Analysis of algorithms for sparse signal representation

Stipić, Dorian

Analysis of algorithms for sparse signal representation

Stipić, Dorian

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Dorian Stipić

ANALYSIS OF ALGORITHMS FOR SPARSE SIGNAL REPRESENTATION

Diplomski rad

Voditelj rada:
prof. dr. sc. Zlatko Drmač

Zagreb, Srpanj 2020

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Mojim roditeljima, čija podrška i savjeti su me doveli tu gdje jesam

Contents

Introduction

The second half of the 20th century is known as the Digital Age, which is characterized by the rapid shift from traditional industry to an economy primarily based upon information technology. This historical period (which is the period we are still living in) has seen a great flourishment of new challenges, ideas and solutions; in the field of Mathematics, Physics, Biology, Chemistry, Economy, Information Technology etc. Many of these new fields are indeed strongly connected and such is the case of the topic of this Master's thesis which lies in the intersection of applied mathematics and signal processing.

In this work I analyze the principal aspects of the sparse representations of signals, give an intuitive explanation, present different efficient algorithms to solve the problem described and finally, in the last chapter, present one of many applications to signal processing.

At the heart of this discussion lies a simple linear system of equations, more precisely a full-rank matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ with $n < m$ generates an underdetermined system of linear equations $\mathbf{Ax} = \mathbf{b}$ having infinitely many solutions. The objective is to find its sparsest solution, i.e. the one with fewest non zero entries and answer some standard questions about the uniqueness of such solution, the methods to find it and if it can even be found in reasonable time.

In the first chapter I define and describe with mathematical formality the problem we are aiming to solve. The second chapter presents theoretical results for uniqueness of the solution in all mathematical beautiness. The third and fourth chapters present many different algorithms and their variants which are able to solve the defined problem, some theoretical guarantees are also given with a final performance demonstration. The fifth chapter introduces a more suitable problem for real-world applications with some unexpected consequences and the sixth (and last) chapter demonstrates a possible application of the results and methods developed for image inpainting. The last chapter chooses a point of view from which it is clear how the field of sparse signals can borrow ideas and fall under the domain of *Machine Learning*.

Chapter 1

Definition of the Main Problem

A central achievement of classical linear algebra is a thorough examination of the problem of solving systems of linear equations. The results give the subject a completely settled appearance. Surprisingly, within this well-understood arena there are elementary problems that are related to sparse solutions of linear systems, which only recently have been explored in depth. The purpose of this initial chapter is to explain and define those basic problems, what is their purpose and how are they related to sparse solutions.

Sparse solutions of Linear Systems

Consider a matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$, with $n < m$, and define the underdetermined linear system $\mathbf{Ax} = \mathbf{b}$. This system has more unknowns than equations, and thus it has either no solution, if $\mathbf{b}$ is not in the span of the columns of the matrix $\mathbf{A}$, or infinitely many solutions. In order to avoid the anomaly of having no solution, we shall hereafter assume that $\mathbf{A}$ is a **full-rank matrix**, implying that its columns span the entire space $\mathbb{R}^n$.

We now state that our main goal is to find the solution $\mathbf{x}$ which is the **sparsest possible**[1] i.e. which has the least possible number of non-zeros. The most mathematically natural way of doing so is to define the following general optimization problem:

Definition 1.1. *The problem $(\mathcal{P}_J)$ is defined as*

$$(\mathcal{P}_J): \quad \min_{x} \; J(\mathbf{x}) \text{ subject to } \mathbf{Ax} = \mathbf{b}. \tag{1.1}$$

It is now in the hands of the function $J(\mathbf{x})$ to govern and promote sparsity. Perhaps the first choice for $J(\mathbf{x})$ that comes to mind is the squared Euclidean norm $\|\mathbf{x}\|_2^2$. As shown in [9, p.4] the problem, called $(\mathcal{P}_2)$, that results from such a choice has indeed a closed form unique solution $\hat{\mathbf{x}}$.

Knowing that the squared ℓ_2-norm is also strictly convex, which is easily verified through the Hessian, $(\mathcal{P}_2)$ seems the right choice to use but it was shown that it performs quite poorly in real applications [9, p.5] and, as we will show briefly, it is not the best choice if we seek sparsity.

[1] We call a solution sparse if it has $\leq n$ non zero entries.

1.1 The ($\mathcal{P}_1$) Problem

Following the same rationale and notation as above, we introduce the problem ($\mathcal{P}_1$).

Definition 1.2. *The problem ($\mathcal{P}_1$) comes from the general prescription ($\mathcal{P}_J$) for $J(\mathbf{x}) = \|\mathbf{x}\|_1$.*

$$(\mathcal{P}_1): \quad \min_{\mathbf{x}} \ \|\mathbf{x}\|_1 \text{ subject to } \mathbf{A}\mathbf{x} = \mathbf{b}. \tag{1.2}$$

It can be easily shown that the norm $\|\mathbf{x}\|_1$ is convex (but not strictly) therefore the problem ($\mathcal{P}_1$) may have more than one solution. The following theorem shows that the ℓ_1-norm has a tendency to prefer sparse solutions.

Theorem 1.3. *Consider the problem ($\mathcal{P}_1$) defined as above and suppose there are infinitely many solutions satisfying it. It follows that:*
(i) The set of solutions is bounded and convex.
(ii) There exists at least one solution with at most n non-zeros[2].

Proof. (i) (**convexity**) Let us take two different solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ to ($\mathcal{P}_1$) and put α as the minimal ℓ_1-norm penalty; consider its convex combination: $\mathbf{x} = \lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2$, we need to show that $\mathbf{x}$ is also an optimal solution to ($\mathcal{P}_1$).

$$\mathbf{A}\mathbf{x} = \mathbf{A}(\lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2) = \lambda\mathbf{A}\mathbf{x}_1 + (1-\lambda)\mathbf{A}\mathbf{x}_2 = \lambda\mathbf{b} + (1-\lambda)\mathbf{b} = \mathbf{b}.$$

$$\|\mathbf{x}\|_1 = \|\lambda\mathbf{x}_1 + (1-\lambda)\mathbf{x}_2\|_1 \leq \lambda\|\mathbf{x}_1\|_1 + (1-\lambda)\|\mathbf{x}_2\|_1 = \lambda\alpha + (1-\lambda)\alpha = \alpha.$$

In the second line we obtained $\|\mathbf{x}\|_1 \leq \alpha$ which is, in fact, an equality due to the optimality of the given solutions.

(i) (**boundedness**) The fact that the solution set is bounded is (also) a direct consequence of the fact that all optimal solutions give a penalty of the same height: $\|\mathbf{x}\|_1 = \alpha < \infty$ for any optimal solution $\mathbf{x}$. Furthermore given any two optimal solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ we obtain

$$\|\mathbf{x}_1 - \mathbf{x}_2\|_1 \leq \|\mathbf{x}_1\|_1 + \|\mathbf{x}_2\|_1 = 2\alpha, \tag{1.3}$$

implying that all solutions are nearby.

(ii) Let us assume that an optimal solution $\mathbf{x}$ to ($\mathcal{P}_1$) has been found, with $k > n$ non-zeros. Clearly, the k columns (assume w.l.o.g. that these are the first k columns in $\mathbf{A}$) combined linearly by $\mathbf{x}$ are linearly dependent and thus there exist a non-trivial vector $\mathbf{h}$ that combines them to zero (the support of $\mathbf{h}$ is contained within the support of $\mathbf{x}$): $\mathbf{A}\mathbf{h} = \mathbf{0}$.

Consider the vector $\mathbf{x}_\epsilon = \mathbf{x} + \epsilon\mathbf{h}$, for very small values of ϵ such that no entry in the new vector changes its sign. Any value satisfying $|\epsilon| \leq \min_i |x_i|/|h_i|$ is suitable. First, it is clear that this vector satisfies the constraint: $\mathbf{A}\mathbf{x}_\epsilon = \mathbf{A}\mathbf{x} + \epsilon\mathbf{A}\mathbf{h} = \mathbf{b}$ and as such it is a feasible solution to ($\mathcal{P}_1$). Furthermore since $\mathbf{x}$ is assumed to be optimal we have

$$\forall\epsilon : |\epsilon| \leq \min_i \frac{|x_i|}{|h_i|}, \quad \|\mathbf{x}_\epsilon\|_1 = \|\mathbf{x} + \epsilon\mathbf{h}\|_1 \geq \|\mathbf{x}\|_1. \tag{1.4}$$

[2] n is the number of rows/constraints.

We argue that the above inequality is in fact an equality. First of all we know that for ϵ chosen as above it holds $\text{sign}(\mathbf{x}_\epsilon) = \text{sign}(\mathbf{x})$, therefore

$$\begin{aligned}\|\mathbf{x}+\epsilon\mathbf{h}\|_1 &= \sum_{j=1}^{k} \text{sign}(x_j+\epsilon h_j)\left(x_j+\epsilon h_j\right) \\ &= \sum_{j=1}^{k} \text{sign}(x_j)\left(x_j+\epsilon h_j\right) \\ &= \sum_{j=1}^{k} \text{sign}(x_j)x_j + \epsilon\sum_{j=1}^{k} \text{sign}(x_j)h_j \\ &= \|\mathbf{x}\|_1 + \epsilon\mathbf{h}^T\text{sign}(\mathbf{x}).\end{aligned} \tag{1.5}$$

We now show $\mathbf{h}^T\text{sign}(\mathbf{x}) = 0$:

$$\begin{aligned}&\mathbf{h}^T\text{sign}(\mathbf{x}) > 0 \Rightarrow \|\mathbf{x}+\epsilon\mathbf{h}\|_1 < \|\mathbf{x}\|_1 \quad \text{for } \epsilon < 0, \\ &\mathbf{h}^T\text{sign}(\mathbf{x}) < 0 \Rightarrow \|\mathbf{x}+\epsilon\mathbf{h}\|_1 < \|\mathbf{x}\|_1 \quad \text{for } \epsilon > 0.\end{aligned} \tag{1.6}$$

Both inequalities lead to a contradiction with the optimality expressed in (1.4), it follows that $\|\mathbf{x}_\epsilon\|_1 = \|\mathbf{x}\|_1$.

Our next step is to choose ϵ in such a way that one entry in $\mathbf{x}$ is nulled, we choose the index i that gives the minimum ratio $|x_i|/|h_i|$ and we pick $\epsilon = -x_i/h_i$. In the resulting vector $\mathbf{x}_\epsilon = \mathbf{x}+\epsilon\mathbf{h}$, the i-th entry is nulled, while all the others keep their sign.

This way we got a new optimal solution with $k-1$ non-zeros at most (because it may be that more than one entry has been simultaneously nulled). This process can be repeated until $k = n$, below that the linear dependence of the columns in $\mathbf{A}$ is not given[3] and therefore a vector $\mathbf{h}$ does not necessarily exist. □

The property we have proven is well known and considered as the fundamental property of linear programming, a tendency towars basic (sparse) solutions. However, as will be clear in the next chapters, getting n non-zeros would be considered as way too dense for our needs, and deeper sparsity would be sought.

1.2 Promoting Sparse Solutions

As shown in Theorem 1.3 the ℓ_1-norm promotes sparse solutions. Using this rationale it is natural to consider the ℓ_p-"norms" with $p < 1$ for sparsity, however one needs to keep in mind that those functions are no longer convex and therefore no longer norms (the triangle inequality is not satisfied). Nevertheless, in this chapter, we shall use the term norm for these functions as well.

[3] In our analysis we assume $\mathbf{A}$ to be a full-rank matrix, even without this assumption column dependence is rarely the case in practical applications.

Definition 1.4. *The problem ($\mathcal{P}_p$) is defined as*

$$(\mathcal{P}_p): \quad \min_{\boldsymbol{x}} \ \|\mathbf{x}\|_p^p \text{ subject to } \mathbf{A}\mathbf{x} = \mathbf{b}. \tag{1.7}$$

The linear system of equations $\mathbf{A}\mathbf{x} = \mathbf{b}$ forming the constraint defines a feasible set of solutions that are on an affine subspace (a subspace shifted by a constant vector). This shift can be any possible solution of this system $\mathbf{x}_0$, while the subspace is the null-space of $\mathbf{A}$; mathematically: $\mathbf{A}(\mathbf{x}_0 + \alpha\mathbf{x}) = \mathbf{A}\mathbf{x}_0 + \alpha\mathbf{A}\mathbf{x} = \mathbf{b}$ for $\mathbf{x} \in \ker(\mathbf{A})$. Geometrically this set appears as a hyperplane of dimension $\mathbb{R}^{m-n}$ [4] embedded in the $\mathbb{R}^m$ space.

It is within this space that we seek the solution to the problem ($\mathcal{P}_p$). Since we are looking for a vector with minimum ℓ_p-norm (to the p-th power) it is intuitively clear that solving ($\mathcal{P}_p$) is done by "inflating" an ℓ_p [5] ball centered around the origin and stopping its inflation when it first touches the feasible set of solutions.

The question is what are the properties of such an intersection point? Figure 1.1 presents a simple demonstration of this process in 3D, where $m = 3$ and $n = 1$, for a tilted hyperplane (serving as the constraint set $\mathbf{A}\mathbf{x} = \mathbf{b}$) and several p values: 2, 1.6, 1 and 0.7. One can see that norms with $p \leq 1$ tend to give the intersection point on the ball corners, which are on the axes. This implies that 2 of the 3 coordinates are zeros, which is the tendency to sparsity we desire.

Even the ℓ_1 ball enjoys this property, in fact, Figure 1.1 shows that one has to be highly unlucky with the angle of the set of feasible solutions to avoid a sparse outcome. Opposed to this, ℓ_2 and even $\ell_{1.6}$ norms give intersection points which are not sparse (three non-zero coordinates).

Based on this discussion, it would seem very natural to attempt to solve the problem ($\mathcal{P}_p$) for some small p, even tending to zero. Unfortunately each choice $0 < p < 1$ leads to a non-convex optimization problem, and this raises some difficulties. Nevertheless, from an engineering and practical point of view, if sparsity is a desired property, and we know that ℓ_p, serves it well, this problem can and should be tested, despite its difficulties.

The ℓ_0 Norm

In the last section we intuitively discussed about the sparsifying norms, the extreme among those is the case of $p \to 0$.

Definition 1.5. *The ℓ_0-norm is defined as*[6]

$$\|\mathbf{x}\|_0 = \lim_{p\to 0} \|\mathbf{x}\|_p^p = \lim_{p\to 0} \sum_{k=1}^{m} |x_k|^p = \#\{i : x_i \neq 0\}. \tag{1.8}$$

This is a very simple and intuitive measure of sparsity of a vector $\mathbf{x}$, counting the number of nonzero entries in it. Similarly to ℓ_p-norms the ℓ_0-norm is **not**[7] a norm in the

[4] m unknowns and n equations lead to a solution space of dimension $m - n$.

[5] $f(x) = x^p$ for $x \geq 0$, $p > 0$ is a non-decreasing function.

[6] A more accurate notation would be $\|\mathbf{x}\|_0^0$ but we stay consistent with the literature.

[7] Somewhat surprisingly the triangle inequality holds: $\|\mathbf{u} + \mathbf{v}\|_0 \leq \|\mathbf{u}\|_0 + \|\mathbf{v}\|_0$, but the homogeneity property does not: $\|t\mathbf{u}\|_0 = \|\mathbf{u}\|_0 \neq |t|\|\mathbf{u}\|_0$ for $t \neq 0, 1$.

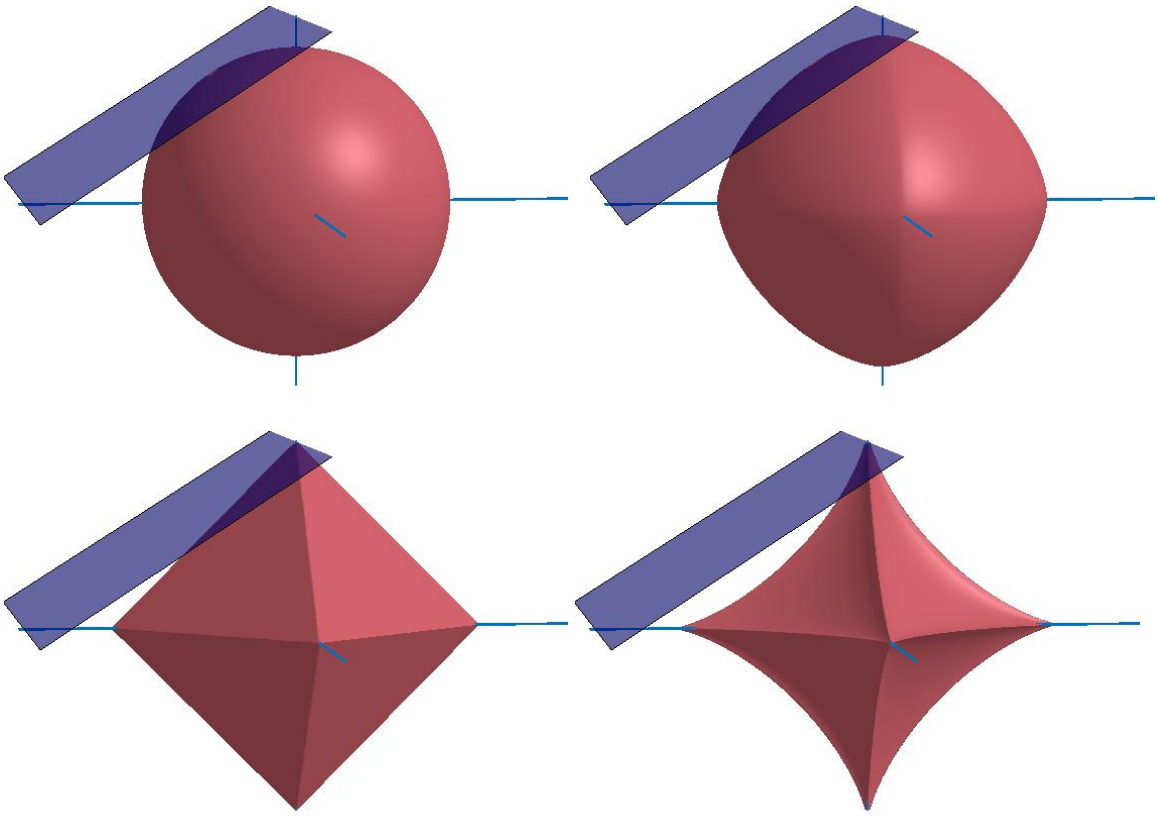

Figure 1.1: The intersection between the ℓ_p-ball and the set $\mathbf{Ax} = \mathbf{b}$ defines the solution of $(\mathcal{P}_p)$. The intersection is demonstrated in 3D for $p = 2$ (top left), $p = 1.6$ (top right), $p = 1$ (bottom left), and $p = 0.7$ (bottom right). When $p \leq 1$ the intersection takes place at a corner of the ball, leading to a sparse solution.

strict sense and we cannot even consider it a continuation of ℓ_p-norms because we would need to take its 0-th root, which is impossible. Alternatively, we can simply refer to the function $\|\mathbf{x}\|_0$ as a candidate function for a norm.

It should be noted that the ℓ_0-norm, while providing a very simple notion of sparsity is not necessarily the right notion for empirical work. A vector of real data would rarely be representable by a vector of coefficients containing many zeros and a more relaxed and forgiving notion of sparsity can and should be built (see [9, p.10-11]). Nevertheless, we proceed the discussion with the assumption that ℓ_0 is the measure of interest[8].

[8]The ℓ_0 norm gives also a sparse solution, by definition, its balls are indeed unions of 1d-lines, 2d-planes, etc. coinciding with the axes, therefore the discussion of Figure 1.1 holds.

1.3 The $(\mathcal{P}_0)$ Problem

It is now time to define the problem $(\mathcal{P}_0)$ which is also obtained from the general prescription $(\mathcal{P}_J)$.

Definition 1.6. *The problem $(\mathcal{P}_0)$ is simply $(\mathcal{P}_J)$ for $J(\mathbf{x}) = \|\mathbf{x}\|_0$*

$$(\mathcal{P}_0): \quad \min_{\boldsymbol{x}} \ \|\mathbf{x}\|_0 \text{ subject to } \mathbf{A}\mathbf{x} = \mathbf{b}. \tag{1.9}$$

Sparsity optimization (1.9) looks superficially like $(\mathcal{P}_1)$ or $(\mathcal{P}_2)$ problems, but there are startling differences. The solution to $(\mathcal{P}_2)$ is always unique[9], and is readily available through now-standard tools from computational linear algebra, similarly $(\mathcal{P}_1)$ can be converted to a linear programming problem[10] and is therefore solvable in reasonable time by available tools. All this is not true for the $(\mathcal{P}_0)$ problem because the underlying norm ℓ_0 is discrete and discontinuous in nature, the standard convex analysis ideas do not apply and many of the most basic questions about $(\mathcal{P}_0)$ seem very difficult to answer:

- Can uniqueness of a solution be claimed? Under what conditions?
- Can a candidate solution be tested to verify its (global) optimality?

Another issue with $(\mathcal{P}_0)$ is the apparent difficulty of solving it. It is a classical problem of combinatorial search; one needs to sweep through all possible sparse subsets, generating corresponding systems $\mathbf{A}_{\mathcal{S}}\mathbf{x}_{\mathcal{S}} = \mathbf{b}$ where $\mathbf{A}_{\mathcal{S}}$ denotes the matrix having $|\mathcal{S}|$ columns chosen from those columns of $\mathbf{A}$ with indices in $\mathcal{S}$; and checking if $\mathbf{A}_{\mathcal{S}}\mathbf{x}_{\mathcal{S}} = \mathbf{b}$ can be solved. The complexity of those exhaustive combinatorial search is exponential in m[11], and indeed, it has been proven that $(\mathcal{P}_0)$ is, in general, NP-Hard. This raises a new set of questions:

- Can $(\mathcal{P}_0)$ be efficiently solved by some other means? What kind of approximations will work? How accurate can those be?

Answers to all the above questions will be given briefly.

[9] The ℓ_2 norm is strictly convex.

[10] See [9, p. 8] or Proposition 3.2.

[11] Example: assume that $\mathbf{A}$ is of size $n = 500$ and $m = 2000$ and suppose that we know that the sparsest solution of $(\mathcal{P}_0)$ has $|\mathcal{S}| = 20$ non-zeros. Thus, we'd need to exhaustively sweep through all $\binom{m}{\mathcal{S}} \approx 3.9 \times 10^{47}$ such options; which is exponential in m by the Stirling's approximation formula and definitely unsolvable.

Chapter 2

Uniqueness and Uncertainty

This chapter addresses some of the questions about $(\mathcal{P}_0)$ we discussed at the end of the last section and some of their extensions. Rather than answering the above questions directly, we first consider special matrices $\mathbf{A}$ for which the analysis is somewhat easier and then extend our answers to the general case.

2.1 The Two-Ortho Case

In this section we will discuss the $(\mathcal{P}_0)$ problem defined in (1.9) in a particular case: where $\mathbf{A}$ is the concatenation of two orthogonal matrices, Ψ and Φ. In other words we are analyzing the underdetermined system $\mathbf{Ax} = \mathbf{b}$ for $\mathbf{A} = [\Psi, \Phi]$.

We proceed with the definition of the mutual-coherence which will have a central role in the following analysis.

Definition 2.1. *The mutual-coherence of a given matrix* $\mathbf{A}$ *is the largest absolute normalized inner product between different columns of* $\mathbf{A}$. *Denoting the k-th column in* $\mathbf{A}$ *by* $\mathbf{a}_k$, *the mutual-coherence is given by*

$$\mu(\mathbf{A}) = \max_{1 \le i,j \le m, i \ne j} \frac{|\mathbf{a}_i^T \mathbf{a}_j|}{\|\mathbf{a}_i\|_2 \cdot \|\mathbf{a}_j\|_2}. \tag{2.1}$$

It is a well known property of orthogonal matrices that their columns represent an orthonormal basis of the given vector space, therefore the mutual-coherence $\mu(\mathbf{A})$ for $\mathbf{A} = [\Psi, \Phi]$ becomes

$$\mu(\mathbf{A}) = \max_{1 \le i,j \le m} |\psi_i^T \phi_j|, \tag{2.2}$$

ψ_i and ϕ_j being respectively the columns of Ψ and Φ.

Example 2.2. (a) Consider the two-ortho matrix $\mathbf{A} = [\mathbf{I}, \mathbf{O}]$ where $\mathbf{O}$ is a rotation matrix by the angle $\theta = \pi/4$.

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \sqrt{2}/2 & -\sqrt{2}/2 \\ 0 & 1 & \sqrt{2}/2 & \sqrt{2}/2 \end{bmatrix}. \tag{2.3}$$

Using (2.2) we calculate $\mu(\mathbf{A}) = 1/\sqrt{2}$.

(b) Consider the two-ortho matrix $\mathbf{B} = [\mathbf{I}, \mathbf{O}]$ where $\mathbf{O}$ is a rotation matrix by the angle $\theta = \pi/180$.

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0.9999 & -0.0175 \\ 0 & 1 & 0.0175 & 0.9999 \end{bmatrix}. \tag{2.4}$$

Using (2.2) we calculate $\mu(\mathbf{B}) = 0.9999 \approx 1$.

(c) Consider the matrix

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0.8 & 1 \\ 0 & 1 & 0.6 & 0 \end{bmatrix}. \tag{2.5}$$

Using the definition we calculate $\mu(\mathbf{C}) = 1$.

We conclude that the mutual-coherence measures *uniformly*[1] the distance of the given matrix from an orthogonal matrix (μ is 0 if the matrix is orthogonal and 1 if there are two equal columns).

Proposition 2.3. *The mutual-coherence $\mu(\mathbf{A})$ for $\mathbf{A} = [\Psi, \Phi]$ satisfies*

$$\frac{1}{\sqrt{n}} \leq \mu(\mathbf{A}) \leq 1. \tag{2.6}$$

Proof. The upper bound holds trivially and it is attained if the matrices Ψ and Φ share a same column (which is normalized by definition). For the lower bound we first have to show that the matrix $\mathbf{B} = \Psi^T \Phi$ is orthogonal

$$\mathbf{B}^T \mathbf{B} = (\Psi^T \Phi)^T (\Psi^T \Phi) = \Phi^T \Psi \Psi^T \Phi = \mathbf{I}. \tag{2.7}$$

A very similar calculation leads to $\mathbf{B}\mathbf{B}^T = \mathbf{I}$. The orthogonality of $\mathbf{B}$ implies that the sum of squares of its entries in each column equals to 1 [2]. All entries cannot therefore be less than $1/\sqrt{n}$ since then we would have that the sum of all squared entries is less than 1. □

We are now ready to demonstrate the so called first uncertainty principle which will bring us to uniqueness and optimality of the solution to $(\mathcal{P}_0)$ for the special case $\mathbf{A} = [\Psi, \Phi]$.

Theorem 2.4. *For an arbitrary pair of orthogonal matrices Ψ and Φ, and $\mathbf{A} = [\Psi, \Phi]$ with mutual-coherence $\mu(\mathbf{A})$ and for an arbitrary non-zero vector $\mathbf{b} \in \mathbb{R}^n$ such that $\mathbf{b} = \Psi\alpha = \Phi\beta$, the following inequality holds true:*

$$\textit{Uncertainty Principle 1:} \quad \|\alpha\|_0 + \|\beta\|_0 \geq \frac{2}{\mu(\mathbf{A})}. \tag{2.8}$$

[1] The max function creates a uniform constraint to the columns and can be ruined by a single pair of columns that are "almost" *parallel*.

[2] The columns are normalized in the ℓ_2 norm.

