end{aligned}$$

where we used the weighted stechkin inequality in Lemma 3.9 and Equation (64). We continue by bounding the first term as

$$\begin{aligned}
\|x_T - x^{(t)}\| &\le \frac{1}{\sqrt{1-\delta_{w,r+2k}}} \|A(x_T - x^{(t)})\| \\
&= \frac{1}{\sqrt{1-\delta_{w,r+2k}}} \|y - Ax_{\overline{T}} - e - Ax^{(t)})\| \\
&\le \frac{1}{\sqrt{1-\delta_{w,r+2k}}} \left(\|y - Ax^{(t)})\| + \|Ax_{\overline{T}}\| + \|e\|\right) \\
&\le \frac{1}{\sqrt{1-\delta_{w,r+2k}}} \|Ax_{\overline{T}}\| + \frac{5}{\sqrt{1-\delta_{w,r+2k}}} \|e\|,
\end{aligned}$$

where we used the wRIP and the result of Proposition 3.7 for the $(2k, w)$-sparse vector x_T.

Also for $\|Ax_{\overline{T}}\|$ in the bound above we take advantage of the partitioning $\overline{T} = S_2 \sqcup \cdots \sqcup S_K$ based on the nonincreasing rearrangement such that $|x_{\rho^{(l)}(1)}|/w_{\rho^{(l)}(1)} \ge \cdots \ge |x_{\rho^{(l)}(M_l)}|/w_{\rho^{(l)}(M_l)}$, $|S_l|_w \le k,\ S_l = \{x_{\rho^{(l)}(1)}, \ldots, x_{\rho^{(l)}(M_l)}\}$ for each $l \ge 2$. Then

$$\|Ax_{\overline{T}}\| = \left\| A \sum_{l\ge 2} x_{S_l} \right\| \le \sum_{l \ge 2} \|Ax_{S_l}\| \le \sqrt{1+\delta_{w,k}} \sum_{l\ge 2} \|x_{S_l}\| = \sqrt{1+\delta_{w,k}} \left(\|x_{S_2}\| + \sum_{l \ge 3} \|x_{S_l}\| \right)$$

For the sum using Lemma 3.10 we note that

$$\sum_{l\ge 3} \|x_{S_l}\| \le \sum_{l \ge 2} \frac{\|x_{S_l}\|_{1,w}}{\sqrt{k}}.$$

However since the type of rearrangement for $T = S_0 \cup S_1$ is different, we cannot use Lemma 3.10 for S_2 as well and need to treat $\|x_{S_2}\|$ differently. We thus write

$$\begin{aligned}
\|x_{S_2}\| = \left(\sum_{j\in S_2} |x_j|^2 \right)^{1/2} &\le \sqrt{k} \max_{j \in S_2} \frac{|x_j|}{w_j} \overset{\text{(i)}}{\le} \sqrt{k} \max_{j\in S_2} w_j |x_j| / \min_{j \in S_2} w_j^2 \\
&\overset{\text{(ii)}}{\le} \sqrt{k} \min_{j\in S_1} w_j |x_j| / \min_{j\in S_2} w_j^2 \overset{\text{(iii)}}{\le} \sqrt{k} \min_{j \in S_1} \frac{|x_j|}{w_j} \Big/ \left(\min_{j\in S_1} \frac{1}{w_j^2} \min_{j \in S_2} w_j^2 \right) \\
&\overset{\text{(iv)}}{\le} \sqrt{k} \frac{\|x_{S_1}\|_{1,w}}{k} \max_{j\in S_1} w_j^2 / \min_{j \in S_2} w_j^2 \le \frac{\|w\|_\infty^2}{\min_{j\in[N]} w_j^2} \frac{\|x_{S_1}\|_{1,w}}{\sqrt{k}}.
\end{aligned}$$

In the chain of inequalities above:

(i) holds because for all $j \in S_2$

$$\min_{j\in S_2} w_j^2 \leq w_j^2 \Rightarrow \frac{|x_j|}{w_j}\min_{j\in S_2} w_j^2 \leq w_j^2\frac{|x_j|}{w_j} = w_j|x_j| \Rightarrow \max_{j\in S_2}\frac{|x_j|}{w_j}\min_{j\in S_2} w_j^2 \leq \max_{j\in S_2} w_j|x_j|,$$

and rearrangement of the final inequality.

(ii) holds because by construction S_1 contains the largest elements of $w_j|x_j|,\ j \in \overline{S_0}$.

(iii) holds because for all $j \in S_1$

$$\min_{j\in S_1} w_j|x_j| \leq w_j|x_j| \Rightarrow \frac{1}{w_j^2}\min_{j\in S_1} w_j|x_j| \leq \frac{1}{w_j^2}w_j|x_j| = \frac{|x_j|}{w_j} \Rightarrow \min_{j\in S_1}\frac{1}{w_j^2}\min_{j\in S_1} w_j|x_j| \leq \min_{j\in S_1}\frac{|x_j|}{w_j},$$

and rearrangement of the final inequality.

(iv) uses the result derived in the proof of Lemma 3.10.

Now putting together all these relations we arrive at

$$\begin{aligned}
\|Ax_{\overline{T}}\| &\leq \sqrt{1+\delta_{w,k}}\left(\|x_{S_2}\| + \sum_{l\geq 3}\|x_{S_l}\|\right) \\
&\leq \sqrt{1+\delta_{w,k}}\left(\frac{\|w\|_\infty^2}{\min_{j\in[N]} w_j^2}\frac{\|x_{S_1}\|_{1,w}}{\sqrt{k}} + \sum_{l\geq 2}\frac{\|x_{S_l}\|_{1,w}}{\sqrt{k}}\right) \\
&\leq \sqrt{1+\delta_{w,k}}\max\left\{\frac{\|w\|_\infty^2}{\min_{j\in[N]} w_j^2}, 1\right\}\sum_{l\geq 1}\frac{\|x_{S_l}\|_{1,w}}{\sqrt{k}} \\
&= \sqrt{1+\delta_{w,k}}\frac{\|w\|_\infty^2}{\min_{j\in[N]} w_j^2}\frac{\|x_{\overline{S_0}}\|_{1,w}}{\sqrt{k}} \\
&\leq \sqrt{1+\delta_{w,k}}\frac{\|w\|_\infty^2}{\min_{j\in[N]} w_j^2}\frac{\tilde{\sigma}_k(x)_{1,w}}{\sqrt{k}}.
\end{aligned}$$

Finally

$$
\begin{aligned}
\|x - x^{(t)}\| &\leq \|x_T - x^{(t)}\| + \frac{\tilde{\sigma}_k(x)_{1,w}}{\sqrt{k}} \\
&\leq \frac{1}{\sqrt{1-\delta_{w,r+2k}}}\|Ax_{\overline{T}}\| + \frac{5}{\sqrt{1-\delta_{w,r+2k}}}\|e\| + \frac{\tilde{\sigma}_k(x)_{1,w}}{\sqrt{k}} \\
&\leq \sqrt{\frac{1+\delta_{w,r+2k}}{1-\delta_{w,r+2k}}\frac{\|w\|_\infty^2}{\min_{j\in[N]} w_j^2}}\frac{\tilde{\sigma}_k(x)_{1,w}}{\sqrt{k}} + \frac{5}{\sqrt{1-\delta_{w,r+2k}}}\|e\| + \frac{\tilde{\sigma}_k(x)_{1,w}}{\sqrt{k}} \\
&= \left(1 + \sqrt{\frac{1+\delta_*}{1-\delta_*}\frac{\|w\|_\infty^2}{\min_{j\in[N]} w_j^2}}\right)\frac{\tilde{\sigma}_k(x)_{1,w}}{\sqrt{k}} + \frac{5}{\sqrt{1-\delta_*}}\|e\| \\
&\leq \sqrt{\frac{1+\delta_*}{1-\delta_*}}\left(\frac{\|w\|_\infty^2}{\min_{j\in[N]} w_j^2} + 1\right)\frac{\sigma_{k-\|w\|_\infty^2}(x)_{1,w}}{\sqrt{k}} + \frac{5}{\sqrt{1-\delta_*}}\|e\|,
\end{aligned}
$$

using Equation (64), and the result follows with

$$
C = \sqrt{\frac{1+\delta_*}{1-\delta_*}} \quad \text{and} \quad D = \frac{5}{\sqrt{1-\delta_*}}.
$$

□

So far we investigated weighted generalizations of OMP. Weighted IHT was also previously proposed in the literature [71]. We further note that with weighted versions of OMP and IHT, weighted CoSaMP could be easily considered. Now we have a full family of greedy algorithms that we aim at unrolling onto neural networks in the subsequent chapters.

Chapter 4

Deep greedy unfolding

"Where is the Life we have lost in living?
Where is the wisdom we have lost in knowledge?
Where is the knowledge we have lost in information?"
— T.S.Eliot

This chapter is in collaboration with Prof. Matthew J. Colbrook at the University of Cambridge and the thesis' supervisor, Prof. Simone Brugiapaglia, and in its current form (modulo very minor edits), has been submitted to the *SIAM journal on Mathematics of Data Science* [88].

4.1 Introduction

The last decade has witnessed great technological developments, ranging from language models to industrial robots, thanks to Artificial Intelligence (AI). These advancements mostly rely on *deep learning*—a flexible data-driven approach that learns from observables to make predictions on unseen data [58]. However, while deep learning continues to push the boundaries of machine intelligence, concerns remain about its safety and trustworthiness in critical sectors such as autonomous vehicles and medical imaging [35, 66]. These concerns stem primarily from a general lack of interpretability and stability in such models.

On the other hand, *sparse recovery* algorithms, which are used to obtain a parsimonious representation of the data and backed by proven breakthroughs in modern compression and data acquisition [3, 55], benefit from concrete mathematical justifications, including recovery guarantees and convergence analysis.

Deep learning and sparse recovery algorithms are unified under the umbrella of *algorithm unrolling* (also known as *unfolding* or *unraveling*) [81, 89, 103]. This paradigm involves unfolding a (sparse recovery) algorithm into the layers of a neural network, where certain parameters of the original algorithm are treated as trainable, thus optimizing the performance of the algorithm with respect to those parameters.

However, training neural networks, which relies on gradient-based optimization, necessitates differentiability at all steps. This requirement clashes with the fact that a broad class of sparse recovery methods, known as *greedy and thresholding algorithms* [55], employs the non-differentiable *argsort* operator—a piece-wise constant operator (see Figure 4.1). This operator poses a challenge for unrolled neural networks, rendering them non-trainable due to its ineffective zero or non-existent gradients. In this chapter we bridge this gap by using *softsort* [96], a differentiable relaxation of the argsort operator, and integrating it into the iterations of two representative algorithms from this family: *Orthogonal Matching Pursuit* (OMP) [42, 93] and *Iterative Hard Thresholding* (IHT) [21, 52]. The resulting differentiable variants, that we call *Soft-OMP* and *Soft-IHT*, are shown theoretically and experimentally to be valid approximations of their non-differentiable counterparts.

4.1.1 Main contributions

Our main contributions are summarized as follows:

(1) We propose gradient-friendly versions of OMP and IHT by reinterpreting these algorithms through a permutation-based lens, approximating the permuation matrices associated with argsort using differentiable approximants that we derive from the softsort operator. To the best of our knowledge, this is the first time the non-differentiability of argsort-based operators in these algorithms has been directly addressed through a mathematical treatment rather than engineering-based deep learning tweaks.

(2) We rigorously analyze and experimentally demonstrate that Soft-OMP and Soft-IHT effectively approximate their original counterparts under suitable conditions on the softsort temperature parameter.

(3) We unroll iterations of these algorithms into neural networks and propose the *OMP-Net* and *IHT-Net* architectures. We show that these networks are trainable with meaningful parameters, i.e., weights that capture latent structure within the data.

(4) We demonstrate that with weights as trainable parameters, OMP- and IHT-Net enable near noise-level recovery in heavily undersampled regimes, provided that the data exhibits sufficient structure. In particular, they are able to outperform, respectively, OMP and IHT in this scenario.

4.1.2 A glimpse at the literature

The advantages of algorithm unrolling have made it a popular model-based deep learning approach. Several review papers (see, e.g., [81, 89, 103]) document this success story. Moreover, we highlight two particularly influential works, [61] and [37], that hold notable relevance to our study.

The work [37] unrolls iterations of a primal-dual algorithm [29] with recovery guarantees, thereby ensuring stability in neural networks while addressing fundamental limits of AI. On the other hand, [61], is one the earliest (if not the first) papers to explore algorithm unrolling, introducing the unrolling of ISTA (Iterative Shrinkage-Thresholding Algorithm) [40]. ISTA follows a recursive relation similar to IHT but employs soft-thresholding, defined as

$$S_\lambda(x)_j = \begin{cases} x_j - \lambda, & x_j \geq \lambda \\ 0 & -\lambda, < x < \lambda \\ x_j + \lambda, & x_j \leq -\lambda \end{cases},$$

which is a continuous operator and thus compatible with neural networks (see Remark 4.1). The introduction of Learned ISTA (LISTA) inspired extensive research in the field, including theoretical analyses [16, 124]. However, OMP and IHT have received comparatively less attention, due to the inherent non-differentiability of their argsort-based operators. Unrolling IHT appeared in two nearly simultaneous works [120, 122], considering IHT with both k-sparse and the ℓ^0-regularized hard-thresholding operators (teminology due to [21]). Unlike soft-thresholding, the ℓ^0-regularized component-wise operator defined as

$$H_\lambda(x)_j = \begin{cases} x_j, & x_j \geq \lambda \text{ or } x_j \leq -\lambda \\ 0, & -\lambda < x < \lambda \end{cases},$$

is discontinuous (and thus non-differentiable), but its implementation in neural networks is straightforward and was done in [120, 122] by using a continuous relaxation of the operator. However, these works do not directly address the non-differentiability of the k-sparse IHT, which includes argsort and is central to our work. The same holds for OMP in DeepPursuit, a deep reinforcement learning implementation of OMP [31], and in the learned greedy method [73]. The present chapter builds upon the authors' previous work [87], introducing Soft-IHT and IHT-Net as a complementary contribution, while extending the theoretical and numerical scope of our earlier proposal on Soft-OMP and OMP-Net.

4.1.3 Notation

Here, we introduce the necessary notations used throughout the chapter, while defining additional ones in context as needed. We use $[N]$ to denote the enumeration set $\{1, \dots, N\}$ and $[\,]$ to denote an empty matrix. $\|\cdot\|$ denotes the ℓ^2-norm of a vector. For a matrix $A \in \mathbb{C}^{m\times N}$, we define its range as $\mathcal{R}(A) = \{y \in \mathbb{C}^m : y = Ax,\ x \in \mathbb{C}^N\}$, its transpose as $A^\top \in \mathbb{C}^{N\times m}$, its conjugate transpose as $A^* \in \mathbb{C}^{N\times m}$, and its ℓ^2-operator norm as $\|A\| = \|A\|_{2\to 2} = \sup_{\|x\|=1} \|Ax\|$. We also adopt MATLAB-style notation, where $A(i,:)$ and $A(:,j)$ refer to the ith row and jth column of A, respectively. To distinguish between row and column representations, we write $A = [a_1, a_2, \dots]$ for a row-wise and $A = [a_1; a_2; \dots]$ for a column-wise arrangement.

The sth *Restricted Isometry Constant* (RIC) of A, denoted by $\delta_s(A)$, is the smallest $\delta > 0$ satisfying $(1-\delta)\|x\| \leq \|Ax\| \leq (1+\delta)\|x\|$, for all s-sparse vectors $x \in \mathbb{C}^N$, i.e., those with $\|x\|_0 = \mathrm{card}(\mathrm{supp}(x)) \leq s$, where $\mathrm{supp}(x)$ denotes the support of the vector x and $\mathrm{card}(\cdot)$ returns the cardinality of a set. For a set of indices S with $\mathrm{card}(S) = s$, we denote by $x_S \in \mathbb{C}^s$ the restriction of x to the indices in S, and by $A_S \in \mathbb{C}^{m\times s}$ the submatrix of $A \in \mathbb{C}^{m\times N}$ consisting of the columns indexed by S. When clear from context, we may use the same notation to refer to zero-padded version of x, where entries outside S (i.e., $[N]\backslash S$) are set to zero, preserving the original dimension. Additionally, the *coherence* parameter of A is given by $\mu(A) = \max_{i,j\in[N],\, i\neq j} |\langle a_i, a_j\rangle|$

where $a_j = A(:, j)$ for $j \in [N]$ (see, e.g., [55]).

Throughout the chapter, we use the tilde symbol '$\sim$' to denote variables in the "Soft" counterparts of OMP and IHT. For example, we represent the equivalent of x in OMP or IHT by $\tilde{x}$ in Soft-OMP or Soft-IHT.

4.1.4 Outline of the chapter

The rest of the chapter is organized as follows. The next section provides preliminaries on greedy sparse recovery algorithms, namely OMP and IHT, and formalizes the argsorting issue in their unrolling. Then, Section 4.3 introduces softsort and explains how this operator addresses the argsort challenge in OMP and IHT. In this section we also present Soft-OMP and Soft-IHT, along with theorems validating their approximation capabilities. To improve readability, proofs are deferred to Section 4.6, Section 4.7, and Section 4.8. Section 4.4 sets forth the neural network realization of Soft-$*$ algorithms. Numerical experiments validating our theoretical results are presented in Section 4.3, while their implications for the trainability of OMP- and IHT-Net are examined in Section 2.4. Finally, we conclude the chapter with remarks on future research directions in Section 4.9.

4.2 Preliminaries

In this section, we first contextualize the sparse recovery problem within our framework and provide an overview of the two greedy solvers that form the focus of this chapter. Next, we elaborate on the argsort operator, which plays a critical role in hindering the neural network realization of these algorithms. This discussion sets the stage for Section 4.3, where we present one of our key contributions: differentiable alternatives to these algorithms, compatible with neural network learning.

4.2.1 Sparse recovery

The goal of sparse recovery is to reconstruct an approximately sparse signal $x \in \mathbb{C}^N$, observed through a linear transformation represented by a matrix $A \in \mathbb{C}^{m \times N}$, and potentially corrupted by some additive source of error $e \in \mathbb{C}^m$. This involves recovering x given observations $y \in \mathbb{C}^m$ assuming a linear measurement model

$$y = Ax + e. \tag{78}$$

Depending on the context, A is referred to as the design, mixing, sensing or dictionary matrix, and e models numerical or noise error. In compressive sensing or sparse representation onto overcomplete dictionaries, the system in (78) is underdetermined, i.e., $m < N$ or $m \ll N$. One can formulate a first recovery method in terms of the following *non-convex* optimization problem

$$\hat{x} \in \operatorname*{argmin}_{z \in \mathbb{C}^N} \left\{ \|y - Az\|_2^2 \quad \text{subject to} \quad \|z\|_0 \leq k \right\}, \tag{79}$$

where k represents the desired sparsity level. Since the sparse recovery problem above is computationally intractable (see, e.g., [55]), one might pursue one of the following paths:

(1) Relax the non-convex ℓ^0-(pseudo) norm in favor of convex ℓ^1-norm. This approach falls within the realm of convex optimization, leading to basis pursuit in its constrained form or LASSO-type programs in its unconstrained form.

(2) Impose the desired sparsity implicitly or locally through iterations of an iterative algorithm. This approach gives rise to iterative greedy and thresholding algorithms.

In this chapter, we focus on the latter as notable alternatives for convex optimization programs, with particular emphasis on OMP [42, 93] and (k-sparse) IHT [21]. For reasons that will be discussed later, we classify IHT as a greedy algorithm, and consequently, study both OMP and IHT under the umbrella of greedy sparse recovery algorithms.

Before discussing these algorithms, we briefly note that we are also concerned with the setting where prior information about the signal is available. Such information can be incorporated into a vector of "weights" $w \in \mathbb{R}^N$, with $w \geq 0$, allowing emphasis to be placed on specific coefficients of the signal, thereby promoting certain structures within the signal (see, e.g., [11, 14, 56, 98] and references therein). With this in mind, in the following sections we will also discuss briefly how w can be effectively incorporated into OMP and IHT, extending them to their weighted versions, WOMP and WIHT respectively. These extensions enable the recovery process to use prior knowledge about the signal structure. Similar weighted extensions have been studied in [71, 72, 86].

4.2.2 Greedy algorithms

The core idea of a greedy strategy in solving computationally intractable problems (typically NP-hard) is to repeatedly select the local optimum when the global solution is unattainable within a feasible timeframe, which simply translates to "visit the nearest city!" in the well-known traveling salesman problem [13]. Similarly, since problem (79) is NP-hard (see, e.g., [55]), a greedy sparse recovery solver suggests to iteratively sparsify the solution at each stage. OMP incrementally builds a solution with the required sparsity level k, whereas IHT consistently enforces the same sparsity level throughout its iterations.

OMP

OMP is one of the most well-studied greedy algorithms for sparse recovery. It tackles problem (79) in a greedy fashion by decomposing the global optimization problem into a sequence of local least-squares optimization problems.

At each iteration, in its simplest form, OMP identifies one additional index to add to the signal's support based on a "greedy selection criterion". The algorithm then reconstructs the signal by iteratively updating the support and computing the corresponding signal coefficients. In this way, OMP ensures a one-to-one correspondence between the number of iterations and sparsity level of the reconstructed signal, effectively imposing the desired sparsity over the course of iterations. This property makes OMP particularly efficient in lower sparsity regimes, i.e., when s is small. The steps of OMP are outlined in Algorithm 4.1.

Algorithm 4.1 Orthogonal Matching Pursuit (OMP).

Input: desired sparsity k, mixing matrix $A \in \mathbb{C}^{m \times N}$ with ℓ^2-normalized columns, observation vector $y \in \mathbb{C}^m$.

```
Let $x^{(0)} = 0$ and $S^{(0)} = \emptyset$.
For $n = 0, \dots, k-1$ repeat:
    $S^{(n+1)} = S^{(n)} \cup \{j^{(n+1)}\}, \quad j^{(n+1)} \in \operatorname*{argmax}_{j \in [N]} \left\{ \left| \left( A^*(y - Ax^{(n)}) \right)_j \right| \right\},$    (OMP.1)
    $x^{(n+1)} \in \operatorname*{argmin}_{z \in \mathbb{C}^N} \left\{ \|y - Az\|_2^2 \quad \text{s.t.} \quad \operatorname{supp}(z) \subseteq S^{(n+1)} \right\}.$    (OMP.2)
```

Output: k-sparse vector $\hat{x} = x^{(k)} \in \mathbb{C}^N$.

