

Michael Elad

# Sparse and Redundant Representations

From Theory to Applications  
in Signal and Image Processing

 Springer

# **Sparse and Redundant Representations**



# **Sparse and Redundant Representations**

From Theory to Applications in  
Signal and Image Processing

**Michael Elad**

*The Technion – Israel Institute of Technology*

*Haifa, Israel*



**Springer**

Michael Elad  
The Technion – Israel Institute of Technology  
Computer Science Department  
32000 Haifa, Israel  
[elad@cs.technion.ac.il](mailto:elad@cs.technion.ac.il)

ISBN 978-1-4419-7010-7 e-ISBN 978-1-4419-7011-4  
DOI 10.1007/978-1-4419-7011-4  
Springer New York Dordrecht Heidelberg London

Library of Congress Control Number: 2010933513

Mathematics Subject Classification (2010): 94A12, 62H35, 62M40, 68U10, 94A08, 94H60, 46N10

© Springer Science+Business Media, LLC 2010

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer Science+Business Media, LLC, 233 Spring Street, New York, NY 10013, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

Printed on acid-free paper

Springer is part of Springer Science+Business Media ([www.springer.com](http://www.springer.com))

*To my loving and devoted family,  
My parents, Rita and Shmuel Elbaz, who  
have shown me the way,  
My children Sharon, Adar, and Yuval, who  
give purpose to this way, and  
My wife, Ritta, who constantly paves this  
way with love and understanding.*



# Foreword

A long long time ago, echoing philosophical and aesthetic principles that existed since antiquity, William of Ockham enounced the principle of parsimony, better known today as Ockham's razor: "Entities should not be multiplied without necessity." This principle enabled scientists to select the "best" physical laws and theories to explain the workings of the Universe and continued to guide scientific research, leading to beautiful results like the minimal description length approach to statistical inference and the related Kolmogorov complexity approach to pattern recognition.

However, notions of complexity and description length are subjective concepts and depend on the language "spoken" when presenting ideas and results. The field of sparse representations, that recently underwent a Big-Bang-like expansion, explicitly deals with the Yin-Yang interplay between the parsimony of descriptions and the "language" or "dictionary" used in them, and it became an extremely exciting area of investigation. It already yielded a rich crop of mathematically pleasing, deep and beautiful results that quickly translated into a wealth of practical engineering applications.

You are holding in your hands the first guide-book to Sparseland, and I am sure you'll find in it both familiar and new landscapes to see and admire, as well as excellent pointers that will help you find further valuable treasures. Enjoy the journey to Sparseland!

Haifa, Israel, December 2009

*Alfred M. Bruckstein*





# Preface

This book was originally written to serve as the material for an advanced one-semester (fourteen 2-hour lectures) graduate course for engineering students at the Technion, Israel. It is originally based on a review paper that I wrote together with Alfred M. Bruckstein and David L. Donoho, which appeared in *SIAM Review* on February 2009. The content of this paper has been massively modified and extended to become an appropriate material for an advanced graduate course. In this book I introduce the topic of sparse and redundant representations, show the questions posed in this field, and the answers they get, present the flavor of the research in this arena, the people involved, and more. It is my desire to expose the readers of this book (and the students in the course) to the beauty of this field, and the potential it encompasses for various disciplines.

So, what is it all about? Mathematicians active in this field (and there are many of those) would mention the deep relation between this emerging field and harmonic analysis, approximation theory, and wavelets, in their answer. However, this is not my answer. As an engineer, my interest is more in the practical sides of this field, and thus my entire motivation for studying sparse and redundant representations comes from the applications they serve in signal and image processing. This is not to say that I find little interest in the theoretical sides of the activity in sparse and redundant representations. However, these theoretical results should be considered in the context of the wider picture.

As I see it, this field is all about one specific mathematical model for signal sources. Modeling sources is key in signal and image processing. Armed with a proper model, one can handle various tasks such as denoising, restoration, separation, interpolation and extrapolation, compression, sampling, analysis and synthesis, detection, recognition, and more. Indeed, a careful study of the signal and image processing literature reveals that there is an evolution of such models and their use in applications. This course is about one such model, which I call *Sparse-Land* for brevity. This specific model is intriguing and fascinating because of the beauty of its theoretical foundations, the superior performance it leads to in various applications, its universality and flexibility in serving various data sources, and its unified view, which makes all the above signal processing tasks clear and simple.

At the heart of this model lies a simple linear system of equations, the kind of which seems long studied in linear algebra. 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. We are interested in seeking its sparsest solution, i.e., the one with the fewest nonzero entries. Can such a solution ever be unique? If so, when? How can such a solution be found in reasonable time? It is hard to believe, but these questions and more of the like form the core engine to this model and the wide field that has grown around it. Positive and constructive answers to these questions in recent years expose a wide variety of surprising phenomena, and those set the stage for *Sparse-Land* to serve in practice. Clearly, research in this field calls for an intensive use of knowledge from linear algebra, optimization, scientific computing, and more.

This field is relatively young. Early signs of its core ideas appeared in a pioneering work by Stephane Mallat and Zhifeng Zhang in 1993, with the introduction of the concept of dictionaries, replacing the more traditional and critically-sampled wavelet transform. Their work put forward some of the core ideas that later became central in this field, such as a greedy pursuit technique that approximates a sparse solution to an underdetermined linear system of equations, characterization of dictionaries by their coherence measure, and more.

A second key contribution in 1995 was by Scott Shaobing Chen, David Donoho, and Michael Saunders, who introduced another pursuit technique that uses the  $\ell_1$ -norm for evaluating sparsity. Surprisingly, they have shown that the quest for the sparsest solution could be tackled as a convex programming task, often leading to the proper solution.

With these two contributions, the stage was set for a deeper analysis of these algorithms and their deployment in applications. A crucial step towards this goal was made in 2001, with the publication of the work by Donoho and Huo. In their daring paper, Donoho and Huo defined and (partly) answered what later became a key question in this field: Can one guarantee the success of a pursuit technique? Under what conditions? This line of analysis later became the skeleton of this field, providing the necessary theoretical backbone for the *Sparse-Land* model. It is amazing to see how fast and how vast this area of research has grown in recent years, with hundreds of interested researchers, various workshops, sessions, and conferences, and an exponentially growing number of papers.

The activity in this field is spread over all major universities and research organizations around the world, with leading scientists from various disciplines. As this field of research lies at the intersection between signal processing and applied mathematics, active in this field are mathematicians interested in approximation theory, applied mathematicians interested in harmonic analysis, statisticians, and engineering from various fields (computer science, electrical engineering, geophysics, and more).

A little bit about myself: I started my activity in this area after a short visit by David Donoho to the Technion, where he gave a talk describing the results of the above-mentioned paper. I am ashamed to admit that I did not attend the talk! Nevertheless, my good friend and mentor, Freddy Bruckstein, was there to appreciate

the opportunity. Freddy's sixth sense for important research directions is already famous at the Technion, and did not fail him this time too. Freddy insisted that we work on improving the results that Donoho presented, and few months later we had our first result in this field. This result led to a joint work with Donoho during a post-doc period at Stanford, and I have found myself deeply involved in research on these topics ever since.

My origins as an engineer, and the fact that I worked with a leading mathematician, give me a unique perspective which I bring to this book. I present a coherent, well-structured, and flowing story that includes some of the theoretical foundations to sparse and redundant representations, numerical tools and algorithm for actual use, and applications in signal and image processing that benefit from these. I should stress that I do not pretend to give a well-balanced view of this entire field, and instead, I am giving *my own point-of-view*. In particular, I am not covering all the accumulated knowledge in this book because: (i) it is impossible to mention everything; (ii) not all details are important to the story I have to tell; (iii) new results in this field are emerging on a daily basis, making it impossible to give a completely updated view; and (iv) I have to admit that I do not understand all the results in this field.

*Compressed-Sensing* is a recent branch that separated from sparse and redundant representations, becoming a center of interest of its own. Exploiting sparse representation of signals, their sampling can be made far more effective compared to the classical Nyquist-Shannon sampling. In a recent work that emerged in 2006 by Emmanuel Candes, Justin Romberg, Terence Tao, David Donoho, and others that followed, the theory and practice of this field were beautifully formed, sweeping many researchers and practitioners in excitement. The impact this field has is immense, strengthened by the warm hug by information-theorists, leading mathematicians, and others. So popular has this field become that many confuse it with being the complete story of sparse representation modeling. In this book I discuss the branch of activity on *Compressed-Sensing* very briefly, and mostly so as to tie it to the more general results known in sparse representation theory. I believe that the accumulated knowledge on compressed-sensing could easily fill a separate book.

The CS Department, the Technion, Israel  
September 2009

*Michael Elad*



# Acknowledgments

This book would not have been written without the encouragement and support from the many friends and colleagues that surround me. First and foremost, I am grateful to Freddy Bruckstein, David Donoho, and Gene Golub, my mentors, teachers, and friends. It is their encouragement and guidance that brought me thus far. Gene passed away 3 years ago (November 16th, 2007), but he is with me all the time.

This book is mostly built on the accumulated knowledge that I gathered in the past decade in this fascinating field of sparse and redundant representation modeling. This vast work was possible because of the excellent researchers I collaborated with, my students, friends, and colleagues.

I owe a lot to my MSc and PhD students, Amir Adler, Michal Aharon, Zvika Ben-Haim, Raviv Brueler, Ori Bryt, Dima Datsenko, Tomer Factor, Raja Giryes, Einat Kidron, Boaz Matalon, Matan Protter, Svetlana Raboy, Ron Rubinstein, Dana Segev, Neta Shoham, Yossi Shtok, Javier Turek, and Roman Zeyde. I am grateful for the experience of working with them, teaching and guiding them, and learning so much from them in this process.

I am thankful to the colleagues and friends that were willing to work with me and guide me, forgiving my shortcomings and being so patient with me. These include Yonina Eldar, Arie Feuer, Mario Figueiredo, Yakov Hel-Or, Ron Kimmel, Yi Ma, Julien Mairal, Stephane Mallat, Peyman Milanfar, Guillermo Sapiro, Yoav Schechner, Jean-Luc Starck, Volodya Temlyakov, Irad Yavneh, and Michael Zibulevsky. I would like to express my gratitude to my friends and colleagues, Remi Gribonval, Jalal Fadili, Ron Kimmel, Miki Lustig, Gabriel Peyre, Doron Shaked, Joel Tropp, and Pierre Vandergheynst, for insightful discussions throughout the years, and their long-standing support.

Finally, special thanks are in order to Michael Bronstein, who designed the book cover, and to Allan Pinkus, who proof-read portions of this book, and made various valuable suggestions, which improved it.

More broadly, and without mentioning additional names, I should add that working in this field means that I am part of well-formed research community. I have been very lucky to belong to this diverse, vivid, ever growing, and warm group of researchers, which make my working experience such a pleasure.



# Contents

## Part I Sparse and Redundant Representations – Theoretical and Numerical Foundations

<b>1</b>	<b>Prologue</b>	3
1.1	Underdetermined Linear Systems	3
1.2	Regularization	4
1.3	The Temptation of Convexity	5
1.4	A Closer Look at $\ell_1$ Minimization	6
1.5	Conversion of $(P_1)$ to Linear Programming	8
1.6	Promoting Sparse Solutions	8
1.7	The $\ell_0$ -Norm and Implications	12
1.8	The $(P_0)$ Problem – Our Main Interest	13
1.9	The Signal Processing Perspective	14
	Further Reading	14
<b>2</b>	<b>Uniqueness and Uncertainty</b>	17
2.1	Treating the Two-Ortho Case	17
2.1.1	An Uncertainty Principle	18
2.1.2	Uncertainty of Redundant Solutions	21
2.1.3	From Uncertainty to Uniqueness	23
2.2	Uniqueness Analysis for the General Case	23
2.2.1	Uniqueness via the Spark	23
2.2.2	Uniqueness via the Mutual-Coherence	25
2.2.3	Uniqueness via the Babel Function	27
2.2.4	Upper-Bounding the Spark	28
2.3	Constructing Grassmannian Matrices	29
2.4	Summary	30
	Further Reading	31



<b>3</b>	<b>Pursuit Algorithms – Practice</b>	35
3.1	Greedy Algorithms	35
3.1.1	The Core Idea	35
3.1.2	The Orthogonal-Matching-Pursuit	36
3.1.3	Other Greedy Methods	39
3.1.4	Normalization	41
3.1.5	Rate of Decay of the Residual in Greedy Methods	43
3.1.6	Thresholding Algorithm	45
3.1.7	Numerical Demonstration of Greedy Algorithms	46
3.2	Convex Relaxation Techniques	48
3.2.1	Relaxation of the $\ell_0$ -Norm	48
3.2.2	Numerical Algorithms for Solving $(P_1)$	51
3.2.3	Numerical Demonstration of Relaxation Methods	51
3.3	Summary	52
	Further Reading	53
<b>4</b>	<b>Pursuit Algorithms – Guarantees</b>	55
4.1	Back to the Two-Ortho Case	55
4.1.1	OMP Performance Guarantee	55
4.1.2	BP Performance Guarantee	58
4.2	The General Case	64
4.2.1	OMP Performance Guarantee	65
4.2.2	Thresholding Performance Guarantee	67
4.2.3	BP Performance Guarantee	68
4.2.4	Performance of Pursuit Algorithms – Summary	71
4.3	The Role of the Sign-Pattern	71
4.4	Tropp’s Exact Recovery Condition	73
4.5	Summary	76
	Further Reading	76
<b>5</b>	<b>From Exact to Approximate Solutions</b>	79
5.1	General Motivation	79
5.2	Stability of the Sparsest Solution	80
5.2.1	Uniqueness versus Stability – Gaining Intuition	80
5.2.2	Theoretical Study of the Stability of $(P_0^\epsilon)$	82
5.2.3	The RIP and Its Use for Stability Analysis	86
5.3	Pursuit Algorithms	89
5.3.1	OMP and BP Extensions	89
5.3.2	Iteratively-Reweighted-Least-Squares (IRLS)	91
5.3.3	The LARS Algorithm	95
5.3.4	Quality of Approximations Obtained	98
5.4	The Unitary Case	101
5.5	Performance of Pursuit Algorithms	103
5.5.1	BPDN Stability Guarantee	103
5.5.2	Thresholding Stability Guarantee	104

5.6	Summary .....	107
	Further Reading .....	108
<b>6</b>	<b>Iterative-Shrinkage Algorithms .....</b>	<b>111</b>
6.1	Background .....	111
6.2	The Unitary Case - A Source of Inspiration .....	112
6.2.1	Shrinkage For the Unitary case .....	112
6.2.2	The BCR Algorithm and Variations .....	113
6.3	Developing Iterative-Shrinkage Algorithms .....	115
6.3.1	Surrogate Functions and the Prox Method .....	115
6.3.2	EM and Bound-Optimization Approaches .....	117
6.3.3	An IRLS-Based Shrinkage Algorithm .....	119
6.3.4	The Parallel-Coordinate-Descent (PCD) Algorithm .....	120
6.3.5	StOMP: A Variation on Greedy Methods .....	123
6.3.6	Bottom Line – Iterative-Shrinkage Algorithms .....	125
6.4	Acceleration Using Line-Search and SESOP .....	127
6.5	Iterative-Shrinkage Algorithms: Tests .....	127
6.6	Summary .....	132
	Further Reading .....	134
<b>7</b>	<b>Towards Average Performance Analysis .....</b>	<b>137</b>
7.1	Empirical Evidence Revisited .....	137
7.2	A Glimpse into Probabilistic Analysis .....	140
7.2.1	The Analysis Goals .....	140
7.2.2	Two-Ortho Analysis by Candes & Romberg .....	141
7.2.3	Probabilistic Uniqueness .....	143
7.2.4	Donoho's Analysis .....	143
7.2.5	Summary .....	144
7.3	Average Performance of Thresholding .....	144
7.3.1	Preliminaries .....	144
7.3.2	The Analysis .....	145
7.3.3	Discussion .....	148
7.4	Summary .....	150
	Further Reading .....	150
<b>8</b>	<b>The Dantzig-Selector Algorithm .....</b>	<b>153</b>
8.1	Dantzig-Selector versus Basis-Pursuit .....	153
8.2	The Unitary Case .....	155
8.3	Revisiting the Restricted Isometry Machinery .....	156
8.4	Dantzig-Selector Performance Guaranty .....	157
8.5	Dantzig-Selector in Practice .....	163
8.6	Summary .....	164
	Further Reading .....	165

## Part II From Theory to Practice – Signal and Image Processing Applications

<b>9</b>	<b>Sparsity-Seeking Methods in Signal Processing</b>	169
9.1	Priors and Transforms for Signals	169
9.2	The Sparse-Land Model	172
9.3	Geometric Interpretation of Sparse-Land	173
9.4	Processing of Sparsely-Generated Signals	176
9.5	Analysis Versus Synthesis Signal Modeling	178
9.6	Summary	180
	Further Reading	181
<b>10</b>	<b>Image Deblurring – A Case Study</b>	185
10.1	Problem Formulation	185
10.2	The Dictionary	186
10.3	Numerical Considerations	188
10.4	Experiment Details and Results	191
10.5	Summary	198
	Further Reading	199
<b>11</b>	<b>MAP versus MMSE Estimation</b>	201
11.1	A Stochastic Model and Estimation Goals	201
11.2	Background on MAP and MMSE	202
11.3	The Oracle Estimation	204
11.3.1	Developing the Oracle Estimator	204
11.3.2	The Oracle Error	206
11.4	The MAP Estimation	208
11.4.1	Developing the MAP Estimator	208
11.4.2	Approximating the MAP Estimator	211
11.5	The MMSE Estimation	212
11.5.1	Developing the MMSE Estimator	212
11.5.2	Approximating the MMSE Estimator	215
11.6	MMSE and MAP Errors	218
11.7	More Experimental Results	220
11.8	Summary	224
	Further Reading	224
<b>12</b>	<b>The Quest for a Dictionary</b>	227
12.1	Choosing versus Learning	227
12.2	Dictionary-Learning Algorithms	228
12.2.1	Core Questions in Dictionary-Learning	229
12.2.2	The MOD Algorithm	230
12.2.3	The K-SVD Algorithm	231
12.3	Training Structured Dictionaries	237
12.3.1	The Double-Sparsity Model	239
12.3.2	Union of Unitary Bases	241
12.3.3	The Signature Dictionary	242

12.4 Summary .....	244
Further Reading .....	244
<b>13 Image Compression – Facial Images .....</b>	<b>247</b>
13.1 Compression of Facial Images .....	247
13.2 Previous Work .....	249
13.3 Sparse-Representation-Based Coding Scheme .....	250
13.3.1 The General Scheme .....	251
13.3.2 VQ Versus Sparse Representations .....	253
13.4 More Details and Results .....	254
13.4.1 K-SVD Dictionaries .....	255
13.4.2 Reconstructed Images .....	255
13.4.3 Run-Time and Memory Usage .....	260
13.4.4 Comparing to Other Techniques .....	261
13.4.5 Dictionary Redundancy .....	262
13.5 Post-Processing for Deblocking .....	263
13.5.1 The Blockiness Artifacts .....	263
13.5.2 Possible Approaches For Deblocking .....	265
13.5.3 Learning-Based Deblocking Approach .....	266
13.6 Deblocking Results .....	267
13.7 Summary .....	268
Further Reading .....	269
<b>14 Image Denoising .....</b>	<b>273</b>
14.1 General Introduction – Image Denoising .....	273
14.2 The Beginning: Global Modeling .....	274
14.2.1 The Core Image-Denoising Algorithm .....	274
14.2.2 Various Improvements .....	276
14.3 From Global to Local Modeling .....	278
14.3.1 The General Methodology .....	278
14.3.2 Learning the Shrinkage Curves .....	279
14.3.3 Learned Dictionary and Globalizing the Prior .....	286
14.3.4 The Non-Local-Means Algorithm .....	292
14.3.5 3D-DCT Shrinkage: BM3D Denoising .....	296
14.4 SURE for Automatic Parameter Setting .....	297
14.4.1 Development of the SURE .....	298
14.4.2 Demonstrating SURE to Global-Thresholding .....	300
14.5 Summary .....	302
Further Reading .....	303
<b>15 Other Applications .....</b>	<b>309</b>
15.1 General .....	309
15.2 Image Separation via MCA .....	310
15.2.1 Image = Cartoon + Texture .....	310
15.2.2 Global MCA for Image Separation .....	312

15.2.3	Local MCA for Image Separation . . . . .	316
15.3	Image Inpainting and Impulsive Noise Removal . . . . .	324
15.3.1	Inpainting Sparse-Land Signals – Core Principles . . . . .	324
15.3.2	Inpainting Images – Local K-SVD . . . . .	327
15.3.3	Inpainting Images – The Global MCA . . . . .	335
15.3.4	Impulse-Noise Filtering . . . . .	338
15.4	Image Scale-Up . . . . .	341
15.4.1	Modeling the Problem . . . . .	343
15.4.2	The Super-Resolution Algorithm . . . . .	346
15.4.3	Scaling-Up Results . . . . .	349
15.4.4	Image Scale-Up: Summary . . . . .	351
15.5	Summary . . . . .	353
	Further Reading . . . . .	354
<b>16</b>	<b>Epilogue . . . . .</b>	<b>359</b>
16.1	What is it All About? . . . . .	359
16.2	What is Still Missing? . . . . .	359
16.3	Bottom Line . . . . .	360
	<b>Notation . . . . .</b>	<b>363</b>
	<b>Acronyms . . . . .</b>	<b>369</b>
	<b>Index . . . . .</b>	<b>371</b>

## **Part I**

# **Sparse and Redundant Representations – Theoretical and Numerical Foundations**



# Chapter 1

## Prologue

A central achievement of classical linear algebra is a thorough examination of the problem of solving systems of linear equations. The results – definite, timeless, and profound – give the subject a completely settled appearance. As linear systems of equations are the core engine in many engineering developments and solutions, much of this knowledge is practically and successfully deployed in applications. Surprisingly, within this well-understood arena there is an elementary problem that has to do with sparse solutions of linear systems, which only recently has been explored in depth; we will see that this problem has surprising answers, and it inspires numerous practical developments. In this chapter we shall concentrate on defining this problem carefully, and set the stage for its answers in later chapters.

### 1.1 Underdetermined Linear Systems

Consider a matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$  with  $n < m$ , and define the underdetermined linear system of equations  $\mathbf{Ax} = \mathbf{b}$ . 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  $\mathbb{R}^n$ .

In engineering we often encounter problems formulated as such underdetermined linear systems of equations. As an example from image processing, consider the image scale-up problem, where an unknown image undergoes a blur and scale-down operation, and the outcome is given as a lower quality and smaller image  $\mathbf{b}$ . The matrix  $\mathbf{A}$  stands for the degradation operations, and our aim is to reconstruct the original image  $\mathbf{x}$  from the given measurements  $\mathbf{b}$ . Clearly, there are infinitely many possible images  $\mathbf{x}$  that may “explain”  $\mathbf{b}$ , among which there are some that may look better than others. How can we find the proper  $\mathbf{x}$ ?



## 1.2 Regularization

Both in the above example and in many other problems that assume the same formulation, we desire a single solution, and the fact that there are infinitely many of those stands as a major obstacle. In order to narrow this choice to one well-defined solution, additional criteria are needed. A familiar way to do this is *regularization*, where a function  $J(\mathbf{x})$  that evaluates the desirability of a would-be solution  $\mathbf{x}$  is introduced, with smaller values being preferred. Defining the general optimization problem ( $P_J$ )

$$(P_J) : \quad \min_{\mathbf{x}} J(\mathbf{x}) \text{ subject to } \mathbf{b} = \mathbf{A}\mathbf{x}, \quad (1.1)$$

it is now in the hands of  $J(\mathbf{x})$  to govern the kind of solution(s) we may obtain. Returning to the image scale-up example, a function  $J(\mathbf{x})$  that prefers smooth or, better yet, piecewise smooth results, is typically used.

The most known choice of  $J(\mathbf{x})$  is the squared Euclidean norm  $\|\mathbf{x}\|_2^2$ . The problem ( $P_2$ ) that results from such a choice has in-fact a unique solution  $\hat{\mathbf{x}}$  – the so called minimum-norm solution. Using Lagrange multipliers, we define the Lagrangian

$$\mathcal{L}(\mathbf{x}) = \|\mathbf{x}\|_2^2 + \lambda^T (\mathbf{A}\mathbf{x} - \mathbf{b}), \quad (1.2)$$

with  $\lambda$  being the Lagrange multipliers for the constraint set. Taking a derivative of  $\mathcal{L}(\mathbf{x})$  with respect to  $\mathbf{x}$ , we obtain the requirement

$$\frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mathbf{x}} = 2\mathbf{x} + \mathbf{A}^T \lambda. \quad (1.3)$$

Thus the solution is obtained as<sup>1</sup>

$$\hat{\mathbf{x}}_{opt} = -\frac{1}{2}\mathbf{A}^T \lambda. \quad (1.4)$$

Plugging this solution into the constraint  $\mathbf{A}\mathbf{x} = \mathbf{b}$  leads to

$$\mathbf{A}\hat{\mathbf{x}}_{opt} = -\frac{1}{2}\mathbf{A}\mathbf{A}^T \lambda = \mathbf{b} \Rightarrow \lambda = -2(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b}. \quad (1.5)$$

Assigning this to Equation (1.4) gives the well-known closed-form pseudo-inverse solution

$$\hat{\mathbf{x}}_{opt} = -\frac{1}{2}\mathbf{A}^T \lambda = \mathbf{A}^T (\mathbf{A}\mathbf{A}^T)^{-1} \mathbf{b} = \mathbf{A}^+ \mathbf{b}. \quad (1.6)$$

Note that since we have assumed that  $\mathbf{A}$  is full-rank, the matrix  $\mathbf{A}\mathbf{A}^T$  is positive-definite and thus invertible. We also note that for a more general choice of the form  $J(\mathbf{x}) = \|\mathbf{B}\mathbf{x}\|_2^2$  (such that  $\mathbf{B}^T \mathbf{B}$  is invertible), a closed-form expression of the solution

---

<sup>1</sup> Here and elsewhere in this book we use matrix calculus formulae.

also exists,

$$\hat{\mathbf{x}}_{opt} = (\mathbf{B}^T \mathbf{B})^{-1} \mathbf{A}^T (\mathbf{A}(\mathbf{B}^T \mathbf{B})^{-1} \mathbf{A}^T)^{-1} \mathbf{b}, \quad (1.7)$$

following the same line of steps.

The use of  $\ell_2$  is widespread in various fields of engineering, and this is mainly due to its simplicity as manifested by the above closed-form and unique solution. In signal and image processing, this regularization has been practiced quite often for various inverse problems, representation of signals, and elsewhere. This, however, is by no means a declaration that  $\ell_2$  is the truly best choice for the various problems it has been brought to serve. In many cases, the mathematical simplicity of  $\ell_2$  is a misleading fact that diverts engineers from making better choices for  $J(\cdot)$ . Indeed, in image processing, after three decades of trying to successfully activate the ( $\ell_2$ -norm based) Wiener filter, it was eventually realized that the solution resides with a different, robust-statistics based choice of  $J(\cdot)$ , of the kind we shall often meet in this book.

### 1.3 The Temptation of Convexity

The fact that  $\ell_2$  leads to a unique solution is a manifestation of a wider phenomenon, as selecting any strictly convex function  $J(\cdot)$  guarantees such uniqueness. Let us recall the definition of convexity, first for sets, and then for functions.

**Definition 1.1.** A set  $\Omega$  is convex if  $\forall \mathbf{x}_1, \mathbf{x}_2 \in \Omega$  and  $\forall t \in [0, 1]$ , the convex combination  $\mathbf{x} = t\mathbf{x}_1 + (1 - t)\mathbf{x}_2$  is also in  $\Omega$ .

It is easy to verify that the set  $\Omega = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} = \mathbf{b}\}$  is convex, using the above definition. Thus, the feasible set of solutions of the optimization problem posed in Equation (1.1) is convex. In order for this optimization problem to be convex as a whole, we have to add a convexity requirement on the penalty  $J(\mathbf{x})$ . The core definition for such a property is given by:

**Definition 1.2.** A function  $J(\mathbf{x}) : \Omega \rightarrow \mathbb{R}$  is convex if  $\forall \mathbf{x}_1, \mathbf{x}_2 \in \Omega$  and  $\forall t \in [0, 1]$ , the convex combination point  $\mathbf{x} = t\mathbf{x}_1 + (1 - t)\mathbf{x}_2$  satisfies

$$J(t\mathbf{x}_1 + (1 - t)\mathbf{x}_2) \leq tJ(\mathbf{x}_1) + (1 - t)J(\mathbf{x}_2). \quad (1.8)$$

Another way to understand the above definition is this: the epigraph of  $J(\mathbf{x})$  (the area defined by  $\{(\mathbf{x}, y) \mid y \geq J(\mathbf{x})\}$ ) is a convex set in  $\mathbb{R}^{m+1}$ .

If  $J(\cdot)$  is twice continuously differentiable, its derivatives can be used for alternative definitions of convexity:

**Definition 1.3.** A function  $J(\mathbf{x}) : \Omega \rightarrow \mathbb{R}$  is convex if  $\forall \mathbf{x}_1, \mathbf{x}_2 \in \Omega$  if and only if

$$J(\mathbf{x}_2) \geq J(\mathbf{x}_1) + \nabla J(\mathbf{x}_1)^T (\mathbf{x}_2 - \mathbf{x}_1), \quad (1.9)$$

or alternatively, if and only if  $\nabla^2 J(\mathbf{x}_1)$  is positive-definite.

Using the definition of the positive-definiteness of the Hessian makes the proof of the convexity of the squared  $\ell_2$ -norm trivial, since  $\nabla^2 \|\mathbf{x}\|_2^2 = 2\mathbf{I} \geq \mathbf{0}$ . Indeed, the squared  $\ell_2$ -norm is strictly convex since this Hessian is strictly positive-definite for all  $\mathbf{x}$ . Returning to the problem posed in Equation (1.2), since the constraint-set is convex and the penalty is strictly so, a unique solution is guaranteed.

So far we have dwelled with the choice  $J(\mathbf{x}) = \|\mathbf{x}\|_2^2$ . However, there are many other choices of functions  $J(\cdot)$  that are convex or strictly so. Even though these options of  $J(\cdot)$  rarely lead to a closed-form solution, the fact that they are strictly convex implies a unique solution.<sup>2</sup> Perhaps more importantly, carefully chosen optimization algorithms for solving (1.2) are guaranteed to converge to the global minimizer of such optimization problems.

These properties make convex problems very appealing in engineering in general, and in signal and image processing in particular. Again, this does not mean that there is no value to non-convex problems – it simply says that when dealing with non-convex optimization tasks, one should be aware of problems that might be encountered because of their imperfection.

Special cases of interest for convex functions are all the  $\ell_p$ -norms for  $p \geq 1$  (use the Hessian to verify that). These are defined as

$$\|\mathbf{x}\|_p^p = \sum_i |x_i|^p. \quad (1.10)$$

In particular, the  $\ell_\infty$ -norm (obviously, without the  $p$ -th power<sup>3</sup>) and the  $\ell_1$ -norm are very interesting and popular; the  $\ell_\infty$  considers the maximal entry of the vector  $\mathbf{x}$ , and  $\ell_1$  sums the absolute entries. We shall have a special interest in  $\ell_1$  due to its tendency to sparsify the solution – a fact that will be discussed and established as we move forward in this book.

## 1.4 A Closer Look at $\ell_1$ Minimization

The choice  $J(\mathbf{x}) = \|\mathbf{x}\|_1$  is convex but not strictly so, a fact easily verified by noticing that if  $\mathbf{x}_1$  and  $\mathbf{x}_2$  are in the same quadrant (i.e., they have the same sign-pattern), then their convex combination gives an equality in Equation (1.8). Thus, the problem

$$(P_1) : \quad \min_{\mathbf{x}} \quad \|\mathbf{x}\|_1 \quad \text{subject to} \quad \mathbf{b} = \mathbf{A}\mathbf{x} \quad (1.11)$$

may have more than one solution. Nevertheless, even if there are infinitely many possible solutions to this problem, we can claim that (i) these solutions are gathered

<sup>2</sup> For non-strict convexity, uniqueness is not guaranteed.

<sup>3</sup> The  $\ell_p$ -norm with  $1 \leq p < \infty$  without the  $p$ -th power is a convex function but not strictly so. This is because these functions behave like a cone with a linear slope, implying that a carefully chosen pair of points (two scaled points) can give exact equality in Equation (1.8).

in a set that is bounded and convex, and (ii) among these solutions, there exists at least one with at most  $n$  non-zeros (as the number of constraints).

The convexity of the solution set (in the first property) is immediate, stemming as a direct consequence of the fact that all optimal solutions have the same penalty ( $\ell_1$ -length in this case); any convex combination of two such solutions must give a lower or equal penalty due to the convexity of  $J(\mathbf{x}) = \|\mathbf{x}\|_1$ , and since the two chosen solutions are optimal, a lower penalty is impossible to obtain. Thus, any convex combination of two optimal solutions is also an optimal solution.

The fact that the solution set is bounded is a direct consequence of the fact that all optimal solutions give a penalty of the same height,  $v_{\min} = \|\mathbf{x}_{opt}\|_1 < \infty$ . Given any two optimal solutions,  $\mathbf{x}_{opt}^1$  and  $\mathbf{x}_{opt}^2$ , the distance between them satisfies

$$\|\mathbf{x}_{opt}^1 - \mathbf{x}_{opt}^2\|_1 \leq \|\mathbf{x}_{opt}^1\|_1 + \|\mathbf{x}_{opt}^2\|_1 = 2v_{\min},$$

implying that all solutions are nearby.

In order to show the second property, let us assume that an optimal solution  $\mathbf{x}_{opt}$  has been found for  $(P_1)$ , with  $k > n$  non-zeros. Clearly, the  $k$  columns combined linearly by  $\mathbf{x}_{opt}$  are linearly-dependent, and thus there exists a non-trivial vector  $\mathbf{h}$  that combines these columns to zero (i.e., the support of  $\mathbf{h}$  is contained within the support of  $\mathbf{x}_{opt}$ ),  $\mathbf{A}\mathbf{h} = \mathbf{0}$ .

We consider the vector  $\mathbf{x} = \mathbf{x}_{opt} + \epsilon\mathbf{h}$ , for very small values of  $\epsilon$  that guarantee that no entry in the new vector changes its sign. Any value satisfying  $|\epsilon| \leq \min_i |x_{opt}^i|/|h^i|$  is suitable. First, it is clear that this vector is guaranteed to satisfy the linear constraint  $\mathbf{A}\mathbf{x} = \mathbf{0}$ , and as such, it is a feasible solution to  $(P_1)$ . Furthermore, since  $\mathbf{x}_{opt}$  is assumed to be optimal, we must assume that

$$\forall |\epsilon| \leq \min_i \frac{|x_{opt}^i|}{|h^i|}, \quad \|\mathbf{x}\|_1 = \|\mathbf{x}_{opt} + \epsilon\mathbf{h}\|_1 \geq \|\mathbf{x}_{opt}\|_1.$$

We argue that the above inequality is in fact an equality. The above relationship applies to both positive and negative values of  $\epsilon$  in a region where the  $\ell_1$  function is continuous and differentiable (because all the vectors  $\mathbf{x}_{opt} + \epsilon\mathbf{h}$  have the same sign-pattern). Thus, as claimed, the only way this could be true is if the above inequality is satisfied as an equality. This implies that the addition/subtraction of  $\mathbf{h}$  under those circumstances does not change the  $\ell_1$ -length of the solution. As an example, if  $\mathbf{x}_{opt}$  has only positive entries, then the entries of  $\mathbf{h}$  should be both positive and negative and sum to zero. More generally, we should require that  $\mathbf{h}^T \text{sign}(\mathbf{x}_{opt}) = 0$ .

Our next step is to tune  $\epsilon$  in such a way that one entry in  $\mathbf{x}_{opt}$  is nulled, while keeping the solution's  $\ell_1$  length. We choose the index  $i$  that gives the minimum ratio  $|x_{opt}^i|/|h^i|$ , and we pick  $\epsilon = -x_{opt}^i/h^i$ . In the resulting vector  $\mathbf{x}_{opt} + \epsilon\mathbf{h}$ , the  $i$ -th entry is nulled, while all the others keep their sign, and we also have that  $\|\mathbf{x}_{opt} + \epsilon\mathbf{h}\|_1 = \|\mathbf{x}_{opt}\|_1$ . This way we got a new optimal solution with  $k - 1$  non-zeros at the most (because it may be that more than one entry have been simultaneously nulled). This process can be repeated until  $k = n$ . Below this, it is possible to null entries only if the columns in  $\mathbf{A}$  are linearly-dependent, and this would rarely be the case.

From the above we have learned that the  $\ell_1$ -norm has a tendency to prefer sparse solutions. The property we have proven is well known and considered as the fundamental property of linear programming — a tendency towards basic (sparse) solutions. However, as we shall see later on, while we are very much interested in sparsity, getting  $n$  non-zeros would be considered as far too dense for our needs, and deeper sparsity would be sought.

## 1.5 Conversion of $(P_1)$ to Linear Programming

Suppose that in  $(P_1)$ , the unknown  $\mathbf{x}$  is replaced by  $\mathbf{x} = \mathbf{u} - \mathbf{v}$ , where  $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$  are both non-negative vectors such that  $\mathbf{u}$  takes all the positive entries in  $\mathbf{x}$ , other entries null, and  $\mathbf{v}$  does the same for the negative entries in  $\mathbf{x}$ . With this replacement, and by denoting the concatenated vector  $\mathbf{z} = [\mathbf{u}^T, \mathbf{v}^T]^T \in \mathbb{R}^{2n}$ , it is easy to see that<sup>4</sup>  $\|\mathbf{x}\|_1 = \mathbf{1}^T(\mathbf{u} + \mathbf{v}) = \mathbf{1}^T\mathbf{z}$  and  $\mathbf{A}\mathbf{x} = \mathbf{A}(\mathbf{u} - \mathbf{v}) = [\mathbf{A}, -\mathbf{A}]\mathbf{z}$ . Thus, the  $(P_1)$  optimization problem posed in Equation (1.11) can be re-written as

$$\min_{\mathbf{z}} \mathbf{1}^T\mathbf{z} \text{ subject to } \mathbf{b} = [\mathbf{A}, -\mathbf{A}]\mathbf{z} \text{ and } \mathbf{z} \geq \mathbf{0}. \quad (1.12)$$

In this way, rather than minimize an  $\ell_1$ -norm problem, we face a new problem having a classical Linear-Programming (LP) structure. For the two problems  $((P_1)$  and the LP) to be equivalent, we must show that our assumption about the decomposition of  $\mathbf{x}$  to positive and negative entries is satisfied, and solutions to (1.12) cannot be such that  $\mathbf{u}^T\mathbf{v} \neq 0$  (i.e., the supports of  $\mathbf{u}$  and  $\mathbf{v}$  overlap).

This is easily verified by observing that if in a given optimal solution of (1.12) 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 column 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 while also reducing the penalty by  $u_k - v_k > 0$ , contradicting the optimality of the initial solution. Thus, the supports of  $\mathbf{u}, \mathbf{v}$  do not overlap, and LP is indeed equivalent to  $(P_1)$ .

## 1.6 Promoting Sparse Solutions

As we move from  $\ell_2$  regularization towards  $\ell_1$ , we promote sparser solutions. Using this rationale, we can consider  $\ell_p$ -“norms” with  $p < 1$ . One has to be careful as such  $\ell_p$  are no longer formal norms for  $p < 1$ , as the triangle inequality is no longer

<sup>4</sup> The notation  $\mathbf{1}$  stands for a vector of ones of the proper length.

satisfied.<sup>5</sup> Nevertheless, we shall use the term norm for these functions as well, keeping in mind this reservation.

Do these norms lead to sparser solutions? In order to get a feeling for the behavior of these norms, we consider the following problem: Assume that  $\mathbf{x}$  is known to have unit length in  $\ell_p$ . Let us find the shortest such vector in  $\ell_q$  for  $q < p$ . Putting this as an optimization problem, this reads

$$\min_{\mathbf{x}} \|\mathbf{x}\|_q^q \text{ subject to } \|\mathbf{x}\|_p^p = 1. \quad (1.13)$$

We shall assume that  $\mathbf{x}$  has  $a$  non-zeros (we shall further assume that these values are all positive, since their sign does not affect the analysis that follows) as the leading entries in the vector, and the rest are zeros. We define the Lagrangian function as

$$\mathcal{L}(\mathbf{x}) = \|\mathbf{x}\|_q^q + \lambda(\|\mathbf{x}\|_p^p - 1) = -\lambda + \sum_{k=1}^a (|x_k|^q + \lambda|x_k|^p). \quad (1.14)$$

This function is separable, handling every entry in  $\mathbf{x}$  independently of the others. The optimal solution is given by  $x_k^{p-q} = \text{Const}$  for all  $k$ , implying that all the non-zero entries are supposed to be the same. The constraint  $\|\mathbf{x}\|_p^p = 1$  leads to  $x_k = a^{-1/p}$ , and the  $\ell_q$ -norm for this solution is given by  $\|\mathbf{x}\|_q^q = a^{1-q/p}$ . Since  $q < p$ , this means that the shortest  $\ell_q$ -norm is obtained for  $a = 1$ , having only one non-zero entry in  $\mathbf{x}$ .

The above result states that for any pair of  $\ell_p$ - and  $\ell_q$ -norms with  $q < p$ , a unit-length  $\ell_p$ -norm vector becomes the shortest in  $\ell_q$  when it is the sparsest possible. A geometrical interpretation of the analysis given above is the following: The unit  $\ell_p$ -ball surface in  $\mathbb{R}^m$  represents the feasible set of the problem posed in (1.13). We blow an  $\ell_q$  “balloon” in the same space and search for the first intersection of it with the  $\ell_p$ -ball. The result we got implies that this intersection takes place along the axes, where all the entries apart from one are zeros. This is demonstrated in Figure 1.1.

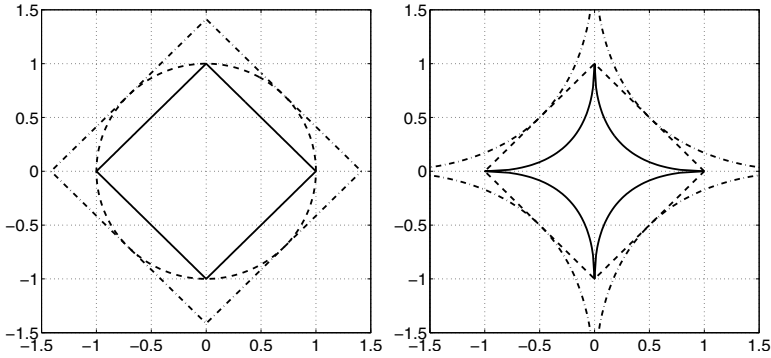
Another way to illustrate the tendency of  $\ell_p$ -norm to drive results to become sparse is the following: Consider the problem we have been looking at,

$$(P_p) : \quad \min \|\mathbf{x}\|_p^p \text{ subject to } \mathbf{Ax} = \mathbf{b}. \quad (1.15)$$

The linear set of equations forming the constraint define 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,  $\mathbf{x}_0$ . A linear combination of  $\mathbf{x}_0$  and any vector from the null-space of  $\mathbf{A}$  forms a feasible solution as well. Geometrically, this set appears as a hyperplane of dimension  $\mathbb{R}^{m-n}$  embedded in the  $\mathbb{R}^m$  space.

It is within this space that we seek the solution to the problem posed as  $(P_p)$ . Geometrically speaking, solving  $(P_p)$  is done by “blowing” again an  $\ell_p$  balloon

<sup>5</sup> The properties a norm should satisfy are: (i) zero vector:  $\|\mathbf{v}\| = 0$  if and only if  $\mathbf{v} = \mathbf{0}$ ; (ii) absolute homogeneity:  $\forall t \neq 0, \|t\mathbf{u}\| = |t|\|\mathbf{u}\|$ , and (iii) the triangle inequality:  $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$ . Note that from properties (ii) and (iii) it is implied that a norm is a convex function because  $\|t\mathbf{u} + (1-t)\mathbf{v}\| \leq t\|\mathbf{u}\| + (1-t)\|\mathbf{v}\|$  for all  $t \in [0, 1]$ , implying convexity.



**Fig. 1.1** Demonstrating the fact that a unit-length  $\ell_p$ -norm vector (dashed) becomes shortest in  $\ell_q$  ( $p > q$ ) (solid) when it is the sparsest possible. On the left this is demonstrated for  $p = 2$  and  $q = 1$ , and the right shows it for  $p = 1$  and  $q = 0.5$ . The dash-dotted curves in both graphs show the reverse claim – maximizing the  $\ell_q$  length leads to the most non-sparse outcome.

centered around the origin, and stopping its inflation when it first touches the feasible set. The question is what characterizes such an intersection point? Figure 1.2 presents a simple demonstration of this process for a tilted hyperplane (serving as the constraint-set) and several  $p$  values: 2, 1.5, 1, and 0.7. One can see that norms with  $p \leq 1$  tend to give that the intersection point is on the ball-corners, which take place on the axes. This implies that 2 of the 3 coordinates are zeros, which is the tendency to sparsity we are referring to. As opposed to this,  $\ell_2$  and even  $\ell_{1.5}$  give intersection points which are not sparse, with three non-zero coordinates.

Put more generally, an intersection of an affine subspace and an  $\ell_p$  ball is expected to take place on the axes for  $p \leq 1$ , and thus lead to a sparse solution. Even the  $\ell_1$ -ball enjoys this property, and in fact, one has to be highly unlucky with the angle of the affine subspace to avoid a sparse outcome.

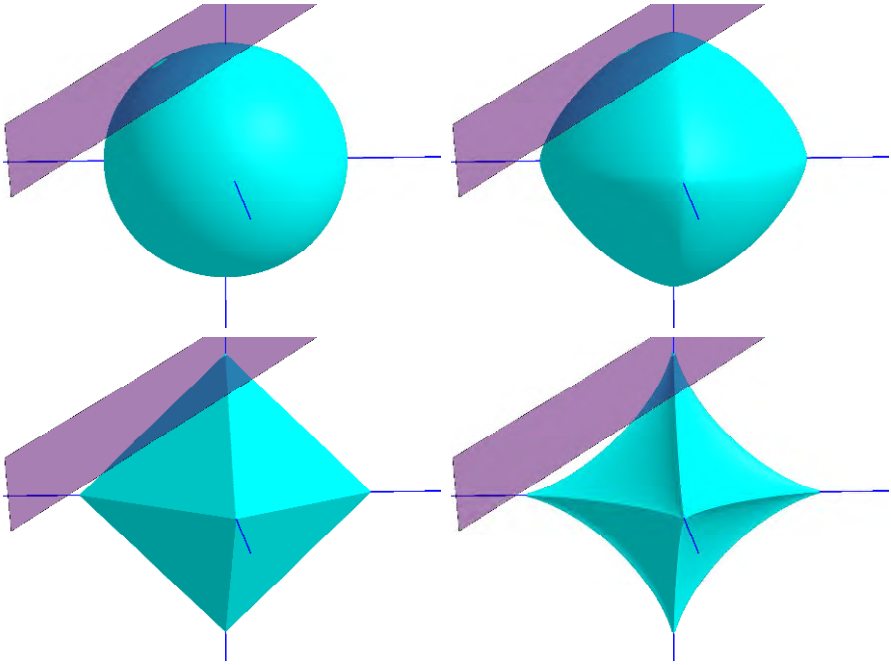
Another similar measure promoting sparsity is the weak  $\ell_p$ -norm. Denoting by  $N(\epsilon, \mathbf{x})$  the number of entries in  $\mathbf{x}$  exceeding  $\epsilon$ , we define the weak  $\ell_p$ -norm as

$$\|\mathbf{x}\|_{w\ell_p}^p = \sup_{\epsilon > 0} N(\epsilon, \mathbf{x}) \cdot \epsilon^p. \quad (1.16)$$

Here, just as before,  $0 < p \leq 1$  is the interesting range of  $p$ , giving very powerful sparsity measure. The weak  $\ell_p$  norm is a popular measure of sparsity used by the mathematical analysis community. The usual  $\ell_p$ -norm is almost equivalent, and in fact, the two become the same for a vector with uniform spread of its energy among its non-zero entries. The plain  $\ell_p$  is easier to handle and thus more popular in most cases.

It would seem very natural, based on our discussion above, to attempt to solve a problem of the form

$$(P_p) : \quad \min \|\mathbf{x}\|_p^p \text{ subject to } \mathbf{Ax} = \mathbf{b}, \quad (1.17)$$

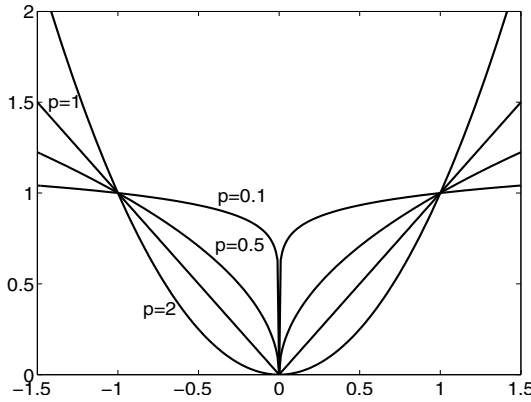


**Fig. 1.2** The intersection between the  $\ell_p$ -ball and the set  $\mathbf{Ax} = \mathbf{b}$  defines the solution of  $(P_p)$ . This intersection is demonstrated here in 3D for  $p = 2$  (top left),  $p = 1.5$  (top right),  $p = 1$  (bottom left), and  $p = 0.7$  (bottom right). When  $p \leq 1$ , the intersection takes place at a corner of the ball, leading to a sparse solution.

for example, with  $p = 2/3$ ,  $p = 1/2$ , or even smaller  $p$ , tending to zero. Unfortunately, each choice  $0 < p < 1$  leads to a non-convex optimization problem, and this raises some difficulties, as we have mentioned above. Nevertheless, from an engineering point of view, if sparsity is a desired property, and we know that  $\ell_p$  serves it well, this problem can and should be used, despite its weaknesses.

While all the discussion above focuses on the  $\ell_p$ -norm, there are other functions of  $\mathbf{x}$  that promote sparsity just as well. In fact, any function  $J(\mathbf{x}) = \sum_i \rho(x_i)$  with  $\rho(x)$  being symmetric, monotonically non-decreasing, and with a monotonic non-increasing derivative for  $x \geq 0$  will serve the same purpose of promoting sparsity. As classic examples of this family, we mention  $\rho(x) = 1 - \exp(-|x|)$ ,  $\rho(x) = \log(1 + |x|)$ , and  $\rho(x) = |x|/(1 + |x|)$ .





**Fig. 1.3** The behavior of  $|x|^p$  for various values of  $p$ . As  $p$  tends to zero,  $|x|^p$  approaches the indicator function, which is 0 for  $x = 0$  and 1 elsewhere.

## 1.7 The $\ell_0$ -Norm and Implications

The extreme among all the sparsifying norms is the case of  $p \rightarrow 0$ . We denote the  $\ell_0$ -norm as<sup>6</sup>

$$\|\mathbf{x}\|_0 = \lim_{p \rightarrow 0} \|\mathbf{x}\|_p^p = \lim_{p \rightarrow 0} \sum_{k=1}^m |x_k|^p = \#\{i : x_i \neq 0\}. \quad (1.18)$$

This is a very simple and intuitive measure of sparsity of a vector  $\mathbf{x}$ , counting the number of nonzero entries in it. In order to see this counting behavior, [Figure 1.3](#) presents the scalar weight function  $|x|^p$  – the core of the norm computation – for various values of  $p$ . It shows that as  $p$  goes to zero, this curve becomes an indicator function, being 0 for  $x = 0$  and 1 for every other value. Thus, summation of such measures on all the entries of  $\mathbf{x}$  becomes a count of the non-zeros in this vector, as claimed.

The term  $\ell_0$ -norm is misleading, as this function does not satisfy all the axiomatic requirements of a norm. More specifically, if we consider the  $\ell_0$  as a continuation of the  $\ell_p$ -norm for  $p \rightarrow 0$ , then for checking its behavior we need to take its 0-th root,<sup>7</sup> and this is impossible. Alternatively, we can simply refer to the function  $\|\mathbf{x}\|_0$  as a candidate function for a norm. While this function satisfies the triangle inequality,  $\|\mathbf{u} + \mathbf{v}\|_0 \leq \|\mathbf{u}\|_0 + \|\mathbf{v}\|_0$ , the homogeneity property is not met: for  $t \neq 0$ ,  $\|t\mathbf{u}\|_0 = \|\mathbf{u}\|_0 \neq t\|\mathbf{u}\|_0$ . This loss of sensitivity to scale will return to haunt us from time to time.

<sup>6</sup> A more accurate notation would be  $\|\mathbf{x}\|_0^0$ , but we shall stick to the one described above, so as to be consistent with the literature.

<sup>7</sup> The homogeneity and the triangle inequality properties are both tested with the  $p$ -th root of the function described in Equation (1.10).

We should note that the  $\ell_0$  norm, while providing a very simple and easily grasped 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. A more relaxed and forgiving notion of sparsity can and should be built on the notion of approximately representing a vector using a small number of nonzeros; this can be quantified by the weak- $\ell_p$  and the usual  $\ell_p$ -norms described above. Nevertheless, we proceed the discussion here with the assumption that  $\ell_0$  is the measure of interest.

## 1.8 The $(P_0)$ Problem – Our Main Interest

Consider the problem  $(P_0)$  obtained from the general prescription  $(P_J)$  with the choice  $J(\mathbf{x}) = J_0(\mathbf{x}) \equiv \|\mathbf{x}\|_0$ ; explicitly:

$$(P_0) : \quad \min_{\mathbf{x}} \quad \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{Ax} = \mathbf{b}. \quad (1.19)$$

Sparsity optimization (1.19) looks superficially like the minimum  $\ell_2$ -norm problem  $(P_2)$ , but the notational similarity masks some startling differences. The solution to  $(P_2)$  is always unique, and is readily available through now-standard tools from computational linear algebra.  $(P_0)$  has probably been considered as a possible goal from time to time for many years, but it seems initially to pose many conceptual challenges that have inhibited its widespread study and application. These are rooted in the discrete and discontinuous nature of the  $\ell_0$  norm; the standard convex analysis ideas which underpin the analysis of  $(P_2)$  do not apply. Many of the most basic questions about  $(P_0)$  seem immune to immediate insight:

- Can uniqueness of a solution be claimed? Under what conditions?
- If a candidate solution is available, can we perform a simple test to verify that the solution is actually the global minimizer of  $(P_0)$ ?

Perhaps in some instances, with very special matrices  $\mathbf{A}$  and left hand sides  $\mathbf{b}$ , ways to answer such questions are apparent, but for general classes of problem instances  $(\mathbf{A}, \mathbf{b})$ , such insights initially seem unlikely.

Beyond conceptual issues of uniqueness and verification of solutions, one is easily overwhelmed by the apparent difficulty of solving  $(P_0)$ . This is a classical problem of combinatorial search; one sweeps exhaustively through all possible sparse subsets, generating corresponding subsystems  $\mathbf{b} = \mathbf{A}_S \mathbf{x}_S$  where  $\mathbf{A}_S$  denotes the matrix having  $|S|$  columns chosen from those columns of  $\mathbf{A}$  with indices in  $S$ ; and checking if  $\mathbf{b} = \mathbf{A}_S \mathbf{x}_S$  can be solved.

Let us illustrate this complexity by the following simple example: Assume that  $\mathbf{A}$  is of size  $500 \times 2000$  ( $n = 500$ ,  $m = 2000$ ), and suppose that we know that the sparsest solution to  $(P_0)$  has  $|S| = 20$  non-zeros. We desire to find the proper set of  $|S|$  columns. Thus, we exhaustively sweep through all  $\binom{m}{|S|} \approx 3.9E+47$  such options, and per each test the small linear system  $\mathbf{b} = \mathbf{A}_S \mathbf{x}_S$  for equality. Assume that each

individual test of such a system requires  $1E-9$  seconds. A simple calculation reveals that it will take more than  $1.2E + 31$  years(!!!) to conclude these series of tests.

The complexity of exhaustive search is exponential in  $m$ , and indeed, it has been proven that  $(P_0)$  is, in general, NP-Hard. Thus, a mandatory and crucial set of questions arise: Can  $(P_0)$  be efficiently solved by some other means? Can approximate solutions be accepted? How accurate can those be? What kind of approximations will work? These are the questions we aim to answer in this book.

## 1.9 The Signal Processing Perspective

We know today that finding sparse solutions to underdetermined linear systems may become better-behaved and may be a much more practically relevant notion than we might have supposed just a few years ago. In parallel with this development, another insight has been developing in signal and image processing, where it has been found that many media types (still imagery, video, acoustic) can be sparsely represented using transform-domain methods, and in fact many important tasks dealing with such media can fruitfully be viewed as finding sparse solutions to underdetermined systems of linear equations. Many readers will be familiar with the media encoding standard JPEG and its successor, JPEG2000. Both standards are based on the notion of transform encoding that leads to a sparse representation.

We can thus make a simple connection between the above algebraic discussion and the signal representation problem formal as follows. Let  $\mathbf{b}$  denote the vector of signal/image values to be represented, let  $\mathbf{A}$  be the matrix whose columns are the elements of the different bases to be used in the representation. The problem we have posed in Equation (1.1) aims to provide a single representation among the many possible ones for  $\mathbf{b}$ . When using the  $\ell_2$ -norm measure, the outcome is a linear operator  $\mathbf{A}^+$  that multiplies  $\mathbf{b}$  in order to compute  $\mathbf{x}$  – this is the well-known Frame approach towards redundant representations, which applies linear operations both for the forward transform (from  $\mathbf{b}$  to  $\mathbf{x}$ ) and its inverse (from  $\mathbf{x}$  to  $\mathbf{b}$ ).

On the other extreme, the problem  $(P_0)$  posed in Equation (1.19) offers literally the sparsest representation of the signal content. This way, while the inverse transform is linear, the forward one is highly non-linear, and quite complex in general. The appeal in such a transform is in the compact representation it provides. This option motivates a closer study of the mathematical problems mentioned above. Much more on this connection will be given in the second part of this book, where we explore ways to harness the understanding of sparse solutions of linear systems of equations to applications in signal and image processing.

## Further Reading

1. D.P. Bertsekas, *Nonlinear Programming*, 2nd Edition, Athena Scientific, 2004.

2. S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004.
3. A.M. Bruckstein, D.L. Donoho, and M. Elad, From sparse solutions of systems of equations to sparse modeling of signals and images, *SIAM Review*, 51(1):34–81, February 2009.
4. S.S. Chen, D.L. Donoho, and M.A. Saunders, Atomic decomposition by basis pursuit, *SIAM Journal on Scientific Computing*, 20(1):33–61 (1998).
5. S.S. Chen, D.L. Donoho, and M.A. Saunders, Atomic decomposition by basis pursuit, *SIAM Review*, 43(1):129–159, 2001.
6. C. Daniel and F.S. Wood, *Fitting Equations to Data: Computer Analysis of Multifactor Data*, 2nd Edition, John Wiley and Sons, 1980.
7. G. Davis, S. Mallat, and Z. Zhang, Adaptive time-frequency decompositions, *Optical-Engineering*, 33(7):2183–91, 1994.
8. G.H. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins Studies in Mathematical Sciences, Third edition, 1996.
9. R.A. Horn C.R. Johnson, *Matrix Analysis*, New York: Cambridge University Press, 1985.
10. A.K. Jain, *Fundamentals of Digital Image Processing*, Englewood Cliffs, NJ, Prentice-Hall, 1989.
11. D. Luenberger, *Linear and Nonlinear Programming*, 2nd Edition, Addison-Wesley, Inc., Reading, Massachusetts 1984.
12. S. Mallat, *A Wavelet Tour of Signal Processing*, Academic-Press, 1998.
13. S. Mallat and E. LePennec, Sparse geometric image representation with bandelets, *IEEE Trans. on Image Processing*, 14(4):423–438, 2005.
14. S. Mallat and Z. Zhang, Matching pursuits with time-frequency dictionaries, *IEEE Trans. Signal Processing*, 41(12):3397–3415, 1993.
15. M. Marcus and H. Minc, *A Survey of Matrix Theory and Matrix Inequalities*, Prindle, Weber & Schmidt, Dover, 1992.



## Chapter 2

# Uniqueness and Uncertainty

We return to the basic problem  $(P_0)$ , which is at the core of our discussion,

$$(P_0) : \quad \min_{\mathbf{x}} \quad \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{b} = \mathbf{A}\mathbf{x}.$$

While we shall refer hereafter to this problem as our main goal, we stress that we are quite aware of its two major shortcomings in leading to any practical tool:

1. The equality requirement  $\mathbf{b} = \mathbf{A}\mathbf{x}$  is too strict, as there are small chances for any vector  $\mathbf{b}$  to be represented by a few columns from  $\mathbf{A}$ . A better requirement would be one that allows for small deviation.
2. The sparsity measure is too sensitive to very small entries in  $\mathbf{x}$ , and a better measure would adopt a more forgiving approach towards such small entries.

Both these considerations will be included in later analysis, but for this to succeed, we must start with the stylized version of the problem as indeed posed by  $(P_0)$ .

For the underdetermined linear system of equations,  $\mathbf{A}\mathbf{x} = \mathbf{b}$  (a full-rank matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$  with  $n < m$ ), we pose the following questions:

Q1: When can uniqueness of the sparsest solution be claimed?

Q2: Can a candidate solution be tested to verify its (global) optimality?

This section addresses these questions, and some of their extensions. Rather than answering the above questions directly, we first consider special matrices  $\mathbf{A}$  for which the analysis seems to be easier, and then extend our answers to the general  $\mathbf{A}$ . In doing so, we also follow the path taken by researchers who originally addressed these questions.

## 2.1 Treating the Two-Ortho Case

We shall first discuss the  $(P_0)$  problem defined above in a concrete setting: the case where  $\mathbf{A}$  is the concatenation of two orthogonal matrices,  $\Psi$  and  $\Phi$ . As a classic

example, we can consider the amalgam of the identity and the Fourier matrices  $\mathbf{A} = [\mathbf{I}, \mathbf{F}]$ . In such a setting, the fact that the system  $\mathbf{b} = \mathbf{A}\mathbf{x}$  is underdetermined means, concretely, that there are many ways of representing a given signal  $\mathbf{b}$  as a superposition of spikes (i.e., columns from the identity matrix) and sinusoids (i.e., columns from the Fourier matrix). A sparse solution of such a system is a representation of said signal as a superposition of a few sinusoids and a few spikes. The uniqueness of such a sparse solution seemed surprising when first noticed.

### 2.1.1 An Uncertainty Principle

Before addressing sparse solutions for the linear system  $[\Psi, \Phi]\mathbf{x} = \mathbf{b}$ , we shall consider a (seemingly) different problem, inspired by the setting of classical uncertainty principles. As the reader no doubt knows, the classical uncertainty principle states that two conjugate variables (e.g., position and momentum, or any other pair coupled by the Fourier transform) cannot both be known with arbitrary precision. Turning to its mathematical formulation, it states that any function  $f(x)$  and its Fourier transform  $F(\omega)$  must obey the inequality<sup>1</sup>

$$\int_{-\infty}^{\infty} x^2 |f(x)|^2 dx \cdot \int_{-\infty}^{\infty} \omega^2 |F(\omega)|^2 d\omega \geq \frac{1}{2}, \quad (2.1)$$

where we have assumed that these functions are  $\ell_2$ -normalized,

$$\int_{-\infty}^{\infty} |f(x)|^2 dx = 1.$$

This claim says that a signal cannot be tightly concentrated both in time and in frequency, and there is a lower bound on the product of the spread in time and the spread in frequency.

A comparable claim in our terminology would be that a signal cannot be sparsely represented both in time and in frequency. We now turn to develop this exact viewpoint, as it will be helpful for understanding some of the discussion that follows. Suppose we have a non-zero vector  $\mathbf{b} \in \mathbb{R}^n$  (a signal, say) and two orthobases  $\Psi$  and  $\Phi$ . Then  $\mathbf{b}$  can be represented either as a linear combination of columns of  $\Psi$  or as a linear combination of columns of  $\Phi$ :

$$\mathbf{b} = \Psi\alpha = \Phi\beta. \quad (2.2)$$

Clearly,  $\alpha$  and  $\beta$  are uniquely defined. In a particularly important case,  $\Psi$  is simply the identity matrix, and  $\Phi$  is the matrix of the Fourier transform. Then  $\alpha$  is the time-domain representation of  $\mathbf{b}$  while  $\beta$  is the frequency-domain representation.

For an arbitrary pair of bases  $\Psi, \Phi$ , an interesting phenomenon occurs: either  $\alpha$  can be sparse, or  $\beta$  can be sparse, but not both! However, this claim is clearly

---

<sup>1</sup> The bound 0.5 stated here depends on a specific definition of the Fourier transform.

dependent on the distance between  $\Psi$  and  $\Phi$ , since if the two are the same, we can easily construct  $\mathbf{b}$  to be one of the columns in  $\Psi$  and get the smallest possible cardinality (being 1) in both  $\alpha$  and  $\beta$ . Thus, we turn now to define the proximity between two bases by their *mutual-coherence*.

**Definition 2.1.** For an arbitrary pair of orthogonal bases  $\Psi, \Phi$  that construct the matrix  $\mathbf{A}$ , we define the mutual-coherence  $\mu(\mathbf{A})$  as the maximal inner product between columns from these two bases,

$$\text{proximity}(\Psi, \Phi) = \mu(\mathbf{A}) = \max_{1 \leq i, j \leq n} |\psi_i^T \phi_j|. \quad (2.3)$$

The *mutual-coherence* of such two-ortho matrices satisfies  $1/\sqrt{n} \leq \mu(\mathbf{A}) \leq 1$ , where the lower bound is achievable for certain pairs of orthogonal bases, such as the identity and the Fourier, the identity and the Hadamard, and more. To see that this is indeed the lower bound on the possible coherence, one simply notices that  $\Psi^T \Phi$  is an orthonormal matrix, having the sum of squares of its entries in each column equal to 1. 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. Using Definition 2.3 above, we have the following inequality result:

**Theorem 2.1.** For an arbitrary pair of orthogonal bases  $\Psi, \Phi$  with mutual-coherence  $\mu(\mathbf{A})$ , and for an arbitrary non-zero vector  $\mathbf{b} \in \mathbb{R}^n$  with representations  $\alpha$  and  $\beta$  correspondingly, the following inequality holds true:

$$\text{Uncertainty Principle 1: } \|\alpha\|_0 + \|\beta\|_0 \geq \frac{2}{\mu(\mathbf{A})}. \quad (2.4)$$

**Proof:** We shall assume hereafter, without loss of generality, that  $\|\mathbf{b}\|_2 = 1$ . Since  $\mathbf{b} = \Psi\alpha = \Phi\beta$ , and  $\mathbf{b}^T \mathbf{b} = 1$ , we have that

$$\begin{aligned} 1 &= \mathbf{b}^T \mathbf{b} \\ &= \alpha^T \Psi^T \Phi \beta \\ &= \sum_{i=1}^n \sum_{j=1}^n \alpha_i \beta_j \psi_i^T \phi_j \leq \mu(\mathbf{A}) \cdot \sum_{i=1}^n \sum_{j=1}^n |\alpha_i| \cdot |\beta_j|. \end{aligned} \quad (2.5)$$

Here we have exploited the definition of the coherence between the two bases. This inequality leads to

$$1 \leq \mu(\mathbf{A}) \cdot \sum_{i=1}^n \sum_{j=1}^n |\alpha_i| \cdot |\beta_j| = \mu(\mathbf{A}) \cdot \|\alpha\|_1 \|\beta\|_1. \quad (2.6)$$

This could be interpreted as yet another uncertainty principle for the  $\ell_1$ -norm case, suggesting that two representations cannot be jointly  $\ell_1$ -short. Indeed, using the relation between the geometric and the algebraic means ( $\forall a, b \geq 0, \sqrt{ab} \leq (a + b)/2$ ), we have



$$\|\alpha\|_1 \|\beta\|_1 \geq \frac{1}{\mu(\mathbf{A})} \Rightarrow \|\alpha\|_1 + \|\beta\|_1 \geq \frac{2}{\sqrt{\mu(\mathbf{A})}}. \quad (2.7)$$

However, this is a distraction from our goal, and we return to an  $\ell_0$ -based uncertainty principle.

Let us consider the following problem: Among all possible representations  $\alpha$  that satisfy  $\|\alpha\|_2 = 1$  and have  $A$  non-zeros (i.e.,  $\|\alpha\|_0 = A$ ), what would be the one giving the longest  $\ell_1$  length? This defines an optimization problem of the form

$$\max_{\alpha} \|\alpha\|_1 \text{ subject to } \|\alpha\|_2^2 = 1 \text{ and } \|\alpha\|_0 = A. \quad (2.8)$$

Suppose that this problem leads to a solution  $g(A) = g(\|\alpha\|_0)$ . Similarly, a parallel definition for  $\beta$  with  $B$  non-zeros gives a result  $g(\|\beta\|_0)$ . This means that using Equation (2.6) we have an inequality of the form

$$\frac{1}{\mu(\mathbf{A})} \leq \|\alpha\|_1 \|\beta\|_1 \leq g(\|\alpha\|_0) \cdot g(\|\beta\|_0), \quad (2.9)$$

since each  $\ell_1$  length is replaced with its upper-bound. Such an inequality is our target, and thus solution of the problem posed in Equation (2.8) is needed.

We can assume, without loss of generality, that the  $A$  non-zeros in  $\alpha$  are its first entries, the rest being zeros. We further assume that all of these entries are strictly positive (since only absolute values are used in this problem). Using Lagrange multipliers, the  $\ell_0$  constraint vanishes, and we obtain

$$\mathcal{L}(\alpha) = \sum_{i=1}^A \alpha_i + \lambda \left( 1 - \sum_{i=1}^A \alpha_i^2 \right). \quad (2.10)$$

The derivative of this Lagrangian is

$$\frac{\partial \mathcal{L}(\alpha)}{\partial \alpha_i} = 1 - 2\lambda \alpha_i = 0, \quad (2.11)$$

implying that the optimal values are given by  $\alpha_i = 1/2\lambda$ , and are all the same. This means that the optimal value (due to the  $\ell_2$  constraint) is  $\alpha_i = 1/\sqrt{A}$ , and thus  $g(A) = A/\sqrt{A} = \sqrt{A}$  is the value of the maximal  $\ell_1$ -norm of the vector  $\alpha$ . Using this and a parallel result for  $\beta$ , plugged into (2.9) leads to

$$\frac{1}{\mu(\mathbf{A})} \leq \|\alpha\|_1 \|\beta\|_1 \leq g(\|\alpha\|_0) \cdot g(\|\beta\|_0) = \sqrt{\|\alpha\|_0 \cdot \|\beta\|_0}. \quad (2.12)$$

Using again the geometric-algebraic mean relationship we obtain

$$\frac{1}{\mu(\mathbf{A})} \leq \sqrt{\|\alpha\|_0 \cdot \|\beta\|_0} \leq \frac{1}{2} (\|\alpha\|_0 + \|\beta\|_0), \quad (2.13)$$

as claimed.  $\square$

An alternative and simpler proof (by Allan Pinkus) is the following: Since  $\Phi$  and  $\Psi$  are unitary matrices, we have that  $\|\mathbf{b}\|_2 = \|\alpha\|_2 = \|\beta\|_2$ . Let us denote the support of  $\alpha$  by  $I$ . From  $\mathbf{b} = \Psi\alpha = \sum_{i \in I} \alpha_i \psi_i$  we have

$$\begin{aligned} |\beta_j|^2 &= |\mathbf{b}^T \phi_j|^2 = \left| \sum_{i \in I} \alpha_i \psi_i^T \phi_j \right|^2 \\ &\leq \|\alpha\|_2^2 \cdot \left| \sum_{i \in I} (\psi_i^T \phi_j)^2 \right| \\ &\leq \|\mathbf{b}\|_2^2 \cdot |I| \cdot \mu(\mathbf{A})^2, \end{aligned} \quad (2.14)$$

where we have used the Cauchy-Schwartz inequality,<sup>2</sup> and the definition of the mutual-coherence. Summing the above over all  $j \in J$ , the support of  $\beta$ , we obtain

$$\sum_{j \in J} |\beta_j|^2 = \|\mathbf{b}\|_2^2 \leq \|\mathbf{b}\|_2^2 \cdot |I| \cdot |J| \cdot \mu(\mathbf{A})^2. \quad (2.15)$$

This leads to the inequality we have posed in Equation (2.13), thus proving the theorem.  $\square$

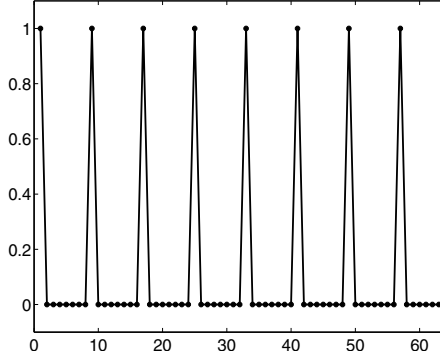
This result suggests that if the mutual-coherence of two bases is small, then  $\alpha$  and  $\beta$  cannot both be very sparse. For example, if, as above,  $\Psi$  is the identity and  $\Phi$  is the Fourier matrix, then  $\mu([\Psi, \Phi]) = 1/\sqrt{n}$ . It follows that a signal cannot have fewer than  $2\sqrt{n}$  nonzeros in both the time and frequency-domains. In such a case, we also know that this is a tight relationship, since the picket-fence signal with a uniform spacing of  $\sqrt{n}$  (assuming it is an integer) is converted by the Fourier transform (due to Poisson formula) to the same signal, thus accumulating  $2\sqrt{n}$  non-zeros. Figure 2.1 shows this signal.

Heisenberg's classical uncertainty principle, in the discrete setting, says that, if we view  $\alpha$  and  $\beta$  as probability distributions (by taking the absolute value of the entries and normalizing) then the product of their variances satisfies  $\sigma_\alpha^2 \sigma_\beta^2 \geq \text{const.}$  In contrast, (2.4) gives a lower bound on the sum of the nonzeros, regardless of their locations.

### 2.1.2 Uncertainty of Redundant Solutions

We now make a connection to the uniqueness problem. Consider the problem of finding a solution to  $\mathbf{Ax} = [\Psi, \Phi]\mathbf{x} = \mathbf{b}$  in light of the uncertainty principle (2.4). Suppose there are two solutions,  $\mathbf{x}_1, \mathbf{x}_2$  for the underlying linear system, and that one is very sparse. We will see that the other cannot be very sparse as well. Necessarily, the difference  $\mathbf{e} = \mathbf{x}_1 - \mathbf{x}_2$  must be in the null-space of  $\mathbf{A}$ . Partition  $\mathbf{e}$  into sub-vectors

<sup>2</sup> The Cauchy-Schwartz inequality is given by:  $|\mathbf{x}^T \mathbf{y}|^2 \leq \|\mathbf{x}\|_2^2 \cdot \|\mathbf{y}\|_2^2$ . Equality is obtained if and only if  $\mathbf{x}$  and  $\mathbf{y}$  are linearly-dependent.



**Fig. 2.1** The picket-fence signal for  $n = 64$ . It has 8 uniformly spread non-zeros with equal height. The Discrete-Fourier-Transform (DFT) of this signal looks exactly the same.

$\mathbf{e}_\psi$  and  $\mathbf{e}_\phi$  of the first  $n$  entries and last  $n$  entries of  $\mathbf{e}$ , respectively. We have

$$\Psi \mathbf{e}_\psi = -\Phi \mathbf{e}_\phi = \mathbf{y} \neq \mathbf{0}. \quad (2.16)$$

The vector  $\mathbf{y}$  is nonzero because  $\mathbf{e}$  is nonzero, and both  $\Psi$  and  $\Phi$  are nonsingular. Now invoke (2.4):

$$\|\mathbf{e}\|_0 = \|\mathbf{e}_\psi\|_0 + \|\mathbf{e}_\phi\|_0 \geq \frac{2}{\mu(\mathbf{A})}. \quad (2.17)$$

Since  $\mathbf{e} = \mathbf{x}_1 - \mathbf{x}_2$ , we have

$$(\text{Uncertainty Principle 2}) : \|\mathbf{x}_1\|_0 + \|\mathbf{x}_2\|_0 \geq \|\mathbf{e}\|_0 \geq \frac{2}{\mu(\mathbf{A})}. \quad (2.18)$$

Here we have used the triangle inequality for the  $\ell_0$  norm,  $\|\mathbf{x}_1\|_0 + \|\mathbf{x}_2\|_0 \geq \|\mathbf{x}_1 - \mathbf{x}_2\|_0$ , which is trivially verified, by counting the non-zeros of the two vectors and considering no overlap of supports (which leads to equality) and an overlap (which gives this inequality). To summarize, we have proven the following result:

**Theorem 2.2.** *Any two distinct solutions  $\mathbf{x}_1, \mathbf{x}_2$  of the linear system  $[\Psi, \Phi]\mathbf{x} = \mathbf{b}$  cannot both be very sparse, governed by the following uncertainty principle:*

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

We refer to this result as an uncertainty of redundant solutions, as we discuss here solutions to the underdetermined system.

### 2.1.3 From Uncertainty to Uniqueness

A direct consequence of inequality (2.18) is a uniqueness result:

**Theorem 2.3.** *If a candidate solution for  $[\Psi, \Phi]\mathbf{x} = \mathbf{b}$  has fewer than  $1/\mu(\mathbf{A})$  non-zeros, then it is necessarily the sparsest one possible, and any other solution must be “denser.”*

This seemingly simple claim is wonderful and powerful. At least for the special case discussed here,  $\mathbf{A} = [\Psi, \Phi]$ , we have a complete answer for the two questions we have posed at the beginning of this chapter. Namely, we can certainly claim uniqueness for sparse enough solutions, and once such a sufficiently sparse solution is given to us, we can immediately claim its global optimality. Notice that in general non-convex optimization problems, a given solution can at best be verified as being locally optimal, and here we have the ability to verify its optimality globally.

Since the discussion so far concentrated on the two-ortho case, it is now time to dare and address general matrices  $\mathbf{A}$ , using similar treatment. Clearly, though, an uncertainty result of the kind posed in Theorem 1 would be impossible to imitate, and we will have to bypass it somehow.

## 2.2 Uniqueness Analysis for the General Case

### 2.2.1 Uniqueness via the Spark

A key property that is crucial for the study of uniqueness is the *spark* of the matrix  $\mathbf{A}$ , a term coined and defined by Donoho and Elad in 2003. The *spark* is a way of characterizing the null-space of a matrix  $\mathbf{A}$  using the  $\ell_0$ -norm. We start with the following definition:

**Definition 2.2.** : The spark of a given matrix  $\mathbf{A}$  is the smallest number of columns from  $\mathbf{A}$  that are linearly-dependent.

Recall that the rank of a matrix is defined as the *largest* number of columns from  $\mathbf{A}$  that are linearly *independent*. Clearly, there is a resemblance between these two definitions – replace largest with smallest, and independent by dependent, and you return to the definition of the *spark*. Nevertheless, the *spark* of a matrix is far more difficult to obtain, compared to its rank, as it calls for a combinatorial search over all possible subsets of columns from  $\mathbf{A}$ .

The importance of this property of matrices for the study of the uniqueness of sparse solutions has been unraveled already by Rao and Gorodnitsky in 1998. Interestingly, this property has previously appeared in the literature of psychometrics (termed *Kruskal rank*), used in the context of studying uniqueness of tensor decomposition. The *spark* is also related to established notions in matroid theory; formally it is precisely the girth of the linear matroid defined by  $\mathbf{A}$ , i.e., the length of

the shortest cycle in that matroidmatroid. Finally, if we consider the same definition where the arithmetic underlying the matrix product is performed *not* over the fields of real or complex numbers but instead over the ring of integers mod  $q$ , the same quantity arises in coding theory, where it allows to compute the minimum distance of a code. The resemblance between all these concepts is striking and instructive.

The *spark* gives a simple criterion for uniqueness of sparse solutions. By definition, the vectors in the null-space of the matrix  $\mathbf{A}\mathbf{x} = \mathbf{0}$  must satisfy  $\|\mathbf{x}\|_0 \geq \text{spark}(\mathbf{A})$ , since these vectors combine linearly columns from  $\mathbf{A}$  to give the zero vector, and at least *spark* such columns are necessary by definition. Using the *spark* we obtain the following result:

**Theorem 2.4. (Uniqueness – Spark):** *If a system of linear equations  $\mathbf{A}\mathbf{x} = \mathbf{b}$  has a solution  $\mathbf{x}$  obeying  $\|\mathbf{x}\|_0 < \text{spark}(\mathbf{A})/2$ , this solution is necessarily the sparsest possible.*

**Proof:** Consider an alternative solution  $\mathbf{y}$  that satisfies the same linear system  $\mathbf{A}\mathbf{y} = \mathbf{b}$ . This implies that  $\mathbf{x} - \mathbf{y}$  must be in the null-space of  $\mathbf{A}$ , i.e.,  $\mathbf{A}(\mathbf{x} - \mathbf{y}) = \mathbf{0}$ . By definition of *spark*,

$$\|\mathbf{x}\|_0 + \|\mathbf{y}\|_0 \geq \|\mathbf{x} - \mathbf{y}\|_0 \geq \text{spark}(\mathbf{A}). \quad (2.19)$$

The left-most term in the above inequality simply states that the number of non-zeros in the difference vector  $\mathbf{x} - \mathbf{y}$  cannot exceed the sum of the number of non-zeros within each of the vectors  $\mathbf{x}$  and  $\mathbf{y}$  separately – this is the triangle inequality mentioned before. Since we have a solution satisfying  $\|\mathbf{x}\|_0 < \text{spark}(\mathbf{A})/2$ , we conclude that any alternative solution necessarily has more than  $\text{spark}(\mathbf{A})/2$  non-zeros, as claimed.  $\square$

Again, this result is very elementary and yet quite surprising, bearing in mind that  $(P_0)$  is a highly complicated optimization task of combinatorial flavor. In general combinatorial optimization problems, when considering a proposed solution, one hopes only to check local optimality – i.e., that no simple modification gives a better result. Here, we find that simply checking the solution sparsity, and comparing that with the *spark*, lets us check global optimality.

Clearly, the value of *spark* can be very informative, and large values of *spark* are evidently very useful. How large can *spark* be? By definition, *spark* must be in the range<sup>3</sup>  $2 \leq \text{spark}(\mathbf{A}) \leq n + 1$ . For example, if  $\mathbf{A}$  comprises random independent and identically distributed entries (say Gaussian), then with probability 1 we have  $\text{spark}(\mathbf{A}) = n + 1$ , implying that no  $n$  columns are linearly-dependent. The same *spark* is obtained for Vandermonde matrices constructed from distinct  $m$  scalars. In these cases, uniqueness is ensured for every solution with  $n/2$  or fewer non-zero entries. Similarly, the *spark* of the two-ortho identity-Fourier pair is<sup>4</sup>  $2\sqrt{n}$ , using

<sup>3</sup> The *spark* can be as low as 1 if there exists a zero-column in  $\mathbf{A}$ , but we do not consider such a case as relevant to our analysis.