Proof. We present a very elegant proof from [9, p.21] a more complex and general proof can be found at [11]. Since Ψ and Φ are orthogonal matrices we have that $\|\mathbf{b}\|_2 = \|\alpha\|_2 = \|\beta\|_2$. Denoting the support of α by I, from $\mathbf{b} = \Psi\alpha = \sum_{i\in I}\alpha_i\psi_i$ we calculate

$$\begin{aligned} |\beta_j|^2 = |\mathbf{b}^T\phi_j|^2 &= \left|\sum_{i\in I}\alpha_i\psi_i^T\phi_j\right|^2 \\ &\leq \left(\sum_{i\in I}\alpha_i^2\right)\cdot\left|\sum_{i\in I}(\psi_i^T\phi_j)^2\right| \\ &\leq \|\alpha\|_2^2\cdot\sum_{i\in I}(\psi_i^T\phi_j)^2 \\ &\leq \|\mathbf{b}\|_2^2\cdot|I|\cdot\mu(\mathbf{A})^2. \end{aligned} \tag{2.9}$$

In the third step we have used the Cauchy-Schwartz inequality[3] and in the last the definition of the mutual-coherence. Summing the above over all $j \in J$, J being the support of β, we obtain

$$\sum_{j\in J}|\beta_j|^2 = \|\mathbf{b}\|_2^2 \leq \|\mathbf{b}\|_2^2\cdot|I|\cdot|J|\cdot\mu(\mathbf{A})^2. \tag{2.10}$$

Rewriting the above in a more convenient way and using the fact that $|I| = \|\alpha\|_0$ and $|J| = \|\beta\|_0$ it follows

$$\frac{1}{\mu(\mathbf{A})} \leq \sqrt{\|\alpha\|_0\cdot\|\beta\|_0} \tag{2.11}$$

$$\leq \frac{1}{2}\left(\|\alpha\|_0 + \|\beta\|_0\right), \tag{2.12}$$

where in the last step we used the inequality of arithmetic and geometric means[4]. □

This result suggests that if the mutual-coherence of a matrix $\mathbf{A} = [\Psi, \Phi]$ is small (considering (2.2) we can refer to this as mutual-coherence of two orthonormal bases), then α and β cannot both be very sparse. It is absolutely stunning but what we just showed is indeed somehow related to the Heisenberg's uncertainty principle, more can be found at [9, p.18-21], [7] and related articles.

We now make a connection to the uniqueness problem by introducing the second uncertainty principle.

Theorem 2.5. *Any two distinct solutions* $\mathbf{x}_1$ *and* $\mathbf{x}_2$ *of the linear system* $\mathbf{Ax} = \mathbf{b}$ *for* $\mathbf{b}$ *arbitrary and non-zero and* $\mathbf{A} = [\Psi, \Phi]$ *cannot both be very sparse, indeed they are governed by the following uncertainty principle:*

$$\textit{Uncertainty Principle 2:} \quad \|\mathbf{x}_1\|_0 + \|\mathbf{x}_2\|_0 \geq \frac{2}{\mu(\mathbf{A})}. \tag{2.13}$$

[3] The Cauchy-Schwartz inequality is given by: $|\mathbf{x}^T\mathbf{y}|^2 \leq \|\mathbf{x}\|_2^2\cdot\|\mathbf{y}\|_2^2$.

[4] $\sqrt{xy} \leq \frac{1}{2}(x+y)$ for two nonnegative real numbers x and y.

Proof. We define the vector $\mathbf{e} := \mathbf{x}_1 - \mathbf{x}_2$ which must be in the null-space of $\mathbf{A}$, indeed $\mathbf{Ae} = \mathbf{Ax}_1 - \mathbf{Ax}_2 = \mathbf{b} - \mathbf{b} = \mathbf{0}$. We partition $\mathbf{e}$ into $\mathbf{e}_\Psi$ and $\mathbf{e}_\Phi$ of the first n entries and last n entries, respectively.

Using the known composition of $\mathbf{A}$ it follows that $\mathbf{Ae} = \mathbf{0}$ can be written as $\Psi\mathbf{e}_\Psi + \Phi\mathbf{e}_\Phi = 0$. In other words

$$\Psi\mathbf{e}_\Psi = -\Phi\mathbf{e}_\Phi = \mathbf{y} \neq \mathbf{0}. \tag{2.14}$$

The vector $\mathbf{y}$ is non-zero because $\mathbf{e}$ is non-zero and because Ψ and Φ, being orthogonal matrices, are nonsingular[5]. Invoking (2.8):

$$\|\mathbf{e}\|_0 = \|\mathbf{e}_\Psi\|_0 + \|\mathbf{e}_\Phi\|_0 \geq \frac{2}{\mu(\mathbf{A})}. \tag{2.15}$$

Since $\mathbf{e} = \mathbf{x}_1 - \mathbf{x}_2$ applying the triangle inequality for the ℓ_0 norm to (2.15) we obtain

$$\|\mathbf{x}_1\|_0 + \|\mathbf{x}_2\|_0 \geq \|\mathbf{e}\|_0 \geq \frac{2}{\mu(\mathbf{A})}. \tag{2.16}$$

□

A direct consequence of the *Uncertainty Principle 2* (2.13) is the uniqueness-optimality result:

Corollary 2.6. ***(Uniqueness-Optimality)*** *If a candidate solution of the linear system* $\mathbf{Ax} = \mathbf{b}$ *for* $\mathbf{A} = [\Psi, \Phi]$ *has fewer than* $1/\mu(\mathbf{A})$ *non-zeros, then it is necessarily the sparsest possible, and any other solution must be "denser".*

This seemingly simple claim is wonderful and pretty unexpected, namely we can claim both uniqueness and (global) optimality for sparse enough solutions[6]. Having showed that uniqueness and optimality can be claimed for the two-ortho case it is now time to address general matrices $\mathbf{A}$.

2.2 The General Case

A key property that is crucial for the study of uniqueness in the general case is the *spark* of the matrix $\mathbf{A}$, first defined in 2003 (see [6]).

Definition 2.7. *The spark of a given matrix* $\mathbf{A}$ *is the smallest number of columns from* $\mathbf{A}$ *that are linearly-dependent.*

In other words if $\sigma = Spark(\mathbf{A})$ then there exists a subgroup of σ columns from $\mathbf{A}$ that are linearly dependent.

[5]The null-space of a nonsingular matrix is trivial, $\mathbf{Ax} = \mathbf{0} \Rightarrow \mathbf{x} = \mathbf{0}$.

[6]Usually in non-convex optimization problems a given solution can at best be verified as being locally optimal.

It can be trivially seen that $1 \leq spark(A) \leq n+1$ for a matrix $\mathbf{A}$ of dimensions $n \times m$, $n \leq m$; the *spark* is 1 if the matrix has a zero column and $n+1$ if the matrix has full rank. The *spark* is somewhat opposite to the *rank* of a matrix which is defined as the *largest* number of columns from $\mathbf{A}$ that are linearly *independent*. Nevertheless, the *spark* of a matrix is far more difficult to obtain being NP-Hard itself[7].

Example 2.8. (a) Consider the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 & 2 & 2 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 2 & 5 & 5 \end{bmatrix}. \tag{2.17}$$

Using the definition we calculate $spark(\mathbf{A}) = 3$ (the 5th column is the sum of the 2nd and the 4th). It is clear that there is no shortcut in calculating the *spark*; indeed if we change the 4th column into $\mathbf{a}_4 = [2\ 0\ 4]^T$ then $spark(\mathbf{A}) = 2$ (the 3rd column becomes double the 4th).

(b) Consider the matrix $\mathbf{B} = [\mathbf{I}, \mathbf{O}]$ from Example 2.2 (b).

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0.9999 & -0.0175 \\ 0 & 1 & 0.0175 & 0.9999 \end{bmatrix}. \tag{2.18}$$

No pair of columns are linearly dependent: $spark(\mathbf{B}) = 3$.

(c) Consider the matrix $\mathbf{B} = [\mathbf{I}, \mathbf{O}]$ from Example 2.2 (c).

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0.8 & 1 \\ 0 & 1 & 0.6 & 0 \end{bmatrix}. \tag{2.19}$$

The 1st and the 4th columns are the same, therefore $spark(\mathbf{A}) = 2$.

We conclude that the *spark* (being discrete) is not sensible to almost parallel columns as opposed to the mutual-coherence.

Proposition 2.9. *Every vector from the null-space of the matrix* $\mathbf{A}$*, i.e.* $\mathbf{Ax} = \mathbf{0}$ *satisfies*

$$\|\mathbf{x}\|_0 \geq spark(\mathbf{A}). \tag{2.20}$$

Proof. If $\mathbf{x}$ is a vector from the null-space of $\mathbf{A}$ it linearly combines columns from $\mathbf{A}$ to give the zero vector, by Definition 2.7 at least *spark* such columns are necessary. □

A direct consequence of the above Proposition is the following simple but important Theorem.

Theorem 2.10. ***(Uniqueness-Spark)*** *If a system of linear equations* $\mathbf{Ax} = \mathbf{b}$ *has a solution* $\mathbf{x}$ *obeying* $\|\mathbf{x}\|_0 < spark(\mathbf{A})/2$*, this solution is necessarily the sparsest possible.*

[7] One needs to do a combinatorial search over all possible subsets of columns of $\mathbf{A}$.

Proof. Consider an alternative solution $\mathbf{y}$ that satisfies the same linear system $\mathbf{Ay} = \mathbf{b}$. This implies $\mathbf{A}(\mathbf{x}-\mathbf{y})=\mathbf{0}$, in words the vector $\mathbf{x}-\mathbf{y}$ is in the null-space of $\mathbf{A}$. We obtain

$$\|\mathbf{x}\|_0 + \|\mathbf{y}\|_0 \geq \|\mathbf{x}-\mathbf{y}\|_0 \geq spark(\mathbf{A}). \tag{2.21}$$

The first inequality follows from the triangle inequality of the ℓ_0 norm and the second follows from (2.20). The claim follows trivially. □

It seems that the job is already done and a uniqueness-optimality result, for the general case, has been found. While this is absolutely true from a theoretical point of view, the claim of Theorem 2.10 is unaffordable in a practical situation because computing the *spark* is not less difficult than solving (exactly) $\mathcal{P}_0$ in the first place[8]. We, therefore, continue to look for a computationally feasible uniqueness result.

Before continuing we need to mention the following famous theorem which is key for the next proof (and for the Stability result in chapter 5)

Theorem 2.11. ***(Gershgorin Disk Theorem)*** *Let $\mathbf{A} \in \mathbb{C}^{n\times n}$ and let $\mathcal{G}_i$ be the closed disk in the complex plane centered at A_{ii} with radius given by the row sum $r_i = \sum_{j\neq i} |A_{ij}|$:*

$$\mathcal{G}_i = \{z \in \mathbb{C} : |z - A_{ii}| \leq r_i\} \equiv B(A_{ii}, r_i).$$

Then:

1. *All the eigenvalues of $\mathbf{A}$ lie in the union of the disks $\mathcal{G}_i$, $\mathfrak{S}(\mathbf{A}) \subseteq \bigcup_{i=1}^{n} \mathcal{G}_i$.*
2. *If a union of some k disks is disjoint from the union of the other $n-k$ disks, then exactly k eigenvalues lie in the union of these k disks (algebraic multiplicity is intended).*

Proof. See [19, p.4-9]. □

Proposition 2.12. *For any matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$, the following holds:*

$$spark(\mathbf{A}) \geq 1 + \frac{1}{\mu(\mathbf{A})} \tag{2.22}$$

Proof. First of all, we modify the matrix $\mathbf{A}$ by normalizing its columns to be of unit ℓ_2 norm obtaining $\tilde{\mathbf{A}}$. It can be easily seen that this operation preserves both the *spark* and the mutual-coherence. The entries of the resulting Gram matrix $\mathbf{G} = \tilde{\mathbf{A}}^T\tilde{\mathbf{A}}$ satisfy the following properties:

$$G_{kk} = 1 \text{ for } 1 \leq k \leq m, \tag{2.23}$$
$$|G_{kj}| \leq \mu(\mathbf{A}) \text{ for } 1 \leq k, j \leq m,\ k \neq j. \tag{2.24}$$

[8] Both problems are NP-Hard.

Let $p = spark(\mathbf{A})$ and consider a minor from $\mathbf{G}$ of size $p \times p$, built by choosing a subgroup of p linearly-dependent columns from $\tilde{\mathbf{A}}$ and computing their sub-Gram matrix, call it $\mathbf{G}^{(\mathbf{p})}$.

Assuming the opposite of (2.22) i.e. $p < 1 + 1/\mu$, we calculate:

$$\sum_{j \neq i} |G_{ij}^{(p)}| \leq (p-1)\mu(\mathbf{A}) < 1 = |G_{ii}^{(p)}| \quad \text{for } 1 \leq i \leq m, \tag{2.25}$$

where the first inequality follows from (2.23), the third equality from (2.24) and the second from an algebraic manipulation of the assumption made above.

It is easy to see that $\mathbf{G}^{(\mathbf{p})}$ is a symmetric matrix and therefore all its eigenvalues are real, (2.25) simply means that $\mathbf{G}^{(\mathbf{p})}$ is diagonally dominant and applying the Gershgorin Disk Theorem 2.11 we obtain that it is also *positive-definite*[9]. From [15, p.441] it follows that the p columns of $\mathbf{G}^{(\mathbf{p})}$ are linearly-independent which is clearly a contradiction. □

Theorem 2.13. ***(Uniqueness-Mutual-Coherence)*** *If a system of linear equations* $\mathbf{Ax} = \mathbf{b}$ *has a solution* $\mathbf{x}$ *obeying*

$$\|\mathbf{x}\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{A})}\right), \tag{2.26}$$

then that solution is necessarily the sparsest possible.

Proof. Using (2.22) and the assumption of the theorem we obtain:

$$\|\mathbf{x}\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{A})}\right) \leq \frac{1}{2}\, spark(\mathbf{A}). \tag{2.27}$$

The claim follows directly from Theorem 2.10. □

The last theorem is what we were looking for, a computationally feasible uniqueness-optimality result[10]. However it has to be noted that we obtained this feasibility at the cost of a weaker bound. There are some generalizations of the concept of mutual-coherence which give a tighter bound than Theorem 2.13 (at the cost of higher computational complexity) like the so called Babel function (see [9, p.27, 28]).

[9] From the Gershgorin Disk Theorem it follows that the eigenvalues of $\mathbf{G}^{(\mathbf{p})}$ are strictly positive.

[10] It is clear that $\mu(\mathbf{A})$ has a complexity of $\mathcal{O}(m^2)$, m being the number of columns.

Chapter 3

Pursuit Algorithms

In this chapter we will present reliable and efficient methods for solving (1.9). Let us redefine here ($\mathcal{P}_0$) again for better clarity

$$(\mathcal{P}_0): \quad \min_{\mathbf{x}} \ \|\mathbf{x}\|_0 \text{ subject to } \mathbf{A}\mathbf{x} = \mathbf{b}. \tag{3.1}$$

As discussed at the end of chapter 1 the problem ($\mathcal{P}_0$) is NP-Hard, therefore we need to use some approximate methods. It is straightforward that the unknown $\mathbf{x}$ is composed of two effective parts: the support of the solution and the non-zero values over that support. Thus one way to attack ($\mathcal{P}_0$) is to focus on the support[1] which leads us to the family of greedy algorithms. A second way is to disregard the support and consider the unknown as a vector $\mathbf{x} \in \mathbb{R}^m$ over the continuum and relax the ℓ_0 norm replacing it by a continuous or even smooth approximation of it.

3.1 The Core Greedy Idea

Let us consider a simple case where the matrix $\mathbf{A}$ has $spark(\mathbf{A}) > 2$ and the problem ($\mathcal{P}_0$) has $val(\mathcal{P}_0) = 1$ at the optimal solution, so $\mathbf{b}$ is a scalar multiple of some column of $\mathbf{A}$, and this solution is unique[2]. These columns can be identified by applying m tests, one per column, the j-th test can be done by minimizing $\epsilon(j) = \|\mathbf{a}_j z_j - \mathbf{b}\|_2^2$. We calculate[3]:

$$\begin{aligned} \epsilon(j) &= \langle \mathbf{a}_j z_j - \mathbf{b}, \mathbf{a}_j z_j - \mathbf{b} \rangle \\ &= \langle \mathbf{a}_j z_j, \mathbf{a}_j z_j \rangle - \langle \mathbf{a}_j z_j, \mathbf{b} \rangle - \langle \mathbf{b}, \mathbf{a}_j z_j \rangle + \langle \mathbf{b}, \mathbf{b} \rangle \\ &= z_j^2 \mathbf{a}_j^T \mathbf{a}_j - 2 z_j \mathbf{a}_j^T \mathbf{b} + \mathbf{b}^T \mathbf{b}. \end{aligned} \tag{3.2}$$

[1] Once found the support the non-zero values of $\mathbf{x}$ are easily detected by a plain Least-Squares as will be shown briefly.

[2] Uniqueness follows from Theorem 2.10.

[3] For a simpler notation we write $\epsilon(j)$ instead of the more precise $\epsilon(z_j)$.

Taking the derivative $\frac{\partial}{\partial z_j}$ and setting it to zero, we obtain:

$$2z_j\mathbf{a}_j^T\mathbf{a}_j - 2\mathbf{a}_j^T\mathbf{b} = 0 \tag{3.3}$$

$$\Rightarrow z_j^* = \frac{\mathbf{a}_j^T\mathbf{b}}{\|\mathbf{a}_j\|_2^2}, \tag{3.4}$$

z_j^* being the global minimum because the second derivative is strictly positive. Plugging (3.4) back into the error expression, we obtain:

$$\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{b}\|_2^2 = \left\| \frac{\mathbf{a}_j^T\mathbf{b}}{\|\mathbf{a}_j\|_2^2}\mathbf{a}_j - \mathbf{b} \right\|_2^2 \tag{3.5}$$

$$= \|\mathbf{b}\|_2^2 - 2\frac{(\mathbf{a}_j^T\mathbf{b})^2}{\|\mathbf{a}_j\|_2^2} + \frac{(\mathbf{a}_j^T\mathbf{b})^2}{\|\mathbf{a}_j\|_2^2} \tag{3.6}$$

$$= \|\mathbf{b}\|_2^2 - \frac{(\mathbf{a}_j^T\mathbf{b})^2}{\|\mathbf{a}_j\|_2^2}. \tag{3.7}$$

If this error is zero, we have found the proper solution. Thus, in this special case, the test to be done is basically $\|\mathbf{a}_j\|_2^2 \cdot \|\mathbf{b}\|_2^2 = (\mathbf{a}_j^T\mathbf{b})^2$ (Cauchy-Schwartz inequality satisfied with equality). It is worth noting that we indeed found the **orthogonal projection** of the vector $\mathbf{b}$ into the subspace generated by $\mathbf{a}_j$ and that (3.5)-(3.7) is the **Pythagorean theorem**.

Generalizing the previous calculations one might think to enumerate all $\binom{m}{k_0} = \mathcal{O}(m^{k_0})$ subsets of k_0 columns from $\mathbf{A}$ and test each but this strategy is exponentially complex.

A greedy strategy abandons exhaustive search in favor of a series of locally optimal single-term updates. Starting from $\mathbf{x}^0 = 0$ it iteratively constructs a k-term approximant $\mathbf{x}^k$ by maintaining a set of active columns, initially empty, and expanding it by one additional column at each stage. The column chosen at each stage more or less optimally reduces the residual ℓ_2 error in approximating $\mathbf{b}$ with the currently active columns. After constructing an approximant including the new column, the residual ℓ_2 error is computed; if it falls below a specified threshold the algorithm terminates. The following sections describe four different algorithms which follow this rationale.

3.2 The Greedy Pursuit Algorithms

We now present a series of algorithms which follow the above rationale, their name is known in the literature of signal processing as (Prefix) Matching Pursuit algorithms. We present the Orthogonal Matching Pursuit (OMP) algorithm first.

The OMP Algorithm

Algorithm 3.1: Orthogonal Matching Pursuit

Task: Approximate the solution of $(\mathcal{P}_0)$.
Parameters: Given the matrix $\mathbf{A}$, the vector $\mathbf{b}$ and the error threshold ϵ_0.
Initialization: Initialize $k = 0$ and set

- The initial solution $\mathbf{x}^0 = \mathbf{0}$.
- The initial residual $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0 = \mathbf{b}$.
- The initial solution support $\mathcal{S}^0 = Support\{\mathbf{x}^0\} = \emptyset$.

while $\|\mathbf{r}^k\|_2 > \epsilon_0$ **do**

- **Counter:** Increment k by 1.
- **Sweep:** Compute the errors $\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{r}^{k-1}\|_2^2$ for all j using the optimal choice $z_j^* = \mathbf{a}_j^T \mathbf{r}^{\mathbf{k-1}} / \|\mathbf{a}_j\|_2^2$.
- **Update Support:** Find the minimizer j_0 of $\epsilon(j)$ i.e. $\forall j \ : \ j \notin \mathcal{S}^{k-1} \ \epsilon(j_0) \leq \epsilon(j)$, and update $\mathcal{S}^k = \mathcal{S}^{k-1} \cup \{j_0\}$.
- **Update Provisional Solution:** Compute $\mathbf{x}^k$, the minimizer of $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ subject to $Support\{\mathbf{x}\} = \mathcal{S}^k$.
- **Update Residual:** Compute $\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k$.

end
Output: The proposed solution is $\mathbf{x}^k$ obtained after k iterations.

The first thing that has to be noted is that the *Sweep* stage gives error values of the same form as in (3.5). Calculating as above we obtain

$$\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{r}^{k-1}\|_2^2 = \|\mathbf{r}^{k-1}\|_2^2 - \frac{(\mathbf{a}_j^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_j\|_2^2}. \tag{3.8}$$

Thus the quest for the smallest error is actually equivalent to the quest for the largest (in absolute value) inner product between the residual $\mathbf{r}^{k-1}$ and the normalized columns of the matrix $\mathbf{A}$[4].

In the *Update Provisional Solution* stage we minimize the term $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ with respect to $\mathbf{x}$, such that its support is $\mathcal{S}^k$. We denote $\mathbf{A}_{\mathcal{S}^\mathbf{k}}$ as a matrix of size $n \times |\mathcal{S}^k|$ that contains the columns from $\mathbf{A}$ that belong to this support. Thus, the problem to be solved is a minimization of $\|\mathbf{A}_{\mathcal{S}^k}\mathbf{x}_{\mathcal{S}^k} - \mathbf{b}\|_2^2$, where $\mathbf{x}_{\mathcal{S}^k}$ is the non-zero portion of the vector $\mathbf{x}$. The solution

[4]The function $f(x) = x^2$ is a non-decreasing monotone function.

is, therefore, given by zeroing the derivative of this quadratic form:

$$\mathbf{A}_{\mathcal{S}^k}^T(\mathbf{A}_{\mathcal{S}^k}\mathbf{x}_{\mathcal{S}^k} - \mathbf{b}) = -\mathbf{A}_{\mathcal{S}^k}^T\mathbf{r}^k = 0 \tag{3.9}$$

$$\Rightarrow \quad \mathbf{x}_{\mathcal{S}^k} = \left(\mathbf{A}_{\mathcal{S}^k}^T\mathbf{A}_{\mathcal{S}^k}\right)^{-1}\mathbf{A}_{\mathcal{S}^k}^T\mathbf{b}. \tag{3.10}$$

In the second equality we used the formula of the residual in the *Update Residual* stage and the fact that $\mathbf{A}\mathbf{x}^k = \mathbf{A}_{\mathcal{S}^k}\mathbf{x}_{\mathcal{S}^k}$.

The relation (3.9) simply means that the columns in $\mathbf{A}$ that are part of the support $\mathcal{S}^k$ are necessarily orthogonal to the residual $\mathbf{r}^k$ and this implies that they will never be chosen twice by the OMP algorithm[5]. It is obvious that this orthogonalization is the reason for the name of the algorithm.

The LS-OMP Algorithm

A more complex and somewhat better behaving (less greedy) variant of the above algorithm can be suggested where each of the tests in the sweep stage is done by a full Least-Squares. Indeed if we rewrite the *Sweep* stage using the formula of the residual we obtain

$$\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{r}^{k-1}\|_2^2 = \min_{z_j} \|\mathbf{a}_j z_j + \mathbf{A}_{\mathcal{S}^{k-1}}\mathbf{x}_{\mathcal{S}^{k-1}} - \mathbf{b}\|_2^2. \tag{3.11}$$

It is now straightforward that in the minimization above we are doing a very sub-optimal search because the whole term $\mathbf{A}_{\mathcal{S}^{k-1}}\mathbf{x}_{\mathcal{S}^{k-1}}$ is fixed[6]. The less greedy, Least Squares Orthogonal Matching Pursuit (LS-OMP) approach, is to calculate the minimum w.r.t. z_j and $\mathbf{x}_{\mathcal{S}^{k-1}}$ which values are allowed to change in each iteration. More precisely

$$\epsilon(j) = \min_{z_j, \mathbf{x}_{\mathcal{S}^{k-1}}} \|\mathbf{a}_j z_j + \mathbf{A}_{\mathcal{S}^{k-1}}\mathbf{x}_{\mathcal{S}^{k-1}} - \mathbf{b}\|_2^2. \tag{3.12}$$

A very efficient implementation of the LS-OMP is given at [9, p.38] which uses a recursive approach to considerably speed up the *Sweep* stage of each iteration. The whole strategy of the LS-OMP algorithm is given on the next page.

[5] This also justifies the $j \notin \mathcal{S}^{k-1}$ part in the *Update Support* stage.

[6] It was calculated in the previous iterations.

Algorithm 3.2: Least Squares Orthogonal Matching Pursuit

Task: Approximate the solution of $(\mathcal{P}_0)$.
Parameters: Given the matrix $\mathbf{A}$, the vector $\mathbf{b}$ and the error threshold ϵ_0.
Initialization: Initialize $k = 0$ and set

- The initial solution $\mathbf{x}^0 = \mathbf{0}$.
- The initial residual $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0 = \mathbf{b}$.
- The initial solution support $\mathcal{S}^0 = Support\{\mathbf{x}^0\} = \emptyset$.

while $\|\mathbf{r}^k\|_2 > \epsilon_0$ **do**

- **Counter:** Increment k by 1.
- **Sweep:** Compute the errors $\epsilon(j) = \min_{z_j, \mathbf{x}_{\mathcal{S}^{k-1}}} \|\mathbf{a}_j z_j + \mathbf{A}_{\mathcal{S}^{k-1}}\mathbf{x}_{\mathcal{S}^{k-1}} - \mathbf{b}\|_2^2$ for all j.
- **Update Support:**[a] Find the minimizer j_0 of $\epsilon(j)$ i.e. $\forall j \ : \ j \notin \mathcal{S}^{k-1} \ \epsilon(j_0) \leq \epsilon(j)$, and update $\mathcal{S}^k = \mathcal{S}^{k-1} \cup \{j_0\}$.
- **Update Provisional Solution:** Compute $\mathbf{x}^k$, the minimizer of $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ subject to $Support\{\mathbf{x}\} = \mathcal{S}^k$.
- **Update Residual:** Compute $\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k$.

end
Output: The proposed solution is $\mathbf{x}^k$ obtained after k iterations.

[a] $j \notin \mathcal{S}^{k-1}$ is justified because the errors for $j \in \mathcal{S}^{k-1}$ are always calculated over a smaller set, the minimum of a subset is never lower than the minimum of the whole set.

The MP Algorithm

Another variant of the above OMP algorithm is a more greedy but faster algorithm[7], the so called Matching Pursuit algorithm.

The difference is that there is no orthogonality like in (3.9). After the *Sweep* and the *Update Support* stages, rather than solving a Least-Squares for re-evaluating all the coefficients in $\mathbf{x}$, the coefficients of the $\mathcal{S}^{k-1}$ entries of the last iteration remain unchanged and the new coefficient that refers to the new member $j_0 \in \mathcal{S}^k$ is simply chosen as being $z_{j_0}^*$ [8]. The complete strategy is given on the next page.

[7] But also less accurate.
[8] $z_{j_0}^*$ is the minimizer of $\epsilon(j)$ and is defined as in the *Sweep* stage in Algorithm 3.1 (OMP).

Algorithm 3.3: Matching Pursuit

Task: Approximate the solution of $(\mathcal{P}_0)$.
Parameters: Given the matrix $\mathbf{A}$, the vector $\mathbf{b}$ and the error threshold ϵ_0.
Initialization: Initialize $k = 0$ and set

- The initial solution $\mathbf{x}^0 = \mathbf{0}$.
- The initial residual $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0 = \mathbf{b}$.
- The initial solution support $\mathcal{S}^0 = Support\{\mathbf{x}^0\} = \emptyset$.

while $\|\mathbf{r}^k\|_2 > \epsilon_0$ **do**

- **Counter:** Increment k by 1.
- **Sweep:** Compute the errors $\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{r}^{k-1}\|_2^2$ for all j using the optimal choice $z_j^* = \mathbf{a}_j^T \mathbf{r}^{\mathbf{k-1}} / \|\mathbf{a}_j\|_2^2$.
- **Update Support:** Find the minimizer j_0 of $\epsilon(j)$: $\forall 1 \le j \le m^a$, $\epsilon(j_0) \le \epsilon(j)$, and update $\mathcal{S}^k = \mathcal{S}^{k-1} \cup \{j_0\}$.
- **Update Provisional Solution:** Set $\mathbf{x}^k = \mathbf{x}^{k-1}$ and update the entry $x^k(j_0) = x^{k-1}(j_0) + z_{j_0}^*$.
- **Update Residual**[b]: Compute $\mathbf{r}^k = \mathbf{r}^{k-1} - z_{j_0}^* \mathbf{a}_{j_0}$.

end
Output: The proposed solution is $\mathbf{x}^k$ obtained after k iterations.