Here, $S^{(n+1)}$ is the support of the signal at iteration $n+1$, with the initializations $x^{(0)} = 0$ and $S^{(0)} = \emptyset$. Throughout the chapter, we refer to $|A^*(y - Ax^{(n)})|$ as the "greedy selection quantity". At each iteration of OMP, the greedy selection step introduces a competition among the column indices of A to join the support. The column with the strongest correlation to the residual $r^{(n)} := y - Ax^{(n)}$ wins the competition and its index is added to the support.

If any sort of prior knowledge about the signal is available, OMP can be extended to the weighted setting by appropriately modifying the greedy selection quantity, and introducing a degree of "unfairness" in the competition. A semantic way of achieving this is to embed a weight vector (typically with values in the range $[0, 1]$) as the diagonal entries of a diagonal matrix

$$W = \operatorname{diag}(w) \in \mathbb{R}^{N \times N}, \text{ where } w \in \mathbb{R}^N,\ w \geq 0. \tag{80}$$

The modified greedy selection quantity then becomes $|WA^*(y - Ax^{(n)})|$, effectively prioritizing certain indices over others. This idea has been explored in prior works [23, 78].

IHT

k-sparse IHT was introduced in [21] from an optimization transfer perspective, but can also be viewed as a fixed-point algorithm. For our purpose, however, it is more convenient to interpret IHT as combining two steps: (1) a descent step on the least-square loss function $f(x) = \frac{1}{2}\|y - Ax\|_2^2$, performed via a Bregman-type iteration, and (2) a best k-term approximation step to enforce sparsity.

In the descent step, IHT moves the solution towards the gradient descent direction with an appropriate amount determined by a step size $\eta > 0$, resulting in

$$u^{(n+1)} = x^{(n)} - \eta \nabla_x f(x)\big|_{x = x^{(n)}} = x^{(n)} + \eta A^*(y - Ax^{(n)}).$$

However, $u^{(n+1)}$ is generally non-sparse and thus, to impose the desired sparsity, IHT performs a greedy step via the best k-term approximation

$$x^{(n+1)} \in \operatorname*{argmin}_{z \in \mathbb{C}^N,\ \|z\|_0 \le k} \|z - u^{(n+1)}\|_2^2,$$

which explains why we place k-sparse IHT as a greedy algorithm alongside OMP. This step is equivalent to applying the hard-thresholding operator $H_k : \mathbb{C}^N \to \mathbb{C}^N$, that retains the top k entries of $u^{(n+1)}$ (in absolute value) and sets the rest to zero. More formally, let $\rho : [N] \to [N]$ be a bijection that sorts elements of $|u^{(n+1)}|$ in non-increasing order $|u^{(n+1)}_{\rho(1)}| \ge |u^{(n+1)}_{\rho(2)}| \ge \cdots \ge |u^{(n+1)}_{\rho(N)}|$. The support of $x^{(n+1)}$ is then given by $S^{(n+1)} = \{\rho(1), \ldots, \rho(k)\}$, and the update becomes

$$x^{(n+1)} = H_k(u^{(n+1)}) = u^{(n+1)}_{S^{(n+1)}}.$$

Combining these two steps, k-sparse IHT is summarized in Algorithm 4.2.

Algorithm 4.2 Iterative Hard Thresholding (IHT).

Input: desired sparsity k, mixing matrix $A \in \mathbb{C}^{m \times N}$ with ℓ^2-normalized columns, observation vector $y \in \mathbb{C}^m$, initial signal $x^{(0)} \in \mathbb{C}^N$, step size $\eta > 0$, number of iterations $\bar{n}$.

```
For $n = 0, \ldots, \bar{n} - 1$ repeat:
    $x^{(n+1)} = H_k(x^{(n)} + \eta A^*(y - Ax^{(n)})).$        (IHT)
```

Output: k-sparse vector $\hat{x} = x^{(\bar{n})} \in \mathbb{C}^N$.

To emphasize or suppress specific entries based on provided prior knowledge on the signal, one can replace the hard-thresholding operator $H_k(\cdot)$ with the weighted hard-thresholding operator $H_{w,k}(\cdot)$. This operator relies on the bijection $\rho : [N] \to [N]$ associated this time with the non-increasing rearrangement of $|Wu^{(n+1)}|$, where W is the diagonal matrix defined in Equation (80), with the ordering $|w_{\rho(1)} u^{(n+1)}_{\rho(1)}| \ge |w_{\rho(2)} u^{(n+1)}_{\rho(2)}| \ge \cdots \ge |w_{\rho(N)} u^{(n+1)}_{\rho(N)}|$. As in the standard case, the support of $x^{(n+1)}$ is then defined as $S^{(n+1)} = \{\rho(1), \ldots, \rho(k)\}$, and the update becomes $x^{(n+1)} = H_{w,k}(u^{(n+1)}) = u^{(n+1)}_{S^{(n+1)}}$. It is worth noting that this non-increasing rearrangement corresponds to the weighted ℓ^1-norm, but this does not violate the generality in our case as ℓ^1 and ℓ^2 norms lead to the same best s-term approximation and ℓ^2-norm coincides with the weighted ℓ^2-norm (see Chapter 3). The idea of weighted IHT has been previously proposed in [71], albeit with a technical variation regarding weighted cardinality, which we omit purposefully to avoid unnecessary complications.

4.2.3 The (arg)sorting issue in unrolling greedy algorithms

Algorithm unrolling is the process of mapping iterations of an algorithm onto the layers of a neural network, with each layer playing the role of one iteration of the original algorithm. Treating certain parameters of the neural network as trainable, the objective of algorithm unrolling is to optimize the performance of the algorithm with respect to those parameters through a gradient-based learning process for the neural network, possibly with fewer layers (iterations) than the original algorithm, thereby reducing the computational cost. This paradigm offers significant benefits on both the neural network and algorithmic fronts. Since algorithms typically stem from well-defined models, they bring interpretability to the neural network. Moreover, if the original algorithm enjoys recovery and convergence guarantees, these properties can often be transferred to the neural network. This alleviates common concerns about neural networks, such as their black-box nature, potential instability and lack of interpretability [37, 74].

Despite the appeal of algorithm unrolling, not all algorithms can be readily unrolled. Key elements acting against "unrollability" of some algorithms include the presence of non-differentiable operators (see Remark 4.1) and implicit dependencies between the algorithm's steps, which hinders the flow of gradients during backpropagation in gradient-based learning. Notably, greedy algorithms like OMP and IHT are illustrative examples of these challenges as they rely on the operator argsort.

More precisely, for an input vector $v \in \mathbb{R}^N$ the operators sort : $\mathbb{R}^N \to \mathbb{R}^N$ and argsort : $\mathbb{R}^N \to \mathcal{S} \subset [N]^N$ (assuming non-increasing ordering) are defined as follows:

$$\begin{aligned} &\mathrm{sort}(v) = (v_{\varrho_1}, v_{\varrho_2}, \ldots, v_{\varrho_N}), \quad v_{\varrho_1} \geq v_{\varrho_2} \geq \cdots \geq v_{\varrho_N}, \\ &\mathrm{argsort}(v) = (\varrho_1, \varrho_2, \ldots, \varrho_N), \end{aligned} \tag{81}$$

where $\mathcal{S}$ is the set of all permutations of $[N]$. Figure 4.1 graphically illustrates these operators in a toy example with $N = 2$ and $v_2 = 1$ fixed. Throughout the chapter, we assume there are no ties between the elements of the vector v, i.e., $v_i \neq v_j,\ i \neq j$ for all $i, j = 1, \ldots, N$. This assumption is justified by the observation that, when running OMP and IHT, ties are very rare in practice due to the presence of numerical round-off errors.

The argmax operator in OMP and the hard-thresholding operator in IHT both depend on argsort, which is a discontinuous operator with abrupt kinks and jumps, rendering these algorithms entirely non-differentiable. These algorithms involve the crucial step $\mathcal{J}^{(n+1)} = \mathrm{argsort}(v^{(n+1)})$ within their iterations, where $v^{(n+1)}$ is the vector to be sorted, i.e., $v^{(n+1)} = |A^*(y - x^{(n)})|$ and $v^{(n)} = |u^{(n+1)}|$ for OMP and IHT, respectively. IHT selects the first k indices of $\mathcal{J}^{(n+1)}$ and passes the corresponding entries to $x^{(n+1)}$. This breaks the gradient path of $x^{(n+1)}$ with respect to any parameter (e.g., η), due to the argsort operator. This issue is even more pronounced for OMP. After computing $\mathcal{J}^{(n+1)}$, it adds the first index to the current support $S^{(n)}$ to form the updated support $S^{(n+1)}$. The next step solves a least-squares problem restricted to the indices in $S^{(n+1)}$. This introduces not only non-differentiability due to application of the argsort operator, but also an implicit connection between steps of the algorithm, which is doubly problematic.

Remark 4.1 (Algorithmic differentiability and subgradients)**.** *In neural network optimization, certain types of non-differentiability such as corner singularities, are often manageable by employing subgradient generalizations, i.e., substituting the gradient at the non-differentiable point with an appropriate subgradient. Examples are functions such as $f(x) = |x|$ and $f(x) = ReLU(x)$ at $x = 0$ or $\mathrm{sort}(x)$ as shown in Figure 4.1. However, the more problematic form of non-differentiability in neural networks is discontinuity, as seen in the argsorting operator (see Figure 4.1), which cannot be treated using such standard remedies.*

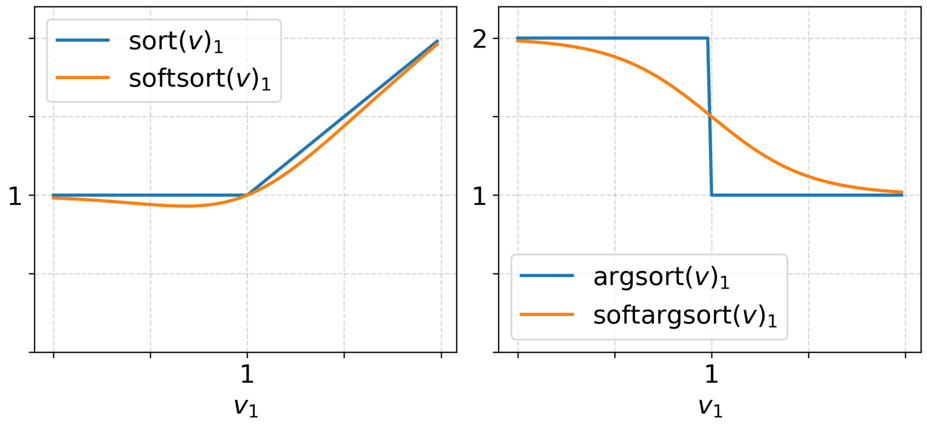

Figure 4.1: First element of the operators sort, softsort, argsort and softargsort applied to the two-dimensional vector $v = (v_1, v_2)$, as defined in Equations (81) to (83), shown as a function of v_1 for fixed $v_2 = 1$ ($\tau = 0.25$).

4.3 Unrolled greedy algorithms

As seen previously, the primary obstacle hindering unrolling greedy algorithms is the presence of argsort operator within their iterations. However, to address this challenge we must first make the connections between algorithmic steps explicit to preserve gradient flow across iterations. The "missing link" to bridging non-differentiable algorithms with implicit steps and differentiable algorithms with explicit gradient path lies in the notion of permutation, intrinsic to (arg)sorting.

More formally, (arg)sorting can be characterized by a permutation matrix $P_{\mathrm{argsort}(\cdot)} \in \mathbb{R}^{N\times N}$, as for a vector $v \in \mathbb{R}^N$

$$\begin{aligned} &\mathrm{sort}(v) = P_{\mathrm{argsort}(v)}v, \quad \mathrm{argsort}(v) = P_{\mathrm{argsort}(v)}\bar{1}_N, \\ &P_{\mathrm{argsort}(v)}(i,j) = \begin{cases} 1, & j = \mathrm{argsort}(v)_i, \\ 0, & \text{otherwise} \end{cases}, \end{aligned} \tag{82}$$

where $\bar{1}_N = (1, \ldots, N)^\top$ is the vector corresponding to the set $[N]$. Note that the matrix operator $P_{\mathrm{argsort}(\cdot)}$ is well-defined for a fixed permutation in $\mathbb{R}^N$; otherwise, argsort is fundamentally a nonlinear operator.

Reinterpreting steps of OMP and IHT through a projection-based lens allows us to explicitly express algorithmic steps in terms of permutation matrices, but this alone does not resolve the non-differentiability issue. The key remaining task is to replace these permutation matrices with differentiable proxies that approximate them with a controllable degree of accuracy. In what follows, we introduce a reasonable choice for this proxy—*softsort*—and use it to construct differentiable approximations of OMP and IHT, which we theoretically justify as valid alternatives to the exact algorithms.

4.3.1 Softsorting

Softsort, introduced in [96], provides a continuous relaxation of the argsort operator (see Figure 4.1) by approximating $P_{\text{argsort}(\cdot)}$ with a permutation matrix $\tilde{P}_{\text{argsort}(\cdot)}$. For a column vector $v \in \mathbb{R}^N$, softsort is defined as

$$\tilde{P}_{\text{argsort}(v)} = \text{softsort}_\tau(v) = \text{softmax}\left(\frac{-|\operatorname{sort}(v)\mathbb{1}^\top - \mathbb{1}v^\top|}{\tau}\right), \tag{83}$$

where the softmax function, applied row-wise is given by

$$\text{softmax}(a)_j = \frac{e^{a_j}}{\sum_{i=1}^N e^{a_i}}, \quad j \in [N], \quad a \in \mathbb{R}^N,$$

$\mathbb{1} = (1, \ldots, 1)^\top \in \mathbb{R}^N$ is an all-one vector, and τ is a *temperature parameter* that controls the approximation accuracy. Softsort applies softmax on the negative distance matrix derived from v and $\operatorname{sort}(v)$, emphasizing locations where v and $\operatorname{sort}(v)$ coincide, corresponding to the positions of elements in $\operatorname{argsort}(v)$ (see Proposition 4.3(d)). With sufficiently small τ, softsort produces dominant values at these positions and small values at others, effectively approximating the discrete argsort operator. This is exemplified below.

Example 4.2. *Let* $x = (3, 4, 2, 1)^\top$.

$$P_{\text{argsort}(x)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$
$$\operatorname{sort}(x) = P_{\text{argsort}(x)} x = (4, 3, 2, 1)^\top,$$
$$\operatorname{argsort}(x) = P_{\text{argsort}(x)} \bar{1}_N = (2, 1, 3, 4)^\top,$$

which for $\tau = 0.5$ *yields the approximation*

$$\tilde{P}_{\text{argsort}(x)} = \text{softsort}(x, \tau = 0.5) = \begin{bmatrix} 0.1171 & \mathbf{0.8650} & 0.0158 & 0.0021 \\ \mathbf{0.7758} & 0.1050 & 0.1050 & 0.0142 \\ 0.1050 & 0.0142 & \mathbf{0.7758} & 0.1050 \\ 0.0158 & 0.0021 & 0.1171 & \mathbf{0.8650} \end{bmatrix},$$
$$\tilde{P}_{\text{argsort}(x)} x = (3.8448, 2.9716, 2.0284, 1.1552)^\top,$$
$$\tilde{P}_{\text{argsort}(x)} \bar{1}_N = (1.9031, 1.3576, 2.8808, 3.8311)^\top.$$

Softsort as in Equation (83) is an instance of a “unimodal row-stochastic matrix”. It possesses several desirable properties listed below.

Proposition 4.3 (Properties of the softsort operator [96]). *Softsort satisfies:*

(a) Non-negativity: $\tilde{P}_{\text{argsort}(v)}(i, j) \geq 0$.

(b) Row affinity: Because of softmax, elements in each row of $\text{softsort}_\tau(v)$ *sum to 1, i.e.,* $\sum_{j=1}^N \tilde{P}_{\text{argsort}(v)}(i, j) = 1,\ \forall i \in [N]$.

(c) *Asymptotic behavior:* $\mathrm{softsort}_\tau(v) \to P_{\mathrm{argsort}(v)}$, *as* $\tau \to 0$.

(d) *Argmax property:* $\mathrm{argsort}(v) = \mathrm{argmax}(\tilde{P}_{\mathrm{argsort}(v)})$, *where argmax is applied row-wise.*

(e) *Permutation equivariance:* $\mathrm{softsort}_\tau(v) = \mathrm{softsort}_\tau(\mathrm{sort}(v))P_{\mathrm{argsort}(v)}$, *which can be viewed as a direct consequence of (d).*

In addition to the above properties, softsort is also a continuous operator. While continuity of softsort was assumed without formal discussion in [96], we establish a stronger result in Proposition 4.4 below. Its proof can be found in Section 4.6.

Proposition 4.4 (Lipschitzness of softsort). $\mathrm{softsort}_\tau(\cdot) : \mathbb{R}^N \to \mathbb{R}^{N\times N}$ *is a continuous function. Moreover, the function defined by each row of the softsort operator, i.e.,* $v \in \mathbb{R}^N \mapsto \mathrm{softsort}_\tau(v)(i,:)$, *is L-Lipschitz, where* $L = (\sqrt{N}+1)/\tau$.

Continuity of softsort, and hence its algorithmic differentiability (see Remark 4.1), enables neural network implementations of the argsort operator. Building on this foundation, in the following subsections we use the approximate permutation matrix obtained from softsort to unfold OMP and IHT, respectively. However, the Lipschitz constant L in the proposition above tends to infinity as $\tau \to 0$, indicating that continuity of softsort deteriorates for small values of τ. This introduces a trade-off for τ between the continuity of softsort and its approximation, as highlighted in Proposition 4.3(c), which states that softsort more closely approximates the true sorting operator as τ decreases. Thus, selecting an appropriate τ is crucial in balancing theory and practice in the upcoming sections. Theoretical constraints impose an upper bound for τ, beyond which the desired approximation is not achieved, while practical considerations in training the neural networks demand for higher values of τ (see Section 2.4).

4.3.2 Soft-OMP

The first step to unroll OMP (see Algorithm 4.1) is to reinterpret its iterations through a projection (permutation-based) lens. This not only enables the direct application of soft sorting but also explicitly reveals the connection between successive OMP iterations, facilitating efficient gradient flow.

At iteration $n+1$, OMP computes the projection of the observation signal y onto an $(n+1)$-dimensional subspace of $\mathcal{R}(A)$, spanned by the columns iteratively selected during the greedy selection step, i.e., $A_{S^{(n+1)}} \in \mathbb{C}^{m\times(n+1)}$. This subspace expands over iterations, progressively improving the approximation. This process is equivalent to applying a permutation matrix $\Pi^{(n+1)} \in \mathbb{R}^{(n+1)\times N}$ to the column space of A, yielding $A(\Pi^{(n+1)})^\top =: B^{(n+1)}$, where rows of $\Pi^{(n+1)}$ correspond to $n+1$ selected indices in $S^{(n+1)}$, i.e., in each row there is a 1 at the location of the selected index and 0 elsewhere. Since OMP selects one additional column at each iteration, the permutation matrix $\Pi^{(n+1)}$ is constructed incrementally by appending a new row to the previous iteration's permutation matrix $\Pi^{(n+1)} = [\Pi^{(n)}; P^{(n+1)}(1,:)]$, where $P^{(n+1)} := P_{\mathrm{argsort}(v^{(n+1)})}$ (as in Equation (82)), with $v^{(n+1)} = |A^*(y - Ax^{(n)})|$. Once this subspace is established, the next step is to solve the least-squares problem within it, followed by mapping the resulting $(n+1)$-dimensional signal back to the original N-dimensional space. These steps define the following algorithm, which we term *pOMP* (projection-based OMP), serving as an intermediary between OMP and its unrolled counterpart.

Algorithm 4.3 Projection-based OMP (pOMP).

Input: desired sparsity k, mixing matrix $A \in \mathbb{C}^{m \times N}$ with ℓ^2-normalized columns, observation vector $y \in \mathbb{C}^m$.

Let $x^{(0)} = 0$ and $\Pi^{(0)} = [\,]$.
For $n = 0, \dots, k-1$ repeat:

$$\begin{cases} v^{(n+1)} = \left|A^*(y - Ax^{(n)})\right|, \\ P^{(n+1)} = P_{\text{argsort}(v^{(n+1)})}, \\ \Pi^{(n+1)} = \left[\Pi^{(n)}; P^{(n+1)}(1,:)\right], \end{cases} \tag{pOMP.1}$$

$$\begin{cases} B^{(n+1)} = A(\Pi^{(n+1)})^\top, \\ w^{(n+1)} \in \operatorname{argmin}_{z \in \mathbb{C}^{(n+1)}} \|y - B^{(n+1)}z\|_2^2, \\ x^{(n+1)} = (\Pi^{(n+1)})^\top w^{(n+1)}. \end{cases} \tag{pOMP.2}$$

Output: k-sparse vector $\hat{x} = x^{(k)} \in \mathbb{C}^N$.

It is evident that pOMP generates the same sequence of output signals as OMP. In pOMP, the relationship between the vector to be sorted, $v^{(n+1)}$, and the output signal at that iteration, $x^{(n+1)}$, is established through the permutation matrix $\Pi^{(n+1)}$ (and $P^{(n+1)}$). The key obstacle to differentiability is the sorting-based selection operator $P_{\text{argsort}(\cdot)}$, which is inherently non-differentiable and blocks gradient flow. To address this, we approximate the permutation matrix $P_{\text{argsort}(\cdot)}$ in a differentiable manner by replacing it with a softsort proxy. This modification immediately yields a differentiable approximation of OMP, which we refer to as *Soft-OMP*, because of the softsort operator.