<sup>4</sup> If  $n$  is prime, the *spark* of the identity-Fourier pair becomes  $n + 1$ , as the picket-fence signals are no longer relevant.

Poisson formula – a concatenation of two picket-fence signals, each with  $\sqrt{n}$  peaks evenly spread is the sparsest possible vector in the null-space of the matrix  $[\mathbf{I}, \mathbf{F}]$ .

### 2.2.2 Uniqueness via the Mutual-Coherence

The *spark* is at least as difficult to evaluate as solving  $(P_0)$ . Thus, simpler ways to guarantee uniqueness are of interest. A very simple way exploits the mutual-coherence of the matrix  $\mathbf{A}$ , defined by generalizing the definition given for the two-ortho case. In the two ortho case, computation of the Gram matrix  $\mathbf{A}^T \mathbf{A}$  leads to

$$\mathbf{A}^T \mathbf{A} = \begin{bmatrix} \mathbf{I} & \Psi^T \Phi \\ \Phi^T \Psi & \mathbf{I} \end{bmatrix}. \quad (2.20)$$

Thus, the above-defined mutual-coherence for this case is obtained as the maximal off-diagonal entry (in absolute value) in this Gram matrix. Similarly, we propose a generalization of this definition, as follows:

**Definition 2.3.** : The mutual-coherence of a given matrix  $\mathbf{A}$  is the largest absolute normalized inner product between different columns from  $\mathbf{A}$ . Denoting the  $k$ -th column in  $\mathbf{A}$  by  $\mathbf{a}_k$ , the mutual-coherence is given by

$$\mu(\mathbf{A}) = \max_{1 \leq i, j \leq m, i \neq j} \frac{|\mathbf{a}_i^T \mathbf{a}_j|}{\|\mathbf{a}_i\|_2 \cdot \|\mathbf{a}_j\|_2}. \quad (2.21)$$

The mutual-coherence is a way to characterize the dependence between columns of the matrix  $\mathbf{A}$ . For a unitary matrix, columns are pairwise orthogonal, and so the mutual-coherence is zero. For general matrices with more columns than rows,  $m > n$ ,  $\mu$  is necessarily strictly positive, and we desire the smallest possible value so as to get as close as possible to the behavior exhibited by unitary matrices.

We have seen that for structured two-ortho matrices  $\mathbf{A} = [\Psi, \Phi]$  the mutual-coherence satisfies  $1/\sqrt{n} \leq \mu(\mathbf{A}) \leq 1$ . When considering random orthogonal matrices of size  $n \times m$ , the work in Donoho and Huo has shown that they tend to be incoherent, implying that  $\mu(\mathbf{A}_{n,m})$  is typically proportional to  $\sqrt{\log(nm)/n}$  for  $n \rightarrow \infty$ . It has been shown that for full-rank matrices of size  $n \times m$  the mutual-coherence is bounded from below by

$$\mu \geq \sqrt{\frac{m-n}{n(m-1)}}.$$

Equality is obtained for a family of matrices named *Grassmannian Frames*. The set of columns in such matrices are called equiangular lines. Indeed, this family of matrices has  $\text{spark}(\mathbf{A}) = n + 1$ , the highest value possible. Numerical construction of such matrices has been addressed by Tropp et al. using an iterative projection

onto (sometimes not so) convex sets. We will return to this topic towards the end of this chapter.

We also mention work by Calderbank in quantum information theory, constructing error-correcting-codes using a collection of orthogonal bases with minimal coherence, obtaining similar bounds on the mutual-coherence for amalgams of orthogonal bases. Also related to this line of activity is the more recent contribution by Sochen, Gurevitz, and Hadani, on constructions of signal sets such that their shifts exhibit low coherence.

Mutual-coherence is relatively easy to compute, and as such, it allows us to lower-bound the *spark*, which is often hard to compute.

**Lemma 2.1.** *For any matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$ , the following relationship holds:*

$$\text{spark}(\mathbf{A}) \geq 1 + \frac{1}{\mu(\mathbf{A})}. \quad (2.22)$$

**Proof:** First, modify the matrix  $\mathbf{A}$  by normalizing its columns to be of unit  $\ell_2$ -norm, obtaining  $\tilde{\mathbf{A}}$ . 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:

$$\{G_{k,k} = 1 : 1 \leq k \leq m\} \text{ and } \{|G_{k,j}| \leq \mu(\mathbf{A}) : 1 \leq k, j \leq m, k \neq j\},$$

Consider an arbitrary leading minor from  $\mathbf{G}$  of size  $p \times p$ , built by choosing a subgroup of  $p$  columns from  $\tilde{\mathbf{A}}$  and computing their sub-Gram matrix. From the Gershgorin disk theorem,<sup>5</sup> if this minor is diagonally-dominant — i.e., if  $\sum_{j \neq i} |G_{i,j}| < |G_{i,i}|$  for every  $i$  — then this sub-matrix of  $\mathbf{G}$  is positive-definite, and so those  $p$  columns from  $\tilde{\mathbf{A}}$  are linearly-independent. The condition  $1 > (p-1)\mu \rightarrow p < 1 + 1/\mu$  implies positive-definiteness of every  $p \times p$  minor. Thus,  $p = 1 + 1/\mu$  is the smallest possible number of columns that might lead to linear dependence, and thus  $\text{spark}(\mathbf{A}) \geq 1 + 1/\mu$ .  $\square$

We have the following analog of the previous uniqueness theorems, and this time based on the mutual-coherence.

**Theorem 2.5. (Uniqueness – Mutual-Coherence):** *If a system of linear equations  $\mathbf{Ax} = \mathbf{b}$  has a solution  $\mathbf{x}$  obeying  $\|\mathbf{x}\|_0 < \frac{1}{2}(1 + 1/\mu(\mathbf{A}))$ , this solution is necessarily the sparsest possible.*

Compare Theorems 2.4 and 2.5. They are parallel in form, but with different assumptions. In general, Theorem 2.4, which uses *spark*, is sharp and is far more powerful than Theorem 2.5, which uses the coherence and so only a lower bound on *spark*. The coherence can never be smaller than  $1/\sqrt{n}$ , and therefore, the cardinality bound of Theorem 2.5 is never larger than  $\sqrt{n}/2$ . However, the *spark* can easily be as large as  $n$ , and Theorem 2.4 then gives a bound as large as  $n/2$ .

<sup>5</sup> For a general (possibly complex) matrix  $\mathbf{H}$  of size  $n \times n$ , Gershgorin's disks are the  $n$  disks formed by the centers  $h(i, i)$  and radiuses  $\sum_{j \neq i} |h(i, j)|$ . The theorem states that all eigenvalues of  $\mathbf{H}$  must lie within the union of these disks.

In fact, for the special matrix  $\mathbf{A} = [\Psi, \Phi]$  we have also obtained such a rule. Interestingly, the lower bound for the general case becomes  $(1 + 1/\mu(\mathbf{A}))/2$  while the special two-ortho case gave a stronger (i.e., higher) lower bound. The general case bound is nearly a factor of 2 weaker than (2.18), because (2.18) uses the special structure  $\mathbf{A} = [\Psi, \Phi]$ .

### 2.2.3 Uniqueness via the Babel Function

In the proof of Lemma 2.1 we considered minors of size  $p \times p$  extracted from the Gram matrix of the normalized matrix  $\tilde{\mathbf{A}}$ . The positive-definiteness of all such minors imply that every  $p$  columns are linearly-independent. However, for simplification of the analysis we bounded all the off-diagonal entries of  $\mathbf{G}$  by a single value  $\mu(\mathbf{A})$ , and thus, lost robustness to possibly few extreme entries in this matrix.

Since we need to check whether the sum of the  $p - 1$  off-diagonal entries in every row in these minors is less than 1 (for the Gershgorin property to hold true), we can define the following *Babel function*, following Tropp:

**Definition 2.4.** For a given matrix  $\tilde{\mathbf{A}}$  with normalized columns, we consider a subset  $\Lambda$  of  $p$  columns from  $\tilde{\mathbf{A}}$ , and compute the sum of the absolute values of their inner product with a column outside this set. Maximizing over both the set  $\Lambda$  and the outside column  $j$  we obtain the Babel function:

$$\mu_1(p) = \max_{\Lambda, |\Lambda|=p} \max_{j \notin \Lambda} \sum_{i \in \Lambda} |\tilde{\mathbf{a}}_i^T \tilde{\mathbf{a}}_j|. \quad (2.23)$$

Clearly, for  $p = 1$ , we get that  $\mu_1(1) = \mu(\mathbf{A})$ . For  $p = 2$  the above implies that we need to sweep through all possible triplets, considering two as those belonging to  $\Lambda$ , and the third as the external vector to compute inner products with. This definition implies that this function is monotonically non-decreasing, and the slower its growth the better the analysis it leads to, compared to the use of the cruder coherence measure.

It might seem that computing this function for large  $p$  becomes exponential and thus prohibitive, but this is in fact not true. By computing  $|\mathbf{G}| = |\tilde{\mathbf{A}}^T \tilde{\mathbf{A}}|$ , and sorting every row in descending order, we obtain the matrix  $\mathbf{G}_S$ . The first entry in every row is 1, being the main diagonal entry, and thus should be disregarded. Computing the sum of the  $p$  leading entries in every row (second and beyond), we get for every  $j$  in the above definition the set of worst  $\Lambda$ 's, and among them we should choose the maximal one, thus

$$\mu_1(p) = \max_{1 \leq j \leq m} \sum_{i=2}^{p+1} |G_S(j, i)|. \quad (2.24)$$



We note that for every  $p$  we have that  $\mu_1(p) \leq p \cdot \mu(\mathbf{A})$ , and the two become equal for Grassmannian matrices, where all the off-diagonal entries in the Gram matrix are of the same magnitude.

How can we use the *Babel function* for better assessing the uniqueness? It is clear that if  $\mu_1(p) < 1$ , we deduce that all  $p + 1$  sets are linearly-independent. Thus, a lower-bound on the *spark* could be

$$\text{spark}(\mathbf{A}) \geq \min_{1 \leq p \leq n} \{p \mid \mu_1(p - 1) \geq 1\}. \quad (2.25)$$

The uncertainty and uniqueness properties follow immediately.

### 2.2.4 Upper-Bounding the Spark

The *spark* is impossible to evaluate in general, as it is even harder than solving  $(P_0)$ . This is because its evaluation requires a sweep through all possible groups of columns from  $\mathbf{A}$  with varying cardinalities, seeking for a linearly-dependent subset of columns. This is a combinatorial search of exponential complexity with  $m$ .

While this difficulty explains the necessity to replace the *spark* with the mutual-coherence, the price paid in loss of tightness of the uniqueness result may be considered as too dear to be permitted. This motivates an alternative method for approximating the *spark*, and this time using an upper bound. Such a bound implies that we cannot guarantee uniqueness based on the obtained value, but it would give us a rough evaluation of the region of uniqueness.

In order to develop the upper-bound, we redefine the *spark* as the outcome of  $m$  optimization problems  $(P_0^i)$  for  $i = 1, 2, \dots, m$  of the form:

$$(P_0^i) : \quad \mathbf{x}_{opt}^i = \arg \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to } \mathbf{A}\mathbf{x} = \mathbf{0} \text{ and } x_i = 1. \quad (2.26)$$

Each of these problems assumes that the sparsest vector in the null-space of  $\mathbf{A}$  uses the  $i$ -th entry. By solving this sequence of  $(P_0)$ -like problems, the sparsest result among  $\{\mathbf{x}_{opt}^i\}_{i=1}^m$  gives the *spark*,

$$\text{spark}(\mathbf{A}) = \min_{1 \leq i \leq m} \|\mathbf{x}_{opt}^i\|_0. \quad (2.27)$$

Since the set of problems  $(P_0^i)$  is too complex, we define an alternative set of problems, replacing the  $\ell_0$  with the  $\ell_1$ -norm,

$$(P_1^i) : \quad \mathbf{z}_{opt}^i = \arg \min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{subject to } \mathbf{A}\mathbf{x} = \mathbf{0} \text{ and } x_i = 1. \quad (2.28)$$

As we have seen in Chapter 1, these problems have a linear programming structure, they are convex, and solvable in reasonable time. Furthermore, it is clear that for every  $i$ , we have that  $\|\mathbf{x}_{opt}^i\|_0 \leq \|\mathbf{z}_{opt}^i\|_0$ , since  $\|\mathbf{x}_{opt}^i\|_0$  is the sparsest solution possible for this problem by definition, and thus we have the promised bound

$$\text{spark}(\mathbf{A}) \leq \min_{1 \leq i \leq m} \|\mathbf{z}_{opt}^i\|_0. \quad (2.29)$$

Numerical experiments show that this bound tends to be quite tight, and close to the true *spark*.

## 2.3 Constructing Grassmannian Matrices

A Grassmannian (real) matrix  $\mathbf{A}$  of size  $n \times m$  with  $m \geq n$  is a matrix with normalized columns such that its Gram matrix  $\mathbf{G} = \mathbf{A}^T \mathbf{A}$  satisfies

$$\forall k \neq j, |G_{k,j}| = \sqrt{\frac{m-n}{n(m-n+1)}}. \quad (2.30)$$

As we have stated above, this is the smallest possible mutual-coherence possible. Such matrices do not necessarily exist, and in particular, they are possible only if  $m < \min(n(n+1)/2, (m-n)(m-n+1)/2)$ .

Grassmannian matrices are special in the sense that the angle between each and every pair of columns in it is the same, and it is also the smallest possible. Thus, construction of such matrices has a strong connection with packing of vectors/subspaces in the  $\mathbb{R}^n$ -space. While the case  $m = n$  leads trivially to unitary matrices that are easy to construct, constructing a general Grassmannian matrix is very hard. A numerical algorithm for this task was proposed by Tropp et al. and we bring it here as an illustration of such design procedures. We should add that while image processing finds little interest in such constructions, they find important application in channel coding, wireless communication, and more.

The key idea in the proposed algorithm is to iterate between projections onto the requirements such matrix should satisfy. Starting with an arbitrary matrix  $\mathbf{A}$ , these projections should refer to the following requirements:

1. The columns in  $\mathbf{A}$  are  $\ell_2$ -normalized: This can be forced by normalizing every column.
2. Property (2.30): We should compute  $\mathbf{G} = \mathbf{A}^T \mathbf{A}$ , detect the off-diagonal entries that are above some threshold, and decrease them. Similarly, since too small values are also not permitted in such Gram matrices, one might also add an increase to such off-diagonal entries.
3. The rank of  $\mathbf{G}$  should not exceed  $n$ : Since the above modifications cause  $\mathbf{G}$  to become a full-rank one, an SVD operation and truncation of the singular-values beyond the first  $n$  ones brings the new Gram matrix to the proper rank.

This process can and should be repeated many times. There is no guarantee for convergence, or arrival at a Grassmannian matrix, but tests show that one can get closer to such matrices with this numerical scheme. A Matlab code that follows these guidelines is given in [Figure 2.2](#).

```

D=randn(N,L); % initialization
D=D*diag(1./sqrt(diag(D'*D))); % normalize columns
G=D'*D; % compute the Gram matrix
mu=sqrt((L-N)/N/(L-1));
for k=1:1:Iter,
    % shrink the high inner products
    gg=sort(abs(G(:)));
    pos=find(abs(G(:))>gg(round(dd1*(L*L-L))) & abs(G(:)) < 1;
    G(pos)=G(pos)*dd2;
    % reduce the rank back to N
    [U,S,V]=svd(G);
    S(N+1:end,1+N:end)=0;
    G=U*S*V';
    % Normalize the columns
    G=diag(1./sqrt(diag(G)))*G*diag(1./sqrt(diag(G)));
    % Show status
    gg=sort(abs(G(:)));
    pos=find(abs(G(:))>gg(round(dd1*(L*L-L))) & abs(G(:)) < 1;
    disp([k,mu,mean(abs(G(pos))),max(abs(G(pos)))]);
end;
[U,S,V]=svd(G);
D=sqrt(S(1:N,1:N))*U(:,1:N);

```

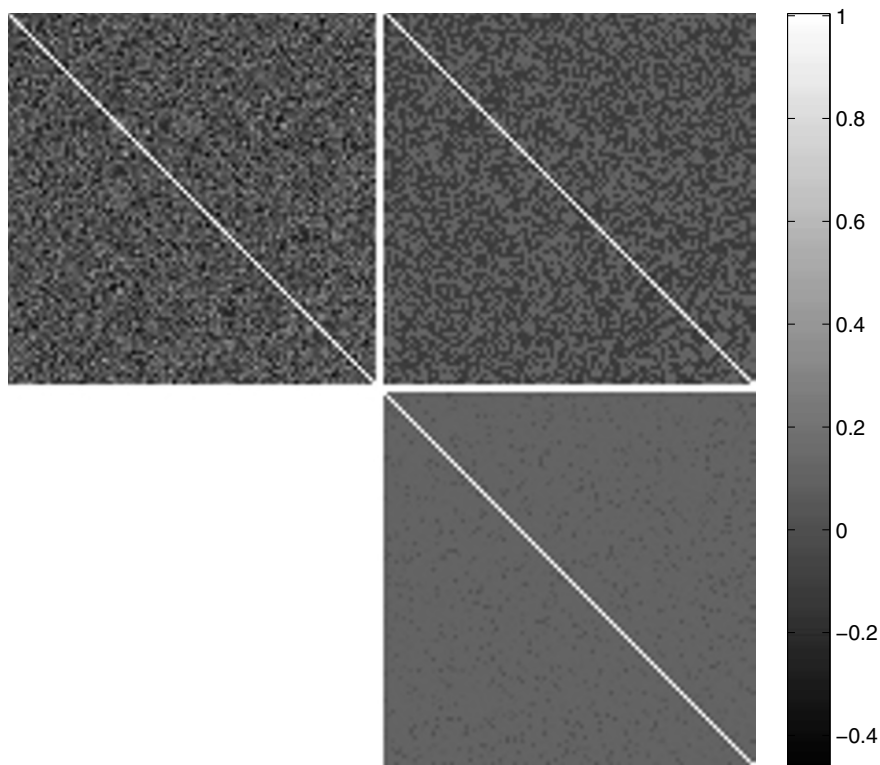
**Fig. 2.2** Matlab code for building a Grassmannian matrix.

Figure 2.3 shows the results for this procedure.<sup>6</sup> For a matrix of size  $50 \times 100$ , the minimal possible coherence is  $\sqrt{1/99} = 0.1005$ . Figure 2.3 shows the initial Gram matrix and the one obtained after 10,000 iterations, using  $dd1 = dd2 = 0.9$ . Figure 2.4 shows the sorted off-diagonal entries in these two Gram matrices, and as can be seen, the iterative procedure succeeds very well in obtaining a matrix very close to a Grassmannian one. In fact, in this run, the maximal off-diagonal entry is 0.119. Figure 2.5 presents the mutual-coherence as it evolves through the iterations of the algorithm.

## 2.4 Summary

We have now given answers to the questions posed at the start of this chapter. We have seen that any sufficiently sparse solution is guaranteed to be unique among all possible solutions. Consequently, any sufficiently sparse solution is necessarily the global optimizer of  $(P_0)$ . These results show that searching for a sparse solution can lead to a well-posed question with interesting properties. We now turn to discuss practical methods for obtaining solutions for  $(P_0)$ .

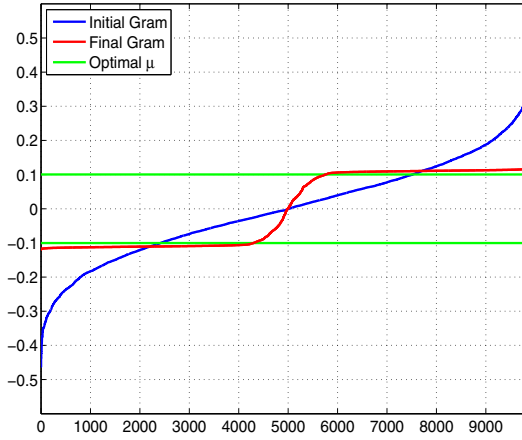
<sup>6</sup> Courtesy of Joel Tropp.



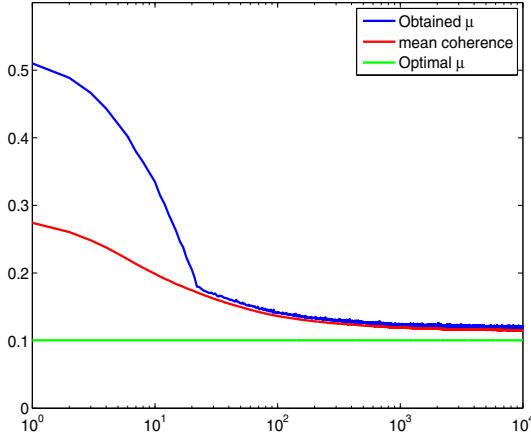
**Fig. 2.3** The initial Gram matrix (top-left) and the final one (top-right) in training a Grassmannian matrix of size  $50 \times 100$ . The bottom part of the figure shows the absolute of the final Gram matrix, showing that the off-diagonal elements tend to have the same value, as required.

## Further Reading

1. A.R. Calderbank and P.W. Shor, Good quantum error-correcting codes exist, *Phys. Rev. A*, 54(2):1098–1105, August 1996.
2. D.L. Donoho and M. Elad, Optimally sparse representation in general (non-orthogonal) dictionaries via  $\ell_1$  minimization, *Proc. of the National Academy of Sciences*, 100(5):2197–2202, 2003.
3. D.L. Donoho and X. Huo, Uncertainty principles and ideal atomic decomposition, *IEEE Trans. On Information Theory*, 47(7):2845–2862, 1999.
4. D.L. Donoho and P.B. Starck, Uncertainty principles and signal recovery, *SIAM Journal on Applied Mathematics*, 49(3):906–931, June, 1989.



**Fig. 2.4** The sorted off-diagonal entries in the initial Gram matrix and the final one.



**Fig. 2.5** The evolved *mutual-coherence* of the matrix as a function of the iteration.

5. M. Elad and A.M. Bruckstein, A generalized uncertainty principle and sparse representation in pairs of bases, *IEEE Trans. On Information Theory*, 48:2558–2567, 2002.
6. I.F. Gorodnitsky and B.D. Rao, Sparse signal reconstruction from limited data using FOCUSS: A re-weighted norm minimization algorithm, *IEEE Trans. On Signal Processing*, 45(3):600–616, 1997.
7. S. Gurevich, R. Hadani, and N. Sochen, The finite harmonic oscillator and its associated sequences *Proc. Natl. Acad. Sci. USA*, 105(29):9869–9873, July, 2008.

8. S. Gurevich, R. Hadani, and N. Sochen, On some deterministic dictionaries supporting sparsity, *Journal of Fourier Analysis and Applications*, 14(5-6):859–876, December, 2008.
9. R. Gribonval and M. Nielsen, Sparse decompositions in unions of bases, *IEEE Trans. on Information Theory*, 49(12):3320–3325, 2003.
10. W. Heisenberg, The physical principles of the quantum theory, (C. Eckart and F.C. Hoyt, trans.), University of Chicago Press, Chicago, IL, 1930.
11. R.A. Horn C.R. Johnson, *Matrix Analysis*, New York: Cambridge University Press, 1985.
12. X. Huo, Sparse Image representation Via Combined Transforms, PhD thesis, Stanford, 1999.
13. J.B. Kruskal, Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics, *Linear Algebra and its Applications*, 18(2):95–138, 1977.
14. Title: P.W.H. Lemmens and J.J. Seidel, Equiangular lines, *Journal of Algebra*, 24(3):494–512, 1973.
15. X. Liu and N.D. Sidiropoulos, Cramer-Rao lower bounds for low-rank decomposition of multidimensional arrays, *IEEE Trans. on Signal Processing*, 49(9):2074–2086, 2001.
16. B.K. Natarajan, Sparse approximate solutions to linear systems, *SIAM Journal on Computing*, 24:227–234, 1995.
17. W.W. Peterson and E.J. Weldon, Jr., *Error-Correcting Codes*, 2nd edition, MIT Press: Cambridge, Mass., 1972.
18. A. Pinkus, *N-Width in Approximation Theory*, Springer, Berlin, 1985.
19. T. Strohmer and R. W. Heath, Grassmannian frames with applications to coding and communication, *Applied and Computational Harmonic Analysis*, 14:257–275, 2004.
20. J.A. Tropp, Greed is good: Algorithmic results for sparse approximation, *IEEE Trans. On Information Theory*, 50(10):2231–2242, October 2004.
21. J.A. Tropp, I.S. Dhillon, R.W. Heath Jr., and T. Strohmer, Designing structured tight frames via alternating projection, *IEEE Trans. Info. Theory*, 51(1):188–209, January 2005.



## Chapter 3

# Pursuit Algorithms – Practice

It is now time to consider reliable and efficient methods for solving  $(P_0)$ , as a straightforward approach seems hopeless. We now discuss methods which, it seems, have no hope of working – but which, under specific conditions, will work. Looking at the problem  $(P_0)$ ,

$$(P_0) : \quad \min_{\mathbf{x}} \quad \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{b} = \mathbf{A}\mathbf{x},$$

one observes that the unknown  $\mathbf{x}$  is composed of two effective parts to be found – the support of the solution, and the non-zero values over this support. Thus, one way to attack the numerical solution of  $(P_0)$  is to focus on the support, with the understanding that once found, the non-zero values of  $\mathbf{x}$  are easily detected by plain Least-Squares. As the support is discrete in nature, algorithms that seek it are discrete as well. This line of reasoning leads to the family of greedy algorithms that will be presented hereafter.

An alternative view of  $(P_0)$  can disregard the support, and consider the unknown as a vector  $\mathbf{x} \in \mathbb{R}^m$  over the continuum. Smoothing the penalty function  $\|\mathbf{x}\|_0$ , one can adopt a continuous optimization point of view for solving  $(P_0)$ . The relaxation methods that are presented later in this chapter adopt this view, by smoothing the  $\ell_0$ -norm in various forms, and handling the revised problem as a smooth optimization.

### 3.1 Greedy Algorithms

#### 3.1.1 The Core Idea

Suppose that the matrix  $\mathbf{A}$  has  $\text{spark}(\mathbf{A}) > 2$ , and the optimization problem  $(P_0)$  has value  $\text{val}(P_0) = 1$  (at the optimal solution), so  $\mathbf{b}$  is a scalar multiple of some column of the matrix  $\mathbf{A}$ , and this solution is known to be unique. We can identify this column by applying  $m$  tests – one per column of  $\mathbf{A}$ . The  $j$ -th test can be done



by minimizing  $\epsilon(j) = \|\mathbf{a}_j z_j - \mathbf{b}\|_2$ , leading to  $z_j^* = \mathbf{a}_j^T \mathbf{b} / \|\mathbf{a}_j\|_2^2$ . Plugged back into the error expression, the error becomes

$$\begin{aligned} \epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{b}\|_2^2 &= \left\| \frac{\mathbf{a}_j^T \mathbf{b}}{\|\mathbf{a}_j\|_2^2} \mathbf{a}_j - \mathbf{b} \right\|_2^2 \\ &= \|\mathbf{b}\|_2^2 - 2 \frac{(\mathbf{a}_j^T \mathbf{b})^2}{\|\mathbf{a}_j\|_2^2} + \frac{(\mathbf{a}_j^T \mathbf{b})^2}{\|\mathbf{a}_j\|_2^2} \\ &= \|\mathbf{b}\|_2^2 - \frac{(\mathbf{a}_j^T \mathbf{b})^2}{\|\mathbf{a}_j\|_2^2}. \end{aligned} \quad (3.1)$$

If this error is zero, we have found the proper solution. Thus the test to be done is simply  $\|\mathbf{a}_j\|_2^2 \|\mathbf{b}\|_2^2 = (\mathbf{a}_j^T \mathbf{b})^2$  (which is equivalent to the statement that the Cauchy-Schwartz inequality should be satisfied with equality), indicating that  $\mathbf{b}$  and  $\mathbf{a}_j$  are parallel. This procedure requires order  $O(mn)$  flops, which may be considered reasonable.

Proceeding with the same rationale, suppose that  $\mathbf{A}$  has  $\text{spark}(\mathbf{A}) > 2k_0$ , and the optimization problem is known to have value  $\text{val}(P_0) = k_0$ . Then  $\mathbf{b}$  is a linear combination of at most  $k_0$  columns of  $\mathbf{A}$ . Generalizing the previous solution, one might think to enumerate all  $\binom{m}{k_0} = O(m^{k_0})$  subsets of  $k_0$  columns from  $\mathbf{A}$ , and test each. Enumeration takes  $O(m^{k_0} n k_0^2)$  flops, which seems prohibitively slow in many settings.

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, at each stage, expanding that set by one additional column. The column chosen at each stage maximally reduces the residual  $\ell_2$  error in approximating  $\mathbf{b}$  from the currently active columns. After constructing an approximant including the new column, the residual  $\ell_2$  error is evaluated; if it now falls below a specified threshold, the algorithm terminates.

### 3.1.2 The Orthogonal-Matching-Pursuit

Figure 3.1 presents a formal description of this strategy, and its associated notation. This procedure is known in the literature of signal processing by the name *Orthogonal-Matching-Pursuit* (OMP), but is very well known (and was used much earlier) by other names in other fields – see below.

Note that the *Sweep* stage gives error values of the following form, which is very similar to what we have seen above in Equation (3.1),

$$\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{r}^{k-1}\|_2^2 = \left\| \frac{\mathbf{a}_j^T \mathbf{r}^{k-1}}{\|\mathbf{a}_j\|_2^2} \mathbf{a}_j - \mathbf{r}^{k-1} \right\|_2^2 \quad (3.2)$$

**Task:** Approximate the solution of  $(P_0)$ :  $\min_{\mathbf{x}} \|\mathbf{x}\|_0$  subject to  $\mathbf{Ax} = \mathbf{b}$ .

**Parameters:** We are given the matrix  $\mathbf{A}$ , the vector  $\mathbf{b}$ , and the error threshold  $\epsilon_0$ .

**Initialization:** Initialize  $k = 0$ , and set

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

**Main Iteration:** Increment  $k$  by 1 and perform the following steps:

- **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  $z_j^* = \mathbf{a}_j^T \mathbf{r}^{k-1} / \|\mathbf{a}_j\|_2^2$ .
- **Update Support:** Find a minimizer,  $j_0$  of  $\epsilon(j)$ :  $\forall j \notin S^{k-1}, \epsilon(j_0) \leq \epsilon(j)$ , and update  $S^k = S^{k-1} \cup \{j_0\}$ .
- **Update Provisional Solution:** Compute  $\mathbf{x}^k$ , the minimizer of  $\|\mathbf{Ax} - \mathbf{b}\|_2^2$  subject to  $\text{Support}\{\mathbf{x}\} = S^k$ .
- **Update Residual:** Compute  $\mathbf{r}^k = \mathbf{b} - \mathbf{Ax}^k$ .
- **Stopping Rule:** If  $\|\mathbf{r}^k\|_2 < \epsilon_0$ , stop. Otherwise, apply another iteration.

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

**Fig. 3.1** Orthogonal-Matching-Pursuit – a greedy algorithm for approximating the solution of  $(P_0)$ .

$$\begin{aligned}
 &= \|\mathbf{r}^{k-1}\|_2^2 - 2 \frac{(\mathbf{a}_j^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_j\|_2^2} + \frac{(\mathbf{a}_j^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_j\|_2^2} \\
 &= \|\mathbf{r}^{k-1}\|_2^2 - \frac{(\mathbf{a}_j^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_j\|_2^2}.
 \end{aligned}$$

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 vectors of the matrix  $\mathbf{A}$ .

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

$$\mathbf{A}_{S^k}^T (\mathbf{A}_{S^k} \mathbf{x}_{S^k} - \mathbf{b}) = -\mathbf{A}_{S^k}^T \mathbf{r}^k = 0. \quad (3.3)$$

Here we have used the formula of the residual in the  $k$ -th iteration,  $\mathbf{r}^k = \mathbf{b} - \mathbf{Ax}^k = \mathbf{b} - \mathbf{A}_{S^k} \mathbf{x}_{S^k}$ . The above relation suggests that the columns in  $\mathbf{A}$  that are part of the support  $S^k$  are necessarily orthogonal to the residual  $\mathbf{r}^k$ . This in turn implies that in the next iteration (and afterwards), these columns will not be chosen again for the support. This orthogonalization is the reason for the name of the algorithm — Orthogonal-Matching-Pursuit.

A more complex and somewhat better behaving variant of the above algorithm can be suggested, where each of the tests in the sweep stage is done as full Least-Squares (LS), considering all the accumulated columns and the candidate one together, and solving for all the coefficients at once. As in the OMP,  $m - |\mathcal{S}^{k-1}|$  LS steps are performed at the  $k$ -th step. This method is known as LS-OMP. The LS steps can be made efficient using the formula

$$\begin{bmatrix} \mathbf{M} & \mathbf{b} \\ \mathbf{b}^T & c \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{M}^{-1} + p\mathbf{M}^{-1}\mathbf{b}\mathbf{b}^T\mathbf{M}^{-1} & -p\mathbf{M}^{-1}\mathbf{b} \\ -p\mathbf{b}^T\mathbf{M}^{-1} & p \end{bmatrix}, \quad (3.4)$$

where  $p = 1/(c - \mathbf{b}^T\mathbf{M}^{-1}\mathbf{b})$ . Denoting the sub-matrix that gathers the  $k - 1$  chosen columns from  $\mathbf{A}$  as  $\mathbf{A}_s$ , the LS problem we should solve is

$$\min_{\mathbf{x}_s, z} \left\| \begin{bmatrix} \mathbf{A}_s & \mathbf{a}_i \end{bmatrix} \begin{bmatrix} \mathbf{x}_s \\ z \end{bmatrix} - \mathbf{b} \right\|_2^2. \quad (3.5)$$

The solution is given by zeroing the derivative of the above expression

$$\begin{bmatrix} \mathbf{A}_s^T \\ \mathbf{a}_i^T \end{bmatrix} \begin{bmatrix} \mathbf{A}_s & \mathbf{a}_i \end{bmatrix} \begin{bmatrix} \mathbf{x}_s \\ z \end{bmatrix} - \begin{bmatrix} \mathbf{A}_s^T \\ \mathbf{a}_i^T \end{bmatrix} \mathbf{b} = \mathbf{0}, \quad (3.6)$$

leading to

$$\begin{bmatrix} \mathbf{x}_s \\ z \end{bmatrix}_{opt} = \begin{bmatrix} \mathbf{A}_s^T \mathbf{A}_s & \mathbf{A}_s^T \mathbf{a}_i \\ \mathbf{a}_i^T \mathbf{A}_s & \|\mathbf{a}_i\|_2^2 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{A}_s^T \mathbf{b} \\ \mathbf{a}_i^T \mathbf{b} \end{bmatrix}. \quad (3.7)$$

Thus, based on the formula given in Equation (3.4), if we have stored the matrix  $(\mathbf{A}_s^T \mathbf{A}_s)^{-1}$ , then the above formula becomes easy, and leads to an update of this matrix for  $k$  columns. Once the solution is found, the residual error should be evaluated, and the column  $\mathbf{a}_i$  that leads to the smallest error should be chosen. Observe that if  $\mathbf{a}_i$  is orthogonal to the columns of  $\mathbf{A}_s$ , the above matrix to be inverted becomes block-diagonal, and the result would be  $\mathbf{x}_s = \mathbf{A}_s^+ \mathbf{b}$  and  $z = \mathbf{a}_i^T \mathbf{b} / \|\mathbf{a}_i\|_2^2$ . This means that the coefficients of the previous solution remain intact, and a new non-zero is introduced with a coefficient  $z$ .

There are numerical shortcuts for the evaluation of the residuals, but we shall not explore this matter here. Note that the above recursive approach towards the Least-Squares task in the LS-OMP is also relevant for speeding up the OMP, implying that the *Update Provisional Solution* stage can be made far more efficient.

If the delivered approximation has  $k_0$  non-zeros, the LS-OMP and OMP methods require  $O(k_0 mn)$  flops in general; this can be dramatically better than the exhaustive search, which requires  $O(nm^{k_0} k_0^2)$  flops. Thus, the single-term-at-a-time strategy can be much more efficient than exhaustive search – if it works! The strategy can fail badly, i.e., there are explicit examples constructed by Vladimir Temlyakov where a simple  $k$ -term representation is possible, but this approach yields an  $n$ -term (i.e., dense) representation. In general, all that can be said is that among single-term-at-a-time strategies, the approximation error is always reduced by as much as

possible, given the starting approximation and the single-term-at-a-time constraint. This property has earned this algorithm the name “Greedy algorithm” in approximation theory.

### 3.1.3 Other Greedy Methods

Many variants on the above algorithm are available, offering improvements either in accuracy and/or in complexity. This family of greedy algorithms is well known and extensively used, and in fact, these algorithms have been re-invented in various fields. In the setting of statistical modeling, greedy stepwise Least-Squares is called *forward stepwise regression*, and has been widely practiced since at least the 1960s. When used in the signal processing setting this goes by the name of *Matching-Pursuit* (MP) or *Orthogonal-Matching-Pursuit* (OMP). Approximation theorists refer to these algorithms as *Greedy Algorithms* (GA), and consider several variants of them – the Pure (PGA), the Orthogonal (OGA), the Relaxed (RGA), and the Weak Greedy Algorithm (WGA).

The MP algorithm is similar to the OMP, but with an important difference that makes it simpler, and thus less accurate. In the main iteration, 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  $\mathcal{S}^{k-1}$  original entries remain unchanged, and the new coefficient that refers to the new member  $j_0 \in \mathcal{S}^k$  is chosen as  $z_{j_0}^*$ . This algorithm is described in [Figure 3.2](#).

The Weak-MP is a further simplification of the MP algorithm, allowing for a sub-optimal choice of the next element to be added to the support. Embarking from the MP algorithm description in [Figure 3.2](#), the *update support* is relaxed by choosing any index that is factor  $t$  (in the range  $(0, 1]$ ) away from the optimal choice. This algorithm is described in [Figure 3.3](#).

Let us explain and motivate this method: We have seen the equivalence between the inner products computed,  $|\mathbf{a}_j^T \mathbf{r}^{k-1}|$  (up to a normalization factor), and the corresponding errors  $\epsilon(j)$  among which we seek the minimum. Rather than searching for the largest inner-product value, we settle for the first found that exceeds a  $t$ -weaker threshold. The Cauchy-Schwartz inequality gives

$$\frac{(\mathbf{a}_j^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_j\|_2^2} \leq \max_{1 \leq j \leq m} \frac{(\mathbf{a}_j^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_j\|_2^2} \leq \|\mathbf{r}^{k-1}\|_2^2. \quad (3.8)$$

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

$$\frac{(\mathbf{a}_{j_0}^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_{j_0}\|_2^2} \geq t^2 \cdot \|\mathbf{r}^{k-1}\|_2^2 \geq t^2 \cdot \max_{1 \leq j \leq m} \frac{(\mathbf{a}_j^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}_j\|_2^2}. \quad (3.9)$$

**Task:** Approximate the solution of  $(P_0)$ :  $\min_{\mathbf{x}} \|\mathbf{x}\|_0$  subject to  $\mathbf{Ax} = \mathbf{b}$ .  
**Parameters:** We are given the matrix  $\mathbf{A}$ , the vector  $\mathbf{b}$ , and the error threshold  $\epsilon_0$ .  
**Initialization:** Initialize  $k = 0$ , and set

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

**Main Iteration:** Increment  $k$  by 1 and perform the following steps:

- **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  $z_j^* = \mathbf{a}_j^T \mathbf{r}^{k-1} / \|\mathbf{a}_j\|_2^2$ .
- **Update Support:** Find a minimizer,  $j_0$  of  $\epsilon(j)$ :  $\forall 1 \leq j \leq m, \epsilon(j_0) \leq \epsilon(j)$ , and update  $S^k = S^{k-1} \cup \{j_0\}$ .
- **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:** Compute  $\mathbf{r}^k = \mathbf{b} - \mathbf{Ax}^k = \mathbf{r}^{k-1} - z_{j_0}^* \mathbf{a}_{j_0}$ .
- **Stopping Rule:** If  $\|\mathbf{r}^k\|_2 < \epsilon_0$ , stop. Otherwise, apply another iteration.

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

**Fig. 3.2** Matching-Pursuit for approximating the solution of  $(P_0)$ .

**Task:** Approximate the solution of  $(P_0)$ :  $\min_{\mathbf{x}} \|\mathbf{x}\|_0$  subject to  $\mathbf{Ax} = \mathbf{b}$ .  
**Parameters:** We are given the matrix  $\mathbf{A}$ , the vector  $\mathbf{b}$ , the error threshold  $\epsilon_0$ , and the scalar  $0 < t < 1$ .  
**Initialization:** Initialize  $k = 0$ , and set

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

**Main Iteration:** Increment  $k$  by 1 and perform the following steps:

- **Sweep:** Compute the errors  $\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{r}^{k-1}\|_2^2$  for  $j = 1, 2, \dots$  using the optimal choice  $z_j^* = \mathbf{a}_j^T \mathbf{r}^{k-1} / \|\mathbf{a}_j\|_2^2$ . Stop the sweep when  $|\mathbf{a}_j^T \mathbf{r}^{k-1}| / \|\mathbf{a}_j\|_2 \geq t \cdot \|\mathbf{r}^{k-1}\|_2$ .
- **Update Support:** Update  $S^k = S^{k-1} \cup \{j_0\}$ , 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  $\mathbf{r}^k = \mathbf{b} - \mathbf{Ax}^k = \mathbf{r}^{k-1} - z_{j_0}^* \mathbf{a}_{j_0}$ .
- **Stopping Rule:** If  $\|\mathbf{r}^k\|_2 < \epsilon_0$ , stop. Otherwise, apply another iteration.

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

**Fig. 3.3** Weak-Matching-Pursuit for approximating the solution of  $(P_0)$ .

for a pre-chosen  $t$  in the range  $(0, 1)$ . The above condition can be written alternatively as

$$(\mathbf{a}_{j_0}^T \mathbf{r}^{k-1})^2 \geq t^2 \cdot \|\mathbf{r}^{k-1}\|_2^2 \cdot \|\mathbf{a}_{j_0}\|_2^2. \quad (3.10)$$

This way, the chosen index clearly points to a column that is at the most factor  $t$  away from the possible maximum. Note that it is possible that the complete sweep is performed without any index satisfying the above condition, and then we simply choose the maximum that was found as a by-product of the search. This way, the search could become much faster, with the understanding that the rate-of-decay of the residual has been somewhat compromised.

### 3.1.4 Normalization

All the above greedy algorithms (OMP, MP, and weak-MP) are described for a general matrix  $\mathbf{A}$ , which may have columns that are not of unit  $\ell_2$ -norm. We can normalize the columns by the operation  $\tilde{\mathbf{A}} = \mathbf{A}\mathbf{W}$ , using a diagonal matrix  $\mathbf{W}$  containing  $1/\|\mathbf{a}_i\|_2$  on the main-diagonal, and then use  $\tilde{\mathbf{A}}$  in the above algorithms. Will the outcome be different? The answer is stated in the following Theorem.

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

**Proof:** We start with the OMP's  $k$ -th step, and observe that the choice of the next index is made by finding  $j_0$  that leads to the smallest of the errors, given by  $\epsilon(j) = \|\mathbf{r}^{k-1}\|_2^2 - (\mathbf{a}_j^T \mathbf{r}^{k-1})^2 / \|\mathbf{a}_j\|_2^2$ . Denoting  $\tilde{\mathbf{a}}_j = \mathbf{a}_j / \|\mathbf{a}_j\|_2$ , we clearly have that  $\|\tilde{\mathbf{a}}_j\|_2 = 1$ , and thus we get

$$\begin{aligned} \epsilon(j) &= \|\mathbf{r}^{k-1}\|_2^2 - \left( \frac{\mathbf{a}_j^T \mathbf{r}^{k-1}}{\|\mathbf{a}_j\|_2} \right)^2 \\ &= \|\mathbf{r}^{k-1}\|_2^2 - (\tilde{\mathbf{a}}_j^T \mathbf{r}^{k-1})^2 \\ &= \|\mathbf{r}^{k-1}\|_2^2 - \left( \frac{\tilde{\mathbf{a}}_j^T \mathbf{r}^{k-1}}{\|\tilde{\mathbf{a}}_j\|_2} \right)^2. \end{aligned} \quad (3.11)$$

Thus, using the normalized columns leads to the same outcome for the choice of the index  $j_0$ .

The Least-Squares step that follows finds the solution to  $\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$ , subject to the found support  $S^k$ . Denoting by  $\mathbf{A}_S$  the sub-matrix of  $\mathbf{A}$  that contains the columns from the support, the solution to the above problem is easily given by

$$\mathbf{x}_S^k = (\mathbf{A}_S^T \mathbf{A}_S)^{-1} \mathbf{A}_S^T \mathbf{b}, \quad (3.12)$$

and the resulting residual is

$$\mathbf{r}^k = \mathbf{b} - \mathbf{A}_S \mathbf{x}_S^k = \left[ \mathbf{I} - \mathbf{A}_S (\mathbf{A}_S^T \mathbf{A}_S)^{-1} \mathbf{A}_S^T \right] \mathbf{b}. \quad (3.13)$$

Working instead with the normalized matrix, we use the sub-matrix  $\tilde{\mathbf{A}}_S = \mathbf{A}_S \mathbf{W}_S$ , where  $\mathbf{W}_S$  is the extracted portion from  $\mathbf{W}$  referring to the support  $\mathcal{S}$ . The residual obtained by the OMP that uses this normalized matrix is

$$\begin{aligned} \tilde{\mathbf{r}}^k &= \left[ \mathbf{I} - \tilde{\mathbf{A}}_S (\tilde{\mathbf{A}}_S^T \tilde{\mathbf{A}}_S)^{-1} \tilde{\mathbf{A}}_S^T \right] \mathbf{b} \\ &= \left[ \mathbf{I} - \mathbf{A}_S \mathbf{W}_S (\mathbf{W}_S \mathbf{A}_S^T \mathbf{A}_S \mathbf{W}_S)^{-1} \mathbf{W}_S \mathbf{A}_S^T \right] \mathbf{b} \\ &= \left[ \mathbf{I} - \mathbf{A}_S \mathbf{W}_S \mathbf{W}_S^{-1} (\mathbf{A}_S^T \mathbf{A}_S)^{-1} \mathbf{W}_S^{-1} \mathbf{W}_S \mathbf{A}_S^T \right] \mathbf{b} \\ &= \left[ \mathbf{I} - \mathbf{A}_S (\mathbf{A}_S^T \mathbf{A}_S)^{-1} \mathbf{A}_S^T \right] \mathbf{b} = \mathbf{r}^k. \end{aligned} \quad (3.14)$$

We got that the residual is the same for the two options, original or normalized matrix  $\mathbf{A}$ . Therefore, as the residual drives the next stage of choosing the support, the OMP is indifferent to the normalization, as claimed.

MP uses the same procedure for choosing the index  $j_0$  and as we have already shown, this does not change under normalization. The residual update in MP is even simpler, done by the formula

$$\mathbf{r}^k = \mathbf{r}^{k-1} - z_{j_0}^* \mathbf{a}_{j_0} = \mathbf{r}^{k-1} - \frac{\mathbf{a}_{j_0}^T \mathbf{r}^{k-1}}{\|\mathbf{a}_{j_0}\|_2^2} \mathbf{a}_{j_0}. \quad (3.15)$$

Applying the same step with the normalized matrix gives

$$\tilde{\mathbf{r}}^k = \tilde{\mathbf{r}}^{k-1} - \tilde{z}_{j_0}^* \tilde{\mathbf{a}}_{j_0} = \tilde{\mathbf{r}}^{k-1} - \tilde{\mathbf{a}}_{j_0}^T \tilde{\mathbf{r}}^{k-1} \tilde{\mathbf{a}}_{j_0}. \quad (3.16)$$

Assuming that  $\tilde{\mathbf{r}}^{k-1} = \mathbf{r}^{k-1}$  and using the connection  $\tilde{\mathbf{a}}_j = \mathbf{a}_j / \|\mathbf{a}_j\|_2$  we obtain

$$\begin{aligned} \tilde{\mathbf{r}}^k &= \tilde{\mathbf{r}}^{k-1} - \tilde{\mathbf{a}}_{j_0}^T \tilde{\mathbf{r}}^{k-1} \tilde{\mathbf{a}}_{j_0} \\ &= \mathbf{r}^{k-1} - \left( \frac{\mathbf{a}_{j_0}}{\|\mathbf{a}_{j_0}\|_2} \right)^T \mathbf{r}^{k-1} \left( \frac{\mathbf{a}_{j_0}}{\|\mathbf{a}_{j_0}\|_2} \right) \\ &= \mathbf{r}^{k-1} - \frac{\mathbf{a}_{j_0}^T \mathbf{r}^{k-1}}{\|\mathbf{a}_{j_0}\|_2^2} \mathbf{a}_{j_0} = \mathbf{r}^k, \end{aligned} \quad (3.17)$$

and again we got that the residual remain unchanged as well, guaranteeing the MP steps are also indifferent to the normalization.

The last algorithm to consider, the weak-MP, searches for  $j_0$  that satisfies

$$|\mathbf{a}_{j_0}^T \mathbf{r}^{k-1}| / \|\mathbf{a}_{j_0}\|_2 \geq t \cdot |\mathbf{a}_j^T \mathbf{r}^{k-1}| / \|\mathbf{a}_j\|_2, \quad (3.18)$$

when using the original matrix. The same step with the normalized matrix leads to the very same rule, and thus the same choice of index, due to the relation  $\tilde{\mathbf{a}}_j = \mathbf{a}_j / \|\mathbf{a}_j\|_2$ . Also, the residual update is the very same one used with the MP, and thus the residual does not change too, leading to the fact that weak-MP is also indifferent to the normalization of the matrix  $\mathbf{A}$ , and this concludes this proof.  $\square$

It is simpler to work with the normalized matrix, since then plain inner-products are used in all these algorithms for choosing the next support index. Therefore, we shall assume hereafter that these matrices are preceded with a normalization.

Note that if the original problem is given with a non-normalized matrix, the normalization does have an effect on the obtained result  $\mathbf{x}^k$ . Thus, if a normalized matrix is used in any of these algorithms, a post de-normalization step of the form  $\mathbf{x}^k = \mathbf{W}\tilde{\mathbf{x}}^k$  must be performed. This is because the algorithm provides a solution that satisfies  $\tilde{\mathbf{A}}\tilde{\mathbf{x}}^k = \mathbf{b}$ , and since  $\tilde{\mathbf{A}} = \mathbf{A}\mathbf{W}$  we get that

$$\mathbf{b} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}^k = \mathbf{A}\mathbf{W}\tilde{\mathbf{x}}^k = \mathbf{A}\mathbf{x}^k \Rightarrow \mathbf{x}^k = \mathbf{W}\tilde{\mathbf{x}}^k. \quad (3.19)$$

### 3.1.5 Rate of Decay of the Residual in Greedy Methods

In the MP algorithm we have seen (see [Figure 3.2](#)) that the residual is updated by the recursive formula<sup>1</sup>

$$\mathbf{r}^k = \mathbf{r}^{k-1} - z_{j_0}^* \mathbf{a}_{j_0} = \mathbf{r}^{k-1} - (\mathbf{a}_{j_0}^T \mathbf{r}^{k-1}) \mathbf{a}_{j_0}, \quad (3.20)$$

where  $j_0$  is chosen such that  $|\mathbf{a}_{j_0}^T \mathbf{r}^{k-1}|$  is maximized. Thus, the energy of the residual behaves like

$$\begin{aligned} \epsilon(j_0) &= \|\mathbf{r}^k\|_2^2 \\ &= \|\mathbf{r}^{k-1}\|_2^2 - 2(\mathbf{a}_{j_0}^T \mathbf{r}^{k-1})^2 + (\mathbf{a}_{j_0}^T \mathbf{r}^{k-1})^2 \\ &= \|\mathbf{r}^{k-1}\|_2^2 - (\mathbf{a}_{j_0}^T \mathbf{r}^{k-1})^2 \\ &= \|\mathbf{r}^{k-1}\|_2^2 - \max_{1 \leq j \leq m} (\mathbf{a}_j^T \mathbf{r}^{k-1})^2. \end{aligned} \quad (3.21)$$

Let us define the following decay-factor, following the idea developed by Mallat and Zhang (1993):

**Definition 3.1.** For the matrix  $\mathbf{A}$  with  $m$  normalized columns  $\{\mathbf{a}_j\}_{j=1}^m$ , and an arbitrary vector  $\mathbf{v}$ , the decay-factor  $\delta(\mathbf{A}, \mathbf{v})$  is defined by

$$\delta(\mathbf{A}, \mathbf{v}) = \max_{1 \leq j \leq m} \frac{|\mathbf{a}_j^T \mathbf{v}|^2}{\|\mathbf{v}\|_2^2}. \quad (3.22)$$

<sup>1</sup> From now on we shall assume that the columns of  $\mathbf{A}$  are normalized.



Clearly, by this definition we have that the relation  $\|\mathbf{v}\|_2^2 \cdot \delta(\mathbf{A}, \mathbf{v}) = \max_j |\mathbf{a}_j^T \mathbf{v}|$ . Also evident is the fact that  $0 \leq \delta(\mathbf{A}, \mathbf{v}) \leq 1$  by its definition as a normalized correlation. The maximum is attained for  $\mathbf{v} = \mathbf{a}_j$ . The minimum could be attained only if  $\mathbf{A}$  is rank-deficient. We shall return to this discussion shortly, after using this definition for analyzing the MP error decay rate.

Based on the above definition we now define the following universal decay-factor that minimizes the above over all possible vectors  $\mathbf{v}$ .

**Definition 3.2.** For the matrix  $\mathbf{A}$  with  $m$  normalized columns  $\{\mathbf{a}_j\}_{j=1}^m$ , the universal decay-factor  $\delta(\mathbf{A})$  is defined by

$$\delta(\mathbf{A}) = \inf_{\mathbf{v}} \max_{1 \leq j \leq m} \frac{|\mathbf{a}_j^T \mathbf{v}|^2}{\|\mathbf{v}\|_2^2} = \inf_{\mathbf{v}} \delta(\mathbf{A}, \mathbf{v}). \quad (3.23)$$

This definition seeks the vector  $\mathbf{v}$  that leads to the smallest possible inner product with the columns of  $\mathbf{A}$ . As we show next, this implies the weakest decay of the residual. Returning to Equation (3.21) and using  $\delta(\mathbf{A})$ , the decay-factor of the matrix  $\mathbf{A}$ , we obtain

$$\begin{aligned} \|\mathbf{r}^k\|_2^2 &= \|\mathbf{r}^{k-1}\|_2^2 - \max_{1 \leq j \leq m} |\mathbf{a}_j^T \mathbf{r}^{k-1}|^2 \\ &= \|\mathbf{r}^{k-1}\|_2^2 - \max_{1 \leq j \leq m} \frac{|\mathbf{a}_j^T \mathbf{r}^{k-1}|^2}{\|\mathbf{r}^{k-1}\|_2^2} \cdot \|\mathbf{r}^{k-1}\|_2^2 \\ &= \|\mathbf{r}^{k-1}\|_2^2 - \delta(\mathbf{A}, \mathbf{r}^{k-1}) \cdot \|\mathbf{r}^{k-1}\|_2^2 \\ &\leq \|\mathbf{r}^{k-1}\|_2^2 - \delta(\mathbf{A}) \|\mathbf{r}^{k-1}\|_2^2 = (1 - \delta(\mathbf{A})) \cdot \|\mathbf{r}^{k-1}\|_2^2. \end{aligned} \quad (3.24)$$

Application of this formula recessively leads to

$$\|\mathbf{r}^k\|_2^2 \leq (1 - \delta(\mathbf{A}))^k \cdot \|\mathbf{r}^0\|_2^2 = (1 - \delta(\mathbf{A}))^k \cdot \|\mathbf{b}\|_2^2, \quad (3.25)$$

establishing an exponential rate-of-convergence bound of the residual. A similar analysis for the OMP reveals that this is also the bound on its residual rate of decay, as the LS update in the OMP reduces the error even more, when compared to the MP. Similarly, the Weak-MP leads to an exponential rate of convergence with  $t\delta(\mathbf{A})$  replacing  $\delta(\mathbf{A})$ , leading to a weaker decay-factor.

For all the above to show convergence, we must verify that  $\delta(\mathbf{A})$  is strictly positive and not zero. Since the  $\mathbf{A}$  we consider is a full-rank matrix with more columns than rows, its columns span the entire  $\mathbb{R}^n$  space. Thus, no vector  $\mathbf{v}$  could be orthogonal to all these columns, and thus  $\delta(\mathbf{A}) > 0$ .

As an interesting example, if  $\mathbf{A} = \mathbf{I}$  (or any orthogonal matrix for that matter), it is easy to verify that  $\delta(\mathbf{A}) = 1/n$ . Interestingly, the explanation for this is very similar to the one explaining the smallest mutual-coherence for two-ortho matrices. Take the identity matrix, concatenate the vector  $\mathbf{v}$ , and then compute the Gram matrix. The decay-factor  $\delta(\mathbf{A})$  is simply the largest off-diagonal of this Gram matrix, and this is the square of the largest entry in  $\mathbf{v}$ . Thus, the (normalized) vector that gives the smallest possible inner product with the columns of  $\mathbf{I}$  is  $\mathbf{v}_{opt} = \mathbf{1} / \sqrt{n}$ .

**Task:** Approximate the solution of  $(P_0)$ :  $\min_{\mathbf{x}} \|\mathbf{x}\|_0$  subject to  $\mathbf{Ax} = \mathbf{b}$ .  
**Parameters:**  $\mathbf{A}$ ,  $\mathbf{b}$  are given, along with 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  $\mathcal{S}$  of cardinality  $k$  that contains the smallest errors:  $\forall j \in \mathcal{S}, \epsilon(j) \leq \min_{i \notin \mathcal{S}} \epsilon(i)$ .  
**Update Provisional Solution:** Compute  $\mathbf{x}^k$ , the minimizer of  $\|\mathbf{Ax} - \mathbf{b}\|_2^2$  subject to  $\text{Support}\{\mathbf{x}\} = \mathcal{S}^k$ .  
**Output:** The proposed solution is  $\mathbf{x}$ , the minimizer of  $\|\mathbf{Ax} - \mathbf{b}\|_2^2$  subject to  $\text{Support}\{\mathbf{x}\} = \mathcal{S}$ .

**Fig. 3.4** The Thresholding Algorithm – an algorithm for approximating the solution of  $(P_0)$ .

An alternative expression can be used for the definition of  $\delta(\mathbf{A})$ , by normalizing the columns of  $\mathbf{A}$  as we did before, obtaining  $\tilde{\mathbf{A}}$ . Using this leads to

$$\begin{aligned}
 \delta(\tilde{\mathbf{A}}) &= \inf_{\mathbf{v}} \max_{1 \leq j \leq m} \frac{|\tilde{\mathbf{a}}_j^T \mathbf{v}|^2}{\|\mathbf{v}\|_2^2} \\
 &= \inf_{\mathbf{v}} \frac{1}{\|\mathbf{v}\|_2^2} \max_{1 \leq j \leq m} |\tilde{\mathbf{a}}_j^T \mathbf{v}|^2 \\
 &= \inf_{\mathbf{v}} \frac{\|\tilde{\mathbf{A}}^T \mathbf{v}\|_\infty^2}{\|\mathbf{v}\|_2^2}.
 \end{aligned} \tag{3.26}$$

If the  $\ell_\infty$  would have been replaced by  $\ell_2$ , this would have become the smallest eigenvalue of the matrix  $\tilde{\mathbf{A}} \tilde{\mathbf{A}}^T$ , which is also known to be strictly positive since this matrix is positive-definite.

### 3.1.6 Thresholding Algorithm

We conclude the discussion on the greedy algorithms by introducing another method, which is far simpler than all the previous ones, and slightly different in the way the greediness is practiced. A simplification of the OMP algorithm can be proposed, where the decision made about the support of the solution is based on the first projection alone. The idea is to choose the  $k$  largest inner products as the desired support. This method, called the thresholding algorithm, is depicted in [Figure 3.4](#).

The error  $\epsilon(j)$  according to which the support is chosen is the same one used in earlier algorithms, and is given by

$$\epsilon(j) = \min_{z_j} \|\mathbf{a}_j z_j - \mathbf{b}\|_2^2 \tag{3.27}$$

$$\begin{aligned}
&= \left\| \frac{\mathbf{a}_j^T \mathbf{b}}{\|\mathbf{a}_j\|_2^2} \cdot \mathbf{a}_j - \mathbf{b} \right\|_2^2 \\
&= \|\mathbf{b}\|_2^2 - \frac{(\mathbf{a}_j^T \mathbf{b})^2}{\|\mathbf{a}_j\|_2^2},
\end{aligned}$$

Thus, the same result is obtained by first normalizing the columns in the matrix  $\mathbf{A}$  and then using the simpler formula,

$$\epsilon(j) = \|\mathbf{b}\|_2^2 - (\mathbf{a}_j^T \mathbf{b})^2. \quad (3.28)$$

This also implies that the search for the  $k$  elements of the support amounts to a simple sort of the entries of the vector  $|\mathbf{A}^T \mathbf{b}|$ . Clearly, this algorithm is far simpler than the greedy techniques presented above.

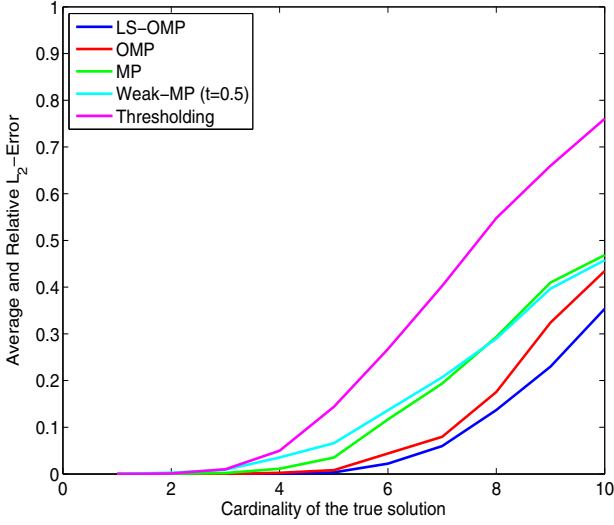
We note that in the above algorithm description, we assume that  $k$ , the number of required non-zeros, is known. Alternatively, we can increase  $k$  until the error  $\|\mathbf{A}\mathbf{x}^k - \mathbf{b}\|_2$  reaches a pre-specified value  $\epsilon_0$ .

### 3.1.7 Numerical Demonstration of Greedy Algorithms

We conclude the discussion on the various greedy algorithms by proposing a simple demonstration of their comparative behavior on a simple case study. We create a random matrix  $\mathbf{A}$  of size  $30 \times 50$  with entries drawn from the normal distribution. We normalize the columns of this matrix to have a unit  $\ell_2$ -norm. We generate sparse vectors  $\mathbf{x}$  with independent and identically-distributed (iid) random supports of cardinalities in the range  $[1, 10]$ , and non-zero entries drawn as random uniform variables in the range  $[-2, -1] \cup [1, 2]$ . We deliberately choose the non-zeroes to be away from zero, as it has a strong effect on the success of greedy methods (and especially the thresholding algorithm). Once  $\mathbf{x}$  is generated, we compute  $\mathbf{b} = \mathbf{A}\mathbf{x}$ , and then apply the above described algorithms to seek for  $\mathbf{x}$ . We perform 1,000 such tests per each cardinality, and present average results. We note that based on ideas developed in Chapter 2 we know that in all our tests the original solution is also the sparsest, as the spark of  $\mathbf{A}$  is 31.

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 –  $\ell_2$ -error, and recovery of the support. The  $\ell_2$ -error is computed as the ratio  $\|\mathbf{x} - \hat{\mathbf{x}}\|^2 / \|\mathbf{x}\|^2$ , indicating that we are interested in  $\ell_2$ -proximity between the two solutions, and we measure this distance as relative to the energy in the true solution.

Such measure does not reveal the complete story of these algorithms, as it fails to indicate when the support is recovered perfectly or at least partially. Thus, we add another measure, computing the distance between the supports of the two solutions. Denoting the two supports as  $\hat{\mathcal{S}}$  and  $\mathcal{S}$ , we define this distance by



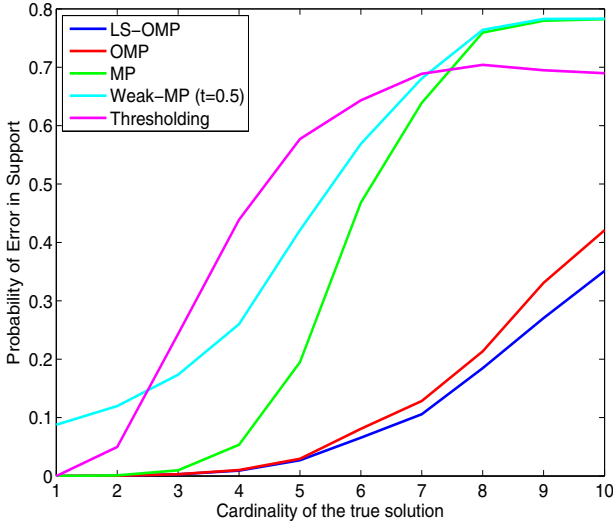
**Fig. 3.5** Greedy algorithms' performance in terms of relative  $\ell_2$  recovery error.

$$\text{dist}(\hat{\mathcal{S}}, \mathcal{S}) = \frac{\max\{|\hat{\mathcal{S}}|, |\mathcal{S}|\} - |\hat{\mathcal{S}} \cap \mathcal{S}|}{\max\{|\hat{\mathcal{S}}|, |\mathcal{S}|\}}. \quad (3.29)$$

If the two supports are the same, the distance is zero. If they are different, the distance is dictated by the size of their intersection, relative to the length of the longer of the two. A distance close to 1 indicate that the two supports are entirely different, with no overlap.

The algorithms compared are the LS-OMP, the OMP, the MP, the Weak-MP with  $t = 0.5$ , and the thresholding algorithm. All these algorithms seek the proper solution till the residual is below a certain threshold ( $\|\mathbf{r}^k\|_2^2 \leq 1e-4$ ). The results of this experiment are summarized in [Figures 3.5](#) and [3.6](#).

As expected, the best performing method is the LS-OMP, closely followed by the OMP. There is a slight difference between the MP and its weak version. The thresholding algorithm performs poorly, being the worst among these techniques. Overall, these algorithms behave quite well for low-cardinalities, in both measures presented, and there is a general agreement between the two measures on how the algorithms are ordered from best to worst. We shall further analyze some of these algorithms theoretically in the next chapter, which will help in understanding the results obtained.



**Fig. 3.6** Greedy algorithms' performance in terms of the success rate in detecting the true support.

## 3.2 Convex Relaxation Techniques

### 3.2.1 Relaxation of the $\ell_0$ -Norm

As already mentioned, a second way to render  $(P_0)$  more tractable is to relax the (highly discontinuous)  $\ell_0$ -norm, replacing it by a continuous or even smooth approximation. Examples of such relaxation options include replacing it with  $\ell_p$  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  $\sum_j (1 - \exp(-\alpha x_j^2))$ .

An interesting example of this family is the FOCal Underdetermined System Solver (FOCUSS) algorithm by Gorodnitsky and Rao. This method uses a method called Iterative-Reweighted-Least-Squares (IRLS) to represent  $\ell_p$  (for some fixed  $p \in (0, 1]$ ) as a weighted  $\ell_2$ -norm. In an iterative algorithm, given a current approximate solution  $\mathbf{x}_{k-1}$ , set  $\mathbf{X}_{k-1} = \text{diag}(|\mathbf{x}_{k-1}|^q)$ . Assuming, for a moment, that this matrix is invertible, the term  $\|\mathbf{X}_{k-1}^{-1} \mathbf{x}\|_2^2$  is equal to  $\|\mathbf{x}\|_{2-2q}^{2-2q}$ , since every entry in  $\mathbf{x}$  is raised to the power  $2-2q$  and then they are summed. Thus, by choosing  $q = 1 - p/2$ , this expression imitates the  $\ell_p$ -norm,  $\|\mathbf{x}\|_p^p$ .

Instead of using the inverse of  $\mathbf{X}_{k-1}$  as done above, we use the pseudo-inverse,  $\|\mathbf{X}_{k-1}^+ \mathbf{x}\|_2^2$ , defined as plain inversion for non-zero entries  $x_i$  and zeros elsewhere. Note that with this change the above interpretation of the  $\ell_p$ -norm is not ruined, as those entries contribute zeros to the summation of the norm anyhow. Based on this, if we attempt to solve the problem

$$(M_k) : \quad \min_{\mathbf{x}} \|\mathbf{X}_{k-1}^+ \mathbf{x}\|_2^2 \quad \text{subject to } \mathbf{b} = \mathbf{A}\mathbf{x}, \quad (3.30)$$

this is an imitation of the  $(P_p)$  problem around the previous solution  $\mathbf{x}_{k-1}$ . In particular, for  $q = 1$ , this would be a variant of the  $(P_0)$  problem. However, as opposed to the direct formulation of  $(P_p)$ , this problem is solvable using standard linear algebra and Lagrange multipliers, as the penalty uses a plain  $\ell_2$ -norm. Thus

$$\begin{aligned} \mathcal{L}(\mathbf{x}) &= \|\mathbf{X}_{k-1}^+ \mathbf{x}\|_2^2 + \lambda^T (\mathbf{b} - \mathbf{A}\mathbf{x}) \\ \Rightarrow \quad \frac{\partial \mathcal{L}(\mathbf{x})}{\partial \mathbf{x}} &= \mathbf{0} = 2(\mathbf{X}_{k-1}^+)^2 \mathbf{x} - \mathbf{A}^T \lambda. \end{aligned} \quad (3.31)$$

If we assume that  $\mathbf{X}_{k-1}^+$  is invertible, this implies that  $\mathbf{X}_{k-1}^+$  has no zero entries on the main-diagonal, and thus  $(\mathbf{X}_{k-1}^+)^{-1} = \mathbf{X}_{k-1}$ . In this case the solution is given by

$$\mathbf{x}_k = 0.5 \mathbf{X}_{k-1}^2 \mathbf{A}^T \lambda. \quad (3.32)$$

In the general case, where some entries in the main-diagonal of  $\mathbf{X}_{k-1}$  are zero, the solution given above nulls the corresponding entries in  $\mathbf{x}_k$ . Assuming hereafter that a zero entry in  $\mathbf{x}_{k-1}$  should remain as such in subsequent iterations (in order to promote sparsity in the solution), this formula remains true and behaves properly. Plugging this to the constraint  $\mathbf{A}\mathbf{x} = \mathbf{b}$  gives

$$0.5 \mathbf{A} (\mathbf{X}_{k-1})^2 \mathbf{A}^T \lambda = \mathbf{b} \Rightarrow \lambda = 2 (\mathbf{A} \mathbf{X}_{k-1}^2 \mathbf{A}^T)^{-1} \mathbf{b}. \quad (3.33)$$

Here again the inversion must be done with care, and in fact should be replaced by a pseudo-inverse.<sup>2</sup> Thus,

$$\mathbf{x}_k = \mathbf{X}_{k-1}^2 \mathbf{A}^T (\mathbf{A} \mathbf{X}_{k-1}^2 \mathbf{A}^T)^+ \mathbf{b}. \quad (3.34)$$

The approximate solution  $\mathbf{x}_k$  is used to update the diagonal matrix  $\mathbf{X}_k$  and a new iteration can begin. This algorithm is formally described in [Figure 3.7](#).

While this algorithm is guaranteed to converge to a fixed-point, it is not necessarily the optimal one. Interestingly, Gorodnitsky and Rao show that the sequence of solutions obtained necessarily give a descent over the function  $\prod_{i=1}^m |x_i|$  for  $p = 0$ . Another intriguing property, already mentioned above, and obvious from Equation (3.34), is the fact that once an entry in  $\mathbf{x}_k$  is nulled, it is never revived. This implies that initialization of this algorithm should be non-zero for all entries, to enable each to be chosen in the converged result.

<sup>2</sup> The matrix  $\mathbf{X}_{k-1}$  chooses a subset of columns from  $\mathbf{A}$ . If the number of columns is  $n$  or more, and those span the complete space, then the inversion is the same as the pseudo-one. In case the number of columns is below  $n$  or if their rank is smaller than  $n$ , there might be a difference between the two inversions. Nevertheless, If in addition those columns span a space that contains  $\mathbf{b}$ , then the pseudo-inverse deployed provides the proper solution for  $\lambda$ . Otherwise,  $\mathbf{b}$  is not contained in those columns' span, and no  $\lambda$  satisfies this equation anyhow. However, this option never takes place if the algorithm prunes the columns of  $\mathbf{A}$  wisely (requires a proof).

**Task:** Find  $\mathbf{x}$  that approximates the solution of  $(P_p)$ :  $\min_{\mathbf{x}} \|\mathbf{x}\|_p^p$  subject to  $\mathbf{b} = \mathbf{A}\mathbf{x}$ .

**Initialization:** Initialize  $k = 0$ , and set

- The initial approximation  $\mathbf{x}_0 = \mathbf{1}$ .
- The initial weight matrix  $\mathbf{X}_0 = \mathbf{I}$ .

**Main Iteration:** Increment  $k$  by 1, and apply these steps:

- **Solve:** the linear system

$$\mathbf{x}_k = (\mathbf{X}_{k-1})^2 \mathbf{A}^T (\mathbf{A} \mathbf{X}_{k-1}^2 \mathbf{A}^T)^+ \mathbf{b}.$$

either directly or iteratively (several Conjugate-Gradient iterations may suffice), producing result  $\mathbf{x}_k$ .

- **Weight Update:** Update the diagonal weight matrix  $\mathbf{X}$  using  $\mathbf{x}_k$ :  $X_k(j, j) = |x_k(j)|^{1-p/2}$ .
- **Stopping Rule:** If  $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2$  is smaller than some predetermined threshold, stop. Otherwise, apply another iteration.

**Output:** The desired result is  $\mathbf{x}_k$ .

**Fig. 3.7** The IRLS strategy for solving  $(P_p)$ .

The FOCUSS algorithm is a practical strategy, but little is known about circumstances where it will be successful, i.e., when a numerical local minimum will actually be a good approximation to a global minimum of  $(P_0)$ . Another popular strategy is to replace the  $\ell_0$  norm by the  $\ell_1$ -norm, which is, in a natural sense, its best convex approximant; many optimization tools are available ‘off the shelf’ for solving  $(P_1)$  (see next section).

Turning from  $(P_0)$  to its relaxed version,  $(P_p)$  with  $0 < p \leq 1$ , care must be taken with respect to normalization of the columns in  $\mathbf{A}$ . While the  $\ell_0$ -norm is indifferent to the magnitude of the non-zero entries in  $\mathbf{x}$ , the  $\ell_p$ -norms tend to penalize higher magnitudes, and thus bias the solution towards choosing to put non-zero entries in  $\mathbf{x}$  in locations that multiply large norm columns in  $\mathbf{A}$ . In order to avoid this bias, the columns should be scaled appropriately. Convexifying with the  $\ell_1$ -norm, the new objective becomes:

$$(P_1) : \quad \min_{\mathbf{x}} \|\mathbf{W}^{-1}\mathbf{x}\|_1 \quad \text{subject to } \mathbf{b} = \mathbf{A}\mathbf{x}. \quad (3.35)$$

The matrix  $\mathbf{W}$  is a diagonal positive-definite matrix that introduces the above-described pre-compensating weights. A natural choice for the  $(i, i)$  entry in this matrix for this case is  $w(i, i) = 1/\|\mathbf{a}_i\|_2$ . Assuming that  $\mathbf{A}$  has no zero columns, all these norms are strictly positive and the problem  $(P_1)$  is well-defined. The case where all the columns of  $\mathbf{A}$  are normalized (and thus  $\mathbf{W} = \mathbf{I}$ ) was named *Basis-Pursuit* (BP) by Chen, Donoho, and Saunders (1995). We will use this name for the more general setup in Equation (3.35) hereafter.

Considering (3.35) and defining  $\tilde{\mathbf{x}} = \mathbf{W}^{-1}\mathbf{x}$ , this problem can be reformulated as

$$(P_1) : \quad \min_{\tilde{\mathbf{x}}} \|\tilde{\mathbf{x}}\|_1 \quad \text{subject to} \quad \mathbf{b} = \mathbf{A}\mathbf{W}\tilde{\mathbf{x}} = \tilde{\mathbf{A}}\tilde{\mathbf{x}}. \quad (3.36)$$

This means that, just as before, we can normalize the columns of  $\mathbf{A}$  and use its normalized version  $\tilde{\mathbf{A}}$ , getting the classic BP format. Just as for the greedy methods, once a solution  $\tilde{\mathbf{x}}$  has been found, it should be de-normalized to provide the required vector  $\mathbf{x}$ . Thus, from now on we can safely assume that  $(P_1)$  is given in a canonic structure with a normalized matrix.

### 3.2.2 Numerical Algorithms for Solving $(P_1)$

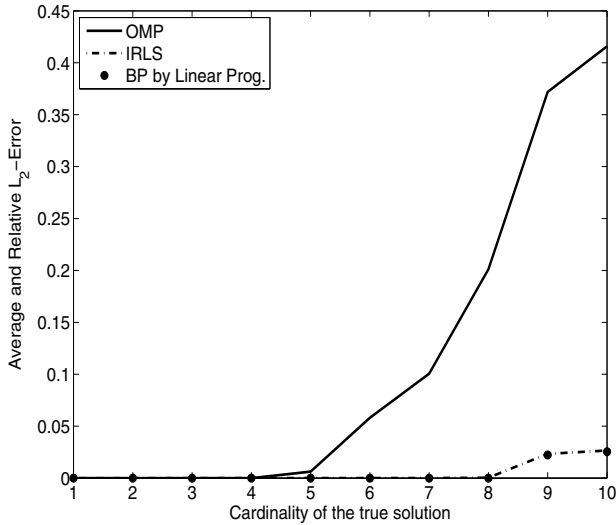
We have seen that the problem  $(P_1)$  can be cast as a linear programming (LP) problem. As such, it can be solved using modern interior-point methods, simplex methods, homotopy methods, or other techniques. Such algorithms are far more sophisticated than the greedy algorithms mentioned earlier, as they obtain the global solution of a well-defined optimization problem. However, this also means that programming these techniques is far more complicated, compared to the greedy algorithms. There are several well-developed software packages that handle this problem, which are freely shared over the web. These include the  $\ell_1$ -magic by Candes and Romberg, the CVX and the L1-LS developed by Boyd and his students, Sparselab managed by David Donoho, SparCo by Michael Friedlander, SPAMS by Julien Mairal, and more.

We also note that the FOCUSS method we have seen above is an interesting and far simpler algorithm for a numerical approximate solution of  $(P_1)$ . However, even though  $(P_1)$  is convex, FOCUSS is not guaranteed to converge to its global minimum. Instead, it may get stuck on a steady-state solution (not a local minimum, as there are none of those!!), or even give fluctuations instead of descending monotonically.

### 3.2.3 Numerical Demonstration of Relaxation Methods

We return to the tests presented in Section 3.1.7 and add two relaxation-based algorithms to the comparison. The first is the IRLS as described above, for  $p = 1$  (i.e., it is expecting to approximate the  $\ell_1$  minimization). The second method tested is the Linear-Programming solver by Matlab, for computing the solution of the Basis Pursuit. We should note that the BP and IRLS take much more time to run compared to the OMP (we shall not provide a complexity analysis of these algorithms – see the relevant literature), and therefore in this experiment we used only 200 experiments per cardinality.



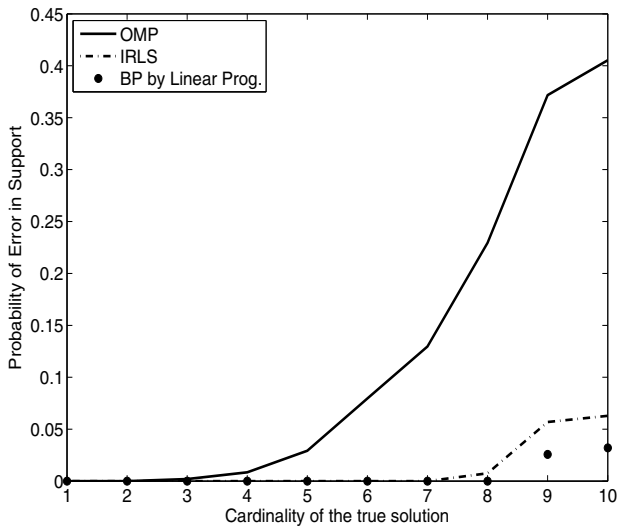


**Fig. 3.8** IRLS and BP (using Matlab’s Linear-Programming) algorithms’ performance in terms of relative  $\ell_2$  recovery error.

Figures 3.8 and 3.9 present the obtained results. For the convenience of comparison to the greedy techniques, we add the OMP performance to these graphs. It is clear that the relaxation methods are performing much better in both measures of quality evaluated. It is also evident that IRLS provides a good approximation to the  $\ell_1$  minimization as done by the Basis-Pursuit.

### 3.3 Summary

In this chapter we have concentrated on numerical solutions to  $(P_0)$ , by either greedy techniques or relaxation ones. Each family of methods has its own merits, and choosing between the various options depends on the application at hand. All the presented algorithms are approximation ones, implying that they may fail from time to time to provide the truly sparsest solution of the linear system  $\mathbf{Ax} = \mathbf{b}$ . The natural question to ask is whether there could be conditions under which such algorithms are guaranteed to operate well. This is exactly the topic of our next chapter.



**Fig. 3.9** IRLS and BP (using Matlab's Linear-Programming) algorithms' performance in terms of the success rate in detecting the truer support.

## Further Reading

1. S.S. Chen, D.L. Donoho, and M.A. Saunders, Atomic decomposition by basis pursuit, *SIAM Journal on Scientific Computing*, 20(1):33–61 (1998).
2. S.S. Chen, D.L. Donoho, and M.A. Saunders, Atomic decomposition by basis pursuit, *SIAM Review*, 43(1):129–159, 2001.
3. A. Cohen, R.A. DeVore, P. Petrushev, and H. Xu, Nonlinear approximation and the space  $BV(\mathbb{R}^2)$ , *American Journal of Mathematics*, 121(3):587–628, June, 1999.
4. G. Davis, S. Mallat, and M. Avellaneda, Adaptive greedy approximations, *Journal of Constructive Approximation*, 13:57–98, 1997.
5. G. Davis, S. Mallat, and Z. Zhang, Adaptive time-frequency decompositions, *Optical-Engineering*, 33(7):2183–91, 1994.
6. R.A. DeVore and V. Temlyakov, Some Remarks on Greedy Algorithms, *Advances in Computational Mathematics*, 5:173–187, 1996.
7. I.F. Gorodnitsky and B.D. Rao, Sparse signal reconstruction from limited data using FOCUSS: A re-weighted norm minimization algorithm, *IEEE Trans. on Signal Processing*, 45(3):600–616, 1997.
8. R. Gribonval and P. Vandergheynst, On the exponential convergence of Matching Pursuits in quasi-incoherent dictionaries, *IEEE Trans. Information Theory*, 52(1):255–261, 2006.
9. L.A. Karlovitz, Construction of nearest points in the  $\ell_p$ ,  $p$  even and  $\ell_\infty$  norms, *Journal of Approximation Theory*, 3:123–127, 1970.