[a] Because of no orthogonality MP does not guarantee that a column cannot be chosen twice, indeed it can happen.

[b] $\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^{k-1} - z_{j_0}^* \mathbf{a}_{j_0} = \mathbf{r}^{k-1} - z_{j_0}^* \mathbf{a}_{j_0}$ where in the second equality we used the *Update Provisional Solution* stage and in the third the *Update Residual* stage of the $k-1$ iteration.

The Weak-MP Algorithm

The Weak Matching Pursuit algorithm is a further simplification of the MP algorithm, it allows for a sub-optimal choice of the next element to be added to the support. The *Update Support* stage is relaxed by choosing any index that is a factor $t \in [0,1]$ away from the optimal choice.

More precisely, rather than searching, as in (3.8), for the largest inner-product $|\mathbf{a}_j^T \mathbf{r}^{k-1}|$ value (up to a normalization factor and a non-decreasing monotone function), we settle for the first column that exceeds a t-weaker threshold. The Cauchy-Schwartz inequality gives

$$\frac{(\mathbf{a}_j^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_j\|_2^2} \le \max_{1 \le j \le m} \frac{(\mathbf{a}_j^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_j\|_2^2} \le \|\mathbf{r}^{k-1}\|_2^2. \tag{3.13}$$

This puts an upper bound on the maximal achievable inner-product. Thus, we can compute $\|\mathbf{r}^{k-1}\|_2^2$ at the beginning of the *Sweep* stage and, as we search for j_0 that gives the smallest error $\epsilon(j)$, we choose the first that gives

$$\frac{(\mathbf{a}_{j_0}^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_{j_0}\|_2^2} \geq t^2 \cdot \|\mathbf{r}^{k-1}\|_2^2 \geq t^2 \cdot \max_{1 \leq j \leq m} \frac{(\mathbf{a}_j^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_j\|_2^2}. \tag{3.14}$$

Note that it is possible that the complete sweep is performed without any index satisfying the above condition, then we simply choose the maximum that was found as a by-product of the search.

The Thresholding Algorithm

The simplest of all algorithms is the so called Thresholding algorithm which is also slightly different in the way greediness is practiced.

It is basically a simplificaton of the OMP algorithm, the key difference is that the decision made about the support of the solution is based on the first projection alone, i.e. it chooses the k largest inner products $|\mathbf{a}_j^T \mathbf{b}|$ (see (3.5)-(3.7)), as the desired support[9]. That is done in the *Quality Evaluation* stage by a simple sort of the entries of the vector $|\mathbf{A}^T \mathbf{b}|$.

Another important thing to notice is that we assume that k, the number of required non-zeros, is known. Alternatively a simple modification can be done by simply increasing k until the error of $\|\mathbf{Ax} - \mathbf{b}\|_2$ reaches a pre-specified value ϵ_0.

The complete strategies of the last two algorithms are given on the next page.

[9]There is no normalization term because of the following Theorem 3.1.

Algorithm 3.4: Weak Matching Pursuit

Task: Approximate the solution of $(\mathcal{P}_0)$.

Parameters: Given the matrix $\mathbf{A}$, the vector $\mathbf{b}$, the error threshold ϵ_0 and the scalar $0 < t < 1$.

Initialization: Initialize $k = 0$ and set

- The initial solution $\mathbf{x}^0 = \mathbf{0}$.
- The initial residual $\mathbf{r}^0 = \mathbf{b} - \mathbf{A}\mathbf{x}^0 = \mathbf{b}$.
- The initial solution support $\mathcal{S}^0 = Support\{\mathbf{x}^0\} = \emptyset$.

while $\|\mathbf{r}^k\|_2 > \epsilon_0$ **do**

- **Counter:** Increment k by 1.
- **Sweep:** Compute the errors $\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{r}^{k-1}\|_2^2$ for all j using the optimal choice $z_j^* = \mathbf{a}_j^T \mathbf{r}^{\mathbf{k-1}} / \|\mathbf{a}_j\|_2^2$. Stop the sweep when $|\mathbf{a}_j^T \mathbf{r}^{k-1}| / \|\mathbf{a}_j\|_2 \geq t \cdot \|\mathbf{r}^{k-1}\|_2$.[a]
- **Update Support:** Update $\mathcal{S}^k = \mathcal{S}^{k-1} \cup \{j_0\}$, with j_0 found in the sweep stage.
- **Update Provisional Solution:** Set $\mathbf{x}^k = \mathbf{x}^{k-1}$ and update the entry $x^k(j_0) = x^{k-1}(j_0) + z_{j_0}^*$.
- **Update Residual**: Compute $\mathbf{r}^k = \mathbf{r}^{k-1} - z_{j_0}^* \mathbf{a}_{j_0}$.

end

Output: The proposed solution is $\mathbf{x}^k$ obtained after k iterations.

[a] This is simply the square root of the condition in equation (3.14).

Algorithm 3.5: Thresholding

Task: Approximate the solution of $(\mathcal{P}_0)$.

Parameters: Given the matrix $\mathbf{A}$, the vector $\mathbf{b}$ and the number of atoms desired k.

Quality Evaluation: Compute the errors $\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{b}\|_2^2$ for all j using the optimal choice $z_j^* = \mathbf{a}_j^T \mathbf{b} / \|\mathbf{a}_j\|_2^2$.

Update Support: Find the set of indices $\mathcal{S}^k$ of cardinality k that contains the smallest errors: $\forall j \in \mathcal{S}^k, \epsilon(j) \leq \min_{i \notin \mathcal{S}} \epsilon(i)$.

Update Provisional Solution: Compute $\mathbf{x}^k$, the minimizer of $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ subject to $Support\{\mathbf{x}\} = \mathcal{S}^k$.

Output: The proposed solution is $\mathbf{x}^k$.

The following technical result is of great importance because it is simpler and computationally more feasible to work with a matrix $\mathbf{A}$ which has ℓ_2 normalized columns. We can normalize the columns by the operation $\tilde{\mathbf{A}} = \mathbf{AW}$, where $\mathbf{W}$ is a diagonal matrix containing $1/\|\mathbf{a}_i\|_2$ on the main diagonal.

Theorem 3.1. *The greedy algorithms (OMP,MP and weak-MP)*[10] *produce the same solution support* $\mathcal{S}^k$ *when using either the original matrix* $\mathbf{A}$ *or its normalized version* $\tilde{\mathbf{A}}$.

Proof. See [9, p.41-43]. □

We shall assume hereafter, when dealing with these algorithms, that the matrix $\mathbf{A}$ is already normalized. In case we normalized a previously non-normalized matrix $\mathbf{A}$ and used on it any of those greedy algorithms a de-normalization step has to be taken. Suppose that $\tilde{\mathbf{A}} = \mathbf{AW}$ and that the algorithm provides a solution $\tilde{\mathbf{x}}$ that satisfies $\tilde{\mathbf{A}}\tilde{\mathbf{x}}^k = \mathbf{b}$ we obtain

$$\mathbf{b} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}^k = \mathbf{AW}\tilde{\mathbf{x}}^k = \mathbf{A}\mathbf{x}^k \quad \Rightarrow \quad \mathbf{x}^k = \mathbf{W}\tilde{\mathbf{x}}^k. \tag{3.15}$$

3.3 Convex Relaxation Algorithms

As mentioned at the beginning of the chapter a second way to render $(\mathcal{P}_0)$ more tractable is to relax the (highly discontinuous) ℓ_0 norm, replacing it by a continuous or even smooth approximation.

There are many ways to do so the most popular options include replacing it with ℓ_p norms for some $p \in (0,1]$ or even by smooth functions such as $\sum_j \log(1+\alpha x_j^2)$, $\sum_j x_j^2/(\alpha + x_j^2)$ or $\sum_j (1-\exp(-\alpha x_j^2))$, another very efficient algorithm is the so called FOCal Undetermined System Solver (FOCUSS), for more details see [14] and [9, p.48-50].

In this section we will discuss in details the ℓ_1 norm relaxation also called Basis Pursuit (BP) which has a very elegant solution and strong theoretical foundations.

The Basis Pursuit

The ℓ_1 norm relaxation is done by simply replacing the ℓ_0 norm with the ℓ_1 norm which is straightforward. We therefore treat the $(\mathcal{P}_0)$ problem as a $(\mathcal{P}_1)$ problem which we define again for clarity.

$$(\mathcal{P}_1): \quad \min_{\mathbf{x}} \ \|\mathbf{x}\|_1 \text{ subject to } \mathbf{Ax} = \mathbf{b}. \tag{3.16}$$

It is important to note that while the ℓ_0 norm is indifferent to the magnitude of the non-zero entries in $\mathbf{x}$ the ℓ_1 norm is not and tends to penalize higher magnitudes i.e. it biases the solution towards choosing non-zero entries in locations of $\mathbf{x}$ that multiply large norm columns in $\mathbf{A}$ (see [9, p.50]. Therefore, in order to avoid any kind of bias, the columns of $\mathbf{A}$ should be scaled appropriately.

[10] Following the proof given at [9] it is clear that the result holds for the LS-OMP and Thresholding algorithms too.

Identically as above we normalize the matrix $\mathbf{A}$ by multiplying it by a matrix $\mathbf{W}$ which is a diagonal positive-definite matrix whose diagonal elements are $w(i,i) = 1/\|\mathbf{a}_i\|_2$. Considering the above and putting $\tilde{\mathbf{x}} = \mathbf{W}^{-1}\mathbf{x}$ we transform the problem $(\mathcal{P}_1)$ into

$$(\tilde{\mathcal{P}}_1): \quad \min_{\tilde{\mathbf{x}}} \ \|\tilde{\mathbf{x}}\|_1 \text{ subject to } \tilde{\mathbf{A}}\tilde{\mathbf{x}} = \mathbf{A}\mathbf{W}\tilde{\mathbf{x}} = \mathbf{b}. \tag{3.17}$$

Just as for greedy methods, once a solution $\tilde{\mathbf{x}}$ has been found, it should be de-normalized[11] to provide the required vector $\mathbf{x}$. The problem described[12], and the algorithms implemented to solve it, is called Basis Pursuit.

It was already mentioned that the $(\mathcal{P}_1)$ problem can be cast as a linear programming (LP) problem, for its importance we report this result here.

Proposition 3.2. *Consider $(\mathcal{P}_1)$ and put $\mathbf{x} = \mathbf{u} - \mathbf{v}$ where $\mathbf{u}, \mathbf{v} \in R^m$ where $\mathbf{u}$ takes all the positive entries in $\mathbf{x}$, with all other entries null, and $\mathbf{v}$ does the same for the negative ones. Putting $\mathbf{z} = [\mathbf{u}^T, \mathbf{v}^T]^T \in \mathbb{R}^{2m}$ the $(\mathcal{P}_1)$ optimization problem is equivalent to the following Linear-Programming (LP) problem of classical structure*

$$\min_{\mathbf{z}} \ \mathbf{1}^T\mathbf{z} \text{ subject to } \mathbf{b} = [\mathbf{A}, -\mathbf{A}]\mathbf{z}\,,\ \mathbf{z} \geq \mathbf{0}. \tag{3.18}$$

Proof. ($\Rightarrow$) It is easy to see that[13]:

$$\|\mathbf{x}\|_1 = \mathbf{1}^T(\mathbf{u}+\mathbf{v}) = \mathbf{1}^T\mathbf{z} \text{ and } \mathbf{A}\mathbf{x} = \mathbf{A}(\mathbf{u}-\mathbf{v}) = [\mathbf{A}, -\mathbf{A}]\mathbf{z}. \tag{3.19}$$

($\Leftarrow$) We must show that the decomposition of $\mathbf{x}$ to positive and negative entries is satisfied, i.e. that solutions to (3.18) cannot be such that the supports of $\mathbf{u}$ and $\mathbf{v}$ overlap.

This is easily proved by contradiction. Suppose that in a given optimal solution of (3.18) the k-th entry in both $\mathbf{u}$ and $\mathbf{v}$ is non-zero (and positive, due to the last constraint) then these two coefficients multiply the same columns in $\mathbf{A}$ with opposing signs. Without loss of generality if we assume $u_k > v_k$, then by replacing these two entries by $u'_k = u_k - v_k$ and $v'_k = 0$ we satisfy both the positivity and the linear constraints of (3.18) while also reducing the penalty by

$$(u_k + v_k) - (u'_k + v'_k) = u_k + v_k - u_k + v_k = 2v_k > 0, \tag{3.20}$$

which is a contradiction to the optimality of the initial solution. □

The proposition above shows that standard techniques for linear programming problems apply to $(\mathcal{P}_1)$ and therefore to Basis Pursuit, this techniques include modern interior-point methods, simplex methods, homotopy methods and others.

Such algorithms are usually far more sophisticated than the greedy algorithms mentioned earlier, as they obtain the global solution of a well-defined optimization problem,

[11] $\mathbf{x} = \mathbf{W}\tilde{\mathbf{x}}$.

[12] The term Basis Pursuit was originally used only for the case $\mathbf{W} = \mathbf{I}$ but each case can be reduced to that as shown.

[13] The notation $\mathbf{1}$ stands for a vector of ones of the proper length.

but this also means that programming these techniques is far more complicated. As one would expect there are several well-developed software packages that already exist and handle this problem which are free and shared on the web, the ℓ_1-magic, the CVX or Sparselab just to mention a few (see [9, p.51] for more).

Chapter 4

Theoretical Guarantees of Pursuit Algorithms

In this chapter we discuss some theoretical guarantees for optimal solution recovery of the pursuit algorithms presented before, clearly this guarantees will not be as tight as Theorem 2.10 or 2.13 since this would conflict with the known NP-hardness of the problem in the general case. However, if the system $\mathbf{Ax} = \mathbf{b}$ has a *sufficiently sparse* solution, success can be guaranteed. Before we embark to this discussion we shall demonstrate another important result, the rate of decay of the residual in greedy methods.

4.1 Residual Decay in Greedy Methods

In this subsection we show an important result which provides another reason why greedy methods described above are likely to get a sparse solution to the $(\mathcal{P}_0)$ problem. We start with the definition of the decay-factor of a matrix, inspired by the idea at [16], which will be helpful in further analysis.

Definition 4.1. *For the matrix* $\mathbf{A}$ *with m normalized columns, the universal decay-factor* $\delta(\mathbf{A})$ *is defined by*

$$\delta(\mathbf{A}) = \inf_{\mathbf{v}\neq\mathbf{0}} \frac{\left\|\mathbf{A}^T\mathbf{v}\right\|_\infty^2}{\|\mathbf{v}\|_2^2}. \tag{4.1}$$

The following theorem is the key result.

Theorem 4.2. *The worst decay of the MP, OMP and LS-OMP residual is exponential, with a rate given by*

$$\|\mathbf{r}^k\|_2^2 \leq (1-\delta(\mathbf{A}))^k \cdot \|\mathbf{b}\|_2^2. \tag{4.2}$$

Proof. Consider Algorithm 3.3 (MP) and keep in mind the normalization of the matrix $\mathbf{A}$, as a consequence of Theorem 3.1, the residual is therefore updated by the recursive formula

$$\mathbf{r}^k = \mathbf{r}^{k-1} - z_{j_0}^* \mathbf{a}_{j_0} = \mathbf{r}^{k-1} - (\mathbf{a}_{j_0}^T \mathbf{r}^{k-1})\mathbf{a}_{j_0}. \tag{4.3}$$

Taking the norm of the above expression and squaring it we calculate

$$\begin{aligned}\|\mathbf{r}^k\|_2^2 &= \left\|\mathbf{r}^{k-1} - (\mathbf{a}_{j_0}^T\mathbf{r}^{k-1})\mathbf{a}_{j_0}\right\|_2^2 \\ &= \|\mathbf{r}^{k-1}\|_2^2 - 2(\mathbf{a}_{j_0}^T\mathbf{r}^{k-1})^2 + (\mathbf{a}_{j_0}^T\mathbf{r}^{k-1})^2 \\ &= \|\mathbf{r}^{k-1}\|_2^2 - (\mathbf{a}_{j_0}^T\mathbf{r}^{k-1})^2 \\ &= \|\mathbf{r}^{k-1}\|_2^2 - \max_{1\le j\le m}(\mathbf{a}_{j_0}^T\mathbf{r}^{k-1})^2 \\ &= \|\mathbf{r}^{k-1}\|_2^2 - \left\|\mathbf{A}^T\mathbf{r}^{k-1}\right\|_\infty^2.\end{aligned} \tag{4.4}$$

In the first three equations we simply used well known properties of the inner product and the ℓ_2 norm, in the fourth the fact that j_0 is the optimal choice and in the last the definition of the ℓ_∞ norm.

Continuing from the last step of the previous calculation and applying the decay-factor we calculate

$$\begin{aligned}\|\mathbf{r}^k\|_2^2 &= \|\mathbf{r}^{k-1}\|_2^2 - \left\|\mathbf{A}^T\mathbf{r}^{k-1}\right\|_\infty^2 \\ &= \|\mathbf{r}^{k-1}\|_2^2 - \|\mathbf{r}^{k-1}\|_2^2 \cdot \frac{\left\|\mathbf{A}^T\mathbf{r}^{k-1}\right\|_\infty^2}{\|\mathbf{r}^{k-1}\|_2^2} \\ &\le \|\mathbf{r}^{k-1}\|_2^2 - \|\mathbf{r}^{k-1}\|_2^2 \cdot \delta(\mathbf{A}) \\ &= (1-\delta(\mathbf{A}))\cdot\|\mathbf{r}^{k-1}\|_2^2.\end{aligned} \tag{4.5}$$

Applying the last formula recursively leads to

$$\|\mathbf{r}^k\|_2^2 \le (1-\delta(\mathbf{A}))^k \cdot \|\mathbf{r}^0\|_2^2 \tag{4.6}$$

$$= (1-\delta(\mathbf{A}))^k \cdot \|\mathbf{b}\|_2^2. \tag{4.7}$$

Which establishes clearly an *exponential* rate of decay. It is important to notice that we immediately obtain the same bound for the OMP and the LS-OMP since these two algorithms have a more sophisticated *Sweep* stage and thus they reduce the energy of the residual at each stage even more[1]. □

Similarly applying (3.14) to the previous calculations we obtain an analogous result for the Weak-MP.

Theorem 4.3. *The worst decay of the Weak-MP residual is exponential, with a rate given by*

$$\|\mathbf{r}^k\|_2^2 \le (1 - t\cdot\delta(\mathbf{A}))^k \cdot \|\mathbf{b}\|_2^2. \tag{4.8}$$

One last thing we need to verify to show convergence for all the above is that $\delta(\mathbf{A})$ is strictly positive and not zero. This result is given by the following Lemma.

[1]The following holds: $\|\mathbf{r}_{LS-OMP}^k\|_2^2 \le \|\mathbf{r}_{OMP}^k\|_2^2 \le \|\mathbf{r}_{MP}^k\|_2^2$, the subscripts simply indicate from which algorithm the k-th residual comes from.

Lemma 4.4. *The decay rate $\delta(\mathbf{A})$ of every full-rank matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ (with $n < m$) is strictly positive ($\delta(\mathbf{A}) > 0$).*

Proof. From the given assumptions for the matrix $\mathbf{A}$ it follows that its columns span the entire $\mathbb{R}^n$ space, thus no vector $\mathbf{v}$ could be orthogonal to all these columns and that implies $\delta(\mathbf{A}) > 0$. □

4.2 Performance for the Two-Ortho Case

In this section we discuss the performance guarantees of the Orthogonal Matching Pursuit (OMP) and the Basis Pursuit (BP), similar results are valid for other algorithms but we omit them since this section refers to a particular case presented mainly for completeness. Proofs of these results are not explicitly written for the same reason given above.

Theorem 4.5. ***(Performance Guarantee-OMP-Two Ortho Case)**: Given a system of linear equations $\mathbf{Ax} = [\Psi, \Phi]\mathbf{x} = \mathbf{b}$ with two orthogonal matrices Ψ and Φ of size $n \times n$, if a (unique[2]) solution $\mathbf{x}$ exists such that it has k_p non-zeros in its first half, k_q in the second [3] and the two obey*

$$\max(k_p, k_q) < \frac{1}{2\mu(\mathbf{A})}, \tag{4.9}$$

*then OMP, run with threshold parameter $\epsilon_0 = 0$, is guaranteed to find it **exactly** in $k_0 = k_p + k_q$ steps.*

Proof. See [9, p.55-57]. □

Theorem 4.6. ***(Performance Guarantee-BP-Two Ortho Case)**: Given a system of linear equations $\mathbf{Ax} = [\Psi, \Phi]\mathbf{x} = \mathbf{b}$ with two orthogonal matrices Ψ and Φ of size $n \times n$, if a (unique) solution $\mathbf{x}$ exists such that it has k_p non-zeros in its first half, $k_q \leq k_p$ in the second and the two obey*

$$2\mu(\mathbf{A})^2 k_p k_q + \mu(\mathbf{A}) k_p - 1 < 0, \tag{4.10}$$

*then BP is guaranteed to find it **exactly**, i.e. that solution is both the unique solution of ($\mathcal{P}_1$) and ($\mathcal{P}_0$).*

Proof. See [9, p.58-64]. □

Since the above condition may seem obscure, we provide a weaker but simpler version to interpret it:

$$\|\mathbf{x}\|_0 = k_p + k_q < \frac{\sqrt{2} - 0.5}{\mu(\mathbf{A})}. \tag{4.11}$$

Figure 4.1 presents a comparison between the performance bounds for the OMP and BP algorithms together with the uniqueness result of Theorem 2.6. It is clear that OMP is

[2] In each of these Performance Guarantee Theorems the bounds are always tighter than the one in Corollary 2.6 therefore the solutions presented are always unique.

[3] $\mathbf{x}$ can be partitioned into two parts, $\mathbf{x}_\Psi$ and $\mathbf{x}_\Phi$, each of them is labelled with the matrix it multiplies.

found to be weaker compared to the BP in handling the $(\mathcal{P}_0)$ problem, BP is even not so far away from the Uniqueness bound which is the best possible bound. However it has to be said that the tightness of BP was established by Feuer and Nemirovsky [13] while the tightness of OMP remains questionable[4].

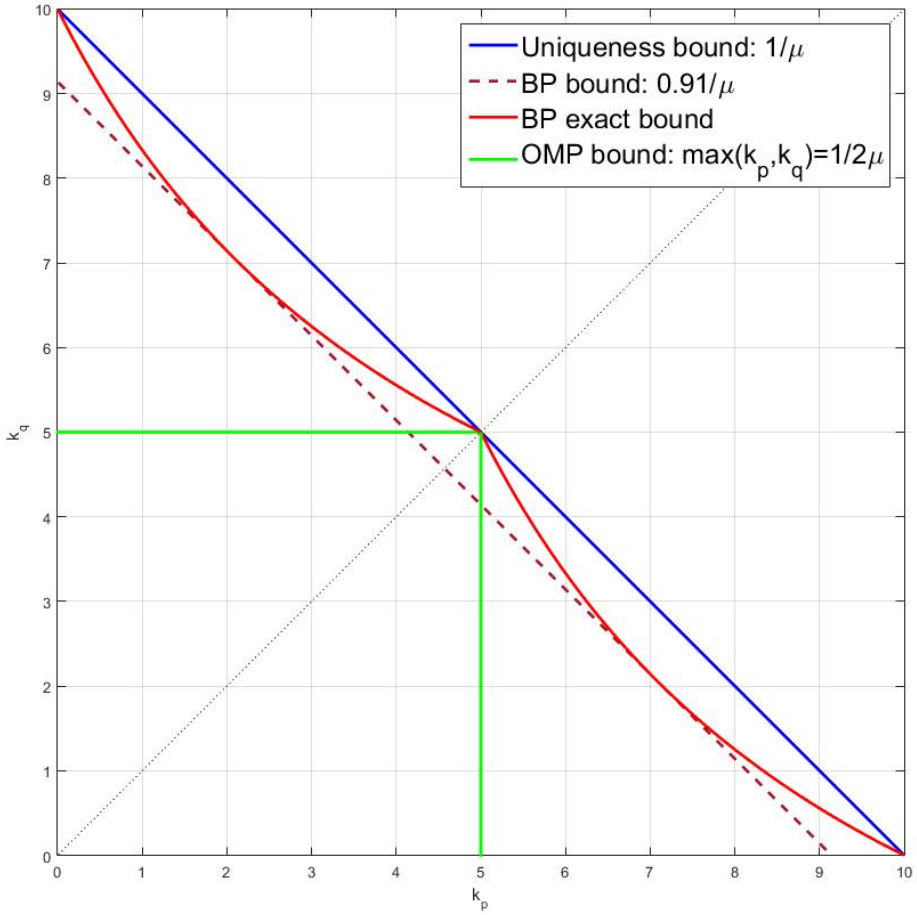

Figure 4.1: Uniqueness, Corollary 2.6, and performance bounds for the OMP, Equation (4.9), and BP, Equations (4.10) and (4.11), for the two-ortho case, assuming $\mu(\mathbf{A}) = 0.1$.

4.3 Performance for the General Case

We now turn to handle the general case with an arbitrary matrix $\mathbf{A}$. As already shown before we can assume, without loss of generality, that the columns of this matrix are normalized.

[4]Until this result remains unproven a better bound for OMP is still possible.

The following result generalizes the one in Theorem 4.5 for general matrices and therefore it is necessarily weaker.

Theorem 4.7. *(**Performance Guarantee - OMP - General Case**): Given a system of linear equations* $\mathbf{Ax} = \mathbf{b}$ *(*$\mathbf{A} \in \mathbb{R}^{n \times m}$ *full-rank with* $n < m$*), if a (unique*[5]*) solution* $\mathbf{x}$ *exists obeying*

$$\|\mathbf{x}\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{A})}\right), \tag{4.12}$$

then OMP, run with threshold parameter $\epsilon_0 = 0$*, is guaranteed to find it **exactly**.*

Proof. Suppose, without loss of generality, that the (unique) sparsest solution of the linear system is such that all its k_0 non-zero entries are at the beginning of the vector in decreasing order i.e. $|x_1| \geq |x_2| \geq ... \geq |x_{k_0}|$. Thus,

$$\mathbf{b} = \mathbf{Ax} = \sum_{t=1}^{k_0} x_t \mathbf{a}_t. \tag{4.13}$$

At the first iteration ($k = 0$) of the algorithm $\mathbf{r}^k = \mathbf{r}^0 = \mathbf{b}$ and, using Equation (3.8), the set of computed errors from the *Sweep* stage is given by

$$\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{b}\|_2^2 = \|\mathbf{b}\|_2^2 - (\mathbf{a}_j^T \mathbf{b})^2 \geq 0. \tag{4.14}$$

For that reason, for the first iteration to choose the first of the k_0 entries in the vector and do well, we must require that for all $i > k_0$ (columns outside the true support)

$$|\mathbf{a}_1^T \mathbf{b}| > |\mathbf{a}_i^T \mathbf{b}|. \tag{4.15}$$

Substituting $\mathbf{b}$ as defined in Equation (4.13) we obtain

$$\left|\sum_{t=1}^{k_0} x_t \mathbf{a}_1^T \mathbf{a}_t\right| > \left|\sum_{t=1}^{k_0} x_t \mathbf{a}_i^T \mathbf{a}_t\right|. \tag{4.16}$$

The idea is to construct a lower bound for the left side, an upper bound for the right side and then pose the above inequality again[6]. For the left side we obtain

$$\begin{aligned}
\left|\sum_{t=1}^{k_0} x_t \mathbf{a}_1^T \mathbf{a}_t\right| &= \left|x_1 + \sum_{t=2}^{k_0} x_t \mathbf{a}_1^T \mathbf{a}_t\right| \\
&\geq |x_1| - \sum_{t=2}^{k_0} |x_t| \cdot \left|\mathbf{a}_1^T \mathbf{a}_t\right| \\
&\geq |x_1| - \sum_{t=2}^{k_0} |x_t| \cdot \mu(\mathbf{A}) \\
&\geq |x_1| - |x_1|(k_0 - 1)\mu(\mathbf{A}) \\
&\geq |x_1| \cdot (1 - \mu(\mathbf{A})(k_0 - 1)).
\end{aligned} \tag{4.17}$$

[5]Theorem 2.13 guarantees uniqueness.

[6]**LS** $\geq$ **Lbound** $>$ **Ubound** $\geq$ **RS**.