Remark 4.5. *Although we choose softsort for its close-form and intuitive formulation, it is worth noting that the core idea in this chapter is to provide a suitable differentiable approximation for* $P_{\text{argsort}(\cdot)}$ *and replacing it with the sofsort operator is just one of several possible techniques. See Section 4.9, for other alternatives.*

Remark 4.6. *Since each iteration of Soft-OMP requires only the first row of the softsort matrix, this reduces to simply applying softmax to the vector* $\tilde{v}^{(n+1)}$ *because*

$$\tilde{P}^{(n+1)}(1,:) = \text{softsort}_\tau(\tilde{v}^{(n+1)})(1,:) = \text{softmax}\left(\frac{-\left|\max_{j\in[N]}(\tilde{v}^{(n+1)})\mathbb{1} - \tilde{v}^{(n+1)}\right|}{\tau}\right),$$

$$= \text{softmax}\left(\frac{\tilde{v}^{(n+1)} - \max_{j\in[N]}(\tilde{v}^{(n+1)})\mathbb{1}}{\tau}\right) = \text{softmax}\left(\frac{\tilde{v}^{(n+1)}}{\tau}\right),$$

where the last step holds thanks to the invariance of softmax when adding a constant to all elements of its input. Similar arguments can be applied to the more general case of $\tilde{P}^{(n+1)}(1:k,:)$ *leading to more memory-efficient and computationally optimized implementations.*

How well can Soft-OMP approximate OMP? As noted earlier, softsort approximates $P_{\text{argsort}(\cdot)}$

Algorithm 4.4 Soft-OMP.

Input: desired sparsity k, mixing matrix $A \in \mathbb{C}^{m\times N}$ with ℓ^2-normalized columns, observation vector $y \in \mathbb{C}^m$, temperature parameter τ of softsort.

```
Let $\tilde{x}^{(0)} = 0$ and $\tilde{\Pi}^{(0)} = [\,]$.
For $n = 0, \dots, k-1$ repeat:
```

$$\begin{cases} \tilde{v}^{(n+1)} = \left|A^*(y - A\tilde{x}^{(n)})\right|, \\ \tilde{P}^{(n+1)} = \mathrm{softsort}_\tau(\tilde{v}^{(n+1)}), \\ \tilde{\Pi}^{(n+1)} = \left[\tilde{\Pi}^{(n)}; \tilde{P}^{(n+1)}(1,:)\right], \end{cases} \tag{Soft-OMP.1}$$

$$\begin{cases} \tilde{B}^{(n+1)} = A(\tilde{\Pi}^{(n+1)})^\top, \\ \tilde{w}^{(n+1)} \in \mathrm{argmin}_{z\in\mathbb{C}^{(n+1)}} \|y - \tilde{B}^{(n+1)}z\|_2^2, \\ \tilde{x}^{(n+1)} = (\tilde{\Pi}^{(n+1)})^\top \tilde{w}^{(n+1)}. \end{cases} \tag{Soft-OMP.2}$$

Output: Approximately k-sparse vector $\hat{x} = \tilde{x}^{(k)} \in \mathbb{C}^N$.

with accuracy controlled by its temperature parameter τ. The following theorem formalizes this idea, relating Soft-OMP to pOMP (and consequently OMP) under an appropriate condition on τ.

Theorem 4.7 (Soft-OMP is a good approximation to OMP)**.** *Let $v^{(n)} \in \mathbb{R}^N$, $\Pi^{(n)} \in \mathbb{R}^{n\times N}$ and $x^{(n)} \in \mathbb{C}^N$ be sequences generated by Algorithm 4.3; and likewise $\tilde{v}^{(n)} \in \mathbb{R}^N$, $\tilde{\Pi}^{(n)} \in \mathbb{R}^{n\times N}$ and $\tilde{x}^{(n)} \in \mathbb{C}^N$ be sequences generated by Algorithm 4.4. Then, the following hold:*

(i) Asymptotic convergence: As $\tau \to 0$, we have $\tilde{\Pi}^{(n)} \to \Pi^{(n)}$ and $\tilde{x}^{(n)} \to x^{(n)}$.

(ii) Non-asymptotic convergence: Choose

$$0 < \epsilon < g^{(1:n)}/2\|A\|,$$

where

$$\begin{aligned} g^{(1:n)} &:= \min_{i\in[n]} g^{(i)}, \\ g^{(i)} &:= \min_{j\in[N], j\neq j^{(i)}} \left|v_j^{(i)} - v_{j^{(i)}}^{(i)}\right| \text{ with } j^{(i)} := \mathrm{argmax}(v^{(i)}), \end{aligned} \tag{84}$$

are OMP's "global min-gap" and "local min-gap", respectively. Then

$$\mathrm{argmax}(\Pi^{(n)}) = \mathrm{argmax}(\tilde{\Pi}^{(n)}),$$

where argmax is applied row-wise.

Moreover, assume that τ, the temperature parameter of softsort, satisfies the condition

$$\tau \le \left(g^{(1:k)} - 2\|A\|\epsilon\right) / \log\left(C^{(k)}/\epsilon\right),$$

where

$$C^{(n)} = \sqrt{2n}(N-1)\left(\frac{\sqrt{1-\delta_n(A)} + (\sqrt{n}+1)\|A\|}{1-\delta_n(A)}\right)\|y\|_2,$$

and $\delta_n(A) < 1$ is the nth restricted isometry constant of A. Then

$$\max_{i\in[0:n]} \left\|x^{(i)} - \tilde{x}^{(i)}\right\| \leq \epsilon.$$

Proof (sketch). We present a proof sketch here. For the complete proof, see Section 4.7.

- We first ensure that Soft-OMP selects the same indices as OMP. Given the same initialization, $x^{(0)} = \tilde{x}^{(0)}$, we establish conditions under which Soft-OMP preserves the order of elements across iterations (in fact, only the maximum is sufficient). This guarantees that $\text{argmax}(\Pi^{(n)}) = \text{argmax}(\tilde{\Pi}^{(n)})$.

- Once this is established, we formalize approximation of $\Pi^{(n)}$ by $\tilde{\Pi}^{(n)}$, providing upper bounds for $\|\Pi^{(n)} - \tilde{\Pi}^{(n)}\|$ and $\|\tilde{\Pi}^{(n)}\|$, in terms of τ in Lemma 4.12.

- Finally, we relate $\|x^{(n)} - \tilde{x}^{(n)}\|$ to the bound derived for $\|\Pi^{(n)} - \tilde{\Pi}^{(n)}\|$, analyzing the deviations introduced in $B^{(n)} = A_{S^{(n)}} = A(\Pi^{(n)})^\top$ through the sensitivity analysis of least-squares, establishing this in Lemmas 4.11 and 4.13. We then prove the claim by induction on n.

□

In words, Theorem 4.7 states that to achieve a desired accuracy ϵ, the gap between elements of $v^{(i)}$ must be sufficiently large. This ascertains that the maximum index of $v^{(i)}$ at each iteration of Soft-OMP matches that of OMP, meaning Soft-OMP retrieves the same indices as OMP, i.e., $\text{argmax}(\Pi^{(n)}) = \text{argmax}(\tilde{\Pi}^{(n)})$ (Soft-OMP follows OMP's indices). Furthermore, if τ is small enough relative to the gap between elements of Soft-OMP, the column selector $\tilde{\Pi}^{(n+1)}$ will closely approximate $\Pi^{(n+1)}$, the column selector in OMP. As a result, the signal recovered at each iteration of Soft-OMP remains within an ϵ-distance of the signal obtained by OMP, ensuring that Soft-OMP approximates OMP to the desired ϵ-accuracy. We conclude by making a few remarks.

Remark 4.8. *We note that $g^{(i)}$, $\forall i \in [n]$, is not a tunable parameter but rather depends on the underlying "physics" of the problem, including the values of A, x, and y (see Equation (78)). In the first iteration, if OMP and Soft-OMP are initialized with the same signal, i.e., $x^{(0)} = \tilde{x}^{(0)}$, it suffices for $g^{(i)}$ to be strictly positive. However, as iterations progress, approximation errors accumulate, requiring $g^{(i)}$ to be sufficiently large for Soft-OMP to continue closely following OMP.*

Remark 4.9. *In light of recovery guarantee results available in the literature for OMP (see, e.g., [125]) and the inequality $\|\tilde{x}^{(n)} - x\| \leq \|\tilde{x}^{(n)} - x^{(n)}\| + \|x^{(n)} - x\|$, where $x \in \mathbb{C}^N$ is the true signal in Equation (78), Theorem 4.7 also implies a recovery guarantee for Soft-OMP.*

4.3.3 Soft-IHT

To unroll IHT (see Algorithm 4.2), we follow the same approach as for OMP and introduce a projection-based variant for IHT called *pIHT*. We define $Q^{(n+1)} = P_{\text{argsort}(v^{(n+1)})} \in \mathbb{R}^{N\times N}$, where $v^{(n+1)} := |u^{(n+1)}|$ is the quantity to be sorted and $u^{(n+1)} = x^{(n)} + \eta A^*(y - Ax^{(n)})$ defining each iteration of IHT. Then the first k rows of $Q^{(n+1)}$ determine the vectors $\{e_{\rho(i)}\}_{i=1}^k$, spanning the k-dimensional signal subspace $\Sigma_k^N \subset \mathbb{R}^N$, where $e_{\rho(i)}$ is a canonical (one-hot) vector with 1 at position $\rho(i)$ and 0 elsewhere, and $\rho : [N] \to [N]$ is the bijection defined by the hard thresholding operator $H_k(\cdot)$ introduced in Section 4.2.2. Consequently, the output of the hard thresholding operator simply expands onto these vectors using the top-k elements of $u^{(n+1)}$ in magnitude as expansion coefficients

$$x^{(n+1)} = H_k(u^{(n+1)}) = \sum_{i=1}^k u_{\rho(i)}^{(n+1)} Q^{(n+1)}(i,:).$$

This sum can be expressed more compactly as $x^{(n+1)} = q^{(n+1)} \odot u^{(n+1)}$, where $\odot$ denotes the Hadamard (or componentwise) product and

$$q^{(n+1)} = \sum_{i=1}^k Q^{(n+1)}(i,:),$$

the filter (0/1 mask) that selects the top-k elements of $u^{(n+1)}$ in magnitude. In other words, $H_k(\cdot)$ acts as a multiplication operator characterized by $q^{(n+1)}$. This leads to the following algorithm.

Algorithm 4.5 Projection-based IHT (pIHT).

Input: desired sparsity k, mixing matrix $A \in \mathbb{C}^{m\times N}$ with ℓ^2-normalized columns, observation vector $y \in \mathbb{C}^N$, initial signal $x^{(0)} \in \mathbb{C}^N$, step size $\eta > 0$, number of iterations $\bar{n}$.

```
Let $Q^{(0)} = [\,]$.
For $n = 0, \dots, \bar{n} - 1$ repeat:
```

$$\begin{cases} u^{(n+1)} = x^{(n)} + \eta A^*(y - Ax^{(n)}) \\ Q^{(n+1)} = P_{\text{argsort}(v^{(n+1)})}, \quad v^{(n+1)} = \left|u^{(n+1)}\right| \\ q^{(n+1)} = \sum_{i=1}^k Q^{(n+1)}(i,:) \\ x^{(n+1)} = q^{(n+1)} \odot u^{(n+1)}. \end{cases} \tag{pIHT}$$

Output: k-sparse vector $\hat{x} = x^{(\bar{n})} \in \mathbb{C}^N$.

Similar to OMP, achieving a neural network-compatible implementation of IHT requires replacing the permutation matrix $Q^{(n+1)}$ in Algorithm 4.5 with an approximate permutation matrix computed via softsort. In doing so, the rows of the softsort matrix serve as approximate canonical vectors, with the first k vectors spanning an anisotropic N-dimensional subspace of $\mathbb{R}^N$, stretched in the directions $\rho(1), \dots, \rho(k)$ (assuming τ is not too large), thanks to Proposition 4.3(d). The resulting differentiable version of IHT, which we refer to as *Soft-IHT*, is presented below.

Algorithm 4.6 Soft-IHT.

Input: desired sparsity k, mixing matrix $A \in \mathbb{C}^{m\times N}$ with ℓ^2-normalized columns, observation vector $y \in \mathbb{C}^N$, initial signal $\tilde{x}^{(0)} \in \mathbb{C}^N$, step size $\eta > 0$, number of iterations $\bar{n}$.

```
Let $\tilde{Q}^{(0)} = [\,]$.
For $n = 0, \dots, \bar{n}-1$ repeat:
```

$$\begin{cases} \tilde{u}^{(n+1)} = \tilde{x}^{(n)} + \eta A^*(y - A\tilde{x}^{(n)}), \\ \tilde{Q}^{(n+1)} = \text{softsort}_\tau(\tilde{v}^{(n+1)}), \quad \tilde{v}^{(n+1)} = \left|u^{(n+1)}\right| \\ \tilde{q}^{(n+1)} = \sum_{i=1}^k \tilde{Q}^{(n+1)}(i,:), \\ \tilde{x}^{(n+1)} = \tilde{q}^{(n+1)} \odot \tilde{u}^{(n+1)}, \end{cases} \tag{Soft-IHT}$$

Output: Approximately k-sparse vector $\hat{x} = \tilde{x}^{(\bar{n})} \in \mathbb{C}^N$.

Just like $q^{(n+1)}$, $\tilde{q}^{(n+1)}$ also represents a multiplication operator that acts as a mask whose dominant entries correspond to top values of the signal. The performance of Soft-IHT relative to IHT depends on how well this mask approximates the true mask. Similar to the OMP case, the following theorem quantifies this approximation in terms of a condition on τ, the temperature parameter of softsort.

Theorem 4.10 (Soft-IHT is a good approximation for IHT)**.** *Let $x^{(n)} \in \mathbb{C}^N$, $v^{(n)} \in \mathbb{R}^N$ and $Q^{(n)} \in \mathbb{R}^{N\times N}$ be sequences generated by Algorithm 4.5, and $\tilde{x}^{(n)} \in \mathbb{C}^N$, $\tilde{v}^{(n)} \in \mathbb{R}^N$ and $\tilde{Q}^{(n)} \in \mathbb{R}^{N\times N}$ be sequences generated by Algorithm 4.6. Further, assume that $x^{(0)} = \tilde{x}^{(0)}$. Then, the following hold:*

(i) Asymptotic convergence: As $\tau \to 0$, we have $\tilde{Q}^{(n)} \to Q^{(n)}$ and $\tilde{x}^{(n)} \to x^{(n)}$.

(ii) Non-asymptotic convergence: Choose

$$0 < \epsilon < g^{(1:n)}/2L,$$

with $L = \|I - A^*A\|$ and $g^{(1:n)}$ defined as

$$g^{(1:n)} := \min_{k\in[n]} g^{(k)}, \quad g^{(k)} := \min_{i,j\in[N],i\neq j} \left|v_i^{(k)} - v_j^{(k)}\right|, \tag{85}$$

representing IHT's "global min-gap" and "local min-gap", respectively. If τ, the temperature parameter of softsort, satisfies the condition

$$\tau \le \left(g^{(1:n)} - 2L\epsilon\right) / \log(C^{(n)}/\epsilon),$$

where

$$C^{(n)} = 2sN\frac{(sL)^n - 1}{sL - 1}\left(\|y\| + s\mu \max_{1\le k\le n-1} \|x^{(k)}\|\right),$$

and $\mu(A)$ is the coherence parameter of the matrix A, then

$$\max_{i\in[0:n]} \left\| x^{(i)} - \tilde{x}^{(i)} \right\| \leq \epsilon.$$

Proof (sketch). The proof of this theorem is derived through a perturbation analysis of a single IHT iteration, ensuring that a single iteration of Soft-IHT follows IHT in maximal indices passed through the next iteration and approximation error (Lemma 4.14). This result is then employed in an induction to extend the result to all previous iterations down to the first iteration. For the full proof, see Section 4.8. □

Similar to the conclusions drawn in the OMP case, achieving smaller gaps (inherent to the model) and a higher approximation accuracy requires smaller values of τ. Moreover, Remarks 4.8 and 4.9 remain valid in the IHT case as well.

4.4 Greedy networks

Algorithms 4.4 and 4.6 yield good approximations of Algorithms 4.1 and 4.2, respectively, achieving an accuracy level prescribed by Theorems 4.7 and 4.10 and regulated by τ. However, this approximation comes with some accuracy loss due to replacing the exact argsort operator in Equation (81) with the approximate softsort operator in Equation (83). This approximation error can be tolerated as it facilitates the implementation of Soft-$*$ algorithms within neural networks.

In this section, we introduce these neural networks that we call *OMP-Net* and *IHT-Net*, placing them under the umbrella of *greedy networks*, and describe their training procedure. In Section 2.4, we will provide numerical evidence that these networks can not only compensate for the introduced approximation error through training, but even outperform the original algorithms.

OMP-Net and IHT-Net Our goal is to approximate the reconstruction map associated with problem (78), i.e., $\Delta : \mathbb{C}^m \times \mathbb{C}^{m\times N} \ni (y, A) \mapsto x = \Delta(y, A) \in \mathbb{C}^N$ (or simply $\Delta(y)$ as A is fixed throughout), using a neural network built upon Soft-OMP and Soft-IHT. Although a variety of choices exist for trainable parameters (see Section 4.9), we specifically take the weights as trainable parameters, embedding them into the greedy criterion of Soft-OMP and the hard-thresholding operator of Soft-IHT, as explained in Section 4.2. This aims to uncover the latent structure within the signal and naturally sets forth an application of our construction in the context of weighted sparse recovery. Thus, constructing L layers of Soft-$*$ algorithms leads to trainable parameters $\Theta = (w^{(l)})_{l=1}^{L}$. Given the dataset $\mathcal{D} = \mathcal{S} \sqcup \mathcal{S}'$, where the training data $\mathcal{S} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N_{\text{tr}}}$ consists of N_{tr} instances of input-output pairs for training, and correspondingly the validation data $\mathcal{S}' = \{(x^{(i)}, y^{(i)})\}_{i=1}^{N_{\text{val}}}$, we train the neural network $y \in \mathbb{C}^m \mapsto \mathcal{NN}_{\mathcal{S},\Theta}(y) \in \mathbb{C}^N$. The network $\mathcal{NN}_{\mathcal{S},\Theta}(\cdot)$ uses the training data $\mathcal{S}$ to learn a data-driven approximation of the mapping Δ via gradient-based optimization. The parameters are iteratively updated over N_{epoch} epochs, generating the sequence $\Theta^{(j)}$, $j = 0, \ldots, N_{\text{epoch}}$, where $\Theta^{(0)}$ represents the network's initial state. The optimization minimizes the *squared-loss*

$$\text{Loss}_{sq}(x, x') = \|x - x'\|^2 \in [0, +\infty),$$

yielding the training error

$$\mathrm{MSE}_{\mathrm{tr}}^{(j)}(\mathcal{S}) = \frac{1}{N_{\mathrm{tr}}} \sum_{i=1}^{N_{\mathrm{tr}}} \mathrm{Loss}_{sq}(\mathcal{NN}_{\mathcal{S},\Theta^{(j)}}(y^{(i)}), x^{(i)}), \ j = 0, \ldots, N_{\mathrm{epoch}},$$

computed over the training data $\mathcal{S}$, with a similar validation error $\mathrm{MSE}_{\mathrm{val}}^{(j)}(\mathcal{S}')$ for the validation data $\mathcal{S}'$.

4.5 Numerical experiments

We now present numerical experiments to validate our construction. The source code of our numerical experiments can be found on the GitHub repository https://github.com/sina-taheri/Deep_Greedy_Unfolding. We first demonstrate that Soft-OMP and Soft-IHT serve as viable alternatives to OMP and IHT, capable of approximating the original algorithms with the desired accuracy depending on τ, the temperature parameter of softsort. We then illustrate their effectiveness in the scenario of severe undersampling, i.e., when m is so small that conventional compressed sensing struggles to reach a good recovery accuracy. By training neural networks based on these algorithms, we surpass the performance of conventional compressed sensing. Since all our experiments rely on a random compressed sensing setup with Gaussian or Fourier measurements, we first introduce this setup.

Random compressed sensing We generate an s-sparse signal $x \in \mathbb{R}^N$ by selecting $s \ll N$ random indices $i \in S \subset [N]$ without replacement from a discrete uniform distribution and assigning to each nonzero entry x_i a value from a standard random Gaussian distribution, i.e., $x_i \sim \mathcal{N}(0, 1)$. The signal x is then passed through the measurement matrix $A = A'/\sqrt{m} \in \mathbb{F}^{m \times N}$, where A' is either (1) a real-valued Gaussian matrix ($\mathbb{F} = \mathbb{R}$) with elements identically and independently distributed according to the $\mathcal{N}(0, 1)$ distribution, or (2) a partial Fourier matrix ($\mathbb{F} = \mathbb{C}$) given by $A' = PF$, where $P \in \mathbb{R}^{m \times N}$ is a row-selector matrix and $F \in \mathbb{C}^{m \times N}$ is the Fourier matrix with elements $F_{kl} = \exp(-i2\pi kl/N)$ for $k = -N/2 + 1, \ldots, N/2$ and $l = 0, \ldots, N - 1$. Our goal is to recover x from the measurements (78), where e is a random Gaussian noise vector with $e_i \sim \mathcal{N}(0, \sigma^2/m)$, real or complex-valued in the case of Gaussian or Fourier matrix respectively. We achieve this by running l iterations of OMP and IHT, or their differentiable counterparts, tackling the sparse recovery problem (79). This yields $x^{(l)} = \mathcal{A}(y, l)$, where $\mathcal{A}$ corresponds to OMP or IHT (pOMP or pIHT), or $\tilde{x}^{(l)}(\tau) = \tilde{\mathcal{A}}(y, l, \tau)$ for Soft-OMP or Soft-IHT.