10. S. Mallat, *A Wavelet Tour of Signal Processing*, Academic-Press, 1998.
11. S. Mallat and Z. Zhang, Matching pursuits with time-frequency dictionaries, *IEEE Trans. Signal Processing*, 41(12):3397–3415, 1993.
12. D. Needell and J. A. Tropp, CoSaMP: Iterative signal recovery from incomplete and inaccurate samples, *Applied Computational Harmonic Analysis*, 26:301–321, May 2009.
13. B.D. Rao and K. Kreutz-Delgado, An affine scaling methodology for best basis selection, *IEEE Trans. on signal processing*, 47(1):187–200, 1999.
14. V.N. Temlyakov, Greedy algorithms and m-term approximation, *Journal of Approximation Theory*, 98:117–145, 1999.
15. V.N. Temlyakov, Weak greedy algorithms, *Advances in Computational Mathematics*, 5:173–187, 2000.
16. J.A. Tropp, Greed is good: Algorithmic results for sparse approximation, *IEEE Trans. On Information Theory*, 50(10):2231–2242, October 2004.

## Chapter 4

# Pursuit Algorithms – Guarantees

Assume that the linear system  $\mathbf{Ax} = \mathbf{b}$  has a sparse solution with  $k_0$  non-zeros, i.e.,  $\|\mathbf{x}\|_0 = k_0$ . Furthermore, assume that  $k_0 < \text{spark}(\mathbf{A})/2$ . Will matching pursuit or Basis-Pursuit succeed in recovering the sparsest solution? Clearly, such success cannot be expected for all  $k_0$  and for all matrices  $\mathbf{A}$ , since this would conflict with the known NP-hardness of the problem in the general case. However, if the equation actually has a “sufficiently sparse” solution, the success of these algorithms in addressing the original objective ( $P_0$ ) can be guaranteed.

We present here several such results that correspond to the matching pursuit (OMP, as described in [Figure 3.1](#)), the basis pursuit (i.e., solving ( $P_1$ ) in place of ( $P_0$ )), and the thresholding method. We start our discussion with the special case where  $\mathbf{A}$  is a union of two unitary matrices,  $\mathbf{A} = [\mathbf{\Psi}, \mathbf{\Phi}]$ , and then turn to develop these results for the general matrix  $\mathbf{A}$ .

Before we embark to the exciting discussion ahead of us, we should mention that all the results derived in this chapter are worst-case ones, implying that the kind of guarantees we obtain are over-pessimistic, as they are supposed to hold for all signals, and for all possible supports of a given cardinality. We shall revisit this matter in a later chapter, discussing ways to relax those assumptions.

## 4.1 Back to the Two-Ortho Case

### 4.1.1 OMP Performance Guarantee

In order to simplify our notations, we shall assume that by a rearrangement of the columns in  $\mathbf{\Psi}$  and  $\mathbf{\Phi}$ , the vector  $\mathbf{b}$  is created as a linear combination of the first  $k_p$  columns in  $\mathbf{\Psi}$ , and the first  $k_q$  columns in  $\mathbf{\Phi}$ , such that  $k_0 = k_p + k_q$ . Thus,

$$\mathbf{b} = \mathbf{Ax} = \sum_{i=1}^{k_p} x_i^\psi \psi_i + \sum_{i=1}^{k_q} x_i^\phi \phi_i. \quad (4.1)$$

We shall further denote all the support indices as  $\mathcal{S}_p$  and  $\mathcal{S}_q$ , respectively, where clearly we have that  $|\mathcal{S}_p| = k_p$  and  $|\mathcal{S}_q| = k_q$ .

At the first step ( $k = 0$ ) of the algorithm  $\mathbf{r}^k = \mathbf{r}^0 = \mathbf{b}$ , and the set of computed errors from the **Sweep** step (see Figure 3.1) are given by

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

Thus, for the first step to choose one of the  $k_0$  non-zero entries in the proper support, we should require that for all  $j \notin \mathcal{S}_p$  and  $j \notin \mathcal{S}_q$  we have that

$$(i) \quad |\psi_1^T \mathbf{b}| > |\psi_j^T \mathbf{b}| \quad \text{and} \quad (ii) \quad |\psi_1^T \mathbf{b}| > |\phi_j^T \mathbf{b}|. \quad (4.2)$$

Here we have assumed, without loss of generality, that  $x_1^\psi$  is the largest among the non-zeros in  $\mathbf{x}$ , and we desire to see it chosen in this greedy step. We shall return to this assumption in a short while and handle the opposite case, when the largest entry belongs to  $\mathcal{S}_q$ .

We refer hereafter to the requirement (i), and then turn to treat the second in a similar manner. Substitution of this in Equation (4.1), this requirement translates into

$$\left| \sum_{i=1}^{k_p} x_i^\psi \psi_1^T \psi_i + \sum_{i=1}^{k_q} x_i^\phi \psi_1^T \phi_i \right| > \left| \sum_{i=1}^{k_p} x_i^\psi \psi_j^T \psi_i + \sum_{i=1}^{k_q} x_i^\phi \psi_j^T \phi_i \right|. \quad (4.3)$$

In order to consider the worst-case scenario, we should construct a lower bound for the left-hand-side, an upper-bound for the right-hand-side, and then pose the above inequality again. For the left hand side we have

$$\begin{aligned} \left| \sum_{i=1}^{k_p} x_i^\psi \psi_1^T \psi_i + \sum_{i=1}^{k_q} x_i^\phi \psi_1^T \phi_i \right| &\geq |x_1^\psi| - \sum_{i=1}^{k_q} |x_i^\phi| \cdot \mu(\mathbf{A}) \\ &\geq |x_1^\psi| (1 - k_q \cdot \mu(\mathbf{A})). \end{aligned} \quad (4.4)$$

Here we have exploited the orthogonality of the columns of  $\Psi$  that causes all the elements of the first term to drop, apart from the first one. We used the definition of the mutual-coherence  $\mu(\mathbf{A})$  in Equation (2.3), and the assumption that  $|x_1^\psi|$  is the largest non-zero entry in  $\mathbf{x}$ . We also used the inequality relations  $|a + b| \geq |a| - |b|$  and  $|\sum_i a_i| \leq \sum_i |a_i|$ .

A similar treatment for the right-hand-side of Equation (4.3), this time bounding from above, leads to

$$\left| \sum_{i=1}^{k_p} x_i^\psi \psi_j^T \psi_i + \sum_{i=1}^{k_q} x_i^\phi \psi_j^T \phi_i \right| \leq |x_1^\psi| \cdot k_q \cdot \mu(\mathbf{A}). \quad (4.5)$$

Using these two bounds plugged into the inequality in (4.3), we obtain

$$|x_1^\psi| (1 - k_q \cdot \mu(\mathbf{A})) > |x_1^\psi| \cdot k_q \cdot \mu(\mathbf{A}) \Rightarrow k_q < \frac{1}{2\mu(\mathbf{A})}. \quad (4.6)$$

Since all the above is done with the assumption that  $\mathcal{S}_p$  contains the largest non-zero entry, a parallel requirement  $k_p < \frac{1}{2\mu(\mathbf{A})}$  is also necessary for the opposite case.

Returning now to requirement (ii) above, and following the same treatment, the lower bound on the left-hand-side remains unchanged, whereas the right-hand side becomes

$$\left| \sum_{i=1}^{k_p} x_i^\psi \phi_j^T \psi_i + \sum_{i=1}^{k_q} x_i^\phi \phi_j^T \phi_i \right| \leq |x_1^\psi| \cdot k_p \cdot \mu(\mathbf{A}), \quad (4.7)$$

leading to the requirement

$$|x_1^\psi| (1 - k_q \mu(\mathbf{A})) > |x_1^\psi| \cdot k_p \cdot \mu(\mathbf{A}) \Rightarrow k_p + k_q < \frac{1}{\mu(\mathbf{A})}. \quad (4.8)$$

This requirement is non-informative, as it is covered by the previous two inequalities,  $k_p, k_q < 1/2\mu(\mathbf{A})$ . Satisfying these guarantees that the first step of the OMP performs well, in the sense that a non-zero is chosen from within the proper support.

Once done, the next step is an update of the residual  $\mathbf{r}^1$ . In this update, we subtract a coefficient multiplying the column  $\psi_{t_p(1)}$ , and thus the new residual is also a linear combination of  $k_0$  non-zeros at the most, with the same supports  $\mathcal{S}_p$  and  $\mathcal{S}_q$ . Thus, using the same set of steps as above guarantees that the algorithm finds again an index from the true support of the solution, if the conditions on  $k_p, k_q$  are met. Furthermore, due to the orthogonality between the chosen directions and the residual (a property enforced by the LS,<sup>1</sup>) we never choose the same index twice.

Repeating this reasoning, the same holds true for  $k_0$  such iterations, hence the algorithm selects always values from the correct set of indices, and always an index that has not been chosen yet. After  $k_0$  such iterations exactly, the residual becomes zero and the algorithm stops, ensuring the success of the overall algorithm in recovering the correct solution  $\mathbf{x}$ . All this can now be posed as a theorem for the success of the OMP.

**Theorem 4.1. (Equivalence – OMP – Two-Ortho Case):** *For a system of linear equations  $\mathbf{Ax} = [\Psi, \Phi]\mathbf{x} = \mathbf{b}$  with two ortho-matrices  $\Psi$  and  $\Phi$  of size  $n \times n$ , if a solution  $\mathbf{x}$  exists such that it has  $k_p$  non-zeros in its first half and  $k_q$  non-zeros in the second, and the two obey*

<sup>1</sup> In the OMP algorithm, when we update the residual, we solve  $\min_{\mathbf{x}_{s_k}} \|\mathbf{A}_{s_k} \mathbf{x}_{s_k} - \mathbf{b}\|_2^2$ , and the optimal result satisfies  $\mathbf{A}_{s_k}^T (\mathbf{A}_{s_k} \mathbf{x}_{s_k}^* - \mathbf{b}) = \mathbf{A}_{s_k}^T \mathbf{r}^k = \mathbf{0}$ . This means that the current support columns are all orthogonal to the new residual, and thus in the next sweep phase they give zero inner-product, and are not chosen. On the other hand, in the MP algorithm the residual update is given by  $\mathbf{r}^k = \mathbf{r}^{k-1} - \mathbf{a}_j z_{j_0}^*$ , where  $z_{j_0}^*$  is chosen as the minimizer of  $\epsilon(j) = \|\mathbf{a}_j z - \mathbf{r}^{k-1}\|_2^2$ . Thus, as the derivative of this error for the optimal  $z$  is  $\mathbf{a}_{j_0}^T (\mathbf{a}_{j_0} z_{j_0}^* - \mathbf{r}^{k-1}) = -\mathbf{a}_{j_0}^T \mathbf{r}^k = 0$ , we have that the residual is orthogonal *only* to the last chosen column, and thus the other might be chosen.

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

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

**Proof:** Given above. □

This result is the first of its kind we show here, and its implications are far-reaching. The ability to claim that OMP safely leads to the necessary result is wonderful and encouraging. Interestingly, this result never appeared in a published paper, and instead, a generalization of it to the general matrix  $\mathbf{A}$  was shown. Nevertheless, this result is important by its own, because it will help us understand better the gap in behaviors between OMP and BP. This leads us now naturally to the discussion of the two-ortho case handled by the Basis-Pursuit.

### 4.1.2 BP Performance Guarantee

We now turn to analyze the performance of the Basis-Pursuit in the two-ortho case. This follows the work of Donoho and Huo (2001) and the later improvements by Elad and Bruckstein (2002).

**Theorem 4.2. (Equivalence – Basis Pursuit – Two-Ortho Case):** *For a system of linear equations  $\mathbf{Ax} = [\Psi, \Phi]\mathbf{x} = \mathbf{b}$  with two ortho-matrices  $\Psi$  and  $\Phi$  of size  $n \times n$ , if a solution  $\mathbf{x}$  exists such that it has  $k_p$  non-zeros in its first half and  $k_q \geq k_p$  non-zeros in the second, and the two obey*

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

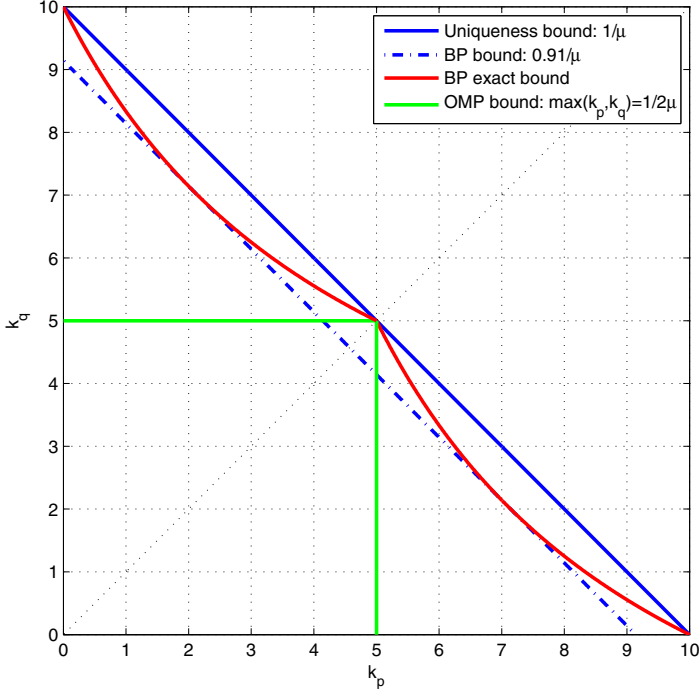
*that solution is both the unique solution of  $(P_1)$  and the unique solution of  $(P_0)$ .*

Since the above condition may seem obscure, we provide a weaker but more “civilized” version of the above requirement, which is simpler to interpret:

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

Figure 4.1 presents a comparison between these two bounds and the result obtained for the OMP. It is clear that OMP is found to be weaker compared to the BP in handling the  $(P_0)$  problem, and denser solutions are anticipated to lead to a successful result with the BP. The question (left open at this stage) is whether the two bounds are tight? The tightness of the BP result was established by Feuer and Nemirovsky, but the tightness of the OMP remains questionable.

**Proof:** Define the following set of alternative solutions:



**Fig. 4.1** The uniqueness and the equivalence bounds for the OMP and BP for the two-ortho case, assuming  $\mu = 0.1$ .

$$C = \left\{ \mathbf{y} \mid \begin{array}{l} \mathbf{y} \neq \mathbf{x}, \quad \|\mathbf{y}\|_1 \leq \|\mathbf{x}\|_1 \\ \|\mathbf{y}\|_0 > \|\mathbf{x}\|_0, \text{ and } \mathbf{A}(\mathbf{y} - \mathbf{x}) = \mathbf{0} \end{array} \right\}. \quad (4.12)$$

This set contains all the possible solutions that are different from  $\mathbf{x}$ , have larger support, satisfy the linear system of equations  $\mathbf{A}\mathbf{y} = \mathbf{b}$ , and are equivalent or better in an  $\ell_1$ -length perspective. This set being non-empty implies that there is an alternative solution that the Basis-Pursuit might find, rather than the desired  $\mathbf{x}$ .

The condition in Equation (4.10) guarantees, due to Theorem 2.3, that  $k_p + k_q = \|\mathbf{x}\|_0 < 1/\mu(\mathbf{A})$ ,  $\mathbf{x}$  is necessarily the unique sparsest possible solution (see Figure 4.1). Thus, alternative solutions ( $\mathbf{y} \neq \mathbf{x}$ ) are necessarily “denser,” and the requirement  $\|\mathbf{y}\|_0 > \|\mathbf{x}\|_0$  can be omitted from the definition of  $C$ .

Defining  $\mathbf{e} = \mathbf{y} - \mathbf{x}$ , we can rewrite  $C$  as a shifted version of it around  $\mathbf{x}$ ,

$$C_s = \{ \mathbf{e} \mid \mathbf{e} \neq \mathbf{0}, \quad \|\mathbf{e} + \mathbf{x}\|_1 - \|\mathbf{x}\|_1 \leq 0, \quad \text{and } \mathbf{A}\mathbf{e} = \mathbf{0} \}. \quad (4.13)$$

The strategy of the proof we are about to present is to enlarge this set and show that even this enlarged set is empty. This will prove that Basis-Pursuit indeed succeeds



in recovering  $\mathbf{x}$ . The enlargement will be done so as to simplify the description of the set, up to the point where its volume can be assessed.

We start with the requirement  $\|\mathbf{e} + \mathbf{x}\|_1 - \|\mathbf{x}\|_1 \leq 0$ . We shall denote by  $\mathbf{x}^p$  and  $\mathbf{x}^q$ , and similarly  $\mathbf{e}^p$  and  $\mathbf{e}^q$ , as the two parts of the vectors  $\mathbf{x}$  and  $\mathbf{e}$ , respectively, each portion corresponding to one of the unitary matrices  $\Psi$  and  $\Phi$ . Using the notations from the OMP analysis,  $\mathcal{S}_p$  and  $\mathcal{S}_q$  of size  $k_p$  and  $k_q$  representing the non-zero supports over the two parts of  $\mathbf{x}$ , this requirement can be re-written as

$$\begin{aligned} 0 \geq \|\mathbf{e} + \mathbf{x}\|_1 - \|\mathbf{x}\|_1 &= \sum_{i=1}^n |e_i^p + x_i^p| - |x_i^p| + \sum_{i=1}^n |e_i^q + x_i^q| - |x_i^q| \\ &= \sum_{i \notin \mathcal{S}_p} |e_i^p| + \sum_{i \notin \mathcal{S}_q} |e_i^q| \\ &\quad + \sum_{i \in \mathcal{S}_p} (|e_i^p + x_i^p| - |x_i^p|) + \sum_{i \in \mathcal{S}_q} (|e_i^q + x_i^q| - |x_i^q|). \end{aligned} \quad (4.14)$$

Here we have exploited the facts that  $x_i^p$  and  $x_i^q$  are zeros outside their support. Using the inequality  $|a+b|-|b| \geq -|a|$ , we can relax the above condition and demand instead

$$\begin{aligned} 0 &\geq \sum_{i \notin \mathcal{S}_p} |e_i^p| + \sum_{i \notin \mathcal{S}_q} |e_i^q| \\ &\quad + \sum_{i \in \mathcal{S}_p} (|e_i^p + x_i^p| - |x_i^p|) + \sum_{i \in \mathcal{S}_q} (|e_i^q + x_i^q| - |x_i^q|) \\ &\geq \sum_{i \notin \mathcal{S}_p} |e_i^p| + \sum_{i \notin \mathcal{S}_q} |e_i^q| - \sum_{i \in \mathcal{S}_p} |e_i^p| - \sum_{i \in \mathcal{S}_q} |e_i^q|. \end{aligned} \quad (4.15)$$

The new inequality poses a weaker requirement over the vector  $\mathbf{e}$ , and thus, with this replacement of the condition we have enlarged the set  $C_s$ . The new inequality can be written more compactly by adding and subtracting the terms  $\sum_{i \in \mathcal{S}_p} |e_i^p|$  and  $\sum_{i \in \mathcal{S}_q} |e_i^q|$  and denoting them as  $\mathbf{1}_p^T \cdot |\mathbf{e}^p|$  and  $\mathbf{1}_q^T \cdot |\mathbf{e}^q|$  to indicate that it is a sum of the non-zeros of the vector  $|\mathbf{e}^p|$  and  $|\mathbf{e}^q|$ , respectively. This leads to

$$\|\mathbf{e}^p\|_1 + \|\mathbf{e}^q\|_1 - 2 \cdot \mathbf{1}_p^T \cdot |\mathbf{e}^p| - 2 \cdot \mathbf{1}_q^T \cdot |\mathbf{e}^q| \leq 0, \quad (4.16)$$

Thus, substituting into the definition of  $C_s$  we get

$$C_s \subseteq \left\{ \mathbf{e} \left| \begin{array}{l} \mathbf{e} \neq \mathbf{0} \\ \|\mathbf{e}^p\|_1 + \|\mathbf{e}^q\|_1 - 2 \cdot \mathbf{1}_p^T \cdot |\mathbf{e}^p| - 2 \cdot \mathbf{1}_q^T \cdot |\mathbf{e}^q| \leq 0 \\ \text{and } \mathbf{A}\mathbf{e} = \mathbf{0} \end{array} \right. \right\}. \quad (4.17)$$

We shall denote this set as  $C_s^1$ .

We now turn to handle the requirement  $\mathbf{A}\mathbf{e} = \Psi\mathbf{e}^p + \Phi\mathbf{e}^q = \mathbf{0}$ , replacing it with a relaxed requirement that expands the set  $C_s^1$  further. First, a multiplication by  $\mathbf{A}^T$  yields the conditions

$$\mathbf{e}^p + \Psi^T \Phi \mathbf{e}^q = \mathbf{0} \text{ and } \Phi^T \Psi \mathbf{e}^p + \mathbf{e}^q = \mathbf{0}. \quad (4.18)$$

Notice that every entry in the matrix  $\Psi^T \Phi$  (and its adjoint) is an inner product bounded from above by the mutual-coherence  $\mu(\mathbf{A})$ . Thus, applying absolute value on the above relationships leads to

$$\begin{aligned} |\mathbf{e}^p| &= |\Psi^T \Phi \mathbf{e}^q| \leq \mu(\mathbf{A}) \cdot \mathbf{1} \cdot |\mathbf{e}^q| \\ \text{and} \\ |\mathbf{e}^q| &= |\Phi^T \Psi \mathbf{e}^p| \leq \mu(\mathbf{A}) \cdot \mathbf{1} \cdot |\mathbf{e}^p|. \end{aligned} \quad (4.19)$$

The term  $\mathbf{1}$  stands for a rank-1  $n \times n$  matrix filled with ones. Returning to the set  $C_s^1$ , we can write

$$C_s^1 \subseteq \left\{ \mathbf{e} \left| \begin{array}{l} \mathbf{e} \neq \mathbf{0} \\ \|\mathbf{e}^p\|_1 + \|\mathbf{e}^q\|_1 - 2 \cdot \mathbf{1}_p^T \cdot |\mathbf{e}^p| - 2 \cdot \mathbf{1}_q^T \cdot |\mathbf{e}^q| \leq 0 \\ |\mathbf{e}^p| \leq \mu(\mathbf{A}) \cdot \mathbf{1} \cdot |\mathbf{e}^q| \\ \text{and } |\mathbf{e}^q| \leq \mu(\mathbf{A}) \cdot \mathbf{1} \cdot |\mathbf{e}^p| \end{array} \right. \right\} = C_s^2. \quad (4.20)$$

Defining  $\mathbf{f}^p = |\mathbf{e}^p|$  and  $\mathbf{f}^q = |\mathbf{e}^q|$ , the above can be written differently as

$$C_f = \left\{ \mathbf{f} \left| \begin{array}{l} \mathbf{f} \neq \mathbf{0} \\ \mathbf{1}^T \mathbf{f}^p + \mathbf{1}^T \mathbf{f}^q - 2 \cdot \mathbf{1}_p^T \cdot \mathbf{f}^p - 2 \cdot \mathbf{1}_q^T \cdot \mathbf{f}^q \leq 0 \\ \mathbf{f}^p \leq \mu(\mathbf{A}) \cdot \mathbf{1} \cdot \mathbf{f}^q \\ \mathbf{f}^q \leq \mu(\mathbf{A}) \cdot \mathbf{1} \cdot \mathbf{f}^p \\ \text{and } \mathbf{f}^p \geq \mathbf{0}, \mathbf{f}^q \geq \mathbf{0} \end{array} \right. \right\}. \quad (4.21)$$

The obtained set  $C_f$  is unbounded since if  $\mathbf{f} \in C_f$  then  $\alpha \mathbf{f} \in C_f$  for all  $\alpha \geq 0$ . Thus, in order to study its behavior, we can restrict our quest for normalized vectors,  $\mathbf{1}^T \mathbf{f} = \mathbf{1}^T \mathbf{f}^p + \mathbf{1}^T \mathbf{f}^q = 1$ . This new set, denoted as  $C_r$  becomes

$$C_r = \left\{ \mathbf{f} \left| \begin{array}{l} \mathbf{f} \neq \mathbf{0} \\ 1 - 2 \cdot \mathbf{1}_p^T \cdot \mathbf{f}^p - 2 \cdot \mathbf{1}_q^T \cdot \mathbf{f}^q \leq 0 \\ \mathbf{f}^p \leq \mu(\mathbf{A}) \cdot \mathbf{1} \cdot \mathbf{f}^q \\ \mathbf{f}^q \leq \mu(\mathbf{A}) \cdot \mathbf{1} \cdot \mathbf{f}^p \\ \mathbf{1}^T (\mathbf{f}^p + \mathbf{f}^q) = 1 \\ \text{and } \mathbf{f}^p \geq \mathbf{0}, \mathbf{f}^q \geq \mathbf{0} \end{array} \right. \right\}. \quad (4.22)$$

The set  $C_r$  is very comfortable to analyze because of several reasons:

- Compared to the definition of the set  $C$ , in  $C_r$  there is no explicit use for  $\mathbf{A}$ , and instead, the mutual-coherence represents it. Similarly, there is no trace to the optimal solution  $\mathbf{x}$ .
- The set  $C_r$  is posed in terms of linear constraints and positivity, which make it possible to convert to a Linear-Programming problem.
- The order of the non-zeros in  $\mathbf{f}^p$  and  $\mathbf{f}^q$  has no influence on the solution, and thus, without loss of generality, we can assume that the  $k_p, k_q$  non-zeroes are located at the start of these vectors.

```

n = 50; kp = 7; kq = 9; mu = 0.1; % Setting parameters
C = [ones(1,kp), ...           % Penalty
     zeros(1,n-kp), ...
     ones(1,kq), ...
     zeros(1,n-kq)];
A = [ones(1,2*n);             % Inequality Constraints Matrix
     -ones(1,2*n);
     eye(n), -ones(n)*mu;
     -ones(n)*mu, eye(n)];
b=[1; -1; zeros(2*n,1)];      % Inequality Constraints vector
x=linprog(-C,A,b,[],[],zeros(2*n,1)); % Linear Programming Solver
plot(x,');                    % Showing the Result

```

**Fig. 4.2** Matlab code for solving the LP problem posed in Equation (4.23).

We can now define the following Linear-Programming problem based on  $C_r$ ,

$$\begin{aligned}
 \max_{\mathbf{f}^p, \mathbf{f}^q} \quad & \mathbf{1}_p^T \cdot \mathbf{f}^p + \mathbf{1}_q^T \cdot \mathbf{f}^q \quad \text{subject to} \quad \begin{cases} \mathbf{f}^p \leq \mu(\mathbf{A}) \cdot \mathbf{1} \cdot \mathbf{f}^q \\ \mathbf{f}^q \leq \mu(\mathbf{A}) \cdot \mathbf{1} \cdot \mathbf{f}^p \\ \mathbf{1}^T(\mathbf{f}^p + \mathbf{f}^q) = 1 \\ \mathbf{f}^p \geq \mathbf{0}, \mathbf{f}^q \geq \mathbf{0} \end{cases}
 \end{aligned} \quad (4.23)$$

The idea is the following. Suppose we are able to find the solution to this problem,  $\mathbf{f}_{opt}^p, \mathbf{f}_{opt}^q$ . If the penalty term satisfies  $\mathbf{1}_p^T \cdot \mathbf{f}_{opt}^p + \mathbf{1}_q^T \cdot \mathbf{f}_{opt}^q < 0.5$ , this implies that the first requirement in the definition of the set  $C_r$  is not met, implying that this set is empty, and thus BP succeeds. As we shall show, for  $k_p$  and  $k_q$  small enough, this is exactly the situation.

Solving the above LP problem is quite easy, and especially so when a numerical solution is considered. Figure 4.2 presents a short Matlab program that does exactly that – try it, and see how the solution behaves.

Based on such numerical tests, if we assume  $k_q \geq k_p$ , the following parametric solution be be proposed:

$$\mathbf{f}_{opt}^p = [\alpha \cdot \mathbf{1}_{k_p}^T, \beta \cdot \mathbf{1}_{n-k_p}^T]^T \quad \text{and} \quad \mathbf{f}_{opt}^q = [\gamma \cdot \mathbf{1}_{k_q}^T, \delta \cdot \mathbf{1}_{n-k_q}^T]^T, \quad (4.24)$$

i.e., the first  $k_p$  entries in  $\mathbf{f}_{opt}^p$  are the same, equal to  $\alpha$ , and the rest are also the same, equal to  $\beta$ . A similar structure is imposed on  $\mathbf{f}_{opt}^q$  with values  $\gamma$  and  $\delta$ . This solution can be tested for feasibility and lead to the desired result, but there is a problem with such an approach: Suppose we show that the proposed result does not lead to a penalty below  $1/2$ . How can we be sure that this is truly the optimal solution, and we have truly exhausted all other possibilities?

The approach taken in the original work by Elad and Bruckstein was to introduce the dual to the above problem. The dual is a minimization problem, known to bound the primal (original) penalty from above, and have the same optimal value. Thus, by

suggesting (wisely) a solution to the dual and checking that it does not exceed  $1/2$ , we prove conclusively that the primal is smaller than  $1/2$  as well.

Here we shall take a different, and simpler, route. We address the solution of (4.23), and denote  $\|\mathbf{f}_{opt}^p\|_1 = \alpha$ , and thus  $\|\mathbf{f}_{opt}^q\|_1 = 1 - \alpha$ , due to the last constraint,  $\mathbf{1}^T(\mathbf{f}^p + \mathbf{f}^q) = 1$ . The first and second constraints translate into

$$\begin{aligned}\mathbf{f}^p &\leq \mu(\mathbf{A}) \cdot \mathbf{1} \cdot \mathbf{f}^q = (1 - \alpha)\mu(\mathbf{A}) \cdot \mathbf{1} \\ \mathbf{f}^q &\leq \mu(\mathbf{A}) \cdot \mathbf{1} \cdot \mathbf{f}^p = \alpha\mu(\mathbf{A}) \cdot \mathbf{1}.\end{aligned}\quad (4.25)$$

Since the objective is the maximization of the term  $\mathbf{1}_p^T \cdot \mathbf{f}^p + \mathbf{1}_q^T \cdot \mathbf{f}^q$ , it is natural to choose all the non-zero entries as large as possible, based on the constraints posed in Equation (4.25). Thus,

$$\begin{aligned}\mathbf{1}_p^T \cdot \mathbf{f}^p + \mathbf{1}_q^T \cdot \mathbf{f}^q &= k_p(1 - \alpha)\mu(\mathbf{A}) + k_q\alpha\mu(\mathbf{A}) \\ &= k_p\mu(\mathbf{A}) - \alpha(k_p - k_q)\mu(\mathbf{A}).\end{aligned}\quad (4.26)$$

Since we assume that  $k_p \geq k_q$ , it is clear that we should choose  $\alpha$  as small as possible so as to maximize this expression. We now add the following two conditions, stating that the energy allocated to the non-zeros in the two vectors should not exceed the entire ( $\ell_1$ ) energy,

$$\begin{aligned}\|\mathbf{f}_{opt}^p\|_1 = \mathbf{1}^T \mathbf{f}^p &= \alpha \geq k_p(1 - \alpha)\mu(\mathbf{A}) \\ \|\mathbf{f}_{opt}^q\|_1 = \mathbf{1}^T \mathbf{f}^q &= (1 - \alpha) \geq k_q\alpha\mu(\mathbf{A}).\end{aligned}\quad (4.27)$$

These lead to the following inequalities, that pose limits on the value of  $\alpha$ :

$$\begin{aligned}\alpha \geq k_p(1 - \alpha)\mu(\mathbf{A}) &\Rightarrow \alpha \geq \frac{k_p\mu(\mathbf{A})}{1 + k_p\mu(\mathbf{A})} \\ (1 - \alpha) \geq k_q \cdot \alpha\mu(\mathbf{A}) &\Rightarrow \alpha \leq \frac{1}{1 + k_q\mu(\mathbf{A})}.\end{aligned}\quad (4.28)$$

For these two conditions to have an intersection we should require

$$\frac{k_p\mu(\mathbf{A})}{1 + k_p\mu(\mathbf{A})} \leq \frac{1}{1 + k_q\mu(\mathbf{A})} \Rightarrow k_p k_q \leq \frac{1}{\mu(\mathbf{A})^2}.\quad (4.29)$$

Assuming that this condition is met (and we should verify this at the end, but looking at [Figure 4.1](#), this is obviously met), the smallest possible value for  $\alpha$  is  $\frac{k_p\mu(\mathbf{A})}{1 + k_p\mu(\mathbf{A})}$ , and thus the maximal penalty in Equation (4.26) becomes

$$\begin{aligned}\mathbf{1}_p^T \cdot \mathbf{f}^p + \mathbf{1}_q^T \cdot \mathbf{f}^q &= k_p\mu(\mathbf{A}) - \alpha\mu(\mathbf{A})(k_p - k_q) \\ &= k_p\mu(\mathbf{A}) - \frac{k_p\mu(\mathbf{A})}{1 + k_p\mu(\mathbf{A})}\mu(\mathbf{A})(k_p - k_q) \\ &= \frac{k_p\mu(\mathbf{A}) + k_p k_q \mu(\mathbf{A})^2}{1 + k_p\mu(\mathbf{A})}.\end{aligned}\quad (4.30)$$

For BP to succeed, we should require that this expression is strictly smaller than  $1/2$  leading to

$$\begin{aligned} \frac{k_p \mu(\mathbf{A}) + k_p k_q \mu(\mathbf{A})^2}{1 + k_p \mu(\mathbf{A})} &< \frac{1}{2} \\ \Rightarrow 2k_p k_q \mu(\mathbf{A})^2 + k_p \mu(\mathbf{A}) - 1 &< 0, \end{aligned} \quad (4.31)$$

which is the claimed requirement.

In order to prove the alternative inequality  $\|\mathbf{x}\|_0 = k_p + k_q < \frac{\sqrt{2}-0.5}{\mu(\mathbf{A})}$ , we will take the above requirement, isolate the term  $k_q$  and then add  $k_p$  to it, leading to

$$\begin{aligned} k_p + k_q = \|\mathbf{x}\|_0 &< k_p + \frac{1 - k_p \mu(\mathbf{A})}{2\mu(\mathbf{A})^2 k_p} \\ &= \frac{2\mu(\mathbf{A})^2 k_p^2 + 1 - \mu(\mathbf{A}) k_p}{2\mu(\mathbf{A})^2 k_p} \\ &= \frac{1}{\mu(\mathbf{A})} \cdot \frac{2(\mu(\mathbf{A}) k_p)^2 + 1 - (\mu(\mathbf{A}) k_p)}{2(\mu(\mathbf{A}) k_p)}, \end{aligned} \quad (4.32)$$

By minimizing the upper-bound with respect to the term  $\mu(\mathbf{A}) k_p$  we obtain

$$\begin{aligned} f(u) &= \frac{2u^2 - u + 1}{2u} \\ \Rightarrow f'(u) &= \frac{2u(4u - 1) - 2(2u^2 - u + 1)}{4u^2} = \frac{2u^2 - 1}{2u^2} = 0. \end{aligned}$$

The optimal value (should be checked for being a minimum and not maximum) is thus  $\mu(\mathbf{A}) k_p = \pm \sqrt{0.5}$ . The negative solution is irrelevant (since both  $\mu(\mathbf{A})$  and  $k_p$  and non-negative), and we obtain

$$\begin{aligned} k_p + k_q = \|\mathbf{x}\|_0 &< \frac{\sqrt{2} - 0.5}{\mu(\mathbf{A})} \\ &\leq \frac{1}{\mu(\mathbf{A})} \cdot \frac{2(\mu(\mathbf{A}) k_p)^2 + 1 - (\mu(\mathbf{A}) k_p)}{2(\mu(\mathbf{A}) k_p)}, \end{aligned} \quad (4.33)$$

as claimed.  $\square$

## 4.2 The General Case

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

of this matrix are normalized, since if this is not the case, we can normalize it and de-normalize the outcome at the end, getting the very same solution.

### 4.2.1 OMP Performance Guarantee

The following result generalizes the one in Theorem 4.1 for general matrices, and therefore it is necessarily weaker. The origin of this weakness is the fact that in the two-ortho case, the Gram matrix contains some inner-products that are known to be zero, while here we do not assume such a structure. The proof provided for the general theorem is very much similar to the one shown above, with several necessary modifications.

**Theorem 4.3. (Equivalence – Orthogonal Greedy Algorithm):** *For a system of linear equations  $\mathbf{Ax} = \mathbf{b}$  ( $\mathbf{A} \in \mathbb{R}^{n \times m}$  full-rank with  $n < m$ ), if a solution  $\mathbf{x}$  exists obeying*

$$\|\mathbf{x}\|_0 < \frac{1}{2} \left( 1 + \frac{1}{\mu(\mathbf{A})} \right), \quad (4.34)$$

*OMP (OGA) run with threshold parameter  $\epsilon_0 = 0$  is guaranteed to find it exactly.*

**Proof:** Suppose, without loss of generality, that the 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 of the values  $|x_j|$ . Thus,

$$\mathbf{b} = \mathbf{Ax} = \sum_{t=1}^{k_0} x_t \mathbf{a}_t. \quad (4.35)$$

At the first step ( $k = 0$ ) of the algorithm  $\mathbf{r}^k = \mathbf{r}^0 = \mathbf{b}$ , and the set of computed errors from the **Sweep** step are given by

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

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

$$|\mathbf{a}_1^T \mathbf{b}| > |\mathbf{a}_i^T \mathbf{b}|. \quad (4.36)$$

Substitution of this in Equation (4.35), this requirement translates into

$$\left| \sum_{t=1}^{k_0} x_t \mathbf{a}_1^T \mathbf{a}_t \right| > \left| \sum_{t=1}^{k_0} x_t \mathbf{a}_i^T \mathbf{a}_t \right|. \quad (4.37)$$

Again, we construct a lower bound for the left-hand-side, an upper-bound for the right-hand-side, and then pose the above requirement again. For the left-hand-side we have

$$\begin{aligned}
 \left| \sum_{t=1}^{k_0} x_t \mathbf{a}_1^T \mathbf{a}_t \right| &\geq |x_1| - \sum_{t=2}^{k_0} |x_t| \cdot |\mathbf{a}_1^T \mathbf{a}_t| \\
 &\geq |x_1| - \sum_{t=2}^{k_0} |x_t| \cdot \mu(\mathbf{A}) \\
 &\geq |x_1| (1 - \mu(\mathbf{A})(k_0 - 1)).
 \end{aligned} \tag{4.38}$$

Here we have exploited the definition of the mutual-coherence  $\mu(\mathbf{A})$  in Equation (2.21), and the descending ordering of the values  $|x_j|$ . Similarly, the right-hand-side term is bounded by

$$\begin{aligned}
 \left| \sum_{t=1}^{k_0} x_t \mathbf{a}_i^T \mathbf{a}_t \right| &\leq \sum_{t=1}^{k_0} |x_t| \cdot |\mathbf{a}_i^T \mathbf{a}_t| \\
 &\leq \sum_{t=1}^{k_0} |x_t| \cdot \mu(\mathbf{A}) \\
 &\leq |x_1| \cdot \mu(\mathbf{A}) k_0.
 \end{aligned} \tag{4.39}$$

Using these two bounds plugged into the inequality in (4.37), we obtain

$$\begin{aligned}
 \left| \sum_{t=1}^{k_0} x_t \mathbf{a}_1^T \mathbf{a}_t \right| &\geq |x_1| \cdot (1 - \mu(\mathbf{A})(k_0 - 1)) \\
 &> |x_1| \cdot \mu(\mathbf{A}) k_0 \\
 &\geq \left| \sum_{t=1}^{k_0} x_t \mathbf{a}_i^T \mathbf{a}_t \right|,
 \end{aligned} \tag{4.40}$$

which leads to

$$1 + \mu(\mathbf{A}) > 2\mu(\mathbf{A})k_0 \quad \Rightarrow \quad k_0 < \frac{1}{2} \left( 1 + \frac{1}{\mu(\mathbf{A})} \right), \tag{4.41}$$

which is exactly the condition of sparsity stated above. This condition guarantees the success of the first stage of the algorithm, implying that the chosen element must be in the correct support of the sparsest decomposition.

Once done, 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. Using the same set of steps we obtain that condition (4.41) guarantees that the algorithm finds again an index from the true support of the solution, and the orthogonality enforced by the LS guarantees that the same entry is never chosen

again. After  $k_0$  such iterations, the residual becomes zero and the algorithm stops, ensuring the success of the overall algorithm in recovering the correct solution  $\mathbf{x}$  as the theorem claims.  $\square$

### 4.2.2 Thresholding Performance Guarantee

The thresholding algorithm presented in the previous chapter is very appealing due to its evident simplicity. The question we should ask is whether such simple method could lead to a guaranteed performance? We provide the following analysis that follows much of the flavor of the above greedy method, and using the same notations.

Success of the thresholding algorithm is guaranteed by the requirement

$$\min_{1 \leq i \leq k_0} |\mathbf{a}_i^T \mathbf{b}| > \max_{j > k_0} |\mathbf{a}_j^T \mathbf{b}|, \quad (4.42)$$

Plugging the expression  $\mathbf{b} = \sum_{t=1}^{k_0} x_t \mathbf{a}_t$ , the left-hand-side term becomes

$$\min_{1 \leq i \leq k_0} |\mathbf{a}_i^T \mathbf{b}| = \min_{1 \leq i \leq k_0} \left| \sum_{t=1}^{k_0} x_t \mathbf{a}_i^T \mathbf{a}_t \right|. \quad (4.43)$$

We exploit the fact that the columns of  $\mathbf{A}$  are normalized, their maximal inner product is bounded from above by  $\mu(\mathbf{A})$ , and we also use the relation  $|a + b| \geq |a| - |b|$ . All these enable us to lower-bound this term by

$$\begin{aligned} \min_{1 \leq i \leq k_0} \left| \sum_{t=1}^{k_0} x_t \mathbf{a}_i^T \mathbf{a}_t \right| &= \min_{1 \leq i \leq k_0} \left| x_i + \sum_{1 \leq t \leq k_0, t \neq i} x_t \mathbf{a}_i^T \mathbf{a}_t \right| \\ &\geq \min_{1 \leq i \leq k_0} \left\{ |x_i| - \left| \sum_{1 \leq t \leq k_0, t \neq i} x_t \mathbf{a}_i^T \mathbf{a}_t \right| \right\} \\ &\geq \min_{1 \leq i \leq k_0} |x_i| - \max_{1 \leq i \leq k_0} \left| \sum_{1 \leq t \leq k_0, t \neq i} x_t \mathbf{a}_i^T \mathbf{a}_t \right| \\ &\geq |x_{\min}| - (k_0 - 1)\mu(\mathbf{A})|x_{\max}|. \end{aligned} \quad (4.44)$$

$|x_{\min}|$  and  $|x_{\max}|$  and the minimum and maximum values of the vector  $|\mathbf{x}|$  on the support  $1 \leq t \leq k_0$ .

Turning to the right-hand-side term in Equation (4.42), and following similar steps, we aim to upper-bound this expression. This gives

$$\max_{j > k_0} |\mathbf{a}_j^T \mathbf{b}| = \max_{j > k_0} \left| \sum_{t=1}^{k_0} x_t \mathbf{a}_j^T \mathbf{a}_t \right| \leq k_0 \mu(\mathbf{A}) |x_{\max}|. \quad (4.45)$$



Thus, requiring

$$|x_{\min}| - (k_0 - 1)\mu(\mathbf{A})|x_{\max}| > k_0\mu(\mathbf{A})|x_{\max}| \quad (4.46)$$

necessarily leads to the satisfaction of the condition posed in Equation (4.42). Put differently, this reads as the condition

$$k_0 < \frac{1}{2} \left( \frac{|x_{\min}|}{|x_{\max}|} \frac{1}{\mu(\mathbf{A})} + 1 \right) \quad (4.47)$$

which guarantees the success of the thresholding algorithm. As can be seen, this requirement is more strict, compared to the OMP requirement found above, and becomes equivalent to it if all the non-zero entries in  $\mathbf{x}$  are of the same magnitude. We conclude by posing this as a theorem.

**Theorem 4.4. (Equivalence – Thresholding Algorithm):** *For a system of linear equations  $\mathbf{Ax} = \mathbf{b}$  ( $\mathbf{A} \in \mathbb{R}^{n \times m}$  full-rank with  $n < m$ ), if a solution  $\mathbf{x}$  (with minimal non-zero value  $|x_{\min}|$  and maximal one  $|x_{\max}|$ ) exists obeying*

$$\|\mathbf{x}\|_0 < \frac{1}{2} \left( 1 + \frac{1}{\mu(\mathbf{A})} \frac{|x_{\min}|}{|x_{\max}|} \right), \quad (4.48)$$

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

### 4.2.3 BP Performance Guarantee

We now turn to consider Basis-Pursuit, i.e., the replacement of  $(P_0)$  by  $(P_1)$  as an optimization problem. Surprisingly, the same bound on the sparsity as in the OMP analysis also guarantees the success of basis pursuit. Note, however, that this does not imply that the two algorithms are always expected to perform similarly! Indeed, we have already seen a gap between these two methods for the two-ortho case. These two general case Theorems do suggest that the worst-case behavior of the two techniques is the same, provided that the two bounds are tight. Empirical evidence will be presented in later chapter.

**Theorem 4.5. (Equivalence – Basis Pursuit):** *For the system of linear equations  $\mathbf{Ax} = \mathbf{b}$  ( $\mathbf{A} \in \mathbb{R}^{n \times m}$  full-rank with  $n < m$ ), if a solution  $\mathbf{x}$  exists obeying*

$$\|\mathbf{x}\|_0 < \frac{1}{2} \left( 1 + \frac{1}{\mu(\mathbf{A})} \right), \quad (4.49)$$

*that solution is both the unique solution of  $(P_1)$  and the unique solution of  $(P_0)$ .*

**Proof:** The proof is similar to the one provided for the two-ortho case, with some necessary modifications. In fact, this proof is easier in some ways.

Define the following set of alternative solutions:

$$C = \left\{ \mathbf{y} \left| \begin{array}{l} \mathbf{y} \neq \mathbf{x} \\ \|\mathbf{y}\|_1 \leq \|\mathbf{x}\|_1 \\ \|\mathbf{y}\|_0 > \|\mathbf{x}\|_0 \\ \text{and } \mathbf{A}(\mathbf{y} - \mathbf{x}) = \mathbf{0} \end{array} \right. \right\}. \quad (4.50)$$

This set contains all the possible solutions that are different from  $\mathbf{x}$ , have larger support, satisfy the linear system of equations  $\mathbf{A}\mathbf{y} = \mathbf{b}$ , and are at least as good from the weighted  $\ell_1$  perspective. This set being non-empty implies that there is an alternative solution that the Basis-Pursuit will find, rather than the desired  $\mathbf{x}$ .

In view of Theorem 2.5 and the fact that  $\|\mathbf{x}\|_0 < (1 + 1/\mu(\mathbf{A}))/2$ ,  $\mathbf{x}$  is necessarily the unique sparsest possible solution, hence alternative solutions ( $\mathbf{y} \neq \mathbf{x}$ ) are necessarily “denser.” Thus, this requirement can be omitted from the definition of  $C$ . Defining  $\mathbf{e} = \mathbf{y} - \mathbf{x}$ , we can rewrite  $C$  as a shifted version of it around  $\mathbf{x}$ ,

$$C_s = \{\mathbf{e} \mid \mathbf{e} \neq \mathbf{0}, \quad \|\mathbf{e} + \mathbf{x}\|_1 - \|\mathbf{x}\|_1 \leq 0, \quad \text{and } \mathbf{A}\mathbf{e} = \mathbf{0}\}. \quad (4.51)$$

The strategy of the proof we are about to present is to enlarge this set and show that even this enlarged set is empty. This will prove that Basis-Pursuit indeed succeeds in recovering  $\mathbf{x}$ .

We start with the requirement  $\|\mathbf{e} + \mathbf{x}\|_1 - \|\mathbf{x}\|_1 \leq 0$ . Assuming, without loss of generality, that by a simple column permutation of  $\mathbf{A}$ , the  $k_0$  non-zeros in  $\mathbf{x}$  are located at the beginning of the vector, this requirement can be written as

$$\|\mathbf{e} + \mathbf{x}\|_1 - \|\mathbf{x}\|_1 = \sum_{j=1}^{k_0} |e_j + x_j| - |x_j| + \sum_{j>k_0} |e_j| \leq 0 \quad (4.52)$$

Using the inequality  $|a+b| - |b| \geq -|a|$ , we can relax the above condition and demand instead

$$-\sum_{j=1}^{k_0} |e_j| + \sum_{j>k_0} |e_j| \leq \sum_{j=1}^{k_0} |e_j + x_j| - |x_j| + \sum_{j>k_0} |e_j| \leq 0. \quad (4.53)$$

This inequality can be written more compactly by adding and subtracting the term  $\sum_{j=1}^{k_0} |e_j|$  and denoting it as  $\mathbf{1}_{k_0}^T \cdot |\mathbf{e}|$  to indicate that it is a sum of the first  $k_0$  entries of the vector  $|\mathbf{e}|$ . This leads to

$$\|\mathbf{e}\|_1 - 2\mathbf{1}_{k_0}^T \cdot |\mathbf{e}| \leq 0. \quad (4.54)$$

Thus, substituting into the definition of  $C_s$  we get

$$C_s \subseteq \left\{ \mathbf{e} \mid \mathbf{e} \neq \mathbf{0}, \quad \|\mathbf{e}\|_1 - 2\mathbf{1}_{k_0}^T \cdot |\mathbf{e}| \leq 0, \quad \text{and } \mathbf{A}\mathbf{e} = \mathbf{0} \right\} = C_s^1. \quad (4.55)$$

As stated above, the modification from the condition  $\|\mathbf{e} + \mathbf{x}\|_1 - \|\mathbf{x}\|_1 \leq 0$  to the new alternative  $\|\mathbf{e}\|_1 - 2\mathbf{1}_{k_0}^T \cdot |\mathbf{e}| \leq 0$  is effectively enlarging the set, because every vector  $\mathbf{e}$  satisfying the initial condition must also satisfy the new one, but not vice-versa,

We now turn to handle the requirement  $\mathbf{A}\mathbf{e} = \mathbf{0}$ , replacing it with a relaxed requirement that expands the set  $C_s^1$  further. First, a multiplication by  $\mathbf{A}^T$  yields the condition  $\mathbf{A}^T \mathbf{A} \mathbf{e} = \mathbf{0}$ , which does not yet change the set  $C_s^1$ . Every entry in the matrix  $\mathbf{A}^T \mathbf{A}$  is a normalized inner product used for the definition of the mutual-coherence  $\mu(\mathbf{A})$ . Also, the main-diagonal of this matrix contains ones. Thus, this condition can be rewritten by adding and removing  $\mathbf{e}$ , as follows

$$-\mathbf{e} = (\mathbf{A}^T \mathbf{A} - \mathbf{I})\mathbf{e}. \quad (4.56)$$

Taking an entry-wise absolute value on both sides we relax the requirement on  $\mathbf{e}$  and obtain

$$\begin{aligned} |\mathbf{e}| &= |(\mathbf{A}^T \mathbf{A} - \mathbf{I})\mathbf{e}| \leq |\mathbf{A}^T \mathbf{A} - \mathbf{I}| \cdot |\mathbf{e}| \\ &\leq \mu(\mathbf{A})(\mathbf{1} - \mathbf{I}) \cdot |\mathbf{e}|. \end{aligned} \quad (4.57)$$

Here we have used the relation  $|\sum_i g_i v_i| \leq \sum_i |g_i| |v_i|$ . This relation could be interpreted as the effect of one row of a matrix  $\mathbf{G}$  multiplying the vector  $\mathbf{v}$ , implying that the  $j$ -th element in such a multiplication satisfies  $|\mathbf{G}\mathbf{v}|_j \leq (|\mathbf{G}||\mathbf{v}|)_j$ .

The term  $\mathbf{1}$  stands for a rank-1 matrix filled with ones. In the last step above we have used the definition of the mutual-coherence and the fact that it bounds from above all normalized inner products of the columns of  $\mathbf{A}$ . Returning to the set  $C_s^1$ , we can write

$$C_s^1 \subseteq \left\{ \mathbf{e} \mid \begin{array}{l} \mathbf{e} \neq \mathbf{0}, \|\mathbf{e}\|_1 - 2\mathbf{1}_{k_0}^T \cdot |\mathbf{e}| \leq 0 \\ \text{and } |\mathbf{e}| \leq \frac{\mu(\mathbf{A})}{1 + \mu(\mathbf{A})} \mathbf{1} \cdot |\mathbf{e}| \end{array} \right\} = C_s^2. \quad (4.58)$$

The obtained set  $C_s^2$  is unbounded since if  $\mathbf{e} \in C_s^2$  then  $\alpha\mathbf{e} \in C_s^2$  for all  $\alpha \neq 0$ . Thus, in order to study its behavior, we can restrict our quest for normalized vectors,  $\|\mathbf{e}\|_1 = 1$ . This new set, denoted as  $C_r$  becomes

$$C_r = \left\{ \mathbf{e} \mid \|\mathbf{e}\|_1 = 1, 1 - 2\mathbf{1}_{k_0}^T \cdot |\mathbf{e}| \leq 0, \text{ and } |\mathbf{e}| \leq \frac{\mu(\mathbf{A})}{1 + \mu(\mathbf{A})} \mathbf{1} \right\}. \quad (4.59)$$

In the last condition we have used the relation  $\mathbf{1}|\mathbf{e}| = \mathbf{1} \cdot \mathbf{1}^T |\mathbf{e}|$ , and the fact that  $\mathbf{1}^T |\mathbf{e}| = \|\mathbf{e}\|_1 = 1$ .

In order for the vector  $\mathbf{e}$  to satisfy the requirement  $1 - 2\mathbf{1}_{k_0}^T \cdot |\mathbf{e}| \leq 0$ , one needs to concentrate its energy in its first  $k_0$  entries. However, the requirements  $\|\mathbf{e}\|_1 = 1$  and  $|e_j| \leq \mu(\mathbf{A})/(1 + \mu(\mathbf{A}))$  restrict these  $k_0$  entries to be exactly  $|e_j| = \mu(\mathbf{A})/(1 + \mu(\mathbf{A}))$ , because these are the maximal allowed values. Thus, returning to the first condition we get the requirement

$$1 - 2\mathbf{1}_{k_0}^T \cdot |\mathbf{e}| = 1 - 2k_0 \frac{\mu(\mathbf{A})}{1 + \mu(\mathbf{A})} \leq 0. \quad (4.60)$$

This means that if  $k_0$  is less than  $(1 + 1/\mu(\mathbf{A}))/2$ , the set will be necessarily empty, hence implying that Basis-Pursuit leads to the desired solution as claims.  $\square$

The above proof amounts to showing that if there are two solutions to an incoherent underdetermined system, one of them being sparse, moving along the line segment between the two solutions causes an increase in the  $\ell_1$ -norm as we move away from the sparse solution.

Historically, The general BP result was found before the OMP one, and the two were preceded by the two-ortho case analysis, which provided a kind of existence proof that something interesting could be said about solving underdetermined systems of equations under some conditions. The fact that the same form of assumptions lead to the same form of result for both algorithms is tantalizing: is there some deeper meaning? To our aid comes the two-ortho case, in which we saw that the two methods actually depart, with better performance to BP.

#### 4.2.4 Performance of Pursuit Algorithms – Summary

The above-described theorems motivate using OMP and BP to approximate the solution of  $(P_0)$ , and as such, they are conceptually important. However, these results are weak; they guarantee success only when the object is extremely sparse, with fewer than  $\sqrt{n}$  non-zeros out of  $n$ . In many cases such sparsity is out of the question.

Attempts to further strengthen the above results were made based on the *Babel function* (see Chapter 2), leading to somewhat sharper results. However, in cases where the matrix  $\mathbf{A}$  has an unfavorable coherence value, i.e.,  $\mu(\mathbf{A}) \approx 1$ , even the ‘strengthened’ conclusions are quite weak. Actually, incoherence offers only a partial description of the situations under which sparsity-seeking methods succeed. In the case of random matrices  $\mathbf{A}$ , much more favorable results are possible. Empirical evidence by extensive simulations consistently show that OMP and BP perform well even in situations violating the above bounds. This reflects the worst-case nature of the analysis behind the above results. We shall return to this matter at a later chapter, considering a probabilistic viewpoint that yields reasonable and hence more optimistic bounds.

### 4.3 The Role of the Sign-Pattern

Suppose that for a linear system of equations  $\mathbf{Ax} = \mathbf{b}$  we hold a candidate sparse solution  $\mathbf{x}$  with  $\|\mathbf{x}\|_0 < \text{spark}(\mathbf{A})/2$ . Clearly, this is the sparsest possible solution for this system, and thus, it is the global minimizer of  $(P_0)$ . In the analysis of BP (posed as the problem  $(P_1)$ ) we consider a competitive solution,  $\mathbf{y} \neq \mathbf{x}$ , such that  $\mathbf{Ay} = \mathbf{b}$ , which may lead to a shorter  $\ell_1$  length. Thus, we have studied the difference

$\|\mathbf{y}\|_1 - \|\mathbf{x}\|_1$ , such that if it is negative, it implies a failure of BP. In our attempt to simplify the analysis, we defined  $\mathbf{e} = \mathbf{y} - \mathbf{x}$ , and bounded this difference from below, by

$$\|\mathbf{e} + \mathbf{x}\|_1 - \|\mathbf{x}\|_1 \geq \mathbf{1}_{s^c}^T \cdot |\mathbf{e}| - \mathbf{1}_s^T \cdot |\mathbf{e}| = \|\mathbf{e}\|_1 - 2\mathbf{1}_s^T \cdot |\mathbf{e}|, \quad (4.61)$$

where  $\mathbf{1}_s$  and  $\mathbf{1}_{s^c}$  are indicator vectors, with 1-es on the support  $s$  of  $\mathbf{x}$  and zero elsewhere or vice-versa.

The vector  $\mathbf{z}_x = \text{sign}(\mathbf{x})$  is zero outside the support of  $\mathbf{x}$ , and its values are  $\pm 1$  on the support, depending on the sign of the entries in  $\mathbf{s}$ . Clearly,  $\mathbf{1}_s = |\mathbf{z}_x|$ , and thus, the above lower-bound can be written differently as

$$\|\mathbf{e} + \mathbf{x}\|_1 - \|\mathbf{x}\|_1 \geq \|\mathbf{e}\|_1 - 2|\mathbf{z}_x|^T \cdot |\mathbf{e}|. \quad (4.62)$$

However, this alternative expression is of no important consequence.

If  $\mathbf{y}$  is indeed a competitive solution with a shorter  $\ell_1$ -length (i.e.,  $\|\mathbf{x}\|_1 > \|\mathbf{y}\|_1$ ), then  $\|\mathbf{x} + \epsilon\mathbf{e}\|_1 < \|\mathbf{x}\|_1$  for any  $0 < \epsilon \leq 1$ , due to the convexity of the  $\ell_1$  function:

$$\begin{aligned} \|\mathbf{x} + \epsilon\mathbf{e}\|_1 &= \|(1 - \epsilon)\mathbf{x} + \epsilon\mathbf{y}\|_1 \\ &\leq (1 - \epsilon)\|\mathbf{x}\|_1 + \epsilon\|\mathbf{y}\|_1 \\ &= \|\mathbf{x}\|_1 - \epsilon(\|\mathbf{x}\|_1 - \|\mathbf{y}\|_1) < \|\mathbf{x}\|_1. \end{aligned} \quad (4.63)$$

Thus, we choose  $\epsilon$  such that  $\epsilon|\mathbf{e}| < |\mathbf{x}|$  on the support of  $\mathbf{x}$ , and clearly  $\epsilon > 0$ . With this change, we can write

$$\|\mathbf{e} + \mathbf{x}\|_1 - \|\mathbf{x}\|_1 = \mathbf{1}_{s^c}^T \cdot |\mathbf{e}| + \mathbf{z}_x^T \cdot \mathbf{e}. \quad (4.64)$$

The first term,  $\mathbf{1}_{s^c}^T \cdot |\mathbf{e}|$  corresponds to the off-support part of the summation. The second term,  $\mathbf{z}_x^T \cdot \mathbf{e}$ , stems from the formula<sup>2</sup>  $|a + b| - |a| = \text{sign}(a)b$  for  $|b| < |a|$ . Adding and subtracting  $\mathbf{1}_s^T \cdot |\mathbf{e}| = |\mathbf{z}_x|^T \cdot |\mathbf{e}|$  in the above equation leads to

$$\|\mathbf{e} + \mathbf{x}\|_1 - \|\mathbf{x}\|_1 = \mathbf{1}_{s^c}^T \cdot |\mathbf{e}| + \mathbf{z}_x^T \cdot \mathbf{e} = \|\mathbf{e}\|_1 - |\mathbf{z}_x|^T \cdot |\mathbf{e}| + \mathbf{z}_x^T \cdot \mathbf{e}. \quad (4.65)$$

Note that this is not a bound, but an exact equality. Therefore, the study of the success of BP concentrates on checking whether the set

$$\{\mathbf{e} \mid \mathbf{e} \neq \mathbf{0}, \|\mathbf{e}\|_1 - |\mathbf{z}_x|^T \cdot |\mathbf{e}| + \mathbf{z}_x^T \cdot \mathbf{e} < 0, \text{ and } \mathbf{A}\mathbf{e} = \mathbf{0}\} \quad (4.66)$$

is non-empty. This implies that in analyzing whether there could exist a competitive solution to  $\mathbf{x}$ , all that is actually used is its sign-pattern vector  $\mathbf{z}_x$ , and any two sparse vectors having the same sign-pattern will necessarily behave the same (lead to success or failure). Thus, in checking the performance of BP for representations with cardinality  $k$ , all we need to do is to sweep through all the possible  $2^{k-1}$  sign-

<sup>2</sup> This formula is easy to verify: If  $a \geq 0$  and  $-a \leq b \leq a$ , then  $|a + b| - |a| = a + b - a = b$ . Similarly, if  $a \leq 0$  we have  $|a + b| - |a| = -a - b + a = -b$ , as claimed.

patterns,<sup>3</sup> and check for each. While this is unreasonable for large values of  $k$ , it is still far better than a test that takes into account magnitudes of the non-zeros as well.

## 4.4 Tropp's Exact Recovery Condition

It would be impossible to conclude our discussion on the guarantee results for pursuit performance without mentioning the alternative and illuminating analysis developed by Joel A. Tropp in his “Greed Is Good” paper. This analysis is based on the definition of an *Exact Recovery Condition* (ERC). While this condition by itself is non-constructive, it brings an additional and interesting insight towards the performance limits of the OMP and the BP, and it eventually leads to a different and easier way to prove Theorems 4.3 and 4.5.

We shall denote a support of interest as  $\mathcal{S}$ , and the sub-matrix from  $\mathbf{A}$  that contains the columns of this support by  $\mathbf{A}_{\mathcal{S}}$ . We now turn to define the ERC.

**Definition 4.1.** For a given support  $\mathcal{S}$  and for a matrix  $\mathbf{A}$ , the Exact Recovery Condition (ERC) is given by

$$ERC(\mathbf{A}, \mathcal{S}) : \max_{i \notin \mathcal{S}} \|\mathbf{A}_{\mathcal{S}}^{\dagger} \mathbf{a}_i\|_1 < 1. \quad (4.67)$$

This condition essentially considers linear systems of equations of the form  $\mathbf{A}_{\mathcal{S}} \mathbf{x} = \mathbf{a}_i$ , where  $\mathbf{a}_i$  is a column from  $\mathbf{A}$  that is outside the support  $\mathcal{S}$ . The ERC states that the minimum ( $\ell_2$ -)energy solution to *all* these systems should all have an  $\ell_1$ -length smaller than 1.

The importance of the ERC comes from its strong connection to the success of pursuit techniques:

**Theorem 4.6. (ERC and Pursuit Performance:)** *For a sparse vector  $\mathbf{x}$  over the support  $\mathcal{S}$ , which is the solution of the linear system  $\mathbf{A}\mathbf{x} = \mathbf{b}$ , if the ERC is met, then both OMP and BP are guaranteed to succeed in recovering  $\mathbf{x}$ .*

While this is a stronger condition for the success of the pursuit algorithms, compared to the ones posed using the mutual-coherence, it is not constructive – one cannot test it unless the support is known. Furthermore, if we only know the cardinality of the solution,  $|\mathcal{S}|$ , and we desire to use the ERC, we should test this condition for all the  $\binom{m}{|\mathcal{S}|}$  possibilities, in order to guarantee a success, and this is of course prohibitive. Let us prove the above theorem, and show how the ERC is tied to the OMP and BP success, and then discuss what should be done to turn this vague test into a more practical claim. Our next discussion follows the steps taken by Tropp.

**Proof:** We start with the OMP analysis, and assume that we are at the first stage of the algorithm. For this round to end well, we should require that the term  $\|\mathbf{A}_{\mathcal{S}}^T \mathbf{b}\|_{\infty}$

<sup>3</sup> Note that if a given sign-pattern is known to lead to failure/success, this immediately states the same for its negative pattern,  $-\mathbf{z}_x$ , simply by multiplying the solution for  $\mathbf{e}$  by  $-1$ .

(maximal inner product of  $\mathbf{b}$  with one of the support atoms) is larger than the maximal inner product with atoms outside the support,  $\|\mathbf{A}_{S^c}^T \mathbf{b}\|_\infty$ . Thus, we require

$$\rho = \frac{\|\mathbf{A}_{S^c}^T \mathbf{b}\|_\infty}{\|\mathbf{A}_S^T \mathbf{b}\|_\infty} < 1. \quad (4.68)$$

Now, recall that  $\mathbf{b} = \mathbf{A}_S \mathbf{x}_S$ , and thus

$$\begin{aligned} (\mathbf{A}_S^T)^+ \mathbf{A}_S^T \mathbf{b} &= \mathbf{A}_S (\mathbf{A}_S^T \mathbf{A}_S)^{-1} \mathbf{A}_S^T \mathbf{b} \\ &= \mathbf{A}_S (\mathbf{A}_S^T \mathbf{A}_S)^{-1} \mathbf{A}_S^T \mathbf{A}_S \mathbf{x}_S \\ &= \mathbf{A}_S \mathbf{x}_S \\ &= \mathbf{b}, \end{aligned}$$

as this is a linear projection operation onto the span of  $\mathbf{A}_S$ . Thus,

$$\rho = \frac{\|\mathbf{A}_{S^c}^T \mathbf{b}\|_\infty}{\|\mathbf{A}_S^T \mathbf{b}\|_\infty} = \frac{\|\mathbf{A}_{S^c}^T (\mathbf{A}_S^T)^+ \mathbf{A}_S^T \mathbf{b}\|_\infty}{\|\mathbf{A}_S^T \mathbf{b}\|_\infty} \leq \|\mathbf{A}_{S^c}^T (\mathbf{A}_S^T)^+\|_\infty. \quad (4.69)$$

Here we have replaced the original ratio with the  $\ell_\infty$ -induced norm.<sup>4</sup> on the operator  $\mathbf{A}_{S^c}^T (\mathbf{A}_S^T)^+$ . This norm is computed as the maximum absolute row-sum ( $\|\mathbf{B}\|_\infty = \max_i \sum_j |b_{ij}|$ ), and it is equal to the maximum absolute column-sum of the transposed matrix, which is non other than the  $\ell_1$ -induced norm,  $\|\mathbf{B}\|_1 = \max_j \sum_i |b_{ij}|$ ). Note that the pseudo-inverse and the transpose operations applied on  $\mathbf{A}_S$  commute. Thus we obtain

$$\|\mathbf{A}_{S^c}^T (\mathbf{A}_S^T)^+\|_\infty = \|\mathbf{A}_S^+ \mathbf{A}_{S^c}\|_1. \quad (4.70)$$

Thus, by requiring  $\|\mathbf{A}_S^+ \mathbf{A}_{S^c}\|_1 < 1$ , we guarantee that  $\rho < 1$  and thus the first step of the OMP is successful. The obtained requirement is non-other than the ERC, as posed in Equation (4.67) (as it is the maximum over all the absolute column-sums).

The above implies that the first atom to be chosen is from within the true support  $S$ , and thus the residual obtained as a consequence is necessarily within the span of  $\mathbf{A}_S$  as before. Thus, all the above analysis remains the same, with the residual  $\mathbf{r}_k$  replacing  $\mathbf{b}$ , and this leads to the success of all  $|S|$  steps of the overall OMP, as claimed.

Turning to the analysis of the BP, we consider a candidate alternative solution  $\mathbf{y}$ , as we did in Section 4.2.3. We define the difference between this solution and the true one as  $\mathbf{e} = \mathbf{y} - \mathbf{x}$ , and we already saw that this implies that  $\mathbf{e}$  in the null-space of  $\mathbf{A}$ . Breaking  $\mathbf{e}$  into two parts  $\mathbf{e}_S$  and  $\mathbf{e}_{S^c}$ , such that  $\mathbf{0} = \mathbf{A}\mathbf{e} = \mathbf{A}_S \mathbf{e}_S + \mathbf{A}_{S^c} \mathbf{e}_{S^c}$ , we get that

$$\mathbf{e}_S = -\mathbf{A}_S^+ \mathbf{A}_{S^c} \mathbf{e}_{S^c}. \quad (4.71)$$

<sup>4</sup> An induced-norm of a matrix  $\mathbf{B}$  is defined via a vector norm, by maximizing the norm of the vector  $\|\mathbf{B}\mathbf{v}\|$  over all vectors  $\mathbf{v}$  on the sphere  $\|\mathbf{v}\| = 1$ , i.e.,  $\|\mathbf{B}\| = \max_{\mathbf{v}} \|\mathbf{B}\mathbf{v}\|/\|\mathbf{v}\|$ .

We already saw in Equation (4.53) that for the BP to succeed, we should demand that this error vector concentrates most of its  $\ell_1$ -energy outside the support, namely,

$$\|\mathbf{e}_S\|_1 < \|\mathbf{e}_{S^c}\|_1. \quad (4.72)$$

Plugging in the relation we have in Equation (4.71), this reads

$$\|\mathbf{e}_S\|_1 = \|\mathbf{A}_S^+ \mathbf{A}_{S^c} \mathbf{e}_{S^c}\|_1 < \|\mathbf{e}_{S^c}\|_1. \quad (4.73)$$

Since  $\|\mathbf{A}_S^+ \mathbf{A}_{S^c} \mathbf{e}_{S^c}\|_1 \leq \|\mathbf{A}_S^+ \mathbf{A}_{S^c}\|_1 \cdot \|\mathbf{e}_{S^c}\|_1$  (with the  $\ell_1$ -induced norm), this means that if the ERC is met, BP is guaranteed to succeed, as claimed.  $\square$

In order to make all the above practical, one needs to find a simple alternative to verify the correctness of the ERC. As Tropp shows in his work, this can be done using the mutual-coherence or the babel function. We shall concentrate here in the relation to the mutual-coherence, in a way that immediately reveals the connection between this discussion and Theorems 4.3 and 4.5.

**Theorem 4.7.** *For a matrix  $\mathbf{A}$  with mutual-coherence  $\mu(\mathbf{A})$ , if  $k < 0.5(1 + 1/\mu(\mathbf{A}))$ , then for all supports  $S$  with cardinality equal or smaller than  $k$ , the ERC is satisfied.*

**Proof:** For an arbitrary  $i$ -th column from the off-support, we require  $\|\mathbf{A}_S^+ \mathbf{a}_i\|_1 = \|(\mathbf{A}_S^T \mathbf{A}_S)^{-1} \mathbf{A}_S^T \mathbf{a}_i\|_1 < 1$ . Replacing this with a more strict requirement can be done by demanding  $\|(\mathbf{A}_S^T \mathbf{A}_S)^{-1}\|_1 \cdot \|\mathbf{A}_S^T \mathbf{a}_i\|_1 < 1$ . The vector  $\mathbf{A}_S^T \mathbf{a}_i$  has entries in the range  $[-\mu(\mathbf{A}), \mu(\mathbf{A})]$ , and thus  $\|\mathbf{A}_S^T \mathbf{a}_i\|_1 \leq |S|\mu(\mathbf{A})$ .

The matrix  $\mathbf{A}_S^T \mathbf{A}_S$  is strictly diagonally-dominant (SDD), for which using the Ahlberg – Nilson – Varah bound we obtain<sup>5</sup>

$$\|(\mathbf{A}_S^T \mathbf{A}_S)^{-1}\|_1 \leq \frac{1}{1 - (|S| - 1)\mu(\mathbf{A})}. \quad (4.74)$$

Combined with the above we get the condition

$$\|(\mathbf{A}_S^T \mathbf{A}_S)^{-1}\|_1 \cdot \|\mathbf{A}_S^T \mathbf{a}_i\|_1 \leq \frac{|S|\mu(\mathbf{A})}{1 - (|S| - 1)\mu(\mathbf{A})} < 1. \quad (4.75)$$

This leads to the claimed relationship  $|S| < 0.5(1 + 1/\mu(\mathbf{A}))$ . As this condition is the same for all columns  $\mathbf{a}_i$ , we conclude that if  $|S| < 0.5(1 + 1/\mu(\mathbf{A}))$ , the ERC is satisfied.  $\square$

---

<sup>5</sup> Proving this very same bound for the  $\ell_2$ -induced norm  $\|(\mathbf{A}_S^T \mathbf{A}_S)^{-1}\|_2$  is very easy, stemming from the minimal eigenvalue of  $\mathbf{A}_S^T \mathbf{A}_S$  being  $1 - (|S| - 1)\mu(\mathbf{A})$ . However, for the  $\ell_1$ -induced norm, we rely on the above-mentioned paper.



## 4.5 Summary

In a concentrated effort during the years 2001–2006, the above-described equivalence results have been obtained in a series of works. While the ability to guarantee the success of pursuit techniques is surprising and encouraging, it is heavily shadowed by the relatively pessimistic bounds presented here. In Chapter 7 we return to this problem, demonstrate it, and describe more recent work that aimed at getting more optimistic bounds to reflect the empirical observations.

Another limitation of the results posed here is the fact that  $(P_0)$  is rarely the practical problem we would be interested to solve, as requiring exact equality  $\mathbf{Ax} = \mathbf{b}$  is too strict and thus naive. We now turn to a new chapter that relaxes this assumption.

## Further Reading

1. J.H. Ahlberg and E.N. Nilson, Convergence properties of the spline fit, *J. SIAM*, 11:95–104, 1963.
2. S.S. Chen, D.L. Donoho, and M.A. Saunders, Atomic decomposition by basis pursuit, *SIAM Journal on Scientific Computing*, 20(1):33–61 (1998).
3. S.S. Chen, D.L. Donoho, and M.A. Saunders, Atomic decomposition by basis pursuit, *SIAM Review*, 43(1):129–159, 2001.
4. C. Couvreur and Y. Bresler, On the optimality of the Backward Greedy Algorithm for the subset selection problem, *SIAM Journal on Matrix Analysis and Applications*, 21(3):797–808, 2000.
5. D.L. Donoho and M. Elad, Optimally sparse representation in general (non-orthogonal) dictionaries via  $\ell_1$  minimization, *Proc. of the National Academy of Sciences*, 100(5):2197–2202, 2003.
6. D.L. Donoho and X. Huo, Uncertainty principles and ideal atomic decomposition, *IEEE Trans. on Information Theory*, 47(7):2845–2862, 1999.
7. D.L. Donoho and J. Tanner, Sparse nonnegative solutions of underdetermined linear equations by linear programming, *Proceedings of the National Academy of Sciences*, 102(27):9446–9451, July 2005.
8. D.L. Donoho and J. Tanner, Neighborliness of randomly-projected Simplices in high dimensions, *Proceedings of the National Academy of Sciences*, 102(27):9452–9457, July 2005.
9. M. Elad and A.M. Bruckstein, A generalized uncertainty principle and sparse representation in pairs of bases, *IEEE Trans. on Information Theory*, 48:2558–2567, 2002.
10. A. Feuer and A. Nemirovsky, On sparse representation in pairs of bases, *IEEE Trans. on Information Theory*, 49:1579–1581, June 2002.
11. J.J. Fuchs, On sparse representations in arbitrary redundant bases, *IEEE Trans. on Information Theory*, 50:1341–1344, 2004.
12. R. Gribonval and M. Nielsen, Sparse decompositions in unions of bases, *IEEE Trans. on Information Theory*, 49(12):3320–3325, 2003.

13. X. Huo, Sparse Image representation Via Combined Transforms, PhD thesis, Stanford, 1999.
14. S. Mallat and Z. Zhang, Matching pursuits with time-frequency dictionaries, *IEEE Trans. Signal Processing*, 41(12):3397–3415, 1993.
15. N. Morača, Bounds for norms of the matrix inverse and the smallest singular value, *Linear Algebra and its Applications*, 429:2589–2601, 2008.
16. J.A. Tropp, Greed is good: Algorithmic results for sparse approximation, *IEEE Trans. on Information Theory*, 50(10):2231–2242, October 2004.
17. J.M. Varah, A lower bound for the smallest singular value of a matrix, *Linear Algebra Appl.* 11:3–5, 1975.



## Chapter 5

# From Exact to Approximate Solutions

### 5.1 General Motivation

The exact constraint  $\mathbf{Ax} = \mathbf{b}$  is often relaxed, with an approximated equality measured using the quadratic penalty function  $Q(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|_2^2$ . Such relaxation allows us to (i) define a quasi-solution in case no exact solution exists (even in cases where  $\mathbf{A}$  has more rows than columns); (ii) exploit ideas from optimization theory; (iii) measure the quality of a candidate solution; and more.

Following the rationale of the previous sections, one may reconsider  $(P_0)$  and tolerate a slight discrepancy between  $\mathbf{Ax}$  and  $\mathbf{b}$ . We define an error-tolerant version of  $(P_0)$ , with error tolerance  $\epsilon > 0$ , by

$$(P_0^\epsilon) : \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{b} - \mathbf{Ax}\|_2 \leq \epsilon. \quad (5.1)$$

The  $\ell_2$ -norm used here for evaluating the error  $\mathbf{b} - \mathbf{Ax}$  can be replaced by other options, such as  $\ell_1$ ,  $\ell_\infty$ , or a weighted  $\ell_2$ -norm.

In this problem a discrepancy of size  $\epsilon$  is permitted between a proposed representation  $\mathbf{Ax}$  and the signal  $\mathbf{b}$ . When  $(P_0)$  and  $(P_0^\epsilon)$  are applied on the same problem instance, the error-tolerant problem must always give results at least as sparse as those arising  $(P_0)$ , since the feasible set is wider. Indeed, for a typical general problem instance  $(\mathbf{A}, \mathbf{b})$ , the solution of  $(P_0)$  will have  $n$  nonzeros. On the other hand, in some real-world problems (as we shall see below), although the solution of  $(P_0)$  would be dense, the solution of  $(P_0^\epsilon)$  can be seen to have far fewer non-zeros, tending towards sparsity.

An alternative and more natural interpretation for the problem  $(P_0^\epsilon)$  is one of noise removal. Consider a sufficiently sparse vector  $\mathbf{x}_0$ , and assume that  $\mathbf{b} = \mathbf{Ax}_0 + \mathbf{e}$ , where  $\mathbf{e}$  is a nuisance vector of finite energy  $\|\mathbf{e}\|_2^2 = \epsilon^2$ . Roughly speaking  $(P_0^\epsilon)$  aims to find  $\mathbf{x}_0$ , i.e., to do roughly the same thing as  $(P_0)$  would do on noiseless data  $\mathbf{b} = \mathbf{Ax}_0$ . In later chapters we shall return to this interpretation and make it precise, using statistical estimators that lead to formulations similar to  $(P_0^\epsilon)$ .

Past work studied this problem in various forms, and in this chapter we discuss some of what is now known. Results are in some ways parallel to those in the noiseless case. Specifically, we should discuss the uniqueness property – conditions under which a sufficiently sparse solution is known to be the global minimizer of  $(P_0^\epsilon)$ ; practical pursuit techniques to approximate the solution of this problems; and equivalence – theoretical guarantees for their successful recovery of the desired solution. As we shall see next, however, the notions of uniqueness and equivalence no longer apply – they are replaced by the notion of stability.

## 5.2 Stability of the Sparsest Solution

Before turning to approximate the solution of  $(P_0^\epsilon)$ , we must answer a far more fundamental question: Suppose that a sparse vector  $\mathbf{x}_0$  is multiplied by  $\mathbf{A}$ , and we 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  $(P_0^\epsilon)$  to obtain an approximation to  $\mathbf{x}_0$ , and getting a solution  $\mathbf{x}_0^\epsilon$ ,

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

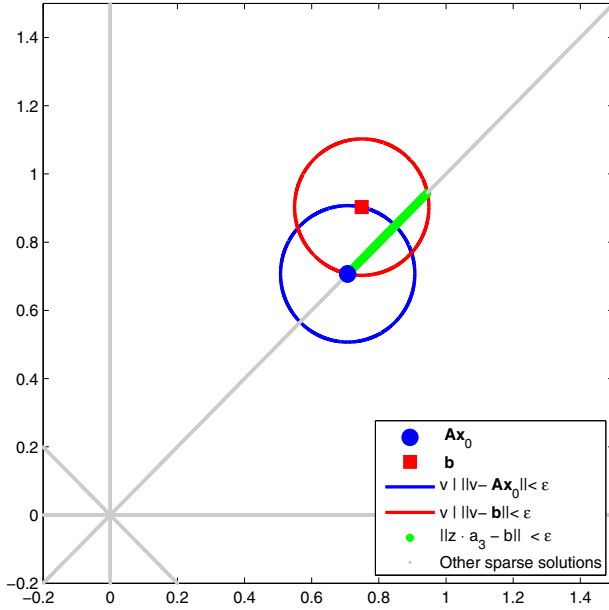
How good shall this approximation be? How its accuracy is affected by the sparsity of  $\mathbf{x}_0$ ? These questions are the natural extension for the uniqueness property of sparse solutions we have discussed in the context of the  $(P_0)$  problem in Chapter 2.