In the first inequality we used $|a+b| \geq |a| - |b|$, in the third Definition 2.1 (mutual-coherence) and the decreasing order of the values $|x_j|$. The right side is bounded by

$$\begin{aligned} \left| \sum_{t=1}^{k_0} x_t \mathbf{a}_i^T \mathbf{a}_t \right| &\leq \sum_{t=1}^{k_0} |x_t| \cdot \left| \mathbf{a}_i^T \mathbf{a}_t \right| \\ &\leq \sum_{t=1}^{k_0} |x_t| \cdot \mu(\mathbf{A}) \\ &\leq |x_1| \cdot \mu(\mathbf{A}) k_0, \end{aligned} \tag{4.18}$$

where, again, we exploited similar properties as above. Using these two bounds and plugging them into (4.16) we obtain

$$\begin{aligned} \left| \sum_{t=1}^{k_0} x_t \mathbf{a}_1^T \mathbf{a}_t \right| &\geq |x_1| \cdot (1 - \mu(\mathbf{A})(k_0 - 1)) \\ &> |x_1| \cdot \mu(\mathbf{A}) k_0 \\ &\geq \left| \sum_{t=1}^{k_0} x_t \mathbf{a}_i^T \mathbf{a}_t \right|. \end{aligned} \tag{4.19}$$

The second inequality leads to

$$1 + \mu(\mathbf{A}) > 2\mu(\mathbf{A}) k_0 \quad \Leftrightarrow \quad \|\mathbf{x}\|_0 = k_0 < \frac{1}{2} \left(1 + \frac{1}{\mu(\mathbf{A})} \right), \tag{4.20}$$

which is the condition we are looking for. However, so far we have showed that the condition is sufficient only for the *first* iteration i.e. the column of $\mathbf{A}$ related to the entry of $\mathbf{x}$ having the highest magnitude will be chosen.

The next step is an update of the residual, and since this is done by decreasing a term proportional to $\mathbf{a}_1$ (or any other atom from within the correct support), this residual is also a linear combination of the same k_0 columns in $\mathbf{A}$ at the most, therefore Equation (4.13) holds in each iteration. The rest of the calculation is a little more complicated and we report it for more clarity.

Suppose we are at the $i_0 < k_0$ iteration, the condition, similar to (4.15), that needs to be satisfied for $i > k_0$ is

$$|\mathbf{a}_{i_0}^T \mathbf{b}| > |\mathbf{a}_i^T \mathbf{b}|. \tag{4.21}$$

Substituting again $\mathbf{b}$ as defined in Equation (4.13) and calculating cleverly the two bounds (majorizing $|x_1| \geq ... \geq |x_{i_0-1}|$ by $|x_1|$ and $|x_{i_0}| \geq ... \geq |x_{k_0}|$ by $|x_{i_0}|$) we obtain

$$\begin{aligned} |x_{i_0}| - |x_{i_0}| \cdot \mu(\mathbf{A})(k_0 - i_0) - |x_1| \cdot \mu(\mathbf{A})(i_0 - 1) > \\ |x_1| \cdot \mu(\mathbf{A})(i_0 - 1) + |x_{i_0}| \cdot \mu(\mathbf{A})(k_0 - i_0 + 1). \end{aligned} \tag{4.22}$$

Simplifying

$$\begin{aligned}
&\Leftrightarrow |x_{i_0}|(1-\mu(\mathbf{A})(k_0-i_0)) - |x_{i_0}|\mu(\mathbf{A})(k_0-i_0+1) > 2|x_1|\mu(\mathbf{A})(i_0-1)\\
&\Rightarrow |x_{i_0}|(1-\mu(\mathbf{A})(k_0-i_0)) - |x_{i_0}|\mu(\mathbf{A})(k_0-i_0+1) > 2|x_{i_0}|\mu(\mathbf{A})(i_0-1)\\
&\qquad\Leftrightarrow (1-\mu(\mathbf{A})(k_0-i_0)) - \mu(\mathbf{A})(k_0-i_0+1) > 2(i_0-1)\mu(\mathbf{A})\\
&\qquad\qquad\Leftrightarrow 1-2\mu(\mathbf{A})k_0 > -\mu(\mathbf{A}).
\end{aligned} \tag{4.23}$$

In the second step we used $|x_1| \geq |x_{i_0}|$ the rest is simple algebra.

From the last step we easily obtain Equation (4.20) which demonstrates that the condition is sufficient for every iteration $1 \leq i_0 \leq k_0$. After k_0 such iterations the residual becomes zero and the algorithm stops ensuring the success in recovering the correct solution $\mathbf{x}$ as the theorem claims[7]. □

The next result is the performance guarantee for the Thresholding algorithm. The Thresholding algorithm is the simplest, and the one with the highest level of greediness, it is then somewhat expected that his success rate is weaker than the OMP.

Theorem 4.8. ***(Performance Guarantee - Thresholding - General Case)****: Given a system of linear equations* $\mathbf{Ax} = \mathbf{b}$ *(*$\mathbf{A} \in \mathbb{R}^{n\times m}$ *full-rank with* $n < m$*), if a (unique) solution* $\mathbf{x}$ *(with minimal non-zero value* $|x_{min}|$ *and maximal one* $|x_{max}|$*) exists obeying*

$$\|\mathbf{x}\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{A})} \cdot \frac{|x_{min}|}{|x_{max}|}\right), \tag{4.24}$$

then the Thresholding algorithm, run with threshold parameter $\epsilon_0 = 0$*, is guaranteed to find it* ***exactly****.*

Proof. Similarly as in the previous proof suppose that the (unique) sparsest solution $\mathbf{x}$ has k_0 non-zero entries and that they are at the beginning of the vector, from the *Update Support* stage success of the Thresholding algorithm is guaranteed by the requirement

$$\min_{1\leq i\leq k_0} |\mathbf{a}_i^T\mathbf{b}| > \max_{j>k_0} |\mathbf{a}_j^T\mathbf{b}|. \tag{4.25}$$

Substituting $\mathbf{b}$ which is defined exactly as in Equation (4.13) the left side term becomes

$$\min_{1\leq i\leq k_0} |\mathbf{a}_i^T\mathbf{b}| = \min_{1\leq i\leq k_0} \left|\sum_{t=1}^{k_0} x_t \mathbf{a}_i^T\mathbf{a}_t\right|. \tag{4.26}$$

As before we are looking for a lower bound for the left side and an upper bound for the right side. We continue the calculations

[7]The *Update Provisional Solution* stage of the OMP, Algorithm 3.1, is powerful enough to guarantee perfect recovery if the exact support is found.

$$\begin{aligned}
\min_{1\le i\le k_0}\left|\sum_{t=1}^{k_0} x_t \mathbf{a}_i^T\mathbf{a}_t\right| &= \min_{1\le i\le k_0}\left|x_i + \sum_{1\le t\le k_0, t\neq i} x_t \mathbf{a}_i^T\mathbf{a}_t\right| \\
&\ge \min_{1\le i\le k_0}\left\{|x_i| - \left|\sum_{1\le t\le k_0, t\neq i} x_t \mathbf{a}_i^T\mathbf{a}_t\right|\right\} \\
&\ge \min_{1\le i\le k_0}|x_i| - \max_{1\le i\le k_0}\left|\sum_{1\le t\le k_0, t\neq i} x_t \mathbf{a}_i^T\mathbf{a}_t\right| \\
&\ge |x_{min}| - |x_{max}|\cdot\mu(\mathbf{A})(k_0-1).
\end{aligned} \tag{4.27}$$

In the first inequality we used the relation $|a+b|\ge|a|-|b|$, in the second we exploited the well-known property of min and max functions[8], in the third the fact that the columns of $\mathbf{A}$ are normalized and that their inner product is bounded from above by $\mu(\mathbf{A})$, see Definition 2.1.

Turning to the right side of Equation (4.25), using the exact same steps as in (4.18) and noting that in this case $|x_1|$ becomes more generically $|x_{max}|$ we obtain

$$\max_{j>k_0}|\mathbf{a}_j^T\mathbf{b}| \le |x_{max}|\cdot\mu(\mathbf{A})k_0. \tag{4.28}$$

Recalling that $\|\mathbf{x}\|_0 = k_0$, the condition we get is

$$|x_{min}| - |x_{max}|\cdot\mu(\mathbf{A})(\|\mathbf{x}\|_0 - 1) > |x_{max}|\cdot\mu(\mathbf{A})\|\mathbf{x}\|_0, \tag{4.29}$$

which, after a few simple algebraic steps, leads to the condition claimed by the Theorem. □

Other greedy algorithms have very similar results but since the proofs have little to no structural difference than the ones already given we omit them. We turn instead to the Basis-Pursuit, surprisingly the same bound on the sparsity as in the OMP analysis also guarantees success. However it is important to note that this does not imply that the two algorithms perform similarly[9].

Theorem 4.9. ***(Performance Guarantee - BP - General Case)****: Given a system of linear equations* $\mathbf{Ax}=\mathbf{b}$ *(*$\mathbf{A}\in\mathbb{R}^{n\times m}$ *full-rank with* $n<m$*), if a (unique) solution* $\mathbf{x}$ *exists obeying*

$$\|\mathbf{x}\|_0 < \frac{1}{2}\left(1+\frac{1}{\mu(\mathbf{A})}\right), \tag{4.30}$$

then BP is guaranteed to find it ***exactly****, i.e. that solution is both the unique solution of* $(\mathcal{P}_1)$ *and* $(\mathcal{P}_0)$*.*

[8] $\min(|f(x)|-|g(x)|)\ge\min|f(x)|+\min(-|g(x)|)=\min|f(x)|-\max|g(x)|$.

[9] Already for the two-ortho case this was found not to be true and it will be further confirmed in Section 4.4

Proof. Let us define the set of alternative solutions:

$$C = \left\{ \mathbf{y} : \ \mathbf{y} \neq \mathbf{x}, \|\mathbf{y}\|_1 \leq \|\mathbf{x}\|_1, \|\mathbf{y}\|_0 > \|\mathbf{x}\|_0 \ \text{ and } \mathbf{A}\mathbf{y} = \mathbf{b} \ \right\}. \tag{4.31}$$

The set C contains all the possible solutions that are different from the given $\mathbf{x}$, have larger ℓ_0 norm, satisfy the linear system of equations $\mathbf{A}\mathbf{y} = \mathbf{b} = \mathbf{A}\mathbf{x}$ and are at least as good from the ℓ_1 norm perspective. This set being non-empty implies that there is an alternative solution that the Basis-Pursuit will (or could, in case $\|\mathbf{y}\|_1 = \|\mathbf{x}\|_1$) find, rather that the desired $\mathbf{x}$.

Considering Theorem 2.13 and $\|\mathbf{x}\|_0 < 1/2\,(1 + 1/\mu(\mathbf{A}))$ which is the given assumption, $\mathbf{x}$ is necessarily the unique sparsest possible solution and therefore the third condition of C is redundant. Defining $\mathbf{e} = \mathbf{y} - \mathbf{x}$ we can rewrite C as a shifted version of it around $\mathbf{x}$:

$$C_s = \{\mathbf{e} : \mathbf{e} \neq \mathbf{0}, \ \|\mathbf{e} + \mathbf{x}\|_1 - \|\mathbf{x}\|_1 \leq 0 \ \text{ and } \mathbf{A}\mathbf{e} = \mathbf{0}\}. \tag{4.32}$$

The strategy of the proof will be to enlarge the set C_s and show that even that enlarged set is empty which will prove that BP indeed succeeds in recovering $\mathbf{x}$ giving a sufficient condition for doing so.

We start with the second requirement. Assuming, without loss of generality, that by a simple columns permutation of $\mathbf{A}$, the k_0 non-zeros in $\mathbf{x}$ are located at the beginning of the vector, the requirement can be written as

$$\|\mathbf{e} + \mathbf{x}\|_1 - \|\mathbf{x}\|_1 = \sum_{j=1}^{k_0} \left(|e_j + x_j| - |x_j| \right) + \sum_{j>k_0} |e_j| \leq 0. \tag{4.33}$$

Using the inequality $-|a| \leq |a+b| - |b|$ we obtain

$$-\sum_{j=1}^{k_0} |e_j| + \sum_{j>k_0} |e_j| \leq \sum_{j=1}^{k_0} \left(|e_j + x_j| - |x_j| \right) + \sum_{j>k_0} |e_j| \leq 0. \tag{4.34}$$

The last inequality can be written more compactly by adding and subtracting the term $\sum_{j=1}^{k_0} |e_j|$ and denoting it as $\mathbf{1}_{k_0}^T |\mathbf{e}|$ ($\mathbf{1}_{k_0}$ is a column vector of length m which has ones at the first k_0 entries and zeros at the rest). This leads to

$$\|\mathbf{e}\|_1 - 2\mathbf{1}_{k_0}^T |\mathbf{e}| \leq 0. \tag{4.35}$$

Substituting into the definition of C_s we get

$$C_s \subseteq C_s^1 = \left\{ \mathbf{e} : \mathbf{e} \neq \mathbf{0}, \ \|\mathbf{e}\|_1 - 2\mathbf{1}_{k_0}^T |\mathbf{e}| \leq 0 \ \text{ and } \mathbf{A}\mathbf{e} = \mathbf{0} \right\}. \tag{4.36}$$

The new set C_s^1 enlarges the initial set C_s because, as shown, every vector $\mathbf{e}$ satisfying the condition $\|\mathbf{e} + \mathbf{x}\|_1 - \|\mathbf{x}\|_1 \leq 0$ must also satisfy $\|\mathbf{e}\|_1 - 2\mathbf{1}_{k_0}^T |\mathbf{e}| \leq 0$ but not vice-versa[10].

[10] This simply derives from $-|a| \leq |a+b| - |b|$ which can also be strictly less.

We now turn to handle the third requirement $\mathbf{Ae} = \mathbf{0}$. A simple multiplication by $\mathbf{A}^T$ (which does not yet change C_s) yields the condition $\mathbf{A}^T\mathbf{Ae} = \mathbf{0}$. Subtracting the vector $\mathbf{e}$ to both sides we obtain

$$-\mathbf{e} = \left(\mathbf{A}^T\mathbf{A} - \mathbf{I}\right)\mathbf{e}. \tag{4.37}$$

Taking an entry-wise absolute value on both sides we calculate

$$\begin{aligned} |\mathbf{e}| = |(\mathbf{A}^T\mathbf{A} - \mathbf{I})\mathbf{e}| &\leq |\mathbf{A}^T\mathbf{A} - \mathbf{I}||\mathbf{e}| \\ &\leq \mu(\mathbf{A})(\mathbf{1}\mathbf{1}^T - \mathbf{I})|\mathbf{e}| \\ &= \mu(\mathbf{A})(\mathbf{1}^T|\mathbf{e}|)\mathbf{1} - \mu(\mathbf{A})|\mathbf{e}| \\ &= \mu(\mathbf{A})(\|\mathbf{e}\|_1)\mathbf{1} - \mu(\mathbf{A})|\mathbf{e}| \\ \Rightarrow |\mathbf{e}| &\leq \frac{\mu(\mathbf{A})}{1+\mu(\mathbf{A})}\|\mathbf{e}\|_1\mathbf{1}. \end{aligned} \tag{4.38}$$

In the first inequality we used the well-known relation $\|\sum_i g_i v_i\| \leq \sum_i |g_i||v_i|$ and applied it for every row of the vector, in the second inequality we simply used the definition of the mutual-coherence, see Definition 2.1, for $\mathbf{A}$ already normalized, and the rest is pure algebraic manipulation where $\mathbf{1}$ is a column vector of ones of length m.

Substituting into the definition of C_s^1 we get

$$C_s^1 \subseteq C_s^2 = \left\{\mathbf{e} : \mathbf{e} \neq \mathbf{0},\ \|\mathbf{e}\|_1 - 2\mathbf{1}_{k_0}^T|\mathbf{e}| \leq 0 \text{ and } |\mathbf{e}| \leq \frac{\mu(\mathbf{A})}{1+\mu(\mathbf{A})}\|\mathbf{e}\|_1\mathbf{1}\right\}. \tag{4.39}$$

The new set C_s^2 enlarges the set C_s^1 as shown in the calculations above and it is a scale-invariant set since if $\mathbf{e} \in C_s^2$ then $\alpha\mathbf{e} \in C_s^2$ for all $\alpha \neq 0$. Since our aim is to investigate whether C_s^2 is empty (or not) it is sufficient to consider the intersection of this set with the unit ℓ_1 sphere[11]. We denote this intersection as C_r:

$$C_r = \left\{\mathbf{e} : \|\mathbf{e}\|_1 = 1,\ 1 - 2\mathbf{1}_{k_0}^T|\mathbf{e}| \leq 0 \text{ and } |\mathbf{e}| \leq \frac{\mu(\mathbf{A})}{1+\mu(\mathbf{A})}\mathbf{1}\right\}. \tag{4.40}$$

Our goal is to find a requirement for which the set $C_r = \emptyset$. In order to do so we first put an equality sign in the third condition, we thus obtain $|e_j| = \mu(\mathbf{A})/(1+\mu(\mathbf{A}))$. We already used the first condition $\|\mathbf{e}\|_1 = 1$ when constructing C_r therefore we turn to the second, i.e. we want the second condition not to be true. It follows

$$1 - 2\mathbf{1}_{k_0}^T|\mathbf{e}| = 1 - 2k_0\frac{\mu(\mathbf{A})}{1+\mu(\mathbf{A})} > 0. \tag{4.41}$$

After some simple algebraic steps we obtain

$$\|\mathbf{x}\|_0 = k_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{A})}\right). \tag{4.42}$$

□

[11] $C_r = \emptyset \Rightarrow C_s^2 = \emptyset$: supposing $C_s^2 \neq \emptyset$ this means that there exist a vector $\mathbf{e} \in C_s^2$ which from the scale invariance described implies that $\mathbf{e}/\|\mathbf{e}\|_1 \in C_r$. $C_r \subseteq C_s^2$ trivially implies $C_r \neq \emptyset \Rightarrow C_s^2 \neq \emptyset$.

4.4 Numerical Demonstration of Algorithms

After analysing the problem ($\mathcal{P}_0$), demonstrating the uniqueness-optimality results and introducing different greedy and relaxation algorithms and their theoretical guarantees, it is now time to demonstrate their comparative behaviour on a relatively simple but representative case study.

We create a random matrix $\mathbf{A}$ of size 30×50 with entries drawn from the standard normal distribution. We normalize the columns of this matrix to have a unit ℓ_2-norm[12]. We generate sparse vectors $\mathbf{x} \in \mathbb{R}^{50}$ in two steps: first we choose k random entries which will serve as the random support $\mathcal{S}$, secondly each chosen entry is drawn from a continuous uniform distribution; once in the range $[-2,-1] \cup [1,2]$ and once in the range $[-1,1]$[13].

Once $\mathbf{x}$ is generated, we compute $\mathbf{b} = \mathbf{A}\mathbf{x}$ and then apply the algorithms to seek for $\mathbf{x}$. We perform 200 such tests per each cardinality $k \in [1,15]$. The $k \leq 15$ bound is deliberately chosen because it guarantees that in all our tests the original solution $\mathbf{x}$ is also the sparsest possible: $spark(\mathbf{A}) = 31$ (see [9, p.24]) and uniqueness-optimality then follows from Theorem 2.10.

We compare all the six algorithms presented above: the LS-OMP, the OMP, the MP, the Weak-MP with $t = 0.5$, the Thresholding and the BP. All the greedy algorithms **seek** the proper solution **until the residual is below a certain threshold**: $\|\mathbf{r}^k\|_2 \leq \epsilon_0 = \mathbf{10^{-2}}$.

When testing the success of an approximation algorithm, there are many ways to define the distance between the solution it proposes $\hat{\mathbf{x}}$ and the ideal one $\mathbf{x}$. Here we present two such measures, the standard ℓ_2-error and the recovery of the support.

The ℓ_2-error is computed as

$$err(\hat{\mathbf{x}}) = \frac{\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2}{\|\mathbf{x}\|_2^2}, \tag{4.43}$$

which is the ℓ_2-proximity between the two solutions relative to the energy of the true solution. In this case such measure does not reveal the complete story because it fails to indicate whether the support is recovered perfectly or at least partially. Thus we add another measure which computes the distance between the supports of the two solutions.

Denoting as $\hat{\mathcal{S}}$ the support proposed by the algorithm and as $\mathcal{S}$ the support of the true solution we define this distance by

$$dist(\hat{\mathcal{S}}, \mathcal{S}) = \frac{\max\{|\hat{\mathcal{S}}|, |\mathcal{S}|\} - |\hat{\mathcal{S}} \cap \mathcal{S}|}{\max\{|\hat{\mathcal{S}}|, |\mathcal{S}|\}} = 1 - \frac{|\hat{\mathcal{S}} \cap \mathcal{S}|}{\max\{|\hat{\mathcal{S}}|, |\mathcal{S}|\}}. \tag{4.44}$$

It is straightforward that $dist(\hat{\mathcal{S}}, \mathcal{S}) \in [0,1]$ being zero when the two supports are exactly the same and one when they are entirely different, with no overlap.

It has to be added that, for practical reasons, when evaluating the support we considered as being zero all entries smaller than 10^{-8}.

[12] The importance of normalization was discussed at the end of the last Chapter.

[13] We expect a somewhat different behaviour for the greedy algorithms.

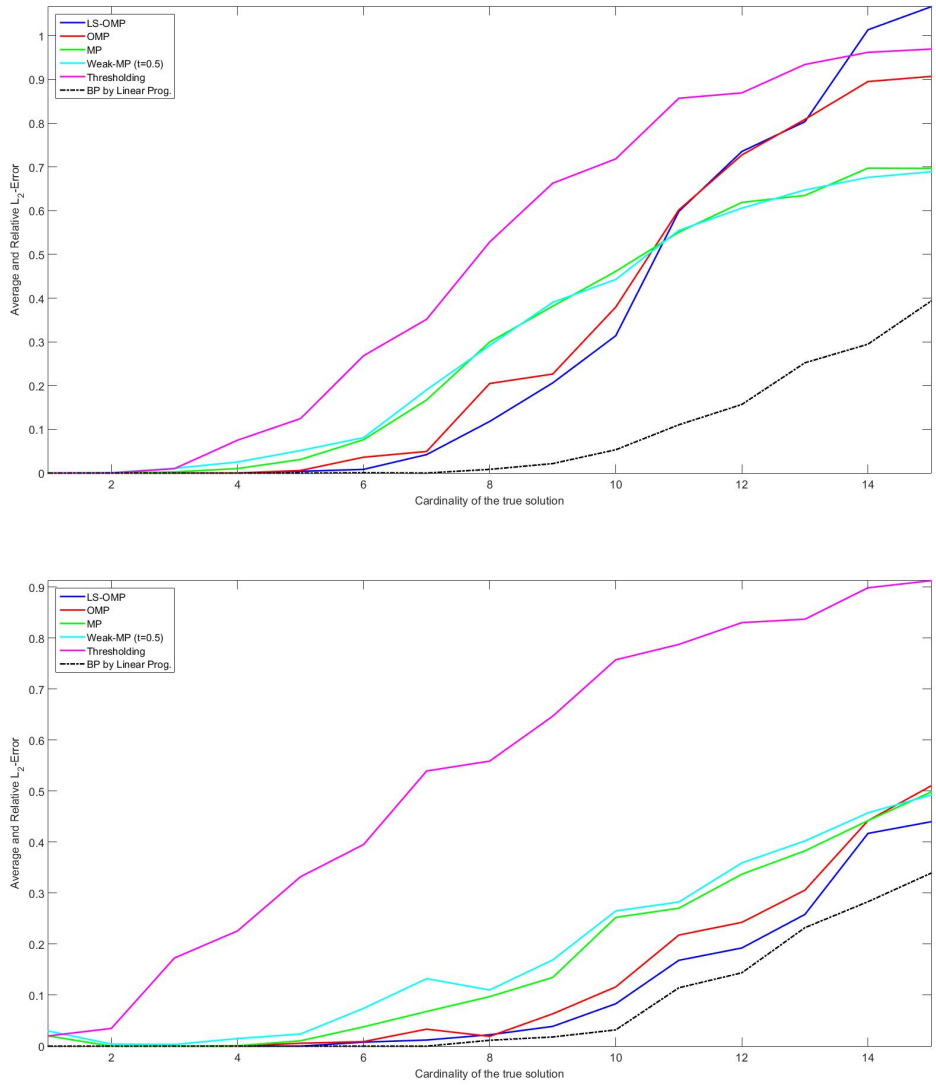

Figure 4.2: Algorithm performance in terms of relative ℓ_2-error for non-zero entries of $\mathbf{x}$ in range $[-2,-1]\cup[1,2]$ (up) and $[-1,1]$ (down).

Looking at Figure 4.2 we observe that pretty much every algorithm has an increase in performance, regarding the relative ℓ_2-error, for entries of $\mathbf{x}$ in the $[-1,1]$ range, the only exception is the Thresholding algorithm which performs worse[14]. The BP algorithm seems to perform similarly in both cases.

Looking at Figure 4.3 we observe a very similar behaviour for the support recovery error. A somewhat unexpected behaviour is the increasingly better performance of the "greediest" algorithms, especially the Thresholding which almost outperforms the LS-OMP and the OMP (in its optimal settings of the $[-2,-1]\cup[1,2]$ range). This behaviour

[14]This is consistent with Theorem 4.8.

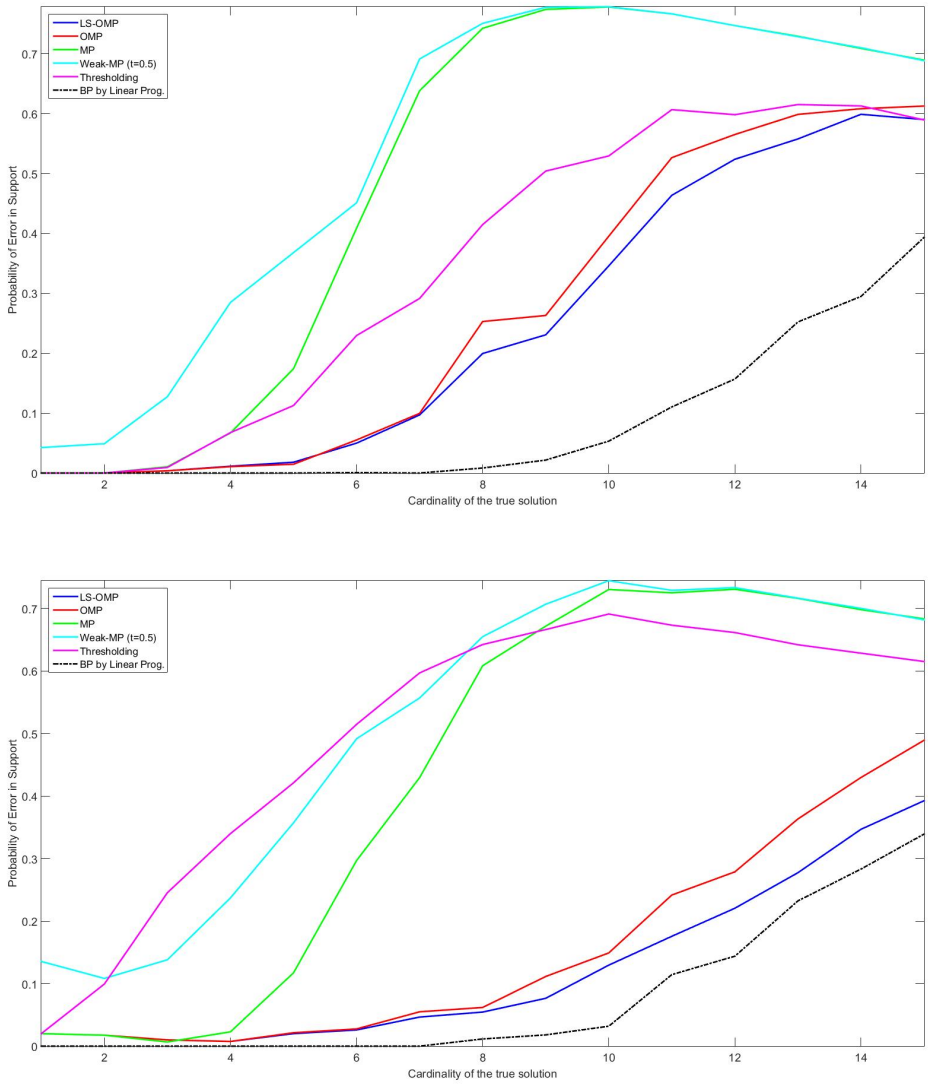

Figure 4.3: Algorithm performance in terms of success rate in detecting the support for non-zero entries of $\mathbf{x}$ in range $[-2,-1]\cup[1,2]$ (up) and $[-1,1]$ (down).

could be further explored in future analysis.