4.5.1 Experiment I: Validation of Theorems 4.7 and 4.10

We numerically verify that Soft-OMP and Soft-IHT serve as accurate approximations of OMP and IHT, experimentally validating Theorems 4.7 and 4.10. Our compressed sensing setup employs Gaussian measurements with the following parameters:

$$N = 400, \quad m = 200, \quad s = 15, \quad \sigma = 10^{-3}, \quad \eta = 0.6 \text{ (for IHT)}.$$

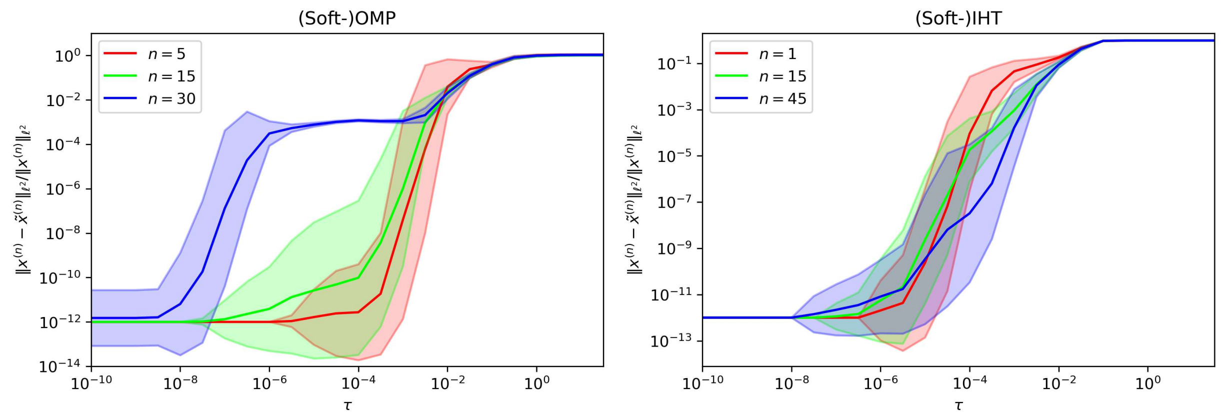

Figure 4.2: Relative ℓ^2-error as a function of τ (see Section 4.5.1). Recovery accuracy of (Soft-)OMP on the left and (Soft-)IHT on the right for various iteration counts.

We reconstruct the signals $x^{(n)}$ and $\tilde{x}^{(n)}(\tau)$ while varying τ and compute the relative ℓ^2-error

$$E^{(n)}(\tau) = \frac{\|x^{(n)} - \tilde{x}^{(n)}(\tau)\|}{\|x^{(n)}\|}.$$

This error is plotted as a function of τ for different iteration counts: $n \in \{5, 15, 30\}$ for (Soft-)OMP and $n \in \{1, 15, 45\}$ for (Soft-)IHT, respectively. The procedure is repeated N_{trial} times, and the results are visualized as shaded plots in Figure 4.2. The solid curves $(\tau, 10^{\mu_\tau^{(n)}})$ are bounded by $(\tau, 10^{\mu_\tau^{(n)} + \rho_\tau^{(n)}})$ and $(\tau, 10^{\mu_\tau^{(n)} - \rho_\tau^{(n)}})$ where the logarithmic mean $\mu_\tau^{(n)}$ and standard deviation $\rho_\tau^{(n)}$ are given by

$$\mu_\tau^{(n)} = \frac{1}{N_{\text{trial}}} \sum_{i=1}^{N_{\text{trial}}} \left(\log(E^{(n)}(\tau))\right)_i \text{ and } \rho_\tau^{(n)} = \sqrt{\frac{1}{N_{\text{trial}} - 1} \sum_{i=1}^{N_{\text{trial}}} \left[\left(\log(E^{(n)}(\tau))\right)_i - \mu_\tau^{(n)}\right]^2}.$$

To prevent numerical issues with logarithms, we compute $E^{(n)}(\tau) + \epsilon$ in practice, where $\epsilon = 10^{-12}$. Figure 4.2 confirms that for sufficiently small τ, Soft-OMP and Soft-IHT converge to their original counterparts. As expected, achieving this convergence for larger iteration counts requires smaller values of τ, consistent with Theorems 4.7 and 4.10. However, the precise rate of convergence varies across experiments, as the optimal τ depends on the problem-specific global min-gap, which tend to diminish with further iterations.

4.5.2 Experiment II: Weighted recovery in the undersampled regime

We now turn to the core motivation behind Soft-OMP and Soft-IHT: their unfolding into neural networks, corresponding to the trainable architectures OMP-Net and IHT-Net, respectively. In experiments of this section for IHT, the step size η is fixed (see, e.g., [11]). We construct a dataset using a sparse random model with a partial Fourier matrix $A \in \mathbb{C}^{m \times N}$, deliberately setting m low enough that standard OMP and IHT fail to reconstruct the signal efficiently. To exploit the learning capabilities of these networks, we generate signals whose support S lies within a larger set $T \supset S$

of size $|T| = ks,\ k \in \mathbb{N}$, uniformly distributed over $[N]$. This motivates the definition of oracle weights $w^{\text{oracle}} \in \mathbb{R}^N$

$$w_j^{\text{oracle}} = \begin{cases} 1 & j \in T, \\ 0 & \text{otherwise.} \end{cases}$$

OMP-Net and IHT-Net do not have access to the oracle information. Instead, they are initialized with uniform weights $w^{\text{NN}} = \mathbb{1}$, and are expected to infer the latent structure during training.

We generate the dataset $\mathcal{D}$ with parameters

$$N = 256, \quad s = 10, \quad m = 22 \text{ for OMP and } 36 \text{ for IHT}, \quad \eta = 0.5 \text{ (for IHT)}$$
$$\sigma = 10^{-3}, \quad k = 2, \quad N_{\text{train}} = 1024, \quad N_{\text{val}} = 512.$$

We construct OMP-Net and IHT-Net with L layers and employ weight-sharing across layers, i.e., $w^{(l)} = w^{\text{NN}} \in \mathbb{R}^N$ for all $l = 1, \dots, L$. We train them using RMSprop for N_{epoch} epochs with an empirically chosen learning rate lr and gradient norm clipping with maximum gradient norm set to 1. We further employ checkpointing to improve training: every N_{checkpt} epochs, the network parameters are saved, and the final model corresponds to the checkpoint with the best performance on the training data $\mathcal{S}$. The network parameters are:

$$L = 10 \text{ for OMP and } 30 \text{ for IHT}, \quad N_{\text{epoch}} = 1000, \quad N_{\text{checkpt}} = 10, \quad lr = 10^{-2}, \quad \tau = 10^{-3}.$$

As mentioned earlier, careful selection of τ is essential. While Theorems 4.7 and 4.10 suggest small values, excessively small τ, can lead to vanishing gradients, while overly large τ sacrifices accuracy to an extent that would make it very difficult to compensate for by training. Empirically, a good balance is achieved by choosing τ within the transition phase of Figure 4.2.

The results, presented in Figure 4.3, compare OMP and IHT (first and second columns, respectively). The top row shows the evolution of MSE_{tr} and MSE_{val}, truncated at the best checkpoint. For IHT number of iterations is always equal to L and $\eta = 0.5$. The stem plots in the second and third rows compare oracle and learned weights, illustrating the networks' ability to detect the set T from data. Since mean-squared error is sensitive to outliers, we complement our analysis with boxplots of the relative ℓ^2-error (fourth row), capturing the probabilistic nature of the experiment and confirming near noise-level signal recovery. Notably, OMP-Net and IHT-Net are able to outperform their classical counterpart by orders of magnitude.

4.6 Proof of Proposition 4.4

Proof. Continuity of softsort comes from the fact that softsort is a composition of continuous functions. Let $u^* = \text{sort}(u)$ and $v^* = \text{sort}(v)$, for any $u, v \in \mathbb{R}^N$. Then knowing that $\mathrm{softmax}$ is a

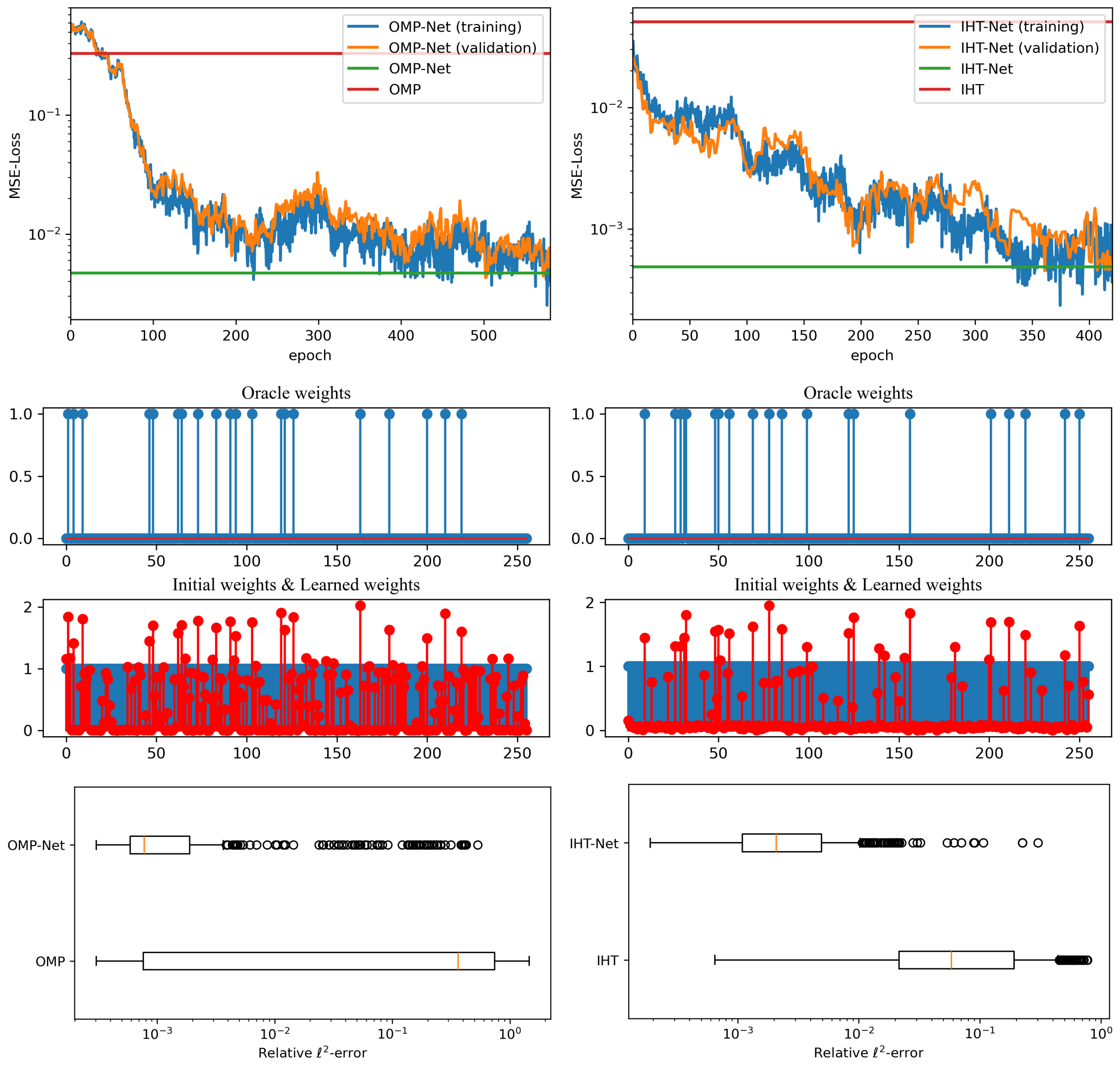

Figure 4.3: From top to bottom: MSE-Loss, oracle weights, learned weights and relative ℓ^2-error boxplots for (Soft-)OMP on the left and (Soft-)IHT on the right column (see Section 4.5.2 for further details).

1-Lipschitz continuous function [57], for all $i \in [N]$ we can write

$$\begin{aligned}
&\|\mathrm{softsort}_\tau(u)(i,:) - \mathrm{softsort}_\tau(v)(i,:)\| \\
&= \left\| \mathrm{softmax}\left(\frac{-|u_i^*\mathbb{1} - u|}{\tau}\right) - \mathrm{softmax}\left(\frac{-|v_i^*\mathbb{1} - v|}{\tau}\right) \right\| \le \frac{1}{\tau}\, \| |u_i^*\mathbb{1} - u| - |v_i^*\mathbb{1} - v| \| \\
&\le \frac{1}{\tau}\, \|(u_i^* - v_i^*)\mathbb{1} - (u - v)\| \le \frac{1}{\tau}\left(\|(u_i^* - v_i^*)\mathbb{1}\| + \|u - v\|\right) \\
&\le \frac{1}{\tau}\left(\sqrt{N}\|u^* - v^*\|_\infty + \|u - v\|\right) \overset{(*)}{\le} \frac{1}{\tau}\left(\sqrt{N}\|u - v\|_\infty + \|u - v\|\right) \\
&\le \frac{\sqrt{N} + 1}{\tau}\|u - v\|,
\end{aligned}$$

where $(*)$ holds due to the inequality $\|u^* - v^*\|_\infty \leq \|u - v\|_\infty$. For a proof of the latter when $u, v \in \mathbb{R}_+^N$, see [55, Lemma 2.12]. The generic case $u, v \in \mathbb{R}^N$ can be established by applying [55, Lemma 2.12] to $\tilde{u} = u - h$ and $\tilde{v} = v - h$ where $h := \min\{u_1, \ldots, u_N, v_1, \ldots, v_N\}$, observing that additive shifts do not impact the sorting operation. □

4.7 Proof of Theorem 4.7

At the heart of our proof lies the sensitivity analysis of least-squares, which quantifies how perturbations in the decomposition space affect the least-squares solution w. The following lemma provides a precise bound for this deviation in terms of $\kappa(B)$, the ℓ^2-norm condition number of B.

Lemma 4.11 (§Theorem 18.1 [111])**.** *Let $y \in \mathbb{C}^m$ and $A \in \mathbb{C}^{m \times n}$ of full rank be fixed. The least squares problem, i.e.,*

$$x = \operatorname*{argmin}_{z \in \mathbb{C}^n} \|y - Az\|_2^2,$$

has the following ℓ^2-norm relative condition number describing the sensitivity of x with respect to A:

$$\sup_{\delta A} \left(\frac{\|\delta x\|_2}{\|x\|_2} / \frac{\|\delta A\|}{\|A\|} \right) \leq \kappa(A) + \frac{\kappa(A)^2 \tan\theta}{\eta}.$$

Here $\delta x \in \mathbb{C}^n$ and $\delta A \in \mathbb{C}^{m \times n}$ represent perturbations in x and A respectively, $1 \leq \kappa(A) = \|A\|\|A^\dagger\|$ is the ℓ^2-norm condition number of A, and

$$0 \leq \theta = \cos^{-1} \frac{\|Ax\|_2}{\|y\|_2} \leq \pi/2, \quad 1 \leq \eta = \frac{\|A\|\|x\|_2}{\|Ax\|_2} \leq \kappa(A).$$

Assuming $\operatorname{argmax}(\Pi^{(n)}) = \operatorname{argmax}(\tilde{\Pi}^{(n)})$, the next lemma bounds $\|\Pi^{(n)} - \tilde{\Pi}^{(n)}\|$ and $\|\tilde{\Pi}^{(n)}\|$.

Lemma 4.12. *Let $\Pi^{(n)}$ and $\tilde{\Pi}^{(n)}$ be sequences of projection matrices generated by Algorithms 4.3 and 4.4 respectively and assume* $\operatorname{argmax} v^{(i)} = \operatorname{argmax} \tilde{v}^{(i)}$, $i \in [n]$. *Then we have*

$$\|\tilde{\Pi}^{(n)}\|_F \leq \sqrt{n}, \quad \|\Pi^{(n)} - \tilde{\Pi}^{(n)}\|_F \leq \sqrt{2n}(N-1)e^{-\tilde{g}^{(1:n)}/\tau},$$

where $\tilde{g}^{(1:n)}$ represents Soft-OMP's "global min-gap", defined as

$$\begin{aligned} \tilde{g}^{(1:n)} &:= \min_{i \in [n]} \tilde{g}^{(i)}, \\ \tilde{g}^{(i)} &:= \min_{j \in [N], j \neq \tilde{j}^{(i)}} \left| \tilde{v}_j^{(i)} - \tilde{v}_{\tilde{j}^{(i)}}^{(i)} \right| \text{ with } \tilde{j}^{(i)} := \operatorname{argmax}(\tilde{v}^{(i)}). \end{aligned} \tag{86}$$

Proof. For $\|\tilde{\Pi}^{(n)}\|_F$, adopting the notation

$$\tilde{j}^{(i)} := \operatorname*{argmax}_{j \in [N]} \tilde{v}_j^{(i)}, \quad \alpha^{(i)} := \sum_{j=1}^N \exp\left(- \left| \tilde{v}_{\tilde{j}^{(i)}}^{(i)} - \tilde{v}_j^{(i)} \right| / \tau \right), \tag{87}$$

we write

$$\|\tilde{\Pi}^{(n)}\|_F = \sqrt{\sum_{i=1}^{n} \|\mathrm{softsort}_\tau\,(\tilde{v}^{(i)})(1,:)\|^2} \leq \sqrt{\sum_{i=1}^{n} \|\mathrm{softsort}_\tau\,(\tilde{v}^{(i)})(1,:)\|_1^2} = \sqrt{n},$$

where we used Proposition 4.3(b).

For the second inequality, knowing that $\mathrm{argmax}(\Pi^{(n)}) = \mathrm{argmax}(\tilde{\Pi}^{(n)})$, Proposition 4.3(e) of softsort implies for all $i \in [n]$,

$$\begin{aligned}\tilde{P}^{(i)} &= \mathrm{softsort}_\tau\,(\tilde{v}^{(i)}) = \mathrm{softsort}_\tau\,(\mathrm{sort}\,(\tilde{v}^{(i)}))P_{\mathrm{argsort}\,(\tilde{v}^{(i)})} \\ &= \mathrm{softsort}_\tau\,(\mathrm{sort}\,(\tilde{v}^{(i)}))P_{\mathrm{argsort}\,(v^{(i)})} = \mathrm{softsort}_\tau\,(\mathrm{sort}\,(\tilde{v}^{(i)}))P^{(i)}.\end{aligned}$$

ℓ^2-norm is permutation invariant and thanks to Proposition 4.3(d) of softsort, the location of maximum entries in each row of the matrix $\mathrm{softsort}_\tau\,(\mathrm{sort}\,(\tilde{v}^{(i)}))$ occurs on the main diagonal of the matrix, thus we write

$$\begin{aligned}\left\|\Pi^{(n)} - \tilde{\Pi}^{(n)}\right\|_F^2 &= \sum_{i=1}^{n} \left\|P^{(i)}(1,:) - \tilde{P}^{(i)}(1,:)\right\|_2^2 \\ &= \sum_{i=1}^{n} \left\|P^{(i)}(1,:) - \left(\mathrm{softsort}_\tau\left(\mathrm{sort}\,(\tilde{v}^{(i)})\right)P^{(i)}\right)(1,:)\right\|_2^2 \\ &= \sum_{i=1}^{n} \left\|\left(I(i,:) - \mathrm{softsort}_\tau\left(\mathrm{sort}\,(\tilde{v}^{(i)})\right)(1,:)\right)P^{(i)}\right\|_2^2 \\ &= \sum_{i=1}^{n} \left\|I(i,:) - \mathrm{softsort}_\tau\left(\mathrm{sort}\,(\tilde{v}^{(i)})\right)(1,:)\right\|_2^2 \\ &= \sum_{i=1}^{n} \left(\left(1 - \frac{1}{\alpha^{(i)}}\right)^2 + \sum_{j=1,j\neq i}^{N} \left(\exp\left(-\left|\tilde{v}_i^{(i)} - \tilde{v}_j^{(i)}\right|/\tau\right)/\alpha^{(i)}\right)^2\right) \\ &\leq \sum_{i=1}^{n} \left(\left(1 - \frac{1}{\alpha^{(i)}}\right)^2 + \left(\sum_{j=1,j\neq i}^{N} \exp\left(-\left|\tilde{v}_i^{(i)} - \tilde{v}_j^{(i)}\right|/\tau\right)/\alpha^{(i)}\right)^2\right) \\ &= 2\sum_{i=1}^{n} \left(\frac{\alpha^{(i)} - 1}{\alpha^{(i)}}\right)^2 \leq 2\sum_{i=1}^{n} \left(\alpha^{(i)} - 1\right)^2 \\ &= 2\sum_{i=1}^{n} \left(\sum_{j=1,j\neq i}^{N} \exp\left(-\left|\tilde{v}_i^{(i)} - \tilde{v}_j^{(i)}\right|/\tau\right)\right)^2 \leq 2n(N-1)^2 e^{-2\tilde{g}^{(1:n)}/\tau},\end{aligned}$$

which concludes the proof. Here, $\tilde{g}^{(1:n)}$ is as defined in Equation (86) and $\alpha^{(i)} \geq 1$ is as in Equation (87), where $\tilde{j}^{(i)} = \mathrm{argmax}_{j\in[N]}\,\tilde{v}_j^{(i)} = i$ for all $i \in [n]$. □

Lemma 4.13. *If* $\mathrm{argmax}(\Pi^{(n)}) = \mathrm{argmax}(\tilde{\Pi}^{(n)})$ *with argmax being applied row-wise, then*

$$\left\|x^{(n)} - \tilde{x}^{(n)}\right\| \leq C^{(n)} e^{-\tilde{g}^{(1:n)}/\tau},$$

where

$$C^{(n)} = \sqrt{2n}(N-1)\left(\frac{\sqrt{1-\delta_n(A)} + (\sqrt{n}+1)\|A\|}{1-\delta_n(A)}\right)\|y\|_2,$$

and $\tilde{g}^{(1:n)}$ *as defined in Equation* (86).