### 5.2.1 Uniqueness versus Stability – Gaining Intuition

As we show next, we can no longer claim uniqueness for  $(P_0^\epsilon)$  in the general case. In order to demonstrate that, we present a simple experiment:  $\mathbf{A}$  is chosen to be the two-ortho  $[\mathbf{I}, \mathbf{F}]$  of size  $2 \times 4$ ,  $\mathbf{x}_0 = [0 \ 0 \ 1 \ 0]^T$ . We generate a random noise  $\mathbf{e}$  with pre-specified norm  $\epsilon = \|\mathbf{e}\|_2$  and create the vector  $\mathbf{b} = \mathbf{A}\mathbf{x}_0 + \mathbf{e}$ . Thus,

$$\mathbf{A}\mathbf{x}_0 + \mathbf{e} = \begin{bmatrix} 1 & 0 & 0.707 & 0.707 \\ 0 & 1 & 0.707 & -0.707 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} = \begin{bmatrix} 0.707 + e_1 \\ 0.707 + e_2 \end{bmatrix}$$

Figure 5.1 presents the locations of  $\mathbf{A}\mathbf{x}_0$ ,  $\mathbf{b}$ , 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  $\epsilon = 0.2$ . The last of those is the region of the image (i.e., after multiplication by  $\mathbf{A}$ ) of all the feasible solutions  $\mathbf{x}$  to the problem  $(P_0^\epsilon)$ . Naturally, the vector  $\mathbf{x}_0$  poses a feasible solution, while being very sparse. Indeed, it is an optimal solution to  $(P_0^\epsilon)$ , in the sense that no sparser solution exists (a sparser solution is only the null vector, and it is outside the feasible set). Could there be alternative feasible



**Fig. 5.1** A 2D demonstration of the lack of uniqueness for the noisy case, with a relatively weak noise.

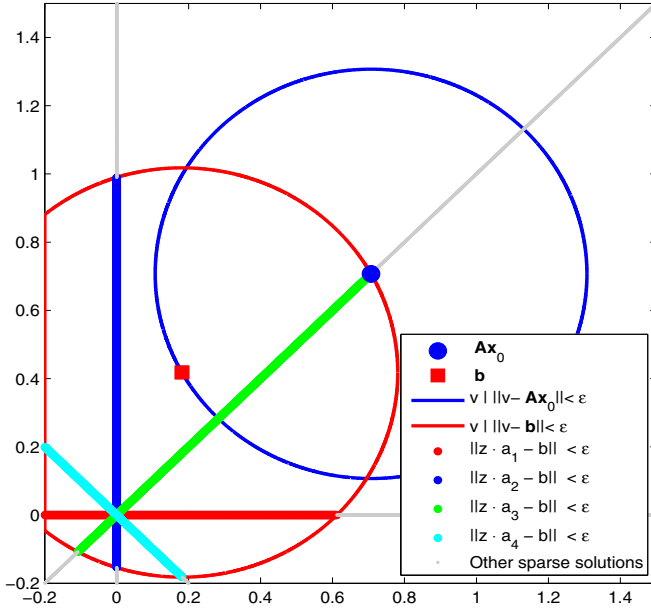
solutions that are sparse? **Figure 5.1** shows that solutions of the form  $\mathbf{x}_0 = [0 \ 0 \ x \ 0]^T$  with some values of  $x$  are also feasible, while having the same cardinality.

**Figure 5.2** presents the same experiment, this time with a stronger noise,  $\epsilon = 0.6$ . 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, and in fact, even the null solution is included, implying that this is the optimal solution to  $(P_0^\epsilon)$ .

Here is a more formal way of explaining this. We shall denote  $\mathbf{x}_S$  and  $\mathbf{A}_S$  the portions of  $\mathbf{x}$  and  $\mathbf{A}$  that contain the support  $S$  elements/columns, respectively. Suppose that  $\mathbf{x}$  is a sparse candidate solution to this problem over the support  $S$ , with  $\|\mathbf{x}\|_0 = |S|$ , and it satisfies the constraint,  $\|\mathbf{b} - \mathbf{A}_S \mathbf{x}_S\|_2 \leq \epsilon$ .

If it so happens that  $\mathbf{x}_S$  is also the minimizer of the term  $f_S(\mathbf{z}) = \|\mathbf{b} - \mathbf{A}_S \mathbf{z}\|_2$ , and  $f_S(\mathbf{x}_S^{opt}) = \epsilon$ , we can propose no alternative solution over this support, since any perturbation around  $\mathbf{x}_S$  leads to an increase in this term and thus violation of the constraint. In terms of **Figure 5.1**, this case takes place when the closest point to  $\mathbf{b}$  on the green line is  $\mathbf{A}\mathbf{x}_0$ , or put differently, if the distortion  $\mathbf{e} = \mathbf{b} - \mathbf{A}_S \mathbf{x}_S$  is orthogonal to the columns of  $\mathbf{A}_S$ .<sup>1</sup> In all other cases, the fact that  $\min_{\mathbf{z}} f_S(\mathbf{z}) < \epsilon$  implies an ability to perturb the so-called optimal solution  $\mathbf{x}_S$  in a way that preserves its feasibility and the support, and thus we get a set of solutions that are as good as  $\mathbf{x}$ .

<sup>1</sup> As the minimizer of  $f_S(\mathbf{z}) = \|\mathbf{b} - \mathbf{A}_S \mathbf{z}\|_2$ , the vector  $\mathbf{x}_S$  should satisfy  $\mathbf{A}_S^T(\mathbf{b} - \mathbf{A}_S \mathbf{x}_S) = \mathbf{A}_S^T \mathbf{e} = \mathbf{0}$ .



**Fig. 5.2** A 2D demonstration of the lack of uniqueness for the noisy case, as shown in Figure 5.1, but with a stronger noise, that permits alternative solutions with a different support.

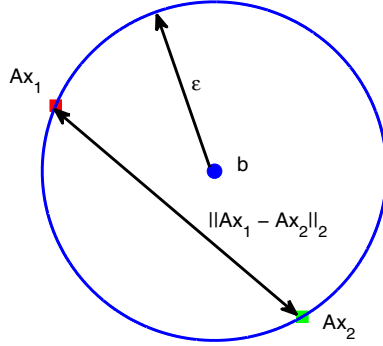
Furthermore, if some of the non-zero entries in  $\mathbf{x}$  are small enough, this perturbation may null them, leading to a sparser solution.

### 5.2.2 Theoretical Study of the Stability of $(P^\epsilon)$

So, returning to the original question we have posed, instead of claiming uniqueness of a sparse solution, we replace this with a notion of stability – a claim that if a sufficiently sparse solution is found, then all alternative solutions necessarily reside very close to it. The following analysis, taken from the work by Donoho, Elad, and Temlyakov, leads to a stability claim of this sort.

We start by returning to the definition of the *spark* and extending it by considering a relaxed notion of linear-dependency. In the noiseless case we considered two competing solutions  $\mathbf{x}_1$  and  $\mathbf{x}_2$  to the linear system  $\mathbf{Ax} = \mathbf{b}$ , and this led to the relation  $\mathbf{A}(\mathbf{x}_1 - \mathbf{x}_2) = \mathbf{Ad} = \mathbf{0}$ . This motivates a study of the sparsity of vectors  $\mathbf{d}$  in the null-space of  $\mathbf{A}$ , which naturally leads to the definition of the *spark*.

Following the same rationale, we should consider now two feasible solutions  $\mathbf{x}_1$  and  $\mathbf{x}_2$  to the requirement  $\|\mathbf{Ax} - \mathbf{b}\|_2 \leq \epsilon$ . Considering  $\mathbf{b}$  as the center of a sphere of radius  $\epsilon$ , both  $\mathbf{Ax}_1$  and  $\mathbf{Ax}_2$  reside in it or on its surface. Thus, the distance between



**Fig. 5.3** The vector  $\mathbf{b}$  as the center of a circle, and the image of two candidate solutions,  $\mathbf{Ax}_1$  and  $\mathbf{Ax}_2$  that are  $\epsilon$  away. This leads to a distance of  $2\epsilon$  at the most between  $\mathbf{Ax}_1$  and  $\mathbf{Ax}_2$ .

these two vectors is  $2\epsilon$  at the most, as shown in Figure 5.3, and we have the relation  $\|\mathbf{A}(\mathbf{x}_1 - \mathbf{x}_2)\|_2 = \|\mathbf{Ad}\|_2 \leq 2\epsilon$ . A different way to see this is the following, which exploits the triangle inequality:

$$\begin{aligned} \|\mathbf{A}(\mathbf{x}_1 - \mathbf{x}_2)\|_2 &= \|\mathbf{Ax}_1 - \mathbf{b} + \mathbf{b} - \mathbf{Ax}_2\|_2 \\ &\leq \|\mathbf{Ax}_1 - \mathbf{b}\|_2 + \|\mathbf{Ax}_2 - \mathbf{b}\|_2 \leq 2\epsilon. \end{aligned} \quad (5.2)$$

Therefore we should generalize the *spark* to allow for  $\epsilon$ -proximity to the null-space, as the following definition suggests.

**Definition 5.1.** Given a matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$ , we consider all possible sub-sets of  $s$  columns, each such set forms a sub-matrix  $\mathbf{A}_s \in \mathbb{R}^{n \times s}$ . We define  $\text{spark}_\eta(\mathbf{A})$  as the smallest possible  $s$  (number of columns) that guarantees

$$\min_s \sigma_s(\mathbf{A}_s) \leq \eta. \quad (5.3)$$

In words, this is the smallest (integer) number of columns that can be gathered from  $\mathbf{A}$ , such that the smallest singular-value<sup>2</sup> of  $\mathbf{A}_s$  is no larger than  $\eta$ .

For  $\eta = 0$ , this demand reduces to linear dependence of the chosen columns, and thus we have  $\text{spark}_0(\mathbf{A}) = \text{spark}(\mathbf{A})$ . Naturally,  $\text{spark}_\eta(\mathbf{A})$  is monotone decreasing in  $\eta$ , and

$$\forall 0 \leq \eta \leq 1, \quad 1 \leq \text{spark}_\eta(\mathbf{A}) \leq \text{spark}(\mathbf{A}) \leq n + 1. \quad (5.4)$$

<sup>2</sup> This is the last one among the  $s$  singular-values, and its index is therefore  $s$ .



We also know that for  $\eta = 1$  we get  $\text{spark}_\eta(\mathbf{A}) = 1$ , since any single (and normalized) column has one singular-value equal to 1.

A fundamental property of the *spark* is that the equation  $\mathbf{A}\mathbf{v} = 0$  implies  $\|\mathbf{v}\|_0 \geq \text{spark}(\mathbf{A})$ . Similarly, a generalized version of this property is the following:

**Lemma 5.1.** *If  $\|\mathbf{A}\mathbf{v}\|_2 \leq \eta$  and  $\|\mathbf{v}\|_2 = 1$ , then  $\|\mathbf{v}\|_0 \geq \text{spark}_\eta(\mathbf{A})$ .*

**Proof:** This property is a direct consequence of the above definition, and the fundamental properties of singular-values. By the definition of  $\text{spark}_\eta(\mathbf{A})$ , there exists at least one subset of columns from  $\mathbf{A}$  with  $\text{spark}_\eta(\mathbf{A})$  columns, such that its smallest singular-value is  $\eta$  or below. Also, the definition states that any set of less than  $\text{spark}_\eta(\mathbf{A})$  columns guarantees that all their singular-values are strictly above  $\eta$ .

Now, consider a case where  $\|\mathbf{v}\|_0 = s < \text{spark}_\eta(\mathbf{A})$ . By fundamental properties of singular-values, the corresponding sub-matrix  $\mathbf{A}_s$  of size  $n \times s$  ( $s < n$ ) must satisfy  $\|\mathbf{A}\mathbf{v}\|_2 = \|\mathbf{A}_s \mathbf{v}_s\|_2 \geq \sigma_s \|\mathbf{v}_s\|_2 = \sigma_s$  for any normalized vector  $\mathbf{v}$ . Since  $s < \text{spark}_\eta(\mathbf{A})$  we know that  $\sigma_s > \eta$ , and thus we got that  $\|\mathbf{A}\mathbf{v}\|_2 \geq \sigma_s > \eta$ . As this contradicts the knowledge that  $\|\mathbf{A}\mathbf{v}\|_2 \leq \eta$ , we conclude that  $s = \|\mathbf{v}\|_0 \geq \text{spark}_\eta(\mathbf{A})$ .  $\square$

We now turn to a more fundamental observation, that ties  $\text{spark}_\eta(\mathbf{A})$  to the mutual-coherence of this matrix:

**Lemma 5.2.** *If  $\mathbf{A}$  has normalized columns and mutual-coherence  $\mu(\mathbf{A})$ , then*

$$\text{spark}_\eta(\mathbf{A}) \geq \frac{1 - \eta^2}{\mu(\mathbf{A})} + 1. \quad (5.5)$$

**Proof:** We use a simple observation about eigenvalues of diagonally-dominant matrices, based on Gershgorin's disk theorem (see Chapter 2). Given an  $s$ -by- $s$  symmetric matrix  $\mathbf{H}$  with diagonal entries equal to 1 and off-diagonal entries  $\leq \mu$  in amplitude, we have that the smallest eigenvalue of  $\mathbf{H}$  can be bounded from below by  $\lambda_{\min}(\mathbf{H}) \geq 1 - (s - 1)\mu$ .

With  $\mathbf{A}$  obeying the above assumptions (normalized columns, and fixed coherence), every leading minor in the Gram matrix  $\mathbf{G} = \mathbf{A}^T \mathbf{A}$  has ones on the diagonal (assumed normalized columns), and its off-diagonal entries are bounded by  $\mu(\mathbf{A})$  in absolute value. Thus, the above claim gives a lower bound on the minimum eigenvalue of every such minor of size  $s$  by  $s$ , denoted by  $\mathbf{A}_s^T \mathbf{A}_s$ . We have

$$\begin{aligned} \lambda_{\min}(\mathbf{A}_s^T \mathbf{A}_s) &\geq 1 - (s - 1)\mu(\mathbf{A}). \\ \Rightarrow s &\geq \frac{1 - \lambda_{\min}(\mathbf{A}_s^T \mathbf{A}_s)}{\mu(\mathbf{A})} + 1. \end{aligned} \quad (5.6)$$

For the choice  $\lambda_{\min}(\mathbf{A}_s^T \mathbf{A}_s) \leq \eta^2$ , every set of  $s$  such columns have singular-values bounded from above by  $\eta$  and thus  $\text{spark}_\eta(\mathbf{A}) \geq \frac{1 - \eta^2}{\mu(\mathbf{A})} + 1$ .  $\square$

Based on the above developed tools, we now turn to present a variant of the uncertainty result we had for the noiseless case.

**Lemma 5.3.** *If  $\mathbf{x}_1$  and  $\mathbf{x}_2$  satisfy  $\|\mathbf{b} - \mathbf{A}\mathbf{x}_i\|_2 \leq \epsilon$ ,  $i = 1, 2$ , then*

$$\|\mathbf{x}_1\|_0 + \|\mathbf{x}_2\|_0 \geq \text{spark}_\eta(\mathbf{A}), \text{ where } \eta = \frac{2\epsilon}{\|\mathbf{x}_1 - \mathbf{x}_2\|_2}. \quad (5.7)$$

**Proof.** The triangle inequality yields  $\|\mathbf{A}(\mathbf{x}_1 - \mathbf{x}_2)\|_2 \leq 2\epsilon$ . Put differently, we have  $\|\mathbf{A}\mathbf{v}\|_2 \leq \eta$ , where  $\mathbf{v} = (\mathbf{x}_1 - \mathbf{x}_2)/\|\mathbf{x}_1 - \mathbf{x}_2\|_2$ . Based on the property shown in Lemma 5.1, we get  $\|\mathbf{v}\|_0 \geq \text{spark}_\eta(\mathbf{A})$ . But

$$\|\mathbf{x}_1\|_0 + \|\mathbf{x}_2\|_0 \geq \|\mathbf{x}_1 - \mathbf{x}_2\|_0 = \|\mathbf{v}\|_0, \quad (5.8)$$

which establishes (5.7).  $\square$

Now, just as for the exact case, we use the above uncertainly rule to derive a uniqueness result, but this time in the form of a localization to a single Euclidean ball.

**Theorem 5.1.** *Given a distance  $D \geq 0$  and  $\epsilon$ , set  $\eta = 2\epsilon/D$ . Suppose there are two approximate representations  $\mathbf{x}_i$ ,  $i = 1, 2$  both obeying*

$$\|\mathbf{b} - \mathbf{A}\mathbf{x}_i\|_2 \leq \epsilon \text{ and } \|\mathbf{x}_i\|_0 \leq \frac{1}{2}\text{spark}_\eta(\mathbf{A}). \quad (5.9)$$

*Then  $\|\mathbf{x}_1 - \mathbf{x}_2\|_2 \leq D$ .*

**Proof:** Using the details above and Lemma 5.3 we have that

$$\text{spark}_\eta(\mathbf{A}) \geq \|\mathbf{x}_1\|_0 + \|\mathbf{x}_2\|_0 \geq \text{spark}_\nu(\mathbf{A}) \quad (5.10)$$

for  $\nu = 2\epsilon/\|\mathbf{x}_1 - \mathbf{x}_2\|_2$ . Because of the monotonicity of  $\text{spark}_\eta(\mathbf{A})$  this implies

$$\eta = \frac{2\epsilon}{D} \leq \nu = \frac{2\epsilon}{\|\mathbf{x}_1 - \mathbf{x}_2\|_2}, \quad (5.11)$$

which leads to the claimed result.  $\square$

Finally, we are now ready to pose a stability result that claims that the solution of  $(P_0^\epsilon)$  cannot be far of from a sufficiently sparse initial vector  $\mathbf{x}_0$ .

**Theorem 5.2. (Stability of  $(P_0^\epsilon)$ ):** *Consider the instance of problem  $(P_0^\epsilon)$  defined by the triplet  $(\mathbf{A}, \mathbf{b}, \epsilon)$ . Suppose that a sparse vector  $\mathbf{x}_0 \in \mathbb{R}^m$  satisfies the sparsity constraint  $\|\mathbf{x}_0\|_0 < (1 + 1/\mu(\mathbf{A}))/2$ , and gives a representation of  $\mathbf{b}$  to within error tolerance  $\epsilon$  (i.e.,  $\|\mathbf{b} - \mathbf{A}\mathbf{x}_0\|_2 \leq \epsilon$ ). Every solution  $\mathbf{x}_0^\epsilon$  of  $(P_0^\epsilon)$  must obey*

$$\|\mathbf{x}_0^\epsilon - \mathbf{x}_0\|_2^2 \leq \frac{4\epsilon^2}{1 - \mu(\mathbf{A})(2\|\mathbf{x}_0\|_0 - 1)}. \quad (5.12)$$

Note that this result parallels Theorem 5 that presents the uniqueness result for the noiseless case, and indeed it reduces to it exactly for the case of  $\epsilon = 0$ .

**Proof:** The solution to  $(P_0^\epsilon)$ , denoted as  $\mathbf{x}_0^\epsilon$ , is at least as sparse as the ideal sparse solution  $\mathbf{x}_0$ , since  $\mathbf{x}_0$  is only one of the many possible feasible solutions, while  $(P_{0,\epsilon})$  seeks the sparsest of them all. Since  $\|\mathbf{x}_0\|_0 < (1 + 1/\mu(\mathbf{A}))/2$ , there exists<sup>3</sup> a value of  $\eta \geq 0$  such that

$$\|\mathbf{x}_0\|_0, \|\mathbf{x}_0^\epsilon\|_0 \leq \frac{1}{2} \text{spark}_\eta(\mathbf{A}), \quad (5.13)$$

and using Lemma 5.2 we replace this inequality with a more strict requirement

$$\|\mathbf{x}_0\|_0 \leq \frac{1}{2} \left( \frac{1 - \eta^2}{\mu(\mathbf{A})} + 1 \right) \leq \frac{1}{2} \text{spark}_\eta(\mathbf{A}), \quad (5.14)$$

which poses a lower bound on  $\eta$  being

$$\eta^2 \leq 1 - \mu(\mathbf{A}) \cdot (2\|\mathbf{x}_0\|_0 - 1). \quad (5.15)$$

Invoking Theorem 5.1, we have two feasible and sparse solutions, each with cardinality below  $\frac{1}{2} \text{spark}_\eta(\mathbf{A})$ , with  $\eta$  as given above. Thus,  $\eta$  should satisfy  $\eta = 2\epsilon/D$  where  $D \geq \|\mathbf{x}_0 - \mathbf{x}_0^\epsilon\|_2$  and combining this with the upper bound on  $\eta$  in Equation (5.15) we obtain

$$\|\mathbf{x}_0 - \mathbf{x}_0^\epsilon\|_2^2 \leq D^2 = \frac{4\epsilon^2}{\eta^2} \leq \frac{4\epsilon^2}{1 - \mu(\mathbf{A}) \cdot (2\|\mathbf{x}_0\|_0 - 1)} \quad (5.16)$$

as claimed.  $\square$

### 5.2.3 The RIP and Its Use for Stability Analysis

Before concluding the discussion on the stability of  $(P_0^\epsilon)$ , we introduce a somewhat different point of view, which leads to an alternative stability claim. In order to present this theoretical analysis, we start by introducing a new measure of quality of a given matrix  $\mathbf{A}$  that replaces the mutual-coherence and the spark. This measure developed by Candes and Tao, termed *Restricted Isometry Property* (RIP), is powerful in grasping a key property of the matrix  $\mathbf{A}$ , that makes consequent analysis easy.

**Definition 5.2.** For a matrix  $\mathbf{A}$  of size  $n \times m$  ( $m > n$ ) with  $\ell_2$ -normalized columns, and for an integer scalar  $s \leq n$ , consider sub-matrices  $\mathbf{A}_s$  containing  $s$  columns from  $\mathbf{A}$ . Define  $\delta_s$  as the smallest quantity such that

$$\forall \mathbf{c} \in \mathbb{R}^s \quad (1 - \delta_s) \|\mathbf{c}\|_2^2 \leq \|\mathbf{A}_s \mathbf{c}\|_2^2 \leq (1 + \delta_s) \|\mathbf{c}\|_2^2, \quad (5.17)$$

<sup>3</sup> Since  $\text{spark}_0(\mathbf{A}) \geq (1 + 1/\mu(\mathbf{A}))/2$ .

hold true for any choice of  $s$  columns. Then  $\mathbf{A}$  is said to have an  $s$ -RIP with a constant  $\delta_s$ .

The key idea in the above definition is the claim that *any* subset of  $s$  columns from  $\mathbf{A}$  behave like an orthogonal transform that loses/gains almost no energy. Clearly, this definition is only informative for  $\delta_s < 1$ .

It is quite easy to see the resemblance between the RIP and the  $\text{spark}_\eta$  properties. Whereas  $\text{spark}_\eta$  is the minimal number of required columns  $s$  to get  $\eta^2$  away from singularity,<sup>4</sup> RIP fixes  $s$  and seeks the minimal value of  $1 - \delta_s$  (5.17), again implying that these  $s$  columns are  $(1 - \delta_s)$ -away from singularity. Note that, in that respect, RIP is richer, as it also proposes a bound from above. As we shall see next, RIP makes the stability analysis far easier. Figure 5.4 describes the connection between  $\text{spark}_\eta$  and RIP.

Given a matrix  $\mathbf{A}$ , it is hard or even impossible to evaluate  $\delta_s$  for  $s \gg 1$ , as this requires a sweep through all the  $\binom{m}{s}$  supports. In that respect, this measure is as complex as the  $\text{spark}$  for assessment. Indeed, just as we have bounded the  $\text{spark}$  using the mutual-coherence, this can be done here, leading to  $\delta_s \leq (s - 1)\mu(\mathbf{A})$ . Note that for  $s = 1$  we have  $\delta_1 = 0$ , since the columns are normalized. The above bound on the RIP constant is easily seen by

$$\begin{aligned} \|\mathbf{A}_s \mathbf{c}\|_2^2 &= \mathbf{c}^T \mathbf{A}_s^T \mathbf{A}_s \mathbf{c} \\ &\leq |\mathbf{c}|^T [\mathbf{I} + \mu(\mathbf{A})(\mathbf{1} - \mathbf{I})] |\mathbf{c}| \\ &\leq (1 - \mu(\mathbf{A}))\|\mathbf{c}\|_2^2 + \mu(\mathbf{A})\|\mathbf{c}\|_1^2 \\ &\leq [1 + (s - 1)\mu(\mathbf{A})]\|\mathbf{c}\|_2^2. \end{aligned} \quad (5.18)$$

In the above we used the structure of the minor from the Gram matrix, having ones on the main-diagonal and off-diagonal entries bounded by  $\mu(\mathbf{A})$ . We also used the norm-equivalence inequality  $\|\mathbf{c}\|_1 \leq \sqrt{s}\|\mathbf{c}\|_2$  for vectors  $\mathbf{c}$  of length  $s$ . The lower-bound can be established similarly as

$$\begin{aligned} \|\mathbf{A}_s \mathbf{c}\|_2^2 &= \mathbf{c}^T \mathbf{A}_s^T \mathbf{A}_s \mathbf{c} \\ &\geq |\mathbf{c}|^T [\mathbf{I} - \mu(\mathbf{A})(\mathbf{1} - \mathbf{I})] |\mathbf{c}| \\ &\geq (1 + \mu(\mathbf{A}))\|\mathbf{c}\|_2^2 - \mu(\mathbf{A})\|\mathbf{c}\|_1^2 \\ &\geq [1 - (s - 1)\mu(\mathbf{A})]\|\mathbf{c}\|_2^2. \end{aligned} \quad (5.19)$$

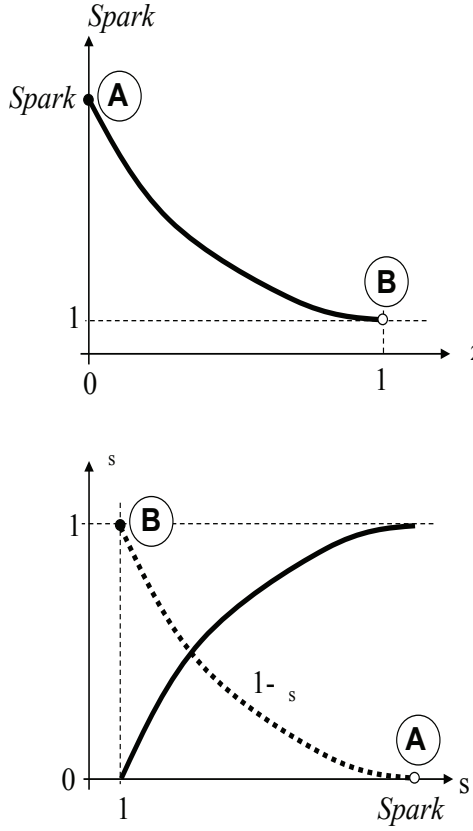
Here we used again the norm-equivalence inequality  $\|\mathbf{c}\|_1 \leq \sqrt{s}\|\mathbf{c}\|_2$ .

An alternative and easier derivation of the above bounds (from above and below) on  $\delta_s$  can be obtained using the Gershgorin disk theorem, that states that the eigenvalues of  $\mathbf{A}_s^T \mathbf{A}_s$  are necessarily bounded by

$$1 - (s - 1)\mu(\mathbf{A}) \leq \lambda(\mathbf{A}_s^T \mathbf{A}_s) \leq 1 + (s - 1)\mu(\mathbf{A}), \quad (5.20)$$

and this directly leads to the above claim, since

<sup>4</sup> In terms of the smallest eigenvalue, and not singular one.



**Fig. 5.4** The top graph presents the behavior of  $\text{spark}_\eta$  as a function of  $\eta^2$ . Point A in this graph corresponds to  $\eta = 0$ , leading to the regular (noseless) *spark*. Point B is the other extreme, where  $\eta = 1$  and thus  $\text{spark}_\eta = 1$ . The bottom graph presents the RIP constant  $\delta_s$  as a function of  $s$ . There is an exact analogy between points A and B in the two graphs, when considering  $1 - \delta_s$  as a function of  $s$ . As such,  $\text{spark}_\eta$  and RIP are related to each other as a function and its inverse.

$$\lambda_{\min}(\mathbf{A}_s^T \mathbf{A}_s) \cdot \|\mathbf{c}\|_2^2 \leq \|\mathbf{A}_s \mathbf{c}\|_2^2 \leq \lambda_{\max}(\mathbf{A}_s^T \mathbf{A}_s) \cdot \|\mathbf{c}\|_2^2. \quad (5.21)$$

Armed with this definition of RIP, we make the following observation: Suppose that a sparse vector  $\mathbf{x}_0$  of cardinality  $s_0$  is multiplied by  $\mathbf{A}$ , and contaminated by an  $\epsilon$ -bounded-norm vector  $\mathbf{e}$  (i.e.,  $\|\mathbf{e}\|_2 \leq \epsilon$ ), generating  $\mathbf{b} = \mathbf{A}\mathbf{x}_0 + \mathbf{e}$ . Thus, clearly,  $\|\mathbf{b} - \mathbf{A}\mathbf{x}_0\|_2 \leq \epsilon$ .

Now, assume that we solve  $(P_0^\epsilon)$  and get a candidate solution, denoted by  $\tilde{\mathbf{x}}$ . Clearly, this solution is sparse, with  $s_0$  non-zeros at the most (since  $\mathbf{x}_0$  is also feasible, and it has  $s_0$  non-zeros). Also, this candidate solution satisfies the inequality  $\|\mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}\|_2 \leq \epsilon$ .

Defining  $\mathbf{d} = \tilde{\mathbf{x}} - \mathbf{x}_0$ , we have that  $\|\mathbf{A}\mathbf{x}_0 - \mathbf{A}\tilde{\mathbf{x}}\|_2 = \|\mathbf{A}\mathbf{d}\|_2 \leq 2\epsilon$ , since both these vectors ( $\mathbf{A}\mathbf{x}_0$  and  $\mathbf{A}\tilde{\mathbf{x}}$ ) are  $\epsilon$  away from the same center vector  $\mathbf{b}$ . We also observe that  $\mathbf{d}$  is a sparse vector with  $2s_0$  non-zeros at the most, since  $\|\mathbf{d}\|_0 = \|\tilde{\mathbf{x}} - \mathbf{x}_0\|_0 \leq \|\mathbf{x}_0\|_0 + \|\tilde{\mathbf{x}}\|_0 \leq 2s_0$ .

Let us assume that the matrix  $\mathbf{A}$  satisfies the RIP property for  $2s_0$  with the corresponding parameter  $\delta_{2s_0} < 1$ . Thus, using this property and exploiting the lower-bound part in Equation (5.17), we get

$$(1 - \delta_{2s_0})\|\mathbf{d}\|_2^2 \leq \|\mathbf{A}\mathbf{d}\|_2^2 \leq 4\epsilon^2, \quad (5.22)$$

and thus, we get a stability claim of the form

$$\|\mathbf{d}\|_2^2 = \|\tilde{\mathbf{x}} - \mathbf{x}_0\|_2^2 \leq \frac{4\epsilon^2}{1 - \delta_{2s_0}}. \quad (5.23)$$

Using our bound for  $\delta_{2s_0}$  from above using the mutual-coherence, this reads

$$\|\tilde{\mathbf{x}} - \mathbf{x}_0\|_2^2 \leq \frac{4\epsilon^2}{1 - \delta_{2s_0}} \leq \frac{4\epsilon^2}{1 - (2s_0 - 1)\mu(\mathbf{A})}. \quad (5.24)$$

The obtained result is the same as the  $(P_0^\epsilon)$  stability claim reported in Theorem 5.2. Note that this bound is true only if the denominator is positive, implying that we require  $s_0 < 0.5(1 + 1/\mu(\mathbf{A}))$ . If  $\epsilon = 0$ , the stability result obtained suggests that  $\tilde{\mathbf{x}} = \mathbf{x}_0$ , which is aligned with the results of Chapter 2

The very same analysis as shown above can be made more general. Assume that an arbitrary approximation algorithm is used for solving  $(P_0^\epsilon)$ , and a feasible candidate solution  $\tilde{\mathbf{x}}$  is proposed with  $\|\tilde{\mathbf{x}}\|_0 = s_1$  non-zeros. Using the same rationale we get that the proposed solution is necessarily close to the ideal one,

$$\|\tilde{\mathbf{x}} - \mathbf{x}_0\|_2^2 \leq \frac{4\epsilon^2}{1 - \delta_{s_0+s_1}} \leq \frac{4\epsilon^2}{1 - (s_0 + s_1 - 1)\mu(\mathbf{A})}, \quad (5.25)$$

due to its sparsity. This claim brings us naturally to the next Section, where we discuss specific approximation algorithms and their tendency to yield such sparse solutions.

## 5.3 Pursuit Algorithms

### 5.3.1 OMP and BP Extensions

Since  $(P_0)$  is impractical to solve in the general case, it seems unlikely that a direct attack on  $(P_0^\epsilon)$  is a sensible goal. The pursuit algorithms discussed above can be adapted to allow error tolerances; how will they perform? Referring to the two op-

tions we had for devising such algorithms – the greedy approach and the relaxation of the  $\ell_0$  functional – variants of these methods may be investigated.

Consider, for example, the greedy algorithm described in Figure 3.1 – the OMP algorithm. By choosing  $\epsilon_0 = \epsilon$  in the stopping rule, the algorithm accumulates non-zero elements in the solution vector until the constraint  $\|\mathbf{b} - \mathbf{Ax}\|_2 \leq \epsilon$  is satisfied. With this minor change, the OMP is very easy to implement and use, and this explains its popularity. The same modification holds true for all the other greedy methods.

Similarly, relaxing  $\ell_0$  to an  $\ell_1$ -norm, we get the following variant of  $(P_1)$ , known in the literature as *basis pursuit denoising* (BPDN) (note that just as in the exact case, we assume that  $\mathbf{A}$  has normalized columns, and thus there is no need for a weighting matrix):

$$(P_1^\epsilon) : \quad \min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{subject to} \quad \|\mathbf{b} - \mathbf{Ax}\|_2 \leq \epsilon, \quad (5.26)$$

This can be written as a standard problem in optimization with a linear penalty and a quadratic and linear inequality constraints. Such problems are very well-studied by specialists in optimization and there are many practical methods for solving them – practically the whole of modern convex optimization theory is applicable, and particularly the recent advances in solving large systems by interior-point and related methods. We shall not review that literature here, and instead discuss two relatively simple and tailored approaches for this problem, suitable for readers without background in convex optimization.

There are various specialized optimization packages that take into account the tendency of the solution of our problem to be sparse ( $\ell_1$ -magic by Candes and Romberg, CVX and L1-LS by Stephen Boyd and students, Sparselab by David Donoho and students, GPSR by Mario Figueiredo, SparCo by Michael Friedlander, PDSCO by Michael Saunders, ShrinkPack by Elad, Zibulevsky, and Shtok, MCAlab by Fadili, Starck, Donoho, and Elad, and more). These pose a good way to get a reliable solution to  $(P_1^\epsilon)$  with little programming effort – one only has to set up  $(P_1^\epsilon)$  as a problem of the type each optimizer can solve. We note, however, that for large-scale applications general-purpose quadratic programming optimizers seem slow and can perhaps be improved by special purpose techniques, as indeed provided by some of the above-mentioned packages.

For an appropriate Lagrange multiplier  $\lambda$ , the solution to (5.26) is precisely the solution to the unconstrained optimization problem

$$(Q_1^\lambda) : \quad \min_{\mathbf{x}} \lambda \|\mathbf{x}\|_1 + \frac{1}{2} \|\mathbf{b} - \mathbf{Ax}\|_2^2, \quad (5.27)$$

where the Lagrange multiplier  $\lambda$  is a function of  $\mathbf{A}$ ,  $\mathbf{b}$ , and  $\epsilon$ .

We mention below the Iterative-Reweighed-Least-Squares (IRLS) (Iterative-Reweighed-Least-Squares) algorithm, which was already mentioned in the context of the exact problem  $(P_0)$ . We shall also briefly present the LARS algorithm due to its feature to provide the complete regularization path (i.e., solutions for all  $\lambda$ ). We stress, however, that there are other options as well for solving the relaxed prob-

**Task:** Find  $\mathbf{x}$  that approximately solves  $(Q_1^\lambda)$ :  $\min_{\mathbf{x}} \lambda \|\mathbf{x}\|_1 + \frac{1}{2} \cdot \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2$ .  
**Initialization:** Initialize  $k = 0$ , and set

- The initial approximation (chosen arbitrarily)  $\mathbf{x}_0 = \mathbf{1}$ .
- The initial weight matrix  $\mathbf{X}_0 = \mathbf{I}$ .

**Main Iteration:** Increment  $k$  by 1, and apply these steps:

- **Regularized Least-Squares:** approximately solve the linear system

$$(2\lambda \mathbf{X}_{k-1}^{-1} + \mathbf{A}^T \mathbf{A}) \mathbf{x} = \mathbf{A}^T \mathbf{b}$$

iteratively (several Conjugate-Gradient iterations may suffice), producing result  $\mathbf{x}_k$ .

- **Weight Update:** Update the diagonal weight matrix  $\mathbf{X}$  using  $\mathbf{x}_k$ :  $X_k(j, j) = |x_k(j)| + \epsilon$ .
- **Stopping Rule:** If  $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2$  is smaller than some predetermined threshold, stop. Otherwise, apply another iteration.

**Output:** The desired result is  $\mathbf{x}_k$ .

**Fig. 5.5** The IRLS strategy for approximately solving  $(Q_1^\lambda)$ .

lem (e.g., Homotopy, gradient projection, and more). Specifically, in Chapter 6 we present a family of specialized algorithms for the minimization of  $(Q_1^\lambda)$ , based on iterative-shrinkage procedures.

### 5.3.2 Iteratively-Reweighed-Least-Squares (IRLS)

A simple strategy to attack  $(Q_1^\lambda)$  is the Iterative-Reweighed-Least-Squares (IRLS) algorithm, similar to the way we took in the exact case in Chapter 3. We note, though, that while the core idea is similar, the derivation here is somewhat different.

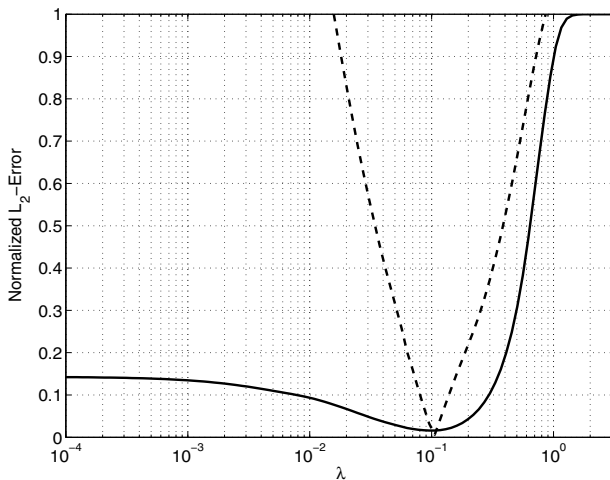
Setting  $\mathbf{X} = \text{diag}(|\mathbf{x}|)$ , we have  $\|\mathbf{x}\|_1 \equiv \mathbf{x}^T \mathbf{X}^{-1} \mathbf{x}$ . Thus we may view the  $\ell_1$ -norm as an (adaptively-weighted) version of the squared  $\ell_2$ -norm. Given a current approximate solution  $\mathbf{x}_{k-1}$ , set  $\mathbf{X}_{k-1} = \text{diag}(|\mathbf{x}_{k-1}|)$  and attempt to solve

$$(M_k) : \quad \min_{\mathbf{x}} \lambda \mathbf{x}^T \mathbf{X}_{k-1}^{-1} \mathbf{x} + \frac{1}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2; \quad (5.28)$$

this is a quadratic optimization problem, solvable using standard linear algebra. Obtain an (approximate) solution  $\mathbf{x}_k$  (say); a diagonal matrix  $\mathbf{X}_k$  is constructed with the entries of  $\mathbf{x}_k$  on the diagonal, and a new iteration can begin. This algorithm is formally described in [Figure 5.5](#).

Let us demonstrate this algorithm on a specific case study, and through this also gain some insight into the problems  $(Q_1^\lambda)$  and  $(P_1^\epsilon)$ . We generate a random matrix  $\mathbf{A}$





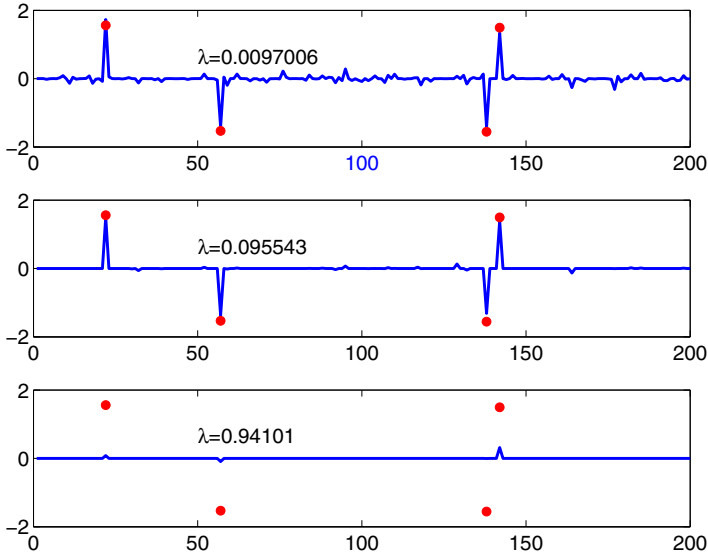
**Fig. 5.6** Normalized  $\ell_2$ -error of the solution ( $\|\hat{\mathbf{x}}_\lambda - \mathbf{x}_0\|^2 / \|\mathbf{x}_0\|^2$ ) as a function of  $\lambda$ . Each point in this curve is the outcome of 15 iterations of the IRLS algorithm. The dashed curve presents the value of  $|\log(\|\mathbf{A}\hat{\mathbf{x}}_\lambda - \mathbf{b}\|^2 / (n\sigma^2))|$ , getting a value close to zero implies that the residual is equal to the noise power.

of size  $100 \times 200$ , create a random sparse vector  $\mathbf{x}_0$  of length 200 with cardinality  $\|\mathbf{x}_0\|_0 = 4$ , where its non-zero entries are drawn uniformly from the range  $[-2, -1] \cup [1, 2]$ . We compute  $\mathbf{b} = \mathbf{A}\mathbf{x}_0 + \mathbf{e}$ , where  $\mathbf{e}$  is an additive random Gaussian iid noise vector with standard-deviation  $\sigma = 0.1$ . Our goal is to recover  $\mathbf{x}_0$ , and for that we shall solve  $(Q_1^t)$  with the IRLS algorithm.

The first problem we encounter is the choice of  $\lambda$ . Whereas the problem  $(P_1^\epsilon)$  can be clearly posed with  $\epsilon \approx \sqrt{n}\sigma$  to represent the deviation we expect between  $\mathbf{A}\mathbf{x}$  and  $\mathbf{b}$ , turning to the format of  $(Q_1^t)$  loses this simple intuition. The thumb-rule we shall use is to choose  $\lambda$  as approximately the ratio between  $\sigma$  and the standard-deviation of the expected non-zeros in the solution. More on this choice and its justification are given in Chapter 11. As the non-zero's STD is roughly 2, we choose  $\lambda$  around the value  $\sigma/2 = 0.05$ .

Figure 5.6 presents the quality of the solution, measured by  $\|\hat{\mathbf{x}}_\lambda - \mathbf{x}_0\|^2 / \|\mathbf{x}_0\|^2$ , as a function of  $\lambda$ . Each solution is obtained using 15 iterations of the IRLS algorithm. The same graph shows also the term  $|\log(\|\mathbf{A}\hat{\mathbf{x}}_\lambda - \mathbf{b}\|^2 / (n\sigma^2))|$ , which shows for which  $\lambda$  the residual  $\|\mathbf{A}\hat{\mathbf{x}}_\lambda - \mathbf{b}\|^2$  becomes equal to the noise power  $n\sigma^2$ . This is another thumb-rule that can be used for determining  $\lambda$ , and as we see, there is a good agreement between this choice of  $\lambda$  and the optimal one.

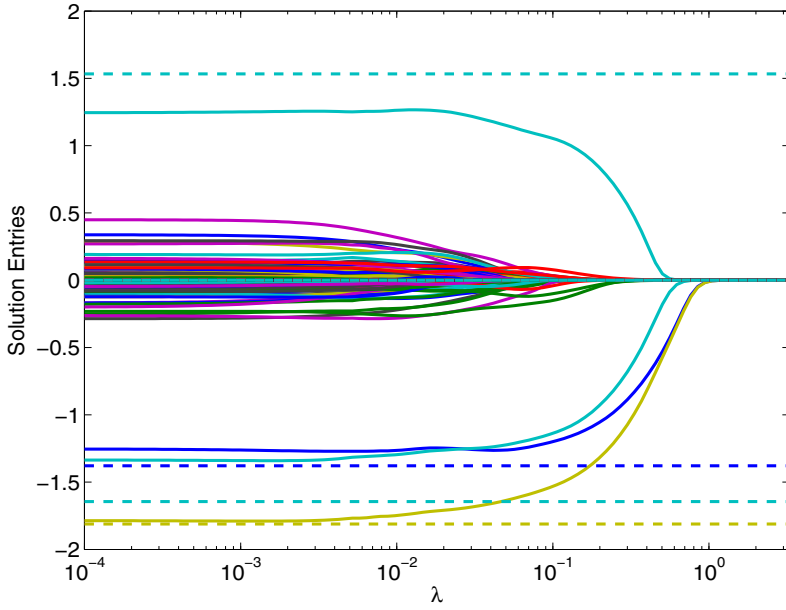
Figure 5.7 presents three possible solutions, for the best found  $\lambda$ , and two values – one above and one below it. These solutions are compared to the ideal  $\mathbf{x}_0$ , and as can be seen, the optimal  $\lambda$  value indeed finds a good solution with a good detection of the non-zeros in the ideal vector  $\mathbf{x}_0$ . A smaller  $\lambda$  leads to a more dense solution, while a larger one further sparsifies it.



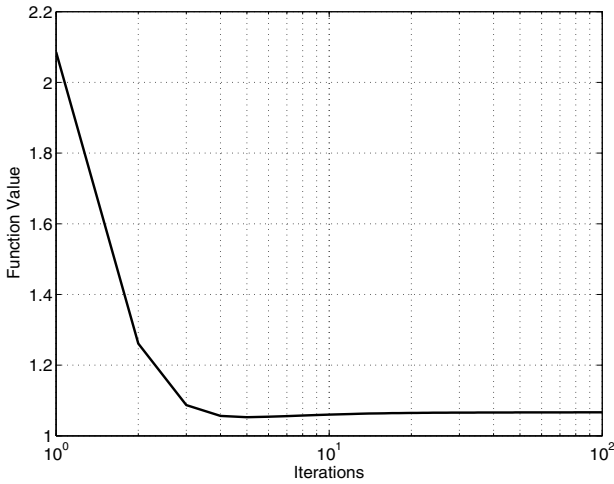
**Fig. 5.7** Solutions obtained by the IRLS algorithm (15 iterations) for three values of  $\lambda$  – the best one (in the middle), a smaller one (top), and a larger one (bottom). The true non-zero values of  $\mathbf{x}_0$  are added as dots for reference.

There is an interesting way to present all the solutions for all  $\lambda$  values together in one graph. Building a 2D array that contains all the solutions as its columns, simply plot all its rows as graphs. This approach is commonly used in the context of an algorithm called Least-Absolute-Shrinkage-and-Selection-Operator (LASSO) and another termed Least-Angle-Regression (LARS) (see next subsection). These methods expose the path each entry takes from zero (for large enough value  $\lambda$ ) to a possibly non-zero value. Such a graph is presented in [Figure 5.8](#). we see that values of  $\lambda$  above 1 lead to a zero solution. As  $\lambda$  becomes smaller, more and more entries are “awakened” (become non-zeros). The dotted curves represents the entries of  $\mathbf{x}_0$ , and thus stand as the ideal result we should strive to.

All the above discussion is general and could refer to any alternative solver of  $(Q_1^\lambda)$ . We now turn to show the specific behavior of the IRLS. We already see in some of the above figures that the IRLS does not fully sparsifies the proposed solution, even for the optimal  $\lambda$ . [Figure 5.9](#) presents the  $(Q_1^\lambda)$  function value for the computed solution as a function of the iteration. We see that the IRLS is very effective in the first several iterations, but then slows down dramatically. Furthermore, we see that it is even not guaranteed to give a descent in every iteration. These observations mean that this method can be effective mostly if we are interested in a crude solution. By a relaxation of the IRLS iteration, one can give better convergence behavior and guarantee a descent in each iteration, but we shall not dwell on this here.



**Fig. 5.8** All the IRLS solutions for all  $\lambda$  values. Each curve represents one entry in the solution and the path it takes as a function of  $\lambda$ . The dotted curves are those of  $\mathbf{x}_0$ .



**Fig. 5.9** The function value  $(\lambda \|\mathbf{x}\|_1 + \frac{1}{2} \cdot \|\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}\|_2^2)$  as a function of the iterations of the IRLS, for the optimal value of  $\lambda$ .

### 5.3.3 The LARS Algorithm

In the statistical machine learning community, the problem we have defined as  $(Q_1^\lambda)$  is known as LASSO (Least Absolute Shrinkage and Selection Operator), used for regression over a wide set of features. LASSO was conceived by Friedman, Hastie, and Tibshirani, three researchers from the statistics department at Stanford University. This took place around the very same time the Basis-Pursuit was invented for signal processing purposes by Chen, Donoho, and Saunders – from the same department (actually, the same corridor!!).

In the LASSO formulation, the columns of  $\mathbf{A}$  represent each a feature with varying values, and the vector  $\mathbf{b}$  is the output of a complex system. We desire to find a simple linear combination of *few* features that could explain the output  $\mathbf{b}$ . Thus, solving  $(Q_1^\lambda)$  not only provides a way to get such regression, but it also selects a small subset of features, known as model selection.

While LASSO and BPDN target different applications, their mathematical formulation is exactly the same. In the statistical machine learning community that studies LASSO, the goal is a numerical method that could solve  $(Q_1^\lambda)$  for all possible values of  $\lambda$  at once. We refer to this type of solutions as the complete path of the regularization. The motivation is natural – this can help us in migrating from the solution of  $(Q_1^\lambda)$  to  $(P^\epsilon)$  easily, and better tune  $\epsilon$  or  $\lambda$  for our needs.

Surprisingly, such path of solutions can be obtained with almost the same computational effort as solving for one value of  $\lambda$  alone. Indeed, in order to see why this is at all possible, think of the OMP as a technique that also sweeps through  $\lambda$ , as the residual energy is decreased by adding one atom at a time. LARS (Least Angle Regression Stagewise), proposed by the LASSO team and Efron, is very similar to OMP, but with a guarantee that the solution path is the global optimizer of  $(Q_1^\lambda)$ . We now turn to describe the core of this algorithm, adopting an interesting and appealing interpretation of this algorithm proposed by Julien Mairal.

We start by observing that for a solution to be the optimal minimizer of  $(Q_1^\lambda)$ , it should lead to a sub-gradient set that contains the zero. For the convex function

$$f(\mathbf{x}) = \lambda \|\mathbf{x}\|_1 + \frac{1}{2} \|\mathbf{b} - \mathbf{Ax}\|_2^2, \quad (5.29)$$

the sub-gradient set is given as the set of all vectors

$$\partial f(\mathbf{x}) = \left\{ \mathbf{A}^T (\mathbf{Ax} - \mathbf{b}) + \lambda \mathbf{z} \right\} \quad \forall \mathbf{z} = \begin{cases} +1 & x[i] > 0 \\ [-1, +1] & x[i] = 0 \\ -1 & x[i] < 0 \end{cases}. \quad (5.30)$$

When searching the minimizer of the function  $f(\mathbf{x})$ , we should seek both  $\mathbf{x}$  and  $\mathbf{z}$  that suites it, such that  $\mathbf{0} \in \partial f(\mathbf{x})$ .

The sub-gradient set at location  $\mathbf{x}_0$  defines all the possible directions  $\mathbf{v}$  that satisfy  $f(\mathbf{x}) - f(\mathbf{x}_0) \geq \mathbf{v}^T (\mathbf{x} - \mathbf{x}_0)$  for a sufficiently small neighborhood,  $\|\mathbf{x} - \mathbf{x}_0\|_2 \leq \delta$ . This should remind the reader of tangent planes, passing through  $\mathbf{x}_0$  and bounding the convex function  $f(\mathbf{x})$  from below at  $\mathbf{x}_0$ . For differentiable functions this aligns

with the single-vector regular gradient. The fact that the sub-gradient may contain more than just one vector is a direct consequence of the non-differentiability of  $f(\mathbf{x})$  around zero.

We now turn to describe how this sub-gradient requirement is used for the creation of LARS, where we focus on the main theme of this method, disregarding small (but important) details.

**Step 1:** Starting with  $\lambda \rightarrow \infty$ , we obviously have  $\mathbf{x}_\lambda = \mathbf{0}$  as our optimal solution. Observe that as  $\lambda$  is decreased towards zero, for all values  $\lambda \geq \|\mathbf{A}^T \mathbf{b}\|_\infty$ , the zero solution remains optimal, because using Equation (5.30), the requirement  $\mathbf{0} \in \partial f(\mathbf{x})$  and the assumption  $\mathbf{x}_\lambda = \mathbf{0}$  imply

$$\mathbf{0} = -\mathbf{A}^T \mathbf{b} + \lambda \mathbf{z}_\lambda \quad (5.31)$$

and  $\mathbf{z}_\lambda = \mathbf{A}^T \mathbf{b} / \lambda$  satisfies the condition posed in Equation (5.30), i.e. all the entries in  $\mathbf{z}$  are in the range  $[-1, 1]$ .

**Step 2:** As  $\lambda$  is decreased and gets to  $\|\mathbf{A}^T \mathbf{b}\|_\infty$ , assume that it is the  $i$ -th entry in this vector that gets the maximal value. Keeping  $x_\lambda[i] = 0$  for slightly smaller  $\lambda$  clearly violates (5.30). Thus, we change this entry and accordingly set  $z_\lambda[i] = \text{sign}(x_\lambda[i])$ . This single non-zero value  $x_\lambda[i]$  is obtained by solving

$$\begin{aligned} 0 &= \mathbf{a}_i^T (\mathbf{a} \mathbf{x} - \mathbf{b}) + \lambda z_\lambda[i] \\ \Rightarrow x_\lambda[i] &= \frac{\mathbf{a}_i^T \mathbf{b} - \lambda z_\lambda[i]}{\mathbf{a}_i^T \mathbf{a}_i} = \frac{\mathbf{a}_i^T \mathbf{b} - \lambda \text{sign}(x_\lambda[i])}{\mathbf{a}_i^T \mathbf{a}_i}. \end{aligned} \quad (5.32)$$

It is easily seen that this value has the same sign as the inner product  $\mathbf{a}_i^T \mathbf{b}$ , as required.

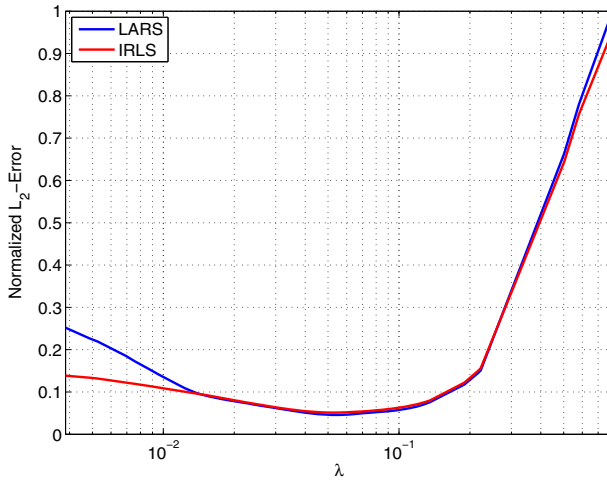
The proposed solution is correct for  $\lambda = \|\mathbf{A}^T \mathbf{b}\|_\infty$  and nearby values below, and the condition on  $\mathbf{z}$  translates to the assignment  $\mathbf{z}_\lambda = \mathbf{A}^T (\mathbf{b} - \mathbf{A} \mathbf{x}_\lambda) / \lambda$ . The  $i$ -th row of this requirement is satisfied trivially with equality due to (5.32), and the rest are met by setting  $\mathbf{z}_\lambda$  based on the above formula.

**Step 3:** We proceed by decreasing the value of  $\lambda$ , changing linearly the solution  $\mathbf{x}_\lambda$  based on Equation (5.32), and updating  $\mathbf{z}_\lambda$  based on the above assignment. Assume that for the current value of  $\lambda$  we hold the support of  $\mathbf{x}$ , denoted by  $S$ . We denote by  $\mathbf{x}_\lambda^s$  the non-zero part of  $\mathbf{x}_\lambda$ , and  $\mathbf{A}_s$  as the sub-matrix of  $\mathbf{A}$  that contains the chosen columns. The current solution is thus given by

$$\mathbf{x}_\lambda^s = (\mathbf{A}_s^T \mathbf{A}_s)^{-1} (\mathbf{A}_s^T \mathbf{b} - \lambda \mathbf{z}_\lambda^s), \quad (5.33)$$

where  $\mathbf{z}_\lambda^s = \text{sign}(\mathbf{x}_\lambda^s)$ , is the sign-vector of the non-zero portion of the vector  $\mathbf{x}_\lambda$ . This solution changes linearly with  $\lambda$ . As this process of decreasing  $\lambda$  continues, two situations can be encountered:

1. The  $i$ -th entry in  $\mathbf{z}_\lambda$  outside the support of  $\mathbf{x}_\lambda$  becomes  $\pm 1$  and is about to exit the permitted range  $[-1, +1]$ . In this case we add this entry to the support, and update the solution to include it, using the above formula.

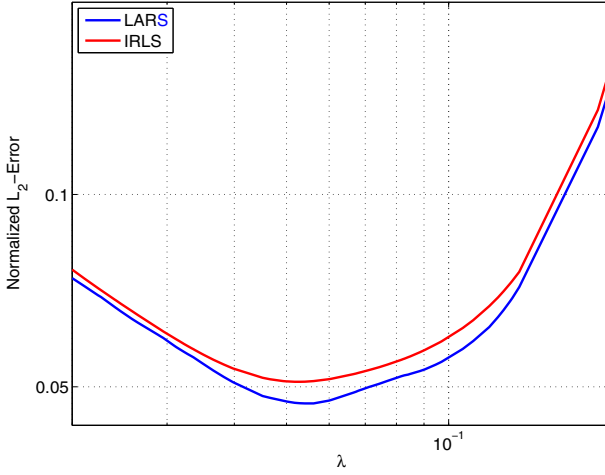


**Fig. 5.10** Normalized  $\ell_2$ -error of the solution ( $\|\hat{\mathbf{x}}_\lambda - \mathbf{x}_0\|^2 / \|\mathbf{x}_0\|^2$ ) as a function of  $\lambda$ . Each point in these curves is the outcome of LARS (solid) or 15 iterations of the IRLS algorithm (x marks).

2. One of the non-zero entries in  $\mathbf{x}_\lambda^s$  becomes zero. In this case it should be removed from the support and the solution updated, again using the above formula.

In practice, the value of  $\lambda$  is decreased from one check-point to the next, testing for these cases, and updating the support  $S$ , the solution  $\mathbf{x}_\lambda$ , and the vector  $\mathbf{z}_\lambda$ . Note that throughout the algorithm we assume that there is a unique optimal solution to  $(Q_1^\lambda)$ , and build it explicitly. Also, note that we assume that at every stage only one entry in the solution joins/leaves the support. In cases where these assumptions are not met, the algorithm requires delicate modifications.

Before concluding this discussion, we turn to demonstrate LARS. We perform the same test we have applied on the IRLS (and in fact, on the same data). We use a Matlab software written by Karl Sjöstrand from the Technical University of Denmark, which provides LARS' full regularization path. As the results are quite close to those of the IRLS, we present only the following two figures that correspond to Figure 5.6. Figure 5.10 shows both the LARS and the IRLS (15 iterations) curves, and as can be seen, they are quite similar, and especially so around the optimal value of  $\lambda$ . Figure 5.11 zooms-in on the area around the minimum, showing that a difference does exist in favor of LARS. Applying more iterations by IRLS is not likely to close this gap, due to the much-slower convergence we have observed. LARS is more accurate, and also cheaper in terms of computations, making it an ideal solver for low-dimensional problems. As the number of steps required by LARS roughly equals the dimensions of  $\mathbf{x}$ , using this technique in high dimensions is prohibitive.

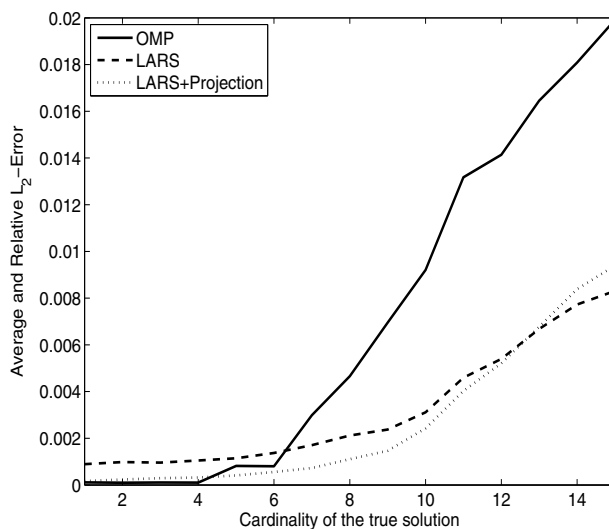


**Fig. 5.11** A zoomed-in version of Figure 5.10 that exposes the small differences between the two algorithms.

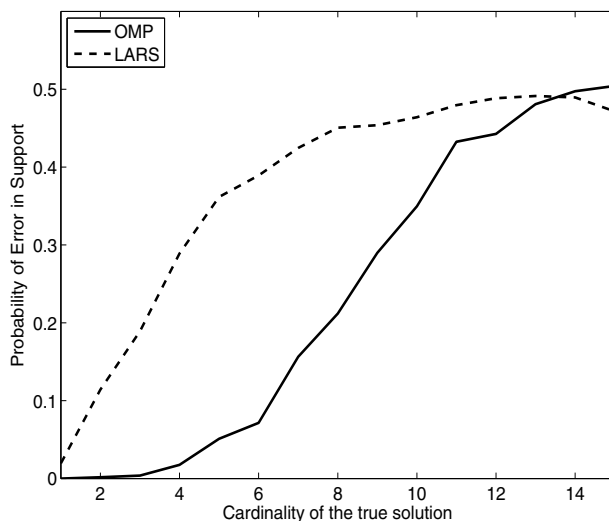
### 5.3.4 Quality of Approximations Obtained

LARS and IRLS, as solvers of  $(Q_1^\lambda)$ , are only approximation algorithms to our true desire – the solution of  $(P_0^\epsilon)$ . In this context, OMP and other greedy methods also target this objective, which gives us common grounds to compare these very different techniques. How good are these approximations? A partial answer appears in Figure 5.10, showing that  $\hat{\mathbf{x}}_\lambda$  obtained by LARS/IRLS tends to be close to  $\mathbf{x}_0$  for a proper choice of  $\lambda$ . Just as in Chapter 3, we can also ask how good are those algorithms in recovering the true support and entry-values of  $\mathbf{x}_0$ , but this time noisy measurements are involved. We address these questions here, by an experiment similar to the one reported in Sections 3.1.7 and 3.2.3, in which we compare LARS and OMP.

We create a random matrix  $\mathbf{A}$  of size  $30 \times 50$  with entries drawn from the normal distribution, and then we normalize its columns. We generate sparse vectors  $\mathbf{x}_0$  with iid random supports of cardinalities in the range  $[1, 15]$ , and non-zero entries drawn as random uniform variables in the range  $[-2, -1] \cup [1, 2]$ . Once  $\mathbf{x}_0$  is generated, we compute  $\mathbf{b} = \mathbf{A}\mathbf{x}_0 + \mathbf{e}$ , where  $\mathbf{e}$  is a zero mean iid random Gaussian noise. We set its standard-deviation to  $\sigma = 0.1$ . We then apply the LARS and OMP algorithms to seek for  $\mathbf{x}$ . Both methods provide a set of solutions, both adding one atom at a time. From those we choose the sparsest solution that satisfies  $\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}\|_2^2 \leq 2n\sigma^2$  (chosen empirically to give good results). We perform 200 such tests per each cardinality, and present average results on  $\ell_2$  recovery error (between  $\mathbf{x}_0$  and its estimate  $\hat{\mathbf{x}}$ ), and on the support distance (see Section 3.1.7 for more details on these performance measures). These results are given in Figures 5.12 and 5.13.

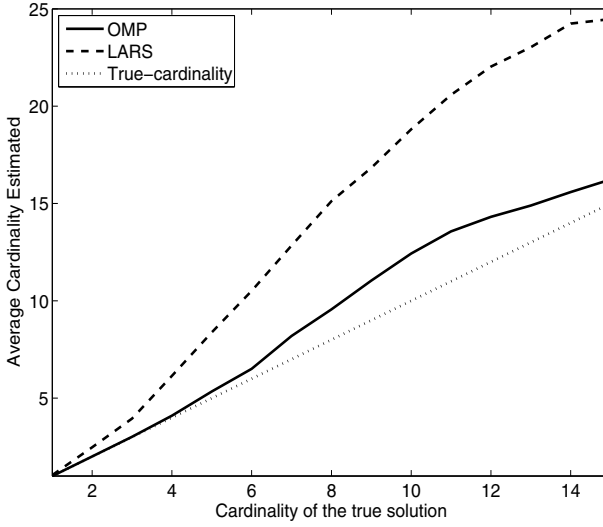


**Fig. 5.12** OMP and LARS performance as approximate solvers of  $(P_0^\epsilon)$ . The performance shown here refers to a relative  $\ell_2$  recovery error. The projected-LARS takes the proposed LARS solution, adopts only its support, and find the non-zero coefficients by Least-Squares fitting.



**Fig. 5.13** OMP and LARS performance as approximate solvers of  $(P_0^\epsilon)$ . The performance shown here refers to the success rate in detecting the true support.





**Fig. 5.14** OMP and LARS performance as approximate solvers of  $(P_0^\epsilon)$ . The performance shown here refers to the average cardinality of the obtained solutions, and our goal is getting as sparse as possible solution.

The results we see in these two figures indicate that OMP is better in recovering the support, while LARS is better in getting smaller  $\ell_2$  error. In terms of  $\ell_2$  error, it seems that both algorithms perform well, as the errors obtained are quite small. Since LARS minimizes  $(Q_1^d)$ , it tends to under-estimate the values of the non-zeros, as those are penalized by the  $\ell_1$  term. Thus, we can adopt the support given by the LARS solution, and re-evaluate the non-zero entries in  $\hat{\mathbf{x}}$  by solving a simple Least-Squares problem of the form  $\min_{\mathbf{x}} \|\mathbf{A}_S \mathbf{x} - \mathbf{b}\|_2^2$ . As can be seen in Figure 5.12, this helps in slightly reducing the error.

Figure 5.14 shows another aspect of the OMP and LARS performance. For the error threshold chosen, both methods ultimately aim to give the sparsest possible solution, and thus it is natural to ask, how sparse is the proposed solution? In Figure 5.14 we show the average cardinality of the solutions obtained by the OMP and the LARS in the above-described experiment. It is evident that OMP performs very well, getting nearly the original cardinality, whereas LARS is far behind, tending to give a denser solution. This aligns well with the results shown in Figure 5.13 that indicate that LARS gives a poor detection of the support – we now know that this is due to its tendency to give too many non-zeros.

## 5.4 The Unitary Case

If  $\mathbf{A}$  is unitary, the minimization of  $(P_1^\epsilon)$ ,  $(Q_1^\lambda)$ , and even  $(P_0^\epsilon)$  can all be manipulated in a sequence of simple steps and turned into a set of  $n$  independent and identical one-dimensional optimization tasks that are easily handled. Let us start with a treatment of  $(P_0^\epsilon)$ , as defined in Equation (5.1),

$$(P_0^\epsilon) : \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{b} - \mathbf{Ax}\|_2 \leq \epsilon.$$

Using the identity  $\mathbf{AA}^T = \mathbf{I}$ , we get an equivalent problem of the form

$$(P_0^\epsilon) : \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{A}^T \mathbf{b} - \mathbf{x}\|_2 \leq \epsilon. \quad (5.34)$$

Here we have exploited the fact that  $\ell_2$ -norm is unitary-invariant, which implies the equality.

$$\|\mathbf{b} - \mathbf{Ax}\|_2 = \|\mathbf{A}^T (\mathbf{b} - \mathbf{Ax})\|_2.$$

Denoting  $\mathbf{x}_* = \mathbf{A}^T \mathbf{b}$ , we obtain a simplified problem of the form

$$(P_0^\epsilon) : \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad (5.35)$$

$$\text{subject to} \quad \|\mathbf{x}_* - \mathbf{x}\|_2^2 = \sum_{i=1}^n (x_*[i] - x[i])^2 \leq \epsilon^2.$$

this problem defines the optimal solution as the sparsest vector that is  $\epsilon$ -close to  $\mathbf{x}_*$ . Thus, the optimal solution is simply obtained by sorting the entries of the vector  $\mathbf{x}_* = \mathbf{A}^T \mathbf{b}$  in a descending order of their absolute values, and choosing the leading terms till the constraint  $\|\mathbf{x}_* - \mathbf{x}\|_2 \leq \epsilon$  is satisfied. This implies that there exists a threshold value  $T(\epsilon)$  such that

$$\mathbf{x}_0^\epsilon = \begin{cases} x_0^\epsilon[i] = x_*[i] & |x_*[i]| \geq T(\epsilon) \\ x_0^\epsilon[i] = 0 & |x_*[i]| < T(\epsilon) \end{cases}, \quad (5.36)$$

where  $T(\epsilon)$  is chosen as the largest possible value that satisfies  $\|\mathbf{x}_* - \mathbf{x}_0^\epsilon\|_2 \leq \epsilon$ . The operation described in Equation (5.36) is known as *hard-thresholding*, used often in signal processing for sparsifying vectors. Note that the solution we have obtained is the *globally optimal* one, even though the initial problem is non-convex and highly combinatorial in nature.

Turning to handle  $(P_1^\epsilon)$  and  $(Q_1^\lambda)$ , the treatment is similar, although a bit different. Following the same steps, we get the equivalent problems

$$(P_1^\epsilon) : \quad \min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{subject to} \quad \|\mathbf{x}_* - \mathbf{x}\|_2^2 \leq \epsilon^2 \quad (5.37)$$

$$(Q_1^\lambda) : \quad \min_{\mathbf{x}} \lambda \|\mathbf{x}\|_1 + \frac{1}{2} \|\mathbf{x}_* - \mathbf{x}\|_2^2. \quad (5.38)$$

As already mentioned above, these two problems can be made equivalent by a proper choice of  $\lambda$ , as a function of  $\epsilon$  and  $\mathbf{x}_*$ . Thus, we proceed with the formulation of  $(Q_1^\lambda)$ , and observe that its penalty function is separated into  $n$  disjoint, independent, and equally formulated 1D problems of the form ( $x_*$  is now a scalar),

$$x_{opt} = \arg \min_x \lambda|x| + \frac{1}{2}(x - x_*)^2. \quad (5.39)$$

Since the absolute value function is not differentiable at the origin, we replace the classical approach of zeroing the derivative with the requirement that the zero is included in the sub-gradient set (see the discussion about the LARS algorithm). This poses a requirement on  $x_{opt}$  of the form

$$0 \in x - x_* + \lambda \begin{cases} +1 & x > 0 \\ [-1, +1] & x = 0 \\ -1 & x < 0 \end{cases}. \quad (5.40)$$

This states that the solution should be in the set that nulls the sub-gradient of the function. Denoting

$$z = \begin{cases} +1 & x > 0 \\ [-1, +1] & x = 0 \\ -1 & x < 0 \end{cases}, \quad (5.41)$$

we require  $x - x_* + \lambda z = 0$ . Assuming that  $x_* \geq \lambda$ , we get  $x_{opt} = x_* - \lambda$  as the optimal solution by assigning  $z = 1$ . Similarly, for  $x_* \leq -\lambda$ , the solution is  $x_{opt} = x_* + \lambda$  for the assignment  $z = -1$ . For  $-\lambda < x_* < \lambda$ , the optimal solution is necessarily  $x_{opt} = 0$ , with the choice  $z = x_*/\lambda$ , which is indeed in the range  $[-1, +1]$ . Thus, returning to the problems  $(P_1^\epsilon)$  and  $(Q_1^\lambda)$  posed in Equation (5.37), we propose the solution

$$\begin{aligned} \mathbf{x}_1^\epsilon &= \begin{cases} x_1^\epsilon[i] = x_*[i] - \lambda & x_*[i] \geq \lambda \\ x_1^\epsilon[i] = 0 & |x_*[i]| \leq \lambda \\ x_1^\epsilon[i] = x_*[i] + \lambda & x_*[i] \leq -\lambda \end{cases} \\ &= \text{sign}(x_*[i]) \cdot [|x_*[i]| - \lambda]_+, \end{aligned} \quad (5.42)$$

where  $\lambda$  should be chosen as the largest possible value that satisfies  $\|\mathbf{x}_* - \mathbf{x}_0^\epsilon\|_2 \leq \epsilon$ , if we are solving  $(P_1^\epsilon)$ . This operation is known as *soft-thresholding*.

We see that the three problems  $(P_0^\epsilon)$ ,  $(P_1^\epsilon)$  and  $(Q_1^\lambda)$  admit simple and closed-form optimal solutions. Furthermore, as already said, even though  $(P_0^\epsilon)$  is non-convex, we guarantee that the solution we provide is globally optimal.

Interestingly, if the simple thresholding algorithm (see Chapter 3) is applied for approximating the solution of  $(P_0^\epsilon)$  in this case, it leads to the optimal result. This is because it computes the vector  $\mathbf{A}^T \mathbf{b}$  and chooses its leading entries (in absolute value), just as the optimal formula suggests. Furthermore, although less obvious to see, MP and OMP also lead to the optimal solution in this case. Their first iteration chooses the largest entry from the vector  $|\mathbf{A}^T \mathbf{b}|$ , and this aligns well with the optimal

solution formula. Subsequent iterations require the evaluation of the residual, but due to the orthogonality of  $\mathbf{A}$ , this is equivalent to the removal of the chosen column from  $\mathbf{A}$ . Thus, all iterations remove leading entries from  $|\mathbf{A}^T \mathbf{b}|$ , and the stopping rule guarantees that the smallest possible non-zeros are used.

## 5.5 Performance of Pursuit Algorithms

Can pursuit methods approximately solve  $(P_0^\epsilon)$ ? A partial and empirical answer has been given above. We bring in this section several theoretical results that address this question. The first part is taken from the work by Donoho, Elad, and Temlyakov, with some modifications, corresponding to the BPDN stability. Rather than presenting a parallel result for OMP, we develop such a property for the far simpler thresholding algorithm.

### 5.5.1 BPDN Stability Guarantee

**Theorem 5.3. (Stability of BPDN):** *Consider the instance of problem  $(P_1^\epsilon)$  defined by the triplet  $(\mathbf{A}, \mathbf{b}, \epsilon)$ . Suppose that a vector  $\mathbf{x}_0 \in \mathbb{R}^m$  is a feasible solution to  $(P_1^\epsilon)$  satisfying the sparsity constraint  $\|\mathbf{x}_0\|_0 < (1 + 1/\mu(\mathbf{A}))/4$ . The solution  $\mathbf{x}_1^\epsilon$  of  $(P_1^\epsilon)$  must obey*

$$\|\mathbf{x}_1^\epsilon - \mathbf{x}_0\|_2^2 \leq \frac{4\epsilon^2}{1 - \mu(\mathbf{A})(4\|\mathbf{x}_0\|_0 - 1)}. \quad (5.43)$$

**Proof:** In this proof we follow closely with the steps of proof of Theorem 4.5, with necessary modifications.

Both  $\mathbf{x}_0$  and  $\mathbf{x}_1^\epsilon$  are feasible and thus satisfy  $\|\mathbf{b} - \mathbf{A}\mathbf{x}_0\|_2 \leq \epsilon$  and  $\|\mathbf{b} - \mathbf{A}\mathbf{x}_1^\epsilon\|_2 \leq \epsilon$ . Considering the vector  $\mathbf{b}$  as a center of a sphere of radius  $\epsilon$ , we know that both  $\mathbf{A}\mathbf{x}_0$  and  $\mathbf{A}\mathbf{x}_1^\epsilon$  are on the surface of this sphere, and thus the distance between them is at most  $2\epsilon$ . Denoting  $\mathbf{d} = \mathbf{x}_1^\epsilon - \mathbf{x}_0$ , we therefore have

$$\|\mathbf{A}\mathbf{x}_1^\epsilon - \mathbf{A}\mathbf{x}_0\|_2 = \|\mathbf{A}\mathbf{d}\|_2 \leq 2\epsilon. \quad (5.44)$$

We write this requirement differently using the Gram matrix  $\mathbf{A}^T \mathbf{A} = \mathbf{G}$ , exploiting the fact that its main-diagonal contains ones, and its off-diagonal entries are bounded by the mutual-coherence  $\mu(\mathbf{A})$ . This leads to

$$\begin{aligned} 4\epsilon^2 &\geq \|\mathbf{A}\mathbf{d}\|_2^2 = \mathbf{d}^T \mathbf{G} \mathbf{d} \\ &= \|\mathbf{d}\|_2^2 + \mathbf{d}^T (\mathbf{G} - \mathbf{I}) \mathbf{d} \\ &\geq \|\mathbf{d}\|_2^2 - |\mathbf{d}|^T |\mathbf{G} - \mathbf{I}| |\mathbf{d}| \\ &\geq \|\mathbf{d}\|_2^2 - \mu(\mathbf{A}) \cdot |\mathbf{d}|^T |\mathbf{1} - \mathbf{I}| |\mathbf{d}| \end{aligned} \quad (5.45)$$

$$\begin{aligned}
&= \|\mathbf{d}\|_2^2 - \mu(\mathbf{A}) \cdot (|\mathbf{d}|^T \mathbf{1} |\mathbf{d}| - |\mathbf{d}|^T |\mathbf{d}|) \\
&\geq (1 + \mu(\mathbf{A})) \|\mathbf{d}\|_2^2 - \mu(\mathbf{A}) \|\mathbf{d}\|_1^2.
\end{aligned}$$

In the above we used the property  $|\mathbf{d}|^T \mathbf{1} |\mathbf{d}| = |\mathbf{d}|^T \mathbf{1} \mathbf{1}^T |\mathbf{d}| = \|\mathbf{d}\|_1^2$ .

We know that  $\|\mathbf{x}_1^\epsilon\|_1 = \|\mathbf{d} + \mathbf{x}_0\|_1 \leq \|\mathbf{x}_0\|_1$ , reflecting the fact that, as the solution to  $(P_1^\epsilon)$ ,  $\mathbf{x}_1^\epsilon$  has the shortest possible  $\ell_1$ -norm. Using this and the inequality developed in the proof of Theorem 4.5, we obtain

$$\|\mathbf{d}\|_1 - 2\mathbf{1}_s^T |\mathbf{d}| \leq \|\mathbf{d} + \mathbf{x}_0\|_1 - \|\mathbf{x}_0\|_1 \leq 0, \quad (5.46)$$

where  $\mathbf{1}_s$  is a vector of length  $m$  with ones on the support  $\mathcal{S}$  of the vector  $\mathbf{x}_0$ . Thus,

$$\|\mathbf{d}\|_1 \leq 2\mathbf{1}_s^T |\mathbf{d}| + \|\mathbf{d} + \mathbf{x}_0\|_1 - \|\mathbf{x}_0\|_1 \leq 2\mathbf{1}_s^T |\mathbf{d}|. \quad (5.47)$$

Using the  $\ell_1$ - $\ell_2$ -norms relation,  $\forall \mathbf{v} \in \mathbb{R}^n$ ,  $\|\mathbf{v}\|_1 \leq \sqrt{n} \|\mathbf{v}\|_2$ , we obtain

$$\|\mathbf{d}\|_1 \leq 2\mathbf{1}_s^T |\mathbf{d}| \leq 2 \sqrt{|\mathcal{S}| \sum_{i \in \mathcal{S}} |d_i|^2} \leq 2 \sqrt{|\mathcal{S}|} \|\mathbf{d}\|_2. \quad (5.48)$$

Plugged back into the inequality in Equation (5.45), we obtain

$$4\epsilon^2 \geq (1 + \mu(\mathbf{A})) \|\mathbf{d}\|_2^2 - \mu(\mathbf{A}) \|\mathbf{d}\|_1^2 \geq (1 + \mu(\mathbf{A}) - 4|\mathcal{S}|\mu(\mathbf{A})) \|\mathbf{d}\|_2^2. \quad (5.49)$$

This leads directly to the claimed result. Note that we must require  $1 + \mu(\mathbf{A}) - 4|\mathcal{S}|\mu(\mathbf{A}) > 0$  for the above inequality to be true, and thus the requirement on the cardinality of  $\mathbf{x}_0$  as the theorem states.  $\square$

For  $\epsilon = 0$ , we return to the noiseless case. This reveals a loss of tightness of the stated result, as the sparsity required is half the amount mentioned in Theorem 4.5. Another important observation is that the obtained bound on the distance between  $\mathbf{x}_0$  and  $\mathbf{x}_1^\epsilon$  is larger than  $4\epsilon^2$ , which is four times all the “noise” energy in the vector  $\mathbf{b}$ . This presumably suggests that there is no denoising effect in solving  $(P_1^\epsilon)$ . This is a direct consequence of the worst-case analysis that was practiced here, along with an assumption about the noise being adversary.

In addition to the above stability claim, results on successful recovery of the support would also be of interest. Work reporting such results exist, requiring more strict (and thus less realistic) conditions. Similarly, considering the above as signal a denoising procedure, there is an interest in the expected performance, but this will not be explored here.

### 5.5.2 Thresholding Stability Guarantee

Rather than providing a comparable stability result for the OMP, as in the paper by Donoho, Elad, and Temlyakov, we develop a similar theorem for the far sim-

pler thresholding algorithm. This way we show the two extreme methods and their guaranteed performance. Again, we rely on the structure of the proof as shown in Theorem 4.4 that handled the noiseless case. First, we state the theorem, and then provide its proof.

**Theorem 5.4. (Thresholding Performance):** *Consider the thresholding algorithm applied to the problem instance  $(P_0^\epsilon)$  with the triplet  $(\mathbf{A}, \mathbf{b}, \epsilon)$ . Suppose that a vector  $\mathbf{x}_0 \in \mathbb{R}^m$  satisfies  $\|\mathbf{b} - \mathbf{A}\mathbf{x}_0\|_2 \leq \epsilon$  and the sparsity constraint*

$$\|\mathbf{x}_0\|_0 < \frac{1}{2} \left( \frac{|x_{\min}|}{|x_{\max}|} \frac{1}{\mu(\mathbf{A})} + 1 \right) - \frac{\epsilon}{\mu(\mathbf{A})|x_{\max}|}, \quad (5.50)$$

where  $|x_{\min}|$  and  $|x_{\max}|$  are the minimum and maximum values of the vector  $|x_0|$  within its support. The result produced by the thresholding algorithm must obey

$$\|\mathbf{x}_{THR} - \mathbf{x}_0\|_2^2 \leq \frac{\epsilon^2}{1 - \mu(\mathbf{A})(\|\mathbf{x}_0\|_0 - 1)}. \quad (5.51)$$

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

Before turning to prove this result, a short discussion is in order. The sparsity requirement posed in Equation (5.50) is clearly sensitive to both the contrast of the non-zeros in the solution  $\mathbf{x}$ , and to the noise level. This is substantially different from the BPDN requirement that shows no similar sensitivities. On the other hand, the error bound in Equation (5.51) is much better than the one obtained for the BPDN. We assume that this is due to loss of tightness of the BPDN results.