From both Figures we can conclude, as expected, that the performance is positively correlated with the cardinality of the true solution. The algorithm that without a doubt performs best (in the framework of our experiment) is the Basis-Pursuit, which has perfect recovery for all cardinalities k smaller or equal to 7.

It is interesting that for random matrices $\mathbf{A}$ as in our experiment the expected mutual-coherence is $\mu(\mathbf{A}) \approx 0.588$[15] which applying Theorems 4.7 and 4.9 leads to a theoretical

[15]With a 95% confidence interval of $[0.515, 0.687]$ and no value smaller than 0.451 obtained with a random sample of $10,000$.

perfect recovery for approx. $k \leq 1.35$, we therefore conclude that the theory results are probably not tight and far too pessimistic.

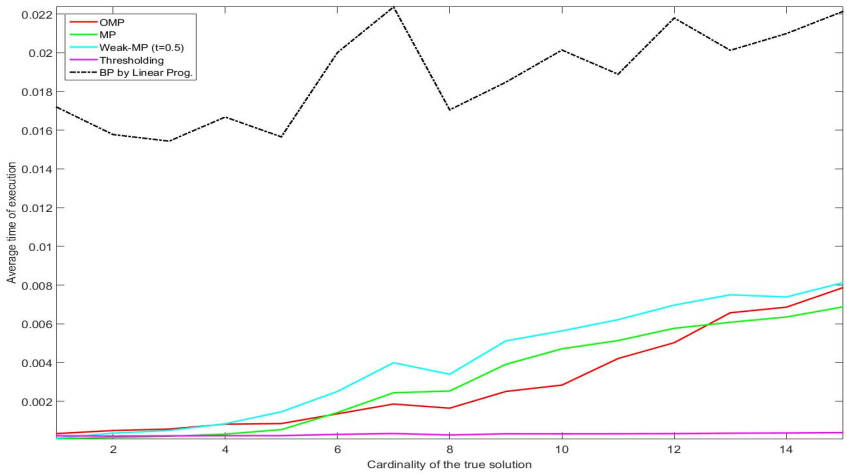

Figure 4.4: Average execution time of algorithms.

We purposely omit extensive analysis of the complexity of the given algorithms since it highly depends on implementational choices which in some cases were not optimal because the performance (and not the speed) was the main focus.

The average time of execution is presented in Figure 4.4, the LS-OMP is omitted since it was not implemented optimally[16] and therefore its results are not representative. The most important information the Figure gives is that the time of execution (i.e. the complexity) increases with the cardinality k. Secondly, as expected, the Basis-Pursuit is slower than the greedy methods.

We implemented the greedy algorithms directly and the BP applying Proposition 3.2 and then using the standard Matlab solver for linear programming problems: *linprog*[17].

[16]The recursive approach mentiooned in the LS-OMP Algorithm subsection was not implemented.

[17]We used the option which runs the Interior-point methods.

Chapter 5

The Distended Problem

In the previous chapters we thoroughly analyzed the $(\mathcal{P}_0)$ problem, demonstrated its uniqueness - optimality results and introduced different pursuit algorithms analyzing them as well. But, as mentioned already in the first chapter, real life applications are often more complex and therefore a different approach is needed, which includes questioning the fundamentals of our discussion, the problem $(\mathcal{P}_0)$ itself.

5.1 The $(\mathcal{P}_0^\epsilon)$ Problem

For clarity purposes we rewrite the definition of the $(\mathcal{P}_0)$ problem.

$$(\mathcal{P}_0): \quad \min_{\mathbf{x}} \ \|\mathbf{x}\|_0 \ \text{ subject to } \ \mathbf{Ax} = \mathbf{b}. \tag{5.1}$$

The main issue is the exact equality constraint $\mathbf{Ax} = \mathbf{b}$ which happens to be too strict. From this exactness two undesirable behaviours arise:

- Suppose that for a given vector $\mathbf{b}$ the system $\mathbf{Ax} = \mathbf{b}$ has a sparse solution $\mathbf{x}_0$. For a slightly perturbed vector $\mathbf{b}_0 := \mathbf{b} + \mathbf{e}$ ($\mathbf{e}$ being random noise) the system $\mathbf{Ax} = \mathbf{b}_0$ does not need to have a sparse solution at all, and therefore $(\mathcal{P}_0)$ would not find the original solution $\mathbf{x}_0$.

- Suppose that $\mathbf{x}_0$ is very sparse and consider a slight random perturbation of it, $\mathbf{x} := \mathbf{x}_0 + \epsilon\mathbf{u}$ ($0 < \epsilon \ll 1$ and $\|\mathbf{u}\|_2 \leq 1$). It is clear that $\mathbf{x}$ can become fully dense and therefore $\mathbf{x}_0$ would never be considered by $(\mathcal{P}_0)$ as a potential solution[1].

In other words the problem $(\mathcal{P}_0)$ is *not robust* to small perturbations (of the signal $\mathbf{b}$ or of the hypothetical signal source, which leaves in the space of the solution $\mathbf{x}$) and this is a very important property because in real world applications small random perturbations occur very often[2].

[1] More precisely if we define $\mathbf{b} := \mathbf{Ax}$ and then solve for $\mathbf{x}$ the problem $(\mathcal{P}_0)$ that arises would never consider $\mathbf{x_0}$ even though it is "very near".

[2] Physical obstacles, measure errors, numerical approximations etc...

The solution to this issue lies indeed in the introduction of an error-tolerant version of $(\mathcal{P}_0)$.

Definition 5.1. *The problem $(\mathcal{P}_0^\epsilon)$, for error tolerance $\epsilon > 0$, is defined as*

$$(\mathcal{P}_0^\epsilon): \quad \min_{\boldsymbol{x}} \ \|\mathbf{x}\|_0 \ \textit{subject to}^3 \ \|\mathbf{b}-\mathbf{A}\mathbf{x}\|_2 \leq \epsilon. \tag{5.2}$$

Proposition 5.2. *Consider the problem $(\mathcal{P}_0^\epsilon)$ such that $\mathbf{b} := \mathbf{A}\mathbf{x}_0$ for a very sparse $\mathbf{x}_0$. The given problem is robust to small perturbations $\mathbf{e}$ such that $\|\mathbf{e}\|_2 \leq \epsilon_0$, of both the signal $\mathbf{b}$ ($\epsilon_0 = \epsilon$) and the candidate solution $\mathbf{x}_0$ ($\epsilon_0 \leq \epsilon/\|\mathbf{A}\mathbf{u}\|_2$ where $\mathbf{e} := \epsilon_0\mathbf{u}, \|\mathbf{u}\|_2 \leq 1$).*

Proof. (Signal perturbation robustness) Slightly perturbing the vector $\mathbf{b}$ we obtain $\mathbf{b}_0 = \mathbf{b}+\mathbf{e}$. We calculate

$$\|\mathbf{b}_0 - \mathbf{A}\mathbf{x}_0\|_2 = \|\mathbf{b}+\mathbf{e}-\mathbf{A}\mathbf{x}_0\|_2 = \|\mathbf{e}\|_2 \leq \epsilon. \tag{5.3}$$

(Solution perturbation robustness) Slightly perturbing the candidate solution $\mathbf{x}_0$ and putting $\mathbf{e} := \epsilon_0\mathbf{u}, \|\mathbf{u}\|_2 \leq 1$ we obtain $\mathbf{x} = \mathbf{x}_0 + \epsilon_0\mathbf{u}$. We calculate

$$\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2 = \|\mathbf{A}(\mathbf{x}_0+\epsilon_0\mathbf{u})-\mathbf{b}\|_2 = \|\epsilon_0\mathbf{A}\mathbf{u}+\mathbf{A}\mathbf{x}_0-\mathbf{b}\|_2 = \epsilon_0\|\mathbf{A}\mathbf{u}\|_2. \tag{5.4}$$

Robustness is obtained if we choose ϵ_0 such that $\epsilon_0\|\mathbf{A}\mathbf{u}\|_2 \leq \epsilon$. □

It is worth noticing that when $(\mathcal{P}_0)$ and $(\mathcal{P}_0^\epsilon)$ are applied on the same problem instance, the error-tolerant problem must always give results at least as sparse as those arising in $(\mathcal{P}_0)$, since the feasible set is wider[4].

Another important thing to underline is that the signal perturbation robustness means in fact that $(\mathcal{P}_0^\epsilon)$ has a naturally strong capacity for noise removal, and indeed a similar application of that will be explored in the last chapter.

5.2 The Lack of Uniqueness

As will be intuitively shown in this Subsection we can no longer claim uniqueness of a sparse solution for $(\mathcal{P}_0^\epsilon)$ in the general case. Suppose similarly as before that a sparse vector $\mathbf{x}_0$ is multiplied by $\mathbf{A}$ and obtain a noisy observation of this product $\mathbf{b} = \mathbf{A}\mathbf{x}_0 + \mathbf{e}$ with $\|\mathbf{b}-\mathbf{A}\mathbf{x_0}\|_2 \leq \epsilon$. Consider applying $(\mathcal{P}_0^\epsilon)$ to obtain an approximation of $\mathbf{x}_0$ i.e. getting a solution $\mathbf{x}_0^\epsilon$

$$\mathbf{x}_0^\epsilon = \arg\min_{\mathbf{x}} \ \|\mathbf{x}\|_0 \ \text{subject to} \ \|\mathbf{b}-\mathbf{A}\mathbf{x}\|_2 \leq \epsilon. \tag{5.5}$$

The question is how good is this approximation? How its accuracy is affected by the sparsity of $\mathbf{x}_0$? All this will be answered briefly through the *concept of stability* which, put in words, is the claim that if a sufficiently sparse solution exists, then all alternative solutions necessarily reside "not far away from it".

[3]The use of the ℓ_2-norm is completely arbitrary due to the well-known norm equivalence in a finite-dimensional linear space.

[4]It follows from well-known properties of minimums over sets and their subsets.

Example 5.3. In this representative example we demonstrate that uniqueness can no longer be claimed. In the same settings as in (5.5) let us choose a two-ortho matrix $\mathbf{A} = [\mathbf{I}, \mathbf{O}]$ of size 2×4, $\mathbf{x}_0 = [0\ 0\ 1\ 0]^T$ and a generated random noise $\mathbf{e}$ such that $\epsilon = \|\mathbf{e}\|_2$. Thus,

$$\mathbf{A}\mathbf{x}_0 + \mathbf{e} = \begin{bmatrix} 1 & 0 & 0.6 & -0.8 \\ 0 & 1 & 0.8 & 0.6 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} 0.6 + e_1 \\ 0.8 + e_2 \end{bmatrix}. \tag{5.6}$$

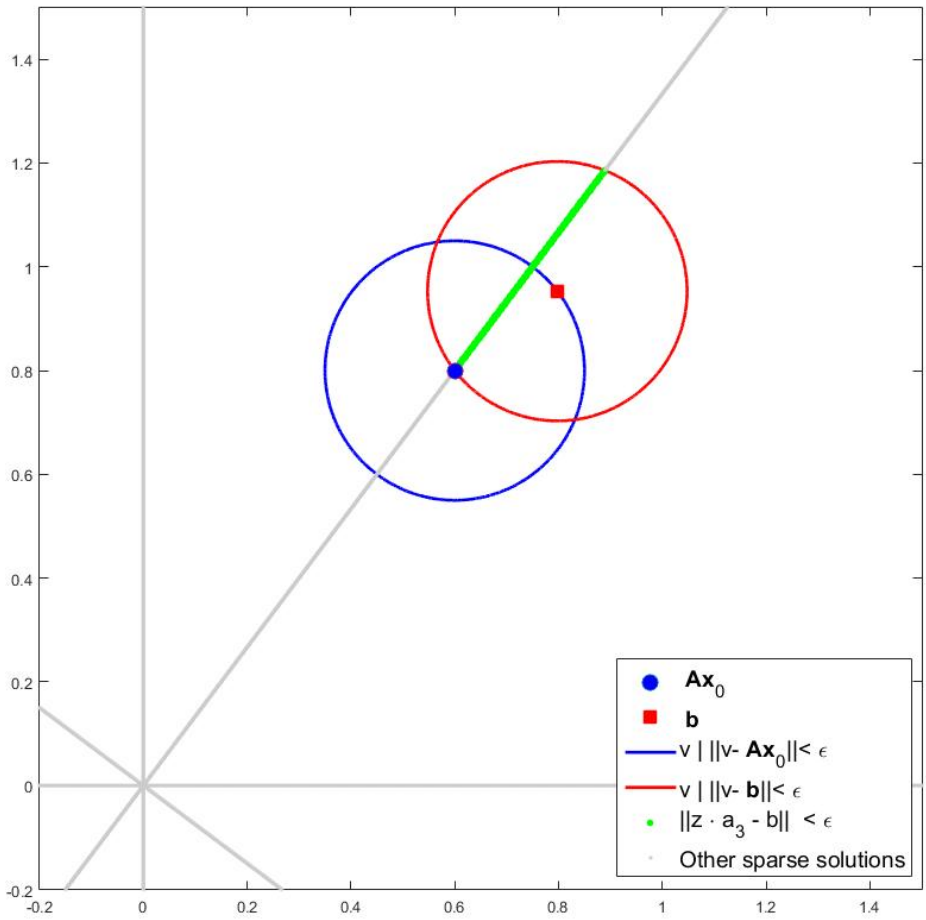

Figure 5.1: 2D demonstration of the lack of uniqueness for $(\mathcal{P}_0^\epsilon)$ with a relatively weak noise.

In Figure 5.1 are presented the locations of $\mathbf{A}\mathbf{x}_0, \mathbf{b}$ and the regions $\{\mathbf{v} :\ \|\mathbf{v} - \mathbf{A}\mathbf{x}_0\|_2 \leq \epsilon\}$ and $\{\mathbf{v} :\ \|\mathbf{v} - \mathbf{b}\|_2 \leq \epsilon\}$ with $\epsilon = 0.25$. The second of those is the region of all the feasible solutions[5] $\mathbf{x}$ to the problem $(\mathcal{P}_0^\epsilon)$.

[5] After multiplication by $\mathbf{A}$.

All the feasible sparse solutions are marked in green and, as already shown in Proposition 5.2, the vector $\mathbf{x}_0$ belongs to that set as well, indeed it is an optimal solution in the sense that no sparser solution exists. More precisely every point of the set marked in green is an optimal sparse solution having the smallest possible cardinality (one) thus uniqueness cannot be claimed.

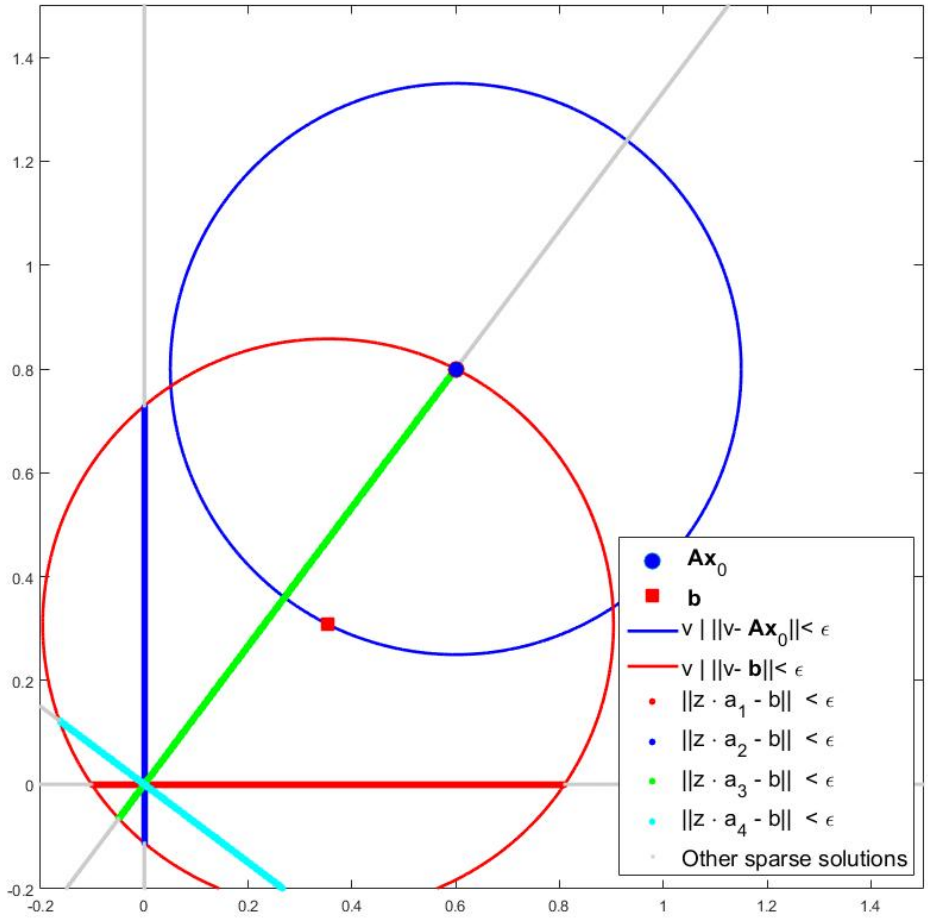

Figure 5.2: 2D demonstration of the lack of uniqueness for $(\mathcal{P}_0^\epsilon)$ with a stronger noise. Alternative solutions, even with different support, are possible.

In Figure 5.2 the same experiment is presented, the only difference is the use of a stronger noise: $\epsilon = 0.55$. This leads to a different scenario where not only we have lost uniqueness with respect to the same support but other supports with cardinality $\|\mathbf{x}\|_0 = 1$ are possible, indeed even the null solution is included, being therefore the optimal solution to $(\mathcal{P}_0^\epsilon)$.

A more formal way of explaining this behaviour is the following. Let us assume that $\mathbf{x}$ is a sparse candidate solution to the problem $(\mathcal{P}_0^\epsilon)$ over the support $\mathcal{S}$, i.e. $\|\mathbf{x}\|_0 = |\mathcal{S}|$ and, similarly as before, Let us denote with $\mathbf{x}_\mathcal{S}$ and $\mathbf{A}_\mathcal{S}$ the portions related to that support. We therefore obtain that the constraint $\|\mathbf{b} - \mathbf{A}_\mathcal{S}\mathbf{x}_\mathcal{S}\|_2 \leq \epsilon$ is satisfied.

Defining the function $f_\mathcal{S}(\mathbf{z})$ as

$$f_\mathcal{S}(\mathbf{z}) = \|\mathbf{b} - \mathbf{A}_\mathcal{S}\mathbf{z}\|_2, \tag{5.7}$$

if $\mathbf{x}_\mathcal{S}$ happens to be the minimizer of $f_\mathcal{S}$ and $f_\mathcal{S}(\mathbf{x}_\mathcal{S}) = \epsilon$, then no other solution over the same support can be proposed[6], since any perturbation around $\mathbf{x}_\mathcal{S}$ leads to an increase in the value of $f_\mathcal{S}$ thus violating the constraint.

Observing Figure 5.1, this case would take place if the closest point to $\mathbf{b}$, on the green line, was $\mathbf{A}\mathbf{x}_0$ i.e. if the distortion $\mathbf{e} = \mathbf{b} - \mathbf{A}_\mathcal{S}\mathbf{x}_\mathcal{S}$ was orthogonal to the columns of $\mathbf{A}_\mathcal{S}$.[7]

In all other cases the fact that $\min_\mathbf{z} f_\mathcal{S}(\mathbf{z}) < \epsilon$ implies an ability to perturb $\mathbf{x}_\mathcal{S}$ in a way that preserves its feasibility and the support and thus we get a set of solutions that are as good as $\mathbf{x}$ (as in Figure 5.1).

Furthermore if some of the non-zero entries in $\mathbf{x}$ are small enough the perturbation may null them, leading to an even sparser solution (as in Figure 5.2).

5.3 Introduction to Stability Analysis

In the last section we showed that for the $(\mathcal{P}_0^\epsilon)$ problem we cannot expect uniqueness of the solution, but there is a less tight result that can be obtained, which is the stability of the solution.

In order to present this theoretical analysis we need to introduce a new measure of quality of a given matrix $\mathbf{A}$ that replaces the spark. This measure is called *Restricted Isometry Property* (RIP) and was introduced by Candes and Tao in [4].

Definition 5.4. ***(RIP)*** *Let $\mathbf{A}$ be a matrix of size $n \times m$ with ℓ_2-normalized columns and let $1 \leq s \leq m$ be an integer. If there exist a constant $\delta_s \geq 0$ such that, for every $m \times s$ sub-matrix A_s of $\mathbf{A}$ and for every s-dimensional[8] vector $\mathbf{z}$,*

$$(1 - \delta_s)\|\mathbf{z}\|_2^2 \leq \|A_s\mathbf{z}\|_2^2 \leq (1 + \delta_s)\|\mathbf{z}\|_2^2, \tag{5.8}$$

then the matrix $\mathbf{A}$ is said to satisfy the RIP with constant δ_s.

Two comments have to be made. The first one is that we can replace (5.8) with an equivalent condition more optimal to our analysis where we never mention the sub-matrices A_s and instead use the complete matrix $\mathbf{A}$ while putting a constraint on the cardinality of $\mathbf{z}$,

$$\begin{gathered}(1 - \delta_s)\|\mathbf{z}\|_2^2 \leq \|\mathbf{A}\mathbf{z}\|_2^2 \leq (1 + \delta_s)\|\mathbf{z}\|_2^2 \\ \forall \mathbf{z} \in \mathbb{R}^m \text{ s.t. } \|\mathbf{z}\|_0 = s.\end{gathered} \tag{5.9}$$

[6] We have uniqueness over the support $\mathcal{S}$, nothing can be guaranteed over a different support.

[7] As the minimizer of $f_\mathcal{S}$, the vector $\mathbf{x}_\mathcal{S}$ should satisfy $\mathbf{A}_\mathcal{S}^T(\mathbf{b} - \mathbf{A}_\mathcal{S}\mathbf{x}_\mathcal{S}) = \mathbf{A}_\mathcal{S}^T\mathbf{e} = \mathbf{0}$.

[8] The matrix $\mathbf{A}$ and the vector $\mathbf{z}$ can be complex or real, but in our analysis we consider them always real.

The second one, is what we are really interested in, is the so called *s-Restricted Isometry Constant* (s-RIC) which is simply defined as the infimum of δ_s over all s of same cardinality

$$\delta_s^C = \inf\left\{\delta_s : \begin{array}{c}(1-\delta_s)\|\mathbf{z}\|_2^2 \leq \|\mathbf{A}\mathbf{z}\|_2^2 \leq (1+\delta_s)\|\mathbf{z}\|_2^2 \\ \forall \mathbf{z} \in \mathbb{R}^m \text{ s.t. } \|\mathbf{z}\|_0 = s\end{array}\right\}. \tag{5.10}$$

The intuition behind the s-RIC is that the matrix $\mathbf{A}$ is guaranteed to only change the length of any vector $\mathbf{z}$ "very little" (behaves almost like an orthogonal transform) as long as the vector $\mathbf{z}$ is at least s-sparse. Also it is easy to comprehend that given a matrix $\mathbf{A}$ it is almost impossible[9] to evaluate δ_s^C for $s \gg 1$.

Proposition 5.5. *The s-RIC, δ_s^C and the mutual-coherence, $\mu(\mathbf{A})$ of the matrix $\mathbf{A}$ are related through the following inequality*

$$\delta_s^C \leq (s-1)\mu(\mathbf{A}). \tag{5.11}$$

Proof. Applying the Gershgorin Disk Theorem 2.11 and the definition of the mutual-coherence to the matrix $A_s^T A_s$[10] we obtain that for all its eigenvalues λ it holds

$$|\lambda - 1| \leq (s-1)\mu(\mathbf{A}), \tag{5.12}$$

where $|\cdot|$ is a complex norm.

Recalling that our analysis takes place in a real vector space where $\mathbf{A}$ is real and therefore $A_s^T A_s$ is a real symmetric matrix it follows

$$1-(s-1)\mu(\mathbf{A}) \leq \lambda \leq 1+(s-1)\mu(\mathbf{A}). \tag{5.13}$$

Applying the properties of the Rayleigh quotient, see [17], to the matrix $A_s^T A_s$ we obtain

$$\lambda_{\min} \leq \frac{\|A_s\mathbf{z}\|_2^2}{\|\mathbf{z}\|_2^2} \leq \lambda_{\max}. \tag{5.14}$$

Combining (5.13) and (5.14) we basically showed that $(s-1)\mu(\mathbf{A})$ is a possible δ_s constant, the claim of the proposition follows from the basic infimum property of δ_s^C. □

Corollary 5.6. *Let $\mathbf{A}$ be a matrix of size $n \times m$, $n, m > 1$ with ℓ_2-normalized columns, let δ_s^C be its s-RIC and $\mu(\mathbf{A})$ its mutual-coherence. The following holds:*

(a) $\delta_1^C = 0$

(b) $\delta_2^C = \mu(\mathbf{A})$

(c) $\delta_s^C \leq \delta_{s+1}^C$

(d) $\delta_s^C \geq 1 \Rightarrow s \geq spark(\mathbf{A})$

(e) $\delta_s^C \leq (s-1)\mu(\mathbf{A})$

[9] The computation of the RIP/s-RIC is a Strong NP-Hard problem.

[10] Recall that $\mathbf{z} \in \mathbb{R}^s$ and $\mathbf{z} \in \mathbb{R}^m$ s.t. $\|\mathbf{z}\|_0 = s$ are equivalent.

Proof. *(a)* If $s = 1$ it follows that $\mathbf{z}$ is a scalar and A_s is a single normalized column from $\mathbf{A}$. Thus

$$\|A_s\mathbf{z}\|_2^2 = \mathbf{z}^T {A_s}^T A_s\mathbf{z} = \mathbf{z}^2 \Rightarrow \delta_s = 0. \tag{5.15}$$

and therefore $\delta_1^C = 0$.
(b) If $s = 2$ it follows that A_s contains two normalized columns from $\mathbf{A}$. Thus

$$A_s^T A_s = \begin{bmatrix} a_1^T \\ a_2^T \end{bmatrix} \begin{bmatrix} a_1^T a_2^T \end{bmatrix} = \begin{bmatrix} 1 & a_1^T a_2 \\ a_2^T a_1 & 1 \end{bmatrix}. \tag{5.16}$$

For all pairs of two columns there must be at least one that gives $a_1^T a_2 = \mu$ meaning that the eigenvalues of that matrix are *exactly* $1-\mu$ and $1+\mu$[11]. Applying (5.14) it follows that $\delta_2^C = \mu$.
(c) Looking at the definition of s-RIC it is immediately clear that $\delta_{s+1}^C < \delta_s^C$ cannot be because the one "extra" scalar in $\mathbf{z}$ cannot decrease the value of the s-RIC (if the new extra scalar in $\mathbf{z}$ is chosen very small it can make δ_{s+1} arbitrarily near but always greater or equal to δ_s^C, therefore $\delta_{s+1}^C \geq \delta_s^C$). See also [9, p.88].
(d) Applying $\delta_s^C \geq 1$ to the left inequality in the definition of s-RIC we obtain

$$\exists \mathbf{z} \in \mathbb{R}^m,\ \|\mathbf{z}\|_0 = s \quad \text{s.t.} \quad \|\mathbf{A}\mathbf{z}\|_2^2 = 0. \tag{5.17}$$

Using the well-known property of norms if follows $\mathbf{A}\mathbf{z} = 0$ which means that $\mathbf{A}$ has at least s linearly dependent columns and therefore

$$s \geq spark(\mathbf{A}), \tag{5.18}$$

with equality obtained for $s_0 = \min\{s : \delta_s^C \geq 1\}$.
(e) See Proposition 5.5. □

We now have all the tools necessary for an elegant proof of the stability result.

Theorem 5.7. ***(Stability of*** $(\mathcal{P}_0^\epsilon)$***)*** *Consider the instance of problem* $(\mathcal{P}_0^\epsilon)$ *defined by the triplet* $(\mathbf{A}, \mathbf{b}, \epsilon)$. *Suppose that a vector* $\mathbf{x}_0 \in \mathbb{R}^m$ *is a feasible potential solution to* $(\mathcal{P}_0^\epsilon)$ *satisfying the sparsity constraint*

$$\|\mathbf{x}_0\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{A})}\right), \tag{5.19}$$

then every solution $\tilde{\mathbf{x}}$ *of* $(\mathcal{P}_0^\epsilon)$ *must obey*[12]

$$\|\tilde{\mathbf{x}} - \mathbf{x}_0\|_2^2 \leq \frac{4\epsilon^2}{1 - \mu(\mathbf{A})(2\|\mathbf{x}_0\|_0 - 1)}. \tag{5.20}$$

[11] The interval $[1-\mu, 1+\mu]$ contains the eigenvalues of every $A_s^T A_s$ matrix due to the well-known basic majorization property of $\mu(\mathbf{A})$.