Proof. We start by noting that

$$\begin{aligned}\left\|x^{(n)} - \tilde{x}^{(n)}\right\| &= \left\|\left(\Pi^{(n)}\right)^\top w^{(n)} - \left(\tilde{\Pi}^{(n)}\right)^\top \tilde{w}^{(n)}\right\| \\ &= \left\|\left(\Pi^{(n)}\right)^\top w^{(n)} - \left(\tilde{\Pi}^{(n)}\right)^\top \left(\tilde{w}^{(n)} - w^{(n)} + w^{(n)}\right)\right\| \\ &\leq \left\|\Pi^{(n)} - \tilde{\Pi}^{(n)}\right\| \left\|w^{(n)}\right\| + \left\|\tilde{\Pi}^{(n)}\right\| \left\|w^{(n)} - \tilde{w}^{(n)}\right\|. \end{aligned} \tag{88}$$

The sensitivity analysis result in Lemma 4.11 implies, for the least-squares step in OMP, that

$$\sup_{\delta B^{(n)}} \left(\frac{\|\delta w^{(n)}\|}{\|w^{(n)}\|}\frac{\|B^{(n)}\|}{\|\delta B^{(n)}\|}\right) \leq \kappa\left(B^{(n)}\right) + \frac{\kappa\left(B^{(n)}\right)^2 \tan\theta^{(n)}}{\eta^{(n)}}.$$

Taking the deviation with respect to the equivalent variables in Soft-OMP, rearranging the inequality and substituting $\theta^{(n)}$ and $\eta^{(n)}$ as in Lemma 4.11, we have

$$\begin{aligned}\left\|w^{(n)} - \tilde{w}^{(n)}\right\| &\leq \left(\kappa\left(B^{(n)}\right) + \frac{\kappa\left(B^{(n)}\right)^2 \tan\theta^{(n)}}{\eta^{(n)}}\right)\frac{\left\|\delta B^{(n)}\right\|}{\left\|B^{(n)}\right\|}\left\|w^{(n)}\right\| \\ &= \left(\kappa\left(B^{(n)}\right) + \kappa\left(B^{(n)}\right)^2 \frac{\left\|r^{(n)}\right\|}{\left\|B^{(n)}\right\|\left\|w^{(n)}\right\|}\right)\frac{\left\|\delta B^{(n)}\right\|}{\left\|B^{(n)}\right\|}\left\|w^{(n)}\right\|, \end{aligned} \tag{89}$$

where δB is the perturbation introduced to the matrix $B^{(n)} = A(\Pi^{(n)})^\top$ due to approximation of $\Pi^{(n)}$ by $\tilde{\Pi}^{(n)}$. We further note that

$$\tan(\theta^{(n)}) = \frac{\sin\left(\cos^{-1}\left(\|Ax^{(n)}\|/\|y\|\right)\right)}{\cos\left(\cos^{-1}\left(\|Ax^{(n)}\|/\|y\|\right)\right)} = \frac{\sqrt{\|y\|^2 - \|Ax^{(n)}\|^2}/\|y\|}{\|Ax^{(n)}\|/\|y\|} = \frac{\|r^{(n)}\|}{\|B^{(n)}w^{(n)}\|},$$

where we defined $r^{(n)} := y - Ax^{(n)} = y - B^{(n)}w^{(n)}$ and used orthogonality of $r^{(n)}$ to $Ax^{(n)}$ thanks to least squares. Substituting Equation (89) in Equation (88) and rearranging, we get

$$\begin{aligned}\left\|x^{(n)}-\tilde{x}^{(n)}\right\| &\leq \left\|\Pi^{(n)}-\tilde{\Pi}^{(n)}\right\|\left\|w^{(n)}\right\|+\left\|\tilde{\Pi}^{(n)}\right\|\left\|w^{(n)}-\tilde{w}_n\right\| \\ &\leq \left(\left\|\Pi^{(n)}-\tilde{\Pi}^{(n)}\right\|+\kappa\left(B^{(n)}\right)\left\|\tilde{\Pi}^{(n)}\right\|\frac{\left\|A\left(\Pi^{(n)}-\tilde{\Pi}^{(n)}\right)^{\top}\right\|}{\left\|B^{(n)}\right\|}\right)\left\|w^{(n)}\right\| \\ &\quad +\kappa\left(B^{(n)}\right)^2\frac{\left\|A\left(\Pi^{(n)}-\tilde{\Pi}^{(n)}\right)^{\top}\right\|}{\left\|B^{(n)}\right\|^2}\left\|r^{(n)}\right\|.\end{aligned}$$

Taking advantage of the relations

$$\begin{aligned}&\kappa\left(B^{(n)}\right)=\left\|B^{(n)}\right\|\left\|B^{(n)\dagger}\right\|, \quad w^{(n)}=B^{(n)\dagger}y \\ &\left\|\Pi^{(n)}-\tilde{\Pi}^{(n)}\right\|\leq\left\|\Pi^{(n)}-\tilde{\Pi}^{(n)}\right\|_F, \qquad \left\|\tilde{\Pi}^{(n)}\right\|\leq\left\|\tilde{\Pi}^{(n)}\right\|_F,\end{aligned}$$

and knowing that $\|r^{(n)}\|\leq\|y\|$ (see, [42, 55]) we have

$$\begin{aligned}\left\|x^{(n)}-\tilde{x}^{(n)}\right\| \leq &\left(\left\|\Pi^{(n)}-\tilde{\Pi}^{(n)}\right\|+\left\|B^{(n)\dagger}\right\|\|A\|\left\|\tilde{\Pi}^{(n)}\right\|\left\|\Pi^{(n)}-\tilde{\Pi}^{(n)}\right\|\right)\left\|B^{(n)\dagger}\right\|\|y\| \\ &+\left\|B^{(n)\dagger}\right\|^2\|A\|\left\|\Pi^{(n)}-\tilde{\Pi}^{(n)}\right\|\|y\|.\end{aligned}$$

We further note that

$$\left\|B^{(n)\dagger}\right\|=\left\|\left(A\Pi^{(n)\top}\right)^{\dagger}\right\|=\left\|A^{\dagger}_{S^{(n)}}\right\|=\frac{1}{\sigma_{\min}(A_{S^{(n)}})}\leq\frac{1}{\sqrt{1-\delta_n(A)}},$$

where the last inequality is well-known in the compressive sensing literature (see, e.g., [55, Ch. 6]). Therefore, applying the upper bounds of Lemma 4.12 and with some rearrangement we conclude that

$$\left\|x^{(n)}-\tilde{x}^{(n)}\right\|\leq\sqrt{2n}(N-1)\left(\frac{\sqrt{1-\delta_n(A)}+(\sqrt{n}+1)\|A\|}{1-\delta_n(A)}\right)\|y\|e^{-\tilde{g}^{(1:n)}/\tau}.$$

□

Proof of Theorem 4.7.

(i) Part (i) of the theorem directly follows from the Proposition 4.3(c). If $\tau\to 0$, Soft-OMP coincides with pOMP and thus with OMP.

(ii) We prove Part (ii) via induction.

$(\boldsymbol{k=0})$ This case trivially holds as OMP and Soft-OMP are initialized as $x^{(0)} = \tilde{x}^{(0)} = 0$.

$(\boldsymbol{k-1 \to k})$ Assume that $\operatorname{argmax}(\Pi^{(k-1)}) = \operatorname{argmax}(\tilde{\Pi}^{(k-1)})$ and $\max_{i\in[0:k-1]}$ $\|x^{(i)} - \tilde{x}^{(i)}\| \le \epsilon$. We will show that $\operatorname{argmax}(\Pi^{(k)}) = \operatorname{argmax}(\tilde{\Pi}^{(k)})$ and $\max_{i\in[0:k]}$ $\|x^{(i)} - \tilde{x}^{(i)}\| \le \epsilon$.

We start by noting that for all $j \in [N]$,

$$\begin{aligned}\left|v_j^{(k)} - \tilde{v}_j^{(k)}\right| &= \left|\left(A^*A\left(x^{(k-1)}\right)\right)_j - \left(A^*A\left(\tilde{x}^{(k-1)}\right)\right)_j\right| \\ &= \left|\left\langle a_j, A\left(x^{(k-1)} - \tilde{x}^{(k-1)}\right)\right\rangle\right| \le \left\|A\left(x^{(k-1)} - \tilde{x}^{(k-1)}\right)\right\| \\ &\le \|A\| \left\|x^{(k-1)} - \tilde{x}^{(k-1)}\right\| \le \|A\|\epsilon.\end{aligned}$$

For $j^{(k)} = \operatorname{argmax}_{j\in[N]}(v_j^{(k)})$, it can be easily verified that the above inequality, combined with the assumption $g^{(k)} > 2\|A\|\epsilon$, implies

$$\left|v_j^{(k)} - \tilde{v}_j^{(k)}\right| < g^{(k)}/2,$$

for all $j \in [N]$, implying $\operatorname{argmax}(v^{(k)}) = \operatorname{argmax}(\tilde{v}^{(k)})$ and as a result $\operatorname{argmax}(\Pi^{(k)}) = \operatorname{argmax}(\tilde{\Pi}^{(k)})$. On the other hand, we notice that

$$\begin{aligned}\left|\tilde{v}_i^{(k)} - \tilde{v}_j^{(k)}\right| &= \left|\tilde{v}_i^{(k)} - v_i^{(k)} + v_i^{(k)} - \tilde{v}_j^{(k)} + v_j^{(k)} - v_j^{(k)}\right| \\ &\ge \left|v_i^{(k)} - v_j^{(k)}\right| - \left|v_i^{(k)} - \tilde{v}_i^{(k)}\right| - \left|v_j^{(k)} - \tilde{v}_j^{(k)}\right| \ge \left|v_i^{(k)} - v_j^{(k)}\right| - 2\|A\|\epsilon, \qquad (90)\end{aligned}$$

which implies $\tilde{g}^{(k)} \ge g^{(k)} - 2\|A\|\epsilon > 0$ (recall Equation (86)). Now we are in a position to apply Lemma 4.13, which yields

$$\left\|x^{(k)} - \tilde{x}^{(k)}\right\| \le C^{(k)} e^{-\tilde{g}^{(1:k)}/\tau} \le C^{(k)} e^{\left(-g^{(1:k)} + 2\|A\|\epsilon\right)/\tau},$$

with $\tilde{g}^{(1:k)}$ and $g^{(1:k)}$ as defined in Equation (86) and Equation (84), respectively. Prescribing the upper bound above to be less than or equal to ϵ and solving the corresponding inequality for τ concludes the proof. □

4.8 Proof of Theorem 4.10

The key component of the proof of Theorem 4.10 is a bound on the distance between IHT and Soft-IHT over two consecutive iterations, as established in the following lemma. We later use this relation in an inductive argument to derive the main result.

Lemma 4.14. *For any s-sparse $x \in \mathbb{C}^N$, let $u = u(x) := (I - A^*A)x + A^*y$, where $A \in \mathbb{C}^{m\times N}$ with ℓ^2-normalized columns and $y \in \mathbb{C}^m$, and define $v = |u|$. Also, let x^+ be the signal after one* *iteration of IHT, i.e., $x^+ = H_s(u) = q \odot u$ where*

$$q = \sum_{i=1}^{s} Q(i,:), \quad Q = P_{\mathrm{argsort}(v)}.$$

Correspondingly, for any $\tilde{x} \in \mathbb{C}^N$ with $\|x - \tilde{x}\| \le g/2L$, where

$$g := \min_{i,j \in [N], i \neq j} |v_i - v_j|,$$

and $L := \|I - A^*A\|$ is the Lipschitz constant of the operator $u(\cdot)$, define $\tilde{u} = u(\tilde{x})$ and $\tilde{v} = |\tilde{u}|$, and let $\tilde{x}^+$ be the signal after one iteration of Soft-IHT, i.e., $\tilde{x}^+ = \tilde{H}_s(\tilde{u}) = \tilde{q} \odot \tilde{u}$ with

$$\tilde{q} = \sum_{i=1}^{s} \tilde{Q}(i,:), \quad \tilde{Q} = \mathrm{softsort}_\tau(\tilde{v}).$$

Then, we have

$$\|x^+ - \tilde{x}^+\| \le 2sN(s\mu\|x\|_\infty + \|y\|)e^{-\tilde{g}/\tau} + sL\|x - \tilde{x}\|,$$

where μ is the coherence of the matrix A and $\tilde{g}$ is the minimum gap between elements of $\tilde{v}$, i.e.,

$$\tilde{g} := \min_{i,j \in [N], i \neq j} |\tilde{v}_i - \tilde{v}_j|.$$

Proof. The proof is broken into three parts:

Part I The submultiplicative inequality implies that u is a Lipschitz operator, i.e., $\|u - \tilde{u}\| \le L\|x - \tilde{x}\|$, where $L := \|I - A^*A\|$.

Part II First note that, for each $i \in [N]$,

$$|v_i - \tilde{v}_i| = ||u_i| - |\tilde{u}_i|| \le |u_i - \tilde{u}_i| \le \|u - \tilde{u}\|_\infty \le \|u - \tilde{u}\|. \tag{91}$$

Let

$$g := \min_{i,j \in [N], i \neq j} |v_i - v_j|.$$

Using Part I, we observe that under the condition $g > 2L\|x - \tilde{x}\|$, it holds

$$|v_i - \tilde{v}_i| < g/2, \tag{92}$$

implying that $\mathrm{argsort}(v) = \mathrm{argsort}(\tilde{v})$.

Part III We then arrive at bounding $\|x^+ - \tilde{x}^+\|$. We see that

$$\begin{aligned}\|x^+ - \tilde{x}^+\| &= \|H_s(u) - \tilde{H}_s(\tilde{u})\| = \|q \odot u - \tilde{q} \odot \tilde{u}\| \\ &= \|q \odot u - \tilde{q} \odot (\tilde{u} - u + u)\| \le \|(q - \tilde{q}) \odot u\| + \|\tilde{q} \odot (u - \tilde{u})\| \\ &\le \|(q - \tilde{q}) \odot u\|_1 + \|\tilde{q} \odot (u - \tilde{u})\|_1 \le \underbrace{\|q - \tilde{q}\|_1}_{(a)} \underbrace{\|u\|_\infty}_{(b)} + \underbrace{\|\tilde{q}\|_1}_{(c)} \underbrace{\|u - \tilde{u}\|_\infty}_{(d)},\end{aligned}$$

where in the last inequality we used $\|a \odot b\|_1 = |\langle a, b\rangle| \le \|a\|_1 \|b\|_\infty$ by Hölder's inequality. Now we bound each term separately:

(a) Knowing that $\operatorname{argsort}(v) = \operatorname{argsort}(\tilde{v})$ as a consequence of Part II, Proposition 4.3(e) of softsort implies

$$\begin{aligned}\operatorname{softsort}_\tau(\tilde{v}) &= \operatorname{softsort}_\tau(\operatorname{sort}(\tilde{v}))P_{\operatorname{argsort}(\tilde{v})} \\ &= \operatorname{softsort}_\tau(\operatorname{sort}(\tilde{v}))P_{\operatorname{argsort}(v)} = \operatorname{softsort}_\tau(\operatorname{sort}(\tilde{v}))Q.\end{aligned}$$

As the ℓ^1-norm is permutation invariant, a similar argument to Lemma 4.12 shows

$$\begin{aligned}\|q - \tilde{q}\|_1 &= \left\| \sum_{i=1}^s Q(i,:) - \sum_{i=1}^s \operatorname{softsort}_\tau(\operatorname{sort}(\tilde{v}))Q(i,:) \right\|_1 \\ &= \left\| \sum_{i=1}^s \left(I(i,:) - \operatorname{softsort}_\tau(\operatorname{sort}(\tilde{v}))(i,:)\right) Q \right\|_1 \\ &= \left\| \sum_{i=1}^s \left(I(i,:) - \operatorname{softsort}_\tau(\operatorname{sort}(\tilde{v}))(i,:)\right) \right\|_1 \\ &= \left\| \sum_{i=1}^s \left(e_i - \operatorname{softmax}\left(-\frac{|\operatorname{sort}(\tilde{v})_i \mathbb{1}^\top - \operatorname{sort}(\tilde{v})^\top|}{\tau} \right)\right) \right\|_1 \\ &\le \sum_{i=1}^s \left(1 - \frac{1}{\alpha_i}\right) + \sum_{j=1, j\neq i}^N \sum_{i=1}^s \frac{e^{-|\tilde{v}_i - \tilde{v}_j|/\tau}}{\alpha_i} \le \sum_{i=1}^s 2\left(1 - \frac{1}{\alpha_i}\right) \\ &= 2\sum_{i=1}^s \frac{\alpha_i - 1}{\alpha_i} \le 2\sum_{i=1}^s (\alpha_i - 1) \le 2\sum_{i=1}^s \sum_{j=1,j\neq i}^N e^{-|\tilde{v}_i - \tilde{v}_j|/\tau} \le 2sNe^{-\tilde{g}/\tau},\end{aligned}$$

where $\alpha_i := \sum_{j=1}^N \exp(-|\tilde{v}_i - \tilde{v}_j|/\tau)$.

(b) Noting that $\|u\|_\infty \le \|(I - A^*A)x\|_\infty + \|A^*y\|_\infty$, for the first term we write

$$\begin{aligned}\|(I - A^*A)x\|_\infty &= \max_{i\in[N]} |((I - A^*A)x)_i| = \max_{i\in[N]} \left|\left\langle e_i - a_i^\top A, x\right\rangle\right| \\ &= \max_{i\in[N]} \left|x_i - \left\langle a_i^\top A, x\right\rangle\right| = \max_{i\in[N]} \left| \left(1 - \|a_i\|^2\right) x_i - \sum_{j=1,j\neq i}^N \langle a_i, a_j\rangle x_j \right| \\ &\le \max_{i\in[N]} \sum_{j\in S, j\neq i} |\langle a_i, a_j\rangle x_j| \le \|x\|_\infty \max_{i\in[N]} \sum_{j\in S,j\neq i} |\langle a_i, a_j\rangle| \le s\mu\|x\|_\infty,\end{aligned}$$

where $S = \operatorname{supp}(x)$. Furthermore, $\|A^*y\|_\infty = \max_{i\in[N]} |\langle a_i, y\rangle| \le \|y\|$. Therefore,

$$\|u\|_\infty \le \|(I - A^*A)x\|_\infty + \|A^*y\|_\infty \le s\mu\|x\|_\infty + \|y\|.$$

(c) As we sum over s rows of the softsort matrix and by switching the sums over rows and columns, thanks to Proposition 4.3(b) of softsort we have $\|\tilde{q}\|_1 = \sum_{i=1}^{s} \|\tilde{Q}(i,:)\|_1 = s$.

(d) We derived earlier that $\|u - \tilde{u}\|_\infty \le \|u - \tilde{u}\| \le L\|x - \tilde{x}\|$.

Putting all bounds of (a)–(d) together concludes the proof. □

Now we are in a position to set out the proof of the main IHT theorem.

Proof of Theorem 4.10.

(i) Part (i) of the theorem directly follows from Proposition 4.3(c) of the softsort as in Proposition 4.3. If $\tau \to 0$, Soft-IHT coincides with pIHT and thus with IHT.

(ii) Using the result of Lemma 4.14 by induction we can write

$$\begin{aligned}
\|x^{(n)} - \tilde{x}^{(n)}\| &\le 2sN\left(\|y\| + s\mu\|x^{(n-1)}\|_\infty\right) e^{-\tilde{g}^{(n)}/\tau} + sL\|x^{(n-1)} - \tilde{x}^{(n-1)}\| \\
&\le 2sN\left(\|y\| + s\mu\|x^{(n-1)}\|_\infty\right) e^{-\tilde{g}^{(n)}/\tau} \\
&\quad + sL\Big(2sN(\|y\| + s\mu\|x^{(n-2)}\|_\infty)e^{-\tilde{g}^{(n-1)}/\tau} + sL\|x^{(n-2)} - \tilde{x}^{(n-2)}\|\Big) \\
&\le 2sN(1+sL)\left(\|y\| + s\mu \max\left\{\|x^{(n-1)}\|_\infty, \|x^{(n-2)}\|_\infty\right\}\right) e^{-\min\left\{\tilde{g}^{(n)},\tilde{g}^{(n-1)}\right\}/\tau} \\
&\quad + (sL)^2\|x^{(n-2)} - \tilde{x}^{(n-2)}\| \\
&\;\;\vdots \\
&\le 2sN\frac{(sL)^n - 1}{sL - 1}\left(\|y\| + s\mu \max_{1\le k\le n-1}\|x^{(k)}\|\right) e^{-\tilde{g}^{(1:n)}/\tau} + (sL)^n\|x^{(0)} - \tilde{x}^{(0)}\|.
\end{aligned} \tag{93}$$

Here $\tilde{g}^{(1:n)}$ is defined as

$$\tilde{g}^{(1:n)} := \min_{k\in[n]} \tilde{g}^{(k)}, \quad \tilde{g}^{(k)} = \min_{i,j\in[N], i\ne j} \left|v_i^{(k)} - v_j^{(k)}\right|,$$

and the result holds under the condition $\max_{1\le k\le n-1}\|x^{(k)} - \tilde{x}^{(k)}\| \le g^{(1:n)}/2L$ with $g^{(1:n)}$ as in Equation (85). Under this condition, a similar analysis as in the proof of Theorem 4.7 (see Section 4.7) leads to $\tilde{g}^{(1:n)} \ge g^{(1:n)} - 2L\max_{1\le k\le n-1}\|x^{(k)} - \tilde{x}^{(k)}\| > 0$, connecting the global min gap of Soft-IHT to the one of IHT. Hence, the upper bound Equation (93) can be written as

$$\|x^{(n)} - \tilde{x}^{(n)}\| \le C^{(n)} e^{\left(-g^{(1:n)} + 2L\max_{1\le k\le n-1}\|x^{(k)} - \tilde{x}^{(k)}\|\right)/\tau}, \tag{94}$$

where

$$C^{(n)} = 2sN\frac{(sL)^n - 1}{sL - 1}\left(\|y\| + s\mu \max_{1\leq k\leq n-1}\|x^{(k)}\|\right).$$

Now using the fact that $x^{(0)} = \tilde{x}^{(0)}$, bounding (94) from above by ϵ, which is chosen so as to satisfy $0 < \epsilon < g^{(1:n)}/2L$, and solving the corresponding inequality for τ concludes the proof. □

4.9 Conclusions and future research

In this chapter, we directly addressed the non-differentiability of the argsort operator in greedy sparse recovery algorithms, such as OMP and IHT, and proposed differentiable approximations, Soft-OMP and Soft-IHT, which were effectively unrolled into neural networks. We also established rigorous theoretical error bounds for Soft-OMP and Soft-IHT in relation to their original counterparts.