**Proof:** We shall assume without loss of generality that the  $|\mathcal{S}| = k_0$  non-zeros in  $\mathbf{x}_0$  are its first entries, and thus  $\mathbf{b} = \mathbf{e} + \sum_{t=1}^{k_0} x_0[t] \mathbf{a}_t$ . Success of the thresholding algorithm is guaranteed by the requirement

$$\min_{1 \leq i \leq k_0} |\mathbf{a}_i^T \mathbf{b}| > \max_{j > k_0} |\mathbf{a}_j^T \mathbf{b}|, \quad (5.52)$$

Plugging the expression for  $\mathbf{b}$ , the left-hand-side term becomes

$$\min_{1 \leq i \leq k_0} |\mathbf{a}_i^T \mathbf{b}| = \min_{1 \leq i \leq k_0} \left| \mathbf{a}_i^T \mathbf{e} + \sum_{t=1}^{k_0} x_0[t] \mathbf{a}_i^T \mathbf{a}_t \right|. \quad (5.53)$$

We exploit the fact that the columns of  $\mathbf{A}$  are normalized, and their maximal inner product is bounded from above by  $\mu(\mathbf{A})$ . We also use the relations  $|a + b| \geq |a| - |b|$ , and  $\mathbf{v}^T \mathbf{u} \leq \|\mathbf{v}\|_2 \cdot \|\mathbf{u}\|_2$ . All these enable to lower-bound this term by

$$\begin{aligned} \min_{1 \leq i \leq k_0} \left| \mathbf{a}_i^T \mathbf{e} + \sum_{t=1}^{k_0} x_t \mathbf{a}_i^T \mathbf{a}_t \right| &= \\ &= \min_{1 \leq i \leq k_0} \left| x_i + \mathbf{a}_i^T \mathbf{e} + \sum_{1 \leq t \leq k_0, t \neq i} x_t \mathbf{a}_i^T \mathbf{a}_t \right| \end{aligned} \quad (5.54)$$

$$\begin{aligned}
&\geq \min_{1 \leq i \leq k_0} \left\{ |x_i| - |\mathbf{a}_i^T \mathbf{e}| - \left| \sum_{1 \leq t \leq k_0, t \neq i} x_t \mathbf{a}_i^T \mathbf{a}_t \right| \right\} \\
&\geq \min_{1 \leq i \leq k_0} |x_i| - \max_{1 \leq i \leq k_0} \left| \sum_{1 \leq t \leq k_0, t \neq i} x_t \mathbf{a}_i^T \mathbf{a}_t \right| - \epsilon \\
&\geq |x_{\min}| - (k_0 - 1)\mu(\mathbf{A})|x_{\max}| - \epsilon.
\end{aligned}$$

Turning to the right-hand-side term in Equation (5.52), and following similar steps, we aim to upper-bound this expression. This gives

$$\max_{j > k_0} |\mathbf{a}_j^T \mathbf{b}| = \max_{j > k_0} \left| \mathbf{a}_j^T \mathbf{e} + \sum_{t=1}^{k_0} x_t \mathbf{a}_j^T \mathbf{a}_t \right| \leq k_0 \mu(\mathbf{A}) |x_{\max}| + \epsilon. \quad (5.55)$$

Thus, requiring

$$|x_{\min}| - (k_0 - 1)\mu(\mathbf{A})|x_{\max}| > k_0 \mu(\mathbf{A})|x_{\max}| + 2\epsilon \quad (5.56)$$

necessarily leads to the satisfaction of the condition posed in Equation (5.52). Put differently, this reads as the condition

$$k_0 < \frac{1}{2} \left( \frac{|x_{\min}|}{|x_{\max}|} \frac{1}{\mu(\mathbf{A})} + 1 \right) - \frac{\epsilon}{\mu(\mathbf{A})|x_{\max}|} \quad (5.57)$$

which guarantees the success of the thresholding algorithm in recovering the proper support.

As for the deviation in the obtained solution, we shall assume that the above requirement is fulfilled, and thus the proper support has been recovered. Thus, the thresholding algorithm amounts to a simple Least-Squares solution of the form

$$\mathbf{x}_{THR}^s = \arg \min_{\mathbf{x}} \|\mathbf{A}_s \mathbf{x} - \mathbf{b}\|_2^2 = (\mathbf{A}_s^T \mathbf{A}_s)^{-1} \mathbf{A}_s^T \mathbf{b} \quad (5.58)$$

where this result refers only to the non-zero entries of the solution over the proper columns from  $\mathbf{A}$ , accumulated by the sub-matrix  $\mathbf{A}_s$ . We denote similarly  $\mathbf{x}_0^s$  as the portion of the vector  $\mathbf{x}_0$  on the proper support. Exploiting the fact that  $\|\mathbf{A}_s \mathbf{x}_0^s - \mathbf{b}\|_2 \leq \epsilon$  we obtain

$$\begin{aligned}
\|\mathbf{x}_{THR}^s - \mathbf{x}_0^s\|_2^2 &= \|(\mathbf{A}_s^T \mathbf{A}_s)^{-1} \mathbf{A}_s^T \mathbf{b} - \mathbf{x}_0^s\|_2^2 \\
&= \|(\mathbf{A}_s^T \mathbf{A}_s)^{-1} \mathbf{A}_s^T \mathbf{b} - (\mathbf{A}_s^T \mathbf{A}_s)^{-1} (\mathbf{A}_s^T \mathbf{A}_s) \mathbf{x}_0^s\|_2^2 \\
&= \|(\mathbf{A}_s^T \mathbf{A}_s)^{-1} \mathbf{A}_s^T (\mathbf{b} - \mathbf{A}_s \mathbf{x}_0^s)\|_2^2 \\
&\leq \|(\mathbf{A}_s^T \mathbf{A}_s)^{-1} \mathbf{A}_s^T\|_2^2 \cdot \|\mathbf{b} - \mathbf{A}_s \mathbf{x}_0^s\|_2^2 \\
&\leq \epsilon^2 \cdot \|(\mathbf{A}_s^T \mathbf{A}_s)^{-1} \mathbf{A}_s^T\|_2^2 = \epsilon^2 \cdot \|\mathbf{A}_s^+\|_2^2.
\end{aligned} \quad (5.59)$$

The term  $\|\mathbf{A}_s^+\|_2^2$  is the  $\ell_2$ -induced (operator) norm, defined as

$$\|\mathbf{B}\|_2^2 = \max_{\mathbf{v} \in \mathbb{R}^n} \frac{\|\mathbf{B}\mathbf{v}\|_2^2}{\|\mathbf{v}\|_2^2} = \lambda_{\max}(\mathbf{B}^T \mathbf{B}) = \lambda_{\max}(\mathbf{B}\mathbf{B}^T).$$

The fact that  $\lambda_{\max}(\mathbf{B}^T \mathbf{B}) = \lambda_{\max}(\mathbf{B}\mathbf{B}^T)$  can be directly derived by exploiting the SVD<sup>5</sup> of the matrix  $\mathbf{B}$ . Returning to our analysis of the thresholding algorithm, we need to bound from above the spectral radius of the matrix  $\mathbf{A}_s^+(\mathbf{A}_s^+)^T = (\mathbf{A}_s^T \mathbf{A}_s)^{-1}$ . If  $0 < \lambda_{\min} \leq \dots \leq \lambda_{\max}$  are the eigenvalues of the matrix  $\mathbf{A}_s^T \mathbf{A}_s$ , then their reciprocals are the eigenvalues of the inverse matrix, and thus the norm of the inverse is  $1/\lambda_{\min}$ , and thus

$$\begin{aligned} \|\mathbf{x}_{THR}^s - \mathbf{x}_0^s\|_2^2 &\leq \epsilon^2 \cdot \|\mathbf{A}_s^+\|_2^2 \\ &= \epsilon^2 \cdot \|(\mathbf{A}_s^T \mathbf{A}_s)^{-1}\|_2^2 \\ &= \frac{\epsilon^2}{\lambda_{\min}(\mathbf{A}_s^T \mathbf{A}_s)}. \end{aligned} \quad (5.60)$$

Thus, we need to bound this eigenvalue from below. Using the Gershgorin's disk theorem, all eigenvalues of  $\mathbf{A}_s^T \mathbf{A}_s$  must be inside the disk centered around 1, with radius  $(|S| - 1)\mu(\mathbf{A}) = (k_0 - 1)\mu(\mathbf{A})$  and thus  $\lambda_{\min}(\mathbf{A}_s^T \mathbf{A}_s) \geq 1 - (k_0 - 1)\mu(\mathbf{A})$ . This leads to

$$\|\mathbf{x}_{THR}^s - \mathbf{x}_0^s\|_2^2 \leq \frac{\epsilon^2}{\lambda_{\min}(\mathbf{A}_s^T \mathbf{A}_s)} \leq \frac{\epsilon^2}{1 - (k_0 - 1)\mu(\mathbf{A})}, \quad (5.61)$$

as claimed.  $\square$

## 5.6 Summary

In this chapter we have generalized ( $P_0$ ) to allow for an error in the equality  $\mathbf{A}\mathbf{x} = \mathbf{b}$ , and shown how most of the results from Chapters 2, 3, and 4 can be generalized to cover this more practical problem. As mentioned in Chapter 4, here as well we are under the dark shadows of too-pessimistic bounds, and these arise because of three main reasons:

1. Noise modeling: Throughout this chapter, the noise  $\mathbf{e}$  contaminating the vector  $\mathbf{A}\mathbf{x}_0$  is assumed as a deterministic adversary vector of known length  $\epsilon$ . This means that it may choose the worst realization in order to interfere the analysis/algorithm. By migrating from such a noise model to a random vector, better results can be obtained, accompanied by a (near 1) probability for the claimed bounds to be true. See Chapter 8, where we adopt such an approach for the analysis of the Dantzig-Selector.

<sup>5</sup> Assuming that the SVD of  $\mathbf{B}$  is given by  $\mathbf{B} = \mathbf{U}\Sigma\mathbf{V}^T$ , we have that  $\mathbf{B}\mathbf{B}^T = \mathbf{U}\Sigma\mathbf{V}^T\mathbf{V}\Sigma^T\mathbf{U}^T = \mathbf{U}\Sigma\Sigma^T\mathbf{U}^T$ , whereas  $\mathbf{B}^T\mathbf{B} = \mathbf{V}\Sigma^T\mathbf{U}^T\mathbf{U}\Sigma\mathbf{V}^T = \mathbf{V}\Sigma^T\Sigma\mathbf{V}^T$ . In both cases, the eigenvalues are simply the squared singular ones.



2. Allowing rare failures: The analysis we perform in this chapter does not allow a failure within the guaranteed range of the pursuit success/uniqueness. By allowing a very small fraction of such failures, the bounds could become much higher, and more optimistic. For most applications, this also reflects the true desire of the implementor – a willingness to sacrifice a pursuit problem from time to time, with the ability to generally claim a high rate of success. We adopt this point of view in Chapter 7.
3. Worst-case characterization of  $\mathbf{A}$ : The coherence, the RIP, and the *spark*, are all worst-case measures that characterize the matrix  $\mathbf{A}$ . Even if we improve our analysis with respect to the above two themes, using these measures in our bounds is likely to lead to overly pessimistic results. The solution is a replacement of these measures by others, more relaxed, such as a probabilistic RIP/coherence, etc.

There are more recent contributions that address the analysis of pursuit algorithms with better results, taking the above ideas into consideration. We mention some of those in Chapter 7, and we refer the interested readers to recent work by Ben-Haim et al.

A fundamental optimization problem that has been defined in this chapter is  $(Q_1^t)$ . This problem will accompany us throughout the next chapters, as it is key in the solution of various problems in this field. A reliable numerical solution of this problem would be of great value, and especially so for high-dimensional problems, where OMP, LARS, and IRLS cannot be trusted to perform well. In the next chapter we focus on this problem and introduce a family of algorithms suited for handling  $(Q_1^t)$ .

## Further Reading

1. Z. Ben-Haim, Y.C. Eldar, and M. Elad, Coherence-based performance guarantees for estimating a sparse vector under random noise, submitted to *IEEE Trans. on Signal Processing*, 2009.
2. S.S. Chen, D.L. Donoho, and M.A. Saunders, Atomic decomposition by basis pursuit, *SIAM Journal on Scientific Computing*, 20(1):33–61 (1998).
3. S.S. Chen, D.L. Donoho, and M.A. Saunders, Atomic decomposition by basis pursuit, *SIAM Review*, 43(1):129–159, 2001.
4. G. Davis, S. Mallat, and M. Avellaneda, Adaptive greedy approximations, *Journal of Constructive Approximation*, 13:57–98, 1997.
5. G. Davis, S. Mallat, and Z. Zhang, Adaptive time-frequency decompositions, *Optical-Engineering*, 33(7):2183–91, 1994.
6. D.L. Donoho and M. Elad, On the stability of the basis pursuit in the presence of noise, *Signal Processing*, 86(3):511–532, March 2006.
7. D.L. Donoho, M. Elad, and V. Temlyakov, Stable recovery of sparse overcomplete representations in the presence of noise, *IEEE Trans. on Information Theory*, 52(1):6–18, 2006.
8. B. Efron, T. Hastie, I.M. Johnstone, and R. Tibshirani, Least angle regression, *The Annals of Statistics*, 32(2):407–499, 2004.

9. A.K. Fletcher, S. Rangan, V.K. Goyal, and K. Ramchandran, Analysis of denoising by sparse approximation with random frame asymptotics, *IEEE Int. Symp. on Inform. Theory*, 2005.
10. A.K. Fletcher, S. Rangan, V.K. Goyal, and K. Ramchandran, Denoising by sparse approximation: error bounds based on rate-distortion theory, *EURASIP Journal on Applied Signal Processing*, Paper No. 26318, 2006.
11. J.J. Fuchs, Recovery of exact sparse representations in the presence of bounded noise, *IEEE Trans. on Information Theory*, 51(10):3601–3608, 2005.
12. A.C. Gilbert, S. Muthukrishnan, and M.J. Strauss, Approximation of functions over redundant dictionaries using coherence, 14th Ann. ACM-SIAM Symposium Discrete Algorithms, 2003.
13. I.F. Gorodnitsky and B.D. Rao, Sparse signal reconstruction from limited data using FOCUSS: A re-weighted norm minimization algorithm, *IEEE Trans. on Signal Processing*, 45(3):600–616, 1997.
14. R. Gribonval, R. Figueras, and P. Vandergheynst, A simple test to check the optimality of a sparse signal approximation, *Signal Processing*, 86(3):496–510, March 2006.
15. T. Hastie, R. Tibshirani, and J.H. Friedman, *Elements of Statistical Learning*. New York: Springer, 2001.
16. L.A. Karlovitz, Construction of nearest points in the  $\ell_p$ ,  $p$  even and  $\ell_\infty$  norms, *Journal of Approximation Theory*, 3:123–127, 1970.
17. S. Mallat, *A Wavelet Tour of Signal Processing*, Academic Press, 1998.
18. M.R. Osborne, B. Presnell, and B.A. Turlach, A new approach to variable selection in least squares problems, *IMA J. Numerical Analysis*, 20:389–403, 2000.
19. V.N. Temlyakov, Greedy algorithms and m-term approximation, *Journal of Approximation Theory*, 98:117–145, 1999.
20. V.N. Temlyakov, Weak greedy algorithms, *Advances in Computational Mathematics*, 5:173–187, 2000.
21. J.A. Tropp, Just relax: Convex programming methods for subset selection and sparse approximation, *IEEE Trans. on Information Theory*, 52(3):1030–1051, March 2006.
22. J.A. Tropp, A.C. Gilbert, S. Muthukrishnan, and M.J. Strauss, Improved sparse approximation over quasi-incoherent dictionaries, IEEE International Conference on Image Processing, Barcelona, September 2003.
23. B. Wohlberg, Noise sensitivity of sparse signal representations: Reconstruction error bounds for the inverse problem. *IEEE Trans. on Signal Processing*, 51(12):3053–3060, 2003.



# Chapter 6

## Iterative-Shrinkage Algorithms

### 6.1 Background

In this chapter, our goal is the minimization of a function of the form

$$f(\mathbf{x}) = \lambda \mathbf{1}^T \rho(\mathbf{x}) + \frac{1}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2, \quad (6.1)$$

which we have seen before in several forms. The function  $\rho(\mathbf{x})$  operates entry-wise on the vector  $\mathbf{x}$ . As an example, for  $\rho(x) = |x|^p$  we get that  $\mathbf{1}^T \rho(\mathbf{x}) = \|\mathbf{x}\|_p^p$ , which gives us the freedom to choose any  $p$  we find as fit. In this chapter we shall keep the discussion general, and allow any “sparsity-promoting” function  $\rho(\cdot)$ . In particular, the function posed above is a generalized version of what we have defined as  $(Q_1^1)$ .

Minimization of such functions can be treated using various classic iterative optimization algorithms ranging from Steepest-Descent and Conjugate-Gradient to more involved interior-point algorithms. However, all these general purpose methods are often found inefficient, requiring too many iterations and too many computations to reach their destination. This is especially the case for high-dimensional problems, as often encountered in image processing. In such cases, the IRLS, OMP, and LARS methods presented in previous chapters are also performing very poorly, and an alternative is needed.

In recent years, a new family of numerical algorithms has been gradually built, addressing the above optimization problem very effectively. This family is the *Iterative-Shrinkage* algorithms, that extend the classical Donoho-Johnston shrinkage method for signal denoising. Roughly speaking, in these iterative methods, each iteration comprises of a multiplication by  $\mathbf{A}$  and its adjoint, along with a scalar shrinkage step on the obtained result. Despite their simple structure, these algorithms are shown to be very effective in minimizing  $f(\mathbf{x})$  in Equation (6.1). A thorough theoretical analysis that took place in the past several years proves the convergence of these techniques, guarantees that the solution is the global minimizer for convex  $f$  (for a convex choice of  $\rho$ ), and studies the rate of convergence these algorithms exhibit.

There are various *Iterative-Shrinkage* methods, with variations between them. These algorithms are shown to emerge from very different considerations, such as the Expectation-Maximization (EM) algorithm in statistical estimation theory, proximal point and surrogate functions, the use of the fixed-point strategy, employment of a parallel Coordinate-Descent (CD) algorithm, variations on greedy methods, and more. This exposes the wealth with which one could design a solver for the above-described optimization task. In this chapter we give a broad view of this group of methods, their derivation, their speedup, and a little bit of their comparative performance. We should note that in this chapter our main focus is the actual construction of these algorithms, and the core intuition they are built on, and therefore some important topics, such as convergence studies, are omitted.

## 6.2 The Unitary Case - A Source of Inspiration

### 6.2.1 Shrinkage For the Unitary case

We already saw in Chapter 5 that if  $\mathbf{A}$  is unitary, the minimization of  $(Q_1^4)$  can be very easy, with a closed-form solution that leads to shrinkage. The function  $f(\mathbf{x})$  is somewhat more general, but as we shall see next, its treatment is exactly the same; in particular,  $f(\mathbf{x})$  can be manipulated in a sequence of simple steps and turned into a set of  $m$  ( $m = n$ ) independent and identical one-dimensional optimization tasks that are easily handled. Starting from Equation (6.1), using the identity  $\mathbf{A}\mathbf{A}^T = \mathbf{I}$ , and exploiting the fact that  $\ell_2$ -norm is unitary-invariant we get

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \mathbf{1}^T \rho(\mathbf{x}) \\ &= \frac{1}{2} \|\mathbf{A}(\mathbf{A}^T \mathbf{b} - \mathbf{x})\|_2^2 + \lambda \mathbf{1}^T \rho(\mathbf{x}) \\ &= \frac{1}{2} \|\mathbf{A}^T \mathbf{b} - \mathbf{x}\|_2^2 + \lambda \mathbf{1}^T \rho(\mathbf{x}) . \end{aligned} \quad (6.2)$$

Denoting  $\mathbf{x}_0 = \mathbf{A}^T \mathbf{b}$  we get

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{2} \|\mathbf{x}_0 - \mathbf{x}\|_2^2 + \lambda \mathbf{1}^T \rho(\mathbf{x}) \\ &= \sum_{k=1}^m \left[ \frac{1}{2} (x_0[k] - x[k])^2 + \lambda \rho(x[k]) \right] = \sum_{k=1}^m g(x[k], x_0[k]) . \end{aligned} \quad (6.3)$$

The minimization of the scalar function of the form  $g(x, a) = 0.5(x - a)^2 + \lambda \rho(x)$  with respect to  $x$  requires that we either zero the gradient (in the case of a smooth  $\rho(x)$ ), or show that the sub-gradient of  $g$  contains the zero, for the non-differentiable case. These can be solved (analytically in some cases, or numerically in others – this is done once and off-line), leading to the expression  $\hat{x}_{opt} = \mathcal{S}_{\rho, \lambda}(a)$  that finds

the global minimizer of the scalar objective function  $g(x, a)$  described above. Note that for non-convex choices of  $\rho$  there might be several solutions zeroing the (sub-)gradient. Thus, all solutions must be found and compared by their value of the function.

The obtained *Shrinkage* function is a curve that maps an input value  $a$  to the desired output value  $\hat{x}_{opt}$ . This function maps values near the origin to zero ( $\mathcal{S}_{\rho, \lambda}(a) = 0$  for  $|a| \leq T$ ), and those outside this interval are “shrunk,” thus the name of this operator. The threshold  $T$  and the shrinkage effects are both functions of the choices of  $\rho$  and  $\lambda$ .

Returning to our original problem, we have found a closed-form solution to the minimizer of  $f(\mathbf{x})$ , based on the following two steps: (i) Compute  $\mathbf{x}_0 = \mathbf{A}^T \mathbf{b}$ ; and (ii) Apply the operator  $\mathcal{S}_{\rho, \lambda}$  to each entry in  $\mathbf{x}_0$  and obtain the desired  $\hat{\mathbf{x}}$ . Note that even if the original function is non-convex, the found solution is globally optimal if  $\mathcal{S}_{\rho, \lambda}$  was designed properly.

The natural question raised by the above discussion is: When turning from a unitary matrix  $\mathbf{A}$  to a non-unitary (and perhaps non-square) one, must we lose all this simplicity? As we show in the next section, one could handle the general problem using similar shrinkage tools, but will be required to perform a sequence of them to obtain the desired result.

### 6.2.2 The BCR Algorithm and Variations

At least when it comes to a matrix  $\mathbf{A}$  built as a union of several unitary matrices, the answer to the above question is positive. This was observed by Sardy, Bruce, and Tseng in 1998, leading to their Block-Coordinate-Relaxation (BCR) algorithm. We present this method here very briefly, as it is a direct extension of the unitary case solver shown above. For simplicity, we shall assume that  $\mathbf{A} = [\mathbf{\Psi}, \mathbf{\Phi}]$ , where  $\mathbf{\Psi}$  and  $\mathbf{\Phi}$  are two  $n \times n$  unitary matrices. Our minimization can be re-written with respect to the two  $n$ -length portions of  $\mathbf{x}$ , denoted as  $\mathbf{x}_\Psi$  and  $\mathbf{x}_\Phi$ ,

$$\begin{aligned} f(\mathbf{x}) &= \frac{1}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \mathbf{1}^T \rho(\mathbf{x}) \\ &= \frac{1}{2} \|\mathbf{b} - \mathbf{\Psi}\mathbf{x}_\Psi - \mathbf{\Phi}\mathbf{x}_\Phi\|_2^2 + \lambda \mathbf{1}^T \rho(\mathbf{x}_\Psi) + \lambda \mathbf{1}^T \rho(\mathbf{x}_\Phi) = f(\mathbf{x}_\Psi, \mathbf{x}_\Phi). \end{aligned} \quad (6.4)$$

The core idea in the BCR algorithm is to minimize  $f(\mathbf{x}_\Psi, \mathbf{x}_\Phi)$  with respect to the two parts of  $\mathbf{x}$  separately and alternately. This is why this algorithm is named Block-Coordinate-Relaxation. Assume that we hold a current solution at the  $k$ -th iteration, denoted by  $\mathbf{x}^k$ , built of the two portions,  $\mathbf{x}_\Psi^k$  and  $\mathbf{x}_\Phi^k$ . Assuming that  $\mathbf{x}_\Phi^k$  is kept fixed, the function  $f(\mathbf{x}_\Psi, \mathbf{x}_\Phi^k)$  can be written as

$$f(\mathbf{x}_\Psi, \mathbf{x}_\Phi^k) = \frac{1}{2} \|\tilde{\mathbf{b}} - \mathbf{\Psi}\mathbf{x}_\Psi\|^2 + \lambda \mathbf{1}^T \rho(\mathbf{x}_\Psi), \quad (6.5)$$

where  $\tilde{\mathbf{b}} = \mathbf{b} - \Phi \mathbf{x}_\phi^k$ . This is the same function as in the unitary case, for which a closed-form solution is readily available to us. This solution is given as

$$\mathbf{x}_\psi^{k+1} = \mathcal{S}_{\rho,\lambda}(\Psi^T \tilde{\mathbf{b}}) = \mathcal{S}_{\rho,\lambda}(\Psi^T (\mathbf{b} - \Phi \mathbf{x}_\phi^k)). \quad (6.6)$$

Similarly, once  $\mathbf{x}_\psi^{k+1}$  has been computed and now kept fixed, the function  $f(\mathbf{x}_\psi^{k+1}, \mathbf{x}_\phi)$  is easily minimized with respect to  $\mathbf{x}_\phi$  with a closed-form expression, given by

$$\mathbf{x}_\phi^{k+1} = \mathcal{S}_{\rho,\lambda}(\Phi^T (\mathbf{b} - \Psi \mathbf{x}_\psi^{k+1})). \quad (6.7)$$

This way, alternating between these two update stages, the overall function is monotonically decreasing, and proven to converge to a local minimum of the penalty function (which becomes the global one for a convex  $\rho(\cdot)$ ).

An alternative to the above sequential process can be proposed, where the two updates are done in parallel. As before, we start with a current solution at the  $k$ -th iteration, denoted by  $\mathbf{x}^k$ , built of the two portions,  $\mathbf{x}_\psi^k$  and  $\mathbf{x}_\phi^k$ . We can propose an update of the two parts in parallel using the formula pair

$$\begin{aligned} \mathbf{x}_\psi^{k+1} &= \mathcal{S}_{\rho,\lambda}(\Psi^T (\mathbf{b} - \Phi \mathbf{x}_\phi^k)) \\ \mathbf{x}_\phi^{k+1} &= \mathcal{S}_{\rho,\lambda}(\Phi^T (\mathbf{b} - \Psi \mathbf{x}_\psi^k)). \end{aligned}$$

The only difference from the previous algorithm is in the update of  $\mathbf{x}_\phi^{k+1}$  that uses  $\mathbf{x}_\psi^k$  instead of  $\mathbf{x}_\psi^{k+1}$ . Merging these two into one formula can be done by the followings simple steps, and exploiting the fact that the shrinkage operator operates on scalars independently,

$$\begin{aligned} \mathbf{x}^{k+1} = \begin{bmatrix} \mathbf{x}_\psi^{k+1} \\ \mathbf{x}_\phi^{k+1} \end{bmatrix} &= \begin{bmatrix} \mathcal{S}_{\rho,\lambda}(\Psi^T (\mathbf{b} - \Phi \mathbf{x}_\phi^k)) \\ \mathcal{S}_{\rho,\lambda}(\Phi^T (\mathbf{b} - \Psi \mathbf{x}_\psi^k)) \end{bmatrix} \\ &= \mathcal{S}_{\rho,\lambda} \left( \begin{bmatrix} \Psi^T (\mathbf{b} - \mathbf{A} \mathbf{x}^k + \Psi \mathbf{x}_\psi^k) \\ \Phi^T (\mathbf{b} - \mathbf{A} \mathbf{x}^k + \Phi \mathbf{x}_\phi^k) \end{bmatrix} \right) \\ &= \mathcal{S}_{\rho,\lambda} \left( \begin{bmatrix} \Psi^T (\mathbf{b} - \mathbf{A} \mathbf{x}^k) + \mathbf{x}_\psi^k \\ \Phi^T (\mathbf{b} - \mathbf{A} \mathbf{x}^k) + \mathbf{x}_\phi^k \end{bmatrix} \right) \\ &= \mathcal{S}_{\rho,\lambda}(\mathbf{A}^T (\mathbf{b} - \mathbf{A} \mathbf{x}^k) + \mathbf{x}^k). \end{aligned} \quad (6.8)$$

This way we got an interesting closed-form formula for the update of the entire solution from one iteration to the next. Note that in this new form, nothing indicates that  $\mathbf{A}$  is expected to be of a special form (two-ortho). Naturally, we should wonder whether this formula could be used for more general matrices  $\mathbf{A}$ . Surprisingly, the answer is positive, with some restrictions,<sup>1</sup> as we shall see next.

We now turn to develop iterative algorithms of similar form that use a shrinkage step within each iteration. These algorithms consider a general matrix  $\mathbf{A}$ , and thus

<sup>1</sup> Those restrictions refer to the operator norm of the matrix  $\mathbf{A}$  that should be taken into account.

generalize the BCR approach done above. As we shall see, the formula obtained for the parallel update emerges naturally in various forms.

## 6.3 Developing Iterative-Shrinkage Algorithms

There are different ways to develop *Iterative-Shrinkage* algorithms. The first to explicitly propose such a technique were Starck, Murtagh, and Bijaoui in 1995, in the context of image deblurring using wavelet. However, their construction was done without a clear connection to the objective function in (6.1).

We choose to start our discussion with the proximal (or surrogate) functions that were used by Daubechies, Defrise, and De-Mol in their construction. As we show, this is also the foundation for the EM and the Bound-Optimization algorithm, as proposed by Figueiredo and Nowak. Then we turn to describe two entirely different constructions: the first, by Adeyemi and Davies, is based on the fixed-point method in optimization, coupled with the IRLS algorithm. The second method uses a parallel coordinate descent algorithm. Finally, we discuss the StOMP method by Donoho, Drori, Starck, and Tsaig, that develops an *Iterative-Shrinkage* method based on the matching pursuit algorithm.

### 6.3.1 Surrogate Functions and the Prox Method

The discussion below is based on the work of Daubechies, Defrise, and De-Mol. Considering the original function in Equation (6.1),

$$f(\mathbf{x}) = \frac{1}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \mathbf{1}^T \rho(\mathbf{x}) ,$$

let us add to it the following term

$$d(\mathbf{x}, \mathbf{x}_0) = \frac{c}{2} \|\mathbf{x} - \mathbf{x}_0\|_2^2 - \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{x}_0\|_2^2 .$$

The parameter  $c$  will be chosen such that the function  $d(\cdot)$  is strictly convex, implying that we require its Hessian to be positive-definite,  $c\mathbf{I} - \mathbf{A}^T \mathbf{A} > \mathbf{0}$ . This is satisfied by the choice  $c > \|\mathbf{A}^T \mathbf{A}\|_2 = \lambda_{\max}(\mathbf{A}^T \mathbf{A})$ . This new objective function

$$\tilde{f}(\mathbf{x}) = \frac{1}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \mathbf{1}^T \rho(\mathbf{x}) + \frac{c}{2} \|\mathbf{x} - \mathbf{x}_0\|_2^2 - \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{x}_0\|_2^2 , \quad (6.9)$$

is the *surrogate* function that will be used in the proposed algorithm. As we shall see next, the fact that the term  $\|\mathbf{A}\mathbf{x}\|_2^2$  drops in the new function, turns the minimization task into a much simpler and effective one. Opening the various terms in Equation (6.9) and re-organizing it, we obtain a new expression of the form



$$\begin{aligned}
\tilde{f}(\mathbf{x}) &= \frac{1}{2}\|\mathbf{b}\|_2^2 + \frac{1}{2}\|\mathbf{A}\mathbf{x}_0\|_2^2 + \frac{c}{2}\|\mathbf{x}_0\|_2^2 - \mathbf{b}^T \mathbf{A}\mathbf{x} + \lambda \mathbf{1}^T \rho(\mathbf{x}) \\
&\quad + \frac{c}{2}\|\mathbf{x}\|_2^2 - c\mathbf{x}^T \mathbf{x}_0 + \mathbf{x}^T \mathbf{A}^T \mathbf{A}\mathbf{x}_0 \\
&= \text{Const}_1 - \mathbf{x}^T \left[ \mathbf{A}^T (\mathbf{b} - \mathbf{A}\mathbf{x}_0) + c\mathbf{x}_0 \right] + \lambda \mathbf{1}^T \rho(\mathbf{x}) + \frac{c}{2}\|\mathbf{x}\|_2^2.
\end{aligned} \tag{6.10}$$

The constant in the above expression contains all the terms that are dependent on  $\mathbf{b}$  and  $\mathbf{x}_0$  alone. Using the definition

$$\mathbf{v}_0 = \frac{1}{c} \mathbf{A}^T (\mathbf{b} - \mathbf{A}\mathbf{x}_0) + \mathbf{x}_0, \tag{6.11}$$

we can rewrite  $\tilde{f}(\mathbf{x})$  (after division by  $c$  which has no impact on the minimization goal) as

$$\begin{aligned}
\tilde{f}(\mathbf{x}) &= \text{Const}_2 - \mathbf{x}^T \mathbf{v}_0 + \frac{\lambda}{c} \mathbf{1}^T \rho(\mathbf{x}) + \frac{1}{2}\|\mathbf{x}\|_2^2 \\
&= \text{Const}_3 + \frac{\lambda}{c} \mathbf{1}^T \rho(\mathbf{x}) + \frac{1}{2}\|\mathbf{x} - \mathbf{v}_0\|_2^2.
\end{aligned} \tag{6.12}$$

As it turns out, this penalty function is exactly the one we minimized for the unitary case (see Equation (6.3)), and the solution is given as

$$\mathbf{x}_{opt} = \mathcal{S}_{\rho, \lambda/c}(\mathbf{v}_0) = \mathcal{S}_{\rho, \lambda/c} \left( \frac{1}{c} \mathbf{A}^T (\mathbf{b} - \mathbf{A}\mathbf{x}_0) + \mathbf{x}_0 \right). \tag{6.13}$$

This is a global minimizer of the function  $\tilde{f}(\mathbf{x})$  in Equation (6.9).

So far we managed to convert the original function  $f$  to a new function  $\tilde{f}$ , for which we are able to get a closed-form expression for its global minimizer. This change of the objective function depends on the choice of the vector  $\mathbf{x}_0$ . The core idea of using the surrogate function is that we minimize the function  $f$  iteratively, producing the sequence of results  $\{\mathbf{x}_i\}_k$ , where at the  $(k+1)$ -th iteration we minimize  $\tilde{f}$  with the assignment  $\mathbf{x}_0 = \mathbf{x}_k$ . Perhaps the more surprising of all is the fact that the sequence of solutions  $\{\mathbf{x}_k\}_k$  is proven to converge to the (local) minimizer of the original function  $f$ . Thus, the proposed algorithm is simply an iterative application of Equation (6.14) in the form

$$\mathbf{x}_{k+1} = \mathcal{S}_{\rho, \lambda/c} \left( \frac{1}{c} \mathbf{A}^T (\mathbf{b} - \mathbf{A}\mathbf{x}_k) + \mathbf{x}_k \right). \tag{6.14}$$

We shall refer hereafter to this algorithm as the Separable Surrogate Functionals (SSF) method, and it is described in details in [Figure 6.1](#), with a possible addition of a line-search. For the choice  $\mu = 1$  this reduces to the plain SSF algorithm as developed here.

The above approach can be interpreted as the proximal-point algorithm, which is well known in optimization theory. The function  $d(\mathbf{x}, \mathbf{x}_0)$  is a distance measure to the previous solution. When added to the original function at the  $k$ -th iteration, it

**Task:** Find  $\mathbf{x}$  that minimizes  $f(\mathbf{x}) = \lambda \mathbf{1}^T \rho(\mathbf{x}) + \frac{1}{2} \cdot \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2$ .

**Initialization:** Initialize  $k = 0$ , and set

- The initial solution  $\mathbf{x}_0 = \mathbf{0}$ .
- The initial residual  $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_k = \mathbf{b}$ .

**Main Iteration:** Increment  $k$  by 1, and apply these steps:

- **Back-Projection:** Compute  $\mathbf{e} = \mathbf{A}^T \mathbf{r}_{k-1}$ .
- **Shrinkage:** Compute  $\mathbf{e}_s = \text{Shrink}(\mathbf{x}_{k-1} + \mathbf{e}/c)$  with threshold  $\lambda$ .
- **Line search (optional):** Choose  $\mu$  to minimize the real valued function  $f(\mathbf{x}_{k-1} + \mu(\mathbf{e}_s - \mathbf{x}_{k-1}))$ .
- **Update Solution:** Compute  $\mathbf{x}_k = \mathbf{x}_{k-1} + \mu(\mathbf{e}_s - \mathbf{x}_{k-1})$ .
- **Update Residual:** Compute  $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$ .
- **Stopping Rule:** If  $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2^2$  is smaller than some predetermined threshold, stop. Otherwise, apply another iteration.

**Output:** The result is  $\mathbf{x}_k$ .

**Fig. 6.1** The SSF iterative-shrinkage algorithm with line-search.

promotes proximity between subsequent estimates of the iterative process. Surprisingly, while this is expected to have an effect of slowing down the algorithm, in the case discussed here it is actually leading to a substantial speed-up. As a quadratic and convex term, this distance function has the shape of an ellipsoid. For directions  $\mathbf{x} - \mathbf{x}_0$  that are close to the null-space of  $\mathbf{A}$ , the distance is nearly Euclidean,  $c\|\mathbf{x} - \mathbf{x}_0\|_2^2/2$ . For directions that are spanned by the rows of  $\mathbf{A}$ , the distance drops to nearly zero. Thus,  $d(\mathbf{x}, \mathbf{x}_0)$  poses some sort of a narrow tunnel of permitted (or preferred) solutions, which supports the overall problem.

As a final point in this section, we draw the reader's attention to the similarity between the SSF algorithm and the iterative formula obtained for the parallel-BCR in Equation (6.8). In fact, there is a slight difference – if the SSF is applied to a two-ortho matrix  $\mathbf{A}$ , then as  $\lambda_{\max}(\mathbf{A}^T \mathbf{A}) = 2$ , we need to add a factor of 0.5 in the SSF formula. However, such a factor is not necessary in practice – in a recent analysis done by Combettes and Wajs it is shown that the constant  $c$  should satisfy  $c > 0.5\lambda_{\max}(\mathbf{A}^T \mathbf{A})$  in order to guarantee a convergence of the SSF algorithm, exactly as the formula in Equation (6.8) employs.

### 6.3.2 EM and Bound-Optimization Approaches

The discussion above could be replaced with a different set of considerations, and yet, one that leads to the very same algorithm. This new perspective is taken from the work of Figueiredo, Nowak, and Bioucas-Dias, that uses the Expectation-

Maximization (EM) estimator, or better-yet, its alternative deterministic optimization foundation - the Bound-Optimization method.<sup>2</sup>

Starting from the function  $f(\mathbf{x})$  in Equation (6.1), which is hard to minimize, the bound-optimization method suggests to use a related function  $Q(\mathbf{x}, \mathbf{x}_0)$  that has the following properties:

1. **Equality at  $\mathbf{x} = \mathbf{x}_0$ :**  $Q(\mathbf{x}_0, \mathbf{x}_0) = f(\mathbf{x}_0)$ ;
2. **Upper-bounding the original function:**  $Q(\mathbf{x}, \mathbf{x}_0) \geq f(\mathbf{x})$  for all  $\mathbf{x}$ ; and
3. **Tangent<sup>3</sup> at  $\mathbf{x}_0$ :**  $\nabla Q(\mathbf{x}, \mathbf{x}_0)|_{\mathbf{x}=\mathbf{x}_0} = \nabla f(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_0}$ .

Such a function necessarily exists if the Hessian of the original function is bounded from above by  $\nabla^2 f(\mathbf{x}) \leq \mathbf{H}$ , because then one can propose a bounding quadratic function of the form  $Q(\mathbf{x}, \mathbf{x}_0) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^T (\mathbf{x} - \mathbf{x}_0) + 0.5(\mathbf{x} - \mathbf{x}_0)^T \mathbf{H}(\mathbf{x} - \mathbf{x}_0)$ , satisfying these conditions exactly.

Using the function  $Q(\mathbf{x}_0, \mathbf{x}_0)$ , the sequence of solutions generated by the recurrent formula

$$\mathbf{x}_{k+1} = \arg \min_{\mathbf{x}} Q(\mathbf{x}, \mathbf{x}_k) \quad (6.15)$$

is guaranteed to converge to a local minimum of the original function  $f(\mathbf{x})$ . As can be easily verified, the choice from the previous sub-section,

$$Q(\mathbf{x}, \mathbf{x}_0) = \frac{1}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \mathbf{1}^T \rho(\mathbf{x}) + \frac{c}{2} \|\mathbf{x} - \mathbf{x}_0\|_2^2 - \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{A}\mathbf{x}_0\|_2^2,$$

satisfies all these three conditions, and as such, leads to the desired optimization algorithm. Note that beyond the above three conditions, we need to make sure that minimization of  $Q$  is easier, as indeed happens, due to its separability. Also, the value  $f(\mathbf{x}_i)$  is necessarily decreasing since

$$f(\mathbf{x}_{k+1}) \leq Q(\mathbf{x}_{k+1}, \mathbf{x}_k) = \min_{\mathbf{x}} Q(\mathbf{x}, \mathbf{x}_k) \leq Q(\mathbf{x}_k, \mathbf{x}_k) = f(\mathbf{x}_k). \quad (6.16)$$

In the above inequality, the relation  $f(\mathbf{x}_{k+1}) \leq Q(\mathbf{x}_{k+1}, \mathbf{x}_k)$  is a direct consequence of the second property of  $Q$ . Also, the relation  $\min_{\mathbf{x}} Q(\mathbf{x}, \mathbf{x}_k) \leq Q(\mathbf{x}_k, \mathbf{x}_k)$  is guaranteed to give a strong decrease in the function's value due to the second property. By the Taylor expansion, it is easily verified that being tangent to the original function  $f$  at  $\mathbf{x}_k$  implies a descent of at least  $\nabla f(\mathbf{x}_k)^T \mathbf{H}^{-1} \nabla f(\mathbf{x}_k)$ , where  $\mathbf{H}$  is an upper bound on the original function's Hessian.<sup>4</sup>

<sup>2</sup> This technique is also known as the Majorization-Maximization method.

<sup>3</sup> In fact, this requirement can be removed without harming the claims made.

<sup>4</sup> Taking the quadratic function proposed above and minimizing it, the descent obtained is the one mentioned here.

### 6.3.3 An IRLS-Based Shrinkage Algorithm

As we have seen in Chapter 5, a popular algorithm for minimization of  $f(\mathbf{x})$  in Equation (6.1) is the Iterative-Reweighted Least-Squares (IRLS) algorithm. It converts non- $\ell_2$ -norms to  $\ell_2$  ones by an introduction of weighting. In this approach, an expression of the form  $\mathbf{1}^T \rho(\mathbf{x})$  is replaced by  $0.5 \mathbf{x}^T \mathbf{W}^{-1}(\mathbf{x}) \mathbf{x}$ , where  $\mathbf{W}(\mathbf{x})$  is a diagonal matrix, holding the values  $W[k, k] = 0.5 x[k]^2 / \rho(x[k])$  in its main-diagonal. While this change by itself is trivial, when plugged wisely into an iterative process, it leads to an appealing algorithm. Consider the function

$$f(\mathbf{x}) = \frac{1}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \lambda \mathbf{1}^T \rho(\mathbf{x}) = \frac{1}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 + \frac{\lambda}{2} \mathbf{x}^T \mathbf{W}^{-1}(\mathbf{x}) \mathbf{x}.$$

Assume that we hold a current solution  $\mathbf{x}_0$ , and we aim to update it. This is done in two stages. The first updates  $\mathbf{x}$  assuming a fixed  $\mathbf{W}$ . This requires a minimization of simple quadratic function, leading to the linear system

$$\nabla f(\mathbf{x}) = -\mathbf{A}^T(\mathbf{b} - \mathbf{A}\mathbf{x}) + \lambda \mathbf{W}^{-1}(\mathbf{x}) \mathbf{x} = \mathbf{0}. \quad (6.17)$$

Thus, inversion of the matrix  $\mathbf{A}^T \mathbf{A} + \lambda \mathbf{W}^{-1}$ , either directly (for low-dimensional problems) or iteratively (several Conjugate-Gradient steps) leads to the desired update. Once found, we update  $\mathbf{W}$  based on the new values of the found solution. This is the IRLS core as already described in Chapter 5, and its performance is poor, especially for high-dimensional problems.

An interesting twist over the above algorithm is proposed by Adeyemi and Davies, leading to yet another *Iterative-Shrinkage* algorithm. Consider Equation (6.17), and write it slightly different, by adding and subtracting  $c\mathbf{x}$ . The constant  $c \geq 1$  is a relaxation constant, whose role is to be discovered soon. Thus,

$$-\mathbf{A}^T \mathbf{b} + (\mathbf{A}^T \mathbf{A} - c\mathbf{I})\mathbf{x} + (\lambda \mathbf{W}^{-1}(\mathbf{x}) + c\mathbf{I})\mathbf{x} = \mathbf{0}. \quad (6.18)$$

A well-known technique in optimization, known as the fixed-point iteration method, suggests a way to construct iterative algorithms by assigning (wisely! so as to guarantee convergence and also allow for a simple computation) iteration counter to the appearances of  $\mathbf{x}$  in the equation. In our case, the proposed assignment would be

$$\mathbf{A}^T \mathbf{b} - (\mathbf{A}^T \mathbf{A} - c\mathbf{I})\mathbf{x}_k = (\lambda \mathbf{W}^{-1}(\mathbf{x}_k) + c\mathbf{I})\mathbf{x}_{k+1}, \quad (6.19)$$

leading to the iterative equation

$$\begin{aligned} \mathbf{x}_{k+1} &= \left( \frac{\lambda}{c} \mathbf{W}^{-1}(\mathbf{x}_k) + \mathbf{I} \right)^{-1} \left( \frac{1}{c} \mathbf{A}^T \mathbf{b} - \frac{1}{c} (\mathbf{A}^T \mathbf{A} - c\mathbf{I})\mathbf{x}_k \right) \\ &= \mathbf{S} \cdot \left( \frac{1}{c} \mathbf{A}^T (\mathbf{b} - \mathbf{A}\mathbf{x}_k) + \mathbf{x}_k \right). \end{aligned} \quad (6.20)$$

where we have defined the diagonal matrix

$$\mathbf{S} = \left( \frac{\lambda}{c} \mathbf{W}^{-1}(\mathbf{x}_k) + \mathbf{I} \right)^{-1} = \left( \frac{\lambda}{c} \mathbf{I} + \mathbf{W}(\mathbf{x}_k) \right)^{-1} \mathbf{W}(\mathbf{x}_k). \quad (6.21)$$

This matrix plays the role of shrinkage of the entries in the vector  $\frac{1}{c} \mathbf{A}^T (\mathbf{b} - \mathbf{A} \mathbf{x}_k) + \mathbf{x}_k$ , just as it is done in Equation (6.14). Indeed, every entry is multiplied by the scalar value

$$\frac{0.5 x_k[i]^2 / \rho(x_k[i])}{\frac{\lambda}{c} + 0.5 x_k[i]^2 / \rho(x_k[i])} = \frac{x_k[i]^2}{\frac{2\lambda}{c} \rho(x_k[i]) + x_k[i]^2}. \quad (6.22)$$

This way, for large values of  $|x_k[i]|$  this factor is nearly 1, whereas for small values of  $|x_k[i]|$  this value tends to zero, just as shrinkage does.

Note that this algorithm cannot be initialized with zeros, as in previous methods,<sup>5</sup> since this vector is a stable solution to this fixed-point iteration. Interestingly, once an entry in the solution becomes zero, it can never be “revived,” implying that this algorithm may get stuck, avoiding the near-by local minimum. Figure 6.2 presents this algorithm in details.

The constant  $c$  is crucial for the convergence of this algorithm, and its choice is similar to the one in the SSF algorithm. The iterative algorithm in Equation (6.20) applies the matrix  $\mathbf{S} \cdot (\frac{1}{c} \mathbf{A}^T \mathbf{A} - \mathbf{I})$  sequentially. For convergence, we must require that the spectral radius of this matrix is below 1. Since  $\|\mathbf{S}\|_2 \leq 1$ , this implies that  $c$  should be chosen such that the matrix  $\frac{1}{c} \mathbf{A}^T \mathbf{A} - \mathbf{I}$  is convergent. As the eigenvalues of this matrix are in the range  $[-1, -1 + \lambda_{\max}(\mathbf{A}^T \mathbf{A})/c]$ , we should require  $c > \lambda_{\max}(\mathbf{A}^T \mathbf{A})/2$ , which is twice more tolerant, compared to the choice made in the SSF algorithm.

### 6.3.4 The Parallel-Coordinate-Descent (PCD) Algorithm

We now turn to describe the PCD algorithm, developed by Elad, Matalon, and Zibulevsky. The construction below starts from a simple coordinate descent algorithm, but then merges a set of such descent steps into one easier joint step, leading to the Parallel-Coordinate-Descent (PCD) *Iterative-Shrinkage* method. The rationale practiced here should remind the reader of the BCR algorithm by Sardy, Bruce, and Tseng, that handled a special case of  $\mathbf{A}$  being a union of unitary matrices.

Returning to  $f(\mathbf{x})$  in Equation (6.1), we start by proposing a Coordinate Descent (CD) algorithm that updates one entry at a time in  $\mathbf{x}$  while keeping the rest untouched. A sequence of such rounds of  $m$  steps (addressing each of the entries in  $\mathbf{x} \in \mathbb{R}^m$ ) is necessarily converging. Interestingly, as we are about to show, each of these steps is obtained via shrinkage.

Assuming that the current solution is  $\mathbf{x}_0$ , we desire to update the  $i$ -th entry around its current value  $x_0[i]$ . This leads to a 1D function of the form

$$g(z) = \frac{1}{2} \|\mathbf{b} - \mathbf{A} \mathbf{x}_0 - \mathbf{a}_i(z - x_0[i])\|_2^2 + \lambda \rho(z). \quad (6.23)$$

<sup>5</sup> In our tests we use a vector with a constant (0.001) in all its entries.

**Task:** Find  $\mathbf{x}$  that minimizes  $f(\mathbf{x}) = \lambda \mathbf{1}^T \rho(\mathbf{x}) + \frac{1}{2} \cdot \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2$ .

**Initialization:** Initialize  $k = 0$ , and set

- The initial solution  $\mathbf{x}_0 = \mathbf{1}$ .
- The initial residual  $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0 = \mathbf{b}$ .

**Main Iteration:** Increment  $k$  by 1, and apply these steps:

- **Back-Projection:** Compute  $\mathbf{e} = \mathbf{A}^T \mathbf{r}_{k-1}$ .
- **Shrink-Update:** Compute the diagonal matrix  $\mathbf{S}$  by  $S[i, i] = x_k[i]^2 / (\frac{2\lambda}{c} \rho(x_k[i]) + x_k[i]^2)$ .
- **Shrinkage:** Compute  $\mathbf{e}_s = \mathbf{S}(\mathbf{x}_{k-1} + \mathbf{e}/c)$ .
- **Line search (optional):** Choose  $\mu$  to minimize the real valued function  $f(\mathbf{x}_{k-1} + \mu(\mathbf{e}_s - \mathbf{x}_{k-1}))$ .
- **Update Solution:** Compute  $\mathbf{x}_k = \mathbf{x}_{k-1} + \mu(\mathbf{e}_s - \mathbf{x}_{k-1})$ .
- **Update Residual:** Compute  $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$ .
- **Stopping Rule:** If  $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2^2$  is smaller than some predetermined threshold, stop. Otherwise, apply another iteration.

**Output:** The result is  $\mathbf{x}_k$ .

**Fig. 6.2** The IRLS-based iterative-shrinkage algorithm with line-search.

The vector  $\mathbf{a}_i$  is the  $i$ -th column in  $\mathbf{A}$ . The term  $\mathbf{a}_i(z - x_0[i])$  removes the effect of the old value and adds the new one. Written differently, and denoting  $\tilde{\mathbf{b}} = \mathbf{b} - \mathbf{A}\mathbf{x}_0 + x_0[i]\mathbf{a}_i$ , this function becomes

$$\begin{aligned}
 g(z) &= \frac{1}{2} \|\tilde{\mathbf{b}} - \mathbf{a}_i z\|_2^2 + \lambda \rho(z) \\
 &= \frac{1}{2} \|\tilde{\mathbf{b}}\|_2^2 - \tilde{\mathbf{b}}^T \mathbf{a}_i z + \frac{1}{2} \|\mathbf{a}_i\|_2^2 \cdot z^2 + \lambda \rho(z) \\
 &= \|\mathbf{a}_i\|_2^2 \left( \frac{\|\tilde{\mathbf{b}}\|_2^2}{2\|\mathbf{a}_i\|_2^2} - \frac{\mathbf{a}_i^T \tilde{\mathbf{b}}}{\|\mathbf{a}_i\|_2^2} z + \frac{z^2}{2} + \frac{\lambda}{\|\mathbf{a}_i\|_2^2} \rho(z) \right) \\
 &= \|\mathbf{a}_i\|_2^2 \left( \frac{1}{2} \left( z - \frac{\mathbf{a}_i^T \tilde{\mathbf{b}}}{\|\mathbf{a}_i\|_2^2} \right)^2 + \frac{\lambda}{\|\mathbf{a}_i\|_2^2} \rho(z) \right) + \text{Const.}
 \end{aligned} \tag{6.24}$$

The 1D function obtained has exactly the same form as the one shown in Equation (6.3) for the unitary case, for which a shrinkage computation provides the minimizer. Thus, the solution for obtaining the optimal  $z$  is given by

$$\begin{aligned}
 z_{opt} &= \mathcal{S}_{\rho, \lambda / \|\mathbf{a}_i\|_2^2} \left( \frac{\mathbf{a}_i^T \tilde{\mathbf{b}}}{\|\mathbf{a}_i\|_2^2} \right) \\
 &= \mathcal{S}_{\rho, \lambda / \|\mathbf{a}_i\|_2^2} \left( \frac{1}{\|\mathbf{a}_i\|_2^2} \mathbf{a}_i^T (\mathbf{b} - \mathbf{A}\mathbf{x}_0) + x_0[i] \right).
 \end{aligned} \tag{6.25}$$

While this algorithm may work well in low-dimensional cases, it is impractical in high-dimension problems in general, where the matrix  $\mathbf{A}$  is not held explicitly, and only multiplications by it and its adjoint are possible. Thus as this algorithm requires an extraction of columns from  $\mathbf{A}$  one at a time, it becomes prohibitive.

We consider a modification of the above method that overcomes the problem posed, and we rely on the following property: When minimizing a function, if there are several descent directions, then any non-negative combination of them is also a descent direction. Thus, we propose a simple linear combination of the set of all the above  $m$  steps in Equation (6.25). Since each of these steps handles one entry in the destination vector, we can write this sum as

$$\begin{aligned} \mathbf{v}_0 &= \sum_{i=1}^m \mathbf{e}_i \cdot \mathcal{S}_{\rho, \lambda / \|\mathbf{a}_i\|_2^2} \left( \frac{1}{\|\mathbf{a}_i\|_2^2} \mathbf{a}_i^T (\mathbf{b} - \mathbf{A}\mathbf{x}_0) + x_0[i] \right) = \\ &= \begin{bmatrix} \mathcal{S}_{\rho, \lambda / \|\mathbf{a}_1\|_2^2} \left( \frac{1}{\|\mathbf{a}_1\|_2^2} \mathbf{a}_1^T (\mathbf{b} - \mathbf{A}\mathbf{x}_0) + x_0[1] \right) \\ \vdots \\ \mathcal{S}_{\rho, \lambda / \|\mathbf{a}_i\|_2^2} \left( \frac{1}{\|\mathbf{a}_i\|_2^2} \mathbf{a}_i^T (\mathbf{b} - \mathbf{A}\mathbf{x}_0) + x_0[i] \right) \\ \vdots \\ \mathcal{S}_{\rho, \lambda / \|\mathbf{a}_m\|_2^2} \left( \frac{1}{\|\mathbf{a}_m\|_2^2} \mathbf{a}_m^T (\mathbf{b} - \mathbf{A}\mathbf{x}_0) + x_0[m] \right) \end{bmatrix}. \end{aligned} \quad (6.26)$$

In the above expression,  $\mathbf{e}_i$  is the trivial  $m$ -dimensional vector with zeros in all entries and 1 in the  $i$ -th one. Rewriting of this expression leads to the following simpler formula

$$\mathbf{v}_0 = \mathcal{S}_{\rho, \text{diag}(\mathbf{A}^T \mathbf{A})^{-1} \lambda} \left( \text{diag}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T (\mathbf{b} - \mathbf{A}\mathbf{x}_0) + \mathbf{x}_0 \right). \quad (6.27)$$

The term  $\text{diag}(\mathbf{A}^T \mathbf{A})$  consists of the norms of the columns of the dictionary  $\mathbf{A}$ . These are used both for the weighting of the back-projected error  $\mathbf{A}^T (\mathbf{b} - \mathbf{A}\mathbf{x}_0)$ , and for the shrinkage operator. This set of weights can be computed off-line, before the algorithm starts, and there exists a fast way to approximate these weights.<sup>6</sup> Notice that the obtained formula does not call for an extraction of columns from  $\mathbf{A}$  as in Equation (6.25), and the operations required are a direct multiplication of  $\mathbf{A}$  and its adjoint by vectors, as in Equation (6.14).

While each of the CD directions is guaranteed to descend, their linear combination is not necessarily descending without a proper scaling. Thus, we consider this direction and perform a Line Search (LS) along it. This means that the actual iterative algorithm is of the form

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mu (\mathbf{v}_k - \mathbf{x}_k) = \quad (6.28)$$

<sup>6</sup> A stochastic algorithm can be proposed, based on the fact that for a white Gaussian vector  $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \in \mathbb{R}^n$ , the multiplication  $\mathbf{A}^T \mathbf{u}$  has a covariance matrix  $\mathbf{A}^T \mathbf{A}$ . Thus, by choosing a set of such random vectors and computing their multiplication by  $\mathbf{A}^T$ , the variance of each entry is an approximation of the required norms.

**Task:** Find  $\mathbf{x}$  that minimizes  $f(\mathbf{x}) = \lambda \mathbf{1}^T \rho(\mathbf{x}) + \frac{1}{2} \cdot \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2$ .

**Initialization:** Initialize  $k = 0$ , and set

- The initial solution  $\mathbf{x}_0 = \mathbf{0}$ .
- The initial residual  $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_k = \mathbf{b}$ .

Prepare the weights  $\mathbf{W} = \text{diag}(\mathbf{A}^T \mathbf{A})^{-1}$ .

**Main Iteration:** Increment  $k$  by 1, and apply these steps:

- **Back-Projection:** Compute  $\mathbf{e} = \mathbf{A}^T \mathbf{r}_{k-1}$ .
- **Shrinkage:** Compute  $\mathbf{e}_s = \text{Shrink}(\mathbf{x}_{k-1} + \mathbf{W}\mathbf{e})$ , using thresholds given as  $\lambda W[i, i]$ .
- **Line search (optional):** Choose  $\mu$  to minimize the real valued function  $f(\mathbf{x}_{k-1} + \mu(\mathbf{e}_s - \mathbf{x}_{k-1}))$ .
- **Update Solution:** Compute  $\mathbf{x}_k = \mathbf{x}_{k-1} + \mu(\mathbf{e}_s - \mathbf{x}_{k-1})$ .
- **Update Residual:** Compute  $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$ .
- **Stopping Rule:** If  $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_2^2$  is smaller than some predetermined threshold, stop. Otherwise, apply another iteration.

**Output:** The result is  $\mathbf{x}_k$ .

**Fig. 6.3** The PCD iterative-shrinkage algorithm with line-search.

$$= \mathbf{x}_k + \mu \left( S_{\rho, \text{diag}(\mathbf{A}^T \mathbf{A})^{-1} \lambda} \left( \text{diag}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T (\mathbf{b} - \mathbf{A}\mathbf{x}_k) + \mathbf{x}_k \right) - \mathbf{x}_k \right),$$

with  $\mu$  chosen by a line-search algorithm of some sort. This means that we are required to perform 1D optimization of the function

$$h(\mu) = \frac{1}{2} \|\mathbf{b} - \mathbf{A}(\mathbf{x}_k + \mu(\mathbf{v}_k - \mathbf{x}_k))\|_2^2 + \lambda \mathbf{1}^T \rho(\mathbf{x}_k + \mu(\mathbf{v}_k - \mathbf{x}_k)),$$

which requires only two more multiplications of  $\mathbf{A}$  by  $\mathbf{v}_k$  and  $\mathbf{x}_k$ . This overall process will be referred to hereafter as the Parallel-Coordinate-Descent (PCD) algorithm. This algorithm is presented in [Figure 6.3](#).

As we can see, compared to the SSF in Equation (6.14), the method derived here is different in two ways: (i) the norms of the atoms in  $\mathbf{A}$  play an important role in weighting the back-projected error, whereas the previous algorithm uses a constant; and (ii) the new algorithm requires a line-search to obtain a descent. We will return to discuss these differences towards the end of this section, in an attempt to bridge between the two methods.

### 6.3.5 StOMP: A Variation on Greedy Methods

The last algorithm we describe within this family of *Iterative-Shrinkage* methods is the Stage-wise Orthogonal-Matching-Pursuit (StOMP). As opposed to the previ-



ous methods, its construction does not emerge from an attempt to minimize  $f(\mathbf{x})$  in Equation (6.1), although it could be attributed to the minimization of this function with the  $\ell_0$ -norm as the sparsity penalty. Instead, the creators of StOMP embark from the Orthogonal-Matching-Pursuit (OMP) and the Least-Angle Regression (LARS) algorithms. We should also mention that the StOMP algorithm is especially tailored for cases where the matrix  $\mathbf{A}$  is random, such as in compressed-sensing, although it may be used in other cases with some success. Despite these major differences between StOMP and the other methods in this chapter, we chose to describe it here due to its general resemblance to an iterative-shrinkage procedure. StOMP algorithm comprises of the following steps:

**Initialization:** We initialize the by fixing the solution to be zero,  $\mathbf{x}_0 = \mathbf{0}$ , and thus the residual vector is  $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0 = \mathbf{b}$ . We also define the support of the found solution and set it to be empty,  $I_0 = \phi$ . Turning to the first iteration and beyond ( $k \geq 1$ ), the following steps are performed.

**Back-Project the Residual:** Compute  $\mathbf{e}_k = \mathbf{A}^T(\mathbf{b} - \mathbf{A}\mathbf{x}_{k-1}) = \mathbf{A}^T\mathbf{r}_{k-1}$ . Most of the obtained values are assumed to be zero-mean iid Gaussian due to the assumed randomness of the dictionary. Few outliers in the above vector are those referring to possible atoms in the construction of  $\mathbf{b}$ .

**Thresholding:** We define the set of dominant entries in  $\mathbf{e}_k$  as  $J_k = \{i | 1 \leq i \leq m, |e_k[i]| > T\}$ , with a threshold chosen based on the permitted error  $\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2$ . If the threshold is chosen such that  $|J_k| = 1$ , we get the OMP, as only one entry is found and updated. In the more general case considered by StOMP, several and even many entries can be chosen at once.

**Union of Supports:** We update the support of the desired solution to be  $I_k = I_{k-1} \cup J_k$ . Thus, this support is incrementally growing.

**Restricted Least-Squares:** Given the support and assuming that  $|I_k| < n$ , we solve the Least-Squares problem

$$\arg \min_{\mathbf{x}} \|\mathbf{b} - \mathbf{A}\mathbf{P}\mathbf{x}\|_2^2, \quad (6.29)$$

where  $\mathbf{P} \in \mathbb{R}^{m \times |I_k|}$  is an operator that chooses the  $|I_k|$  entries from the vector  $\mathbf{x}$  that are permitted to be non-zeros. This minimization is done using a sequence of  $K_0$  Conjugate-Gradient (CG) iterations, where each iteration multiplies by  $\mathbf{A}\mathbf{P}$  and its adjoint.

**Update of the Residual:** Having found the updated vector  $\mathbf{x}_k$ , we compute the residual to be  $\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{x}_k$ .

As said above, when the thresholding step chooses one entry at a time, this is exactly the OMP method. For the StOMP mode of operation with many entries chosen at each step, the above process should be iterative several times (the authors recommend 10 iterations, as each iteration can potentially add many elements to the support), leading to the desired sparse solution of the approximated linear system  $\mathbf{b} \approx \mathbf{A}\mathbf{x}$ . A shortcoming in StOMP is its tendency to over-populate the detected support of the solution. A more recent work by Needell and Tropp proposes an alter-

native algorithm termed CoSaMP, that includes a mechanism that enables removal of on-support elements based on their weak contribution.

StOMP comes with a thorough theoretical analysis that considers the case of random dictionaries. In such a case, the values in  $\mathbf{A}^T \mathbf{e}_i$  could be interpreted as a sparse noisy vector, where the noise can be reliably assumed to be white zero-mean Gaussian. Thus, a thresholding step that follows is nothing but a denoising algorithm. In such a scenario, the creators of StOMP show an interesting phase transition from success to failure.

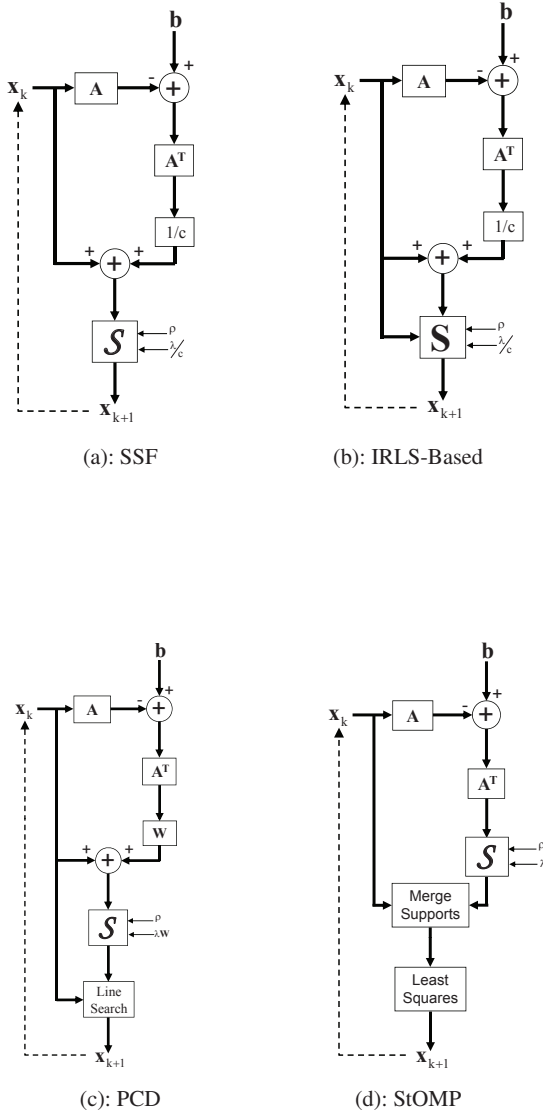
We add the following important observation: Because of the  $K_0$  CG iterations required, every iteration of StOMP parallels in complexity to  $K_0$  iterations of the SSF, IRLS-based, and the PCD algorithms. In that respect, this structure should remind the reader with the interior-point and the truncated Newton algorithms' inner solver.

### 6.3.6 Bottom Line – Iterative-Shrinkage Algorithms

Figure 6.4 presents the four *Iterative Shrinkage* algorithms described above as block diagrams. As can be seen, although different from each other in various ways, all rely on the same main computational stages, such as the term  $\mathbf{b} - \mathbf{A}\mathbf{x}$ , its back projection by multiplication by  $\mathbf{A}^T$ , and a thresholding/shrinkage step.

In this chapter we deliberately avoided discussing in depth the theoretical study of the algorithms presented here. Nevertheless, we should state the following few facts: the SSF and the PCD algorithms are guaranteed to converge to a local minimizer of the function in Equation (6.1). For  $\rho(x) = |x|$ , where this objective function becomes convex, a convergence to the global minimizer is guaranteed.

Among the four core algorithms described, the SSF, the IRLS-based, and the PCD are closer to each other. Which of these is faster? Recall that the constant  $c$  in SSF must satisfy  $c > \lambda_{\max}(\mathbf{A}^T \mathbf{A})$ , and in the IRLS method half this value. On the other hand, the PCD uses a scaling of the same term using the weight matrix  $\text{diag}(\mathbf{A}^T \mathbf{A})^{-1}$ . As an example, for a dictionary built as a union of  $N$  unitary matrices, the SSF requires  $c > N$ , implying a weight of  $1/N$  in Equation (6.14). In the PCD algorithm, the weight matrix is simply identity due to the normalized columns in  $\mathbf{A}$ . Thus, PCD gives  $O(N)$  times stronger effect to the term  $\mathbf{A}^T(\mathbf{b} - \mathbf{A}\mathbf{x}_k)$ , and because of that we expect it to perform better. We shall verify this assumption by conduct a series of comparative experiments. We should mention that the convergence speed is also expected to improve dramatically due to the accelerations, which are discussed next.



**Fig. 6.4** Block diagrams of the four obtained *Iterative-Shrinkage* algorithms. Note that while these four block-diagrams seems to be very similar, there are delicate differences between them in various locations.

## 6.4 Acceleration Using Line-Search and SESOP

All the above algorithms can be further accelerated in several ways. First, the idea of performing a line search, as described in the PCD, is relevant to the other algorithms just as well.<sup>7</sup> All that is required is to use Equations (6.14) or (6.20) to compute a temporary result  $\mathbf{x}_{temp}$ , and then define the solution as  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mu(\mathbf{x}_{temp} - \mathbf{x}_k)$ , optimizing  $f(\mathbf{x}_{k+1})$  with respect to the scalar  $\mu$ .

A second, and much more effective speed-up option is the deployment of the Sequential Subspace Optimization (SESOP) method. The original SESOP algorithm proposes to obtain the next iterate  $\mathbf{x}_{k+1}$  via optimization of function  $f$  over an affine subspace spanned by the set of  $q$  recent steps  $\{\mathbf{x}_{k-i} - \mathbf{x}_{k-j-1}\}_{i=0}^{q-1}$ , and the current gradient. This  $q+1$ -dimensional optimization task can be addressed using a Newton algorithm, as this problem is defined over a very low-dimension space. The main computational burden in this process is the need to multiply these directions by  $\mathbf{A}$ , but these  $q$  multiplications can be stored in previous iterations, thus enabling the SESOP speed-up algorithm with hardly any additional cost.

In the simple quadratic optimization case, SESOP algorithm with  $q \geq 1$  amounts to regular CG. The  $q = 0$  case is equivalent to a line-search. Preconditioning, like multiplication of the current gradient by inverse of the Hessian diagonal, may significantly accelerate the method.

In the context of iterative-shrinkage algorithms, one can use the direction  $(\mathbf{x}_{temp} - \mathbf{x}_k)$ , provided by SSF, IRLS-based, PCD, or even the StOMP algorithm, instead of the gradient used in the standard SESOP. We shall demonstrate this approach in the next section and show how effective it is.

## 6.5 Iterative-Shrinkage Algorithms: Tests

In this section we present a synthetic experiment aiming to study the behavior of the various algorithms described in this chapter. Our tests do not include StOMP, because of the conceptual differences mentioned earlier. The experiment we perform aims to minimize of the function

$$f(\mathbf{x}) = \frac{1}{2} \|\mathbf{b} - \mathbf{Ax}\|_2^2 + \lambda \|\mathbf{x}\|_1.$$

The ingredients forming this problem are:

- **The matrix  $\mathbf{A}$ :** We use a construction of the form  $\mathbf{A} = \mathbf{P}_{out} \mathbf{H} \mathbf{P}_{in}$ , where  $\mathbf{H}$  is the  $2^{16} \times 2^{16}$  Hadamard matrix,  $\mathbf{P}_{in}$  is a diagonal matrix (of the same size) with values on the main-diagonal growing linearly in the range  $[1, 5]$ , and  $\mathbf{P}_{out}$  is built as the identity matrix from which we choose a random subset of  $2^{14}$  rows. Thus, the overall matrix  $\mathbf{A}$  is of size  $2^{14} \times 2^{16}$ , that scales the entries of the input vector  $\mathbf{x}$ ,

<sup>7</sup> Even though StOMP has not been derived as a minimizer of the function in Equation (6.1), one can still apply this line-search and extract the best of each iteration.

rotates it in the  $\mathbb{R}^{2^{16}}$ -space, and then chooses a subset of  $n = 2^{14}$  of the resulting entries. The multiplication by  $\mathbf{A}$  or its adjoint can be done very fast, due to the availability of a Fast-Hadamard Transform (FHT) operation.

- **the reference solution  $\mathbf{x}_0$ :** We generate a sparse vector  $\mathbf{x}_0$  of length  $m = 2^{16}$  that contains 2000 non-zero in random locations, and with values drawn from the normal Gaussian distribution.
- **The vector  $\mathbf{b}$ :** The vector  $\mathbf{b}$  is computed by  $\mathbf{b} = \mathbf{A}\mathbf{x}_0 + \mathbf{v}$ , where  $\mathbf{v}$  is a random vector with iid entries, drawn from a Gaussian distribution with zero mean and standard-deviation  $\sigma = 0.2$ . This choice of variance leads to  $\|\mathbf{A}\mathbf{x}_0\|_2^2 / \|\mathbf{b} - \mathbf{A}\mathbf{x}_0\|_2^2 \approx 8$ , implying that the “signal” part  $\|\mathbf{A}\mathbf{x}_0\|_2^2$  is 8 times stronger than the noise added  $\|\mathbf{v}\|_2^2$ .
- **Choice of  $\lambda$ :** We choose  $\lambda$  empirically so as to lead to a residual that satisfies  $\frac{1}{2}\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2 \approx n\sigma^2$ . The value chosen (after several experiments) is  $\lambda = 1$ .

The algorithms we compare herein are the SSF, the IRLS-based algorithm, and the PCD. For the SSF, we consider its regular form, as given described in Figure 6.1 (with  $\mu = 1$ ), a line-search (LS) version (find  $\mu$ ), and an accelerated version with SESOP of dimension 5. Similarly, we test the IRLS method and a SESOP-5 accelerated version of it. As for the PCD, we test its regular mode that uses a line-search, and a SESOP-5 acceleration.

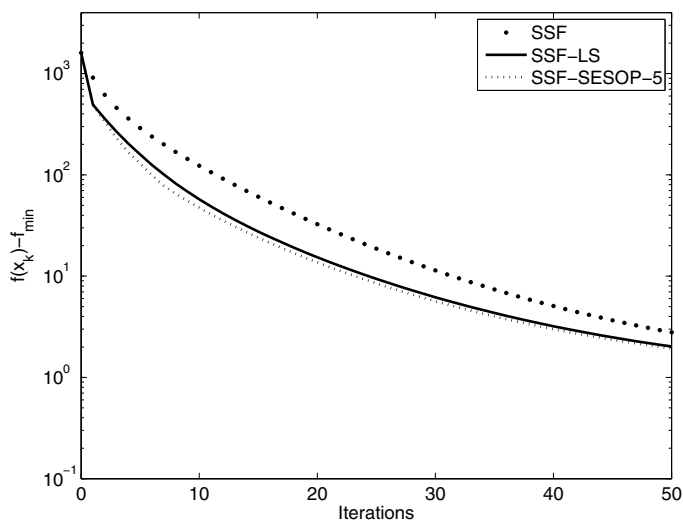
Throughout the simulations presented in this chapter, the inner optimization for the line-search and SESOP are done using 50 plain Steepest-Descent iterations. In Chapter 10 we shall return to the iterative-shrinkage algorithms in the context of a specific image processing application. There we consider a smoother version of  $\ell_1$  for the function  $\rho$ , thus enabling a Newton optimization.

Before turning to present the results obtained, we should discuss the constant  $c$  in the SSF and IRLS algorithms. This should be chosen as  $c > \|\mathbf{A}^T \mathbf{A}\|_2^2 = \|\mathbf{P}_{in}^T \mathbf{H}^T \mathbf{P}_{out}^T \mathbf{P}_{out} \mathbf{H} \mathbf{P}_{in}\|_2^2$ . If we omit  $\mathbf{P}_{out}$  (e.g., assume that  $n = 2^{16}$ ) then this gives  $c > 25$ , due to the maximal magnification by the main-diagonal of  $\mathbf{P}_{in}$ . The fact that  $\mathbf{P}_{out}$  extracts some of this energy by choosing a subset of  $n$  entries implies that the choice  $c = 25$  is correct and close to be the best possible value. Similarly, the squared-norms of the atoms in  $\mathbf{A}$ , which are used by PCD, are easily obtained by taking the entries from the main-diagonal of  $\mathbf{P}_{in}$  squaring them, and multiplying by  $n/m$ . These are the exact norms, due to the properties of  $\mathbf{H}$ .

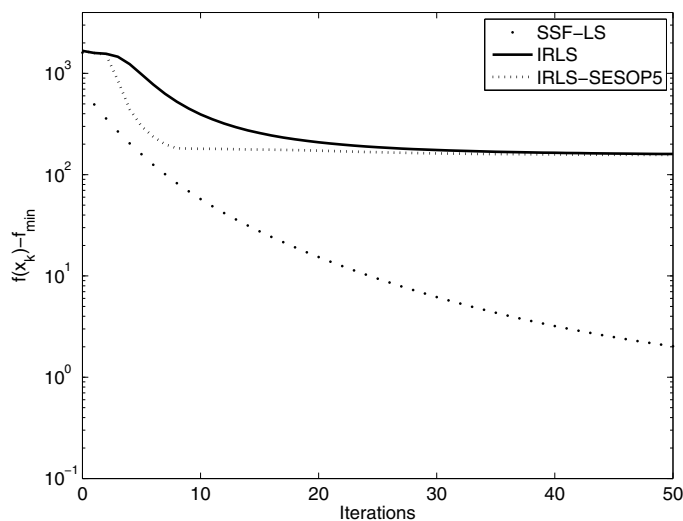
We now turn to present the results obtained. First, we show graphs presenting the value of  $f(\mathbf{x})$  as a function of the iterations per each method. These are shown in Figures 6.5-6.7, where the minimum value of  $f(\mathbf{x})$  (found by the best of these iterative methods) is subtracted. The first of these figures presents the various SSF algorithms (plain, LS, and SESOP-5). The second figures presents the IRLS results, compared to the SSF, and the third show the PCD methods compared to the SSF.

We also show the quality of the solution obtained, measured by the ratio  $\|\mathbf{x}_k - \mathbf{x}_0\|_2^2 / \|\mathbf{x}_0\|_2^2$ . These values should be below 1 and close to 0 to indicate that the relative error in the solution is small. These results are shown in Figures 6.8- 6.10.

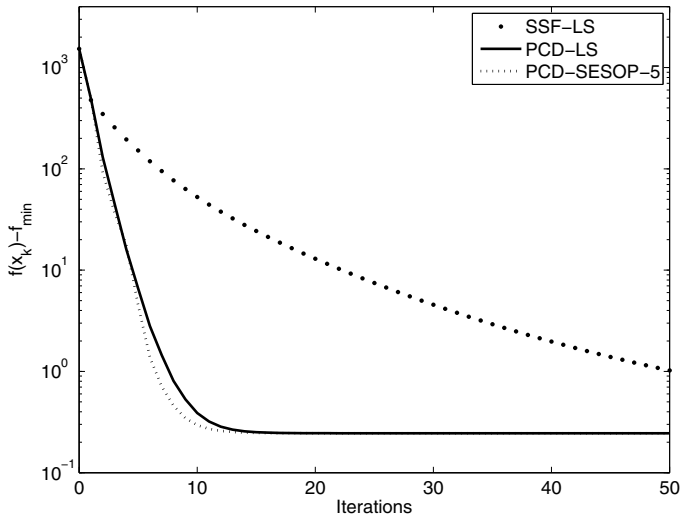
Several conclusions can be drawn from these results:



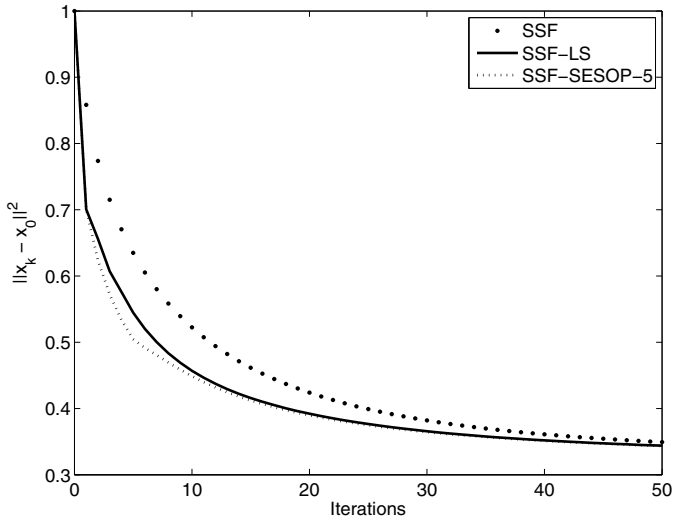
**Fig. 6.5** The objective function value as a function of the iterations for the various tested algorithms – the plain SSF, the SSF with a line-search, and the SSF with a SESOP-5 acceleration.



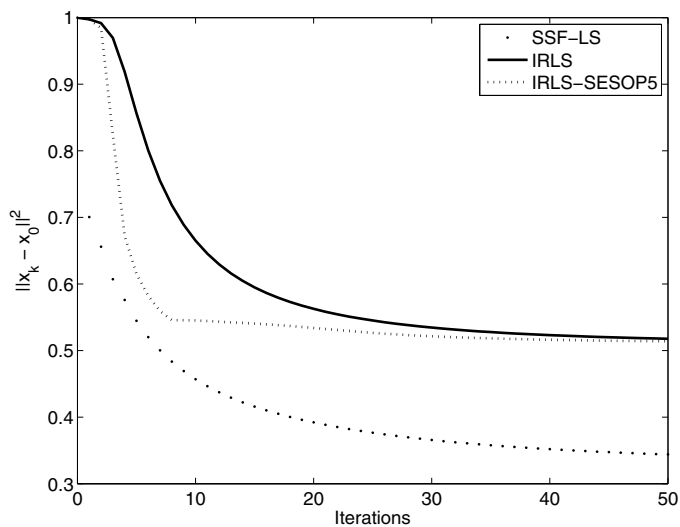
**Fig. 6.6** The objective function value as a function of the iterations for the various tested algorithms – the SSF with a line-search, the plain IRLS, and the a SESOP-5 acceleration of it.



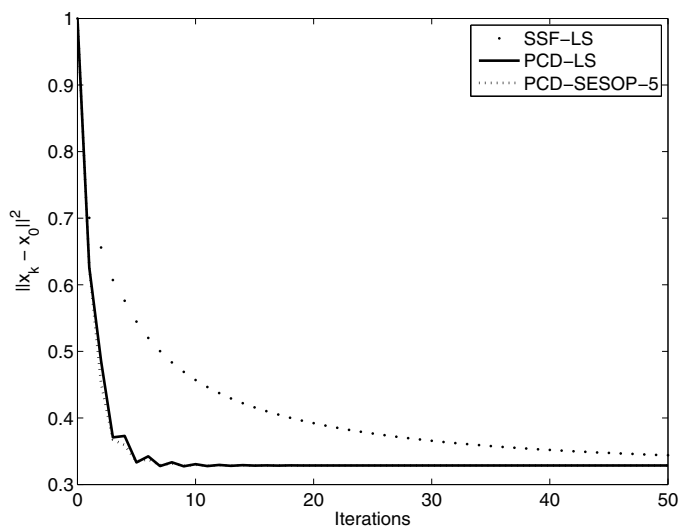
**Fig. 6.7** The objective function value as a function of the iterations for the various tested algorithms – the SSF with a line-search, the PCD (with a line-search), and a SESOP-5 accelerated version of it.