[12] Recall that we do not have uniqueness, therefore many solutions are generally possible.

Proof. First of all let us put $s_0 = \|\mathbf{x}_0\|_0$ and recall that $\mathbf{x}_0$ being a feasible potential solution simply means that $\|\mathbf{b} - \mathbf{A}\mathbf{x}_0\| \leq \epsilon$.

Now, as assumed, we solve $(\mathcal{P}_0^\epsilon)$ and get a candidate solution, denoted by $\tilde{\mathbf{x}}$. As already seen it follows that $\tilde{\mathbf{x}}$ has to be at least as sparse as $\mathbf{x}_0$, i.e. $\|\tilde{\mathbf{x}}\|_0 \leq s_0$ and $\|\mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}\| \leq \epsilon$. Defining $\mathbf{d} := \tilde{\mathbf{x}} - \mathbf{x}_0$ we obtain

$$\|\mathbf{A}\mathbf{d}\|_2 = \|\mathbf{A}\tilde{\mathbf{x}} - \mathbf{A}\mathbf{x}_0\|_2 \leq \|\mathbf{A}\tilde{\mathbf{x}}\|_2 + \|\mathbf{A}\mathbf{x}_0\|_2 \leq 2\epsilon \tag{5.21}$$

and similarly

$$\|\mathbf{d}\|_0 = \|\tilde{\mathbf{x}} - \mathbf{x}_0\|_0 \leq \|\tilde{\mathbf{x}}\|_0 + \|\mathbf{x}_0\|_0 \leq 2s_0. \tag{5.22}$$

After some standard algebraic manipulations we obtain that the assumption (5.19) is equivalent to

$$(2s_0 - 1)\mu(\mathbf{A}) < 1. \tag{5.23}$$

Thus, using Proposition 5.5, it follows that $\delta_{2s_0}^C < 1$. Using the last obtained inequality[13] and exploiting the lower bound in the definition of s-RIC we get

$$(1 - \delta_{2s_0}^C)\|\mathbf{d}\|_2^2 \leq \|\mathbf{A}\mathbf{d}\|_2^2 \leq 4\epsilon^2 \tag{5.24}$$

and thus we get a stability claim of the form

$$\|\mathbf{d}\|_2^2 = \|\tilde{\mathbf{x}} - \mathbf{x}_0\|_2^2 \leq \frac{4\epsilon^2}{1 - \delta_{2s_0}^C}. \tag{5.25}$$

Using the upper bound of Proposition 5.5 again we finally obtain

$$\|\mathbf{d}\|_2^2 = \|\tilde{\mathbf{x}} - \mathbf{x}_0\|_2^2 \leq \frac{4\epsilon^2}{1 - \delta_{2s_0}^C} \leq \frac{4\epsilon^2}{1 - (2s_0 - 1)\mu(\mathbf{A})}. \tag{5.26}$$

The final stability result holds if and only if the denominator is positive, which is equivalent to (5.23) which, as stated, is another way of writing the initial sparsity assumption. □

In words, the stability of the problem $(\mathcal{P}_0^\epsilon)$ means that, if there exist a sufficiently sparse feasible vector $\mathbf{x}_0$ then every solution cannot be far from it. This immediately implies that all the solutions of $(\mathcal{P}_0^\epsilon)$ lie in a bounded set and are very near one to another[14].

It is also very interesting to notice that for $\epsilon = 0$ the stability result suggests that $\tilde{\mathbf{x}} = \mathbf{x}_0$ which is perfectly aligned with the claim of Theorem 2.13 (meaning the two claims are consistent) thus stability can be rightfully referred to as an extension of uniqueness.

[13] Equations (5.21) and (5.22) guarantee the existence of the s-RIC.

[14] Follows from the triangle inequality of the ℓ_2 norm.

5.4 Stability of Pursuit Algorithms

Let us recall the greedy algorithms: the OMP, the LS-OMP, the MP, the Weak-MP and the Thresholding. These algorithms are tailored to efficiently solve the $(\mathcal{P}_0)$ problem, how do we modify them to solve the $(\mathcal{P}_0^\epsilon)$ problem?

The answer is incredibly simple! We just put the error threshold $\epsilon_0 = \epsilon$ i.e. we stop the iterations when the ℓ_2-norm of the residual $\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k$ is equal (or slightly less) to ϵ, which is exactly the radius of the ball centered at $\mathbf{b}$ were the feasible solutions of $(\mathcal{P}_0^\epsilon)$ lie.

We now present, without proofs, the stability results of the most important greedy algorithms.

Theorem 5.8. *(**Stability of OMP**) Consider the OMP algorithm applied to the problem instance $(\mathcal{P}_0^\epsilon)$ defined by the triplet $(\mathbf{A}, \mathbf{b}, \epsilon)$. Suppose that a vector $\mathbf{x}_0 \in \mathbb{R}^m$ is a feasible potential solution to $(\mathcal{P}_0^\epsilon)$ satisfying the sparsity constraint*

$$\|\mathbf{x}_0\|_0 < \frac{1}{2}\left(1 + \frac{1}{\mu(\mathbf{A})}\right) - \frac{\epsilon}{\mu(\mathbf{A})|x_{\min}|}, \tag{5.27}$$

where $|x_{\min}|$ is the minimum value of the vector $|\mathbf{x}_0|$ within its support. The solution produced by the OMP algorithm must obey

$$\|\mathbf{x}_{OMP} - \mathbf{x}_0\|_2^2 \leq \frac{\epsilon^2}{1 - \mu(\mathbf{A})(\|\mathbf{x}_0\|_0 - 1)}. \tag{5.28}$$

Furthermore, this algorithm is guaranteed to recover a solution with the correct support and in exactly $s = \|\mathbf{x}_0\|_0$ steps.

Proof. See [8, p.12-13]. □

Theorem 5.9. *(**Stability of Thresholding**) Consider the Thresholding algorithm applied to the problem instance $(\mathcal{P}_0^\epsilon)$ defined by the triplet $(\mathbf{A}, \mathbf{b}, \epsilon)$. Suppose that a vector $\mathbf{x}_0 \in \mathbb{R}^m$ is a feasible potential solution to $(\mathcal{P}_0^\epsilon)$ satisfying the sparsity constraint*

$$\|\mathbf{x}_0\|_0 < \frac{1}{2}\left(1 + \frac{|x_{\min}|}{|x_{\max}|} \cdot \frac{1}{\mu(\mathbf{A})}\right) - \frac{\epsilon}{\mu(\mathbf{A})|x_{\max}|}, \tag{5.29}$$

where $|x_{\min}|$ and $|x_{\max}|$ are the minimum and maximum values of the vector $|\mathbf{x}_0|$ within its support. The solution produced by the Thresholding algorithm must obey

$$\|\mathbf{x}_{THR} - \mathbf{x}_0\|_2^2 \leq \frac{\epsilon^2}{1 - \mu(\mathbf{A})(\|\mathbf{x}_0\|_0 - 1)}. \tag{5.30}$$

Furthermore, this algorithm is guaranteed to recover a solution with the correct support.

Proof. See [9, p.105-107]. □

Similarly, a few words have to be spent regarding the modification of the Basis Pursuit for the solution of the $(\mathcal{P}_0^\epsilon)$ problem. This modification is straightforward and is called the *Basis Pursuit Denoising* (BPDN). Assume that the matrix $\mathbf{A}$ has normalized columns the $(\mathcal{P}_1^\epsilon)$ is defined as

$$(\mathcal{P}_1^\epsilon): \quad \min_{\mathbf{x}} \ \|\mathbf{x}\|_1 \ \text{ subject to } \|\mathbf{b}-\mathbf{A}\mathbf{x}\|_2 \leq \epsilon. \tag{5.31}$$

This is a more complex optimization problem being quadratic in nature and the presentation of its possible solutions is beyond the purposes of this work. More can be found at [9, p.90-100].

For completeness, the stability result of the BDPN is presented.

Theorem 5.10. *(**Stability of BDPN**) Consider the instance of problem $(\mathcal{P}_1^\epsilon)$ defined by the triplet $(\mathbf{A},\mathbf{b},\epsilon)$. Suppose that a vector $\mathbf{x}_0 \in \mathbb{R}^m$ is a feasible potential solution to $(\mathcal{P}_0^\epsilon)$ satisfying the sparsity constraint*

$$\|\mathbf{x}_0\|_0 < \frac{1}{4}\left(1+\frac{1}{\mu(\mathbf{A})}\right), \tag{5.32}$$

then the solution $\mathbf{x}_1^\epsilon$ of $(\mathcal{P}_1^\epsilon)$ must obey

$$\|\mathbf{x}_1^\epsilon - \mathbf{x}_0\|_2^2 \leq \frac{4\epsilon^2}{1-\mu(\mathbf{A})(4\|\mathbf{x}_0\|_0-1)}. \tag{5.33}$$

Proof. See [9, p.103-104]. □

It can be observed that the theorem does not guarantee support recovery and that there is a noise magnification phenomenon by a factor of 4.

However this does not mean that the BDPN is worse than the other two greedy algorithms presented[15], indeed a more complex theoretical analysis and appropriate simulations would show competitiveness.

Lastly we present a theoretically strong result which guarantees stability for *an arbitrary* pursuit algorithm A used for solving $(\mathcal{P}_0^\epsilon)$.

Theorem 5.11. *(**Stability of arbitrary pursuit algorithm A**) Consider the instance of problem $(\mathcal{P}_0^\epsilon)$ defined by the triplet $(\mathbf{A},\mathbf{b},\epsilon)$. Suppose that a vector $\mathbf{x}_0 \in \mathbb{R}^m$ is a feasible potential solution to $(\mathcal{P}_0^\epsilon)$ satisfying the sparsity constraint*

$$\|\mathbf{x}_0\|_0 < \frac{1}{2}\left(1+\frac{1}{\mu(\mathbf{A})}\right), \tag{5.34}$$

then the solution $\tilde{\mathbf{x}}$ of $(\mathcal{P}_0^\epsilon)$ produced by the algorithm A must obey

$$\|\tilde{\mathbf{x}}-\mathbf{x}_0\|_2^2 \leq \frac{4\epsilon^2}{1-\delta^C_{s_0+s_1}} \leq \frac{4\epsilon^2}{1-\mu(\mathbf{A})(s_0+s_1-1)} \tag{5.35}$$

where $s_0 = \|\mathbf{x}_0\|_0$ and $s_1 = \|\tilde{\mathbf{x}}\|_0$.

[15] All known variants of greedy algorithms are included because they perform "somewhere in between" these two algorithms, the only exception is the LS-OMP.

Proof. Same as Theorem 5.7 but using $\delta^C_{s_0+s_1} < 1$. □

The result is very powerful but useless in practice, since it depends on the sparsity of the proposed solution $\tilde{\mathbf{x}}$ (and therefore is not a constant that can be tied to a specific algorithm).

Chapter 6

Applications to Signal Processing

In all the previous chapters we have shown that the problem of finding a sparse solution to an underdetermined linear system of equation, or approximation of it, can be given a meaningful definition and can be computationally tractable. In this last chapter we show an application to signal and image processing which is image inpainting and noise removal. The application chosen gives a meaningful and strong understanding of how powerful the algorithms and the ideas in this domain can be.

In the following discussion we shall replace the notation $\mathbf{b}$ in the linear system $\mathbf{Ax} = \mathbf{b}$ with $\mathbf{y}$, in order to reflect the nature of this vector as a signal of interest.

It is important to remind that the presented is just a narrow subset of all the possible applications and ideas, for more see [9, p.169-353].

6.1 The Sparseland Model

In order to describe the *Sparseland* model Let us first consider the linear system $\mathbf{Ax} = \mathbf{y}$ and interpret it as a way of constructing signals $\mathbf{y}$.

Every column in $\mathbf{A}$ is a possible signal in $\mathbb{R}^n$, usually these m columns are called *atomic signals* and therefore the matrix $\mathbf{A}$ displays a dictionary of atoms.

The multiplication of $\mathbf{A}$ by a sparse vector $\mathbf{x}$ with $\|\mathbf{x}\|_0 = k_0$ produces a linear combination of k_0 atoms with varying portions, generating the signal $\mathbf{y}$. The vector $\mathbf{x}$ that generates $\mathbf{y}$ will be called its *representation*, since it describes which atoms and what portions of them where used for its construction. This process is called *atomic-composition*.

The actual generation of a signal from the given dictionary is controlled by a prior $P(\mathbf{y})$, which is nothing else but a **probability density function (PDF)** of signals[1].

Many different priors (and models) have been proposed in the last 40 years of development in signal processing and many have a Gibbs distribution form[2] (Total-Variation, Wavelet transform based, Gaussian-mixture-models, etc.). For a relevant and rich list see [9, p.169-172].

[1] The prior is a function that quantifies how likely a particular signal is to exist.

[2] $P(\mathbf{y}) = C \cdot e^{-G(\mathbf{y})}$, for a given function $G(\mathbf{y})$.

Let us describe the above more precisely: we can think of *Sparseland* as being a random generator machine $\mathcal{M}$ with parameters $\mathbf{A}, \lambda, \alpha$ and ϵ, and describe the prior of it through several simple steps.

- **Choose the cardinality of x**, k_0, by sampling from an exponential probability distribution $X_1 \sim Exp(\lambda)$.[3]
- **Choose the support of x** by sampling from a discrete uniform distribution, i.e. between the $s = \binom{m}{k_0}$ possible supports choose one with probability $1/s$.
- **Choose the values of x** by sampling from a multivariate normal distribution $X_2 \sim \mathcal{N}(\mathbf{0}, \Sigma(\alpha))$.[4]
- **Generate the signal y** such that $\mathbf{y} := \mathbf{A}\mathbf{x} + \mathbf{e}$, the vector $\mathbf{e}$ being random gaussian noise s.t. $\|\mathbf{e}\|_2 \leq \epsilon$.

Even though the prior of the *Sparseland* model is not written directly (which is possible but unnecessarily complex) it is clear that the four steps above provide a clear and complete definition of $P(\mathbf{y})$.

Moreover it is straightforward that *Sparseland* is a **generative** model, meaning that it directly describes a way for how signals are synthetized (making sampling of new one very easy). More details and even a geometric interpretation can be found at [9, p.173-180].

Hereafter by *Sparseland* model we imply the machine $\mathcal{M}(\mathbf{A}, \lambda, \alpha, \epsilon)$.

6.2 Dictionary Learning

From the definition of *Sparseland* it is clear that the dictionary $\mathbf{A}$ is a fundamental ingredient and hyperparameter for the successful deployment of the model.

In the quest for the proper dictionary to use in applications, one line of work (the oldest one) considers choosing pre-constructed dictionaries such as Fourier, DCT, Wavelet, etc.

In this section, in order to show the full potential of this field and perhaps to connect it to the very popular field of *Machine Learning* we indeed choose a different line which is the learning approach. This learning option starts by building a *training database* of signal instances and constructing an empirically learned dictionary, in which the atomic signals come from the underlying data, rather than some theoretical model.

This approach has one important advantage and that is the capacity of adaptation to any family of signals that complies with the *Sparseland* model. However all this, firstly, comes at the price of a much higher computational load since learned dictionaries have to be held as explicit structure-less matrices, and, secondly, at a restriction to low dimensional signals[5].

[3] λ chosen such that the sample tends to sparsity, i.e. $k_0 \ll n$.

[4] This notation simply means that α is a parameter of the covariance matrix Σ.

[5] An image has to be divided in small patches in order to get a big enough *training set* for learning, therefore some information is necessarily lost.

We now turn to discuss the learning methodology precisely. Assume that a training database $\{\mathbf{y}_i\}_{i=1}^N$ is given and thought to have been generated by some fixed but unknown *Sparseland* model $\mathcal{M}(\mathbf{A},\lambda,\alpha,\epsilon)$.

The learning objective can be posed as the following optimization problem, which we define as $(\mathcal{DL})$.

$$(\mathcal{DL}): \quad \min_{\mathbf{A},\{\mathbf{x}_i\}_{i=1}^N} \sum_{i=1}^N \|\mathbf{y}_i - \mathbf{A}\mathbf{x}_i\|_2^2 \ \text{ s.t. } \|\mathbf{x}_i\|_0 \le k_0, \ \ 1 \le i \le N. \tag{6.1}$$

For $(\mathcal{DL})$ to be well-posed and admit a unique solution we need:

- The normalization[6] of the columns of $\mathbf{A}$, i.e. $diag\{\mathbf{A}^T\mathbf{A}\} = \mathbf{I}$.
- The assumption that the solution is invariant to permutations of columns of $\mathbf{A}$.

Under the above assumptions there are a lot of theoretical and practical results that prove the usefulness of $(\mathcal{DL})$ for the dictionary learning task, see [3], [9, p.229] and related work.

The K-SVD Algorithm

The K-SVD was developed in 2006 and designed to solve $(\mathcal{DL})$ and, at publication time, it achieved state-of-the-art results, see [2].

Let us rewrite $(\mathcal{DL})$ in a matrix form

$$(\mathcal{DL}'): \quad \min_{\mathbf{A},\mathbf{X}} \|\mathbf{Y} - \mathbf{A}\mathbf{X}\|_F^2 \ \text{ s.t. } \|\mathbf{x}_i\|_0 \le k_0, \ \ 1 \le i \le N, \tag{6.2}$$

where the $n \times N$ matrix $\mathbf{Y}$ is formed by concatenating all the training database vectors column-wise and similarly the $m \times N$ matrix $\mathbf{X}$ by concatenating the corresponding sparse representations. We calculate

$$\begin{aligned} \|\mathbf{Y} - \mathbf{A}\mathbf{X}\|_F^2 &= \left\| \mathbf{Y} - \sum_{j=1}^m \mathbf{a}_j \mathbf{x}_j^T \right\|_F^2 \\ &= \left\| \left(\mathbf{Y} - \sum_{j \ne j_0} \mathbf{a}_j \mathbf{x}_j^T \right) - \mathbf{a}_{j_0} \mathbf{x}_{j_0}^T \right\|_F^2 \\ &= \left\| \mathbf{E}_{j_0} - \mathbf{a}_{j_0} \mathbf{x}_{j_0}^T \right\|_F^2, \end{aligned} \tag{6.3}$$

where $\mathbf{x}_j^T$ stands for the j-th row of $\mathbf{X}$ and $\mathbf{E}_{j_0}$ refers to the term in parentheses as a known pre-computed error matrix.

The core step of the K-SVD is to find the optimal $\mathbf{a}_{j_0}$ and $\mathbf{x}_{j_0}^T$ which minimizes (6.3) for each j_0. It is well-known that this can be obtained via a rank-1 SVD approximation [12, p.57-65] but this typically yields to a dense vector $\mathbf{x}_{j_0}^T$ ruining sparsity.

[6] In order to define a precise scale between $\mathbf{A}$ and $\mathbf{x_i}$.

Algorithm 6.1: K-SVD

Task: Train a dictionary $\mathbf{A}$ which is an approximate solution to $(\mathcal{DL})$.
Parameters: Given the training data $\{\mathbf{y}_i\}_{i=1}^N$, the error threshold ϵ_0 and the maximum number of iterations k_{max}.
Initialization: Initialize $k = 0$, and

- **Initialize Dictionary:** Build $\mathbf{A}_{(0)} \in \mathbb{R}^{n\times m}$, either by using random entries or using m randomly chosen training vectors.
- **Normalization:** Normalize the columns of $\mathbf{A}_{(0)}$.

Main Iteration: Increment k by 1, and apply:

- **Sparse Coding:** Use any pursuit algorithm to approximate the solution of
$$\hat{\mathbf{x}}_i = \arg\min_{\mathbf{x}} \|\mathbf{y}_i - \mathbf{A}_{(k-1)}\mathbf{x}\|_2^2 \text{ s.t. } \|\mathbf{x}\|_0 \leq k_0$$
obtaining sparse representations $\hat{\mathbf{x}}_i$ for $1 \leq i \leq N$. These form the matrix $\mathbf{X}_{(k)}$.
- **K-SVD Dictionary Update:** Use the following to update the columns of the dictionary and obtain $\mathbf{A}_{(k)}$.
For each column $j_0 = 1, 2, ..., m$ in $\mathbf{A}_{(k-1)}$ do
  - Define the group of training signals that use the atom $\mathbf{a}_{j_0}$
$$\omega_{j_0} = \{i \mid 1 \leq i \leq N,\ \mathbf{X}_{(k)}[j_0, i] \neq 0\}.$$
  - Define $\mathbf{P}_{j_0}$ as a matrix of size $N \times |p_{j_0}|$ with ones on the $(\omega_{j_0}(i), i)$th entries and zeros elsewhere.
  - Compute the residual matrix
$$\mathbf{E}_{j_0} = \mathbf{Y} - \sum_{j\neq j_0} \mathbf{a}_j \mathbf{x}_j^T$$
where $\mathbf{x}_j^T$ are the rows of the matrix $\mathbf{X}_{(k)}$.
  - Calculate $\mathbf{E}_{j_0}^R = \mathbf{E}_{j_0}\mathbf{P}_{j_0}$.
  - Apply SVD decomposition $\mathbf{E}_{j_0}^R = \mathbf{U}\Sigma\mathbf{V}^T$. Update the dictionary atom $\mathbf{a}_{j_0} = \mathbf{u}_1$[a] and the representation $\mathbf{x}_{j_0}^R = \Sigma[1,1] \cdot \mathbf{v}_1$.
- **Stopping Rule:** If $\|\mathbf{Y} - \mathbf{A}_{(k)}\mathbf{X}_{(k)}\|_F^2 < \epsilon_0$ or $k > k_{max}$ stop. Otherwise, continue.

Output: The desired result is $\mathbf{A}_{(k)}$.

[a] The column is already normalized because $\mathbf{U}$ is unitary.

The density of the vector $\mathbf{x}_{j_0}^T$ can be solved by taking only a subset of the columns of $\mathbf{E}_{j_0}$, i.e. those columns that correspond to the training signals in $\mathbf{Y}$ which are using the j_0-th atom, namely those columns where the entries in the row $\mathbf{x}_{j_0}^T$ are non-zero. This is done by multiplying $\mathbf{E}_{j_0}$ from the right by a restriction operator[7], $\mathbf{P}_{j_0}$, which has N rows and M_{j_0} columns[8].

Algebraically the last equation in (6.3) becomes

$$\left\|\mathbf{E}_{j_0}\mathbf{P}_{j_0} - \mathbf{a}_{j_0}\mathbf{x}_{j_0}^T\mathbf{P}_{j_0}\right\|_F^2 = \left\|\mathbf{E}_{j_0}\mathbf{P}_{j_0} - \mathbf{a}_{j_0}(\mathbf{x}_{j_0}^R)^T\right\|_F^2. \tag{6.4}$$

For the sub-matrix $\mathbf{E}_{j_0}\mathbf{P}_{j_0}$ a rank-1 SVD approximation can be applied, updating simultaneously both the atom $\mathbf{a}_{j_0}$ and the corresponding coefficients in the sparse representations, $\mathbf{x}_{j_0}^R$. Note that one does not need to use the full SVD as only the first rank part of $\mathbf{E}_{j_0}\mathbf{P}_{j_0}$ is required, thus an alternative numerical approach is possible, see [9, p.233].

The whole K-SVD algorithm is written above. In order to increase the quality of the obtained dictionary we mention a simple *Correction* Stage. After each *Dictionary Update* stage do:

- If two atoms in the dictionary are too similar, discard one of them.
- If an atom in the dictionary is rarely used, discard it.

In both cases choose the worst-represented training signal from $\mathbf{Y}$ as the replacement.

6.3 Image Inpainting

Image inpainting refers to filling-in missing pixels in known locations in the image, due to undesired scratches, occlusions or because of image editing needs (removing objects from the scene). Extensive work on this problem is reported in the image processing literature, using various techniques. In this section we focus on ways to inpaint images using the *Sparseland* model.

Assume that $\mathbf{y}_0 \in \mathbb{R}^n$ is a *Sparseland* signal, meaning that there exists a sparse representation vector $\mathbf{x}_0$ with $\|\mathbf{x}_0\|_0$ such that $\mathbf{y}_0 = \mathbf{A}\mathbf{x}_0$. Assume further that we measure $\mathbf{y} = \mathbf{M}\mathbf{y}_0$, where $\mathbf{M}$ is a degradation operator that removes p samples from the signal[9].

Considering this as a classic inverse problem we may formulate the inpainting task as

$$(\mathcal{INP}): \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \ \text{ subject to } \mathbf{y} = \mathbf{M}\mathbf{A}\mathbf{x}. \tag{6.5}$$

If the representation that solves this problem is denoted as $\hat{\mathbf{x}}_0$ then the recovered (full) signal $\hat{\mathbf{y}}_0 = \mathbf{A}\hat{\mathbf{x}}_0$.

[7] The restriction operator is simply an identity matrix $\mathbf{I}$ with some missing columns.

[8] The number of training signals using the j_0-th atom.

[9] This matrix is of size $(n-p)\times n$, built by taking the $n\times n$ identity matrix $\mathbf{I}$ and removing the p rows that correspond to the dropped samples.

Similarly when noise is added (to the model and to the measurements) the inpaiting task simply becomes

$$(\mathcal{INP}_\epsilon): \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \text{ subject to } \|\mathbf{y}-\mathbf{MAx}\|_2 \leq \delta. \tag{6.6}$$

There are several theoretical and practical reasons why both these methods should (and do) work, for more see [9, p.324-327].

Image Inpainting with Local K-SVD

Let us formulate the inpainting problem in a slightly different way which will allow us to apply the K-SVD dictionary learning algorithm.

- Assume that the unknown and complete image $\mathbf{y}_0$ is such that every **patch** in it is expected to have a sparse representation with respect to a dictionary $\mathbf{A}$.
- Assume that the measured signal is $\mathbf{y} = \mathbf{My}_0$ where $\mathbf{M}$ is a degradation operator.
- Assume that the remaining pixels in $\mathbf{y}$ are noisy, contaminated by white zero-mean additive Gaussian noise with variance σ^2.

It can be shown, see [10] and [9, p.202-211], that the MAP[10] estimator for this problem is

$$(\mathcal{INP}_\epsilon^L): \quad \begin{aligned} \{\hat{\mathbf{y}}, \{\hat{\mathbf{q}}_k\}_{k=1}^N\} = \arg \min_{\mathbf{z}, \{\mathbf{q}_k\}_{k=1}^N} & \lambda\|\mathbf{Mz}-\mathbf{y}\|_2^2 + \\ & \sum_k \mu_k \|\mathbf{q}_k\|_0 + \sum_k \|\mathbf{Aq}_k - \mathbf{R}_k\mathbf{z}\|_2^2. \end{aligned} \tag{6.7}$$

Even without a maximum probability background this minimization problem is very intuitive. The first term requires a proximity between the measured image $\mathbf{y}$ and the unknown version $\mathbf{z}$, only in the existing pixels (multiplication by $\mathbf{M}$). The second and third terms are the image prior that ensures that in the reconstructed image $\mathbf{z}$ every patch $\mathbf{p}_k = \mathbf{R}_k\mathbf{z}$ of size $\sqrt{n} \times \sqrt{n}$ in every location k should have a sparse representation with a bounded error.

The algorithm for solving (6.7) is summarized below. It can be seen that a block-coordinate minimization process is adopted which starts with an initialization of $\hat{\mathbf{y}}$ and then, for a few steps, seeks the optimal $\hat{\mathbf{q}}_k$ while learning the dictionaty $\mathbf{A}$ and finally, in the last step, computes the inpainted signal $\hat{\mathbf{y}}$.

[10] Maximum a posteriori probability estimator. In our case given a (known) measurement vector $\mathbf{y}$ and an unknown vector to be estimated $\mathbf{x}$, then $\hat{\mathbf{x}}^{MAP} = \arg\max_{\mathbf{x}} \mathbb{P}(\mathbf{x}|\mathbf{y})$, i.e. the most probable original signal $\mathbf{x}$ under the condition that a corrupted signal $\mathbf{y}$ is observed.

Algorithm 6.2: Inpainting with local K-SVD

Task: Inpaint an image $\mathbf{y}$ by approximating the solution to the inpainting problem $(\mathcal{INP}_{\epsilon}^{L})$.

Parameters: Given the measured image $\mathbf{y}$, the mask $\mathbf{M}$, the noise gain c, the noise variance σ^2, the parameter λ and the maximum number of iterations j_{max}.