There are several potential avenues for future research. First, softsort is not the only approach for implementing the argsort operator in a differentiable manner within neural networks. The development of neural-network compatible implementations of (arg)sort has long been an area of active interest. However, differentiable realizations of such operators have only emerged in the last decade. Notable examples include Neuralsort [62], which shares a significant resemblance to softsort, optimal transport-based methods using the Sinkhorn algorithm [38, 84], optimization-based approaches [19, 101], and the Gumble trick [70]. Among these, we chose softsort primarily due to its simple formulation and interpretability. That said, a comprehensive comparison of greedy networks based on the various differentiable sorting methods available in the literature remains a path to be pursued.

Second, in this work we unrolled OMP and IHT as prominent representatives of greedy algorithms. Other well-known candidates worth mentioning include, but are not limited to, CoSaMP [91], subspace pursuit [39] and greedy weighted-LASSO algorithms [86]. In the next chapter, we will make some good strides towards unrolling CoSaMP, albeit yet without training.

Third, in our implementation we use the most natural architecture directly imposed by the model. However, generalizations to other architectures, such as Recurrent Neural Networks (RNNs) to capture the recurrent nature of algorithms or Convolutional Neural Networks (CNNs) to reduce the number of parameters, warrant further investigation. Additionally, several interesting extensions could be explored, including experiments with varying numbers of layers, learning additional parameters (e.g., the step size η, the full matrix A^*, $I - A^*A$, etc.), employing distinct parameters across layers, and realization of stopping criterion. In particular, Section 5.4 showcases training involving matrices and the step size for IHT-Net.

Chapter 5

Further results on deep greedy unfolding

In this chapter we present some complementary results and roads taken towards filling some of the gaps mentioned in Section 4.9. In particular, we commence with some complementary experiments for OMP- and IHT-Net. OMP-related contents of this section previously appeared in a conference paper [87]. Then in the two subsequent sections we unroll more algorithms. CoSaMP naturally serves as our first choice as it can be regarded as a combination of OMP and IHT, for which we only present convergence of Soft-CoSaMP to CoSaMP without delving into training. Then, shifting the paradigm a bit, we apply our framework to a fast and memory-efficient algorithm in deterministic compressed sensing for which we present some early results on training as well. This section is thanks to a collaboration with Prof. Mark Iwen at Michigan State University. Finally, in the last section of this chapter, we study the feasibility of our earlier proposal in Section 4.9 on learning other parameters of the unrolled networks beyond weights. Adopting a filter-based perspective, we demonstrate that IHT-Net with matrix weight coupling is trainable and significantly outperforms IHT in comparable settings.

5.1 Complementary numerical results for OMP- and IHT-Net

In this section we provide complementary results firstly on the convergence of Soft-OMP and Soft-IHT to OMP and IHT. Then we investigate the effect of domain knowledge when the structure within the signal is subject to change, showing effectiveness of OMP-Net for highly structured data.

5.1.1 Soft-OMP (IHT) vs. OMP (IHT)

We consider the same numerical setting of Section 4.5.1 with the difference that we plot each algorithm's relative ℓ^2-error, i.e., with respect to the true signal, for various noise levels $\sigma \in 10^{\{-5,-3,-1\}}$ and as a function of τ, the temperature parameter of softsort. Figure 5.1 shows the results as shaded plots. Solid curves represent mean relative ℓ^2-errors as a function of τ. These are surrounded from above and below by the boundaries $(\tau, 10^{\mu_\sigma^\tau+\rho_\sigma^\tau})$ and $(\tau, 10^{\mu_\sigma^\tau-\rho_\sigma^\tau})$ respectively, with

$$\mu_\sigma^\tau = \frac{1}{N_{\text{trial}}}\sum_{i=1}^{N_{\text{trial}}}\left(\log(E_\sigma^\tau)\right)_i, \quad \rho_\sigma^\tau = \sqrt{\frac{1}{N_{\text{trial}}-1}\sum_{i=1}^{N_{\text{trial}}}(\log(E_\sigma^\tau)_i - \mu_\sigma^\tau)^2}.$$

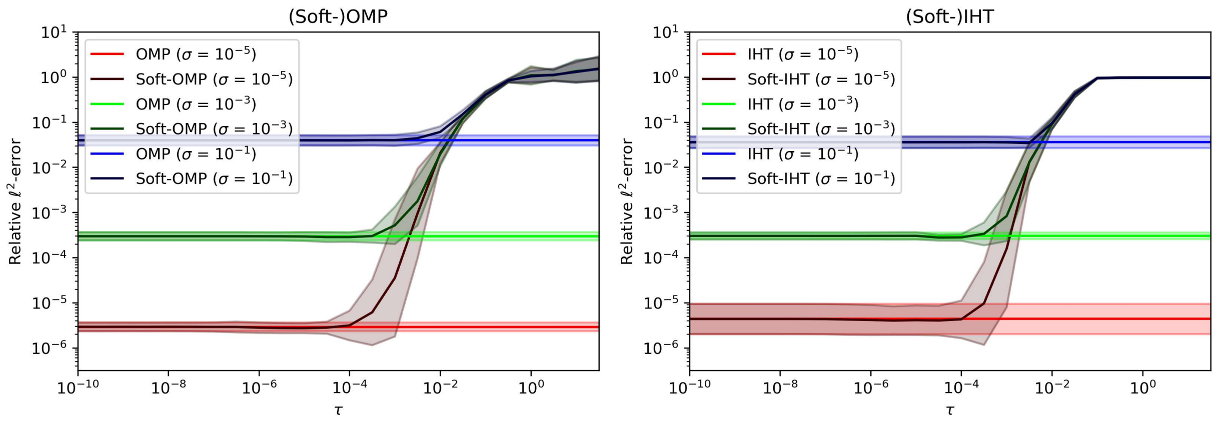

Figure 5.1: Relative ℓ^2-error as a function of τ (see Sections 4.5.1 and 5.1). Relative ℓ^2-error of (Soft-) OMP on the left and (Soft-)IHT on the right for various noise levels.

Here E_σ^τ denotes the relative ℓ^2-error for the noise level σ and at τ, with E defined as $E = \|x - \tilde{x}\|_2/\|x\|_2$, where $\tilde{x}$ is the signal recovered by OMP (IHT) or Soft-OMP (IHT). We observe, as predicted by Theorems 4.7 and 4.10, that if τ is sufficiently small, Soft algorithms approximates their original counterparts with minimal error.

5.1.2 Effect of domain knowledge

In this subsection we further investigate the effectiveness of unrolled networks in extracting the correct structure within the data while the structure level is subject to change. Here we only focus on OMP-Net and mention in passing that similar results can be obtained for IHT-Net. For the sake of comparison we also include WOMP of Algorithm 3.1 with weights multiplied instead of divided due to convenience and in accordance with Chapter 4. The compressed sensing setup that we run for this experiment is as follows:

$$N = 84,\ m = 32,\ s = 6,\ \sigma = 10^{-3},$$

with $A \in \mathbb{N}^{m\times N}$ Guassian as in Section 2.4. Let $[K] \subseteq [N]$, be the proportion of the signal's ambient dimension that consists of the signal's support. For example, if $K = s$ the dataset to train the network consists of sparse signals with nonzero entries from only the support's first s elements. However, we do not assume any oracle knowledge for the network, and OMP-Net is always initialized with $w = \mathbb{1}$ and the weights for WOMP are set to $w_j = 1$ if $j \leq K$ and 0 otherwise i.e., WOMP is provided with the oracle knowledge, but OMP-Net has to learn the structure through the training process. We note that at $K = N$, weights would be $w = \mathbb{1}$; thus, WOMP corresponds to OMP. For each K in the discrete range $[s, N]$, we generate a new dataset and repeat this procedure $N_{\text{trial}} = 10$ times. We devise checkpoints every 10 epochs during training OMP-Net and report the best performance on the test data in Figure 5.2. This experiment's result is summarized in shaded plots with respect to MSE in Figure 5.2. Also, oracle and learned weights of one training instance at $k = 6$ are shown on the right of Figure 5.2 as an example. It is observable that OMP-Net can learn the structure close to the oracle level represented by WOMP. As K approaches N, OMP-Net approaches the performance of OMP, which fails to reconstruct

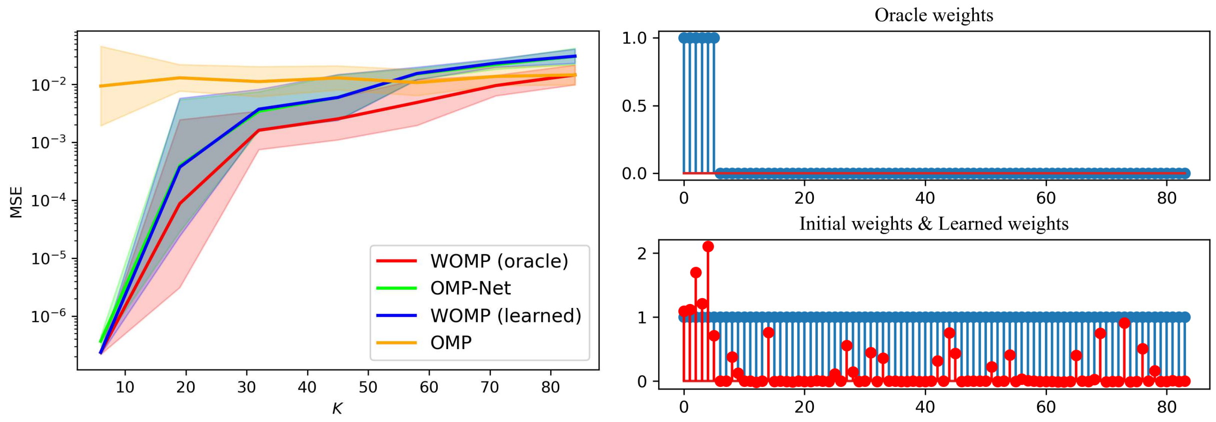

Figure 5.2: On the left, MSE of OMP-Net, OMP, WOMP with oracle and learned weights, with respect to K. On the right, oracle and learned weights for one instance of the experiment at $K = s$.

the signal well due to the severely undersampled regime.

5.2 Towards unrolling CoSaMP

As mentioned in Section 4.9, a natural extension of the framework presented in Chapter 4 can encompass CoSaMP, proposed in [91], as CoSaMP takes benefit of the greedy selection step of OMP combined with the hard-thresholding step of IHT. In this regard, this section deals with CoSaMP, its generalization to the weighted case, unrolling CoSaMP and finally showing that Soft-CoSaMP coincides with CoSaMP in light of Section 4.5.1. We leave training of unrolled CoSaMP to future research as our experiments by the time of composition of this dissertation revealed difficulty in training and hyperparameter tuning of CoSaMP in comparison to OMP and IHT, confronting exploding gradients [58].

5.2.1 CoSaMP

One major drawback of OMP is that when an index is added to the support of the signal based on a local greedy criterion, there is no mechanism to replace it by another throughout the iterations from a more global perspective. Furthermore, the fact that at each iteration OMP allows inclusion of only one index makes it less suitable in applications with higher sparsity regimes. IHT does not directly suffer from any of these issues, but hard-thresholding is applied to the whole ambient dimension which makes it overall more prone to noise and error, so devising a plan to preselect some of the indices would be reasonable. This is achieved using the index selector $\mathcal{I}_{2k}(\cdot) = \text{argsort}(\cdot)(1 : 2k)$ that extracts the indices related to top entries of the input that is added to the current signal support $S^{(n)}$ of cardinality k. So overall least-squares is solved over the set $U^{(n+1)}$ with $\text{card}(U^{(n+1)}) \leq 3s$. Finally the k-sparse hard-thresholding operator prunes the least-squares output to return a new signal $x^{(n+1)}$ which together with its support serve as the input for the next iteration. Algorithm 5.1 represents CoSaMP algorithm.

Thanks to the devised versatility of CoSaMP, it converges with much fewer iterations than OMP and IHT, although each iteration is quite more computationally expensive. Nonetheless, this

Algorithm 5.1 Compressive Sampling Matching Pursuit (CoSaMP).

Input: desired sparsity k, mixing matrix $A \in \mathbb{C}^{m \times N}$ with ℓ^2-normalized columns, observation vector $y \in \mathbb{C}^m$.

```
Let $x^{(0)} = 0$ and $S^{(0)} = \emptyset$.
For $n = 0, \dots, \overline{n}$ repeat:
    $U^{(n+1)} = S^{(n)} \cup \mathcal{I}_{2k}(|(A^*(y - Ax^n)|)$    (CoSaMP.1)
    $u^{(n+1)} \in \operatorname*{argmin}_{z \in \mathbb{C}^N} \{\|y - Az\|_2^2 \quad \text{s.t} \quad \operatorname{supp}(z) \subseteq U^{(n+1)}\}$    (CoSaMP.2)
    $x^{(n+1)} = H_k(u^{(n+1)}),$    (CoSaMP.3)
```

Output: k-sparse vector $\hat{x} = x^{(\overline{n})} \in \mathbb{C}^N$.

versatility makes it a good candidate to become sublinearized (see, [34] and Section 5.3). Now we move on to unrolling CoSaMP.

5.2.2 Soft-CoSaMP

With Soft-OMP and Soft-IHT in our basket it is rather straightforward to unroll CoSaMP as well, so we omit unnecessary details and the intermediate permutation (projection)-based CoSaMP (pCoSaMP) for brevity and directly get to Soft-CoSaMP proposed in Algorithm 5.2.

Comparing Algorithms 5.1 and 5.2, one might raise a potential concern in Soft-CoSaMP: in (Soft-CoSaMP.1) we stack the first k rows of the approximate permutation matrix $\tilde{Q}^{(n)}$ from the previous iteration related to soft hard-thresholding step over the top $2k$ rows of the approximate permutation matrix $\tilde{P}^{(n+1)}$ of the current iteration associated with the greedy selection criterion. Unlike taking the union, stacking would result in a rank-deficient matrix $\tilde{B}^{(n+1)}$ due to repeated columns (modulo error). However, this does not induce a major drawback as in practice one can implement an SVD (Singular Value Decomposition)-based least-squares solver which is stable for rank-deficient matrices [111]. In PyTorch's LAPACK least-squares solver [12], this amounts to running `torch.linalg.lstsq(`$\tilde{B}^{(n+1)}$`, y, driver = gelsd)` where '`gelsd`' uses tridiagonalization SVD and thus produces a stable solution. As a result, the following proposition is immediate.

Proposition 5.1 (Soft-CoSaMP asymptotically converges to CoSaMP). *Let $v^{(n)} \in \mathbb{R}^N$, $\Pi^{(n)} \in \mathbb{R}^{3k \times N}$, $Q^{(n)} \in \mathbb{R}^{k \times N}$ and $x^{(n)} \in \mathbb{C}^N$ be sequences generated by pCoSaMP (replace softsort by exact permuations in Algorithm 5.2); and likewise $\tilde{v}^{(n)} \in \mathbb{R}^N$, $\tilde{\Pi}^{(n)} \in \mathbb{R}^{3k \times N}$, $\tilde{Q}^{(n)} \in \mathbb{R}^{k \times N}$ and $\tilde{x}^{(n)} \in \mathbb{C}^N$ be sequences generated by Algorithm 5.2. Then, $\tilde{x}^{(n)} \to x^{(n)}$ as $\tau \to 0$.*

Proof. The result directly follows from Proposition 4.3(c). If $\tau \to 0$, then $\tilde{\Pi}^{(n)} \to \Pi^{(n)}$ and $\tilde{Q}^{(n)} \to Q^{(n)}$ and Soft-CoSaMP coincides with pCoSaMP and thus with CoSaMP. □

Algorithm 5.2 Soft-CoSaMP.

Input: desired sparsity k, mixing matrix $A \in \mathbb{C}^{m\times N}$ with ℓ^2-normalized columns, observation vector $y \in \mathbb{C}^m$, temperature parameter τ of softsort.

Let $\tilde{x}^{(0)} = 0$ and $\tilde{\Pi}^{(0)} = [\,]$.
For $n = 0, \dots, \overline{n}$ repeat:

$$\begin{cases} \tilde{P}^{(n+1)} = \operatorname{softsort}_\tau\left(\left|\left(A^*(y - Ax^{(n)})\right)\right|\right), \\ \tilde{\Pi}^{(n+1)} = \left[\tilde{Q}^{(n)}(1:k,:); \tilde{P}^{(n+1)}(1:2k,:)\right], \end{cases} \tag{Soft-CoSaMP.1}$$

$$\begin{cases} \tilde{B}^{(n+1)} = A(\tilde{\Pi}^{(n+1)})^\top, \\ \tilde{w}^{(n+1)} = \operatorname{argmin}_{z\in\mathbb{C}^{3k}} \|y - \tilde{B}^{(n+1)}z\|_2^2, \\ \tilde{u}^{(n+1)} = (\tilde{\pi}^{(n+1)})^\top \tilde{w}^{(n+1)}, \end{cases} \tag{Soft-CoSaMP.2}$$

$$\begin{cases} \tilde{Q}^{(n+1)} = \operatorname{softsort}_\tau(|\tilde{u}^{(n+1)}|), \\ \tilde{q}^{(n+1)} = \sum_{i=1}^k \tilde{Q}^{(n+1)}(i,:), \\ \tilde{x}^{(n+1)} = \tilde{q}^{(n+1)} \odot \tilde{u}^{(n+1)}, \end{cases} \tag{Soft-CoSaMP.3}$$

Output: Approximately k-sparse vector $\hat{x} = x^{(\overline{n})} \in \mathbb{C}^N$.

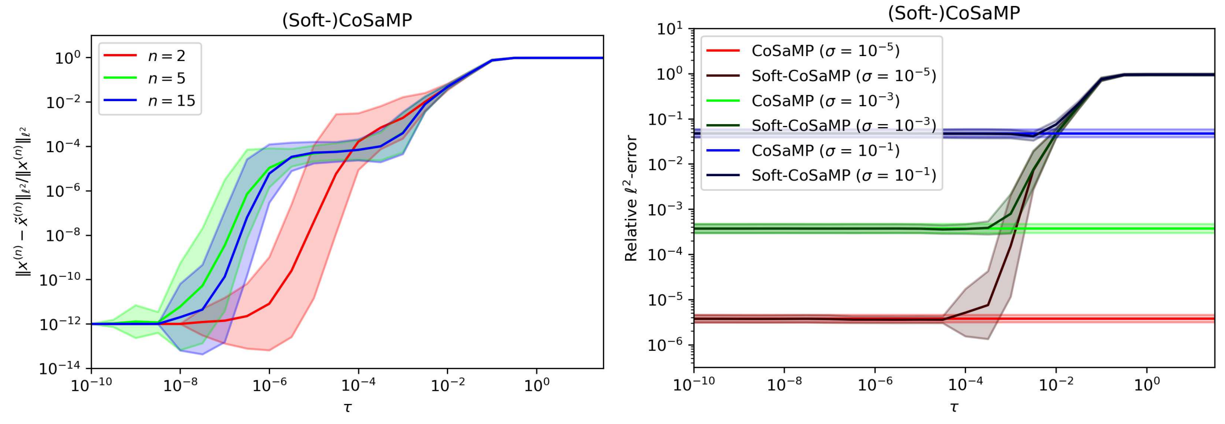

Figure 5.3: Difference and relative ℓ^2-error as a function of τ for CoSaMP and Soft-CoSaMP.

5.2.3 Numerical experiments

We plot the difference and relative ℓ^2-error of CoSaMP and Soft-CoSaMP for various iteration counts and noise levels as a function of τ similar to Sections 4.5.1 and 5.1. We also run the same numerical setting as those sections with the only difference that $n \in \{2, 5, 15\}$. Results of this experiment is exhibited in Figure 5.3. These figures vividly illustrate that Soft-CoSaMP serves as a good approximation for CoSaMP controlled by τ, validating Proposition 5.1.

5.3 Median recovery: Towards deterministic compressed sensing

In this section, we demonstrate that signal structure can be leveraged in conjunction with sparse recovery algorithms beyond greedy methods, all within the same unfolding framework introduced in Chapter 4. Up to this point our focus has been on the compressed sensing setting, with measurement matrices either drawn randomly from subgaussian distributions or derived from bounded orthonormal systems such as partial Fourier matrices. These settings are favorable because they yield matrices satisfying the (weighted) Restricted Isometry Property ((w)RIP), enabling signal recovery with a number of measurements proportional to the sparsity level. However, such guarantees are inherently probabilistic and typically take the form

$$\mathbb{P}\{\text{reconstruction failure}\} \leq \text{something small}$$

provided $m \gtrsim s\log(N/s)$. In this section, we shift our focus toward deterministic constructions of measurement matrices that enable fast and memory-efficient recovery with deterministic guarantees, coming at the cost of quadratic scaling, i.e., $m \gtrsim s^2 \times$ log factors. We begin by showing how to efficiently construct such deterministic matrices from a number-theoretic perspective that still satisfy RIP.

However, combining these matrices with standard compressed sensing algorithms still does not result in sublinear-time computational complexity, i.e., the execution time growing smaller than N (better computational complexity than $\mathcal{O}(N)$). The bottleneck is the recovery process that does not take into account the deterministic structure. To address this, we consider a combinatorial recovery method known as median recovery, which we motivate using an alternative to RIP that facilitates sublinear-time reconstruction. We then propose a natural weighted extension of this algorithm, which we unfold within the same framework introduced in Chapter 4. We conclude the section with numerical experiments that validate the practicality of this approach, while leaving a deeper theoretical analysis to future work.