**Fig. 6.8** The solution's quality measure as a function of the iterations for the various tested algorithms – the plain SSF, the SSF with a line-search, and the SSF with a SESOP-5 acceleration.



**Fig. 6.9** The solution's quality measure as a function of the iterations for the various tested algorithms – the SSF with a line-search, the plain IRLS, and the a SESOP-5 acceleration of it.



**Fig. 6.10** The solution's quality measure as a function of the iterations for the various tested algorithms – the SSF with a line-search, the PCD (with a line-search), and a SESOP-5 accelerated version of it.



- There is a general agreement between the two measures shown, thus enabling us to concentrate on [Figures 6.5-6.7](#).
- The best performing method among the three (SSF, IRLS, and PCD) is the PCD. It provides a nearly perfect converged result after 15 iterations.
- A line-search added to these algorithms helps their convergence properties. Similarly, SESOP acceleration is indeed helping. The accelerated PCD requires 10 iterations to converge. A smaller effect is observed by the accelerated SSF.
- As expected, the IRLS-based algorithm is weaker, and tends to get stuck.

How good is the result obtained in terms of recovering the true sparse combination of atoms that generated  $\mathbf{b}$ ? [Figure 6.11](#) presents the original vector  $\mathbf{x}_0$  and the solution obtained by the PCD after 10 iterations. As said above, there are 2000 non-zeros in the original vector, i.e.,  $\|\mathbf{x}_0\|_0 = 2000$ . The PCD solution detects 2560 non-zeros, out of those 1039 are from the true support (the dominant ones, and mostly those far from the origin, due to the relative strength of the non-zero versus the noise).

A zoomed-in graph in the bottom of 6.11 shows that indeed most major peaks in  $\mathbf{x}_0$  are detected well, and attributed with a value that is somewhat smaller. This tendency is expected, since the objective function we use penalizes for the magnitude of the non-zeros. We can fix this by taking the found support from  $\mathbf{x}$ , denoted as  $\mathcal{S}$ , and solving a constrained Least-Squares problem of the form

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_2^2 \quad \text{subject to} \quad \text{Support}(\mathbf{x}) = \mathcal{S}, \quad (6.30)$$

which can be performed easily by an iterative procedure of the form

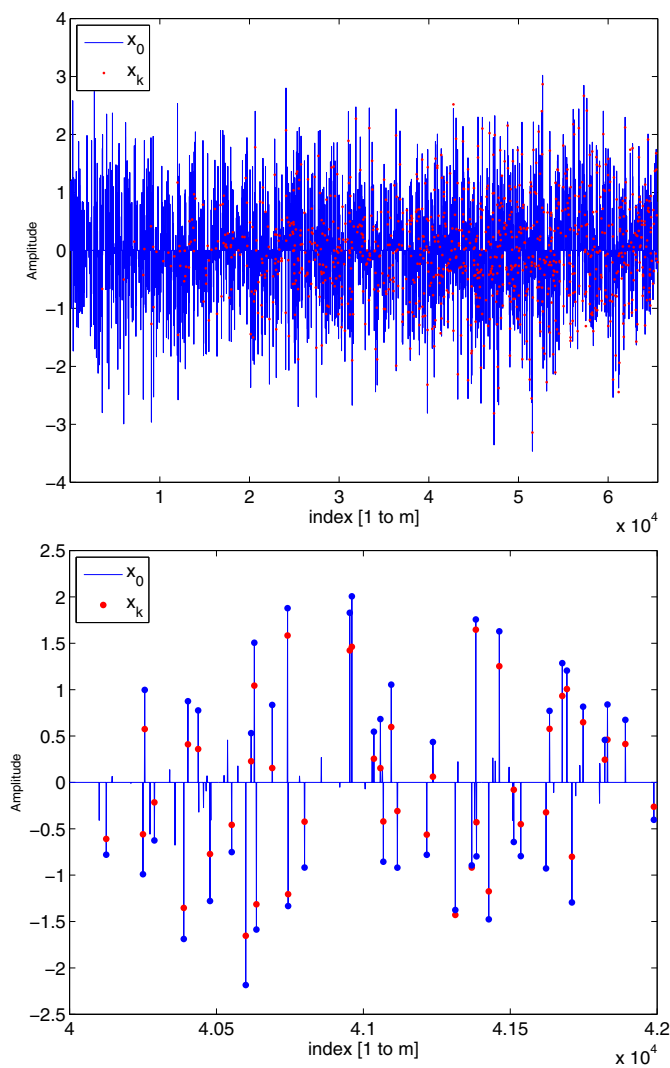
$$\mathbf{x}_{k+1} = \text{Proj}_{\mathcal{S}} \left[ \mathbf{x}_k - \mu \mathbf{A}^T (\mathbf{Ax} - \mathbf{b}) \right]. \quad (6.31)$$

The operator  $\text{Proj}_{\mathcal{S}}$  projects the vector on the support, essentially nulling the entries outside it. The result of this procedure are shown in [Figure 6.12](#), and as can be seen, the non-zero values are indeed getting closer to those of the original  $\mathbf{x}_0$ . Note that in terms of the measure  $\|\mathbf{x}_k - \mathbf{x}_0\|_2^2 / \|\mathbf{x}_0\|_2^2$ , whereas the original value obtained by the 10 PCD iterations is 0.36, after this adjustment it becomes 0.224, showing a substantial improvement.

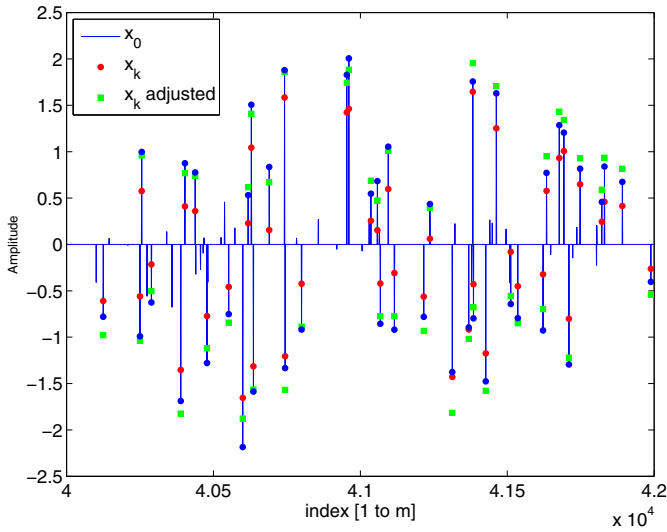
Just before leaving this chapter, we should draw the reader's attention to the fact that in Chapter 10 we revisit the topic of iterative-shrinkage algorithms, deployed to an image deblurring application. The experiments there include further comparisons between some of the algorithms mentioned here, and thus provide both a practical view for the need for these methods, along with extended comparison of SSF, PCD, and their accelerated versions.

## 6.6 Summary

In the field of sparse representation modeling, minimization of the function



**Fig. 6.11** The solution as obtained after 10 iterations of the PCD algorithm. The graphs show the original vector  $\mathbf{x}_0$  and the computed one  $\mathbf{x}_k$ , where the top shows the complete vectors, and the bottom graph presents a small slice of the horizontal support.



**Fig. 6.12** The same graph as the one shown in the bottom of Figure 6.11, with the least-squared adjusted solution over the support found.

$$f(\mathbf{x}) = \lambda \mathbf{1}^T \rho(\mathbf{x}) + \frac{1}{2} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2^2,$$

plays a vital role for various applications, and this will be clearly seen, as we move to later chapters that introduce the practical sides of this field. In this chapter we introduced a family of efficient optimization techniques to achieve this goal. These are the iterative-shrinkage algorithms, which employ iterations that consist of multiplication by the matrix  $\mathbf{A}$  and its adjoint, and a scalar shrinkage operation on the temporary solution found. These methods are extremely important when dealing with high-dimensional problems.

Beyond their relative efficiency in minimizing  $f(\mathbf{x})$ , an important feature of these methods is the close relationship they have with the more classical shrinkage algorithm that was developed originally for the unitary case. We have introduced several such algorithms, and presented a basic experiment to demonstrate their behavior. The research on these methods is ongoing, exploring new techniques (e.g. Bregman's and Nesterov's multi-stage algorithms), analyzing convergence behavior, and proposing applications that employ these in practice.

## Further Reading

1. S. Becker, J. Bobin and E. Candes, NESTA: A fast and accurate first-order method for sparse recovery, Preprint, 2009.

2. J. Bioucas-Dias, Bayesian wavelet-based image deconvolution: a GEM algorithm exploiting a class of heavy-tailed priors, *IEEE Trans. on Image processing*, 15(4):937–951, April 2006.
3. T. Blumensath and M.E. Davies, Iterative thresholding for sparse approximations, *Journal of Fourier Analysis and Applications*, 14(5):629–654, 2008.
4. J. Bobin, Y. Moudden, J.-L. Starck, and M. Elad, Morphological diversity and source separation, *IEEE Signal Processing Letters*, 13(7):409–412, July 2006.
5. J. Cai, S. Osher, and Z. Shen, Convergence of the linearized Bregman iteration for  $l_1$ -norm minimization, *Mathematics of Computation*, 78:2127–2136, 2009.
6. J. Cai, S. Osher, and Z. Shen, Linearized Bregman iterations for compressed sensing, *Mathematics of Computation*, 78:1515–1536, 2009.
7. J. Cai, S. Osher, and Z. Shen, Linearized Bregman iteration for frame based image deblurring, *SIAM Journal on Imaging Sciences*, 2(1):226–252, 2009.
8. P.L. Combettes and V.R. Wajs, Signal recovery by proximal forward-backward splitting, *Multiscale Modeling and Simulation*, 4(4):1168–1200, November 2005.
9. I. Daubechies, M. Defrise, and C. De-Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, *Communications on Pure and Applied Mathematics*, LVII:1413–1457, 2004.
10. D.L. Donoho, De-noising by soft thresholding, *IEEE Trans. on Information Theory*, 41(3):613–627, 1995.
11. D.L. Donoho and I.M. Johnstone, Ideal spatial adaptation by wavelet shrinkage, *Biometrika*, 81(3):425–455, 1994.
12. D.L. Donoho, I.M. Johnstone, G. Kerkyacharian, and D. Picard, Wavelet shrinkage - asymptopia, *Journal of the Royal Statistical Society Series B - Methodological*, 57(2):301–337, 1995.
13. D.L. Donoho and I.M. Johnstone, Minimax estimation via wavelet shrinkage, *Annals of Statistics*, 26(3):879–921, 1998.
14. D.L. Donoho, Y. Tsaig, I. Drori, and J.-L. Starck, Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit, *Technical Report*, Stanford University, 2006.
15. M. Elad, Why simple shrinkage is still relevant for redundant representations?, to appear in the *IEEE Trans. on Information Theory*.
16. M. Elad, B. Matalon, and M. Zibulevsky, Image denoising with shrinkage and redundant representations, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, NY, June 17–22, 2006.
17. M. Elad, B. Matalon, and M. Zibulevsky, On a Class of Optimization methods for Linear Least Squares with Non-Quadratic Regularization, to appear in *Applied and Computational Harmonic Analysis*.
18. M. Elad, J.-L. Starck, P. Querre, and D.L. Donoho, Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA), *Journal on Applied and Comp. Harmonic Analysis*, 19:340–358, 2005.
19. M.J. Fadili, J.-L. Starck, Sparse representation-based image deconvolution by iterative thresholding, *Astronomical Data Analysis ADA'06*, Marseille, France, September, 2006.

20. M.A. Figueiredo and R.D. Nowak, An EM algorithm for wavelet-based image restoration, *IEEE Trans. Image Processing*, 12(8):906–916, 2003.
21. M.A. Figueiredo, and R.D. Nowak, A bound optimization approach to wavelet-based image deconvolution, IEEE International Conference on Image Processing - ICIP 2005, Genoa, Italy, 2:782–785, September 2005.
22. M.A. Figueiredo, J.M. Bioucas-Dias, and R.D. Nowak, Majorization-minimization algorithms for wavelet-based image restoration, to appear in the *IEEE Trans. on Image Processing*.
23. D. Needell and J. A. Tropp, CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. to appear in *Appl. Comp. Harmonic Anal.*.
24. Y. Nesterov, Smooth minimization of non-smooth functions, *Math. Program., Serie A.*, 103:127–152, 2007.
25. S. Sardy, A.G. Bruce, and P. Tseng, Block coordinate relaxation methods for nonparametric signal denoising with wavelet dictionaries, *Journal of computational and Graphical Statistics*, 9:361–379, 2000.
26. J.-L. Starck, M. Elad, and D.L. Donoho, Image decomposition via the combination of sparse representations and a variational approach. *IEEE Trans. on Image Processing*, 14(10):1570–1582, 2005.
27. J.-L. Starck, F. Murtagh, and A. Bijaoui, Multiresolution support applied to image filtering and restoration, *Graphical Models and Image Processing*, 57:420–431, 1995.

## Chapter 7

# Towards Average Performance Analysis

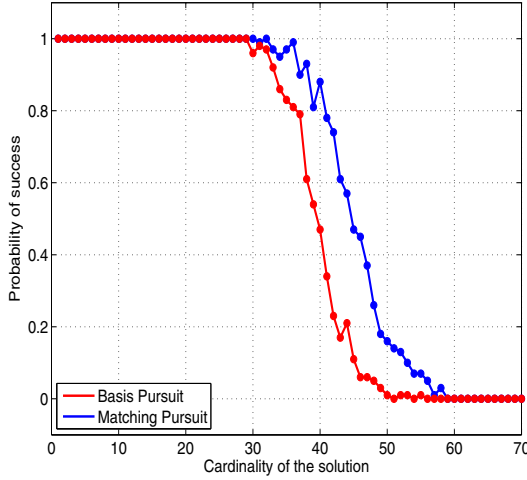
The analysis presented so far presents a simple but limited portrait of the ability of concrete algorithms to find sparse solutions and near-solutions. In this chapter we briefly point to the interesting and challenging research territory that lies beyond these worst-case results. We start with some simple simulations to motivate this discussion.

### 7.1 Empirical Evidence Revisited

We have presented experiments that compare pursuit algorithms, both in Chapter 3 and later in Chapter 5. We return to these here, with an emphasis on the success in recovering the support of the true solution.

Consider a random matrix  $\mathbf{A}$  of size  $100 \times 200$ , with entries independently drawn by random from a Gaussian distribution of zero mean and unit variance,  $\mathcal{N}(0, 1)$ . The *spark* of this matrix is 101 with probability 1, implying that every solution for the system  $\mathbf{Ax} = \mathbf{b}$  with 50 entries and below is necessarily the sparsest possible, and as such, it is the solution of  $(P_0)$ . By randomly generating such sufficiently sparse vectors  $\mathbf{x}_0$  (choosing the non-zero locations uniformly over the support in random, and their values from  $\mathcal{N}(0, 1)$ ), we generate vectors  $\mathbf{b}$ . This way, we know the sparsest solution to  $\mathbf{Ax}_0 = \mathbf{b}$ , and we are able to compare this to algorithmic results.

The graph presented in [Figure 7.1](#) shows the success rate for both OMP and BP (implemented by Matlab's Linear-Programming solver) in recovering the true (sparsest) solution. The success is measured by the computing  $\|\hat{\mathbf{x}} - \mathbf{x}_0\|_2^2 / \|\mathbf{x}_0\|_2^2$  and checking that is below a negligible value (in our experiments this is set to  $1e-5$ ), to indicate a perfect recovery of the original sparse vector  $\mathbf{x}_0$ . For each cardinality, 100 repetitions are conducted and their results averaged to give the probability of success. The value of the mutual-coherence of  $\mathbf{A}$  in this experiment is  $\mu(\mathbf{A}) = 0.424$ , so that only for cardinalities lower than  $(1 + 1/\mu(\mathbf{A}))/2 = 1.65$  pursuit methods are guaranteed to succeed. As we can see, both pursuit algorithms succeed in the



**Fig. 7.1** Probability of success of pursuit algorithms in the recovery of the sparsest solution of the linear system  $\mathbf{Ax} = \mathbf{b}$ . The results are shown as a function of the cardinality of the desired solution.

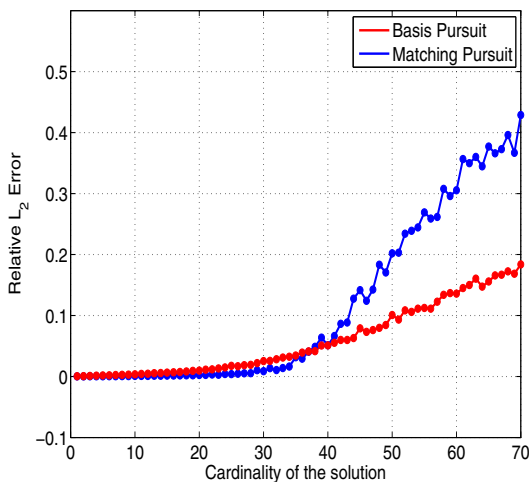
recovery of the sparsest solution for  $1 \leq \|\mathbf{x}_0\|_0 \leq 26$ , far beyond the coverage of the above-described theoretical bound. We can also see that the greedy algorithm (OMP) is performing somewhat better than the BP.

Turning to the noisy case (solution of  $(P_0^\epsilon)$ ), we use the same matrix  $\mathbf{A}$ , and generate a random vector  $\mathbf{x}_0$  with a pre-specified cardinality of non-zeros. It is normalized so that  $\|\mathbf{Ax}_0\|_2 = 1$ . We compute  $\mathbf{b} = \mathbf{Ax}_0 + \mathbf{e}$ , where  $\mathbf{e}$  is a random vector with pre-determined norm  $\|\mathbf{e}\|_2 = \epsilon = 0.1$ . Thus, the original vector  $\mathbf{x}_0$  is a feasible solution for  $(P_0^\epsilon)$ , and close to the optimal due to its sparsity. The theoretical study in Chapter 5 suggests a stable recovery of  $\mathbf{x}_0$  (within an error of order  $\epsilon$ ) for cardinalities smaller than  $(1 + 1/\mu(\mathbf{A}))/2$  (or even division by 4 for the BP case), and again these lead to overly pessimistic prediction of performance.

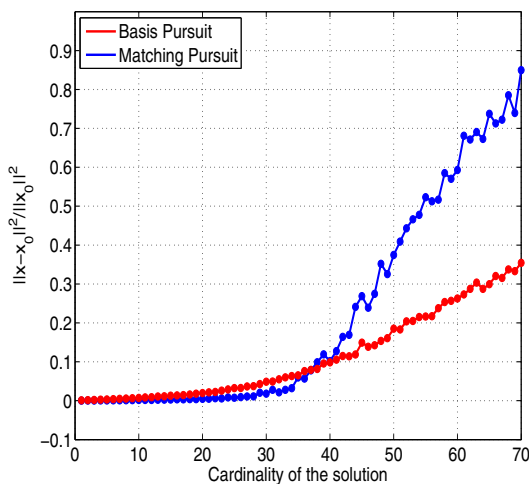
The graph presented in Figure 7.2 shows the relative error for OMP and BP (we use the LARS solver) for this test.<sup>1</sup> This relative error is defined as the ratio  $\|\hat{\mathbf{x}} - \mathbf{x}_0\|_2^2 / \epsilon^2$ , averaged over 100 experiments. In both tested algorithms, the solution is such that  $\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{b}\|_2 \leq \epsilon$ . As can be seen, both methods do very well for all cardinalities tested, showing strong tendency to stability, while the theoretical results set a much lower bound for stability to hold.

As the solution of  $(P_0^\epsilon)$  may be interpreted as noise cleaning, we also present in Figure 7.3 the averaged relative error  $\|\hat{\mathbf{x}} - \mathbf{x}_0\|_2^2 / \|\mathbf{x}_0\|_2^2$  obtained in these experiments. This ratio should be as small as possible, and a value of 1 is comparable to an

<sup>1</sup> The observant reader may detect the similarity between the results shown in this graph, and those that were given in Chapter 5 Figure 5.12. Note, however, that the experiment setting is somewhat different.



**Fig. 7.2** The tendency of pursuit algorithms to provide a stable recovery of the sparsest solution of the linear system  $\mathbf{Ax} = \mathbf{b}$  in the presence of noise.



**Fig. 7.3** The noise cleaning effect of pursuit algorithms, as a function of the cardinality of  $\mathbf{x}_0$ .

approximation of plain zeros for  $\hat{\mathbf{x}}$ . We see that the two methods tested exhibit good noise cleaning behavior, and as expected, it is of stronger effect for low-cardinalities in  $\mathbf{x}_0$ .

These simulations show that the obtained bounds do not tell the whole story. When working with matrices having specific properties, or with problems exhibit-



ing structured sparsity, we might find that the worst-case analysis gives very weak guarantees compared to what actually happens.

## 7.2 A Glimpse into Probabilistic Analysis

### 7.2.1 The Analysis Goals

Realizing that there is such a marked gap, several researchers developed in recent years (2006-2009) new results on the performance of pursuit algorithms, put forward using a probabilistic machinery. In this section we review some of these results and bring them without proofs. In the next section we shall concentrate on one such work by Schnass and Vandergeynst and bring its complete derivation. Their work studies the performance of the thresholding algorithm in the noiseless setting, and it illustrates the potential in adopting a probabilistic point of view when analyzing pursuit algorithms. The appeal in this specific work is the relative ease with which one can obtain stronger bounds, a fact that does not migrate well to other pursuit methods.

Before turning to describe specific attempts to alternative analysis, we should state our goal clearly. The ultimate objective would be a theoretical analysis that provides *the average performance* of pursuit algorithms, namely, the probability performance curves shown in the previous chapter's experiments. As this may be too difficult to obtain, we could relax our goals and be satisfied with probabilistic performance curves that bound the true ones from below, thus providing some guarantees for error-rates per a given cardinality.

Since we are mostly interested in the region of near-perfect performance, we could focus on the low-cardinalities zone in the above curves. Considering a success-rate of  $1 - \delta$  and beyond (higher) for an arbitrarily small value  $\delta$ , we may propose an analysis that aims to show that such error-rates are achievable by some cardinalities below an (hopefully high) upper-bound threshold, which lower-bounds the true point on the empirical curve. In this respect, earlier worst-case results correspond to such an analysis for  $\delta = 0$  (assuming that the results obtained are tight).

Such line of reasoning implies that we allow a small but controlled fraction of errors. Indeed, many of the existing methods that take this analysis route choose to specify their results in an asymptotic terminology with respect to the dimensions of the problem, aiming to show that  $\delta(n)$  is tending to zero, implying that encountering such failures becomes extremely rare. Another immediate and fundamental implication of this type of study is that the problem set defined by the triplet  $\{\mathbf{A}, \mathbf{b}, \epsilon\}$  must be modelled probabilistically, so as to enable any claim about probability of failure. Different works provide different modeling of this randomness, as we shall see below.

### 7.2.2 Two-Ortho Analysis by Candes & Romberg

A pioneering and the very first attempt reported on probabilistic analysis of pursuit algorithms is the work (developed during 2004, and published in 2006) by Emmanuel Candes and Justin Romberg. This work revisits the uniqueness and BP-equivalence results for the two-ortho noiseless case, as presented in Chapters 2 and 4. Their work provides a robustified analysis that permits a negligible and diminishing amount of outliers, and thus leads to stronger claims on both the uniqueness and equivalence fronts. This way, they avoid the worst-case limitations enforced in earlier studies.

As already mentioned above, in order to provide a probabilistic analysis of any sort, one must start with a probabilistic modeling of the problem. The work by Candes and Romberg considers a deterministic two-ortho matrix  $\mathbf{A} = [\Psi, \Phi]$  of size  $n \times 2n$ , multiplied by random sparse vectors  $\mathbf{x}$ . These sparse vectors are randomly generated by considering their two portions,  $\mathbf{x}_\Psi$  and  $\mathbf{x}_\Phi$ , each having a fixed and a-priori chosen cardinality  $N_\Psi$  and  $N_\Phi$ . All possible  $\binom{n}{N_\Psi}$  supports for  $\mathbf{x}_\Psi$  are considered as equably probable, and one of them is drawn at random. The non-zeros in this vector are iid random variables, drawn from an arbitrary continuous and circularly symmetric (for complex numbers) probability density function. The same process is repeated for the creation of  $\mathbf{x}_\Phi$ . The outcome of the multiplication is  $\mathbf{b} = \mathbf{Ax} = \Psi\mathbf{x}_\Psi + \Phi\mathbf{x}_\Phi$ , leading to the definition of the linear system  $\{\mathbf{A}, \mathbf{b}\}$  for which we are interested in the solution of  $(P_0)$ .

Having defined our problem suite, the first question addressed refers to the uniqueness of a sparse solutions of  $(P_0)$ . Allowing for a small and negligible number of outliers, what would be an upper bound on  $N_\Psi + N_\Phi$  so as to guarantee that such  $\mathbf{x}$  is the unique (sparsest) solver of  $(P_0)$ ? The second question considers a similar bound that guarantees a successful recovery of such sparse vectors  $\mathbf{x}$  by Basis-Pursuit, again allowing of a small amount of failures.

Recall that for both the uniqueness and the equivalence to hold true, previously stated results set bounds proportional to  $1/\mu(\mathbf{A})$ , which in turn is proportional to  $\sqrt{n}$  at best. The novelty in the work by Candes and Romberg is the establishment of new bounds, proportional to  $1/\mu(\mathbf{A})^2$  with some log factor (to the power of 6), and thus at best behave like  $n/\log^6 n$ . Those bounds guarantee the uniqueness of the sparsest solution, and the fact that it is recovered by the solution of  $(P_1)$ , and both are shown to be correct with high-confidence as  $n \rightarrow \infty$ . More specifically, the following two theorems are developed by Candes and Romberg.

**Theorem 7.1.** *Consider the linear system  $\mathbf{Ax} = \mathbf{b}$  with  $\mathbf{A} = [\Psi, \Phi]$  and  $\mathbf{b} = \mathbf{Ax}_0$ , where  $\mathbf{x}_0$  is drawn at random as described above with cardinalities  $N_\Psi$  and  $N_\Phi$  such that*

$$N_\Psi + N_\Phi \leq \frac{C'_\beta}{\mu(\mathbf{A})^2 \log^6 n}. \quad (7.1)$$

Then, with probability at least  $1 - O(n^{-\beta})$ , the vector  $\mathbf{x}_0$  is the solution of  $(P_0)$ , i.e., it is the sparsest possible solution to  $\mathbf{Ax} = \mathbf{b}$ .

**Theorem 7.2.** Consider the linear system  $\mathbf{Ax} = \mathbf{b}$  with  $\mathbf{A} = [\Psi, \Phi]$  and  $\mathbf{b} = \mathbf{Ax}_0$ , where  $\mathbf{x}_0$  is drawn at random as described above with cardinalities  $N_\Psi$  and  $N_\Phi$  such that

$$N_\Psi + N_\Phi \leq \frac{C''_\beta}{\mu(\mathbf{A})^2 \log^6 n}. \quad (7.2)$$

Then, with probability at least  $1 - O(n^{-\beta})$ , the vector  $\mathbf{x}_0$  is the solution of  $(P_1)$ , i.e., it is found exactly by the Basis-Pursuit.

The constants  $C'_\beta > C''_\beta > 0$  are not specified, but they are relatively small, and depend only on  $\beta$ , which in turn controls the accuracy of these claims. Both this, and the fact that these Theorems fail with probability proportional to  $n^{-\beta}$  imply that these new results become relevant and stronger than the worst-case ones for large enough values of  $n$ .

While we will not provide the proof for these two results (they are too much involved), we shall briefly explain one critical ingredient in their derivations. Denoting by  $\Phi_{N_\Psi}$  and  $\Psi_{N_\Phi}$  the sub-matrices that contain the columns over the chosen support, the proofs rely strongly on the analysis of the norm of the  $N_\Psi \times N_\Phi$  matrix  $\Phi_{N_\Psi}^T \Phi_{N_\Phi}$ , aiming to show that it is small enough. This should remind the reader of parallel attempts seen earlier to analyze a portion of the Gram matrix  $\mathbf{A}_s^T \mathbf{A}_s$  (where  $s$  stands for the restriction to the support), and claim that it is positive-definite for as large as possible  $|s|$ . Put differently, considering the matrix  $\mathbf{A}_s^T \mathbf{A}_s - \mathbf{I}$ , which removes the main-diagonal, our desire was to show that this matrix has a small spectral radius (and thus norm).

While our attempts in the worst-case analysis were tunnelled towards the desire to have small off-diagonal entries in this portion of the Gram matrix, leading our worst-case analysis to the definition of the mutual-coherence, handling such sub-matrices as a whole, and taking into account the entries' signs may lead to better bounds on the cardinalities that still guarantee positivity of  $\mathbf{A}_s^T \mathbf{A}_s$ . This is the very source from which Candes and Romberg draw their improved bound.

Interestingly, in a more recent work (2008), Tropp extends this treatment to general (non two-ortho) matrices, showing that such leading minors from the Gram matrix of size proportional to  $n$  are still guaranteed to behave well. This leads to theorems similar to the two mentioned above, but for general matrices. Tropp's results too obtain bounds that have the squared mutual-coherence in their denominator.

The fact that two theorems shown above (and Tropp's results) are relying on the mutual-coherence (even though it is squared) should be disturbing. While well-chosen matrices lead to  $O(n/\log n)$  bounds, other choices (e.g., the Discrete Cosine Transform (DCT) and wavelet as a two-ortho matrix, or redundant DCT as a general one) give poor coherence and thus poor overall bounds. Indeed, the mutual-coherence is a worst-case measure by its definition, and as such, should have been avoided in such an analysis. Nevertheless, the mutual-coherence appears quite often

in probabilistic analysis of pursuit algorithms, probably because it helps in simplifying complex proofs.

### 7.2.3 Probabilistic Uniqueness

A different result, presented by Elad, addresses only the uniqueness property, generalizing the spark-based result in Chapter 2. This work, too, assumes a probability density function on the sparse vectors  $\mathbf{x}$ , constructed from an iid set of entries with a fixed cardinality. While the worst-case analysis suggests that only below  $\text{spark}(\mathbf{A})/2$  uniqueness is guaranteed, this work shows that uniqueness can be still claimed with high probability for cardinalities beyond this bound.

The key property used for the construction of this result is the fact that if a random representation  $\mathbf{x}$  is obtained with  $s = \|\mathbf{x}\|_0 < \text{spark}(\mathbf{A})$  non-zeros, it implies that the corresponding vector  $\mathbf{b} = \mathbf{A}\mathbf{x}$  resides in a non-degenerate  $s$ -dimensional space (since the columns from  $\mathbf{A}$  used are necessarily linearly-independent). Any competing sparser representation  $\mathbf{x}_t$  must lead to vectors  $\mathbf{A}\mathbf{x}_t$  that reside in  $s - 1$ -dimensional space and below, and while there are many such constructions, their number is finite. Thus, their overlap with the original  $s$ -dimensional space is necessarily zero, and therefore the original representation is unique with probability 1.

Surprisingly, even for representations with  $\text{spark}(\mathbf{A}) \leq s = \|\mathbf{x}\|_0 < n$ , some sort of uniqueness can be claimed, with high-probability. In order to build such a result, this work proposes to characterize the matrix  $\mathbf{A}$  in a way that extends the spark, forming the *signature* of the matrix. The signature of  $\mathbf{A}$  is a function  $\text{Sig}_{\mathbf{A}}(s)$ , counting the relative<sup>2</sup> number of supports of cardinality  $s$  that are linearly-dependent. Whereas the spark “thinks” worst-case, the signature gets the more general picture by gathering all the subsets of columns from  $\mathbf{A}$  that are linearly-dependent.

Using the (impossible to get!) signature of  $\mathbf{A}$ , the probability for the existence of a sparser solution is shown to be  $1 - \text{Sig}_{\mathbf{A}}(s)$ . This is easy to derive, since by considering all possible supports of cardinality  $s$ ,  $\text{Sig}_{\mathbf{A}}(s)$  of which are linearly-dependent, and are thus reducible, leading to sparser representations. As for the rest, they refer to linearly-independent groups of columns, for which alternative sparser representations cannot be found, leaning on the same rational we have exercised earlier.

### 7.2.4 Donoho’s Analysis

Another work that considers the average performance of the BP is reported by Donoho (2006). Whereas the work by Candes and Romberg assumes a fixed (and

---

<sup>2</sup> normalized with respect to the number of such supports,  $\binom{n}{s}$ .

structured!) matrix  $\mathbf{A}$ , the randomness being due to  $\mathbf{b}$  (via the randomization of the solution  $\mathbf{x}$ ), the analysis provided by Donoho considers  $\mathbf{A}$  as a random matrix, with columns chosen uniformly on the unit-sphere. Despite this major change in point of view, the results obtained are similar in flavor, suggesting that both uniqueness and equivalence are guaranteed with high probability for  $O(n)$  non-zeros in  $\mathbf{x}$ . Here too the results hold asymptotically with growing dimension  $n$  and a diminishing failure probability. Followup results by various authors extend the above results to the approximate case, handling the relation between the solutions of  $(P_0^\epsilon)$  and  $(P_1^\epsilon)$ , thereby establishing a wider zone of guaranteed stability.

### 7.2.5 Summary

Other results that exploit the probabilistic view aimed at getting stronger bounds exist, see e.g. recent work by Candes and Tao, Tropp, DeVore and Cohen, Donoho and Tanner, Shtok and Elad, and others. While all these attempts aim to show the same effect – an equivalence (or stability in case of added noise) of pursuit algorithms for  $O(n)$  non-zeros in  $\mathbf{x}$  – their treatment varies substantially, their characterization of the matrix  $\mathbf{A}$  may differ, and so do the assumptions these works rely on. At this point, the literature on this topic is growing so rapidly that it is difficult to do any justice at all to this field, its achievements, and results. Adding to the confusion in this arena is the fact that many of the newly introduced results are tailored for compressed-sensing, where  $\mathbf{A}$  constitutes a set of random projections. The results obtained are somewhat relevant, with partial ties to the problem posed here.

## 7.3 Average Performance of Thresholding

### 7.3.1 Preliminaries

We have seen that among the various greedy techniques, the thresholding is the simplest possible one, as it derives the solution in one projection step, by calculating  $|\mathbf{A}^T \mathbf{b}|$  and choosing its largest entries till the desired representation error is met. In Chapters 4 and 5 we presented the worst-case analysis of this algorithm, and now we turn to analyze its behavior under stochastic settings. The simplicity of this algorithm makes such analysis relatively easy, compared to the more involved study required in the other pursuit methods. The analysis we provide below follows closely with the work of Schnass and Vandergheynst, and the worst-case analysis that was presented in Chapter 4 for the noiseless case, which is the case considered here.

We assume that  $\mathbf{b}$  is constructed as  $\mathbf{b} = \mathbf{A}\mathbf{x} = \sum_{t \in S} x_t \mathbf{a}_t$ , where the non-zeros of  $\mathbf{x}$  are chosen as arbitrary positive values in the range  $[|x_{\min}|, |x_{\max}|]$  (where  $|x_{\min}|$  is

strictly positive), multiplied by iid Rademacher random variables  $\epsilon_t$  assuming values  $\pm 1$  with equal probabilities.  $\mathcal{S}$  is an arbitrary support of cardinality  $k$ .

In the analysis that follows, we shall use the following large-deviation inequality, true for any  $T > 0$ , an arbitrary sequence of iid Rademacher random variables  $\{\epsilon_t\}_t$ , and an arbitrary vector  $\mathbf{v}$ :

$$P\left(\left|\sum_t \epsilon_t v_t\right| \geq T\right) \leq 2 \exp\left\{-\frac{T^2}{32\|\mathbf{v}\|_2^2}\right\}. \quad (7.3)$$

This inequality is given in the book by Ledoux and Talagrand (Chapter 5, Section 4). As we shall see next, this inequality plays a vital role in the development of the probabilistic bound for the thresholding algorithm.

### 7.3.2 The Analysis

As we have already shown in Chapter 4, the probability of failure of the thresholding algorithm is given by

$$P(\text{Failure}) = P\left(\min_{i \in \mathcal{S}} |\mathbf{a}_i^T \mathbf{b}| \leq \max_{j \notin \mathcal{S}} |\mathbf{a}_j^T \mathbf{b}|\right). \quad (7.4)$$

Choosing an arbitrary threshold value  $T$ , the above probability can be bounded from above by

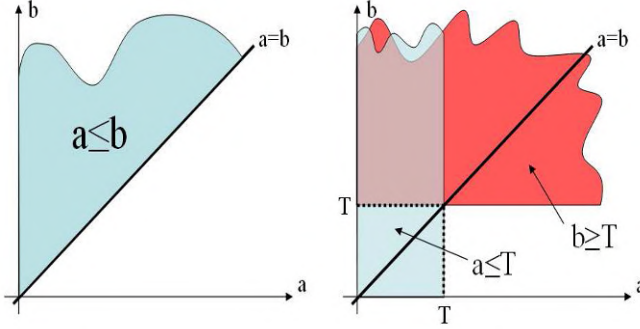
$$P(\text{Failure}) \leq P\left(\min_{i \in \mathcal{S}} |\mathbf{a}_i^T \mathbf{b}| \leq T\right) + P\left(\max_{j \notin \mathcal{S}} |\mathbf{a}_j^T \mathbf{b}| \geq T\right). \quad (7.5)$$

This step relies on the following rationale: Considering two non-negative random values  $a$  and  $b$  we have that

$$\begin{aligned} P(a \leq b) &= \int_{t=0}^{\infty} P(a = t)P(b \geq t)dt \\ &= \int_{t=0}^T P(a = t)P(b \geq t)dt + \int_{t=T}^{\infty} P(a = t)P(b \geq t)dt. \end{aligned}$$

In the first integral (that sums over the range  $[0, T]$ ) we use the relation  $P(b \geq t) \leq 1$ . Similarly, in the second integral we use the fact that  $P(b \geq t) \leq P(b \geq T)$  since  $t \geq T$ . This leads to

$$\begin{aligned} P(a \leq b) &\leq \int_{t=0}^T P(a = t)dt + \int_{t=T}^{\infty} P(a = t)P(b \geq T)dt \\ &= P(a \leq T) + P(b \geq T) \cdot \int_{t=T}^{\infty} P(a = t)dt \\ &= P(a \leq T) + P(b \geq T)P(a \geq T) \leq P(a \leq T) + P(b \geq T). \end{aligned}$$



**Fig. 7.4** A graphic explanation for the inequality  $P(a \leq b) \leq P(a \leq T) + P(b \geq T)$ . Considering the joint probability  $P(a, b)$ , the left graph presents the area to integrate over this joint probability, in order to obtain  $P(a \leq b)$ . The right graph shows the two terms,  $P(a \leq T)$  and  $P(b \geq T)$ , and clearly, their sum covers the previous area and beyond.

This is exactly the step taken in Equation (7.5). [Figure 7.4](#) provides an alternative graphic explanation for the inequality  $P(a \leq b) \leq P(a \leq T) + P(b \geq T)$ .

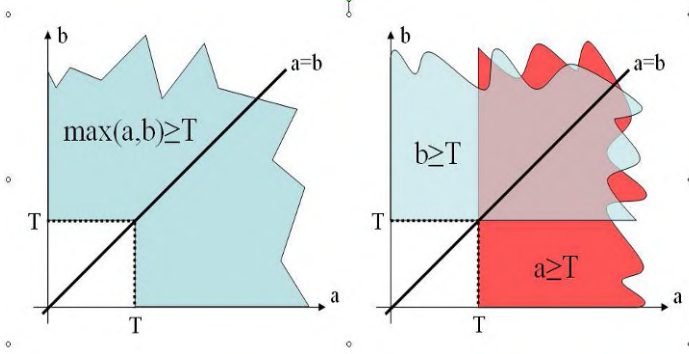
Considering the first probability in the right-hand-side, it is increased if we decrease the term  $\min_{i \in S} |\mathbf{a}_i^T \mathbf{b}|$ . Similarly, an increase of  $\max_{j \notin S} |\mathbf{a}_j^T \mathbf{b}|$  in the second term increases its probability to be above  $T$ . Thus, we follow (some of) the same steps as done for the worst-case analysis in simplifying these expressions. We start with the first term in Equation (7.5), plug the expression  $\mathbf{b} = \sum_{t \in S} x_t \mathbf{a}_t$ , and exploit the fact that  $\|\mathbf{a}_i\|_2 = 1$ , obtaining

$$\min_{i \in S} |\mathbf{a}_i^T \mathbf{b}| = \min_{i \in S} \left| \sum_{t \in S} x_t \mathbf{a}_i^T \mathbf{a}_t \right| = \min_{i \in S} \left| x_i + \sum_{t \in S \setminus i} x_t \mathbf{a}_i^T \mathbf{a}_t \right|. \quad (7.6)$$

Using the relation  $|a + b| \geq |a| - |b|$ , the above can be lower-bounded by

$$\begin{aligned} \min_{i \in S} \left| x_i + \sum_{t \in S \setminus i} x_t \mathbf{a}_i^T \mathbf{a}_t \right| &\geq \min_{i \in S} \left\{ |x_i| - \left| \sum_{t \in S \setminus i} x_t \mathbf{a}_i^T \mathbf{a}_t \right| \right\} \\ &\geq |x_{\min}| - \max_{i \in S} \left\{ \left| \sum_{t \in S \setminus i} x_t \mathbf{a}_i^T \mathbf{a}_t \right| \right\} \\ &\geq |x_{\min}| - \max_{i \in S} \left\{ \left| \sum_{t \in S \setminus i} \epsilon_t |x_t| \mathbf{a}_i^T \mathbf{a}_t \right| \right\}. \end{aligned} \quad (7.7)$$

Thus, returning to Equation (7.5), we get



**Fig. 7.5** A graphic explanation for the inequality  $P(\max(a, b)) \leq P(a \geq T) + P(b \geq T)$ . Considering the joint probability  $P(a, b)$ , the left graph presents the area to integrate over this joint probability, in order to obtain  $P(\max(a, b))$ . The right graph shows the two terms,  $P(a \geq T)$  and  $P(b \geq T)$ , and clearly, their some covers the previous area.

$$\begin{aligned}
 P\left(\min_{i \in S} |\mathbf{a}_i^T \mathbf{b}| \leq T\right) & \quad (7.8) \\
 & \leq P\left(|x_{\min}| - \max_{i \in S} \left\{ \left| \sum_{t \in S \setminus i} \epsilon_t |x_t| \mathbf{a}_i^T \mathbf{a}_t \right| \right\} \leq T\right) \\
 & = P\left(\max_{i \in S} \left\{ \left| \sum_{t \in S \setminus i} \epsilon_t |x_t| \mathbf{a}_i^T \mathbf{a}_t \right| \right\} \geq |x_{\min}| - T\right).
 \end{aligned}$$

Using the rule  $P(\max(a, b) \geq T) \leq P(a \geq T) + P(b \geq T)$  (demonstrated in Figure 7.5), we get

$$P\left(\min_{i \in S} |\mathbf{a}_i^T \mathbf{b}| \leq T\right) \leq \sum_{i \in S} P\left(\left| \sum_{t \in S \setminus i} \epsilon_t |x_t| \mathbf{a}_i^T \mathbf{a}_t \right| \geq |x_{\min}| - T\right). \quad (7.9)$$

Plugging now the large-deviation inequality presented in Equation (7.3) we finally obtain

$$\begin{aligned}
 P\left(\min_{i \in S} |\mathbf{a}_i^T \mathbf{b}| \leq T\right) & \leq \sum_{i \in S} P\left(\left| \sum_{t \in S \setminus i} \epsilon_t |x_t| \mathbf{a}_i^T \mathbf{a}_t \right| \geq |x_{\min}| - T\right) \\
 & \leq 2 \sum_{i \in S} \exp\left\{-\frac{(|x_{\min}| - T)^2}{32 \sum_{t \in S \setminus i} |x_t|^2 (\mathbf{a}_i^T \mathbf{a}_t)^2}\right\} \\
 & \leq 2|S| \exp\left\{-\frac{(|x_{\min}| - T)^2}{32 |x_{\max}|^2 \mu_2 (|S| - 1)}\right\}
 \end{aligned} \quad (7.10)$$



$$\leq 2|S| \exp \left\{ -\frac{(|x_{\min}| - T)^2}{32|x_{\max}|^2\mu_2(|S|)} \right\}.$$

where we have defined

$$\mu_2(k) = \max_{|S|=k} \max_{i \notin S} \sum_{t \in S} (\mathbf{a}_i^T \mathbf{a}_t)^2. \quad (7.11)$$

Note that we obviously have  $\mu_2(k) \leq k\mu(\mathbf{A})^2$ .

Turning to the second probability term in Equation (7.5) and following similar steps, we get

$$\begin{aligned} P\left(\max_{j \notin S} |\mathbf{a}_j^T \mathbf{b}| \geq T\right) &= P\left(\max_{j \notin S} \left| \sum_{t \in S} \epsilon_t |x_t| \mathbf{a}_j^T \mathbf{a}_t \right| \geq T\right) \\ &\leq \sum_{j \notin S} P\left(\left| \sum_{t \in S} \epsilon_t |x_t| \mathbf{a}_j^T \mathbf{a}_t \right| \geq T\right) \\ &\leq 2 \sum_{j \notin S} \exp \left\{ -\frac{T^2}{32 \sum_{t \in S} |x_t|^2 (\mathbf{a}_j^T \mathbf{a}_t)^2} \right\} \\ &\leq 2(m - |S|) \exp \left\{ -\frac{T^2}{32|x_{\max}|^2\mu_2(|S|)} \right\}. \end{aligned} \quad (7.12)$$

Returning to Equation (7.5), we have that the probability of a failure is bounded by

$$\begin{aligned} P(\text{Failure}) &\leq 2|S| \exp \left\{ -\frac{(|x_{\min}| - T)^2}{32|x_{\max}|^2\mu_2(|S|)} \right\} \\ &\quad + 2(m - |S|) \exp \left\{ -\frac{T^2}{32|x_{\max}|^2\mu_2(|S|)} \right\}. \end{aligned} \quad (7.13)$$

The above bound is a function of  $T$  and it should be minimized with respect to  $T$  so as to lead to the most tight upper-bound. The optimal value of  $T$  is hard to obtain in closed form, and therefore we simply assign  $T = |x_{\min}|/2$  leading to

$$P(\text{Failure}) \leq 2m \exp \left\{ -\left(\frac{x_{\min}}{x_{\max}}\right)^2 \cdot \frac{1}{128\mu_2(|S|)} \right\}. \quad (7.14)$$

### 7.3.3 Discussion

Let us explore the implications of this result. Consider the case where  $\mathbf{A}$  is of size  $n \times \rho n$  ( $\rho > 1$  is fixed), and we use the thresholding algorithm to recover the sparsest solution of the linear system  $\mathbf{A}\mathbf{x} = \mathbf{b}$ . We use the relation  $\mu_2(k) \leq k\mu(\mathbf{A})^2$ , as mentioned above, and we further assume that  $\mu(\mathbf{A}) \propto 1/\sqrt{n}$  (as for Grassmannian

Frames or some unitary amalgams). We shall denote  $r = \frac{|x_{\min}|}{|x_{\max}|}$ , so as to simplify the expressions obtained. Thus, the probability of failure of the thresholding algorithm in recovering a solution of cardinality  $k$  becomes under these assumptions

$$P(\text{Failure}) \leq 2\rho n \cdot \exp \left\{ -\frac{r^2 n}{128k} \right\}. \quad (7.15)$$

Thus, with cardinalities of the form  $k = cn / \log n$ , this probability becomes

$$P(\text{Failure}) \leq 2\rho n \cdot \exp \left\{ -\frac{r^2}{128c} \log n \right\} = 2\rho \cdot n^{1 - \frac{r^2}{128c}}. \quad (7.16)$$

When choosing  $c < r^2/128$ , this probability becomes arbitrarily small as  $n \rightarrow \infty$ . This should be contrasted with the worst-case analysis that guarantees recovery of  $k \propto \sqrt{n}$  cardinalities.

For the more general case, where the relation  $\mu(\mathbf{A}) \propto 1/\sqrt{n}$  is not true, sparse solutions are recoverable if their cardinality is  $k = c/(\mu(\mathbf{A})^2 \log n)$ , following the same rationale, and using the same condition on  $c$ .

We could improve the above analysis of the thresholding algorithm in several ways, and this is very likely to lead to more optimistic constants in the above claims. One such improvement is a replacement of the term  $\mu_2(k)$  by a smaller value. The definition given in Equation (7.11) seeks the worst sum of squares of  $k$  inner-products  $\mathbf{a}_i^T \mathbf{a}_t$ , and as such, it necessarily leads to a too pessimistic value. Suppose we have an evaluation of the probability density function of such sums over all  $\binom{m}{k}$  supports. Rather than choosing the maximal value, we can discard of the tail of this distribution (thereby assuming that on those cases of volume  $\delta$  we fail), and use a smaller value in the probability bound. This probability should then be multiplied by  $1 - \delta$  to reflect the added failures.

The obtained result is one in line of a sequence of similar results about the successful recovery of sparse solutions with cardinalities that are near-linear with respect to the dimension  $n$ . This is the simplest to obtain, due to the simplicity of the underlying algorithm. Nevertheless, it reflects well the flavor common to other studies. We summarize this analysis with a Theorem that states the essence of our findings:

**Theorem 7.3. (Probable Equivalence – Thresholding Algorithm):** *For a system of linear equations  $\mathbf{A}\mathbf{x} = \mathbf{b}$  ( $\mathbf{A} \in \mathbb{R}^{n \times \rho n}$  full-rank with  $\rho > 1$ ), if a solution  $\mathbf{x}$  (with minimal non-zero value  $|x_{\min}|$  and maximal one  $|x_{\max}|$ ) exists obeying*

$$\|\mathbf{x}\|_0 < \frac{1}{128} \left( \frac{x_{\min}}{x_{\max}} \right)^2 \cdot \frac{1}{\mu(\mathbf{A})^2 \log n}, \quad (7.17)$$

*then the Thresholding algorithm run with threshold parameter  $\epsilon_0 = 0$  is likely to find it exactly, with probability of failure vanishing to zero for  $n \rightarrow \infty$ .*

## 7.4 Summary

Swinging between the initial excitement about results of the form shown in Chapter 3, and the later disappointment from their tendency to be overly pessimistic, researchers are still interested in the core question of theoretical bounds that predict the success of pursuit algorithms, in a way that matches empirical evidence. In this chapter we have shown some of these attempts, and gave a detailed description of one of those results that corresponds to the thresholding algorithm.

Has the main question been answered already? Unfortunately the answer to this question is negative. Despite the above results, we still do not possess a good and constructive answer for a given matrix  $\mathbf{A}$  of fixed size, which will predict reliably the performance as seen in experiments. In that respect, it would be a mistake to keep using measures such as the mutual-coherence or the RIP in forming such bounds, since this immediately implies a worst-case flavor.

## Further Reading

1. E.J. Candès and J. Romberg, Practical signal recovery from random projections, in *Wavelet XI, Proc. SPIE Conf.* 5914, 2005.
2. E.J. Candès, J. Romberg, and T. Tao, Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information, *IEEE Trans. on Information Theory*, 52(2):489–509, 2006.
3. E. Candès, J. Romberg, and T. Tao, Quantitative robust uncertainty principles and optimally sparse decompositions, to appear in *Foundations of Computational Mathematics*.
4. E. Candès, J. Romberg, and T. Tao, Stable signal recovery from incomplete and inaccurate measurements, to appear in *Communications on Pure and Applied Mathematics*.
5. E.J. Candès and T. Tao, Decoding by linear programming, *IEEE Trans. on Information Theory*, 51(12):4203–4215, December 2005.
6. D.L. Donoho, For most large underdetermined systems of linear equations, the minimal  $\ell_1$ -norm solution is also the sparsest solution, *Communications on Pure and Applied Mathematics*, 59(6):797–829, June 2006.
7. D.L. Donoho, For most large underdetermined systems of linear equations, the minimal  $\ell_1$ -norm near-solution approximates the sparsest near-solution, *Communications on Pure and Applied Mathematics*, 59(7):907–934, July 2006.
8. D.L. Donoho and J. Tanner, Neighborliness of randomly-projected simplices in high dimensions, *Proceedings of the National Academy of Sciences*, 102(27):9452–9457, March 2005.
9. D.L. Donoho and J. Tanner, Counting faces of randomly-projected polytopes when the projection radically lowers dimension, *Journal of the AMS*, 22(1):1–53, 2009.

10. M. Elad, Sparse representations are most likely to be the sparsest possible, *EURASIP Journal on Applied Signal Processing*, Paper No. 96247, 2006.
11. B. Kashin, The widths of certain finite-dimensional sets and classes of smooth functions, *Izv. Akad. Nauk SSSR Ser. Mat.*, 41, pp. 334–351, 1977.
12. S. Mendelson, A. Pajor, , and N. Tomczak-Jaegermann, Uniform uncertainty principle for Bernoulli and subgaussian ensembles, to appear in *Constructive Approximation*, 28(3):277–289, December 2008.
13. S. Mendelson, A. Pajor, and N. Tomczak-Jaegermann, Reconstruction and subgaussian processes, *Comptes Rendus Mathematique*, 340(12):885–888, 2005.
14. K. Schnass and P. Vandergheynst, Average performance analysis for thresholding, *IEEE Signal Processing Letters*, 14(11):828–831, November 2007.
15. S. Szarek, Condition number of random matrices, *J. Complexity*, 7, pp. 131–149, 1991.
16. S. Szarek, Spaces with large distance to  $\ell^\infty$  and random matrices, *Amer. J. Math.*, 112:899–942, 1990.



## Chapter 8

# The Dantzig-Selector Algorithm

In this chapter we present an appealing and surprising alternative pursuit algorithm for sparse approximation. This algorithm is competitive with the Basis Pursuit Denoising (BPDN) and the OMP. This algorithm was proposed in 2007 by Candes and Tao, and termed *Dantzig-Selector* (DS). The name chosen pays tribute to George Dantzig, the father of the simplex algorithm that solves Linear Programming (LP) problems. The connection to LP will become evident shortly.

As we show in this chapter, the elegance in the work of Candes and Tao does not stop in the proposal for an effective pursuit algorithm. Their work is mainly focused on providing theoretical guarantees for this method's affectivity in accurately recovering sparse solutions. In this chapter we bring one such result taken from their paper.

### 8.1 Dantzig-Selector versus Basis-Pursuit

In Chapter 5 we discussed the following problem: a sparse vector  $\mathbf{x}_0$  is multiplied by a matrix  $\mathbf{A}$  of size  $n \times m$ , where in particular  $n$  could be smaller than  $m$ . We shall assume hereafter that the columns of  $\mathbf{A}$  are  $\ell_2$ -normalized. The outcome is contaminated by a white Gaussian additive noise<sup>1</sup>  $\mathbf{e}$ , with iid entries  $e_i \sim \mathcal{N}(0, \sigma^2)$ , and we obtain  $\mathbf{b} = \mathbf{A}\mathbf{x}_0 + \mathbf{e}$ . Our goal is to estimate reliably  $\mathbf{x}_0$ . Using the ideas presented in Chapter 5, it would be natural to propose the estimator of the form

$$(P_0^\epsilon) \quad \hat{\mathbf{x}}_0^\epsilon = \arg \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 \leq \epsilon, \quad (8.1)$$

with  $\epsilon$  chosen proportional to  $\sqrt{n}\sigma$ . We referred to this as  $(P_0^\epsilon)$ , and as this problem is too complex to solve in general, an option we discussed is a replacement of the  $\ell_0$

<sup>1</sup> In Chapter 5 we assumed that the “noise” is a deterministic vector of bounded energy  $\|\mathbf{e}\|_2 = \epsilon$ . Such an assumption makes this distortion an adversary noise that may assume any form so as to work against the reconstruction goals. By replacing this with a stochastic assumption, the bounds developed become stronger, but at the price of a somewhat more involved analysis.

by an  $\ell_1$ -norm, leading to the Basis Pursuit Denoising algorithm,

$$(P_1^\epsilon) \quad \hat{\mathbf{x}}_1^\epsilon = \arg \min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{subject to} \quad \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 \leq \epsilon. \quad (8.2)$$

We now turn to describe the *Dantzig-Selector* (DS), which is a very interesting alternative to  $(P_1^\epsilon)$ . Candes and Tao propose to estimate  $\mathbf{x}_0$  by solving the problem

$$(P_{DS}^\lambda) \quad \hat{\mathbf{x}}_{DS}^\lambda = \arg \min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{subject to} \quad \|\mathbf{A}^T(\mathbf{b} - \mathbf{A}\mathbf{x})\|_\infty \leq \lambda\sigma. \quad (8.3)$$

As can be seen, the constraint has been changed! The original requirement  $\|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 \leq \epsilon$  makes sense, as the residual is expected to be of a limited norm. Furthermore, this expression also emerges as the log-likelihood of the vector  $\mathbf{b}$  given  $\mathbf{x}$ , and as such, it ties the formulation of  $(P_1^\epsilon)$  to the MAP estimation – a relation that will be made explicit in Chapter 11. Nevertheless, one cannot overlook the plain fact that this constraint does not force the residual  $\mathbf{b} - \mathbf{A}\mathbf{x}$  to behave like random white Gaussian noise. In particular, structured and correlated residuals are permitted by this constraint, despite our knowledge that they are highly improbable.

The alternative proposed by DS computes the residual  $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ , and then evaluates its inner products with the columns of  $\mathbf{A}$ . These inner products form various weighted combinations of the entries in  $\mathbf{r}$  and all those are expected to be small. The constraint thus proceeds by requiring that all these are below some pre-specified threshold,  $\lambda\sigma$ .

In order to get an intuition as to why to multiply by  $\mathbf{A}^T$ , consider the following scenario: Suppose that the proposed solution  $\mathbf{x}$  is close to  $\mathbf{x}_0$ , but one of non-zeros in  $\mathbf{x}_0$  is omitted, and this entry refers to the column  $\mathbf{a}_i$  from  $\mathbf{A}$ . This omission is reflected in a residual of the form  $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}_0 + c\mathbf{a}_i = \mathbf{e} + c\mathbf{a}_i$ . This vector correlates strongly with  $\mathbf{a}_i$ , since  $\mathbf{a}_i^T \mathbf{r} = \mathbf{a}_i^T \mathbf{e} + c$ . Whereas the first part is likely to be near-zero, the addition of  $c$  could be large (especially if this atom is dominant in the creation of  $\mathbf{b}$ ), and this would serve as a detector for the infeasibility of the proposed solution. This way, by sweeping through all columns of  $\mathbf{A}$ , considering an inner product with each of them, and requiring all to be relatively small, we make sure that the residual is “whitened,” and no traces of the atoms remain in it.

The problem posed in Equation (8.3) has a linear programming form. This is easily observed by adding an auxiliary variable  $\mathbf{u} \geq |\mathbf{x}|$ , and reformulating the problem to be

$$\min_{\mathbf{u}, \mathbf{x}} \quad \mathbf{1}^T \mathbf{u} \quad \text{subject to} \quad \left\{ \begin{array}{c} -\mathbf{u} \leq \mathbf{x} \leq \mathbf{u} \\ \text{and} \\ -\sigma\lambda\mathbf{1} \leq \mathbf{A}^T(\mathbf{b} - \mathbf{A}\mathbf{x}) \leq \sigma\lambda\mathbf{1} \end{array} \right\}. \quad (8.4)$$

All the terms (penalty and constraints) are linear with respect to the unknowns, and thus this is a classic LP, for which specific solvers can be tailored (e.g., an iterative-shrinkage algorithm). As a convex problem, obtaining the global minimizer of this problem is within reach, and this estimator is therefore quite appealing.

Another way to convert DS into a linear programming form is to split the unknown  $\mathbf{x}$  into two vectors that contain its positive and negative entries,  $\mathbf{x} = \mathbf{u} - \mathbf{v}$ .

Then the problem can be re-written as

$$\begin{aligned} \min_{\mathbf{u}, \mathbf{v}} \quad & \mathbf{1}^T \mathbf{u} + \mathbf{1}^T \mathbf{v} \\ \text{subject to} \quad & \left\{ \begin{array}{c} \mathbf{u} \geq \mathbf{0} \quad \mathbf{v} \geq \mathbf{0} \\ \text{and} \\ -\sigma \lambda \mathbf{1} \leq \mathbf{A}^T (\mathbf{b} - \mathbf{A} \mathbf{u} + \mathbf{A} \mathbf{v}) \leq \sigma \lambda \mathbf{1} \end{array} \right\}. \end{aligned} \quad (8.5)$$

This approach should remind the reader of the technique used to convert the  $\ell_1$  minimization into an LP form, in Chapter 1. Similar to the discussion there, we should convince ourselves that the supports of the two vectors  $\mathbf{u}$  and  $\mathbf{v}$  do not overlap, or put differently, that  $\mathbf{u}^T \mathbf{v} = 0$ . This is easily verified using the same rationale – if two such matching elements become non-zeros in the two vectors, then by decreasing both by the same amount such that the smallest of them becomes zero, the penalty of the problem gets smaller, while the constraint remains unchanged.

Is DS better than the Basis-Pursuit-Denoising for recovering  $\mathbf{x}_0$ ? Based on the description given above, the answer one would give is positive – after all, it seems that the constraint DS uses is far more informative than the one posed by the BP. As we shall see next, this intuition may be quite misleading, and at least when  $\mathbf{A}$  is unitary, there is no difference between the two.

## 8.2 The Unitary Case

When the matrix  $\mathbf{A}$  is unitary, we have seen that both  $(P_0^\epsilon)$  and  $(P_1^\epsilon)$  admit closed-form solutions, in the form of hard or soft shrinkage. Can the same be said for the DS algorithm?

Looking at the constraint  $\|\mathbf{A}^T (\mathbf{b} - \mathbf{A} \mathbf{x})\|_\infty \leq \lambda \sigma$ , and using the fact that  $\mathbf{A}^T \mathbf{A} = \mathbf{I}$ , this now becomes the requirement  $\|\mathbf{A}^T \mathbf{b} - \mathbf{x}\|_\infty \leq \lambda \sigma$ . Thus, DS becomes a set of  $m$  independent scalar optimization problems of the form

$$\min_{\mathbf{x}} |\mathbf{x}| \quad \text{subject to} \quad |\mathbf{a}_i^T \mathbf{b} - \mathbf{x}| \leq \lambda \sigma, \quad i = 1, 2, \dots, m. \quad (8.6)$$

The solution is easily obtained as

$$x_i^{opt} = \begin{cases} \mathbf{a}_i^T \mathbf{b} - \lambda \sigma & \mathbf{a}_i^T \mathbf{b} \geq \lambda \sigma \\ 0 & |\mathbf{a}_i^T \mathbf{b}| < \lambda \sigma \\ \mathbf{a}_i^T \mathbf{b} + \lambda \sigma & \mathbf{a}_i^T \mathbf{b} \leq -\lambda \sigma \end{cases}. \quad (8.7)$$

This is a classic soft-thresholding operation, exactly equivalent to the solution obtained for the problem  $(P_1^\epsilon)$  (for a proper choice of parameters  $\lambda \sigma$  versus  $\epsilon$ ). This equivalence between  $(P_1^\epsilon)$  and  $(P_{DS}^\lambda)$  is not true in general, though.

Returning to the general case, the above result also suggests that it might be possible to propose iterative-shrinkage algorithms for handling DS, just as we harnessed such methods for the  $\ell_2$ -based penalty.



### 8.3 Revisiting the Restricted Isometry Machinery

In order to present a theoretical analysis of DS, we shall use the RIP as defined in Chapter 5. We bring this definition again for completeness of the presentation, and then add a new and related definition that will be found helpful in the analysis of the Dantzig-Selector.

**Definition 8.1.** For a matrix  $\mathbf{A}$  of size  $n \times m$  ( $m > n$ ) with  $\ell_2$ -normalized columns, and for an integer scalar  $s \leq n$ , consider submatrices  $\mathbf{A}_s$  containing  $s$  columns from  $\mathbf{A}$ . Define  $\delta_s$  as the smallest quantity such that

$$\forall \mathbf{c} \in \mathbb{R}^s \quad (1 - \delta_s) \|\mathbf{c}\|_2^2 \leq \|\mathbf{A}_s \mathbf{c}\|_2^2 \leq (1 + \delta_s) \|\mathbf{c}\|_2^2, \quad (8.8)$$

hold true for any choice of  $s$  columns. Then  $\mathbf{A}$  has an  $s$ -restricted isometry property with a constant  $\delta_s$ .

A second definition, in line with the above, is for the *Restricted Orthogonality Property* (ROP):

**Definition 8.2.** For a matrix  $\mathbf{A}$  of size  $n \times m$  ( $m > n$ ) with  $\ell_2$ -normalized columns, and for two integer scalars  $s_1, s_2 \leq n$ , consider two disjoint sets of  $s_1$  and  $s_2$  columns from  $\mathbf{A}$ , denoted as the submatrices  $\mathbf{A}_{s_1}$  and  $\mathbf{A}_{s_2}$ . Define  $\theta_{s_1, s_2}$  as the smallest quantity such that  $\forall \mathbf{c}_1 \in \mathbb{R}^{s_1}$  and  $\mathbf{c}_2 \in \mathbb{R}^{s_2}$

$$|\mathbf{c}_1^T \mathbf{A}_{s_1}^T \mathbf{A}_{s_2} \mathbf{c}_2| = |\langle \mathbf{A}_{s_1} \mathbf{c}_1, \mathbf{A}_{s_2} \mathbf{c}_2 \rangle| \leq \theta_{s_1, s_2} \|\mathbf{c}_1\|_2 \|\mathbf{c}_2\|_2, \quad (8.9)$$

holds true for any choices of  $s_1, s_2$  columns. Then  $\mathbf{A}$  has an  $s_1, s_2$ -restricted orthogonality property with a constant  $\theta_{s_1, s_2}$ .

This definition aims to characterize an independence between disjoint sets of columns, such that they do not interfere with each other in an attempt to represent the same vector. As for the RIP, evaluation of the value of  $\theta_{s_1, s_2}$  is impractical, and here as well one could bound  $\theta_{s_1, s_2}$  using the mutual-coherence by  $\theta_{s_1, s_2} \leq \sqrt{s_1 s_2} \mu(\mathbf{A})$ , but we will not dwell on this here, as these bounds are too crude for practical use. We note two important observations regarding the above definitions:

1. While for a general matrix  $\mathbf{A}$  it is virtually impossible to evaluate  $\delta_s$  and  $\theta_{s_1, s_2}$ , one could evaluate these quite reliably for random matrices. This is why such measures are popular for compressed-sensing applications, where  $\mathbf{A}$  can be made stochastic.
2. For a general matrix  $\mathbf{A}$ , it is most likely that the RIP and the ROP measures lead to pessimistic theoretical results. This is because those measures are, by their definitions, worst-case measures, and as such, they are unforgiving to trivial situations, such as near-by columns.

## 8.4 Dantzig-Selector Performance Guaranty

Armed with the above definitions, we are now able to analyze the performance of DS. We bring here only the first among several of the results that Candes and Tao provide in their important paper, as this result by itself is sufficient to see the strength of DS and how the above definitions can be used.

**Theorem 8.1. (Stability of DS):** *Consider the instance of problem  $(P_{DS}^\lambda)$  defined by the triplet  $(\mathbf{A}, \mathbf{b}, \sigma)$  and  $\lambda = \sqrt{2(1+a)\log m}$ . Suppose that a vector  $\mathbf{x}_0 \in \mathbb{R}^m$  is  $s$ -sparse ( $\|\mathbf{x}_0\|_0 = s$ ), where  $s$  satisfies  $\delta_{2s} + \theta_{s,2s} < 1$ , and  $\mathbf{b} = \mathbf{A}\mathbf{x}_0 + \mathbf{e}$ , where  $\mathbf{e}$  is a zero-mean random Gaussian noise with iid entries having variance  $\sigma^2$ .*

*Then, the solution  $\mathbf{x}_{DS}^\lambda$  of  $(P_{DS}^\lambda)$  obeys*

$$\|\mathbf{x}_{DS}^\lambda - \mathbf{x}_0\|_2^2 \leq C^2 \cdot 2 \log m \cdot s \cdot \sigma^2, \quad (8.10)$$

*with probability exceeding  $1 - (\sqrt{\pi \log mm^a})^{-1}$ , and  $C = 4/(1 - \delta_{2s} + \theta_{s,2s})$ .*

Before we turn to prove this result, we briefly discuss its meaning and implications. We draw attention to the following observations:

- There is a strong resemblance between this result and the one posed in Theorem 5.3 in Chapter 5, for the BPDN stability. However, the bound here is far better, as it is proportional to  $s\sigma^2$  up to a constant (and log) factors, whereas there the bound is proportional to  $n\sigma^2$ , absorbing all the noise power. As was explained earlier, the reason for this difference is the change from an adversary noise to a stochastic one.
- The way this theorem is posed here may mislead the reader to assume that this is an average performance analysis of DS, as the result is conditioned with a probability of correctness. However, this probability has to do with the randomness of the noise and nothing to do with a probabilistic ensemble of supports or random generators of  $\mathbf{x}_0$ .
- The sparsity condition posed is obscure, being indirectly dictated by the RIP and ROP parameters. Thus, this result becomes both clearer and stronger as we turn to handle random matrices  $\mathbf{A}$ . Performance prediction of this theorem for general matrices  $\mathbf{A}$  may be too pessimistic, as it is also of worst-case nature.
- This result is surprising! In order to validate the strength of the bound obtained, consider an estimation of  $\mathbf{x}_0$  by an oracle that knows the correct support  $s$ . In such a case, the estimation boils down to a Least-Squares problem of the form

$$\mathbf{z}^{opt} = \arg \min_{\mathbf{z}} \|\mathbf{A}_s \mathbf{z} - \mathbf{b}\|_2^2 = (\mathbf{A}_s^T \mathbf{A}_s)^{-1} \mathbf{A}_s^T \mathbf{b}. \quad (8.11)$$

The error in such a case is only reflected on the support  $s$ , as the two vectors are the same (zeros) elsewhere. Using the fact that  $\mathbf{b} = \mathbf{A}_s \mathbf{x}_0^s + \mathbf{e}$ , the expectation of is error becomes

$$E(\|\mathbf{z}^{opt} - \mathbf{x}_0^s\|_2^2) = E(\|(\mathbf{A}_s^T \mathbf{A}_s)^{-1} \mathbf{A}_s^T \mathbf{b} - \mathbf{x}_0^s\|_2^2) \quad (8.12)$$

$$\begin{aligned}
&= E \left( \|(\mathbf{A}_s^T \mathbf{A}_s)^{-1} \mathbf{A}_s^T (\mathbf{A}_s \mathbf{x}_0^s + \mathbf{e}) - \mathbf{x}_0^s\|_2^2 \right) \\
&= E \left( \|(\mathbf{A}_s^T \mathbf{A}_s)^{-1} \mathbf{A}_s^T \mathbf{e}\|_2^2 \right) \\
&= \text{trace} \left\{ (\mathbf{A}_s^T \mathbf{A}_s)^{-1} \mathbf{A}_s^T E(\mathbf{e} \mathbf{e}^T) \mathbf{A}_s (\mathbf{A}_s^T \mathbf{A}_s)^{-1} \right\} \\
&= \sigma^2 \cdot \text{trace} \left\{ (\mathbf{A}_s^T \mathbf{A}_s)^{-1} \right\} \geq \frac{s \cdot \sigma^2}{1 + \delta_s}.
\end{aligned}$$

We have used the simple relation  $E(\mathbf{e} \mathbf{e}^T) = \sigma^2 \mathbf{I}$ , and also the fact that the eigenvalues of  $\mathbf{A}_s^T \mathbf{A}_s$  are known to be in the range  $[1 - \delta_s, 1 + \delta_s]$  (due to the RIP), and thus those of the inverse are in the range  $[1/(1 + \delta_s), 1/(1 - \delta_s)]$ . As the trace sums the eigenvalues, the lower bound above is established.

All this means that DS gets a result very close to the best possible one, as the oracle delivers, and this is done without having any clue about the support. This is the surprise mentioned.

We now turn to provide the proof for this result. We follow closely the steps taken by Candes and Tao in their work, with small modifications for better clarity of the proof.

**Proof:** We have two candidate solutions to consider,  $\mathbf{x}_0$  and  $\mathbf{x}_{DS}^\lambda$ , and we are interested in the distance between them,  $\mathbf{d} = \mathbf{x}_{DS}^\lambda - \mathbf{x}_0$ . While  $\mathbf{x}_{DS}^\lambda$  is clearly feasible,  $\mathbf{x}_0$  is not necessarily so. Thus, we shall look only at those fraction of cases where  $\mathbf{x}_0$  is feasible, implying that

$$\|\mathbf{A}^T (\mathbf{A} \mathbf{x}_0 - \mathbf{b})\|_\infty = \|\mathbf{A}^T \mathbf{e}\|_\infty \leq \lambda \sigma. \quad (8.13)$$

Every item in the vector  $\mathbf{A}^T \mathbf{e}$  is a Gaussian random variable  $\sim \mathcal{N}(0, \sigma)$  (since the columns of  $\mathbf{A}$  are normalized), and thus

$$\begin{aligned}
P \left( \frac{1}{\sigma} \|\mathbf{A}^T \mathbf{e}\|_\infty \leq \lambda \right) &= \left( 1 - 2 \int_\lambda^\infty \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \right)^m \\
&\geq 1 - \frac{2m}{\sqrt{2\pi}} \int_\lambda^\infty e^{-\frac{x^2}{2}} dx,
\end{aligned} \quad (8.14)$$

where we have used the inequality  $(1 - \epsilon)^m \geq 1 - m\epsilon$ , true for  $0 \leq \epsilon \leq 1$ . Notice that the vector  $\mathbf{A}^T \mathbf{e}/\sigma$  is normalized and thus the canonic expressions. Using the inequality (verified easily by taking a derivative of this function)

$$\int_u^\infty e^{-\frac{x^2}{2}} dx \leq \frac{e^{-\frac{u^2}{2}}}{u},$$

we obtain

$$\begin{aligned}
P \left( \frac{1}{\sigma} \|\mathbf{A}^T \mathbf{e}\|_\infty \leq \lambda \right) &\geq 1 - \frac{2m}{\sqrt{2\pi}} \int_\lambda^\infty e^{-\frac{x^2}{2}} dx \\
&\geq 1 - \frac{2m}{\sqrt{2\pi}\lambda} e^{-\frac{\lambda^2}{2}}.
\end{aligned} \quad (8.15)$$

Plugging to this expression  $\lambda = \sqrt{2(1+a)\log m}$ , we get

$$\begin{aligned} P\left(\frac{1}{\sigma}\|\mathbf{A}^T \mathbf{e}\|_\infty \leq \lambda\right) &\geq 1 - \frac{2m}{\sqrt{2\pi}\lambda} e^{-\frac{\lambda^2}{2}} \\ &= 1 - \frac{1}{\sqrt{\pi(1+a)\log m}} \cdot m^{-a}. \end{aligned} \quad (8.16)$$

For sufficiently large  $a$ , this probability can be made arbitrarily small, and thus we consider the majority of the cases in which  $\mathbf{x}_0$  is feasible.

We define the residual created by the optimal solution of DS as  $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}_{DS}^\lambda$ . We have that

$$\begin{aligned} \mathbf{A}^T \mathbf{A} \mathbf{d} &= \mathbf{A}^T \mathbf{A} (\mathbf{x}_0 - \mathbf{x}_{DS}^\lambda) \\ &= \mathbf{A}^T (\mathbf{A}\mathbf{x}_0 - \mathbf{b}) - \mathbf{A}^T (\mathbf{A}\mathbf{x}_{DS}^\lambda - \mathbf{b}) \\ &= \mathbf{A}^T (\mathbf{r} - \mathbf{e}). \end{aligned} \quad (8.17)$$

Since the two solutions are feasible, we have that  $\|\mathbf{A}^T \mathbf{e}\|_\infty \leq \lambda\sigma$  and  $\|\mathbf{A}^T \mathbf{r}\|_\infty \leq \lambda\sigma$ . Thus, using the triangle inequality,

$$\|\mathbf{A}^T \mathbf{A} \mathbf{d}\|_\infty = \|\mathbf{A}^T (\mathbf{e} - \mathbf{r})\|_\infty \leq \|\mathbf{A}^T \mathbf{e}\|_\infty + \|\mathbf{A}^T \mathbf{r}\|_\infty \leq 2\lambda\sigma. \quad (8.18)$$

Another direction we take exploits the fact that since  $\mathbf{x}_{DS}^\lambda$  is the solution of DS, it has the shortest  $\ell_1$ -length, compared to any other feasible solution, and in particular,  $\mathbf{x}_0$ . Thus,  $\|\mathbf{x}_{DS}^\lambda\|_1 = \|\mathbf{x}_0 + \mathbf{d}\|_1 \leq \|\mathbf{x}_0\|_1$ . We denote  $\|\mathbf{v}\|_{1,s}$  as the  $\ell_1$ -norm of  $\mathbf{v}$  restricted to the support  $s$  of the sparse vector  $\mathbf{x}_0$ , and similarly,  $\|\mathbf{v}\|_{1,s^c}$  is restricted to summation of the off-support entries of  $\mathbf{v}$ . We use the inequality

$$\begin{aligned} \|\mathbf{x}_0 + \mathbf{d}\|_1 &= \|\mathbf{x}_0 + \mathbf{d}\|_{1,s} + \|\mathbf{x}_0 + \mathbf{d}\|_{1,s^c} \\ &\geq \|\mathbf{x}_0\|_{1,s} - \|\mathbf{d}\|_{1,s} + \|\mathbf{d}\|_{1,s^c}, \end{aligned} \quad (8.19)$$

where we have used the relations  $\|\mathbf{x}_0 + \mathbf{d}\|_{1,s^c} = \|\mathbf{d}\|_{1,s^c}$  and  $\|\mathbf{x}_0 + \mathbf{d}\|_{1,s} \geq \|\mathbf{x}_0\|_{1,s} - \|\mathbf{d}\|_{1,s}$ . Since  $\|\mathbf{x}_0\|_1 = \|\mathbf{x}_0\|_{1,s}$ , we return to the inequality  $\|\mathbf{x}_0 + \mathbf{d}\|_1 \leq \|\mathbf{x}_0\|_1$  and obtain the requirement

$$\|\mathbf{d}\|_{1,s} \geq \|\mathbf{d}\|_{1,s^c}. \quad (8.20)$$

This condition says that the difference vector  $\mathbf{d}$  must have its  $\ell_1$ -energy concentrated on the support  $s$ .

To summarize thus far, the difference vector  $\mathbf{d}$  must satisfy two restrictions posed in Equations (8.18) and (8.20). Among all these, we aim to search for the highest  $\|\mathbf{d}\|_2^2$  and show that it is in fact bounded from above, as the theorem suggests.