Initialization: Initialize $j = 0$, and

- **Initialize Dictionary:** Put the redundant DCT as $\mathbf{A}_{(0)} \in \mathbb{R}^{n \times m}$.
- **Initialize Signal:** Put $\mathbf{z} = \mathbf{M}^T\mathbf{y}$.

Main Iteration (K-SVD): Increment j by 1, and apply:

- **Sparse Coding:** Use any pursuit algorithm to approximate the solution of

$$\hat{\mathbf{q}}_k = \arg\min_{\mathbf{q}} \|\mathbf{q}\|_0 \text{ s.t. } \|\mathbf{M}_k(\mathbf{A}_{(j-1)}\mathbf{q} - \mathbf{p}_k)\|_2^2 \leq c^2 n_k \sigma^2$$

  for $1 \leq k \leq N$. Obtaining sparse representations $\hat{\mathbf{q}}_k$ for every patch $\mathbf{p}_k$.

- **K-SVD Dictionary Update:** Update the columns of the dictionary and obtain $\mathbf{A}_{(j)}$. Similarly to Algorithm 6.1.
- **Stopping Rule:** If $j > j_{max}$ stop. Otherwise, continue.

Final Step: Given $\{\hat{\mathbf{q}}_k\}_{k=1}^{N}$ from the last iteration update $\mathbf{z}$ by

$$\mathbf{z} = \left(\lambda\mathbf{M}^T\mathbf{M} - \sum_k \mathbf{R}_k^T\mathbf{R}_k\right)^{-1}\left(\lambda\mathbf{M}^T\mathbf{y} - \sum_k \mathbf{R}_k^T\mathbf{A}_{j_{max}}\hat{\mathbf{q}}_k\right).$$

Output: The desired result is $\hat{\mathbf{y}} = \mathbf{z}$.

Some clarifications of the algorithm stages need to be done.

In the *Initialization* stage we initialize the dictionary with a redundant DCT which is the *Discrete cosine transform*, more precisely we use the type-II, see [5].

In the *Sparse Coding* stage we are able to completely separate the minimization task of the inpainting problem $(\mathcal{INP}_{\epsilon}^{L})$ into N different simpler problems because they are all **mutually independent**.

Another thing that justifies this minimization approach is the fact the here we consider $\mathbf{z} = \mathbf{M}^T\mathbf{y}$ as being fixed and therefore the first term becomes constant and can be omitted in the minimization. Note how we also turned the quadratic penalty (3rd term) into a constraint[11], thus removing the need to choose the parameters μ_k.

Another thing to notice in this stage is the matrix $\mathbf{M}_k$ which simply constraints the

[11] This is a somewhat "inverse" operation of the Lagrange multipliers method.

patch-error $\mathbf{Aq}-\mathbf{p}_k$ to be evaluated using only the existing pixels of the patch[12], indeed $\mathbf{M}_k = \mathbf{R}_k\mathbf{M}^T\mathbf{M}\mathbf{R}_k^T$ i.e. the local mask that corresponds to the k-th patch. The threshold $c^2 n_k \sigma^2$ to compare with must also take into account the number of existing pixels n_k in each patch i.e. $n_k = \mathbf{1}^T\mathbf{M}_k\mathbf{1}$, while c and σ^2 are parameters considered known or experimentally found.

Summarizing the *Sparse Coding* stage works as a sliding window sparse coding stage, operating on each block of size $\sqrt{n}\times\sqrt{n}$ at a time and thus can be parallelized.

The *K-SVD Dictionary Update* stage is similar to the one of the K-SVD algorithm 6.1 but with some little differences that need to be adressed. As in the classical K-SVD, we target the update of the atoms of $\mathbf{A}$ one at a time while considering only the patches k that use that particular atom. More precisely considering the j_0-th atom $\mathbf{a}_{j_0}$ and denoting by Ω_{j_0} the set of the patches that use this atom, the error we need to minimize is[13]

$$\begin{aligned} Error(\mathbf{A}) &= \sum_{k\in\Omega_{j_0}} \|\mathbf{M}_k(\mathbf{Aq}_k-\mathbf{p}_k)\|_2^2 \\ &= \sum_{k\in\Omega_{j_0}} \|\mathbf{M}_k(\mathbf{Aq}_k-\mathbf{a}_{j_0}\mathbf{q}_k(j_0)-\mathbf{p}_k)+\mathbf{M}_k\mathbf{a}_{j_0}\mathbf{q}_k(j_0)\|_2^2 \\ &= \sum_{k\in\Omega_{j_0}} \|\mathbf{M}_k(\mathbf{y}_k^{j_0}-\mathbf{a}_{j_0}\mathbf{q}_k(j_0))\|_2^2, \end{aligned} \tag{6.8}$$

where we denoted $\mathbf{y}_k^{j_0} = \mathbf{p}_k - \mathbf{Aq}_k + \mathbf{a}_{j_0}\mathbf{q}_k(j_0)$ which is the residual in the representation of the k-th patch, using all atoms apart from the j_0-th. Thus, the task of the K-SVD stage is

$$\min_{\mathbf{a}_{j_0},\{\hat{\mathbf{q}}_k(j_0)\}_{k\in\Omega_{j_0}}} \sum_{k\in\Omega_{j_0}} \|\mathbf{M}_k(\mathbf{y}_k^{j_0}-\mathbf{a}_{j_0}\mathbf{q}_k(j_0))\|_2^2. \tag{6.9}$$

While the above could be solved as a rank-1 SVD approximation, a simpler approach[14] is a short iterative (2-3 iterations) process where we update the two unknowns alternatingly.

Fixing $\{\mathbf{q}_k(j_0)\}_{k\in\Omega_{j_0}}$ and taking the derivative $\frac{\partial}{\partial\mathbf{a}_{j_0}}$ of $Error(\mathbf{A})$, we obtain

$$\begin{aligned} \frac{\partial}{\partial\mathbf{a}_{j_0}} Error(\mathbf{A}) &= \sum_{k\in\Omega_{j_0}} \mathbf{M}_k^T\mathbf{M}_k\left(\mathbf{y}_k^{j_0}-\mathbf{a}_{j_0}\mathbf{q}_k(j_0)\right)\mathbf{q}_k(j_0) \\ &= \sum_{k\in\Omega_{j_0}} \mathbf{M}_k\mathbf{y}_k^{j_0}\mathbf{q}_k(j_0) - \sum_{k\in\Omega_{j_0}} \mathbf{M}_k\mathbf{q}_k(j_0)^2\mathbf{a}_{j_0}, \end{aligned} \tag{6.10}$$

where we used the well-known vector derivative formula and $\mathbf{M}_k^T\,\mathbf{M}_k = \mathbf{M}_k$. Setting the last term to zero, we obtain

$$\hat{\mathbf{a}}_{j_0} = \left[\sum_{k\in\Omega_{j_0}} \mathbf{M}_k\mathbf{q}_k(j_0)^2\right]^{-1} \sum_{k\in\Omega_{j_0}} \mathbf{M}_k\mathbf{y}_k^{j_0}\mathbf{q}_k(j_0) \tag{6.11}$$

[12]The matrix $\mathbf{M}_k$ could be omitted but it leads to faster convergence since the initialization $\mathbf{z} = \mathbf{M}^T\mathbf{y}$ totally corrupts the missing values.

[13]The local mask $\mathbf{M}_k$ makes it more challenging to write the error in a matrix form as in ($\mathcal{DL}'$).

[14]And more straightforward given that we did not provide a matrix form of the error.

and once computed, the atom $\hat{\mathbf{a}}_{j_0}$ should be **normalized**. Similarly as above the update of $\mathbf{q}_k(j_0)$ is attained by

$$\hat{\mathbf{q}}_k(j_0) = \left[\mathbf{a}_{j_0}^T \mathbf{M}_k \mathbf{a}_{j_0}\right]^{-1} \mathbf{a}_{j_0}^T \mathbf{M}_k \mathbf{y}_k^{j_0} \tag{6.12}$$

computed for all $k \in \Omega_{j_0}$.

After the *Main Iteration (K-SVD)* process the calculated representations $\{\hat{\mathbf{q}}_k\}_{k=1}^N$ and the matrix $\mathbf{A}$ can be fixed. Therefore the second term in (6.7) becomes constant thus the optimization we need to solve becomes

$$\hat{\mathbf{y}} = \arg\min_{\mathbf{z}} \lambda \|\mathbf{M}\mathbf{z} - \mathbf{y}\|_2^2 + \sum_k \|\mathbf{A}\hat{\mathbf{q}}_k - \mathbf{R}_k\mathbf{z}\|_2^2, \tag{6.13}$$

which is a simple quadratic term whose closed-form solution (attained by setting the derivative to zero) is

$$\hat{\mathbf{y}} = \left(\lambda \mathbf{M}^T\mathbf{M} + \sum_{k=1}^N \mathbf{R}_k^T\mathbf{R}_k\right)^{-1} \left(\lambda \mathbf{M}^T\mathbf{y} + \sum_{k=1}^N \mathbf{R}_k^T\mathbf{A}\hat{\mathbf{q}}_k\right). \tag{6.14}$$

This rather complex expression is misleading because the matrix to invert is diagonal and simply introduces a normalization factor per each pixel in the resulting image. In words, the expression does the following

- For missing pixels: suggests a simple averaging of the contributions of the recovered patches directly.
- For existing pixels: suggests an averaging which takes into account the measured values of $\mathbf{y}$ with a proper weight.

6.4 Experiments and Results

The image processing experiments were performed on four images (two standard and two non-standard) 8-bit gray-scale[15] images of size 256×256 pixels shown in Figure 6.1.

These images are contaminated by an additive white Gaussian noise with $\sigma = 20$ and then sub-sampled in some locations (first randomly then structurally), such that only a portion of the pixels remains.

Figure 6.1: The images *Lena*, *Peppers*, *Cat* and *Ship* on which the inpainting experiments are performed.

We treat the problem as a $(\mathcal{INP}^L_\epsilon)$ one, using Algorithm 6.2 for its solution. The

[15]The standard images are carefully selected for algorithm evaluation, see [1], 8-bit means that the values of the pixels are between 0 and 255.

inpainting operates on patches of size 8×8 pixels, extracted with **full-overlaps**[16] from the corrupted image.

The dictionary $\mathbf{A}$ is chosen to have 256 atoms, thus its complete size is 64×256. The *Sparse Coding* stage is performed using the OMP algorithm with the parameters $c = 1.1$ and $\lambda = 0$, the parameters are fixed according to [10] and [9, p.326-p.330]. The total number of iteration j_{max} is fixed to be 20 and the inpainted image selected, in each experiment, is the one with the least RMSE[17] error.

Figure 6.2 presents the inpainted images for the image *Peppers* for three different experiments (removing randomly 25%, 50% and 75% of the pixels) the locations of the missing pixels are stored in the degradation operator $\mathbf{M}$ (mask) which is considered known[18].

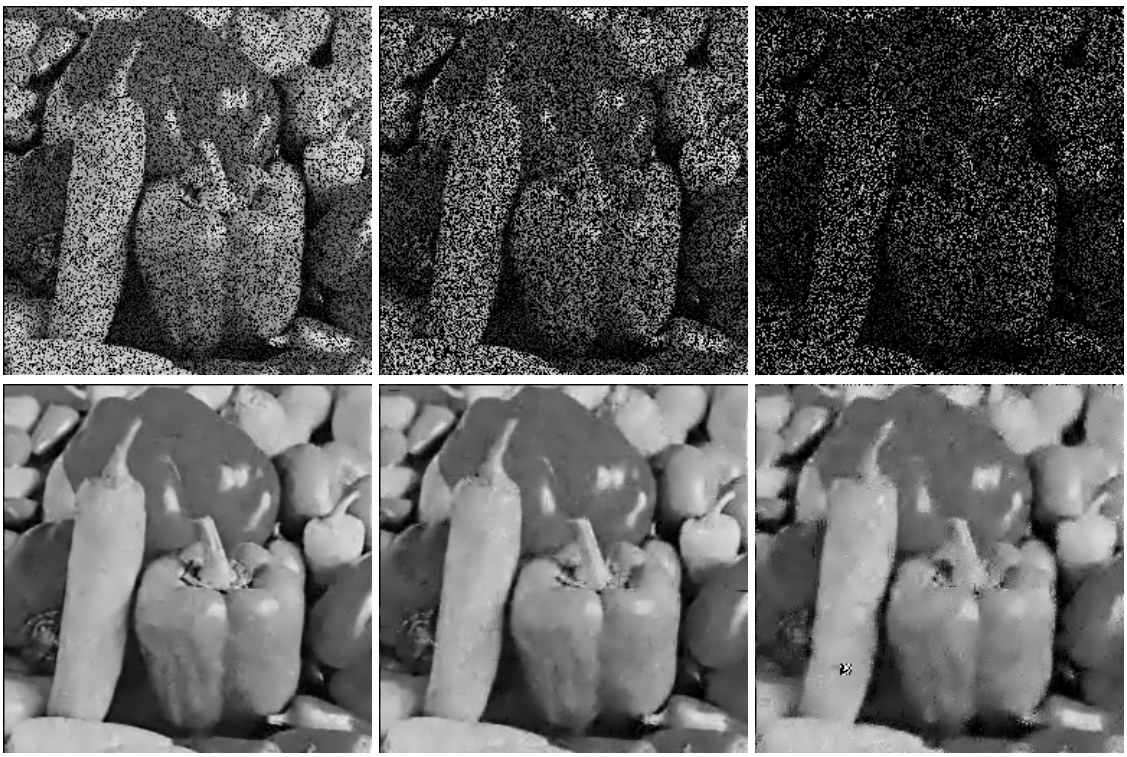

Figure 6.2: Inpainting results of the image *Peppers*. The figure shows the given images (top) and their inpainting results (bottom), for 25% missing pixels (left, RMSE= 8.14), 50% missing pixels (center, RMSE= 9.98) and 75% missing pixels (right, RMSE= 17.61).

Figure 6.3 presents the inpainted images for the images *Ship*, *Cat* and *Lena* when the missing pixels are not random but rather structural. The first four images from the left show the inpainting results when the mask is obtained from some text[19]. We can see

[16]The total number of patches is therefore $(256-8+1)^2 = 62,001$.

[17]RMSE simply means root mean squared error i.e. $\sqrt{\sum_{i=1}^{M}(\mathbf{y}_0^i - \hat{\mathbf{y}}^i)^2/M}$, where $\mathbf{y}_0$ is the original image, $\hat{\mathbf{y}}$ the recovered one and M the number of pixels in each of them.

[18]One of the assumptions of the $(\mathcal{INP}_\epsilon^L)$ problem.

[19]The preface of the book Convex Optimization by S. Boyd and L. Vandenberghe.

visually and from the RMSE values that the results are quite impressive, thus the method shows robustness even to structural perturbation.

The two images on the right, on the other hand, present the limitations of the method. In this case the mask is given by 64 blocks of size 8×8. It can be seen that the inpainted image presents a lot of visual constructs and that the blocks are only partially removed. This is expected because the blocks are quite big relatively to the patches, indeed this can be solved by using bigger patches or, even better, by using a method called *The Global MCA*, see [9, p.335-342].

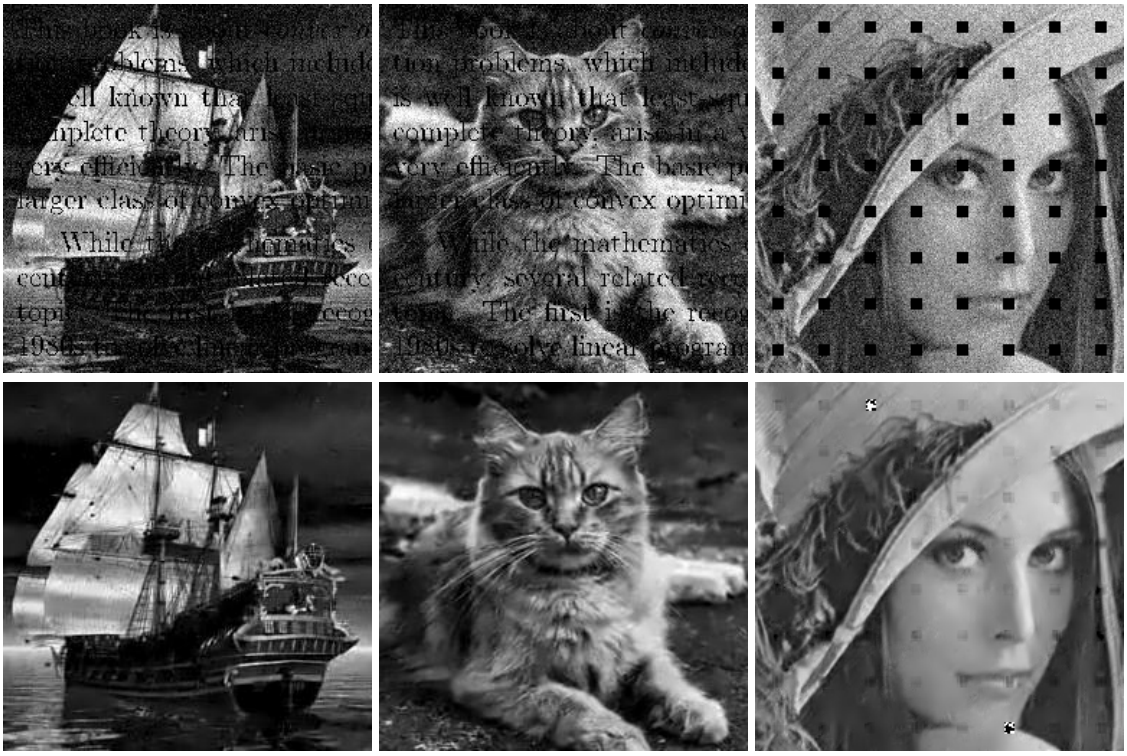

Figure 6.3: Inpainting results of the images *Ship*, *Cat* and *Lena*. The figure shows the given images (top) and their inpainting results (bottom), for text-Mask of *Ship* (left, RMSE= 10.90), text-Mask of *Cat* (center, RMSE= 10.50) and 8×8 blocks Mask of *Lena* (right, RMSE= 10.92).

Table 6.1 summarizes the results in terms of the RMSE for all the images. We can see that the method performs better for the standard gray-scale images *Lena* and *Peppers* even though visually the difference is not substantial (see appendix A).

Figure 6.4 shows the learning curves for the image *Peppers* with 50% missing pixels on the left and with 75% missing pixels on the right. The learning curve on the left shows convergence with a point of minimum at iteration 14 and for a RMSE value of 9.98. The learning curve on the right has a point of minimum at iteration 11 and for a RMSE value of 17.61.

Clear signs of *overfitting* are visible for the right learning curve after the point of minimum, this behaviour is somewhat expected because in that particular case we are training

Root Mean Squared Errors					
Image	25%	50%	75%	text	blocks
Peppers	8.14	9.98	17.61	7.85	9.76
Lena	8.36	9.92	14.56	7.92	10.92
Ship	11.75	14.99	22.16	10.90	13.55
Cat	11.12	13.01	18.95	10.50	12.79

Table 6.1: RMSE for Algorithm 6.2 with 20 iterations.

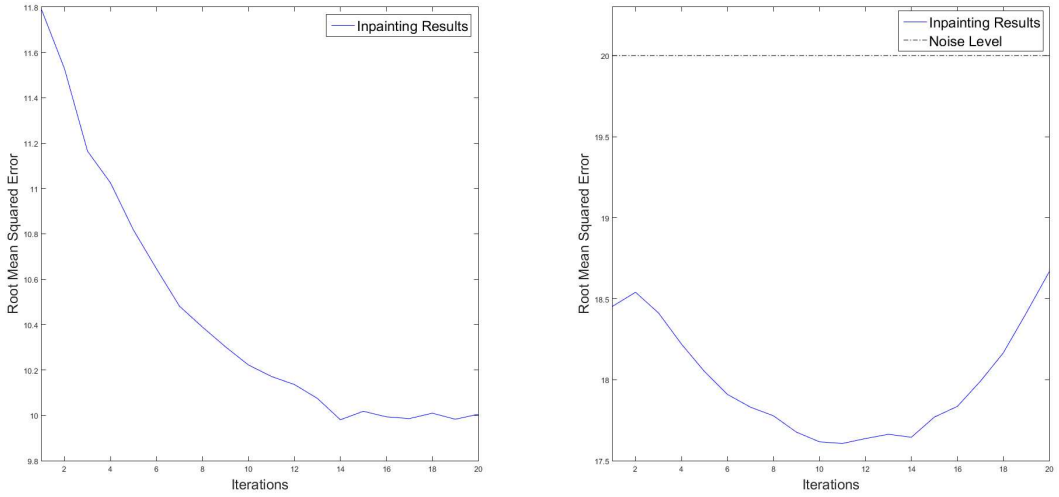

Figure 6.4: Learning curves of the RMSE values relative to the number of iterations. The left figure refers to the image *Peppers* with 50% missing pixels and the right to the same image with 75% missing pixels.

on an image with 75% missing pixels, making the number of parameters, relative to the data, higher and therefore increasing the ability of the method to overfit.

Lastly, it is important to remind that the images are contaminated by an additive white Gaussian noise with $\sigma = 20$, which means that the noise level[20] is around 20. Observing Table 6.1 it is clear that the method chosen has a strong denoising effect being able to obtain a RMSE smaller than 20 (in 3 out of 4 cases) even when the image is so corrupted that there are as many as 75% missing pixels.

[20]The RMSE at iteration 0.

Appendix A

Complete Inpainting Results

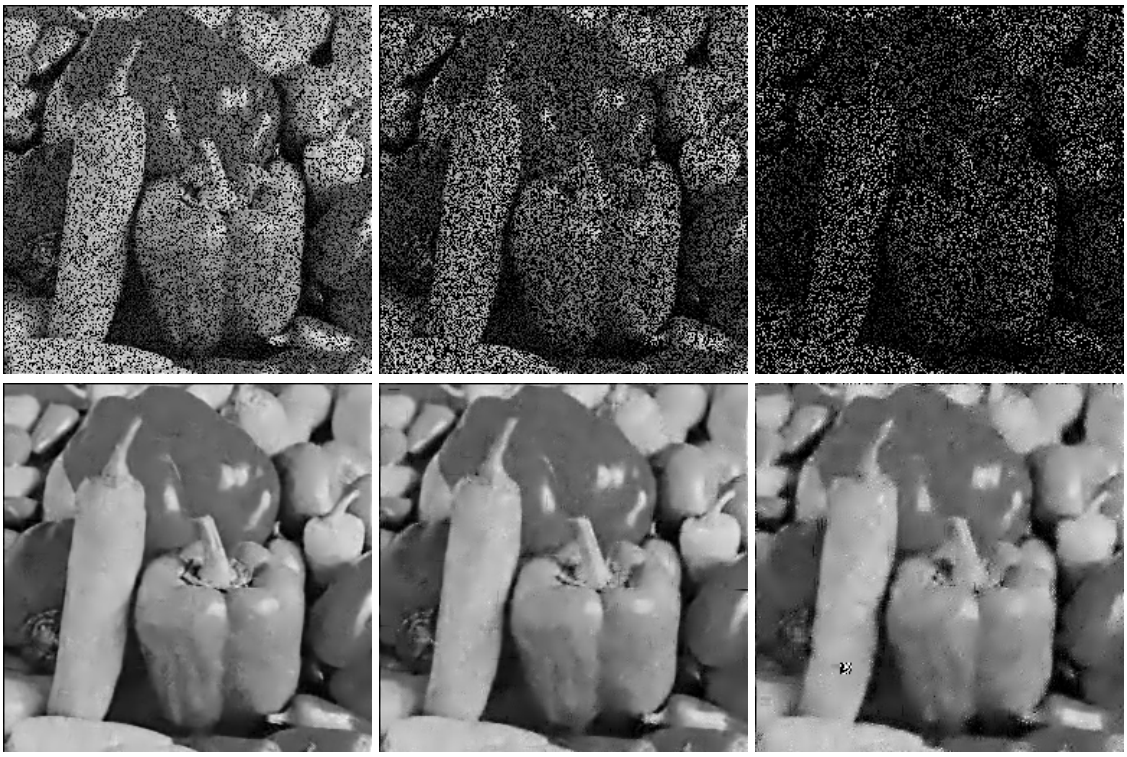

Figure A.1: Inpainting results of the image *Peppers*. The figure shows the given images (top) and their inpainting results (bottom), for 25% missing pixels (left), 50% missing pixels (center) and 75% missing pixels (right).

Figure A.2: Inpainting results of the image *Lena*. The figure shows the given images (top) and their inpainting results (bottom), for 25% missing pixels (left), 50% missing pixels (center) and 75% missing pixels (right).

Figure A.3: Inpainting results of the image *Cat*. The figure shows the given images (top) and their inpainting results (bottom), for 25% missing pixels (left), 50% missing pixels (center) and 75% missing pixels (right).

Figure A.4: Inpainting results of the image *Ship*. The figure shows the given images (top) and their inpainting results (bottom), for 25% missing pixels (left), 50% missing pixels (center) and 75% missing pixels (right).

Figure A.5: The figure shows the given images (top) and their inpainting results (bottom), for text-Mask of *Lena* (left), text-Mask of *Peppers* (center) and text-Mask of *Ship* (right).

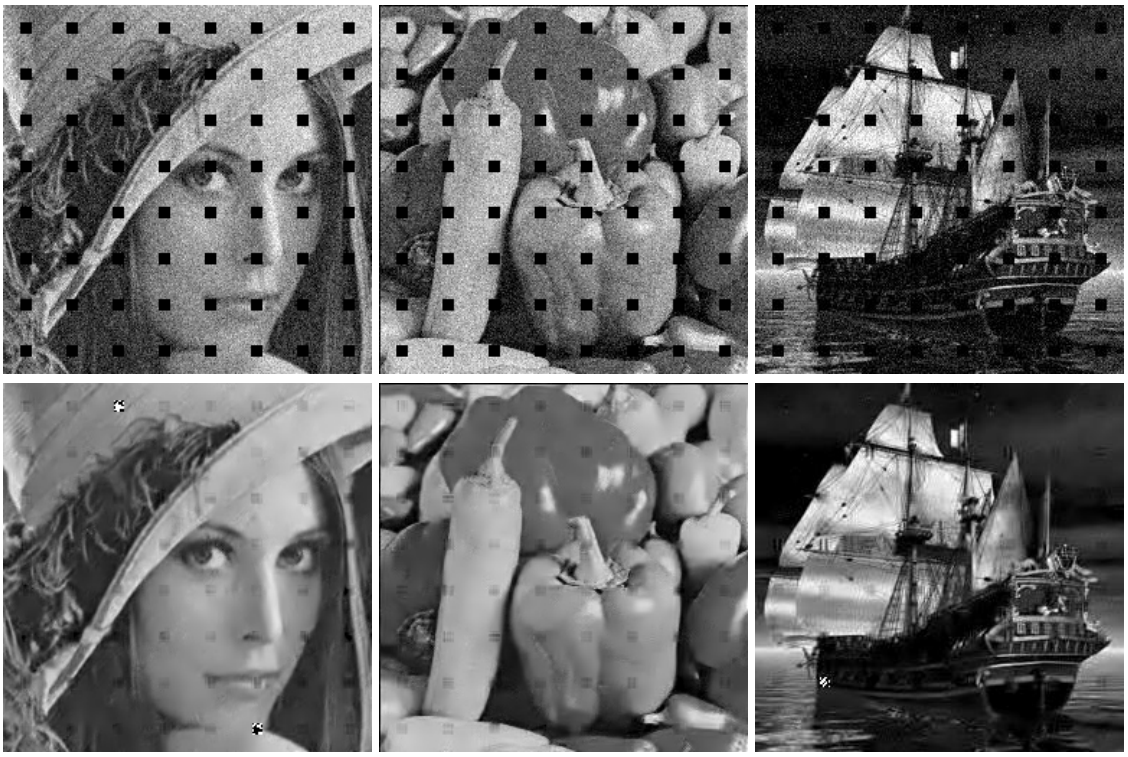

Figure A.6: The figure shows the given images (top) and their inpainting results (bottom), for 8×8 blocks Mask of *Lena* (left), 8×8 blocks Mask of *Peppers* (center) and 8×8 blocks Mask of *Ship* (right).