5.3.1 Deterministic matrix construction

The core idea is to design a matrix $\mathcal{M}$ with K ones in each of its columns. By carefully selecting the locations of these ones, one can obtain matrices with low coherence, thus satisfying the RIP. This construction, originally presented in [68], is included here for completeness.

Deterministic matrix construction [44, 68]: Define $p_0 = 1$ and let p_l be the lth prime natural number. Thus, we have $p_0 = 1, p_1 = 2, p_2 = 3, p_3 = 5, \dots$. Choose $q \in \mathbb{N}$ so that $p_{q-1} < s \leq p_q$. We will use the first $K \in \mathbb{N}$ (to be specified later) primes no smaller than s

$$s \leq p_q < p_{q+1} < \cdots < p_{q+K-1}$$

to create a measurement matrix

$$\mathcal{M} = \left\{0, \frac{1}{\sqrt{K}}\right\}^{(\sum_{j=0}^{K-1} p_{q+j})\times N}.$$

We create $\mathcal{M}$ as follows: We produce a row $r_{j,h}$, $j \in [0, K) \cap \mathbb{Z}$ and $h \in [0, p_{q+j}) \cap \mathbb{Z}$, in $\mathcal{M}$ for each possible residue of each of our p_{q+j} primes. Each $r_{j,h}$ row's nth entry, $n \in [0, N) \cap \mathbb{Z}$, is given by

$$(r_{j,h})_n = \begin{cases} 1 & n \equiv h \text{ mod } p_{q+j} \\ 0 & \text{otherwise} \end{cases}.$$

We then set

$$\mathcal{M} = \frac{1}{\sqrt{K}} \begin{bmatrix} r_{0,0} & r_{0,1} & \dots & r_{0,p_q-1} & r_{1,0} & \dots & r_{1,p_{q+1}-1} & \dots & r_{K-1,p_{q+K-1}-1} \end{bmatrix}^\top. \quad (95)$$

Example 5.2. *We let $N = 6$ and $K = 2$. With $s = 2$, this leads to*

$$\begin{array}{ll} & \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{array}{l} n \equiv 0 \quad \text{mod } 2 \\ n \equiv 1 \quad \text{mod } 2 \\ n \equiv 0 \quad \text{mod } 3 \\ n \equiv 1 \quad \text{mod } 3 \\ n \equiv 2 \quad \text{mod } 3 \end{array} & \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix} = \mathcal{M} \end{array}$$

As is evident from the construction, these matrices can be generated using only the modulo operation, which allows for extremely fast (sublinear-time) construction. Moreover, unlike Gaussian matrices that require storing all entries, these structured matrices can be efficiently represented by simply storing the positions of the ones in each column, resulting in substantial memory savings. Matrix-vector multiplication is also highly efficient, reducing to summing over selected entries of the vector, which significantly accelerates computation.

We now introduce an alternative to the RIP that enables deterministic reconstruction with exact, non-probabilistic recovery guarantees. To proceed, we first present an auxiliary definition.

Definition 5.3. *If $\mathcal{M}$ has at least K ones in every column, then $\mathcal{M}(n) \in \{0, 1\}^{K\times N}$ for $n \in [N]$ is the $K \times N$ submatrix of $\mathcal{M}$ created by selecting the first K rows of $\mathcal{M}$ with nonzero entries in the nth column.*

Example 5.4. *For the matrix of Example 5.2, we have*

$$\mathcal{M}(2) = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}.$$

Now we are in a position to present an alternative for RIP in the deterministic setting.

Definition 5.5 (Majority (s, K)-reconstructing)**.** *Let $s \in \mathbb{N}$, and $\mathcal{M}(n) \in \{0,1\}^{K\times N}$ have at least $K \in [m]$ ones in every column. We will say that $\mathcal{M}$ is majority (s,K)-reconstructing for $x \in \mathbb{C}^N$ if the set*

$$B_n := \{j : |(\mathcal{M}(n)x)_j - x_n| \le (1/s)\|x - H_s(x)\|_1\} \subseteq [K]$$

has cardinality $|B_n| > K/2$ for all $n \in [N]$.

This means that the effect of $\mathcal{M}$ on x, characterized by the submatrices $\mathcal{M}(n)$, must not allow for distortion beyond the median, reminding one of the near-isometry characteristic of RIP matrices.

5.3.2 Median recovery algorithm

A direct consequence of Theorem 5.5 is that since the set B_n has cardinality greater than $K/2$, for any $n \in \mathbb{N}$ it is always true that $|\text{median}(\mathcal{M}(n)x) - x_n|$ can become small and controlled by s. This motivates the simple and intuitive technique presented in Algorithm 5.3 [69], where $y|_{\mathcal{M}(n)} \in \mathbb{C}^K$ is the restriction of y to K nonzero rows in $\mathcal{M}$'s nth column and $H_k(\cdot)$ is the hard-thresholding operator $H_k : \mathbb{C}^N \to \mathbb{C}^N$, that retains the top k entries (in absolute value) and sets the rest to zero. This algorithms runs in $\mathcal{O}(m + |S| \max(K, \log|S|))$, achieving a sublinear computational complexity when $|S| \ll N$, i.e., when combined with a sublinear support estimator [33, 34]–that we do not include here.

Algorithm 5.3 Median Recovery (MR).

Input: desired sparsity s, observation vector $y = \mathcal{M}x + e \in \mathbb{C}^m$, support $S \subseteq [N]$.

$$z(n) = \begin{cases} \text{median}(y|_{\mathcal{M}(n)}) & n \in S \\ 0 & \text{otherwise} \end{cases}, \tag{MR.1}$$

$$x = H_k(z),\ k = \min\{2s, |S|\}. \tag{MR.2}$$

Output: $\min(2s, |S|)$-sparse vector $\hat{x} \in \mathbb{C}^N$.

Algorithm 5.3 is inherently suited to incorporate prior information. One can intuitively interpret the algorithm as viewing the measurement vector from $|S|$ different angles, where each set of ones in $\mathcal{M}(n)$ selects corresponding entries from y, and choosing elements that give the most dominant views in median. While $S = [N]$ leads to a brute-force search over all directions, $S \subset [N]$ restricts attention to a subset informed by prior knowledge.

With this perspective, generalizing the algorithm to the weighted case becomes natural: the prior weights influence which angles (i.e., indices) are emphasized during selection. This modification directly affects step (MR.1) of Algorithm 5.3, which can be updated as follows:

$$z(n) = w(n)\text{median}(y|_{\mathcal{M}(n)}),\ n \in [N].$$

Furthermore, for unrolling MR (in the simplest way), noting that median is an algorithmic differentiable operator (see Remark 4.1), one requires only to tackle the hard-threholding operator. Using the same technique as in Section 4.3 results in the unrolled MR or Soft-MR in Algorithm 5.4.

Algorithm 5.4 Soft-MR.

Input: desired sparsity s, observation vector $y = \mathcal{M}x + e \in \mathbb{C}^m$, weight vector $w \in \mathbb{R}^N$, temperature parameter of softsort τ.

$$z(n) = \text{median}(y|_{\mathcal{M}(n)}),\ n \in [N] \tag{Soft-MR.1}$$

$$\begin{cases} q = \sum_{i=1}^{2s} \text{softsort}_\tau(|z|)(i,:) \\ x = q \odot z \end{cases}. \tag{Soft-MR.2}$$

Output: Approximately $2s$-sparse vector $\hat{x} \in \mathbb{C}^N$.

One notices that asymptotic convergence of MR to Soft-MR in sense of Proposition 5.1 is immediate when $S = [N]$, so we omit it to avoid repetition. However, with this construction we cannot include S unless we utilize oracle weights.

5.3.3 Numerical experiments

We conclude this section by illustrating numerical experiments in form of the results presented in Section 5.1, i.e., (1) shaded plot showing that Soft-MR can well approximate MR as $\tau \to 0$, and (2) some initial results showing that Soft-MR is trainable and outperforms MR in undersampled regime. The following represents our numerical setting:

$$\begin{aligned}\text{Shaded plot: } & N = 1000,\ K = 16,\ m = 622,\ s = 8,\ \sigma = 10^{-3},\ N_{\text{repeat}} = 30\\ \text{Training: } & N = 600,\ K = 10,\ m = 264,\ s = 10,\ \sigma = 10^{-3},\ \tau = 5 \times 10^{-3},\ lr = 10^{-2},\end{aligned}$$

and other training parameters as in previous experiments. The dataset to train the network consists of sparse signals with nonzero entries from only the support's first s elements and the network is expected to learn the structure without access to any oracle knowledge. Figure 5.4 depicts Soft-MR vs. MR relative ℓ^2-error with respect to the true solution, in addition to plots on Soft-MR training progress, learned weights and relative ℓ^2-error boxplots.

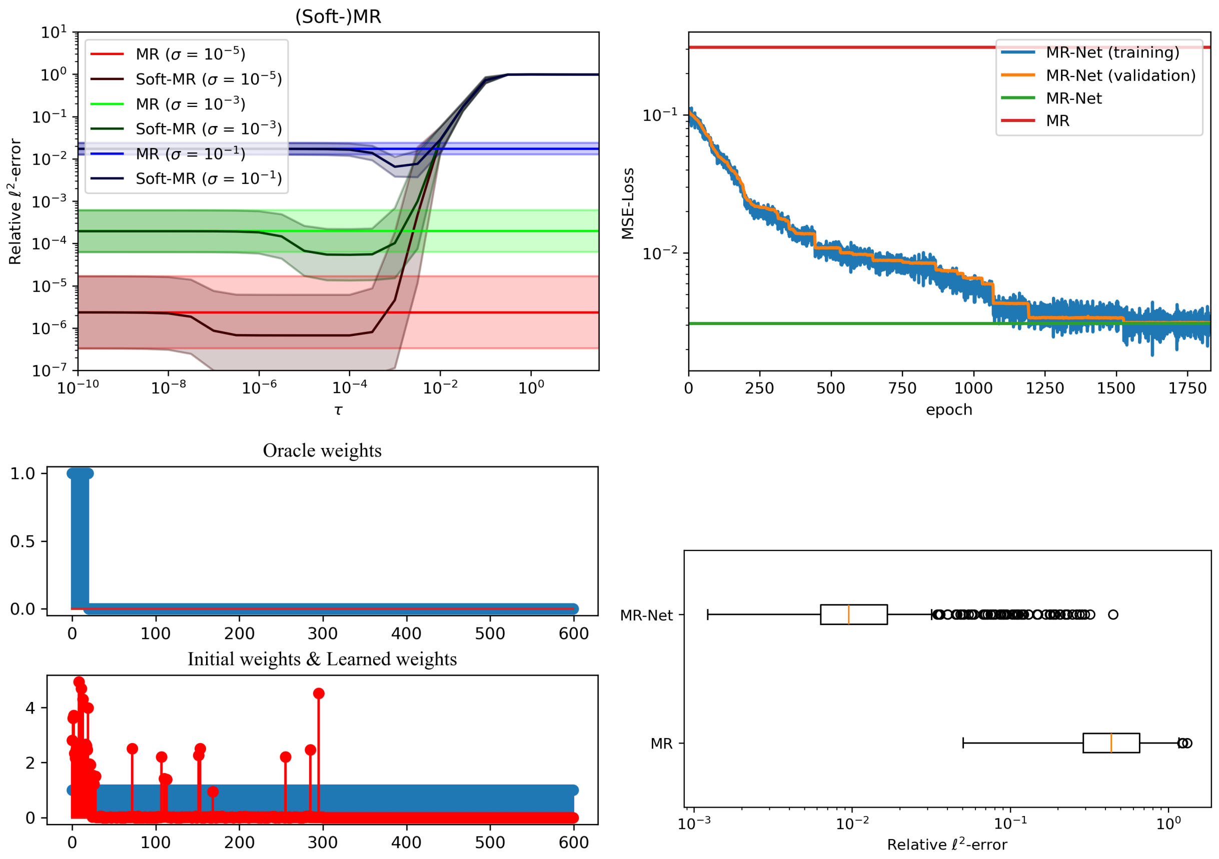

Figure 5.4: top left: Soft-MR vs. MR relative ℓ^2-error, top right: training progress of unrolled MR, bottom left: learned vs. oracle weights, and bottom right: MR-Net vs. MR relative ℓ^2-error boxplots.

5.4 Beyond weights

In this section we briefly demonstrate numerically that the proposed unrolling framework can go beyond learning weights. For simplicity, we stick to the real setting where $A \in \mathbb{R}^{m\times N}$ is a Gaussian matrix (see Section 4.5). We focus on IHT-Net as a case example with the recursive relation

$$x^{(n+1)} = H_k(x^{(n)} + \eta A^*(y - Ax^{(n)})) = H_k((I - \eta A^*A)x^{(n)} + \eta A^*y) = H_k((I - WA)x^{(n)} + Wy), \tag{96}$$

where in the last equation we defined $W := \eta A^*$, $W \in \mathbb{R}^{N\times m}$. This provides a filter perspective to the algorithm, i.e., at each iteration $x^{(n)}$ and y are passed through the filter matrices $I - WA$ and W respectively with "weight coupling", to produce the output $x^{(n+1)}$. This scheme has been previously exploited in the literature in particular for ISTA algorithm whose recursive formulation bears resemblance to our IHT and also for ℓ^0-regularized hard-thresholding [122, 124] (see Section 4.1.2). Now, we let W be trainable, and with that we pursue two goals: (1) roughly speaking, A^* can be seen as a proxy for the inverse mapping and plays an important role in creating the reconstruction path to $x^{(n+1)}$, so by learning this parameter, we aim to improve the performance of IHT beyond what is achievable with the original fixed parameter, while keeping the number of iterations (or layers) constant. (2) By defining $W = \eta A^*$, we no longer have to worry about cross-validating the step-size η and its contribution is learned throughout training.

Numerical experiments. We implement the same experimental procedure as in Section 4.5.2 to train IHT-Net, with W is the training parameter and with the difference that we allow the signal support to be generated from the whole ambient space. We generate the dataset with parameters

$$N = 256, \quad s = 10, \quad m = 128, \quad \sigma = 10^{-3}, \quad N_{\text{train}} = 1024, \quad N_{\text{val}} = 1024.$$

We construct IHT-Net with L layers and employ weight-sharing across layers, i.e., $W^{(1)} = W^{(2)} = \cdots = W^{(L)}$. The network training parameters are:

$$L = 7, \quad N_{\text{epoch}} = 1000, \quad lr = 10^{-3}, \quad \tau = 10^{-3}.$$

The results, presented in Figure 5.5, compare IHT-Net of L layers with L iterations of IHT algorithm. For IHT, we set $\eta = 0.7$, following the same procedure as in Section 2.4 which is also included in Figure 5.5 for completeness. The top row shows the evolution of MSE_{tr} and MSE_{val}, truncated at the best checkpoint. Boxplots of the relative ℓ^2-error are also presented confirming near noise-level signal recovery. This experiment vividly illustrates learnability of other parameters of IHT-Net beyond weights, gaining considerable advantage in performance from the same number of iterations.

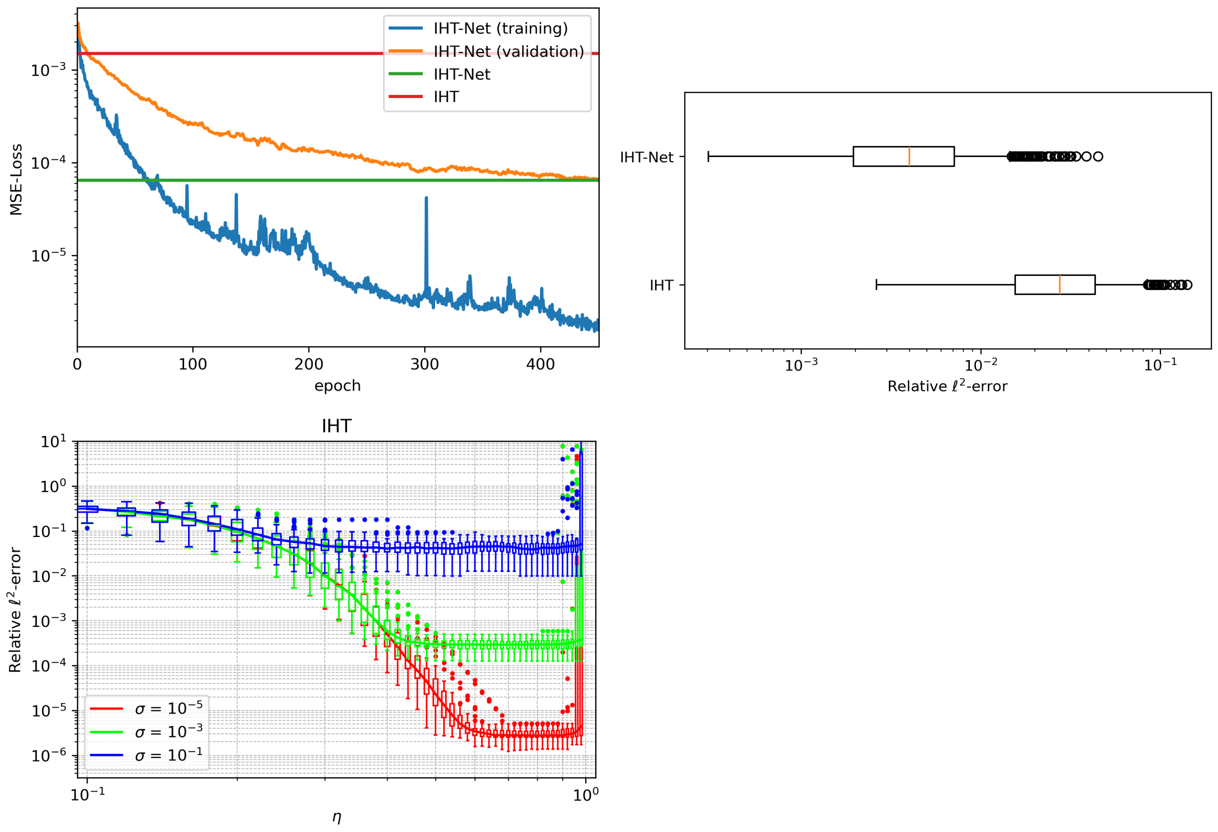

Figure 5.5: Top: MSE-Loss, and relative ℓ^2-error boxplots for IHT-Net. Bottom: Relative ℓ^2-error as a function of the step-size η for IHT.

Chapter 6

Conclusion and future work

In this section we conclude our journey with some final remarks and open research problems. Some of these research directions were previously mentioned in Sections 2.7 and 4.9, but we also include here for completeness.

In this thesis, we studied greedy sparse recovery algorithms from generalization to the weighted sparsity setting to neural network unrolling. In Chapter 2 and Chapter 3 we generalized OMP to the weighted setup, allowing one to incorporate prior knowledge into the reconstruction scheme. Then, in Chapters 4 and 5 we implemented iterations of greedy algorithms onto neural network layers. For OMP and IHT this comes in full circle, a complete pipeline from weighted standard sparse recovery algorithm to a data-driven technique. This is appropriate in different setups when prior knowledge is available or when structure is supposed to be learned from data.

In the sequel we briefly highlight our contributions for each part while discussing related research directions. We will conclude by presenting other research avenues concerning both Part I and Part II.

6.1 Part I: weighted generalizations

Our weighted generalizations encompass two strategies: (1) adopting a loss-function perspective in Chapter 2, we develop ℓ^0 and ℓ^1 variants of the weighted LASSO, SR-LASSO, and LAD-LASSO, with corresponding greedy index selection rules that admit explicit formulas (see Theorems 2.5 to 2.7 and Section 2.6); (2) in Chapter 3, directly embedding weights into the greedy index selection quantity by means of rescaling, we not only justified this choice but also established a novel RIP-based theoretical guarantee. Regarding these chapters, there are several roads not taken left for future research:

- The generalization of loss function-based OMP through the lens of Chapter 2 to other regularizers such as total variation, nuclear norm, $\ell^p - \ell^q$ norms, and group (or joint) sparsity could be of potential interest.

- One might also attempt to apply the weighting generalizations of either of Chapters 2 and 3 to other greedy algorithms that follow a similar construction similar to OMP, such as CoSaMP, or its more recently proposed sublinear-time variant [33, 34, 69]. In this area, RIP-based recovery guarantees and their application in different scenarios would be of high importance.

- The development of RIP or coherence-based recovery theorems for algorithms of Chapter 2 is another important open problem.

- Theoretical guarantees for MR in the weighted setting would be of interest, which demands for a majority reconstruction-based analysis (instead of RIP-based) and requires tools from number theory. Also, there seems to be other weighting strategies, e.g., embedding the weights to the sampling matrix to learn the sampling pattern, also related to Part II.

- Having a weighted theoretical guarantee for WOMP, the immediate next compressed sensing question is whether this offers any improvement on the sample complexity over standard OMP.

6.2 Part II: neural network unrolling

We then unrolled OMP and IHT in Chapter 4, and CoSaMP and MR in Chapter 5 into neural networks with the embedded weights as trainable parameter. To this purpose, we addressed the non-differentiability challenge posed by the argsort operator in greedy sparse recovery algorithms, and proposed differentiable approximations, Soft-OMP, Soft-IHT, Soft-CoSaMP and Soft-MR, respectively. We also established rigorous theoretical error bounds for Soft-OMP and Soft-IHT, in addition to asymptotic convergence for all of these algorithms in relation to their original counterparts. However, there are several potential avenues for future research.

- Softsort was utilized primarily due to its simple formulation and interpretability. That said, a comprehensive comparison of greedy networks based on the various differentiable sorting methods available in the literature remains a path to be pursued.

- We presented some early results in MR training, aiming at showing its trainability, but more extensive numerical experiments remain to be conducted.

- So far our numerical tests on CoSaMP training were not completely successful, facing exploding gradient issue. Whether this is a matter of hyper parameter tuning or a fundamental drawback of softsort is a question to be answered. However, if CoSaMP is successfully trained, the next immediate candidate would be sublinear-CoSaMP allowing for very fast sublinear-time recovery.