In order to proceed, we introduce the following notation: whereas we use  $s$  to denote the support of  $\mathbf{x}_0$ ,  $q$  stands now for a support of size  $2s$ , containing the original support, and the next largest (in absolute value)  $s$  entries from  $\mathbf{d}$ . The sub-matrix  $\mathbf{A}_q$  is of size  $n \times 2s$ , containing the columns indicated by  $q$ . Using the RIP relation we

have that for any vector  $\mathbf{v}$  of length  $2s$ ,

$$\sqrt{1 - \delta_{2s}} \|\mathbf{v}\|_2 \leq \|\mathbf{A}_q \mathbf{v}\|_2 \leq \sqrt{1 + \delta_{2s}} \|\mathbf{v}\|_2, \quad (8.21)$$

and we assume that  $\delta_{2s} < 1$ , implying that  $\mathbf{Q} = \mathbf{A}_q^T \mathbf{A}_q$  is positive-definite. The above inequality states that  $(1 - \delta_{2s}) \mathbf{v}^T \mathbf{v} \leq \mathbf{v}^T \mathbf{Q} \mathbf{v}$ . For the assignment  $\mathbf{w} = \mathbf{Q}^{0.5} \mathbf{v}$ , this can be re-stated as  $(1 - \delta_{2s}) \mathbf{w}^T \mathbf{Q}^{-1} \mathbf{w} \leq \mathbf{w}^T \mathbf{w}$ . Assigning  $\mathbf{w} = \mathbf{A}_q^T \mathbf{A} \mathbf{d}$  we have

$$\|\mathbf{w}\|_2^2 = \mathbf{d}^T \mathbf{A}^T \mathbf{A}_q \mathbf{A}_q^T \mathbf{A} \mathbf{d} = \|\mathbf{A}_q^T \mathbf{A} \mathbf{d}\|_2^2, \quad (8.22)$$

and

$$\begin{aligned} \mathbf{w}^T \mathbf{Q}^{-1} \mathbf{w} &= \mathbf{d}^T \mathbf{A}^T \mathbf{A}_q (\mathbf{A}_q^T \mathbf{A}_q)^{-1} \mathbf{A}_q^T \mathbf{A} \mathbf{d} \\ &= \mathbf{d}^T \mathbf{A}^T \mathbf{P}_q \mathbf{A} \mathbf{d} \\ &= \mathbf{d}^T \mathbf{A}^T \mathbf{P}_q^2 \mathbf{A} \mathbf{d} \\ &= \|\mathbf{P}_q \mathbf{A} \mathbf{d}\|_2^2. \end{aligned} \quad (8.23)$$

The term  $\mathbf{P}_q = \mathbf{A}_q (\mathbf{A}_q^T \mathbf{A}_q)^{-1} \mathbf{A}_q^T$  is a projection onto the space spanned by the columns of  $\mathbf{A}_q$ , and as such it is nilpotent ( $\mathbf{P}_q^2 = \mathbf{P}_q$ ). All this leads to the inequality

$$(1 - \delta_{2s}) \|\mathbf{P}_q \mathbf{A} \mathbf{d}\|_2^2 \leq \|\mathbf{A}_q^T \mathbf{A} \mathbf{d}\|_2^2. \quad (8.24)$$

Using the inequality posed in (8.18),  $\|\mathbf{A}^T \mathbf{A} \mathbf{d}\|_\infty \leq 2\lambda\sigma$ , we know that every entry in the vector  $\mathbf{A}^T \mathbf{A} \mathbf{d}$  is bounded by  $2\lambda\sigma$ . The vector  $\mathbf{A}_q^T \mathbf{A} \mathbf{d}$  accumulates  $2s$  of those entries, and thus

$$\|\mathbf{P}_q \mathbf{A} \mathbf{d}\|_2^2 \leq \frac{1}{1 - \delta_{2s}} \|\mathbf{A}_q^T \mathbf{A} \mathbf{d}\|_2^2 \leq \frac{2s \cdot (4\lambda^2 \sigma^2)}{1 - \delta_{2s}}. \quad (8.25)$$

We now turn to handle the left-hand-side of the above expression, bounding it from below. We slice the difference vector  $\mathbf{d}$  into a set of vectors  $\{\mathbf{d}_j\}_{j=0,1,2,\dots}$ , all of the same length  $m$ . Each  $\mathbf{d}_j$  contains  $s$  non-zeros, where  $\mathbf{d}_0$  has the support  $s$ ,  $\mathbf{d}_1$  has a support  $q \setminus s$ , and so on, in descending order of the magnitude of the non-zeros in  $\mathbf{d}$ . Thus, clearly we have that  $\mathbf{d} = \sum_j \mathbf{d}_j$ , and we can therefore write

$$\begin{aligned} \mathbf{P}_q \mathbf{A} \mathbf{d} &= \mathbf{P}_q \mathbf{A} \mathbf{d}_0 + \mathbf{P}_q \mathbf{A} \mathbf{d}_1 + \sum_{j>1} \mathbf{P}_q \mathbf{A} \mathbf{d}_j \\ &= \mathbf{A}_q \mathbf{d} + \sum_{j>1} \mathbf{P}_q \mathbf{A} \mathbf{d}_j. \end{aligned} \quad (8.26)$$

The first two terms are simplified by removing the projection operation, as these two vectors are in the space projected upon. Also, we use the fact that  $\mathbf{A} \mathbf{d}_0 + \mathbf{A} \mathbf{d}_1 = \mathbf{A}_q \mathbf{d}$ . Using the triangle inequality in reversed mode ( $\|\mathbf{v}\|_2 = \|\mathbf{v} + \mathbf{u} - \mathbf{u}\|_2 \leq \|\mathbf{v} + \mathbf{u}\|_2 + \|\mathbf{u}\|_2$ ), we get

$$\begin{aligned}
\|\mathbf{P}_q \mathbf{A} \mathbf{d}\|_2 &= \left\| \mathbf{A}_q \mathbf{d} + \sum_{j>1} \mathbf{P}_q \mathbf{A} \mathbf{d}_j \right\|_2 \\
&\geq \|\mathbf{A}_q \mathbf{d}\|_2 - \left\| \sum_{j>1} \mathbf{P}_q \mathbf{A} \mathbf{d}_j \right\|_2 \\
&\geq \|\mathbf{A}_q \mathbf{d}\|_2 - \sum_{j>1} \|\mathbf{P}_q \mathbf{A} \mathbf{d}_j\|_2.
\end{aligned} \tag{8.27}$$

The RIP relation posed in Equation (8.21) gives that  $\|\mathbf{A}_q \mathbf{d}\|_2 \geq \sqrt{1 - \delta_{2s}} \|\mathbf{d}\|_{2,q}$ , and thus

$$\|\mathbf{P}_q \mathbf{A} \mathbf{d}\|_2 \geq \sqrt{1 - \delta_{2s}} \|\mathbf{d}\|_{2,q} - \sum_{j>1} \|\mathbf{P}_q \mathbf{A} \mathbf{d}_j\|_2. \tag{8.28}$$

For  $j > 1$ , the term  $\|\mathbf{P}_q \mathbf{A} \mathbf{d}_j\|_2^2$  can be written as an inner product between the vector  $\mathbf{P}_q \mathbf{A} \mathbf{d}_j$  that is in the span of the  $2s$  columns of  $\mathbf{A}_s$ , and  $\mathbf{A} \mathbf{d}_j$ , spanned by a disjoint set of  $s$  columns from  $\mathbf{A}$  (again using the property  $\mathbf{P}_q = \mathbf{P}_q^2$ ). Using the ROP we have that such an inner product is bounded by

$$\|\mathbf{P}_q \mathbf{A} \mathbf{d}_j\|_2^2 = \langle \mathbf{P}_q \mathbf{A} \mathbf{d}_j, \mathbf{A} \mathbf{d}_j \rangle \leq \theta_{2s,s} \cdot \|\mathbf{P}_q \mathbf{A} \mathbf{d}_j\|_2 \|\mathbf{A} \mathbf{d}_j\|_2. \tag{8.29}$$

Exploiting the RIP again, we have that  $\|\mathbf{A} \mathbf{d}_j\|_2 \leq \sqrt{1 + \delta_s} \|\mathbf{d}_j\|_2$ , since this multiplication combines  $s$  columns from  $\mathbf{A}$ . Thus

$$\begin{aligned}
\|\mathbf{P}_q \mathbf{A} \mathbf{d}_j\|_2 &\leq \theta_{2s,s} \cdot \sqrt{1 + \delta_s} \|\mathbf{d}_j\|_2 \\
&\leq \frac{\theta_{2s,s}}{\sqrt{1 - \delta_s}} \|\mathbf{d}_j\|_2 \\
&\leq \frac{\theta_{2s,s}}{\sqrt{1 - \delta_{2s}}} \|\mathbf{d}_j\|_2.
\end{aligned} \tag{8.30}$$

Here we have used the properties  $1 + \delta_s \leq (1 - \delta_s)^{-1}$  and  $\delta_{2s} \geq \delta_s$ .

We now observe that for  $j > 1$ , the  $s$  non-zero entries of  $\mathbf{d}_j$  are smaller than the mean of the entries in  $|\mathbf{d}_{j-1}|$ , expressed as  $\|\mathbf{d}_{j-1}\|_1 / s$ . Summing this upper bound on the entries over the  $s$  non-zeros in  $\mathbf{d}_j$  leads to  $\|\mathbf{d}_j\|_2 \leq \|\mathbf{d}_{j-1}\|_1 / \sqrt{s}$ . Thus,

$$\begin{aligned}
\sum_{j>1} \|\mathbf{P}_q \mathbf{A} \mathbf{d}_j\|_2 &\leq \frac{\theta_{2s,s}}{\sqrt{1 - \delta_{2s}}} \sum_{j>1} \|\mathbf{d}_j\|_2 \\
&\leq \frac{\theta_{2s,s}}{\sqrt{s} \cdot \sqrt{1 - \delta_{2s}}} \sum_{j>0} \|\mathbf{d}_j\|_1 \\
&= \frac{\theta_{2s,s}}{\sqrt{s} \cdot \sqrt{1 - \delta_{2s}}} \|\mathbf{d}\|_{1,s^c}.
\end{aligned} \tag{8.31}$$

Returning to the inequality posed in (8.28), plugging the above leads to

$$\|\mathbf{P}_q \mathbf{A} \mathbf{d}\|_2 \geq \sqrt{1 - \delta_{2s}} \|\mathbf{d}\|_{2,q} - \frac{\theta_{2s,s}}{\sqrt{s} \cdot \sqrt{1 - \delta_{2s}}} \|\mathbf{d}\|_{1,s^c}. \quad (8.32)$$

Combining this with (8.25) we get

$$\|\mathbf{d}\|_{2,q} \leq \frac{\sqrt{2s} \cdot (2\lambda\sigma)}{1 - \delta_{2s}} + \frac{\theta_{2s,s}}{\sqrt{s} \cdot (1 - \delta_{2s})} \|\mathbf{d}\|_{1,s^c}. \quad (8.33)$$

Using the requirement posed in (8.20) and the  $\ell_1 - \ell_2$  relationship for vectors of length  $s$ ,  $\|\mathbf{v}\|_1 \leq \sqrt{s} \|\mathbf{v}\|_2$ , we obtain  $\|\mathbf{d}\|_{1,s^c} \leq \|\mathbf{d}\|_{1,s} \leq \sqrt{s} \|\mathbf{d}\|_{2,s}$ , and thus

$$\|\mathbf{d}\|_{2,q} \leq \frac{\sqrt{2s} \cdot (2\lambda\sigma)}{1 - \delta_{2s}} + \frac{\theta_{2s,s}}{1 - \delta_{2s}} \|\mathbf{d}\|_{2,s}. \quad (8.34)$$

Using the fact that  $\|\mathbf{d}\|_{2,s} \leq \|\mathbf{d}\|_{2,q}$  we write

$$\|\mathbf{d}\|_{2,q} \leq \frac{\sqrt{2s} \cdot (2\lambda\sigma)}{1 - \delta_{2s}} + \frac{\theta_{2s,s}}{1 - \delta_{2s}} \|\mathbf{d}\|_{2,q}, \quad (8.35)$$

and rearranging terms leads to

$$\begin{aligned} \left(1 - \frac{\theta_{2s,s}}{1 - \delta_{2s}}\right) \|\mathbf{d}\|_{2,q} &\leq \frac{\sqrt{2s} \cdot (2\lambda\sigma)}{1 - \delta_{2s}} \\ \Rightarrow \|\mathbf{d}\|_{2,q} &\leq \frac{\sqrt{2s} \cdot (2\lambda\sigma)}{1 - \delta_{2s} - \theta_{2s,s}}, \end{aligned} \quad (8.36)$$

conditioned on the requirement  $1 - \delta_{2s} - \theta_{2s,s} > 0$ .

To conclude the proof, we return to the observation made earlier, stating that  $\|\mathbf{d}_j\|_2 \leq \|\mathbf{d}_{j-1}\|_1 / \sqrt{s}$ , and thus

$$\|\mathbf{d}\|_{2,q^c}^2 = \sum_{j>1} \|\mathbf{d}_j\|_2^2 \leq \frac{1}{s} \sum_{j>0} \|\mathbf{d}_j\|_1^2 = \frac{1}{s} \|\mathbf{d}\|_{1,s^c}^2. \quad (8.37)$$

Using this, the relation (8.20), and the fact that  $\|\mathbf{d}\|_{2,s}^2 \leq \|\mathbf{d}\|_{2,q}^2$ , we obtain

$$\begin{aligned} \|\mathbf{d}\|_2^2 &= \|\mathbf{d}\|_{2,q}^2 + \|\mathbf{d}\|_{2,q^c}^2 \\ &\leq \|\mathbf{d}\|_{2,q}^2 + \frac{1}{s} \|\mathbf{d}\|_{1,s^c}^2 \\ &\leq \|\mathbf{d}\|_{2,q}^2 + \frac{1}{s} \|\mathbf{d}\|_{1,s}^2 \\ &\leq 2\|\mathbf{d}\|_{2,q}^2, \end{aligned} \quad (8.38)$$

which leads to the claimed bound in the theorem by plugging the expression from (8.36).  $\square$

## 8.5 Dantzig-Selector in Practice

So, how does DS perform in practice? How does it compare to the Basis-Pursuit? Of course, we already know that the two are exactly equivalent if  $\mathbf{A}$  is unitary. In this section we present a limited experiment that suggests answers to these questions. We generate a matrix  $\mathbf{A}$  of size  $50 \times 80$ , by taking the unitary DCT matrix, choosing a subset of 50 of its rows at random, and normalizing the columns. We then generate a random vector  $\mathbf{x}_0$  of length 80, such that  $\|\mathbf{x}_0\|_0 = 10$ , in arbitrary and random locations, and with random normal non-zero values. Finally, we compute  $\mathbf{b} = \mathbf{A}\mathbf{x}_0 + \mathbf{e}$ , such that  $\mathbf{e}$  has zero-mean Gaussian iid entries with standard-deviation  $\sigma = 0.05$ . This forms our problem suite, and we apply DS and BP in an attempt to recover  $\mathbf{x}_0$ . The DS is formulated as

$$(P_{DS}^\lambda) \quad \hat{\mathbf{x}}_{DS}^\lambda = \arg \min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{subject to} \quad \|\mathbf{A}^T(\mathbf{b} - \mathbf{A}\mathbf{x})\|_\infty \leq \lambda\sigma.$$

We solve this problem using Matlab's linear programming tool, using the second of the two conversions discussed in the beginning of this chapter. As we do not know which  $\lambda$  to use, we sweep through a set of values in the range  $[0.1, 100]$  (the top value guarantees that the zero solution is also considered, since in this case  $\|\mathbf{A}^T\mathbf{b}\|_\infty \leq \lambda_{\max}\sigma$ ). The BP is formulated as

$$(P_{BP}^\mu) \quad \hat{\mathbf{x}}_{BP}^\mu = \arg \min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{subject to} \quad \|\mathbf{b} - \mathbf{A}\mathbf{x}\|_2 \leq \mu\sigma,$$

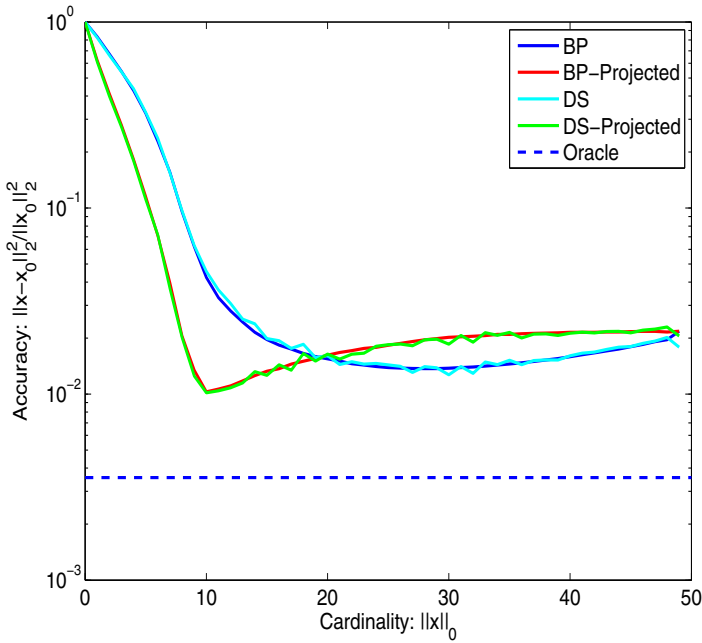
and we use the LARS solver mentioned in Chapter 5 for solving this problem for various thresholds  $\mu\sigma$ , such that we cover all possible output cardinalities, from the empty to the one that contains 49 non-zeros.

As we obtain a series of solutions (as a function of the varying thresholds) for each method, we organize these as a graph that presents the relative accuracy of the solution,  $\|\hat{\mathbf{x}} - \mathbf{x}_0\|_2^2 / \|\mathbf{x}_0\|_2^2$ , as a function of the number of non-zeroes in the solution,  $\|\hat{\mathbf{x}}\|_0$ . Figure 8.1 shows such graphs, averaged over 200 experiments. The figure also presents the oracle performance, obtained by a simple projection of the vector  $\mathbf{b}$  on the columns of  $\mathbf{A}$  in the true support. We see that DS and BP are nearly equivalent, and their best performance is obtained for a cardinality of  $\approx 25$ , with an error that is 4 times larger than the oracle.

Recall that the two methods are biased towards smaller non-zero entries, due to the  $\ell_1$ -penalty. Thus, adopting only the support of the solution and solving for the non-zero values by Least-Squares, we may obtain improved results, and especially so if the found support is near-correct. The projected errors are also plotted in Figure 8.1, and indeed, both methods gain from this modification. This time, the best performance for both methods is obtained for a cardinality of  $\approx 10$ , which is the true one. Again, both methods are nearly equivalent, and getting closer (factor 2.8 between the DS/BP and the oracle errors) to the oracle.

Figure 8.2 is very similar to the one shown in 8.1, but with errors measured as  $\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{A}\mathbf{x}_0\|_2^2 / \|\mathbf{A}\mathbf{x}_0\|_2^2$ . This difference is important in some cases, where the objective is not the recovery of the sparse representation  $\mathbf{x}_0$  but rather the separation of the





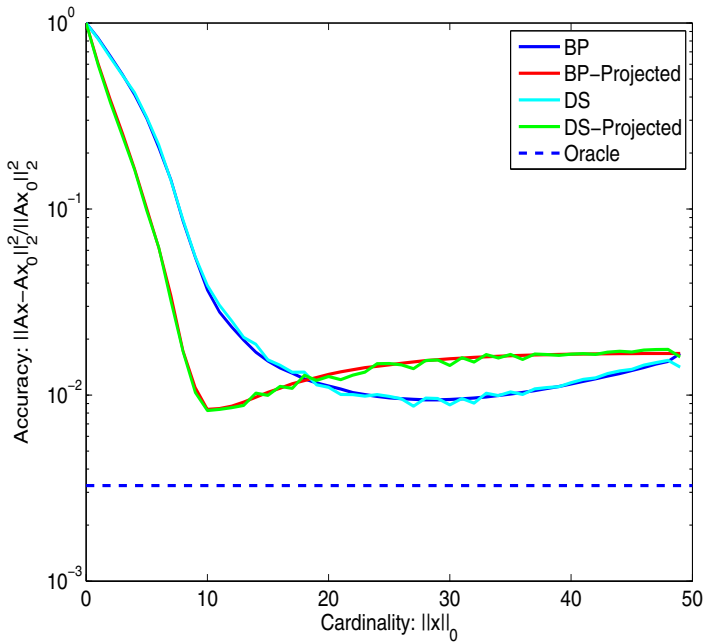
**Fig. 8.1** Accuracy of the solution, measured by  $\|\hat{\mathbf{x}} - \mathbf{x}_0\|_2^2 / \|\mathbf{x}_0\|_2^2$ , versus its cardinality for the Dantzig-Selector and the Basis-Pursuit. The different points in these graphs are created by varying the threshold of the algorithm. The graph shows the final results of the two algorithms and also their Least-Squares projections onto the found support.

clean signal  $\mathbf{A}\mathbf{x}_0$  from the noise  $\mathbf{e}$ . As can be seen, the overall behavior is very similar to the graph in Figure 8.1, leading to the same conclusions.

Could we rely on the above experiment and conclude that BP and DS are nearly equivalent in general? The answer is certainly not. First, this experiment is one among many that could have been done, and it may well be that a different problem-setting would lead to different conclusions. The question of the relative performance between DS and BP is still an open question, which would occupy both theoreticians and practitioners in coming years. One interesting question that comes out from the above experiment is whether the combination of the two (i.e., solving an optimization problem with the two constraints) could lead to better results.

## 8.6 Summary

The Dantzig-Selector is a relatively new pursuit technique in the realm of sparse representation modeling, although earlier signs of it appeared in the literature in 2001 (see the work by Starck, Donoho, and Candes). Many questions regarding this



**Fig. 8.2** Accuracy of the solution, measured by  $\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{A}\mathbf{x}_0\|_2^2 / \|\mathbf{A}\mathbf{x}_0\|_2^2$ , versus its cardinality for the Dantzig-Selector and the Basis-Pursuit. The various graphs are very similar to those shown in Figure 8.1.

tool remain open at this stage – its relation to estimation theory, its comparative performance versus other pursuit techniques, limits on its *average* performance, numerical techniques to practically and efficiently solve it, methods to improve it by merging its force with other existing techniques, and more. These topics and others would occupy many interested scientists and engineers in the coming years.

One important property that has drawn our attention to the Dantzig-Selector is the ability to claim a near-oracle performance for it. Similar results emerged recently, suggesting similar claims for the Basis-Pursuit, the OMP, and even the thresholding algorithm. Still, all these results are of a worst-case flavor, and as such, there is much room for future improvements.

## Further Reading

1. Z. Ben-Haim, Y.C. Eldar, and M. Elad, Coherence-based performance guarantees for estimating a sparse vector under random noise, submitted to *IEEE Transactions on Signal Processing*, 2009.

2. P.J. Bickel, Y. Ritov, and A. Tsybakov, Simultaneous analysis of Lasso and Dantzig selector, to appear in *Ann. Statist.*, 2008.
3. E.J. Candès and T. Tao, Decoding by linear programming, *IEEE Trans. on Information Theory*, 51(12):4203–4215, December 2005.
4. E.J. Candès and T. Tao, The Dantzig selector: Statistical estimation when  $p$  is much larger than  $n$ , *Annals of Statistics*, 35(6):2313–2351, June 2007.
5. G.M. James, P. Radchenko, and J. Lv, DASSO: connections between the Dantzig selector and lasso, *Journal of the Royal Statist. Soc. B*, 71(1):127–142, 2009.
6. N. Meinshausen, G. Rocha, and B. Yu, Discussion: A tale of three cousins: Lasso, L2Boosting and Dantzig, *Ann. Statist.*, 35(6):2373–2384, 2007.
7. J.L. Starck, D.L. Donoho and E. Candès, Very high quality image restoration, in SPIE conference on Signal and Image Processing: Wavelet Applications in Signal and Image Processing IX, Vol 4478, 2001.

**Part II**  
**From Theory to Practice – Signal and  
Image Processing Applications**



## Chapter 9

# Sparsity-Seeking Methods in Signal Processing

All the previous chapters have shown us 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, contrary to expectation, can also be computationally tractable. We now turn to discuss the applicability of these ideas to signal and image processing. As we argue in this chapter, modeling of informative signals is possible using their sparse representation over a well-chosen dictionary. This will give rise to linear systems and their sparse solution, as dealt with earlier.

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

### 9.1 Priors and Transforms for Signals

Consider a family of signals – a set of vectors  $\mathcal{Y} = \{\mathbf{y}_j\}_j \in \mathbb{R}^n$ . To make our discussion more concrete, we shall assume hereafter that each such signal is a  $[\sqrt{n}$ -by- $\sqrt{n}]$  pixels image patch, representing natural and typical image content. The same discussion can be directed towards other data sources, such as sound signals, seismic data, medical signals, financial time series, and more.

While image patches are very diverse vectors in  $\mathbb{R}^n$ , we have no reason to assume that they occupy the entire space. Said more accurately, assuming that the pixels in  $\mathbf{y}$  have values in the range  $[0, 1)$ , these image patches do not populate or sample the hyper-cube  $[0, 1)^n \subset \mathbb{R}^n$  uniformly. For example, we know that spatially smooth patches occur much more often in images, whereas highly non-smooth and disorganized image content is highly unlikely to exist. This mode of reasoning naturally leads to the Bayesian framework of imposing a Probability-Density-Function (PDF) on the signals – a ‘prior’ distribution  $P(\mathbf{y})$ .

Priors are extensively used in signal processing, serving in inverse problems, compression, anomaly detection, and more. This is because they provide a systematic way of measuring the probability of a signal vector. For example, consider the

noise cleaning (denoising) problem: a given image  $\mathbf{y}$  is known to be a noisy version of a clean image  $\mathbf{y}_0$ , contaminated by an additive perturbation vector  $\mathbf{v}$ , known to have a finite energy  $\|\mathbf{v}\|_2 \leq \epsilon$ , i.e.,  $\mathbf{y} = \mathbf{y}_0 + \mathbf{v}$ . The unknown image  $\mathbf{y}_0$  must be in the ball  $\|\mathbf{y}_0 - \mathbf{y}\|_2 \leq \epsilon$ . The optimization problem

$$\max_{\hat{\mathbf{y}}} P(\hat{\mathbf{y}}) \quad \text{subject to} \quad \|\hat{\mathbf{y}} - \mathbf{y}\|_2 \leq \epsilon \quad (9.1)$$

leads to the most probable image  $\hat{\mathbf{y}}$  in this sphere, which is an effective estimate of  $\mathbf{y}_0$ . This way the prior is exploited for solving the denoising problem. The above formulation of the denoising problem is in fact that Maximum-A-posteriori-Probability (MAP) estimator, and more detailed and accurate version of it will be introduced later on in Chapter 11.

Much effort has been allocated in the signal and image processing communities for forming priors as closed-form expressions. One very common way to construct  $P(\mathbf{y})$  is to guess its structure based on intuitive expectations from the data content. For example, the Gibbs distribution  $P(\mathbf{y}) = \text{Const} \cdot \exp\{-\lambda\|\mathbf{L}\mathbf{y}\|_2^2\}$  uses a Laplacian matrix (defined as the linear space-invariant operation that applies the Laplacian filter to the image  $\mathbf{y}$ ) to give an evaluation of the probability of the image  $\mathbf{y}$ . In such a prior, smoothness, measured by the Laplacian operator, is used for judging the probability of the signal.

This prior is well-known and extensively used in signal and image processing, and is known to be related to both *Tikhonov regularization* on one hand, and to *Wiener filtering* on the other. Returning to Equation (9.1), this prior leads to an optimization problem of the form

$$\min_{\hat{\mathbf{y}}} \|\mathbf{L}\hat{\mathbf{y}}\|_2^2 \quad \text{subject to} \quad \|\hat{\mathbf{y}} - \mathbf{y}\|_2 \leq \epsilon, \quad (9.2)$$

which can be converted to

$$\min_{\hat{\mathbf{y}}} \|\mathbf{L}\hat{\mathbf{y}}\|_2^2 + \mu\|\hat{\mathbf{y}} - \mathbf{y}\|_2^2, \quad (9.3)$$

where we have replaced the constraint by an equivalent penalty. The solution is easily obtained as

$$\hat{\mathbf{y}} = \mu \left[ \mathbf{L}^T \mathbf{L} + \mu \mathbf{I} \right]^{-1} \mathbf{y}. \quad (9.4)$$

The value of  $\mu$  should be chosen such that the constraint  $\|\hat{\mathbf{y}} - \mathbf{y}\|_2 \leq \epsilon$  is met.

A similar problem, where  $\mathbf{y} = \mathbf{H}\mathbf{y}_0 + \mathbf{v}$ , with  $\mathbf{H}$  representing a linear degradation operator (e.g., blur), leads to the solution

$$\hat{\mathbf{y}} = \mu \left[ \mathbf{L}^T \mathbf{L} + \mu \mathbf{H}^T \mathbf{H} \right]^{-1} \mathbf{y}. \quad (9.5)$$

This is the well-known *Wiener filter*, which for space-invariant and cyclic operators,  $\mathbf{L}$  and  $\mathbf{H}$ , it can be solved directly in the frequency-domain.

The above specific prior stressing smoothness is known to cause blurring of the image when used in various image enhancement and restoration tasks. The remedy for this problem was found to be the replacement of the  $\ell_2$ -norm by a more robust measure, such as an  $\ell_1$ -norm, that allows heavy tails for the distribution of the values of  $\mathbf{Ly}$ . Thus, a prior of the form  $P(\mathbf{y}) = \text{Const} \cdot \exp\{-\lambda\|\mathbf{Ly}\|_1\}$  is far more versatile and thus became popular in recent years. Similar to this option is the Total-Variation (TV) prior (Rudin, Osher, and Fatemi, 1993) that also promotes smoothness, but differently, by replacing the Laplacian with gradient norms, thereby using first-derivatives rather than second ones. Note that with the intuition gained in previous chapters we see that the adoption of the  $\ell_1$ -norm leads to an enforcement of sparsity of the signal/image derivatives.

A different property that can be used for constructing a prior is assuming a structure on the signal's transform-coefficients. One such example is the JPEG compression algorithm, which relies on the fact that 2D-DCT coefficients of small image-patches tend to behave in a predicted way (being concentrated around the origin). A different example refers to the wavelet transform of signals and images, where the coefficients are expected to be sparse, most of them tending to zero while few remain active.

The above two examples are an excellent illustration for the notion of linear versus non-linear approximation. In linear approximation, we apply a transform and choose a pre-specified set of  $M$  coefficients to approximate it, just as done with the 2D-DCT coefficients above,<sup>1</sup> and PCA in general (see next). A non-linear approximation takes the dominant  $M$  coefficients, regardless of their location. Using these  $M$ -term approximations, the non-linear approach is expected to lead to a better representation of the original signal. Indeed, this is the approach used on the wavelet coefficients.

Lets us look more closely at the wavelet transform, and how it leads to a prior. For a signal  $\mathbf{y}$ , the wavelet transform is given by  $\mathbf{Ty}$  where the matrix  $\mathbf{T}$  is a specially designed orthogonal matrix that contains in its rows spatial derivatives of varying scale, thereby providing what is known as “multi-scale” analysis of the signal. Therefore, the prior in this case becomes  $P(\mathbf{y}) = \text{Const} \cdot \exp\{-\lambda\|\mathbf{Ty}\|_p^p\}$  with  $p \leq 1$  to promote sparsity. Here the resemblance to the Total-Variation and the Laplacian priors becomes evident, as in all these cases derivatives of some sort and a robust measure are combined in forming the prior distribution  $P(\mathbf{y})$ .

Wavelet-based priors shed light on a rich family of signal priors that assign likelihood for an image based on the behavior of its transform coefficients  $\mathbf{Ty}$ . In the signal and image processing literature, such priors were postulated in conjunction with a variety of transforms, such as the Discrete-Fourier-Transform (DFT), the discrete cosine transform (DCT), the Hadamard, and even data-dependent transforms such as the Principal-Component-Analysis (PCA, also known in the signal processing community as the Karhunen-Loeve-Transform - KLT).

<sup>1</sup> Actually, JPEG operates slightly different, by considering the leading DCT coefficients to be active, but allowing some outliers, and in that respect, this is not an exact linear approximation.



Constructing a PCA-based prior starts by gathering a database of signal examples,  $\mathcal{Y} = \{\mathbf{y}_j\}_j \in \mathbb{R}^n$ , as described above. We compute the center of this cloud of  $n$ -dimensional points,  $\mathbf{c} = \frac{1}{N} \sum_j \mathbf{y}_j$ , as the mean of all the examples. Then we compute the auto-correlation matrix  $\mathbf{R} = \frac{1}{N} \sum_j (\mathbf{y}_j - \mathbf{c})(\mathbf{y}_j - \mathbf{c})^T$ . The effective prior is given as the multivariate Gaussian distribution

$$P(\mathbf{y}) = \text{Const} \cdot \exp \left\{ \frac{1}{2} (\mathbf{y} - \mathbf{c})^T \mathbf{R}^{-1} (\mathbf{y} - \mathbf{c}) \right\}. \quad (9.6)$$

This model assumes that the cloud of examples  $\mathcal{Y}$  behaves like a high-dimensional ellipsoid in  $\mathbb{R}^n$ , where  $\mathbf{c}$  is its center and  $\mathbf{R}$  describes its shape. If we assume that  $\mathbf{c} = \mathbf{0}$  (i.e., if we deal with signals shifted to the origin), then this model is very close to the Tikhonov regularization mentioned earlier, by assigning  $\mathbf{L}^T \mathbf{L} = \mathbf{R}^{-1}$ . This model is the one that leads to the 2D-DCT in JPEG mentioned above.

One may adopt a common point of view towards all these signal priors, regarding them as mere attempts to describe a random generator machine  $\mathcal{M}$  that supposedly generates the signals of interest. However, in all these cases, while the relative<sup>2</sup> evaluation of the probability for a given signal is easy via the  $P(\mathbf{y})$  formulae, drawing a random sample from such distributions is quite difficult. This brings us to the *Sparse-Land* model, which is a way for synthesizing signals according to a prior defined by a transform.

## 9.2 The Sparse-Land Model

Let us return to the linear system  $\mathbf{A}\mathbf{x} = \mathbf{y}$  and interpret it as a way of constructing signals  $\mathbf{y}$ . Every column in  $\mathbf{A}$  is a possible signal in  $\mathbb{R}^n$  – we refer to these  $m$  columns as atomic signals, and the matrix  $\mathbf{A}$  displays a dictionary of atoms. One can consider  $\mathbf{A}$  as the periodic table of the fundamental elements in the “chemistry” that describes our signals.

The multiplication of  $\mathbf{A}$  by a sparse vector  $\mathbf{x}$  with  $\|\mathbf{x}\|_0^0 = k_0 \ll n$  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” thereof were used for its construction. This process of combining atoms linearly to form a signal (think of it as a molecule in the chemistry of our signals) may be referred to as *atomic-composition*.

Consider all the possible sparse representation vectors with cardinality  $\|\mathbf{x}\|_0^0 = k_0 \ll n$ , and assume that this set of  $\binom{m}{k_0}$  possible cardinalities are drawn with uniform probability. Assume further that the non-zero entries in  $\mathbf{x}$  are drawn from the zero-mean Gaussian distribution  $\text{Const} \cdot \exp\{-\alpha x_i^2\}$ . This gives us a complete definition of the PDF of  $\mathbf{x}$ , and thus also for the way signals  $\mathbf{y}$  are generated – this constitutes the

<sup>2</sup> Due to the need for the normalization coefficient.

random signal generator  $\mathcal{M}_{\{\mathbf{A}, k_0, \alpha\}}$ . Note that the signal  $\mathbf{y}$  is the outcome of a mixture of Gaussians, each of dimension  $k_0$  with equal probability.

A modification that can be introduced into the above model is the replacement of the fixed cardinality with a random one. In such case, rather than assuming that every signal is composed of a fixed  $k_0$  atoms, one could assume that the cardinality itself is random, with emphasis on sparse representations, such as a probability proportional to  $\exp\{-k\}$ . Such a change is not significant to our discussion here.

An important further addition to the definition of the signal generation model  $\mathcal{M}$  could be postulating a random perturbation (noise) vector  $\mathbf{e} \in \mathbb{R}^n$  with bounded power  $\|\mathbf{e}\|_2 \leq \epsilon$ , such that  $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{e}$ . Such an additive perturbation is useful because of two important reasons:

- The model  $\mathcal{M}$  imposes too strong restrictions on the signals, and using it in practical applications will necessarily lead to a mismatch between actual observed signals and their imposed model. The introduction of the  $\epsilon$  perturbation mends this problem.
- The case  $\epsilon = 0$  implies that the volume of signals generated by  $\mathcal{M}$  is of “zero measure” within  $\mathbb{R}^n$ . This is because we assume a finite number of possible supports,  $\binom{m}{k_0}$ , each leading to a  $k_0$ -dimensional subspace signals in the  $\mathbb{R}^n$  space. If  $\epsilon$  increases, this set assumes a small but positive volume, via the “thickened” cloud around those subspaces.

With the addition of such a perturbation, we refer hereafter to the *Sparse-Land* model as  $\mathcal{M}(\mathbf{A}, k_0, \alpha, \epsilon)$ .

At this point the reader might be troubled by the arbitrariness of the *Sparse-Land* model, and wonder why should such a description fit signals of interest? In order to partially answer this question, we should note that despite the very different appearance of this model, it is in fact tightly related to methods we discussed earlier in Section 9.1. Consider, for example, the case where  $\mathbf{A}$  is square and invertible. The relation  $\mathbf{A}\mathbf{x} = \mathbf{y}$  can be converted to be  $\mathbf{x} = \mathbf{A}^{-1}\mathbf{y} = \mathbf{T}\mathbf{y}$ , where  $\mathbf{T} = \mathbf{A}^{-1}$ . This means that the sparse representation vector we use here plays a role of a transform-coefficients vector. Thus, a model of the sort  $P(\mathbf{y}) = \text{Const} \cdot \exp\{-\lambda \|\mathbf{T}\mathbf{y}\|_p^p\}$ , as described earlier, is in fact equivalent to a model  $P(\mathbf{y}) = \text{Const} \cdot \exp\{-\lambda \|\mathbf{x}\|_p^p\}$ , which describes the probability of the signal  $\mathbf{y}$  in terms of the sparsity of its representation  $\mathbf{x}$ , just as *Sparse-land* does. Does this imply that *Sparse-Land* is included in the transform-based models mentioned earlier? The answer is negative, as we argue towards the end of this chapter, when we discuss analysis versus synthesis models.

### 9.3 Geometric Interpretation of Sparse-Land

In order to gain a geometric insight into the *Sparse-Land* model and some alternatives to it, we return to the beginning of this chapter, considering a large corpus of signal-examples (vectors)  $\mathcal{Y} = \{\mathbf{y}_j\}_j \in \mathbb{R}^n$ . We concentrate on one arbitrary signal

$\mathbf{y}_0 \in \mathcal{Y}$  and its  $\delta$ -neighborhood, and aim to study this neighborhood's behavior in the  $n$ -th dimensional space.

For small enough value of  $\delta$ , moving from  $\mathbf{y}_0$  along the directions  $\mathbf{e} = \mathbf{y} - \mathbf{y}_0$  represent small permissible perturbation directions that lead to feasible signals. The question we pose is: do those fill the entire  $\mathbb{R}^n$  space? We denote this set of permissible directions as

$$\mathcal{Q}_{\mathbf{y}_0}^\delta = \{\mathbf{e} \mid \mathbf{e} = \mathbf{y} - \mathbf{y}_0, \text{ where } \mathbf{y} \in \mathcal{Y} \text{ and } \|\mathbf{y} - \mathbf{y}_0\|_2 \leq \delta\}. \quad (9.7)$$

Gathering all vectors  $\mathbf{e} \in \mathcal{Q}_{\mathbf{y}_0}^\delta$  into an  $n \times |\mathcal{Q}_{\mathbf{y}_0}^\delta|$  matrix  $\mathbf{E}_{\mathbf{y}_0}$ , we aim to study the behavior of its singular-values. We will be interested in families of structured signals, for which the effective rank of such matrices is  $k_{\mathbf{y}_0} \ll n$ , for any  $\mathbf{y}_0 \in \mathcal{Y}$ . This is equivalent to the statement that the  $k_{\mathbf{y}_0} + 1$  singular-value and beyond in these matrices tend to be near-zero and are therefore negligible.

Signals satisfying the above local low-dimensionality assumption essentially give a local behavior that is approximately a linear subspace of dimension  $k_{\mathbf{y}_0}$ , shifted around  $\mathbf{y}_0$ . While this subspace and its dimension may vary from one point to another in the signal space, all of them are characterized as being far smaller than the ambient  $n$  dimensions. Experimental studies show that most signals we work with, such as images, audio, and more, follow such a structure, and this can be exploited in processing them.

As a simple example for how to exploit this structure, consider again the denoising problem. A clean signal  $\mathbf{y}_0$  is to be recovered from a noisy version of it  $\mathbf{y}$ , where  $\|\mathbf{y} - \mathbf{y}_0\|_2 \leq \delta_0 \ll \delta$ . Suppose that from the complete set  $\mathcal{Y}$  we extract all examples that are in the  $\delta$ -neighborhood of  $\mathbf{y}$ . We compute the average vector of this set,  $\mathbf{c}$ , and subtract it,<sup>3</sup> getting a good approximation to  $\mathcal{Q}_{\mathbf{y}_0}^\delta$ , since  $\delta_0 \ll \delta$ . The accuracy of this approximation is dictated by the assumption that the local permissible subspace changes smoothly, enabling the assumption  $\delta \gg \delta_0$ .

The knowledge of  $\mathcal{Q}_{\mathbf{y}_0}^\delta$  implies the knowledge of the  $k_{\mathbf{y}_0}$ -dimensional subspace spanning its members (the columns of  $\mathbf{E}_{\mathbf{y}_0}$ ). Let us denote this subspace by the  $n \times k_{\mathbf{y}_0}$  matrix  $\mathbf{Q}_{\mathbf{y}_0}$ , such that its columns form an orthogonal set of vectors. The permissible signals that may explain the noisy signal  $\mathbf{y}$  can be formulated as a linear combination of the columns of  $\mathbf{Q}_{\mathbf{y}_0}$  with a shift  $\mathbf{c}$ , i.e., the possible solutions are of the form  $\mathbf{Q}_{\mathbf{y}_0} \mathbf{z} + \mathbf{c}$ . Thus, an effective denoising is obtained by computing the projection of  $\mathbf{y}$  onto this affine subspace,

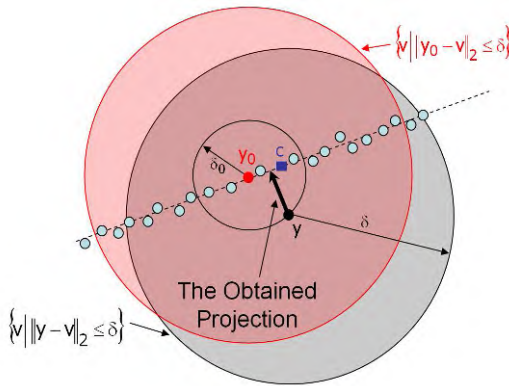
$$\mathbf{z}_{opt} = \arg \min_{\mathbf{z}} \|\mathbf{Q}_{\mathbf{y}_0} \mathbf{z} + \mathbf{c} - \mathbf{y}\|_2^2 = \mathbf{Q}_{\mathbf{y}_0}^T (\mathbf{y} - \mathbf{c}), \quad (9.8)$$

leading to

$$\hat{\mathbf{y}} = \mathbf{Q}_{\mathbf{y}_0} \mathbf{Q}_{\mathbf{y}_0}^T (\mathbf{y} - \mathbf{c}) + \mathbf{c}. \quad (9.9)$$

---

<sup>3</sup> The set gathered forms as affine subspace, and the average computed is the estimation of its origin.



**Fig. 9.1** A graphic demonstration of the denoising process that uses local neighbors in order to compute the center  $\mathbf{c}$ , and the projection directions  $\mathbf{Q}_y$ .

The essence of this process is the assumption that any content orthogonal to the directions in  $\mathbf{Q}_{y_0}$  is known to be pure noise and thus should be discarded. Figure 9.1 presents this explanation graphically.

Based on the above description, it is tempting to suggest a modeling of a signal source by holding many instances of example signals  $\mathcal{Y}$ , and using them directly in the denoising process. We can repeat the above process, of seeking the  $\delta$ -close neighbors, estimating the local subspace, and projecting onto it. This is known as a local Principal-Component-Analysis (PCA) modeling. Its major drawbacks are the need for a very large set of examples in order to guarantee effective modeling, and the high-complexity algorithm for cleaning any signal by sweeping through all this database.

In the above local-PCA model,  $\mathbf{Q}_{y_0}$  are the principal directions of the signal for the location  $y_0$ , and those are permitted to vary freely as a function of  $y_0$ . In general, a complete model of this sort requires either a storage of  $\mathcal{Y}$  and extraction of  $\mathbf{Q}_{y_0}$  per need, or a storage of all the possible unitary bases, gathered off-line for every  $y \in \mathcal{B}$ . Both options are prohibitive in most cases.

If we can further assume that a finite and small number of such projection-matrices and center-vectors,  $\{\mathbf{Q}_p, \mathbf{c}_p\}_{p=1}^P$  (say few hundreds or thousands), cover all possible cases, this model can be made more efficient. This calls for a (possibly fuzzy) clustering of the signals  $y \in \mathcal{Y}$  to subgroups that correspond each to a different matrix  $\mathbf{Q}_p$  and a center  $\mathbf{c}_p$ . Once a signal is to be processed using this model, we start by assigning it to one of these groups (by searching its nearest-neighbor in  $\mathcal{Y}$ ), and then attaching the proper projection-matrix and center vector  $\{\mathbf{Q}_p, \mathbf{c}_p\}$  to it for further processing. Such a model has been proposed by Yi Ma and others, under the name *Generalized PCA*, and shown to be very effective in grasping the images' behavior.

The *Sparse-Land* model is one step further in an attempt to make such local model more concise. Assuming for simplicity that all these local subspaces are of

the same dimensionality,  $k$ , using sparse representations of cardinality  $k$  over a dictionary  $\mathbf{A}$  with  $m$  columns, we have at our disposal  $\binom{m}{k}$  subspaces, just by allowing  $k$ -sparse representations with all the possible supports. This way, our dictionary holds very efficiently all the key directions required to represent the signal anywhere in  $\mathbb{R}^n$ , while enabling a very large number of possibilities. Clearly, though, this comes at the price of further restricting the structure of the model by assuming that the atoms in the dictionary are principal directions shared by many sub-spaces.

For completeness of the above presentation, we should mention yet another approach that could benefit from the availability of the example set  $\mathcal{Y}$ . Given a noisy version of  $\mathbf{y}_0$ , denoted by  $\mathbf{y}$ , such that  $\|\mathbf{y} - \mathbf{y}_0\|_2 \leq \delta_0$ , we can simply search for the nearest-neighbor for  $\mathbf{y}$  in  $\mathcal{Y}$ , and consider this as the denoised result. This direct example-based technique is very appealing, but it would work well only under the very restricting assumption that  $\mathcal{Y}$  is sampled very densely, so as to hold the clean signal  $\mathbf{y}_0$  or others very close to it.

This approach could be considered as an extreme sparse representation modeling of the signal  $\mathbf{y}$ , such that the  $\delta$ -local neighborhood serves as its local (and varying) dictionary, and we allow for use of one atom with a coefficient being 1 to represent it.

Alternatively, one could seek the  $K$ -nearest-neighbor signals to  $\mathbf{y}$  in  $\mathcal{Y}$  and average them. This tends towards the vector  $\mathbf{c}$  used in the above affine modeling as the estimated center, and this result is inferior to the local-PCA method. Thus, the benefits that parameterized models, such as clustered-local-PCA and *Sparse-Land*, bring are the ability to work with reasonably modest example sets, and a stable estimation of the local characteristics.

## 9.4 Processing of Sparsely-Generated Signals

How do we practice signal processing for *Sparse-Land* signals? Suppose we have a signal  $\mathbf{y}$  which has been generated from the model  $\mathcal{M}(\mathbf{A}, k_0, \alpha, \epsilon)$  and the parameters of the model are known. There are numerous signal processing tasks that could be of interest to us; let us discuss them and see how sparsity-seeking representations might enter in.

- **Analysis:** Given  $\mathbf{y}$ , can we determine the underlying vector  $\mathbf{x}_0$  which generated it? This process may be called *atomic-decomposition*, as it leads us to infer the individual atoms that actually generated  $\mathbf{y}$ . Clearly, the true underlying representation obeys  $\|\mathbf{A}\mathbf{x}_0 - \mathbf{y}\|_2 \leq \epsilon$ , but there will be many other vectors  $\mathbf{x}$  generating similarly good approximations to  $\mathbf{y}$ . Suppose we can actually solve  $(P_0^\epsilon)$ :

$$(P_0^\epsilon) : \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2 \leq \epsilon.$$

Under our assumptions, the solution  $\mathbf{x}_0^\epsilon$  of this problem, though not necessarily the underlying  $\mathbf{x}_0$ , will also be sparse and have  $k_0$  or fewer non-zeros. Results

described in earlier chapters show that if  $k_0$  is small enough, then the solution of  $(P_0^\epsilon)$  is at most  $O(\epsilon)$  away from  $\mathbf{x}_0$ .

- **Compression:** Nominally,  $\mathbf{y}$  requires a description by  $n$  numbers. However, if we can solve  $(P_0^\delta)$  where  $\delta \geq \epsilon$ , then the resulting solution  $\mathbf{x}_0^\delta$  affords an approximation  $\hat{\mathbf{y}} = \mathbf{A}\mathbf{x}_0^\delta$  to  $\mathbf{y}$  using at most  $2k_0$  scalars, with an approximation error of at most  $\delta$ . By increasing  $\delta$  we obtain a deeper compression with fewer non-zeros, and a larger approximation error. This way, we obtain a rate-distortion curve for a compression mechanism.
- **Denoising:** Suppose that we observe, not  $\mathbf{y}$ , but instead a noisy version  $\tilde{\mathbf{y}} = \mathbf{y} + \mathbf{v}$  where the noise is known to obey  $\|\mathbf{v}\|_2 \leq \delta$ . If we can solve  $(P_0^{\delta+\epsilon})$ , then the resulting solution  $\mathbf{x}_0^{\delta+\epsilon}$  will have at most  $k_0$  non-zeros; our earlier results show that if  $k_0$  is small enough, then  $\mathbf{x}_0^{\delta+\epsilon}$  is at most  $O(\epsilon + \delta)$  away from  $\mathbf{x}_0$ . Once found, we obtain an approximation of our cleaned signal by  $\mathbf{A}\mathbf{x}_0^{\delta+\epsilon}$ .
- **Inverse Problems:** Even more generally, suppose that we observe, not  $\mathbf{y}$ , but a noisy indirect measurement of it,  $\tilde{\mathbf{y}} = \mathbf{H}\mathbf{y} + \mathbf{v}$ . Here the linear operator  $\mathbf{H}$  could represent blurring, masking (which leads to in-painting), projection, down-scaling, or some other kind of linear degradation, and  $\mathbf{v}$  is noise as before. If we could solve

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\tilde{\mathbf{y}} - \mathbf{H}\mathbf{A}\mathbf{x}\|_2 \leq \delta + \epsilon,$$

then we would expect to identify directly the sparse components of the underlying signal and obtain an approximation  $\mathbf{A}\mathbf{x}_0^{\delta+\epsilon}$ .

- **Compressed-Sensing:** For signals which are sparsely generated, one can obtain good reconstructions from a reduced number of measurements – thereby compressing the *sensing process* rather than the traditionally sensed *data*. Let  $\mathbf{P}$  be a random  $j_0 \times n$  matrix with Gaussian iid entries, and suppose that it is possible to directly measure  $\mathbf{c} = \mathbf{P}\mathbf{y}$ , which has  $j_0 \ll n$  entries, rather than sensing  $\mathbf{y}$ , which requires  $n$  values. Attempt recovery by solving

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{c} - \mathbf{P}\mathbf{A}\mathbf{x}\|_2 \leq \epsilon,$$

to obtain the sparse representation and then synthesizing an approximate reconstruction using  $\mathbf{A}\mathbf{x}_0^\epsilon$ . Results from compressed-sensing suggest that if  $j_0 > 2k_0$ , this recovery is likely to succeed with high-probability.

- **Morphological Component Analysis (MCA):** Suppose that the observed signal is a superposition of two different sub-signals  $\mathbf{y}_1, \mathbf{y}_2$  (i.e.,  $\mathbf{y} = \mathbf{y}_1 + \mathbf{y}_2$ ) where  $\mathbf{y}_1$  is sparsely generated using model  $\mathcal{M}_1$  and  $\mathbf{y}_2$  is sparsely generated using model  $\mathcal{M}_2$ . Can we separate the two sources? Such source separation problems are fundamental in processing of acoustic signals, for example in the separation of speech from impulsive noise by *Independent Component Analysis* (ICA) algorithms. Turning to the signal model presented here, if we could solve

$$\min_{\mathbf{x}_1, \mathbf{x}_2} \|\mathbf{x}_1\|_0 + \|\mathbf{x}_2\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{A}_1\mathbf{x}_1 - \mathbf{A}_2\mathbf{x}_2\|_2^2 \leq \epsilon_1^2 + \epsilon_2^2,$$

the resulting solution  $(\mathbf{x}_1^\epsilon, \mathbf{x}_2^\epsilon)$  would generate a plausible solution  $\hat{\mathbf{y}}_1 = \mathbf{A}\mathbf{x}_1^\epsilon$ ,  $\hat{\mathbf{y}}_2 = \mathbf{A}\mathbf{x}_2^\epsilon$  to the separation problem.

There have been several successful trials of this idea by Starck and others, first in acoustics and later in image processing. An appealing image processing application that relies on MCA is inpainting, where missing pixels in an image are filled-in, based on a sparse representation of the existing pixels. MCA is necessary because the piecewise smooth (cartoon) and texture contents of the image must be separated as part of this recovery process.

A wide range of other applications including encryption, watermarking, scrambling, target detection, recognition, feature extraction, and more, can also be envisioned. All these application ideas call for the solution of  $(P_0^\delta)$ , or variants. We have intentionally described all these proposed applications in the conditional mood, since in general  $(P_0^\delta)$  is not known to be tractable or even well defined. However, our earlier discussion of  $(P_0^\delta)$  shows that the problem, under the right conditions, is sensible and can be approximately solved by practical algorithms.

## 9.5 Analysis Versus Synthesis Signal Modeling

Just before leaving this chapter, we return to the core of the *Sparse-Land* model definition, and contrast it with an interesting alternative. *Sparse-Land* is built around a *generative* model for the signal  $\mathbf{y}$ , by constructing it as a linear combination of atoms,  $\mathbf{A}\mathbf{x} = \mathbf{y}$ . An alternative, more in line with the priors mentioned in the beginning of this chapter, is an analysis modeling that assigns a probability  $P(\mathbf{y})$  to  $\mathbf{y}$  by analyzing it through an analysis operator  $\mathbf{T}$ , effectively computing the *analysis-representation*  $\mathbf{T}\mathbf{y}$ .

Consider, as an example, denoising with *Sparse-Land*. Solving  $(P_0^{\epsilon+\delta})$  as proposed above,

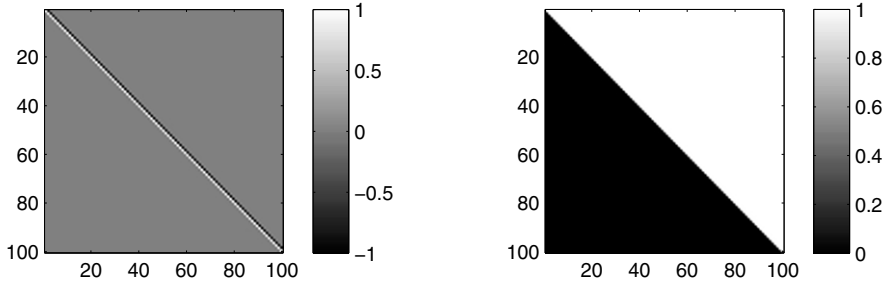
$$\hat{\mathbf{y}} = \mathbf{A} \cdot \arg \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{A}\mathbf{x}\|_2 \leq \epsilon + \delta,$$

leads to denoising. Assume now that  $\mathbf{A}$  is square and invertible and define  $\mathbf{z} = \mathbf{A}\mathbf{x}$ . The above denoising process can be re-written equivalently as

$$\hat{\mathbf{y}} = \arg \min_{\mathbf{z}} \|\mathbf{A}^{-1}\mathbf{z}\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{z}\|_2 \leq \epsilon + \delta.$$

Thus, the matrix  $\mathbf{A}^{-1}$  becomes an analysis operator  $\mathbf{T}$ , and the formulation obtained here coincides with the one that appeared in Equation (9.1).

A classic example for such a pair of analysis and synthesis formulations that are equivalent is the 1D Total-Variation analysis operator and the Heaviside synthesis dictionary. The 1D Total-Variation performs a simple derivative by convolving the signal with the filter  $[+1, -1]$ . Let us assume that at the right border of the signal, we proceed with zero padding (implying that the rightmost sample is multiplied by 1). This results with an  $n \times n$  Toeplitz matrix with the main-diagonal containing  $(+1)$ -



**Fig. 9.2** The matrix  $\mathbf{T}$  for the 1D Total-Variation (left) and the corresponding synthesis operator  $\mathbf{A}$ , the Heaviside basis (right).

es, and the first diagonal to the right containing  $(-1)$ -es. We compute its inverse,  $\mathbf{A} = \mathbf{T}^{-1}$ , and the resulting matrix gives the Heaviside basis. Its columns are simple step functions in all possible  $n$  locations. Figure 9.2 presents these two matrices, both of size  $100 \times 100$ .

The equivalence between these two operators is natural since in analysis terms, we seek a piecewise smooth signal that shows little energy in the sum of the absolute derivatives. In a generative way, this implies that such a signal is built as a linear combination of few step functions. This way, we see that our proposed model can cover known Markov-Random-Field (MRF) models that are extensively used in signal and image processing.

The above may suggest that analysis and synthesis based modelings may become equivalent in the general case, where  $\mathbf{A}$  is a full-rank matrix of size  $n \times m$  with  $n < m$ , with the following two presumably equivalent formulations for denoising:

$$\begin{aligned} (P_{\text{Synthesis}}) \quad \hat{\mathbf{y}}_s &= \mathbf{A} \cdot \arg \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{Ax}\|_2 \leq \epsilon + \delta \\ (P_{\text{Analysis}}) \quad \hat{\mathbf{y}}_a &= \arg \min_{\mathbf{z}} \|\mathbf{Tz}\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{z}\|_2 \leq \epsilon + \delta, \end{aligned}$$

for  $\mathbf{A} = \mathbf{T}^+ = (\mathbf{T}^T \mathbf{T})^{-1} \mathbf{T}^T$ . In fact, starting from the analysis formulation, we define  $\mathbf{Tz} = \mathbf{x}$  and thus we can replace the term  $\|\mathbf{Tz}\|_0$  by the simpler,  $\|\mathbf{x}\|_0$ . Multiplying this equation by  $\mathbf{T}^T$ , we obtain the relation  $\mathbf{T}^T \mathbf{Tz} = \mathbf{T}^T \mathbf{x}$ . Based on the full-rank assumption made above regarding  $\mathbf{A}$  (and thus  $\mathbf{T}$ ), this implies that  $\mathbf{z} = (\mathbf{T}^T \mathbf{T})^{-1} \mathbf{T}^T \mathbf{x} = \mathbf{Ax}$ , and therefore the  $\ell_2$ -term in the constraint can be replaced to from  $\|\mathbf{y} - \mathbf{z}\|_2$  to  $\|\mathbf{y} - \mathbf{Ax}\|_2$ .

While this seems like a perfect transfer from the analysis to the synthesis formulation, it is in fact missing a key element. From the two equations we got,  $\mathbf{Tz} = \mathbf{x}$  and  $\mathbf{z} = (\mathbf{T}^T \mathbf{T})^{-1} \mathbf{T}^T \mathbf{x}$ , immediately follows the requirement on  $\mathbf{x}$ ,  $\mathbf{T}(\mathbf{T}^T \mathbf{T})^{-1} \mathbf{T}^T \mathbf{x} = \mathbf{x}$ . This requirement simply states that  $\mathbf{x}$  must reside in the range of  $\mathbf{T}$ . Adding this as a constraint to the synthesis formulation, we get an exact equivalence, and otherwise, the synthesis gets a larger number of degrees of freedom, and thus its minimum is deeper (in fact, it is upper-bounded by the analysis formulation).



This way we see that the two models depart in general. The discussion throughout this chapter is steered towards the synthesis model, and this would be the focus of the signal processing modeling we adopt here. However, we stress that there is no clear answer to the question which of the two modeling approaches is better.

## 9.6 Summary

In this chapter we have proposed a generative model for describing signals. This model, denoted by  $\mathcal{M}(\mathbf{A}, k_0, \alpha, \epsilon)$  assumes that the signals of interest are created as random sparse combinations of atoms from a given dictionary, with a permitted small perturbation. We saw that this model is related to other ones that are extensively used in signal and image processing, such as MRF, Total-Variation, PCA and local-PCA, and more.

It must be clear that any mathematical model for signals, including the *Sparse-Land* model presented here, must do some injustice to the signals modelled, as it cannot describe their content exactly. Nevertheless, and despite this deviation from the true signal behavior, such models can be used successfully in various signal processing applications. The reason for this success could be explained in the following way. Consider a true signal source of dimension  $n$ , and further assume that the probability density function (prior) for this source is given as the function  $P(\mathbf{y})$ . When we propose a model we essentially propose an alternative PDF,  $\hat{P}(\mathbf{y})$ .

A perfect model would be one that provides a very close approximation to this PDF, implying for example that  $\|P(\mathbf{y}) - \hat{P}(\mathbf{y})\|$  is very small. Less favorable models, but ones that may still perform well, would be those that give a very small weighted error  $\|P(\mathbf{y})(P(\mathbf{y}) - \hat{P}(\mathbf{y}))\|$ . This means that the error is small in the active zone of the signal distribution, while getting a somewhat larger error in the inactive zone. This implies that such models are excellent in stating whether a signal is likely to occur, but are weaker in detecting whether a signal is not.

Put differently, assume that the portion covered in  $\mathbb{R}^n$  by true signals is  $\mu_{true} \ll 1$  (the zone where  $P(\mathbf{y})$  is relatively high). The less favorable models mentioned above may provide a coverage of  $\mu_{true} \ll \mu_{model} \ll 1$  portion in this space, such that it covers all/most-of the true zone and an additional (wrong!!) part. Synthesizing a signal from this model is very likely to result with a non-natural signal, since most of the area endorsed by this model is outside the true zone of likely signals. Thus, synthesis of signals from such a model leads to a failure. Nevertheless, cleaning an additive noise with such a model may work very well. This is because the additive noise puts us in a region where  $P(\mathbf{y})$  is very low, and most chances are that in this area  $\hat{P}(\mathbf{y})$  is also very low (because  $\mu_{model} \ll 1$ ). Denoising amounts to a search of  $\hat{\mathbf{y}}$  in the vicinity of  $\mathbf{y}$  such that it gives the highest value of  $\hat{P}(\mathbf{y})$ , and if this aligns with maximizing the true probability  $P(\mathbf{y})$ , we have a good denoising algorithm.

Further work is required to establish the above preliminary ideas and show that they hold true for popular families of signals of interest.

## Further Reading

1. J. Bobin, Y. Moudden, J.-L. Starck, and M. Elad, Morphological diversity and source separation, *IEEE Signal Processing Letters*, 13(7):409–412, July 2006.
2. J. Bobin, J.-L. Starck, M.J. Fadili, and Y. Moudden, SZ and CMB reconstruction using generalized morphological component analysis, Astronomical Data Analysis ADA'06, Marseille, France, September, 2006.
3. R.W. Buccigrossi and E.P. Simoncelli, Image compression via joint statistical characterization in the wavelet domain, *IEEE Trans. on Image Processing*, 8(12):1688–1701, 1999.
4. E.J. Candès and D.L. Donoho, Recovering edges in ill-posed inverse problems: optimality of curvelet frames, *Annals of Statistics*, 30(3):784–842, 2000.
5. E.J. Candès and D.L. Donoho, New tight frames of curvelets and optimal representations of objects with piecewise- $C^2$  singularities, *Comm. Pure Appl. Math.*, 57:219–266, 2002.
6. E.J. Candès and J. Romberg, Practical signal recovery from random projections, in *Wavelet XI, Proc. SPIE Conf.* 5914, 2005.
7. E.J. Candès, J. Romberg, and T. Tao, Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information, *IEEE Trans. on Information Theory*, 52(2):489–509, 2006.
8. E. Candès, J. Romberg, and T. Tao, Stable signal recovery from incomplete and inaccurate measurements, to appear in *Communications on Pure and Applied Mathematics*.
9. E.J. Candès and T. Tao, Decoding by linear programming, *IEEE Trans. on Information Theory*, 51(12):4203–4215, December 2005.
10. V. Chandrasekaran, M. Wakin, D. Baron, R. Baraniuk, Surflets: a sparse representation for multidimensional functions containing smooth discontinuities, IEEE Symposium on Information Theory, Chicago, IL, 2004.
11. S.S. Chen, D.L. Donoho, and M.A. Saunders, Atomic decomposition by basis pursuit, *SIAM Journal on Scientific Computing*, 20(1):33–61 (1998).
12. S.S. Chen, D.L. Donoho, and M.A. Saunders, Atomic decomposition by basis pursuit, *SIAM Review*, 43(1):129–159, 2001.
13. R. Coifman and D.L. Donoho, Translation-invariant denoising. *Wavelets and Statistics, Lecture Notes in Statistics*, 103:120–150, 1995.
14. R.R. Coifman, Y. Meyer, S. Quake, and M.V. Wickerhauser, Signal processing and compression with wavelet packets. In *Progress in wavelet analysis and applications* (Toulouse, 1992), pp. 77–93.
15. R.R. Coifman and M.V. Wickerhauser, Adapted waveform analysis as a tool for modeling, feature extraction, and denoising, *Optical Engineering*, 33(7):2170–2174, July 1994.
16. R.A. DeVore, B. Jawerth, and B.J. Lucier, Image compression through wavelet transform coding, *IEEE Trans. Information Theory*, 38(2):719–746, 1992.
17. M.N. Do and M. Vetterli, The finite ridgelet transform for image representation, *IEEE Trans. on Image Processing*, 12(1):16–28, 2003.

18. M.N. Do and M. Vetterli, Framing pyramids, *IEEE Trans. on Signal Processing*, 51(9):2329–2342, 2003.
19. M.N. Do and M. Vetterli, The contourlet transform: an efficient directional multiresolution image representation, *IEEE Trans. Image on Image Processing*, 14(12):2091–2106, 2005.
20. D.L. Donoho, Compressed sensing, *IEEE Trans. on Information Theory*, 52(4):1289–306, April 2006.
21. D.L. Donoho, De-noising by soft thresholding, *IEEE Trans. on Information Theory*, 41(3):613–627, 1995.
22. D.L. Donoho, For most large underdetermined systems of linear equations, the minimal  $\ell_1$ -norm solution is also the sparsest solution, *Communications on Pure and Applied Mathematics*, 59(6):797–829, June 2006.
23. D.L. Donoho, For most large underdetermined systems of linear equations, the minimal  $\ell_1$ -norm near-solution approximates the sparsest near-solution, *Communications on Pure and Applied Mathematics*, 59(7):907–934, July 2006.
24. D.L. Donoho and Y. Tsaig, Extensions of compressed sensing, *Signal Processing*, 86(3):549–571, March 2006.
25. M. Elad, B. Matalon, and M. Zibulevsky, Image denoising with shrinkage and redundant representations, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), NY, June 17–22, 2006.
26. M. Elad, J.-L. Starck, P. Querre, and D.L. Donoho, Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA), *Journal on Applied and Comp. Harmonic Analysis*, 19:340–358, 2005.
27. R. Eslami and H. Radha, The contourlet transform for image de-noising using cycle spinning, in Proceedings of Asilomar Conference on Signals, Systems, and Computers, pp. 1982–1986, November 2003.
28. R. Eslami and H. Radha, Translation-invariant contourlet transform and its application to image denoising, *IEEE Trans. on Image Processing*, 15(11):3362–3374, November 2006.
29. O.G. Guleryuz, Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising - Part I: Theory, *IEEE Trans. on Image Processing*, 15(3):539–554, 2006.
30. O.G. Guleryuz, Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising - Part II: Adaptive algorithms, *IEEE Trans. on Image Processing*, 15(3):555–571, 2006.
31. W. Hong, J. Wright, K. Huang, and Y. Ma, Multi-scale hybrid linear models for lossy image representation, *IEEE Transactions on Image Processing*, 15(12):3655–3671, December 2006.
32. A.K. Jain, *Fundamentals of Digital Image Processing*, Englewood Cliffs, NJ, Prentice-Hall, 1989.
33. M. Jansen, *Noise Reduction by Wavelet Thresholding*, Springer-Verlag, New York 2001.
34. E. Kidron, Y.Y. Schechner, and M. Elad, Cross-modality localization via sparsity, to appear in the *IEEE Trans. on Signal Processing*.

35. M. Lang, H. Guo, and J.E. Odegard, Noise reduction using undecimated discrete wavelet transform, *IEEE Signal Processing Letters*, 3(1):10–12, 1996.
36. M. Lustig, D.L. Donoho, and J.M. Pauly, Sparse MRI: The application of compressed sensing for rapid MR imaging, submitted to *Magnetic Resonance in Medicine*.
37. M. Lustig, J.M. Santos, D.L. Donoho, and J.M. Pauly, k-t SPARSE: High frame rate dynamic MRI exploiting spatio-temporal sparsity, Proceedings of the 13th Annual Meeting of ISMRM, Seattle, 2006.
38. Y. Ma, A. Yang, H. Derksen, and R. Fossum, Estimation of subspace arrangements with applications in modeling and segmenting mixed data, *SIAM Review*, 50(3):413–458, August 2008.
39. D.M. Malioutov, M. Cetin, and A.S. Willsky, Sparse signal reconstruction perspective for source localization with sensor arrays, *IEEE Trans. on Signal Processing*, 53(8):3010–3022, 2005.
40. F.G. Meyer, A. Averbuch, and J.O. Stromberg, Fast adaptive wavelet packet image compression, *IEEE Trans. on Image Processing*, 9(5):792–800, 2000.
41. F.G. Meyer, A.Z. Averbuch, R.R. Coifman, Multilayered image representation: Application to image compression, *IEEE Trans. on Image Processing*, 11(9):1072–1080, 2002.
42. F.G. Meyer and R.R. Coifman, Brushlets: a tool for directional image analysis and image compression, *Applied and Computational Harmonic Analysis*, 4:147–187, 1997.
43. P. Moulin and J. Liu, Analysis of multiresolution image denoising schemes using generalized Gaussian and complexity priors, *IEEE Trans. on Information Theory*, 45(3):909–919, 1999.
44. G. Peyré, Sparse modeling of textures, *Journal of Mathematical Imaging and Vision*, 34(1):17–31, May 2009.
45. G. Peyré, Manifold models for signals and images, *Computer Vision and Image Understanding*, 113(2):249–260, February 2009.
46. J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli, Image denoising using scale mixtures of gaussians in the wavelet domain, *IEEE Trans. on Image Processing*, 12(11):1338–1351, 2003.
47. L. Rudin, S. Osher, and E. Fatemi, Nonlinear total variation based noise removal algorithms, *Physica D*, 60:259–268, 1992.
48. E.P. Simoncelli and E.H. Adelson, Noise removal via Bayesian wavelet coring, Proceedings of the International Conference on Image Processing, Lausanne, Switzerland. September 1996.
49. E.P. Simoncelli, W.T. Freeman, E.H. Adelson, and D.J. Heeger, Shiftable multiscale transforms, *IEEE Trans. on Information Theory*, 38(2):587–607, 1992.
50. J.-L. Starck, E.J. Candès, and D.L. Donoho, The curvelet transform for image denoising, *IEEE Trans. on Image Processing*, 11:670–684, 2002.
51. J.-L. Starck, M. Elad, and D.L. Donoho, Redundant multiscale transforms and their application for morphological component separation, *Advances in Imaging and Electron Physics*, 132:287–348, 2004.

52. J.-L. Starck, M. Elad, and D.L. Donoho, Image decomposition via the combination of sparse representations and a variational approach, *IEEE Trans. on Image Processing*, 14(10):1570–1582, 2005.
53. J.-L. Starck, M.J. Fadili, and F. Murtagh, The undecimated wavelet decomposition and its reconstruction, *IEEE Transactions on Image Processing*, 16(2):297–309, 2007.
54. D.S. Taubman and M.W. Marcellin. *JPEG 2000: Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, Norwell, MA, USA, 2001.
55. J.A. Tropp and A.A. Gilbert, Signal recovery from random measurements via orthogonal matching pursuit, Submitted for publication, April 2005.
56. R. Vidal, Y. Ma, and S. Sastry, Generalized principal component analysis (GPCA), *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1945–1959, December 2005.
57. M. Zibulevsky and B.A. Pearlmutter, Blind source separation by sparse decomposition in a signal dictionary, *Neural Computation*, 13(4):863–882, 2001.

## Chapter 10

# Image Deblurring – A Case Study

In this chapter we present an application of the *Sparse-Land* model to image deblurring, in order to demonstrate the applicative side of the above-discussed model and algorithms. As we show next, this long-studied problem can be handled quite effectively using the fundamentals of the model with hardly any changes. The content of this chapter follows closely with the work by M.A.T. Figueiredo and R.D. Nowak that appeared in ICIP 2005, and a later paper by M. Elad, B. Matalon, and M. Zibulevsky (2007). While there exists a more recent work that leads to somewhat improved results, the appeal in this work is the relative simplicity with which near-state-of-the-art results are obtained.

### 10.1 Problem Formulation

We assume that an original image  $\mathbf{y}$  of size  $\sqrt{n} \times \sqrt{n}$  pixels (and in our case  $\sqrt{n} = 256$ ) goes through a *known* linear space-invariant blur operation  $\mathbf{H}$ , followed by an additive zero-mean white Gaussian noise with *known* variance  $\sigma^2$ . Thus, we measure a blurred and noisy image of the same size,  $\tilde{\mathbf{y}} = \mathbf{H}\mathbf{y} + \mathbf{v}$ . We aim to recover  $\mathbf{y}$ , with the assumption that  $\mathbf{y}$  could be described as  $\mathbf{A}\mathbf{x}$  with a *known* dictionary  $\mathbf{A}$  and a sparse vector  $\mathbf{x}$ . As we have seen in Chapters 6 and 9, this problem leads to the following minimization problem

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x}} \frac{1}{2} \|\tilde{\mathbf{y}} - \mathbf{H}\mathbf{A}\mathbf{x}\|_2^2 + \lambda \cdot \mathbf{1}^T \cdot \rho(\mathbf{x}), \quad (10.1)$$

where  $\rho(\mathbf{x})$  operates on every entry in  $\mathbf{x}$  separately. A commonly used choice of  $\rho(x)$  is  $\rho(x) = |x|^p$  with  $p \leq 1$ , which leads to the  $\ell_p$ -norm, but many other options exist. The restored image is given by  $\hat{\mathbf{y}} = \mathbf{A}\hat{\mathbf{x}}$ , and our goal is to get as close as possible to the original image, in terms of the mean-squared-error (MSE). As we will be employing synthetic experiments, in which we start from an original image, degrade it, and then restore  $\hat{\mathbf{y}}$ , this error can be evaluated.

There is a similarity between the restoration technique posed here and the Total-Variation (TV) image restoration, in which the penalty to be minimized is given by

$$\hat{\mathbf{y}} = \arg \min_{\mathbf{y}} \frac{1}{2} \|\tilde{\mathbf{y}} - \mathbf{H}\mathbf{y}\|_2^2 + \lambda \cdot \mathbf{1}^T \cdot \rho_{TV}(\mathbf{y}). \quad (10.2)$$

The term  $\rho_{TV}(\mathbf{y})$  computes the local (discrete) 2D-gradient length at each pixel,

$$\sqrt{(\partial \mathbf{y}(h, v)/\partial h)^2 + (\partial \mathbf{y}(h, v)/\partial v)^2},$$

( $h$  and  $v$  represent horizontal and vertical directions) and list those in one column vector of length  $n$ .

In terms of our definitions in Chapter 9, this is an analysis-based inverse problem, since the unknown is the image itself and not its representation. We have seen that in the 1D case, the first-derivative in the analysis-form corresponds to a Heaviside basis in the synthesis counterpart. As we are dealing here with a 2D image, such a relation no longer holds. Nevertheless, one could replace the TV penalty  $\mathbf{1}^T \cdot \rho_{TV}(\mathbf{y})$  with an approximated version of it that computes four first-derivatives in orientations  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$ , and  $135^\circ$ , and sums their absolute values with proper weights,

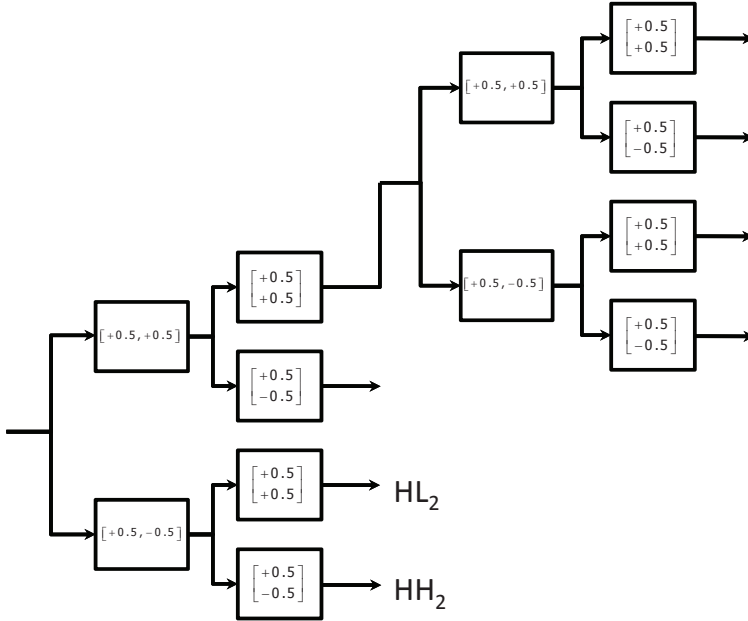
$$\|\partial \mathbf{y}/\partial d_0\|_1 + \frac{1}{\sqrt{2}} \|\partial \mathbf{y}/\partial d_{45}\|_1 + \|\partial \mathbf{y}/\partial d_{90}\|_1 + \frac{1}{\sqrt{2}} \|\partial \mathbf{y}/\partial d_{135}\|_1.$$

This penalty can be written as  $\|\mathbf{T}\mathbf{y}\|_1$ , where  $\mathbf{T}$  is of size  $4n \times n$ , containing the four derivative operations. The resulting scheme has an exact analysis structure, for which we may explore the corresponding synthesis one. We note that while the obtained operator  $\mathbf{T}$  is nearly rotation-invariant, it is not of a multi-scale nature, i.e., all the computed derivatives are of the same scale. This should be contrasted with the wavelet alternative which we introduce next.

## 10.2 The Dictionary

For the formulation in Equation 10.1 to lead to a practical image deblurring algorithm, we have to specify the dictionary  $\mathbf{A}$ , such that it could lead to sparse representation modeling of image content. Following the work by Figueiredo and others, we use the translation-invariant (undecimated) wavelet transform using Daubechies' 2-taps filter (Haar), with 2 layers of resolution, leading to a redundancy factor of 7 : 1.

The chosen Haar transform is obtained by filtering the image with the 1D kernels  $[+1, +1]/2$  and  $[+1, -1]/2$  horizontally, and applying the same filters vertically on the two outputs. This gives 4 output images of the same size as the input one. The process proceeds with the low-low pass image (the one that went through the kernel  $[+1, +1]/2$  in both directions), applying the same set of operations, thus getting an overall of 7 images of size  $\sqrt{n} \times \sqrt{n}$  each, at the output. This entire process describes



**Fig. 10.1** The forward filtering process that computes  $\mathbf{A}^T \mathbf{y}$ .

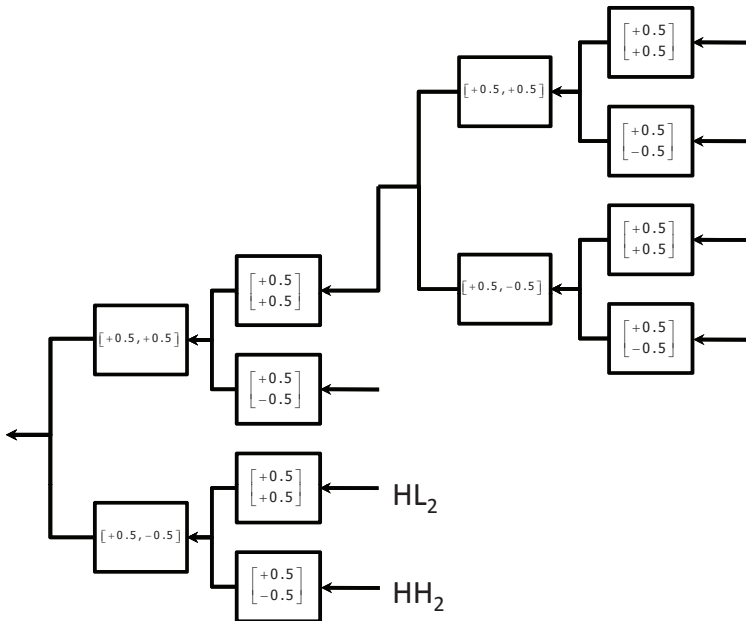
the multiplication  $\mathbf{A}^T \mathbf{y}$  in our terminology. Figure 10.1 shows this filtering process. Figure 10.2 shows the inverse operation that folds these filtered images back to the original image, thus computing  $\mathbf{A} \mathbf{x}$ . Figure 10.3 presents the 7 bands obtained using the traditional wavelet presentation, although, as opposed to the unitary Haar wavelet described here, each band gives an image of the same size.

We note that this choice of dictionary is far from being the optimal choice for sparsely representing images, and better results may be obtained if a more adequate dictionary is used. Such a transform is available within *Wavelab*, the Rice wavelet toolbox, and other public domain wavelet packages.

The operation  $\mathbf{A} \mathbf{x}$  is accomplished by the inverse Haar transform for this redundant case. Since this transform represents a tight Frame (i.e.,  $\mathbf{A} \mathbf{A}^T = \mathbf{I}$ ), the adjoint and the (pseudo) inverse are the same.<sup>1</sup> The adjoint operation can admit a filtering interpretation, much like the operation  $\mathbf{A}^T \mathbf{y}$ , but we shall not dwell on this here.

<sup>1</sup> Verifying that the inverse transform is also the adjoint (transpose) can be done by computing the inner-product between the forward transform of two random vectors,  $\mathbf{v}$  and  $\mathbf{u}$ ,  $\langle \mathbf{A}^T \mathbf{v}, \mathbf{A}^T \mathbf{u} \rangle$ , and verifying that it coincides with  $\mathbf{v}^T \mathbf{u}$ . Several such tests leading to exact equality are sufficient.





**Fig. 10.2** The inverse filtering process that computes  $\mathbf{Ax}$ .

Applying the inverse transform on a vector  $\mathbf{x} = \mathbf{e}_i$  (i.e., only the  $i$ -th entry is equal to 1 and all the rest are zeros) gives the  $i$ -th column in  $\mathbf{A}$ , and this can be used for visualizing the atoms in this dictionary. [Figure 10.4](#) presents 20 such atoms that were chosen at random, out of 2,800 atoms that serve for representing images of size  $20 \times 20$  pixels with the same redundancy factor.

### 10.3 Numerical Considerations

The numerical solvers used in our experiments for the deblurring task are the SSF and the PCD, as described in Chapter 6, using 100 iterations, with and without a line search or SESOP speedup. The parameter  $\lambda$  is set manually to lead to the best results. The method used in the original work by Figueiredo and Nowak is the plain SSF, given by

LL <sub>2</sub>	LH <sub>2</sub>	LH <sub>1</sub>
HL <sub>2</sub>	HH <sub>2</sub>	
HL <sub>1</sub>		HH <sub>1</sub>

**Fig. 10.3** The 7 bands of the 2 resolution layers Haar wavelet. This graphic description fits the unitary Haar wavelet, and in our implementation every band leads to an image of the same (original) size.

$$\mathbf{x}_{k+1} = \mathcal{S}_{\rho, \lambda/c} \left( \mathbf{x}_k + \frac{1}{c} \mathbf{A}^T \mathbf{H}^T (\tilde{\mathbf{y}} - \mathbf{H} \mathbf{A} \mathbf{x}_k) \right), \quad (10.3)$$

with  $c = 1$ , since  $\|\mathbf{H} \mathbf{A}\|_2^2 = 1$ .