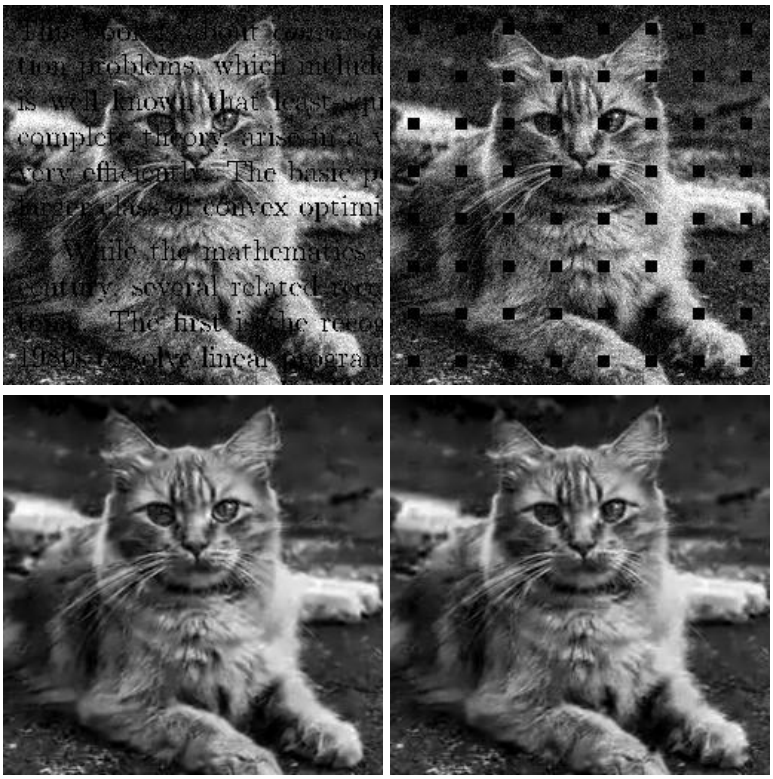

Figure A.7: Inpainting results of the image *Cat*. The figure shows the given images (top) and their inpainting results (bottom), for text-Mask (left), 8×8 blocks Mask (right)

Appendix B

Codes

The codes are taken from [18] and modified for the specific purposes of this work.

The following Matlab code generates the numerical simulations of section 4.4.

```
% ======================================================================
% In this program we test the performance of the LS-OMP, MP, OMP, WMP,
% Thresholding and BP algorithms by running them on a set of test
% signals and checking whether they provide the desired outcome
% (L2 error, support error and performance time are computed)
% ======================================================================
n=30; m=50; Smax=15; Exper=200;
A=randn(n,m);
W=sqrt(diag(A'*A));
for k=1:1:m,
    A(:,k)=A(:,k)/W(k);
end;
Er2=zeros(Smax,Exper,6);
ErS=zeros(Smax,Exper,6);
time=zeros(Smax,Exper,6);
for S=1:1:Smax,
    for experiment=1:1:Exper
        % Generate a test signal of cardinality S
        x=zeros(m,1);
        pos=randperm(m);
        % range [-1,1]
        x(pos(1:S))=sign(randn(S,1)).*rand(S,1);
        b=A*x;

        % Apply LS-OMP
        thrLSMP=1e-4;
        tic;
        r=b;
        SS=[];
        while r'*r>thrLSMP,
            Z=zeros(m,1);
            for jj=1:1:m
                SStemp=[SS,jj];
                rtemp=b-A(:,SStemp)*pinv(A(:,SStemp))*b;
                Z(jj)=rtemp'*rtemp;
            end;
            posZ=find(Z==min(Z),1);
            SS=sort([SS,posZ(1)]);
            r=b-A(:,SS)*pinv(A(:,SS))*b;
        end;
```

```
41          xLSMP=zeros(m,1);
42          xLSMP(SS)=pinv(A(:,SS))*b;
43          time(S,experiment,1)=double(toc);
44          Er2(S,experiment,1)=mean((xLSMP-x).^2)/mean(x.^2);
45          ErS(S,experiment,1)=(max(S,length(SS))-length(intersect(SS,pos(1:S))))/
                max(S,length(SS));
46
47          % Apply OMP
48          thrOMP=1e-4;
49          tic;
50          r=b;
51          SS=[];
52          while r'*r>thrOMP,
53              Z=abs(A'*r);
54              posZ=find(Z==max(Z));
55              SS=sort([SS,posZ(1)]);
56              r=b-A(:,SS)*pinv(A(:,SS))*b;
57          end;
58          xOMP=zeros(m,1);
59          xOMP(SS)=pinv(A(:,SS))*b;
60          time(S,experiment,2)=double(toc);
61          Er2(S,experiment,2)=mean((xOMP-x).^2)/mean(x.^2);
62          ErS(S,experiment,2)=(max(S,length(SS))-length(intersect(SS,pos(1:S))))/
                max(S,length(SS));
63
64          % Apply MP
65          thrMP=1e-4;
66          tic;
67          r=b;
68          xMP=zeros(m,1);
69          while r'*r>thrMP,
70              Z=abs(A'*r);
71              posZ=find(Z==max(Z),1);
72              xMP(posZ)=xMP(posZ)+A(:,posZ)'*r;
73              r=r-A(:,posZ)*A(:,posZ)'*r;
74          end;
75          SS=find(abs(xMP)>1e-8)';
76          time(S,experiment,3)=double(toc);
77          Er2(S,experiment,3)=mean((xMP-x).^2)/mean(x.^2);
78          ErS(S,experiment,3)=(max(S,length(SS))-length(intersect(SS,pos(1:S))))/
                max(S,length(SS));
79
80          % Apply WMP
81          thrWMP=1e-4; t=0.5;
82          tic;
83          r=b;
84          xWMP=zeros(m,1);
85          while r'*r>thrMP,
86              Z=abs(A'*r);
87              posZ=find(Z>=t*sqrt(r'*r),1);
88              if isempty(posZ)
89                  posZ=find(Z==max(Z),1);
90              end;
91              xWMP(posZ)=xWMP(posZ)+A(:,posZ)'*r;
92              r=r-A(:,posZ)*A(:,posZ)'*r;
93          end;
94          SS=find(abs(xWMP)>1e-8)';
95          time(S,experiment,4)=double(toc);
96          Er2(S,experiment,4)=mean((xWMP-x).^2)/mean(x.^2);
97          ErS(S,experiment,4)=(max(S,length(SS))- length(intersect(SS,pos(1:S))))/
                max(S,length(SS));
```

```
98
99
100           % Apply Thr
101           thrTH=1e-4;
102           tic;
103           Z=A'*b;
104           [Za,posZ]=sort(abs(Z),'descend');
105           r=b;
106           SS=[];
107           xTH=zeros(m,1);
108           while r'*r>thrTH,
109                SS=[SS,posZ(length(SS)+1)];
110                xTH=zeros(m,1);
111                xTH(SS)=pinv(A(:,SS))*b;
112                r=b-A(:,SS)*xTH(SS);
113           end;
114           SS=find(abs(xTH)>1e-8)';
115           time(S,experiment,5)=double(toc);
116           Er2(S,experiment,5)=min(mean((xTH-x).^2)/mean(x.^2),1);
117           ErS(S,experiment,5)=(max(S,length(SS))-length(intersect(SS,pos(1:S))))/
                  max(S,length(SS));
118
119           % BP using L1 by Matlab
120           V=ones(2*m,1);
121           tic;
122           xBP=linprog(V,[],[],[A, -A],b,0*V,V*100);
123           xBP=xBP(1:m)-xBP(m+1:end);
124           SS=find(abs(xBP)>1e-8)';
125           time(S,experiment,6)=double(toc);
126           Er2(S,experiment,6)=mean((xBP-x).^2)/mean(x.^2);
127           ErS(S,experiment,6)=(max(S,length(SS))-length(intersect(SS,pos(1:S))))/
                  max(S,length(SS));
128      end;
129 end;
```

The following Matlab code plots the graphs of section 4.4.

```
% L2 error
figure(1); clf;
h=plot(1:1:Smax,mean(Er2(:,:,1),2),'b'); hold on;
set(h,'LineWidth',2);
h=plot(1:1:Smax,mean(Er2(:,:,2),2),'r');
set(h,'LineWidth',2);
h=plot(1:1:Smax,mean(Er2(:,:,3),2),'g');
set(h,'LineWidth',2);
h=plot(1:1:Smax,mean(Er2(:,:,4),2),'c');
set(h,'LineWidth',2);
h=plot(1:1:Smax,mean(Er2(:,:,5),2),'m');
set(h,'LineWidth',2);
h=plot(1:1:Smax,mean(ErS(:,:,6),2),'k-.');
set(h,'LineWidth',2);
h=xlabel('Cardinality of the true solution'); set(h,'FontSize',14);
h=ylabel('Average and Relative L_2-Error'); set(h,'FontSize',14);
set(gca,'FontSize',14);
h=legend({'LS-OMP','OMP','MP','Weak-MP (t=0.5)','Thresholding','BP by Linear
    Prog.'},'Location','northwest',2);
axis tight;
% Support error
figure(2); clf;
h=plot(1:1:Smax,mean(ErS(:,:,1),2),'b'); hold on;
set(h,'LineWidth',2);
h=plot(1:1:Smax,mean(ErS(:,:,2),2),'r');
set(h,'LineWidth',2);
h=plot(1:1:Smax,mean(ErS(:,:,3),2),'g');
set(h,'LineWidth',2);
h=plot(1:1:Smax,mean(ErS(:,:,4),2),'c');
set(h,'LineWidth',2);
h=plot(1:1:Smax,mean(ErS(:,:,5),2),'m');
set(h,'LineWidth',2);
h=plot(1:1:Smax,mean(ErS(:,:,6),2),'k-.');
set(h,'LineWidth',2);
h=xlabel('Cardinality of the true solution'); set(h,'FontSize',14);
h=ylabel('Probability of Error in Support'); set(h,'FontSize',14);
set(gca,'FontSize',14);
h=legend({'LS-OMP','OMP','MP','Weak-MP (t=0.5)','Thresholding','BP by Linear
    Prog.'},'Location','northwest',2);
axis tight;
% Time execution
figure(3); clf;
h=plot(1:1:Smax,mean(time(:,:,1),2),'b'); hold on;
set(h,'LineWidth',2);
h=plot(1:1:Smax,mean(time(:,:,2),2),'r');
set(h,'LineWidth',2);
h=plot(1:1:Smax,mean(time(:,:,3),2),'g');
set(h,'LineWidth',2);
h=plot(1:1:Smax,mean(time(:,:,4),2),'c');
set(h,'LineWidth',2);
h=plot(1:1:Smax,mean(time(:,:,5),2),'m');
set(h,'LineWidth',2);
h=plot(1:1:Smax,mean(time(:,:,6),2),'k-.');
set(h,'LineWidth',2);
h=xlabel('Cardinality of the true solution'); set(h,'FontSize',14);
h=ylabel('Average time of execution'); set(h,'FontSize',14);
set(gca,'FontSize',14);
h=legend({'LS-OMP','OMP','MP','Weak-MP (t=0.5)','Thresholding','BP by Linear
    Prog.'},'Location','northwest',2);
axis tight;
```

The following Matlab code is the implementation of the inpainting problem with the local-KSVD algorithm described in chapter 6.

```
% ========================================================================
% This program implements the local-KSVD algorithm for image inpainting
% input arguments: filename - name of the image
%                  ratios - percentage of missing pixels
%                  iterTot - total number of iterations
% filename options: 'peppers256.png';'lena.png';'ship.jpg';'cat.jpg'
% ratios is a list of doubles
% ratio = 0    ... text missing
% ratio = 0.25 ... 25% missing
% ratio = 0.5  ... 50% missing
% ratio = 0.75 ... 75% missing
% ratio = 1 ... blocks missing
% ========================================================================
function [DictRes]=Chapter_06_KSVDinpainting(filename,ratios,iterTot)
% directory of images
cd 'C:\Users\User\Desktop\diplomski rad\diplomski'
bb=8; % block size
K=256; % number of atoms in the dictionary
N=256; % dimensions of image
[IMin0,pp]=imread(filename);
if strcmp(filename,'cat.jpg')
    IMin0 = rgb2gray(IMin0);
    IMin0 = imresize(IMin0, 0.4);
    IMin0=IMin0(10:265,90:345);
elseif strcmp(filename,'ship.jpg')
    IMin0 = rgb2gray(IMin0);
    IMin0 = imresize(IMin0, 0.35);
    IMin0=IMin0(10:265,120:375);
elseif strcmp(filename,'lena.png')
    IMin0=IMin0(148:403,128:383);
end
IMin0=im2double(IMin0);
IMin0 = IMin0*255;
if isempty(pp)
    figure(1); clf;
    imagesc(IMin0); axis image; axis off; colormap(gray(256));
else
    figure(1); clf;
    imagesc(IMin0); axis image; axis off; colormap(pp);
end
% Adding noise
sigma=20;
IMin=IMin0+sigma*randn(N);
% Extracting the noisy patches
blkMatrixIm=im2col(IMin,[bb,bb],'sliding');
for i = 1:size(ratios, 2)
    ratio = ratios(i);
    DictRes=Inp_KSVD(i*100,pp,bb,K,N,IMin,IMin0,blkMatrixIm,sigma,ratio,iterTot,
        filename);
end
return;

function [DictRes]=Inp_KSVD(i_fig,pp,bb,K,N,IMin,IMin0,blkMatrixIm,sigma,ratio,
    iterTot,filename)
text= sprintf('Inpainting results: %.2f missing pixels', ratio*100);
disp(text);
disp('================================');

```

```
% The initial dictionary
[Dict]=dict_DCT(bb,K);
% Creating the mask and extracting its patches
[Mask,blkMask]=mask_creation(N,ratio,bb);

txt_save=strcat(sprintf('%.2f',ratio*100),filename);
if isempty(pp)
    imwrite(IMin.*Mask,colormap(gray(256)),txt_save);
else
    imwrite(IMin.*Mask,colormap(pp),txt_save);
end
[Coeff,~]=initialization_and_first_step(Dict,blkMask,blkMatrixIm,sigma,IMin,
    IMin0);
[Result,Error_list,DictRes]=inpainting(Dict,blkMask,blkMatrixIm,Coeff,sigma,IMin
    ,IMin0,iterTot);
figure(i_fig); clf;
h=plot(1:1:iterTot,Error_list,'b'); hold on;
h=plot(1:1:iterTot,repmat(sigma,1,iterTot),'k-.');
h=xlabel('Iterations'); set(h,'FontSize',10);
h=ylabel('Root Mean Squared Error'); set(h,'FontSize',10);
h=legend({'Inpainting Results','Noise Level'},'Location','northeast',2);
xlim([1 iterTot])
ylim([0 30])
hold off;

txt_save=strcat(sprintf('recovered%.2f',ratio*100),filename);
if isempty(pp)
    imwrite(Result,colormap(gray(256)),txt_save);
else
    imwrite(Result,colormap(pp),txt_save);
end
return;

function [Dict]=dict_DCT(bb,K)
Dict=zeros(bb,sqrt(K));
for k=0:1:sqrt(K)-1,
    V=cos([0:1:bb-1]'*k*pi/sqrt(K));
    if k>0, V=V-mean(V); end;
    Dict(:,k+1)=V/norm(V);
end;
Dict=kron(Dict,Dict);
return;

function [Mask,blkMask]=mask_creation(N,ratio,bb)
if ratio==0
    Mask=imread('TextImage3.png');
    Mask=Mask(1:N,1:N);
    Mask=double(Mask(1:N,1:N)/255);
elseif ratio==1
    Mask=zeros(8,8);
    Mask=padarray(Mask,[12 12],1,'both');
    Mask2=ones(8,8);
    Mask=kron(Mask2,Mask);
else
    Pos=randperm(N^2);
    Mask=ones(N,N);
    Mask(Pos(1:N^2*ratio))=0;
end
blkMask=im2col(Mask,[bb,bb],'sliding');
return;

```

```
116 function [Coeff,Result]=initialization_and_first_step(Dict,blkMask,blkMatrixIm,
        sigma,IMin,IMin0)
117 % Inpainting the Patches
118 Coeff=OMPerrInpaint(Dict,blkMatrixIm.*blkMask,blkMask,sigma*1.1);
119 Result=RecoverImage(IMin,Dict,Coeff);
120 Result=max(min(Result,255),0);
121 disp(['Recovery error =',num2str(sqrt(mean(mean((Result-IMin0).^2))))]);
122 return;
123
124 function [Result,Error_list,DictRes]=inpainting(Dict,blkMask,blkMatrixIm,Coeff,
        sigma,IMin,IMin0,iterTot)
125
126 for KSVDiter=1:1:iterTot,
127     for atom=1:1:size(Dict,2)
128         Omega=find(abs(Coeff(atom,:))>0);
129         if isempty(Omega), continue; end; % this atom will be useless
130         CoeffM=Coeff;
131         CoeffM(atom,:)=0;
132         Err=blkMask(:,Omega).*(blkMatrixIm(:,Omega)-Dict*CoeffM(:,Omega));
133         for SVDiter=1:1:3
134             dd=diag(1./(blkMask(:,Omega)*(Coeff(atom,Omega).^2)'))*...
135                 (Err*Coeff(atom,Omega)');
136             dd=dd/norm(dd);
137             Dict(:,atom)=dd;
138             Coeff(atom,Omega)=(dd'*Err)./sum((blkMask(:,Omega).*...
139                     (dd*ones(1,length(Omega)))).^2,1);
140         end;
141     end;
142     Coeff=OMPerrInpaint(Dict,blkMatrixIm.*blkMask,blkMask,sigma*1.1);
143     Result=RecoverImage(IMin,Dict,Coeff);
144     Result=max(min(Result,255),0);
145     Result_tensor(:,:,KSVDiter)=Result;
146     Dict_tensor(:,:,KSVDiter)=Dict;
147     Error_list(KSVDiter)=sqrt(mean(mean((Result-IMin0).^2)));
148     disp(['Iteration number: ',num2str(KSVDiter),...
149             '     Recovery error =',num2str(sqrt(mean(mean((Result-IMin0).^2))))
                ]);
150 end;
151 [M,I]=min(Error_list);
152 disp(['Minimum recovery error:',sprintf('\n'),'Iteration number: ',num2str(I)
        ,...
153         '     Recovery error =',num2str(M)]);
154 Result=Result_tensor(:,:,I);
155 DictRes=Dict_tensor(:,:,I);
156 return;
157
158 function [A]=OMPerrInpaint(D,X,Mask,errorGoal)
159 % Sparse coding of a group of signals based on a given dictionary and
160 % specified representation error to get
161 % input arguments: D - the dictionary
162 %                  X - the signals to represent
163 %                  Mask - the missing pixels
164 %                  errorGoal - the maximal allowed representation error
165 % output arguments: A - sparse coefficient matrix
166 [~,P]=size(X);
167 [n,~]=size(D);
168 E2 = errorGoal^2*n;
169 maxNumCoef = n/2;
170 A = sparse(size(D,2),size(X,2));
171 h=waitbar(0,'OMP on each example ...');
172 for k=1:1:P,
```

```
173         waitbar(k/P);
174         Mpos=find(Mask(:,k));
175         E2M=E2*length(Mpos)/n;
176         Dict=D(Mpos,:);
177         W=1./sqrt(diag(Dict'*Dict));
178         Dict=Dict*diag(W);
179         x=X(Mpos,k);
180         residual=x;
181         indx = [];
182         a = [];
183         currResNorm2 = sum(residual.^2);
184         j = 0;
185         while currResNorm2>E2M && j < maxNumCoef,
186             j = j+1;
187             proj=Dict'*residual;
188             pos=find(abs(proj)==max(abs(proj)));
189             pos=pos(1);
190             indx(j)=pos;
191             a=pinv(Dict(:,indx(1:j)))*x;
192             residual=x-Dict(:,indx(1:j))*a;
193             currResNorm2=sum(residual.^2);
194         end;
195         if (~isempty(indx))
196             A(indx,k)=a;
197             A(:,k)=W.*A(:,k);
198         end
199     end;
200     close(h);
201     return;
202     % Last step of KSVD
203     function [yout]=RecoverImage(y,D,CoefMatrix)
204     N=size(y,1);
205     n=sqrt(size(D,1));
206     yout=zeros(N,N);
207     Weight=zeros(N,N);
208     i=1; j=1;
209     for k=1:1:(N-n+1)^2,
210         patch=reshape(D*CoefMatrix(:,k),[n,n]);
211         yout(i:i+n-1,j:j+n-1)=yout(i:i+n-1,j:j+n-1)+patch;
212         Weight(i:i+n-1,j:j+n-1)=Weight(i:i+n-1,j:j+n-1)+1;
213         if i<N-n+1
214             i=i+1;
215         else
216             i=1; j=j+1;
217         end;
218     end;
219     yout=yout./Weight;
220     return;
```

Bibliography

[1] URL: https://homepages.cae.wisc.edu/~ece533/images/.

[2] M. Aharon, M. Elad, and A. M. Bruckstein. "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation". In: *IEEE Transactions on Signal Processing* 54.11 (2006), pp. 4311–4322. DOI: 10.1109/tsp.2006.881199.

[3] M. Aharon, M. Elad, and A. M. Bruckstein. "On the uniqueness of overcomplete dictionaries, and a practical way to retrieve them". In: *Linear Algebra and its Applications* 416.1 (2006), pp. 48–67. DOI: 10.1016/j.laa.2005.06.035.

[4] E.J. Candes and T. Tao. "Decoding by Linear Programming". In: *IEEE Transactions on Information Theory* 51.12 (2005), pp. 4203–4215. DOI: 10.1109/tit.2005.858979.

[5] *Discrete cosine transform*. May 2020. URL: https://en.wikipedia.org/wiki/Discrete_cosine_transform#DCT-II.

[6] D. L. Donoho and M. Elad. "Optimally sparse representation in general (nonorthogonal) dictionaries via 1 minimization". In: *Proceedings of the National Academy of Sciences* 100.5 (2003), pp. 2197–2202. DOI: 10.1073/pnas.0437847100.

[7] D. L. Donoho and P. B. Stark. "Uncertainty Principles and Signal Recovery". In: *SIAM Journal on Applied Mathematics* 49.3 (1989), pp. 906–931. DOI: 10.1137/0149053.

[8] D.L. Donoho, M. Elad, and V.N. Temlyakov. "Stable recovery of sparse overcomplete representations in the presence of noise". In: *IEEE Transactions on Information Theory* 52.1 (2006), pp. 6–18. DOI: 10.1109/tit.2005.860430.

[9] M. Elad. *Sparse and redundant representations*. Springer, 2010.

[10] M. Elad and M. Aharon. "Image Denoising Via Sparse and Redundant Representations Over Learned Dictionaries". In: *IEEE Transactions on Image Processing* 15.12 (2006), pp. 3736–3745. DOI: 10.1109/tip.2006.881969.

[11] M. Elad and A.M. Bruckstein. "A generalized uncertainty principle and sparse representation in pairs of bases". In: *IEEE Transactions on Information Theory* 48.9 (2002), pp. 2558–2567. DOI: 10.1109/tit.2002.801410.

[12] L. Eldén. "Matrix Methods in Data Mining and Pattern Recognition". In: (2007). DOI: 10.1137/1.9780898718867.

[13] A. Feuer and A. Nemirovski. “On sparse representation in pairs of bases”. In: *IEEE Transactions on Information Theory* 49.6 (2003), pp. 1579–1581. DOI: 10.1109/tit.2003.811926.

[14] I.F. Gorodnitsky and B.D. Rao. “Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm”. In: *IEEE Transactions on Signal Processing* 45.3 (1997), pp. 600–616. DOI: 10.1109/78.558475.

[15] R.A. Horn and C.R. Johnson. *Matrix analysis*. Cambridge University Press, 2017.

[16] S.G. Mallat and Z. Zhang. “Matching pursuits with time-frequency dictionaries”. In: *IEEE Transactions on Signal Processing* 41.12 (1993), pp. 3397–3415. DOI: 10.1109/78.258082.

[17] *Rayleigh quotient*. Mar. 2020. URL: https://en.wikipedia.org/wiki/Rayleigh_quotient.

[18] *Software*. URL: https://elad.cs.technion.ac.il/software/?pn=273.

[19] R.S. Varga. *Geršgorin and his circles*. Springer, 2004.

Sažetak

Naslov i tema ovog rada je analiza algoritama za rijetku reprezentaciju signala. Rad uglavnom prati knjigu *Sparse and redundant representations* od Michaela Elada i podijeljen je u pet dijelova.

U prvom dijelu, koji se sastoji od prvog poglavlja, uvodi se problem ($\mathcal{P}_0$) i njemu slični problemi te opisuje zašto je dobro utemeljen.

U drugom dijelu, koji se sastoji od drugog poglavlja, izlažu se i dokazuju dovoljni uvjeti za koje je rješenje problema ($\mathcal{P}_0$), za specijalni dvo-ortogonalni te općeniti slučaj, jedinstveno i optimalno. U tom postupku uvedena su dva pojma vezana za matricu $\mathbf{A}$: *Iskra* i međusobna-usklađenost.

U trećem dijelu, koji se sastoji od trećeg i četvrtog poglavlja, opisuje se šest različitih algoritama za rješavanje ($\mathcal{P}_0$) problema koji su podijeljeni u dvije glavne kategorije: pohlepni i konveksno relaksacijski. Četvrto poglavlje je posvećeno teorijskim aspektima tih algoritama.

U četvrtom dijelu, koji se sastoji od petog poglavlja, uvodi se problem ($\mathcal{P}_0^\epsilon$) koji je robusnija verzija od ($\mathcal{P}_0$) i zato prikladniji za primjene. Za novouvedeni problem jedinstvenost (koja više ne vrijedi) je zamijenjena sa pojmom stabilnosti. Elegantni rezultat stabilnosti od ($\mathcal{P}_0^\epsilon$) je dokazan korištenjem pojma *Ograničenog Izometrijskog Svojstva* matrice.

U petom dijelu, koji se sastoji od šestog poglavlja, uvodi se generirajući model pod nazivom *Sparseland* i pokazuje se snažna primjena jedne verzije od ($\mathcal{P}_0^\epsilon$) za rekonstrukciju oštećenih slika. Korišten je tzv. K-SVD algoritam čiji su rezultati bili na *State-of-the-art* razini na dan publikacije 2006. godine. Eksperimenti su detaljno opisani u zadnjoj sekciji te potvrđuju odlične rezultate i pokazuju puno potencijala za daljni razvoj.

Summary

The topic of this thesis is sparse representations of signals. The thesis follows mainly the book *Sparse and redundant representations* by Michael Elad and is divided into five parts.

The first part, consisting of chapter one, introduces the problem ($\mathcal{P}_0$) and related and describes the reason why is well founded.

The second part, consisting of chapter two, presents the conditions under which the solution to the problem ($\mathcal{P}_0$), for the special two-orthogonal and general case, exhibits uniqueness and optimality. In doing so two special measures of a matrix $\mathbf{A}$ are introduced: the *spark* and the mutual-coherence.

The third part, consisting of chapters three and four, describes six different algorithms for solving ($\mathcal{P}_0$) which are divided into two main categories: greedy and convex relaxational. Chapter four deals with theoretical guarantees of those algorithms.

The fourth part, consisting of chapter five, introduces the ($\mathcal{P}_0^\epsilon$) problem which is a more robust version of ($\mathcal{P}_0$) and thus better for applications. For this new problem uniqueness (which cannot be claimed) is replaced by stability of the solution. The elegant result of the stability of ($\mathcal{P}_0^\epsilon$) is proved using the subtle concept of *Restricted Isometry Property* of a matrix.

The fifth part, consisting of chapter six, introduces the generative model named *Sparseland* and presents a powerful application of a version of ($\mathcal{P}_0^\epsilon$) to image inpainting. The algorithm used is called K-SVD and it achieved *State-of-the-art* performance by the time of first publication in 2006. The experiments are presented in the last section showing excellent results and great potential for future work.

Životopis

Rođen sam 23. rujna 1994. u Zagrebu. Godine 2001. preselio sam se s obitelji u Italiju, u Milano, gdje sam završio osnovnu školu (Istituto Comprensivo "Ilaria Alpi") i prirodoslovno-matematičku gimnaziju (Liceo Scientifico "Elio Vittorini"). Tijekom gimnazijskih dana doživio sam pomak u interesima s društvenih na prirodne znanosti te sam od 3. razreda sudjelovao na brojnim natjecanjima iz matematike i fizike s nekoliko visokih plasmana na školskoj i županijskoj razini a posebno bih istaknuo 10. mjesto na državnom natjecanju iz matematike u 4. razredu.

Godine 2014. vratio sam se u Zagreb i upisao preddiplomski studij Matematike na Matematičkom odsjeku Prirodoslovno-matematičkog fakulteta. Titulu sveučilišnog prvostupnika stekao sam 2017. godine te upisao diplomski studij *Matematička Statistika* na istom fakultetu. U listopadu 2019. počeo sam raditi kao vanjski suradnik na Zagrebačkoj školi ekonomije i managementa.