- There are several other candidates to be unrolled worth mentioning including, but not limited to, CoSaMP, subspace pursuit and most importantly our greedy weighted-LASSO algorithms. In particular for the latter, the tuning parameter λ, whose choice is usually challenging in LASSO-based methods, could be directly learned from the data through unrolling. Furthermore, as greedy LASSO is robust to the iteration number, its corresponding neural network's depth is also expected to be disentangled from signal sparsity unlike OMP-Net.

- The extension of unrolled networks to other architectures, e.g., ResNet or CNN could be worth considering. Additionally, several interesting extensions could be explored, including experiments with varying numbers of layers, learning additional parameters beyond weights (e.g., the step size η, the full matrix A^*, $I - A^*A$, etc.) similar to Section 5.4, employing distinct parameters across layers, and realization of stopping criterion.

- Our theoretical analysis in Chapter 4 illustrates that Soft-OMP and Soft-IHT are good approximations of OMP and IHT. However, a complete learning theory that also involves the training process is still lacking. Recovery guarantees showing that Soft-MR and Soft-CoSaMP can approximate their original counterparts is also missing and remains a future research direction.

6.3 Further research directions

In this work we tested our methods on synthetic data, but more experiments on high-dimensional function approximation and applications such as biomedical signal analysis is of particular interest. One potential application could be video reconstruction from compressive measurements arising in contexts such as *dynamic MRI*, where one can incorporate information on the ambient signal of the previously reconstructed frames into weights in order to improve the reconstruction quality of subsequent frames (see, e.g., [56]). OMP-based algorithms do not seem to be the right candidates in high-dimensional applications as their number of iterations are entangled with the signal sparsity, which could become moderately large (e.g., of the order of 10^2) in those scenarios. IHT, CoSaMP and in particular MR could be interesting candidates to explore. Another potential application could be in reconstruction of (pseudo-) periodic signals such as *Electrocardiogram* (ECG), where information of one period can be leveraged in reconstruction of the subsequent periods. Another application of unrolled greedy networks could be to embed these networks into dictionary learning pipelines to achieve an end-to-end training process. Beyond these, there are many other application domains in inverse problems remained to be explored that fully fit into our frameworks, both to promote prior information and in data-driven contexts.

Bibliography

[1] Ben Adcock. Infinite-dimensional compressed sensing and function interpolation. *Foundations of Computational Mathematics*, 18(3):661–701, 2018.

[2] Ben Adcock and Simone Brugiapaglia. Sparse approximation of multivariate functions from small datasets via weighted orthogonal matching pursuit. In Spencer J. Sherwin, David Moxey, Joaquim Peiró, Peter E. Vincent, and Christoph Schwab, editors, *Spectral and High Order Methods for Partial Differential Equations ICOSAHOM 2018*, pages 611–621, Cham, 2020. Springer International Publishing. ISBN 978-3-030-39647-3.

[3] Ben Adcock and Anders C. Hansen. *Compressive Imaging: Structure, Sampling, Learning*. Cambridge University Press, Cambridge, UK, 2021.

[4] Ben Adcock, Simone Brugiapaglia, and Clayton G. Webster. Compressed sensing approaches for polynomial approximation of high-dimensional functions. In Holger Boche, Giuseppe Caire, Robert Calderbank, Maximilian März, Gitta Kutyniok, and Rudolf Mathar, editors, *Compressed Sensing and its Applications: Second International MATHEON Conference 2015*, pages 93–124, Cham, 2017. Springer International Publishing.

[5] Ben Adcock, Anders C. Hansen, Clarice Poon, and Bogdan Roman. Breaking the coherence barrier: A new theory for compressed sensing. In *Forum of Mathematics, Sigma*, volume 5. Cambridge University Press, 2017.

[6] Ben Adcock, Anyi Bao, John D Jakeman, and Akil Narayan. Compressed sensing with sparse corruptions: Fault-tolerant sparse collocation approximations. *SIAM/ASA Journal on Uncertainty Quantification*, 6(4):1424–1453, 2018.

[7] Ben Adcock, Anyi Bao, and Simone Brugiapaglia. Correcting for unknown errors in sparse high-dimensional function approximation. *Numerische Mathematik*, 142(3):667–711, 2019.

[8] Ben Adcock, Anne Gelb, Guohui Song, and Yi Sui. Joint sparse recovery based on variances. *SIAM Journal on Scientific Computing*, 41(1):A246–A268, 2019.

[9] Ben Adcock, Simone Brugiapaglia, and Matthew King-Roskamp. The benefits of acting locally: Reconstruction algorithms for sparse in levels signals with stable and robust recovery guarantees. *IEEE Transactions on Signal Processing*, 69:3160–3175, 2021.

[10] Ben Adcock, Simone Brugiapaglia, and Matthew King-Roskamp. Do log factors matter? On optimal wavelet approximation and the foundations of compressed sensing. *Foundations of Computational Mathematics*, 22(1):99–159, 2022.

[11] Ben Adcock, Simone Brugiapaglia, and Clayton G. Webster. *Sparse Polynomial Approximation of High-Dimensional Functions*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2022.

[12] Edward Anderson, Zhaojun Bai, Christian Bischof, L Susan Blackford, James Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, Alan McKenney, et al. *LAPACK Users' Guide*. SIAM, 1999.

[13] Sanjeev Arora and Boaz Barak. *Computational Complexity: a Modern Approach*. Cambridge University Press, 2009.

[14] Bubacarr Bah and Rachel Ward. The sample complexity of weighted sparse approximation. *IEEE Transactions on Signal Processing*, 64(12):3145–3155, 2016.

[15] Richard G. Baraniuk, Volkan Cevher, Marco F. Duarte, and Chinmay Hegde. Model-based compressive sensing. *IEEE Transactions on Information Theory*, 56(4):1982–2001, 2010.

[16] Arash Behboodi, Holger Rauhut, and Ekkehard Schnoor. Compressive sensing and neural networks from a statistical learning perspective. In *Compressed Sensing in Information Processing*, pages 247–277. Springer, 2022.

[17] Alexandre Belloni, Victor Chernozhukov, and Lie Wang. Square-root lasso: pivotal recovery of sparse signals via conic programming. *Biometrika*, 98(4):791–806, 2011.

[18] Peter J. Bickel, Ya'acov Ritov, and Alexandre B. Tsybakov. Simultaneous analysis of Lasso and Dantzig selector. *The Annals of Statistics*, 37(4):1705–1732, 2009.

[19] Mathieu Blondel, Olivier Teboul, Quentin Berthet, and Josip Djolonga. Fast differentiable sorting and ranking. In *International Conference on Machine Learning*, pages 950–959. PMLR, 2020.

[20] Peter Bloomfield and William L. Steiger. *Least Absolute Deviations: Theory, Applications, and Algorithms*. Birkhäuser, Boston, MA, 1983.

[21] Thomas Blumensath and Mike E Davies. Iterative thresholding for sparse approximations. *Journal of Fourier Analysis and Applications*, 14:629–654, 2008.

[22] Thomas Blumensath and Mike E Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009.

[23] Jean-Luc Bouchot, Holger Rauhut, and Christoph Schwab. Multi-level compressed sensing Petrov-Galerkin discretization of high-dimensional parametric PDEs. *arXiv preprint arXiv:1701.01671*, 2017.

[24] Simone Brugiapaglia, Sjoerd Dirksen, Hans Christian Jung, and Holger Rauhut. Sparse recovery in bounded riesz systems with applications to numerical methods for pdes. *Applied and Computational Harmonic Analysis*, 53:231–269, 2021.

[25] Emmanuel J. Candès and Terence Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[26] Emmanuel J. Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

[27] Emmanuel J Candès, Justin K Romberg, and Terence Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 59(8):1207–1223, 2006.

[28] Emmanuel J. Candès, Michael B. Wakin, and Stephen P. Boyd. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications*, 14(5):877–905, 2008.

[29] Antonin Chambolle and Thomas Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40:120–145, 2011.

[30] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, 2001.

[31] Ziheng Chen, Sichen Zhong, Jianshu Chen, and Yue Zhao. Deeppursuit: Uniting classical wisdom and deep rl for sparse recovery. In *2021 55th Asilomar Conference on Signals, Systems, and Computers*, pages 1361–1366. IEEE, 2021.

[32] Abdellah Chkifa, Nick Dexter, Hoang Tran, and Clayton Webster. Polynomial approximation via compressed sensing of high-dimensional functions on lower sets. *Mathematics of Computation*, 87(311):1415–1450, 2018.

[33] Bosu Choi, Mark Iwen, and Toni Volkmer. Sparse harmonic transforms II: best s-term approximation guarantees for bounded orthonormal product bases in sublinear-time. *Numerische Mathematik*, 148:293–362, 2021.

[34] Bosu Choi, Mark A Iwen, and Felix Krahmer. Sparse harmonic transforms: a new class of sublinear-time algorithms for learning functions of many variables. *Foundations of Computational Mathematics*, 21(2):275–329, 2021.

[35] Charles Q Choi. 7 revealing ways ais fail: Neural networks can be disastrously brittle, forgetful, and surprisingly bad at math. *IEEE Spectrum*, 58(10):42–47, 2021.

[36] Albert Cohen, Wolfgang Dahmen, and Ronald DeVore. Orthogonal matching pursuit under the restricted isometry property. *Constructive Approximation*, 45(1):113–127, 2017.

[37] Matthew J Colbrook, Vegard Antun, and Anders C. Hansen. The difficulty of computing stable and accurate neural networks: On the barriers of deep learning and smale's 18th problem. *Proceedings of the National Academy of Sciences*, 119(12):e2107151119, 2022.

[38] Marco Cuturi, Olivier Teboul, and Jean-Philippe Vert. Differentiable ranking and sorting using optimal transport. *Advances in Neural Information Processing Systems*, 32, 2019.

[39] Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory*, 55(5):2230–2249, 2009.

[40] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, 57(11):1413–1457, 2004.

[41] Michael E. Davies and Thomas Blumensath. Faster & greedier: algorithms for sparse reconstruction of large datasets. In *2008 3rd International Symposium on Communications, Control and Signal Processing*, pages 774–779. IEEE, 2008.

[42] Geoffrey M Davis, Stephane G Mallat, and Zhifeng Zhang. Adaptive time-frequency decompositions. *Optical Engineering*, 33(7):2183–2191, 1994.

[43] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[44] Ronald A DeVore. Deterministic constructions of compressed sensing matrices. *Journal of Complexity*, 23(4-6):918–925, 2007.

[45] David L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4): 1289–1306, 2006.

[46] David L. Donoho, Yaakov Tsaig, Iddo Drori, and Jean-Luc Starck. Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 58(2):1094–1121, 2012.

[47] Marco F. Duarte and Yonina C. Eldar. Structured compressed sensing: From theory to applications. *IEEE Transactions on Signal Processing*, 59(9):4053–4085, 2011.

[48] Marco F Duarte, Mark A Davenport, Dharmpal Takhar, Jason N Laska, Ting Sun, Kevin F Kelly, and Richard G Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2):83–91, 2008.

[49] Michael Elad. *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*. Springer, New York, NY, 2010.

[50] Yonina C. Eldar and Gitta Kutyniok. *Compressed Sensing: Theory and Applications*. Cambridge University Press, Cambridge, UK, 2012.

[51] Yonina C. Eldar, Patrick Kuppinger, and Helmut Bolcskei. Block-sparse signals: Uncertainty relations and efficient recovery. *IEEE Transactions on Signal Processing*, 58(6): 3042–3054, 2010.

[52] Massimo Fornasier and Holger Rauhut. Iterative thresholding algorithms. *Applied and Computational Harmonic Analysis*, 25(2):187–208, 2008.

[53] Simon Foucart. Flavors of compressive sensing. In *Approximation Theory XV: San Antonio 2016 15*, pages 61–104. Springer, 2017.

[54] Simon Foucart. The sparsity of lasso-type minimizers. *Applied and Computational Harmonic Analysis*, 62:441–452, 2023.

[55] Simon Foucart and Holger Rauhut. *A Mathematical Introduction to Compressive Sensing*. Birkhäuser, New York, NY, 2013.

[56] Michael P Friedlander, Hassan Mansour, Rayan Saab, and Özgür Yilmaz. Recovering compressively sampled signals using partial support information. *IEEE Transactions on Information Theory*, 58(2):1122–1134, 2011.

[57] Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017.

[58] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[59] Michael Grant and Stephen P. Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008.

[60] Michael Grant and Stephen P. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx, March 2014.

[61] Karol Gregor and Yann LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th International Conference on Machine Learning*, pages 399–406, 2010.

[62] Aditya Grover, Eric Wang, Aaron Zweig, and Stefano Ermon. Stochastic optimization of sorting networks via continuous relaxations. *arXiv preprint arXiv:1903.08850*, 2019.

[63] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, NY, second edition, 2009.

[64] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. CRC Press, Boca Raton, FL, 2015.

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[66] Douglas Heaven et al. Why deep-learning ais are so easy to fool. *Nature*, 574(7777):163–166, 2019.

[67] Jian Huang, Shuangge Ma, and Cun-Hui Zhang. Adaptive Lasso for sparse high-dimensional regression models. *Statistica Sinica*, 18:1603–1618, 2008.

[68] Mark A Iwen. Simple deterministically constructible rip matrices with sublinear fourier sampling requirements. In *2009 43rd Annual Conference on Information Sciences and Systems*, pages 870–875. IEEE, 2009.

[69] Mark A Iwen. Combinatorial sublinear-time fourier algorithms. *Foundations of Computational Mathematics*, 10(3):303–338, 2010.

[70] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[71] Jason Jo. Iterative hard thresholding for weighted sparse approximation. *arXiv preprint arXiv:1312.3582*, 2013.

[72] M. Amin Khajehnejad, Weiyu Xu, A. Salman Avestimehr, and Babak Hassibi. Analyzing weighted ℓ_1 minimization for sparse recovery with nonuniform sparse models. *IEEE Transactions on Signal Processing*, 59(5):1985–2001, 2011.

[73] Rajaei Khatib, Dror Simon, and Michael Elad. Learned greedy method (lgm): A novel neural architecture for sparse coding and beyond. *Journal of Visual Communication and Image Representation*, 77:103095, 2021.

[74] Gitta Kutyniok. The mathematics of reliable artificial intelligence. *Collections*, 57(06), 2024.

[75] Ming-Jun Lai and Yang Wang. *Sparse Solutions of Underdetermined Linear Systems and Their Applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2021.

[76] Jason N. Laska, Mark A. Davenport, and Richard G. Baraniuk. Exact signal recovery from sparsely corrupted measurements through the pursuit of justice. In *2009 Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers*, pages 1556–1560, 2009.

[77] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 2002.

[78] Guo Zhu Li, De Qiang Wang, Zi Kai Zhang, and Zhi Yong Li. A weighted OMP algorithm for compressive UWB channel estimation. In *Applied Mechanics and Materials*, volume 392, pages 852–856, 2013.

[79] Xiaodong Li. Compressed sensing and matrix completion with constant proportion of corruptions. *Constructive Approximation*, 37:73–99, 2013.

[80] Yingying Li and Stanley Osher. Coordinate descent optimization for ℓ^1 minimization with application to compressed sensing; a greedy algorithm. *Inverse Problems and Imaging*, 3 (3):487–503, 2009.

[81] Dong Liang, Jing Cheng, Ziwen Ke, and Leslie Ying. Deep magnetic resonance image reconstruction: Inverse problems meet neural networks. *IEEE Signal Processing Magazine*, 37(1):141–151, 2020.

[82] Stéphane Mallat. *A Wavelet Tour of Signal Processing: The Sparse Way, 3rd Edition*. Elsevier Science, 2009.

[83] Stéphane G. Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

[84] Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. *arXiv preprint arXiv:1802.08665*, 2018.

[85] Qun Mo and Yi Shen. Remarks on the restricted isometry property in orthogonal matching pursuit algorithm. *arXiv preprint arXiv:1101.4458*, 2011.

[86] Sina Mohammad-Taheri and Simone Brugiapaglia. The greedy side of the lasso: New algorithms for weighted sparse recovery via loss function-based orthogonal matching pursuit. *Sampling Theory, Signal Processing, and Data Analysis*, 23(1):3, 2025.

[87] Sina Mohammad-Taheri, Matthew J Colbrook, and Simone Brugiapaglia. OMP-Net: Neural network unrolling of weighted Orthogonal Matching Pursuit. In *2024 International Workshop on the Theory of Computational Sensing and its Applications to Radar, Multimodal Sensing and Imaging (CoSeRa)*, pages 61–65. IEEE, 2024.

[88] Sina Mohammad-Taheri, Matthew J Colbrook, and Simone Brugiapaglia. Deep greedy unfolding: Sorting out argsorting in greedy sparse recovery algorithms. *arXiv preprint arXiv:2505.15661*, 2025.

[89] Vishal Monga, Yuelong Li, and Yonina C. Eldar. Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing. *IEEE Signal Processing Magazine*, 38 (2):18–44, 2021.

[90] APS Mosek and Denmark Copenhagen. The mosek optimization toolbox for matlab manual. version 9.0., 2019. URL http://docs.mosek.com/9.0/toolbox/index.html.

[91] Deanna Needell and Joel A. Tropp. CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301–321, 2009.

[92] Deanna Needell and Roman Vershynin. Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. *Foundations of Computational Mathematics*, 9(3):317–334, 2009.

[93] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, pages 40–44. IEEE, 1993.

[94] Ji Peng, Jerrad Hampton, and Alireza Doostan. A weighted ℓ_1-minimization approach for sparse polynomial chaos expansions. *Journal of Computational Physics*, 267:92–111, 2014.

[95] Hendrik Bernd Petersen and Peter Jung. Robust instance-optimal recovery of sparse signals at unknown noise levels. *Information and Inference: A Journal of the IMA*, 11(3):845–887, 2022.

[96] Sebastian Prillo and Julian Eisenschlos. Softsort: A continuous relaxation for the argsort operator. In *International Conference on Machine Learning*, pages 7793–7802. PMLR, 2020.

[97] GRB Prony. Essai experimental et analytique sur les lois de la dilatabilite de fluides elastiques et sur celles da la force expansion de la vapeur de l'alcool, a differentes temperatures. *Journal de l'Ecole Polytechnique*, 1(2), 1795.

[98] Holger Rauhut and Rachel Ward. Interpolation via weighted ℓ_1 minimization. *Applied and Computational Harmonic Analysis*, 40(2):321–351, 2016.

[99] Laura Rebollo-Neira and David Lowe. Optimized orthogonal matching pursuit approach. *IEEE Signal Processing Letters*, 9(4):137–140, 2002.

[100] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[101] Michael Eli Sander, Joan Puigcerver, Josip Djolonga, Gabriel Peyré, and Mathieu Blondel. Fast, differentiable and sparse top-k: a convex analysis perspective. In *International Conference on Machine Learning*, pages 29919–29936. PMLR, 2023.

[102] Fadil Santosa and William W. Symes. Linear inversion of band-limited reflection seismograms. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1307–1330, 1986.

[103] Jonathan Scarlett, Reinhard Heckel, Miguel RD Rodrigues, Paul Hand, and Yonina C Eldar. Theoretical perspectives on deep learning methods in inverse problems. *IEEE Journal on Selected Areas in Information Theory*, 3(3):433–453, 2022.

[104] Shai Shalev-Shwartz, Nathan Srebro, and Tong Zhang. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM Journal on Optimization*, 20(6): 2807–2832, 2010.

[105] Yi Shen, Bin Han, and Elena Braverman. Stable recovery of analysis based approaches. *Applied and Computational Harmonic Analysis*, 39(1):161–172, 2015.

[106] Ralph C. Smith. *Uncertainty Quantification: Theory, Implementation, and Applications*, volume 12. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2013.

[107] Ralph C Smith. *Uncertainty quantification: theory, implementation, and applications*. SIAM, 2024.

[108] Matus Telgarsky. Benefits of depth in neural networks. In *Conference on Learning Theory*, pages 1517–1539. PMLR, 2016.

[109] Vladimir Temlyakov. *Greedy Approximation*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, UK, 2011.

[110] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[111] Lloyd N Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 2022.

[112] Joel A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

[113] Joel A. Tropp and Anna C. Gilbert. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Transactions on Information Theory*, 53(12):4655–4666, 2007.

[114] Sara van de Geer. *Estimation and Testing Under Sparsity*. Lecture Notes in Mathematics. Springer Cham, Switzerland, 2016.

[115] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[116] Martin Vetterli, Pina Marziliano, and Thierry Blu. Sampling signals with finite rate of innovation. *IEEE Transactions on Signal Processing*, 50(6):1417–1428, 2002.

[117] Mathukumalli Vidyasagar. *An Introduction to Compressed Sensing*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2019.

[118] Hansheng Wang, Guodong Li, and Guohua Jiang. Robust regression shrinkage and consistent variable selection through the LAD-Lasso. *Journal of Business & Economic Statistics*, 25(3):347–355, 2007.

[119] Haohan Wang and Bhiksha Raj. On the origin of deep learning. *arXiv preprint arXiv:1702.07800*, 2017.

[120] Zhangyang Wang, Qing Ling, and Thomas Huang. Learning deep ℓ^0 encoders. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[121] Xiao-chuan Wu, Wei-bo Deng, and Ying-ning Dong. A weighted OMP algorithm for doppler superresolution. In *2013 Proceedings of the International Symposium on Antennas & Propagation*, volume 2, pages 1064–1067. IEEE, 2013.

[122] Bo Xin, Yizhou Wang, Wen Gao, David Wipf, and Baoyuan Wang. Maximal sparsity with deep networks? *Advances in Neural Information Processing Systems*, 29, 2016.

[123] Xiaohan Yu and Seung Jun Baek. Sufficient conditions on stable recovery of sparse signals with partial support information. *IEEE Signal Processing Letters*, 20(5):539–542, 2013.

[124] Jian Zhang and Bernard Ghanem. Ista-net: Interpretable optimization-inspired deep network for image compressive sensing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1828–1837, 2018.

[125] Tong Zhang. Sparse recovery with orthogonal matching pursuit under RIP. *IEEE Transactions on Information Theory*, 57(9):6215–6221, 2011.

[126] Hui Zou. The adaptive lasso and its oracle properties. *Journal of the American Statistical Association*, 101(476):1418–1429, 2006.