In our tests we used  $\rho(x) = |x| - s \log(1 + |x|/s)$ , which leads to near- $\ell_1$ -norm for small values of  $s > 0$  ( $s = 0.01$  in our experiments). [Figure 10.5](#) presents the function  $\rho(x)$  for several choices of the parameter  $s$ . Despite its first appearance, this function is smooth everywhere and convex, as its derivatives manifest,

$$\frac{d\rho(x)}{dx} = \frac{|x|\text{sign}(x)}{s + |x|} \quad \text{and} \quad \frac{d^2\rho(x)}{dx^2} = \frac{s}{(s + |x|)^2}.$$

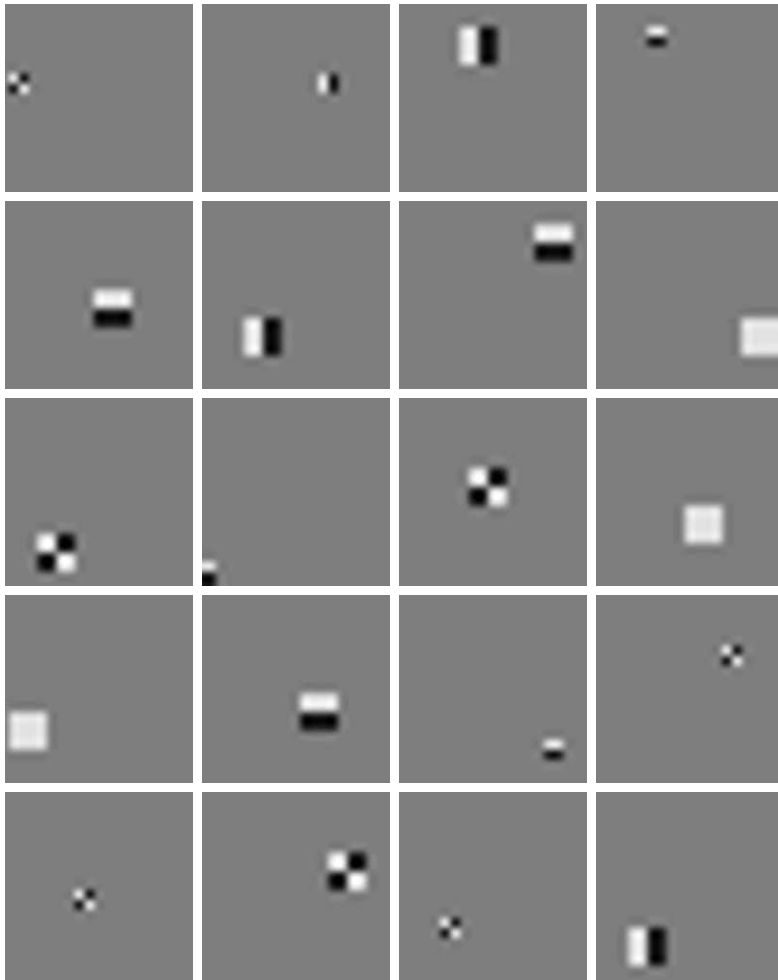
One important advantage of this choice of  $\rho$  is the fact that it leads to a closed-form analytical expression for the shrinkage step. Considering the function

$$f(x) = \frac{(x - x_0)^2}{2} + \lambda\rho(x), \quad (10.4)$$

the minimum is obtained for

$$0 = \frac{df(x)}{dx} = x - x_0 + \lambda \frac{|x|\text{sign}(x)}{s + |x|}. \quad (10.5)$$

Assuming that  $x_0 \geq 0$ , this necessarily implies that the solution to the above equation,  $x_{opt}$ , is also non-negative. This is because in the first term  $((x - x_0)^2/2)$  in Equation (10.4) would prefer a positive value over a negative one, and the second term ( $\lambda\rho(x)$ ) is indifferent to this sign. Therefore, the equation we face is



**Fig. 10.4** A set of 20 randomly chosen atoms from a dictionary of size  $400 \times 2800$ , used for representing images of size  $20 \times 20$  pixels. These atoms refer to the Haar translation-invariant transform with 2 resolution levels. We see in these atoms horizontal, vertical, or combined derivatives, we see two possible support sizes, and we also see atoms which are constant over their support, representing the low-low part of the transform.

$$0 = \frac{df(x)}{dx} = x - x_0 + \lambda \frac{x}{s+x} = \frac{(x+s)(x-x_0) + \lambda x}{s+x}. \quad (10.6)$$

The solution to this equation is given by (the negative option is discarded)

$$x_{opt} = \frac{(x_0 - s - \lambda) + \sqrt{(s + \lambda - x_0)^2 + 4sx_0}}{2}. \quad (10.7)$$

Naturally, for  $x_0 < 0$  the same formula applies with negating and mirroring this curve. This analytic shrinkage rule for  $\lambda = 1$  is presented in [Figure 10.6](#) for the same choices of the parameter  $s$  as in [Figure 10.5](#). As can be seen, as  $s$  gets smaller, we are approaching a soft-thresholding function.

Another benefit in the use of the above smoothed  $\ell_1$ -norm expression, is the ability to apply Newton optimization within the line-search (or SESOP optimization) that takes place within every iteration.

The work by Figueiredo and Nowak is different with respect to the choice of the function  $\rho$ , as they consider an  $\ell_p$ -norm with  $p = 0.7$ , or an alternative called Jeffrey's uninformative prior. Another difference between the two solutions refers to the initialization: Our simulations are initialized with  $\mathbf{x}_0 = \mathbf{0}$ , whereas the work by Figueiredo et al., proposes a potentially better initialization based on an adaptive Wiener restoration. For a convex penalty function as the one we use, this has no important consequences, but as we tend to non-convex choices of  $\rho(x)$  this difference may be critical.

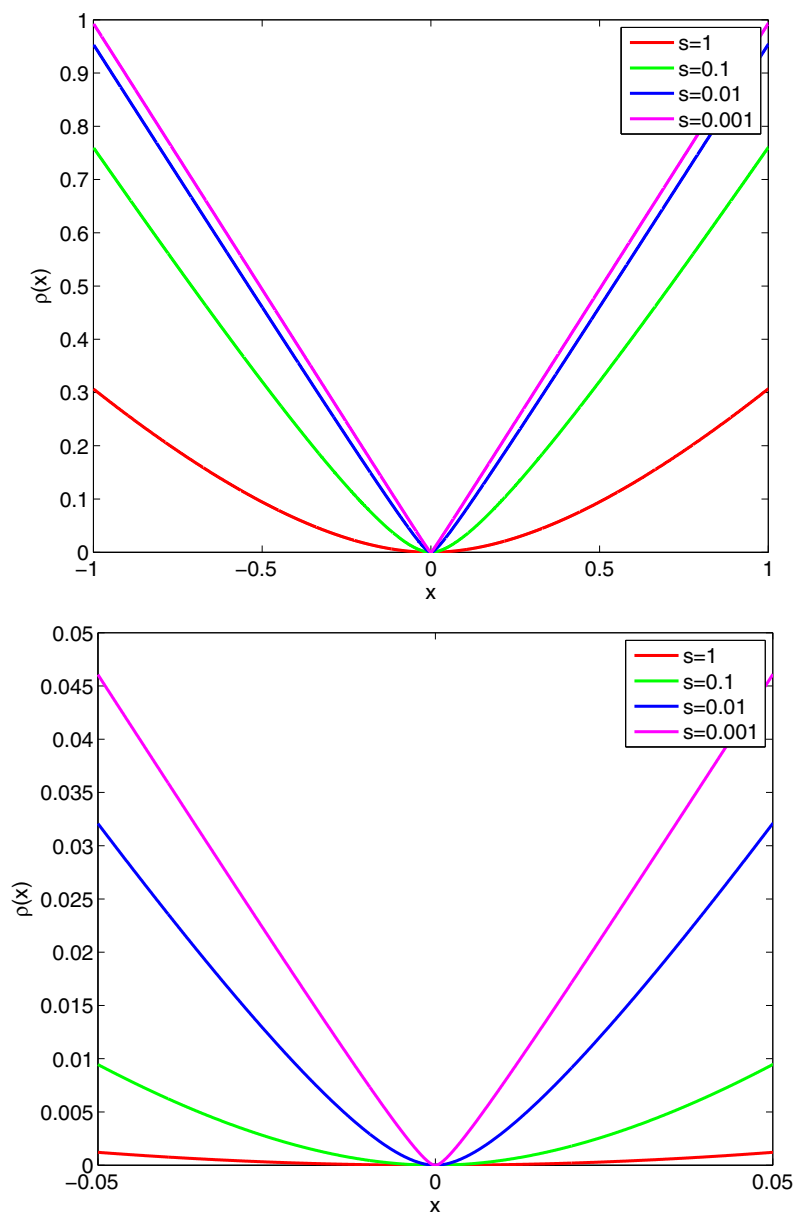
## 10.4 Experiment Details and Results

In the following experiments we apply the above algorithms for the restoration of the image shown in [Figure 10.7](#). We assume that the blur kernel is a  $15 \times 15$  filter with values  $1/(i^2 + j^2 + 1)$  for  $-7 \leq i, j \leq 7$ , normalized to have a unit sum. This blur kernel is shown in [Figure 10.8](#).

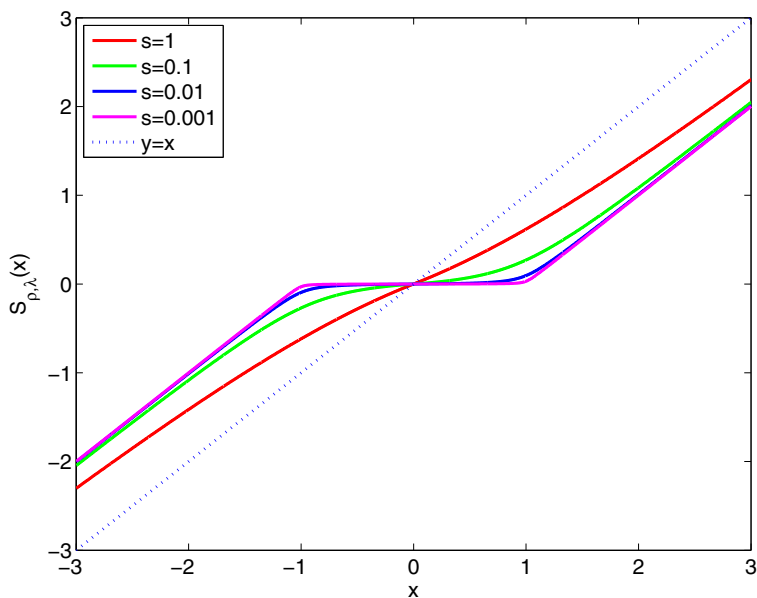
We start by presenting the deblurring performance for  $\sigma^2 = 2$  ( $\lambda = 0.075$ ). [Figure 10.9](#) presents the comparative performance of the SSF (plain, with line-search, and with SESOP-5 acceleration) and the PCD (regular and with SESOP-5 too). The graphs show the value of the objective as a function of the iteration. Similarly to the conclusions drawn in Chapter 6, it is clear that (i) line-search is helpful for SSF; (ii) SESOP acceleration helps further in accelerating SSF and PCD; and (iii) PCD-SESOP-5 is the fastest to converge.

For a quantitative evaluation of the results, we use the Improvement-Signal-to-Noise Ratio (ISNR) measure, defined as

$$ISNR = 10 \cdot \log \left\{ \frac{\|\mathbf{y} - \tilde{\mathbf{y}}\|_2^2}{\|\mathbf{A}\hat{\mathbf{x}} - \mathbf{y}\|_2^2} \right\} \text{ [dB]}. \quad (10.8)$$



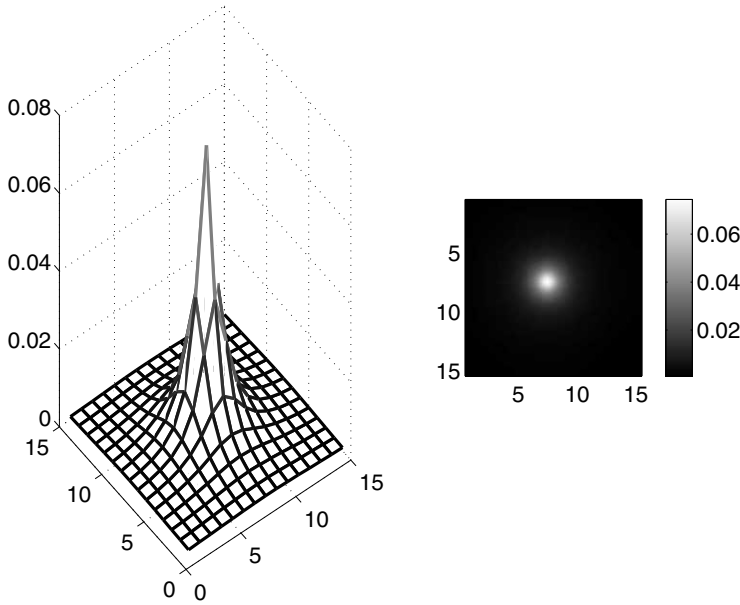
**Fig. 10.5** The function  $\rho(x)$  for several choices of the parameter  $s$ . The top graph shows the general structure, and the bottom one presents a zoomed version of this graph around the origin.



**Fig. 10.6** The shrinkage rule for the function  $\rho(x)$ , for several choices of the parameter  $s$  and assuming  $\lambda = 1$ .



**Fig. 10.7** The original image cameraman, on which the restoration experiments are performed.



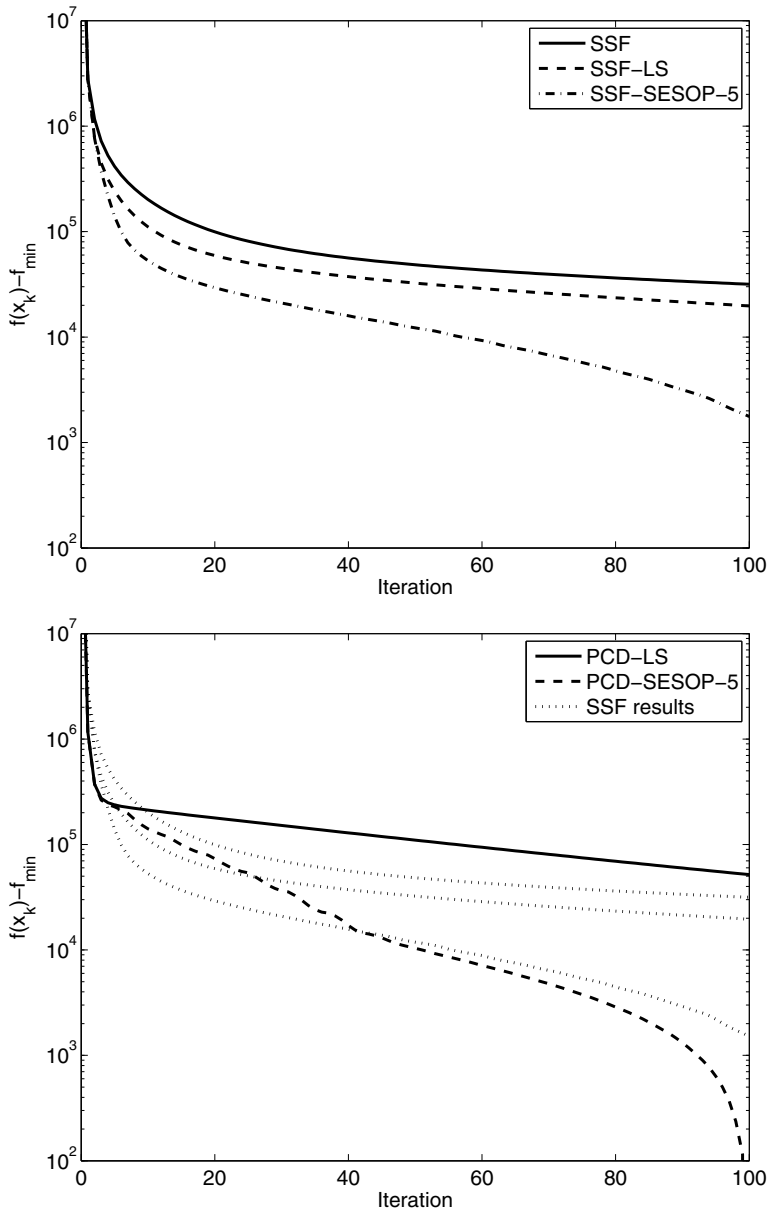
**Fig. 10.8** The blur kernel used, a  $15 \times 15$  filter with values  $1/(i^2 + j^2 + 1)$  for  $-7 \leq i, j \leq 7$ , normalized to a unit sum. The blur is presented both as a mesh-graph and as an image.

This way, if we propose  $\tilde{\mathbf{y}}$  as the solution,  $ISNR = 0\text{dB}$ . For a restored image of better quality, the ISNR is expected to be positive, and the higher it is, the better the image quality obtained. Figure 10.10 shows the ISNR results for the test with  $\sigma^2 = 2$ , as a function of the iteration for the various iterative-shrinkage algorithms tested. As can be seen, PCD (with or even without a SESOP speedup) is the fastest to obtain the maximal quality.<sup>2</sup>

Figure 10.11 shows the restored images obtained after 30 iterations of the PCD algorithm. Figure 10.12 similarly shows the result obtained for  $\sigma^2 = 8$  (for which we choose  $\lambda = 0.15$  for best ISNR). As can be seen, the resulting images are restored very well, even by this very simple and crude algorithm.

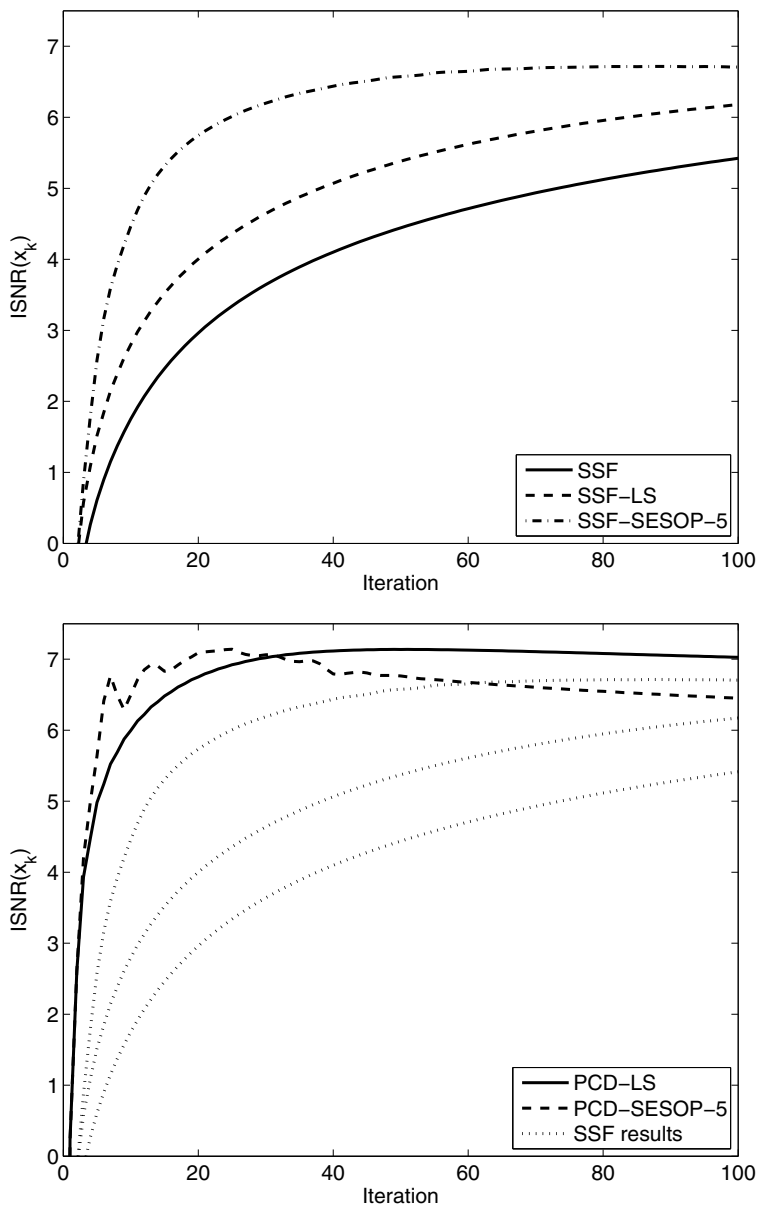
As a teaser, just before we conclude this chapter, we draw the reader's attention to the following strange behavior. As we minimize the penalty function discussed in this chapter,  $f(\mathbf{x}) = 0.5\|\mathbf{Ax} - \mathbf{y}\|_2^2 + \lambda \cdot \mathbf{1}^T \cdot \rho(\mathbf{x})$ , our goal is the recovery of the sparse representation  $\mathbf{x}$  of the image we are restoring. As such, we expect this vector to be sparse, so as to fulfil the *Sparse-Land* model assumptions. Thus, it is natural to ask: How sparse is this vector at the solution? Figure 10.13 shows a sorting of the absolute values of the entries in  $\mathbf{x}$ . As can be seen, the result is not sparse at all!

<sup>2</sup> There are ways to stop these algorithms when they get to the peak-ISNR value based on the SURE estimator.



**Fig. 10.9** The value of the penalty function in Equation 10.1 as a function of the iteration, shown for several iterative-shrinkage algorithms: (i) Plain SSF; (ii) SSF boosted by a line-search; (iii) PCD (with line-search); (iv) SSF with SESOP speedup; and (v) PCD with SESOP speedup.





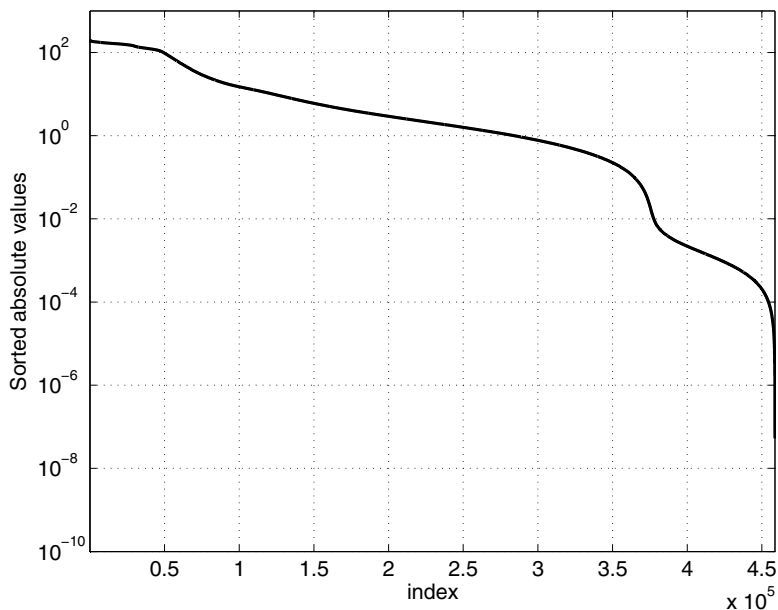
**Fig. 10.10** Deblurring results in terms of ISNR as a function of the iteration, shown for several iterative-shrinkage algorithms: (i) Plain SSF; (ii) SSF boosted by a line-search; (iii) PCD (with line-search); (iv) SSF with SESOP speedup; and (v) PCD with SESOP speedup.



**Fig. 10.11** Deblurring results with 30 iterations of the PCD algorithm: Measured blurry and noisy image (left), and restored (right). This result refers to the experiment with  $\sigma^2 = 2$  (resulting with  $ISNR = 7.05\text{dB}$ ).



**Fig. 10.12** Deblurring results with 30 iterations of the PCD algorithm: Measured blurry and noisy image (left), and restored (right). This result refers to the experiment with  $\sigma^2 = 8$  (resulting with  $ISNR = 5.17\text{dB}$ ).



**Fig. 10.13** The PCD solution after 30 iterations presented by sorting its absolute entries. As can be seen, the result is not sparse at all!

The reason for this result is clearly the choice of  $\lambda$ , which gives a too weak emphasis to the second term in the penalty, thus allowing a non-sparse solution. As we have said above, this choice of  $\lambda$  is deliberate, giving the best result in terms of ISNR (or the smallest Mean-Squared-Error – the two are coupled). Thus we should ask: Why the best performing  $\lambda$  leads to a dense result, and is this a contradiction to the *Sparse-Land* model we have described so enthusiastically in the previous chapter? An interesting answer to this question is provided in Chapter 11, where we consider explicit ways to compute the best (in terms of Mean-Squared-Error) estimate of  $\mathbf{x}$  for the *Sparse-Land* model.

## 10.5 Summary

It seems that a direct use of the *Sparse-Land* model from Chapter 9 to the image deblurring problem leads to a successful application. This stands as the first among many testimonials for the practicality of this model for image processing tasks. The reason we have chosen to begin our journey to applications with this problem in particular is the purity and immediacy with which we have migrated from the core ideas in Chapter 9 to an actual solution of this inverse problem.

Better image deblurring algorithms exist, and one such method is the BM3D-based method by Dabov et al. Interestingly, while this method differs quite substantially from the algorithm posed here, it is also based on the *Sparse-Land* model. This method is applied on image patches, with a collaborative treatment of the image content. We shall return to these themes and discuss patch-based sparsity methods and BM3D in particular, in later chapters.

This chapter also revisits the topic of iterative-shrinkage algorithms from Chapter 6. We see again the behavior of the SSF and the PCD and their accelerated versions. The bottom line we have reached here suggests that in a matter of 10–20 iterations, each applying one multiplication by  $\mathbf{A}$  and its adjoint, one can get to the desired result. There is one fact shadowing this conclusion – we observed that the outcome does not show any tendency to be sparse. Does this stand in contradiction to the *Sparse-Land* model? We now turn to Chapter 11, where this question is given an interesting and surprising answer.

## Further Reading

1. J. Bioucas-Dias, Bayesian wavelet-based image deconvolution: a GEM algorithm exploiting a class of heavy-tailed priors, *IEEE Trans. on Image processing*, 15(4):937–951, April 2006.
2. K. Dabov, A. Foi, and K. Egiazarian, Image restoration by sparse 3D transform-domain collaborative filtering, Proc. SPIE Electronic Imaging '08, no. 6812-07, San Jose, California, USA, January 2008.
3. I. Daubechies, M. Defrise, and C. De-Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, *Communications on Pure and Applied Mathematics*, LVII:1413–1457, 2004.
4. M. Elad, B. Matalon, and M. Zibulevsky, On a Class of Optimization methods for Linear Least Squares with Non-Quadratic Regularization, *Applied and Computational Harmonic Analysis*, 23:346–367, November 2007.
5. M. Elad, B. Matalon, J. Shtok, and M. Zibulevsky, A wide-angle view at iterated shrinkage algorithms, SPIE (Wavelet XII) 2007, San-Diego CA, August 26-29, 2007.
6. M.J. Fadili, J.-L. Starck, Sparse representation-based image deconvolution by iterative thresholding, Astronomical Data Analysis ADA'06, Marseille, France, September, 2006.
7. M.A. Figueiredo and R.D. Nowak, An EM algorithm for wavelet-based image restoration, *IEEE Trans. Image Processing*, 12(8):906–916, 2003.
8. M.A. Figueiredo, and R.D. Nowak, A bound optimization approach to wavelet-based image deconvolution, IEEE International Conference on Image Processing - ICIP 2005, Genoa, Italy, 2:782–785, September 2005.
9. M.A. Figueiredo, J.M. Bioucas-Dias, and R.D. Nowak, Majorization-minimization algorithms for wavelet-based image restoration, *IEEE Trans. on Image Processing*, 16(12):2980–2991, December 2007.

10. L. Rudin, S. Osher, and E. Fatemi, Nonlinear total variation based noise removal algorithms, *Physica D*, 60:259–268, 1992.
11. H. Takeda, S. Farsiu, and P. Milanfar, Deblurring using regularized locally-adaptive kernel regression, *IEEE Trans. on Image Processing*, 17(4):550–563, April 2008

# Chapter 11

## MAP versus MMSE Estimation

So far we kept the description of the pursuit algorithms on a deterministic level, as an intuitive optimization procedure. We mentioned in Chapter 9 that these algorithms correspond to an approximation of the Maximum-A-posteriori-Probability (MAP) estimator, but this connection was not explicitly derived. In this chapter we make this claim exact by defining the quest for sparse representations as an estimation task. As we shall see, this calls for a clear and formal definition of the stochastic model assumed to generate the sparse representation vector. A benefit of such treatment is an ability to derive the Minimum-Mean-Squared-Error (MMSE) estimator as well, and this in turn leads to the need to approximate it. These and more are the topics we cover in this chapter.

### 11.1 A Stochastic Model and Estimation Goals

We start by defining a random generator of sparse vectors. While this definition is arbitrary and can be modified in various ways, it grasps the essence of sparse representations and their general behavior.

The generator we define produces a sparse vector  $\mathbf{x} \in \mathbb{R}^m$  by first drawing at random an integer scalar  $|s|$  from the Probability-Density-Function (PDF)  $P_s(|s|)$ , which represents the cardinality,  $\|\mathbf{x}\|_0 = |s|$ . A typical behavior of  $P_s(|s|)$  would be one that promotes sparser vectors, such as  $P_s(|s|) \sim 1/|s|$ ,  $P_s(|s|) \sim \exp(-\alpha|s|)$ , or even  $P_s(|s|) \sim \delta(k - |s|)$  that imposes a fixed  $k$  non-zeros. For simplicity, we shall proceed with this limited case where  $|s| = k$ .

Once  $k$  is chosen (or set, as mentioned above), we consider all  $\binom{m}{k}$  possible supports as equi-probable, and we draw one support  $s$  at random, such that  $|s| = k$ . Finally, we assume that the non-zero entries in  $s$  are drawn from  $\mathcal{N}(0, \sigma_x^2)$  as iid entries. This completes the description of how  $\mathbf{x}$  is generated, and this description induces a PDF for this source, denoted by  $P_X(\mathbf{x})$ . We shall refer to this random generator of vectors as  $\mathcal{G}_1$ .

As stated above, this model can be modified in various ways. Examples for generalizing this model include the possibility to assign different variances to the chosen non-zero coefficients, an ability to assign different probabilities for each entry to be chosen, instead of spreading this uniformly, replacing the Gaussianity of the entries with a heavy tailed distribution, and more.

Another way to pose a prior on  $\mathbf{x}$  is to assign to each atom a probability  $P_i$  to participate in the active support. Once chosen, its entry value follows the same iid Gaussian rule as above. Such a model may result with various supports, from the very empty to the very full. Assuming that all atoms share the same probability and the same variance  $\sigma_x^2$ , this implies that 2 parameters govern the complete behavior of this model. The great appeal in this model is the total independence between the  $m$  elements of  $\mathbf{x}$ , as opposed to the (somewhat hidden) dependency that exists in  $\mathcal{G}_1$ . We shall refer to this model as  $\mathcal{G}_2$ . Much of the analysis presented in this chapter refers to both models,  $\mathcal{G}_1$  and  $\mathcal{G}_2$ , equally well.

Returning to our random generator  $\mathcal{G}_1$  described above, the random vector  $\mathbf{x}$  is assumed to be multiplied by the dictionary  $\mathbf{A}$  of size  $n \times m$ , giving a signal  $\mathbf{z}$  that is known to have a sparse representation. This signal is multiplied by a degradation matrix  $\mathbf{H}$  of size  $q \times n$ . The measurement is the vector  $\mathbf{y} = \mathbf{H}\mathbf{A}\mathbf{x} + \mathbf{e}$ , where  $\mathbf{e}$  is a zero-mean white Gaussian noise vector with iid entries, drawn from  $\mathcal{N}(0, \sigma_e^2)$ . The estimation goal is the recovery of  $\mathbf{x}$  (or  $\mathbf{z}$ ) based on  $\mathbf{y}$  and the statistical information available to us on  $\mathbf{x}$  and  $\mathbf{e}$ .

The estimation we shall construct depends on various considerations. First, we should define the goal for the estimator. The MAP estimator aims to find  $\hat{\mathbf{x}}$  that maximizes the posterior probability  $P(\mathbf{x}|\mathbf{y})$ . The MMSE aims to find the solution  $\hat{\mathbf{x}}$  that minimizes the expected  $\ell_2$ -error of the estimation. Other goals and estimators can be proposed, but we shall mostly restrict the discussion in this chapter to these two. Beyond the objective the estimator targets, we shall see that practical numerical considerations are also of interest, and especially so for cases where the exact estimation cannot be computed or derived. As we shall see next, all the practical estimators formed for addressing the above defined problem are in general only approximations.

## 11.2 Background on MAP and MMSE

Since we use the MAP and the MMSE estimators quite extensively in this chapter, we provide here a general definition of these two Bayesian tools. Given a measurement vector  $\mathbf{y}$  and an unknown vector to be estimated,  $\mathbf{x}$ , the MAP estimator proposes the following estimate:

$$\hat{\mathbf{x}}^{MAP} = \max_{\mathbf{x}} P(\mathbf{x}|\mathbf{y}), \quad (11.1)$$

where  $P(\mathbf{x}|\mathbf{y})$  is the posterior probability of  $\mathbf{x}$  given the measurement  $\mathbf{y}$ . Clearly, the complexity of constructing the MAP estimate is mainly manifested in formulating

well this conditional probability. Once this is done, the estimation problem becomes an optimization task. In cases where this optimization is non-convex with many possible local minimum solutions, the estimation becomes problematic and may divert from the true objective posed.

Using Bayes' rule, the posterior probability of  $\mathbf{x}$  given the measurements is given by

$$P(\mathbf{x}|\mathbf{y}) = \frac{P(\mathbf{y}|\mathbf{x})P(\mathbf{x})}{P(\mathbf{y})}. \quad (11.2)$$

Considering the fact that the denominator  $P(\mathbf{y})$  is not a function of the unknown  $\mathbf{x}$ , and as such it can be disregarded, the MAP estimation amounts to

$$\hat{\mathbf{x}}^{MAP} = \max_{\mathbf{x}} P(\mathbf{x}|\mathbf{y}) = \max_{\mathbf{x}} P(\mathbf{y}|\mathbf{x})P(\mathbf{x}). \quad (11.3)$$

The probability  $P(\mathbf{y}|\mathbf{x})$  is known as the likelihood function, and the probability  $P(\mathbf{x})$  is the unknown's prior. If this prior probability is flat and uninformative, the above process effectively maximizes the likelihood function, leading to the well-known Maximum-Likelihood estimator.

Turning to the MMSE, a well-known and classic result states that this estimator is obtained as the conditional mean of the unknown. This is easily seen by writing the mean-squared error as

$$\text{MSE}_x = E(\|\hat{\mathbf{x}} - \mathbf{x}\|_2^2 | \mathbf{y}) = \int_{\mathbf{x}} \|\hat{\mathbf{x}} - \mathbf{x}\|_2^2 P(\mathbf{x}|\mathbf{y}) d\mathbf{x}, \quad (11.4)$$

and taking a derivative with respect to the estimate  $\hat{\mathbf{x}}$  in order to minimize the error. This leads to

$$\frac{\partial \text{MSE}_x}{\partial \hat{\mathbf{x}}} = 2 \int_{\mathbf{x}} (\hat{\mathbf{x}} - \mathbf{x}) P(\mathbf{x}|\mathbf{y}) d\mathbf{x} = \mathbf{0}. \quad (11.5)$$

Thus, the optimal (MMSE) estimator is given as

$$\hat{\mathbf{x}}^{MMSE} = \frac{\int_{\mathbf{x}} \mathbf{x} P(\mathbf{x}|\mathbf{y}) d\mathbf{x}}{\int_{\mathbf{x}} P(\mathbf{x}|\mathbf{y}) d\mathbf{x}} = \int_{\mathbf{x}} \mathbf{x} P(\mathbf{x}|\mathbf{y}) d\mathbf{x} = E(\mathbf{x}|\mathbf{y}). \quad (11.6)$$

The denominator in the above expression equals 1 since it is a complete integration over the probability rule  $P(\mathbf{x}|\mathbf{y})$ .

Interestingly, if we define our goal as minimizing a weighted error with a matrix  $\mathbf{W}$ , the error considered is

$$\begin{aligned} \text{WMSE}_x &= E(\|\mathbf{W}(\hat{\mathbf{x}} - \mathbf{x})\|_2^2 | \mathbf{y}) \\ &= \int_{\mathbf{x}} \|\mathbf{W}(\hat{\mathbf{x}} - \mathbf{x})\|_2^2 P(\mathbf{x}|\mathbf{y}) d\mathbf{x}. \end{aligned} \quad (11.7)$$

The WMMSE estimator is obtained by requiring



$$\frac{\partial \text{WMSE}_x}{\partial \hat{\mathbf{x}}} = 2\mathbf{W}^T \mathbf{W} \int_{\mathbf{x}} (\hat{\mathbf{x}} - \mathbf{x}) P(\mathbf{x}|\mathbf{y}) d\mathbf{x} = \mathbf{0}, \quad (11.8)$$

If  $\mathbf{W}^T \mathbf{W}$  is invertible, the above expression leads to the same estimator as before, since the multiplication by  $\mathbf{W}^T \mathbf{W}$  can be omitted. If this matrix is not invertible, this implies that our estimate is not unique, as it can remain optimal by adding any portion that is found in the null-space of  $\mathbf{W}^T \mathbf{W}$ .

Considering our estimation task,  $\mathbf{y} = \mathbf{H}\mathbf{A}\mathbf{x} + \mathbf{e}$ , we could consider the minimum MSE in the representation domain,  $E(\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2|\mathbf{y})$ , or focus on the error in the signal domain,  $E(\|\mathbf{A}\mathbf{x} - \hat{\mathbf{z}}\|_2^2|\mathbf{y})$ . It is quite easy to see that the two estimation tasks align, both requiring the minimizer of  $E(\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2|\mathbf{y})$ , and if we desire the signal, we further multiply by  $\mathbf{A}$ . This is because

$$\begin{aligned} \frac{\partial E(\|\mathbf{A}\mathbf{x} - \hat{\mathbf{z}}\|_2^2|\mathbf{y})}{\partial \hat{\mathbf{z}}} &= \frac{\partial}{\partial \hat{\mathbf{z}}} \int_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \hat{\mathbf{z}}\|_2^2 P(\mathbf{x}|\mathbf{y}) d\mathbf{x} \\ &= 2 \int_{\mathbf{x}} (\hat{\mathbf{z}} - \mathbf{A}\mathbf{x}) P(\mathbf{x}|\mathbf{y}) d\mathbf{x} = \mathbf{0}, \end{aligned} \quad (11.9)$$

which leads to

$$\hat{\mathbf{z}} = \mathbf{A} \int_{\mathbf{x}} \mathbf{x} P(\mathbf{x}|\mathbf{y}) d\mathbf{x} = \mathbf{A} E(\mathbf{x}|\mathbf{y}). \quad (11.10)$$

## 11.3 The Oracle Estimation

### 11.3.1 Developing the Oracle Estimator

We have discussed the oracle in the context of the Dantzig-Selector, and we return to it now, developing it for the estimation problem at hand. The oracle is an infeasible estimator, as it assumes the knowledge of the chosen support for  $\mathbf{x}$ , information that is unknown in the actual problem. Nevertheless, there are several good reasons to consider its construction:

- The oracle is the core ingredient in the practical estimators we derive next;
- The oracle gives a reference performance quality to compare against;
- The oracle provides simple and convenient closed-form expressions for the estimation and the induced MSE; and
- The oracle is relatively easy to derive, and its development reveals some interesting insight into the more complicated estimators that follow.

Since the support  $s$  is known, all that remains to estimate are the values of the non-zero entries in  $\mathbf{x}$ . We denote by  $\mathbf{x}_s$  a vector of length  $s$  containing those unknowns. Similarly,  $\mathbf{A}_s$  is a sub-matrix of size  $n \times k$  that corresponds to the support. Clearly,  $\mathbf{y} = \mathbf{H}\mathbf{A}_s \mathbf{x}_s + \mathbf{e}$ , where  $\mathbf{x}_s \sim \mathcal{N}(\mathbf{0}, \sigma_x^2 \mathbf{I})$ . Using Bayes' rule, the posterior probability of  $\mathbf{x}_s$  given the measurements is given by

$$P(\mathbf{x}_s|\mathbf{y}) = \frac{P(\mathbf{y}|\mathbf{x}_s)P(\mathbf{x}_s)}{P(\mathbf{y})}. \quad (11.11)$$

The denominator  $P(\mathbf{y})$  can be disregarded as it is a constant factor normalizing this probability. The probabilities  $P(\mathbf{y}|\mathbf{x}_s)$  and  $P(\mathbf{x}_s)$  are easily obtained as simple multi-variate Gaussians,

$$P(\mathbf{x}_s) = \frac{1}{(2\pi)^{s/2}\sigma_x^s} \exp\left\{-\frac{\mathbf{x}_s^T \mathbf{x}_s}{2\sigma_x^2}\right\}, \quad (11.12)$$

and

$$P(\mathbf{y}|\mathbf{x}_s) = \frac{1}{(2\pi)^{q/2}\sigma_e^q} \exp\left\{-\frac{\|\mathbf{H}\mathbf{A}_s \mathbf{x}_s - \mathbf{y}\|_2^2}{2\sigma_e^2}\right\}. \quad (11.13)$$

Thus, the posterior for the oracle is given, up to a normalization factor, as

$$P(\mathbf{x}_s|\mathbf{y}) \propto \exp\left\{-\frac{\mathbf{x}_s^T \mathbf{x}_s}{2\sigma_x^2} - \frac{\|\mathbf{H}\mathbf{A}_s \mathbf{x}_s - \mathbf{y}\|_2^2}{2\sigma_e^2}\right\}. \quad (11.14)$$

The MAP-oracle estimation would be the vector  $\mathbf{x}_s$  that maximizes the above probability, and this is given by

$$\begin{aligned} \hat{\mathbf{x}}_s^{MAP-oracle} &= \max_{\mathbf{x}_s} P(\mathbf{x}_s|\mathbf{y}) \\ &= \max_{\mathbf{x}_s} \exp\left\{-\frac{\mathbf{x}_s^T \mathbf{x}_s}{2\sigma_x^2} - \frac{\|\mathbf{H}\mathbf{A}_s \mathbf{x}_s - \mathbf{y}\|_2^2}{2\sigma_e^2}\right\} \\ &= \left(\frac{1}{\sigma_e^2} \mathbf{A}_s^T \mathbf{H}^T \mathbf{H} \mathbf{A}_s + \frac{1}{\sigma_x^2} \mathbf{I}\right)^{-1} \frac{1}{\sigma_e^2} \mathbf{A}_s^T \mathbf{H}^T \mathbf{y}. \end{aligned} \quad (11.15)$$

Notice a (linear) shrinkage effect in the above expression, as the term  $\frac{1}{\sigma_x^2} \mathbf{I}$  causes the result to be somewhat smaller, compared to a direct pseudo-inverse of  $\mathbf{H}\mathbf{A}_s$ .

Rather than developing the MAP, the oracle can target instead a minimization of the MSE as its design goal. As we have seen above, the MMSE-oracle can be obtained as the mean of the posterior probability  $P(\mathbf{x}_s|\mathbf{y})$ . Since this probability is Gaussian, its peak is also its mean, which implies that the MAP-oracle is also the MMSE one. This outcome is classic and well-known, in the context of the Wiener filter.

If we desire to minimize the oracle's error in the signal domain, the error to minimize is  $E(\|\mathbf{A}_s(\hat{\mathbf{x}} - \mathbf{x})\|_2^2|\mathbf{y})$ . Assuming that  $\mathbf{A}_s^T \mathbf{A}_s$  is invertible, we have seen that the same estimator  $\hat{\mathbf{x}}$  remains optimal for this goal as well. This implies that we assume  $n \geq k$ , which is readily satisfied for the sparse vectors  $\mathbf{x}$  we are interested in. Similarly, if we target the error in the measurements domain, non-singularity of  $\mathbf{A}_s^T \mathbf{H}^T \mathbf{H} \mathbf{A}_s$  implies the same estimator, and this requirement also suggests that  $q \geq k$ .

### 11.3.2 The Oracle Error

We now turn to discuss the MSE induced by the oracle estimator. Since we have seen that the expression obtained fits the two discussed estimators (MMSE and MAP), we shall refer to it hereafter as  $\hat{\mathbf{x}}^{oracle}$ . The MSE is given by

$$\begin{aligned} E\left(\left\|\hat{\mathbf{x}}_s^{oracle} - \mathbf{x}_s\right\|_2^2\right) &= E\left(\left\|\left(\frac{1}{\sigma_e^2}\mathbf{A}_s^T\mathbf{H}^T\mathbf{H}\mathbf{A}_s + \frac{1}{\sigma_x^2}\mathbf{I}\right)^{-1} \frac{1}{\sigma_e^2}\mathbf{A}_s^T\mathbf{H}^T\mathbf{y} - \mathbf{x}_s\right\|_2^2\right) \\ &= E\left(\left\|\left(\frac{1}{\sigma_e^2}\mathbf{A}_s^T\mathbf{H}^T\mathbf{H}\mathbf{A}_s + \frac{1}{\sigma_x^2}\mathbf{I}\right)^{-1} \frac{1}{\sigma_e^2}\mathbf{A}_s^T\mathbf{H}^T(\mathbf{H}\mathbf{A}_s\mathbf{x}_s + \mathbf{e}) - \mathbf{x}_s\right\|_2^2\right), \end{aligned}$$

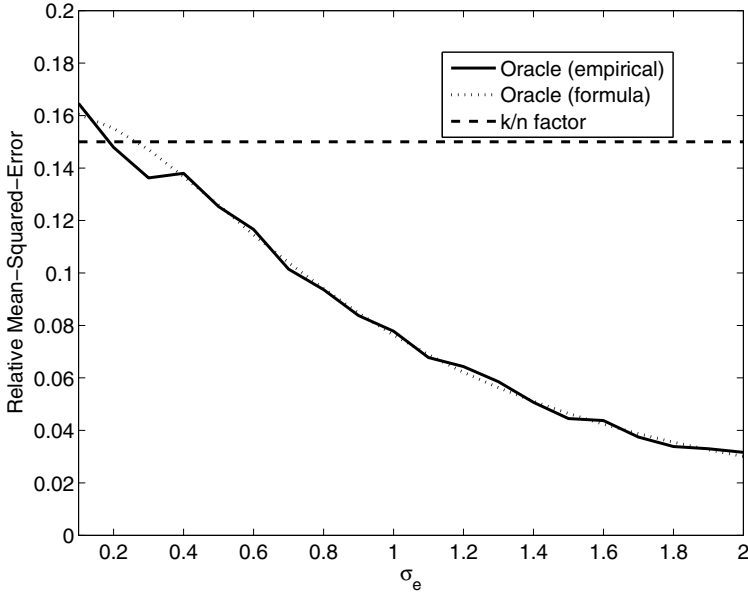
where we have used the fact that  $\mathbf{y} = \mathbf{H}\mathbf{A}_s\mathbf{x}_s + \mathbf{e}$ . Rearranging terms and defining  $\mathbf{Q}_s = \frac{1}{\sigma_e^2}\mathbf{A}_s^T\mathbf{H}^T\mathbf{H}\mathbf{A}_s + \frac{1}{\sigma_x^2}\mathbf{I}$ , we obtain

$$\begin{aligned} E\left(\left\|\hat{\mathbf{x}}_s^{oracle} - \mathbf{x}_s\right\|_2^2\right) &= E\left(\left\|\mathbf{Q}_s^{-1}\left(\frac{1}{\sigma_e^2}\mathbf{A}_s^T\mathbf{H}^T\mathbf{H}\mathbf{A}_s - \frac{1}{\sigma_e^2}\mathbf{A}_s^T\mathbf{H}^T\mathbf{H}\mathbf{A}_s - \frac{1}{\sigma_x^2}\mathbf{I}\right)\mathbf{x}_s\right.\right. \\ &\quad \left.\left.+ \mathbf{Q}_s^{-1}\frac{1}{\sigma_e^2}\mathbf{A}_s^T\mathbf{H}^T\mathbf{e}\right\|_2^2\right) \\ &= E\left(\left\|-\mathbf{Q}_s^{-1}\frac{1}{\sigma_x^2}\mathbf{x}_s + \mathbf{Q}_s^{-1}\frac{1}{\sigma_e^2}\mathbf{A}_s^T\mathbf{H}^T\mathbf{e}\right\|_2^2\right) \\ &= \text{trace}\left(\frac{1}{\sigma_x^4}\mathbf{Q}_s^{-2}E(\mathbf{x}_s\mathbf{x}_s^T) - \frac{2}{\sigma_e^2\sigma_x^2}\mathbf{Q}_s^{-2}\mathbf{A}_s^T\mathbf{H}^TE(\mathbf{e}\mathbf{x}_s^T)\right. \\ &\quad \left.+ \frac{1}{\sigma_e^4}\mathbf{Q}_s^{-2}\mathbf{A}_s^T\mathbf{H}^TE(\mathbf{e}\mathbf{e}^T)\mathbf{H}\mathbf{A}_s\right) \\ &= \text{trace}\left(\mathbf{Q}_s^{-2}\left[\frac{1}{\sigma_x^2}\mathbf{I} + \frac{1}{\sigma_e^2}\mathbf{A}_s^T\mathbf{H}^T\mathbf{H}\mathbf{A}_s\right]\right) = \text{trace}\left(\mathbf{Q}_s^{-1}\right). \end{aligned}$$

Here we have used the fact that  $E(\mathbf{x}_s\mathbf{x}_s^T) = E(\mathbf{e})E(\mathbf{x}_s^T) = 0$ , because the two are independent. Thus, the oracle gives an MSE error

$$E\left(\left\|\hat{\mathbf{x}}_s^{oracle} - \mathbf{x}_s\right\|_2^2\right) = \text{trace}\left(\left[\frac{1}{\sigma_e^2}\mathbf{A}_s^T\mathbf{H}^T\mathbf{H}\mathbf{A}_s + \frac{1}{\sigma_x^2}\mathbf{I}\right]^{-1}\right). \quad (11.16)$$

As an example, if the columns of  $\mathbf{H}\mathbf{A}_s$  are orthogonal and  $\mathbf{H}$  is square  $n \times n$ , this error becomes  $\sigma_x^2\sigma_e^2 \cdot k/(\sigma_x^2 + \sigma_e^2)$ . Whereas the additive noise power is  $n\sigma_e^2$ , the oracle error is less than  $k/n$  times this energy (it is in fact slightly smaller because of



**Fig. 11.1** The oracle performance as a function of the input noise. The performance is shown both by an empirical calculation of the oracle estimate and a comparison to the ground-truth, and by using the exact formula for the expected error. Results are shown relative to the input noise energy  $n\sigma_e^2$ .

the additional factor  $\sigma_x^2/(\sigma_x^2 + \sigma_e^2)$ ). For non-orthogonal columns, this noise decay is smaller.

Figure 11.1 presents the oracle results for the following experiment: A random dictionary  $\mathbf{A}$  of size  $20 \times 30$  is generated with iid Gaussian entries, and with normalized columns. We assume hereafter that  $\mathbf{H} = \mathbf{I}$ . Sparse random vectors  $\mathbf{x}$  are generated, using the  $\mathcal{G}_1$  model, with a fixed cardinality  $k = 3$ , and  $\sigma_x = 1$ . A Gaussian iid noise vector is added to  $\mathbf{Ax}$ , with standard-deviation in the range  $[0.1, 2]$ . For each such parameter setting we perform 1,000 experiments, and average the outcomes.

The results in Figure 11.1 are presented as the ratio between the oracle's error and the input noise  $E(\|\mathbf{e}\|^2) = n\sigma_e^2$ , such that a ratio smaller than 1 implies an effective noise removal. One graph presents the empirical error obtained by computing  $\hat{\mathbf{x}}^{oracle}$  using Equation (11.15) and evaluating the error directly by  $\|\mathbf{x} - \hat{\mathbf{x}}^{oracle}\|_2^2$ . The second graph shows the error as predicted by the formula in Equation (11.16). This figure also shows the constant factor  $k/n = 0.15$ , which is the expected noise decay-factor for the unitary case. Since this experiment uses a general matrix  $\mathbf{A}$ , the actual ratio obtained by the oracle error is not necessarily smaller than  $k/n$ , but as we can see, it tends to meet this ratio and beyond.

## 11.4 The MAP Estimation

### 11.4.1 Developing the MAP Estimator

We return to the estimation problem defined at the beginning of this chapter, and assume no knowledge about the support. In order to propose a MAP estimator, we should formulate the posterior  $P(\mathbf{x}|\mathbf{y})$ . Marginalizing over all the possible supports  $s \in \Omega$ , we have

$$P(\mathbf{x}|\mathbf{y}) = \sum_{s \in \Omega} P(\mathbf{x}|s, \mathbf{y})P(s|\mathbf{y}). \quad (11.17)$$

Consider a candidate  $\mathbf{x}$  – if it has a support that does not align with any  $s \in \Omega$ , the probabilities  $P(\mathbf{x}|s, \mathbf{y})$  in the above summation are all zeros, and the overall probability of such a candidate solution is zero as well. Therefore, such a solution can hardly compete as being the MAP estimate, maximizing of the posterior.

Thus, we consider a potential solution  $\mathbf{x}$  with a specific support  $s^* \in \Omega$ . Clearly, since  $s^*$  is different from all other supports in  $\Omega \setminus s^*$ , only the term that corresponds to  $s^*$  remains in the above summation, and the probability for such  $\mathbf{x}$  reduces to  $P(\mathbf{x}|\mathbf{y}) = P(\mathbf{x}|s^*, \mathbf{y})P(s^*|\mathbf{y})$ . In fact, as we consider a specific  $s^*$  on which  $\mathbf{x}$  is supported, we can replace  $\mathbf{x}$  by  $\mathbf{x}_s$ , as the remaining entries are known to be zeros. Using this rationale, the MAP estimation is simplified to

$$\hat{\mathbf{x}}^{MAP} = \arg \max_{\mathbf{x}} P(\mathbf{x}|\mathbf{y}) = \arg \max_{s \in \Omega, \mathbf{x}_s} P(\mathbf{x}_s|s, \mathbf{y})P(s|\mathbf{y}). \quad (11.18)$$

This means that our search for the support  $s$  and the vector  $\mathbf{x}_s$  that corresponds to this support are coupled into one process. In a way, one could regard this as a model selection problem, where the MAP chooses the model (the support  $s$ ) that best explains the measurements. Using Bayes' rule, we can rewrite the above probability expression as

$$\begin{aligned} P(\mathbf{x}_s|s, \mathbf{y})P(s|\mathbf{y}) &= \frac{P(\mathbf{y}|s, \mathbf{x}_s)P(\mathbf{x}_s|s)}{P(\mathbf{y}|s)} \cdot \frac{P(\mathbf{y}|s)P(s)}{P(\mathbf{y})} \\ &= \frac{P(\mathbf{y}|s, \mathbf{x}_s)P(\mathbf{x}_s|s)P(s)}{P(\mathbf{y})}. \end{aligned}$$

The MAP estimator is further simplified by removing the division by  $P(\mathbf{y})$ , because it is constant with respect to the unknowns, leading to

$$\begin{aligned} \hat{\mathbf{x}}^{MAP} &= \arg \max_{\mathbf{x}} P(\mathbf{x}|\mathbf{y}) \\ &= \arg \max_{\mathbf{x}_s, s} P(\mathbf{y}|s, \mathbf{x}_s)P(\mathbf{x}_s|s)P(s). \end{aligned} \quad (11.19)$$

If we assume that  $s$  is known, as the expression  $P(\mathbf{x}_s|s)$  suggests, the random vector  $\mathbf{x}_s$  becomes a multivariate Gaussian with zero mean and  $\sigma_x^2 \mathbf{I}$  as its covariance matrix, and thus

$$P(\mathbf{x}_s|s) = \frac{1}{(2\pi)^{k/2}\sigma_x^k} \exp\left\{-\frac{\|\mathbf{x}_s\|_2^2}{2\sigma_x^2}\right\}. \quad (11.20)$$

Similarly, for a known  $\mathbf{x}_s$  and support  $s$ , the measurement  $\mathbf{y}$  is a sum of a deterministic vector  $\mathbf{H}\mathbf{A}_s\mathbf{x}_s$  and the random noise  $\mathbf{e}$ , and thus this vector is also Gaussian,

$$P(\mathbf{y}|s, \mathbf{x}_s) = \frac{1}{(2\pi)^{q/2}\sigma_e^q} \exp\left\{-\frac{\|\mathbf{H}\mathbf{A}_s\mathbf{x}_s - \mathbf{y}\|_2^2}{2\sigma_e^2}\right\}. \quad (11.21)$$

Plugging these two expressions into the MAP definition in Equation (11.19) and applying a  $-\log$  on the obtained expression, we get an equivalent optimization problem of the form

$$\hat{\mathbf{x}}^{MAP} = \arg \min_{\mathbf{x}, s} \left\{ \frac{\|\mathbf{H}\mathbf{A}_s\mathbf{x}_s - \mathbf{y}\|_2^2}{2\sigma_e^2} + \frac{\|\mathbf{x}_s\|_2^2}{2\sigma_x^2} + k \log(\sqrt{2\pi}\sigma_x) - \log(P(s)) \right\}.$$

The term  $k \log(\sqrt{2\pi}\sigma_x)$  comes from the normalization factor of the probability  $P(\mathbf{x}_s|s)$ . If  $k$  is fixed, this term can be disregarded, but when it is not, it must be inserted, as it affects the number of non-zeros,  $k$ , in the optimal solution.

For example, if we replace the originally assumed support-prior,  $P(s) = \delta(|s| - k)$ , by the more general option,  $P(s) = \text{Const} \cdot \exp(-\alpha|s|)$ , this leads to

$$\hat{\mathbf{x}}^{MAP} = \arg \min_{\mathbf{x}, s} \left\{ \frac{\|\mathbf{H}\mathbf{A}_s\mathbf{x}_s - \mathbf{y}\|_2^2}{2\sigma_e^2} + \frac{\|\mathbf{x}_s\|_2^2}{2\sigma_x^2} + (\alpha + \log(\sqrt{2\pi}\sigma_x))|s| \right\}. \quad (11.22)$$

Written differently, and exploiting that fact that  $|s| = \|\mathbf{x}\|_0$ , we get

$$\hat{\mathbf{x}}^{MAP} = \arg \min_{\mathbf{x}} \left\{ \frac{\|\mathbf{H}\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2}{2\sigma_e^2} + \frac{\|\mathbf{x}\|_2^2}{2\sigma_x^2} + (\alpha + \log(\sqrt{2\pi}\sigma_x))\|\mathbf{x}\|_0 \right\}. \quad (11.23)$$

This formulation reveals the connection between this estimation and the problems we have posed and treated in earlier chapters – this is a variation of the  $(P_0^\epsilon)$  problem. Interestingly, the term  $\|\mathbf{x}\|_2^2$  did not appear in these chapters, and its presence here is part of the more exact formulation that we derive, and in particular the fact that the non-zeros in  $\mathbf{x}$  are Gaussian.

Let us return to the assumption that only supports with a pre-specified cardinality  $k$  are permitted (i.e.,  $P(s) = \delta(|s| - k)$ ), and  $P(s)$  is uniform over all these, being equal to  $1/\binom{m}{k}$ . Thus, the last term can be discarded from Equation (11.22) as it does not influence the maximization. For a fixed  $s$ , the minimization posed in Equation (11.22) is easy, and we have seen it already in the context of developing the oracle,

$$\hat{\mathbf{x}}_s^* = \left( \frac{1}{\sigma_e^2} \mathbf{A}_s^T \mathbf{H}^T \mathbf{H} \mathbf{A}_s + \frac{1}{\sigma_x^2} \mathbf{I} \right)^{-1} \frac{1}{\sigma_e^2} \mathbf{A}_s^T \mathbf{H}^T \mathbf{y} \quad (11.24)$$

$$= \frac{1}{\sigma_e^2} \mathbf{Q}_s^{-1} \mathbf{A}_s^T \mathbf{H}^T \mathbf{y}.$$

However, we have to sweep over all possible supports  $s \in \Omega$ , compute this expression, plug it into the penalty function posed in Equation (11.22), and choose the one that leads to the minimal value. Plugging this expression into the penalty as suggested gives (after some tedious algebraic steps):

$$\begin{aligned} \text{Penalty} &= \frac{\|\mathbf{H}\mathbf{A}_s \mathbf{x}_s^* - \mathbf{y}\|_2^2}{2\sigma_e^2} + \frac{\|\mathbf{x}_s^*\|_2^2}{2\sigma_x^2} \\ &= \frac{\|\frac{1}{\sigma_e^2} \mathbf{H}\mathbf{A}_s \mathbf{Q}_s^{-1} \mathbf{A}_s^T \mathbf{H}^T \mathbf{y} - \mathbf{y}\|_2^2}{2\sigma_e^2} + \frac{\|\frac{1}{\sigma_e^2} \mathbf{Q}_s^{-1} \mathbf{A}_s^T \mathbf{H}^T \mathbf{y}\|_2^2}{2\sigma_x^2} \\ &= \mathbf{y}^T \left( \frac{1}{2\sigma_e^6} (\mathbf{H}\mathbf{A}_s \mathbf{Q}_s^{-1} \mathbf{A}_s^T \mathbf{H}^T)^2 - \frac{1}{\sigma_e^4} (\mathbf{H}\mathbf{A}_s \mathbf{Q}_s^{-1} \mathbf{A}_s^T \mathbf{H}^T) \right. \\ &\quad \left. + \frac{1}{2\sigma_e^2} \mathbf{I} + \frac{1}{2\sigma_e^4 \sigma_x^2} \mathbf{H}\mathbf{A}_s \mathbf{Q}_s^{-2} \mathbf{A}_s^T \mathbf{H}^T \right) \mathbf{y} \\ &= \frac{\|\mathbf{y}\|_2^2}{2\sigma_e^2} - \frac{1}{\sigma_e^4} \mathbf{y}^T \mathbf{H}\mathbf{A}_s \mathbf{Q}_s^{-1} \left( \mathbf{I} - \frac{1}{2\sigma_x^2} \mathbf{Q}_s^{-1} \right. \\ &\quad \left. - \frac{1}{2\sigma_e^2} \mathbf{A}_s^T \mathbf{H}^T \mathbf{H}\mathbf{A}_s \mathbf{Q}_s^{-1} \right) \mathbf{A}_s^T \mathbf{H}^T \mathbf{y}. \\ &= \frac{\|\mathbf{y}\|_2^2}{2\sigma_e^2} - \frac{1}{2\sigma_e^4} \mathbf{y}^T \mathbf{H}\mathbf{A}_s \mathbf{Q}_s^{-1} \mathbf{A}_s^T \mathbf{H}^T \mathbf{y}. \end{aligned} \tag{11.25}$$

In the last step in the above equation we have used the fact that the term in the parenthesis is equal to  $0.5\mathbf{I}$ . This expression suggests that the search for the MAP support should be done by maximizing the term

$$\text{Val}(s) = \mathbf{y}^T \mathbf{H}\mathbf{A}_s \mathbf{Q}_s^{-1} \mathbf{A}_s^T \mathbf{H}^T \mathbf{y} = \|\mathbf{Q}_s^{-0.5} \mathbf{A}_s^T \mathbf{H}^T \mathbf{y}\|_2^2$$

This implies a sweep through all  $\binom{m}{k}$  possible supports in  $\Omega$ , an impossible task in general. To summarize, MAP estimation is obtained by

$$\begin{aligned} \hat{\mathbf{x}}^{MAP} &= \frac{1}{\sigma_e^2} \mathbf{Q}_{s^*}^{-1} \mathbf{A}_{s^*}^T \mathbf{H}^T \mathbf{y} \\ \text{where } s^* &= \arg \max_{s \in \Omega} \|\mathbf{Q}_s^{-0.5} \mathbf{A}_s^T \mathbf{H}^T \mathbf{y}\|_2^2. \end{aligned} \tag{11.26}$$

In the case where  $\mathbf{H}\mathbf{A}$  is square and unitary, this estimation can be further simplified. The matrix  $\mathbf{Q}_s$  becomes proportional to  $\mathbf{I}$ , and thus does not influence the above maximization. Using the notation  $\beta = \mathbf{A}^T \mathbf{H}^T \mathbf{y}$ , the MAP optimal support is found by computing this vector and choosing its  $k$  largest entries in absolute value. Thus, as opposed to the general MAP that cannot be practically computed, the unitary case admits a closed-form and cheap solution.

Just before turning to approximations of the MAP estimation, we note the following. An alternative way to define the MAP estimation is by maximizing the posterior  $P(s|\mathbf{y})$ , i.e., seeking the most probable support that explains the measured data. Using Bayes' rule, this would be equivalent to the maximization of  $P(\mathbf{y}|s)P(s)$ , where  $P(\mathbf{y}|s)$  is a multivariate Gaussian distribution that can be built by marginalizing over  $\mathbf{x}_s$  (see Equation (11.31) and beyond). In the case considered here (with a fixed cardinality  $|s| = k$ ), this approach leads to a very similar solution as the one we derive above. In more complex scenarios, though, this approach may lead to an entirely different and more stable result—see a discussion about this issue in the paper by Turek et al.

### 11.4.2 Approximating the MAP Estimator

For general cardinality  $k$ , the process described above is infeasible, as it requires a sweep through all  $\binom{m}{k}$  possibilities, and thus there is no hope to provide an exact MAP estimate for our problem. Interestingly, for  $k = 1$ , we have only  $\binom{m}{1} = m$  options to consider and this becomes practical. Indeed, denoting the  $m$  columns of the matrix  $\mathbf{H}\mathbf{A}$  by  $\tilde{\mathbf{a}}_i$  for  $i = 1, 2, \dots, m$ , for  $k = 1$  the above expression uses these single columns maximizing the term

$$\text{Val}(i) = \frac{\|\tilde{\mathbf{a}}_i^T \mathbf{y}\|_2^2}{\frac{\|\tilde{\mathbf{a}}_i\|_2^2}{\sigma_e^2} + \frac{1}{\sigma_x^2}}. \quad (11.27)$$

If we further assume that all the columns  $\tilde{\mathbf{a}}_i$  have the same  $\ell_2$ -norm, this expression simply suggests to maximize the absolute inner product of the measurement vector with each column in  $\mathbf{H}\mathbf{A}$  in the quest for the best column (and thus, support of cardinality  $k = 1$ ).

Based on this insight, one could propose a greedy algorithm that accumulates the  $k$  elements of the support one by one – apply the process for  $k = 1$  and choose the first support element. Then proceed to the next stage by adding one column to the support, performing  $m - 1$  tests (since one column has been already chosen). This process should be repeated  $k$  times, accumulating the  $k$  elements of the support in a greedy fashion. This leads to a variant of the Matching-Pursuit algorithm, as this method too relies on maximizing inner products between the columns in  $\mathbf{H}\mathbf{A}$  and the measurements. Thus, we can regard the OMP (or any of its variants) as an approximation of the MAP for the estimation problem we have posed.

As a side note we add that the OMP is slightly different from the greedy algorithm that emerges above, and this might suggest that modifications to the OMP are required to better suit our estimation goals, depending on the validity of the proposed model. For example, considering the second stage in the above-proposed greedy method, assume that  $i_1$  is the index chosen in the first stage, and we are searching now for the second one,  $i_2$ . The term to maximize is



$$\begin{aligned}
\text{Val}(i_2) &= \mathbf{y}^T \mathbf{H} \mathbf{A}_s \mathbf{Q}_s^{-1} \mathbf{A}_s^T \mathbf{H}^T \mathbf{y} \\
&= \begin{bmatrix} \tilde{\mathbf{a}}_{i_1}^T \mathbf{y} & \tilde{\mathbf{a}}_{i_2}^T \mathbf{y} \end{bmatrix} \begin{bmatrix} \frac{\|\tilde{\mathbf{a}}_{i_1}\|_2^2}{\sigma_e^2} + \frac{1}{\sigma_x^2} & \frac{\tilde{\mathbf{a}}_{i_1}^T \tilde{\mathbf{a}}_{i_2}}{\sigma_e^2} \\ \frac{\tilde{\mathbf{a}}_{i_1}^T \tilde{\mathbf{a}}_{i_2}}{\sigma_e^2} & \frac{\|\tilde{\mathbf{a}}_{i_2}\|_2^2}{\sigma_e^2} + \frac{1}{\sigma_x^2} \end{bmatrix}^{-1} \begin{bmatrix} \tilde{\mathbf{a}}_{i_1}^T \mathbf{y} \\ \tilde{\mathbf{a}}_{i_2}^T \mathbf{y} \end{bmatrix}
\end{aligned} \tag{11.28}$$

and this is clearly different from computing the residual from  $\mathbf{y}$  after removal of the first column and then repeating the  $k = 1$  test again. While the above numerical process might seem daunting, there are numerical shortcuts that can be proposed, such that every iteration requires roughly the same amount of computations as the first iteration.

Returning to the experiment for which we have demonstrated the oracle's behavior in Figure 11.1, we can now add the MAP performance. Figure 11.2 shows the empirical MAP error, obtained by computing the estimate  $\hat{\mathbf{x}}^{MAP}$  using Equation (11.26), and evaluating the error directly by  $\|\mathbf{x} - \hat{\mathbf{x}}^{MAP}\|_2^2$ . This figure also shows an approximation of the MAP by the OMP. The method used here is different from the ones discussed above – we employ the very same OMP as discussed in Chapter 5, extracting a solution with cardinality  $\|\mathbf{x}\|_0 = 3$ . Using this as the chosen support for our approximate estimate, we compute the final result as an oracle estimate over this support. As can be seen, this approximation is quite close to the exhaustive MAP that sweeps over all  $\binom{m}{k} = 4060$  supports in searching the optimal one.

## 11.5 The MMSE Estimation

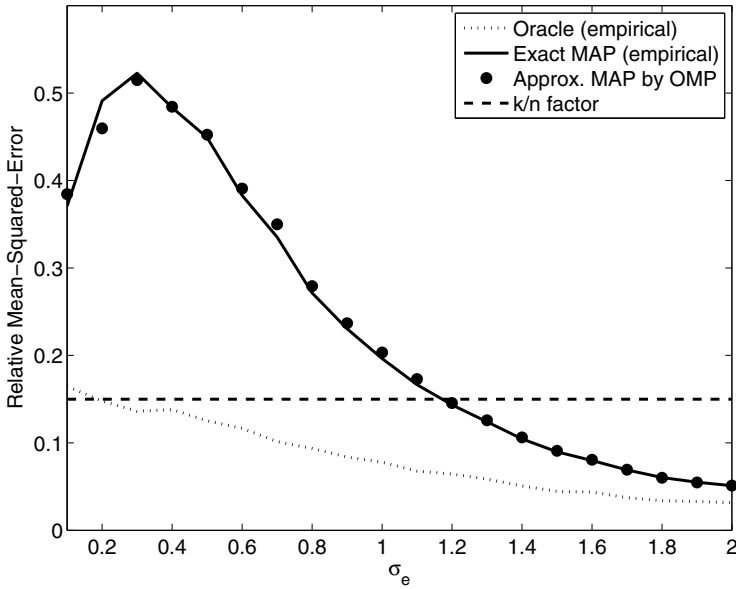
### 11.5.1 Developing the MMSE Estimator

We have already seen that the MMSE estimate is given by  $E(\mathbf{x}|\mathbf{y})$ , and using the expression for the posterior probability in Equation (11.17) this gives

$$\begin{aligned}
E(\mathbf{x}|\mathbf{y}) &= \int_{\mathbf{x}} \mathbf{x} P(\mathbf{x}|\mathbf{y}) d\mathbf{x} \\
&= \sum_{s \in \Omega} P(s|\mathbf{y}) \int_{\mathbf{x}} \mathbf{x} P(\mathbf{x}|s, \mathbf{y}) d\mathbf{x}
\end{aligned} \tag{11.29}$$

The inner integral stands for the MMSE estimate for the case where the support  $s$  is known, and this is the oracle we have already seen, given by

$$\begin{aligned}
\int_{\mathbf{x}} \mathbf{x} P(\mathbf{x}|s, \mathbf{y}) d\mathbf{x} &= E(\mathbf{x}|\mathbf{y}, s) = \hat{\mathbf{x}}_s^{oracle} \\
&= \left( \frac{1}{\sigma_e^2} \mathbf{A}_s^T \mathbf{H}^T \mathbf{H} \mathbf{A}_s + \frac{1}{\sigma_x^2} \mathbf{I} \right)^{-1} \frac{1}{\sigma_e^2} \mathbf{A}_s^T \mathbf{H}^T \mathbf{y}.
\end{aligned} \tag{11.30}$$



**Fig. 11.2** The exact (and thus exhaustive) and the approximated MAP performance as a function of the input noise. The approximation is obtained by the plain OMP used to detect the support, and then an oracle formula over this support. Results are shown relative to the input noise energy  $no\sigma_e^2$ .

The above already indicates that the MMSE estimator is about to be a weighted average of many “oracle” solutions, each with a different support, and each weighted by its likelihood to be correct, defined as  $P(s|\mathbf{y})$ . We therefore turn now to develop an expression for this probability.

Using Bayes’ rule again, we have that  $P(s|\mathbf{y}) = P(\mathbf{y}|s)P(s)/P(\mathbf{y})$ . The probability  $P(s)$  is constant over all the supports  $s \in \Omega$  such that  $|s| = k$ , and thus can be omitted. Similarly,  $P(\mathbf{y})$  can be considered as a normalizing factor in our development of  $P(s|\mathbf{y})$  and thus can be disregarded as well. Thus, we end up with the need to evaluate  $P(\mathbf{y}|s)$ . This probability can be written by marginalization over  $\mathbf{x}_s$ , integrating over all possible values of the non-zeros in the support  $s$ ,

$$P(\mathbf{y}|s) = \int_{\mathbf{x}_s} P(\mathbf{y}|s, \mathbf{x}_s)P(\mathbf{x}_s|s)d\mathbf{x}_s. \quad (11.31)$$

This integral is  $k$ -dimensional, averaging  $P(\mathbf{y}|s, \mathbf{x}_s)$  over all  $\mathbf{x}_s \in \mathbb{R}^k$ . The probability  $P(\mathbf{x}_s|s)$  is known to be Gaussian,  $\mathcal{N}(\mathbf{0}, \sigma_x^2 \mathbf{I})$ . Similarly, once both the support and the non-zero values are given ( $s$  and  $\mathbf{x}_s$ ),  $\mathbf{y}$  is also Gaussian, with  $\mathbf{H}\mathbf{A}_s\mathbf{x}_s$  as its mean, and  $\sigma_e^2 \mathbf{I}$  as its covariance. Plugging these into Equation (11.31), we have

$$P(\mathbf{y}|s) = \int_{\mathbf{x}_s} P(\mathbf{y}|s, \mathbf{x}_s)P(\mathbf{x}_s|s)d\mathbf{x}_s \quad (11.32)$$

$$\propto \int_{\mathbf{v} \in \mathbb{R}^k} \exp \left\{ -\frac{\|\mathbf{H}\mathbf{A}_s \mathbf{v} - \mathbf{y}\|_2^2}{2\sigma_e^2} - \frac{\|\mathbf{v}\|_2^2}{2\sigma_x^2} \right\} d\mathbf{v}.$$

Rearrangement of the term in the exponent gives

$$\begin{aligned} \frac{\|\mathbf{H}\mathbf{A}_s \mathbf{v} - \mathbf{y}\|_2^2}{2\sigma_e^2} + \frac{\|\mathbf{v}\|_2^2}{2\sigma_x^2} &= \frac{1}{2}(\mathbf{v} - \mathbf{h}_s)^T \mathbf{Q}_s (\mathbf{v} - \mathbf{h}_s) \\ &\quad - \frac{1}{2} \mathbf{h}_s^T \mathbf{Q}_s \mathbf{h}_s + \frac{1}{2\sigma_e^2} \|\mathbf{y}\|_2^2, \end{aligned} \quad (11.33)$$

where

$$\begin{aligned} \mathbf{Q}_s &= \frac{1}{\sigma_e^2} \mathbf{A}_s^T \mathbf{H}^T \mathbf{H} \mathbf{A}_s + \frac{1}{\sigma_x^2} \mathbf{I} \\ &\text{and} \\ \mathbf{h}_s &= \frac{1}{\sigma_e^2} \mathbf{Q}_s^{-1} \mathbf{A}_s^T \mathbf{H}^T \mathbf{y}, \end{aligned} \quad (11.34)$$

$\mathbf{Q}_s$  is the very same matrix defined earlier, and  $\mathbf{h}_s$  is the oracle solution for the support  $s$ . Thus, plugging these into Equation (11.32) we obtain

$$\begin{aligned} P(\mathbf{y}|s) &\propto \int_{\mathbf{v} \in \mathbb{R}^k} \exp \left\{ -\frac{\|\mathbf{H}\mathbf{A}_s \mathbf{v} - \mathbf{y}\|_2^2}{2\sigma_e^2} - \frac{\|\mathbf{v}\|_2^2}{2\sigma_x^2} \right\} d\mathbf{v} \\ &= \exp \left\{ \frac{1}{2} \mathbf{h}_s^T \mathbf{Q}_s \mathbf{h}_s - \frac{1}{2\sigma_e^2} \|\mathbf{y}\|_2^2 \right\} \\ &\quad \cdot \int_{\mathbf{v} \in \mathbb{R}^k} \exp \left\{ -\frac{1}{2} (\mathbf{v} - \mathbf{h}_s)^T \mathbf{Q}_s (\mathbf{v} - \mathbf{h}_s) \right\} d\mathbf{v}. \end{aligned} \quad (11.35)$$

The integral in the above expression is over the complete volume of a multivariate Gaussian distribution, apart from the missing constant coefficient. Thus,

$$\int_{\mathbf{v} \in \mathbb{R}^k} \exp \left\{ -\frac{1}{2} (\mathbf{v} - \mathbf{h}_s)^T \mathbf{Q}_s (\mathbf{v} - \mathbf{h}_s) \right\} d\mathbf{v} = \sqrt{(2\pi)^k \det(\mathbf{Q}_s^{-1})}. \quad (11.36)$$

This leads to the result

$$\begin{aligned} P(\mathbf{y}|s) &\propto \exp \left\{ \frac{1}{2} \mathbf{h}_s^T \mathbf{Q}_s \mathbf{h}_s \right\} \cdot \sqrt{\det(\mathbf{Q}_s^{-1})} \\ &= \exp \left\{ \frac{1}{2} \mathbf{h}_s^T \mathbf{Q}_s \mathbf{h}_s + \frac{1}{2} \log(\det(\mathbf{Q}_s^{-1})) \right\}. \end{aligned} \quad (11.37)$$

Notice that we have removed the terms  $\exp \{-\|\mathbf{y}\|_2^2/2\sigma_e^2\}$  and  $(2\pi)^k$ , since both can be absorbed into the proportionality constant, as we are only interested in the terms that depend on the support  $s$ . To conclude, in order to compute the MMSE estimate, we should compute the terms

$$\begin{aligned}
q_s &= \exp \left\{ \frac{1}{2} \mathbf{h}_s^T \mathbf{Q}_s \mathbf{h}_s + \frac{1}{2} \log(\det(\mathbf{Q}_s^{-1})) \right\} \\
&= \exp \left\{ \frac{\mathbf{y}^T \mathbf{H} \mathbf{A}_s \mathbf{Q}_s^{-1} \mathbf{A}_s^T \mathbf{H}^T \mathbf{y}}{2\sigma_e^4} + \frac{1}{2} \log(\det(\mathbf{Q}_s^{-1})) \right\}
\end{aligned} \tag{11.38}$$

for all  $s \in \Omega$ , and normalize them to have a unit sum. The values obtained are in fact  $P(s|\mathbf{y})$  to be used in the weighted sum in Equation (11.30) for the construction of the MMSE estimate. We should draw the reader's attention to the term  $\mathbf{y}^T \mathbf{H} \mathbf{A}_s \mathbf{Q}_s^{-1} \mathbf{A}_s^T \mathbf{H}^T \mathbf{y}$  in the above expression – this is the very same term used for testing the supports in the MAP (see Equation (11.25)).

To summarize, the MMSE estimation is obtained by combining Equations (11.29) and (11.30), and the definition of  $q_s$  as given in Equation (11.38):

$$\hat{\mathbf{x}}^{MMSE} = \frac{\sum_{s \in \Omega} q_s \left( \frac{1}{\sigma_e^2} \mathbf{A}_s^T \mathbf{H}^T \mathbf{H} \mathbf{A}_s + \frac{1}{\sigma_x^2} \mathbf{I} \right)^{-1} \frac{1}{\sigma_e^2} \mathbf{A}_s^T \mathbf{H}^T \mathbf{y}}{\sum_{s \in \Omega} q_s}. \tag{11.39}$$

Since we have seen that the MAP estimation admits a closed-form and cheap solution for the unitary case, it is of interest to know if a similar claim can be made with respect to the MMSE estimation. We shall not provide the analysis here, but we do state that such a closed-form formula can and has been established (see recent papers by Protter et. al. and Turek et. al.) for the MMSE estimation, avoiding the combinatorial complexity altogether. Furthermore, under what we have defined as model  $\mathcal{G}_2$  for generating  $\mathbf{x}$ , the MMSE estimator takes the shape of a shrinkage operation, just like the MAP. The difference between the two is thus only in the shape of the shrinkage curve.

### 11.5.2 Approximating the MMSE Estimator

Whereas the exact MAP requires a sweep through all  $s \in \Omega$  in a search for the most appropriate support, the MMSE requires a similar sweep, but for computing an oracle estimate for each, and averaging these estimates with proper weights. Clearly, the MMSE is even more complex than the MAP, and as infeasible as the MAP in general, having an exponentially large possibilities to explore.

Just as we did for the MAP, we consider the simplified case of  $k = 1$ . In this case,  $q_i$   $i = 1, 2, \dots, m$  in Equation (11.38) become

$$\begin{aligned}
q_i &= \exp \left\{ \frac{\mathbf{y}^T \mathbf{H} \mathbf{A}_i \mathbf{Q}_i^{-1} \mathbf{A}_i^T \mathbf{H}^T \mathbf{y}}{2\sigma_e^4} + \frac{1}{2} \log(\det(\mathbf{Q}_i^{-1})) \right\} \\
&= \exp \left\{ \frac{1}{2\sigma_e^2} \cdot \frac{(\tilde{\mathbf{a}}_i^T \mathbf{y})^2}{\|\tilde{\mathbf{a}}_i\|_2^2 + \frac{\sigma_e^2}{\sigma_x^2}} + \frac{1}{2} \log \left( \frac{\|\tilde{\mathbf{a}}_i\|_2^2}{\sigma_e^2} + \frac{1}{\sigma_x^2} \right) \right\}.
\end{aligned} \tag{11.40}$$

These can be computed and used for a weighted averaging of all the  $m$  cardinality  $k = 1$  possibilities. Assuming that the columns  $\tilde{\mathbf{a}}_i$  are normalized, these weights are proportional to  $\exp(C(\tilde{\mathbf{a}}_i^T \mathbf{y})^2)$ , giving preference to atoms parallel to the measurement vector.

An approximation algorithm for the MMSE can be proposed for the case of  $k > 1$ , based on pruning of small  $q_i$ . If we choose a few (say  $p \ll m$ ) indices  $1 \leq i \leq m$  for which the value  $q_i$  are the largest and omit the rest, we keep most of the important elements in the weighted average. Turning to  $k = 2$ , rather than exploring all  $\binom{m}{2}$  support possibilities, we explore only the  $p(m - 1)$  options that correspond to the  $p$  indices already chosen for  $k = 1$ , and for each of these we test the  $m - 1$  remaining atoms as candidate second ones. For these  $p(m - 1)$  tests we have a quality assessment attached,  $q_s$ , as in Equation (11.38). Thus, we can again omit most of the elements and choose the best  $p$ . The process then repeats for  $k = 3$  and beyond, and when we reach  $k$  we remain with a set of  $p$  chosen supports that were found greedily to give high probability  $q_s$ . A weighted average of these could replace the exhaustive MMSE, approximating its outcome.

An alternative algorithm for approximating the MMSE for a general value of  $k$  can be proposed based on randomization. This approach relies on a random sampling of the supports  $s \in \Omega$  in order to draw a sub-set to average upon. For  $k = 1$ , the  $q_i$  (after their normalization) are the exact probabilities  $P(s|\mathbf{y})$  for supports with cardinality  $|s| = 1$ . Rather than choosing the top ones deterministically, as proposed above, we could draw indices at random with probabilities proportional to  $q_i$ . Having a set of such drawn indices, a plain average of the solutions for these indices is an unbiased estimate for the MMSE weighted average.

The above process could in principle be done for  $|s| = k > 1$  as well, randomly drawing supports and averaging their oracle estimates. However, for the approximation of the average to be relatively accurate, we should use too-many random draws, which makes this idea impractical. Instead, we may suggest to greedily accumulate the drawn supports. For example, after choosing the first atom  $i_1$  as described above, we compute the  $m - 1$  probabilities  $q_i = P(s = [i_1, i]|\mathbf{y})$  for pairs that include the chosen atom  $i_1$  as the first item, and considering all the  $m - 1$  possibilities for  $i$ . Using this distribution, we can draw the second atom and this way choose  $i_2$ . As this process proceeds, we arrive at a support of cardinality  $k$  by  $k$  such random draws, each from a set of  $O(m)$  probabilities (it is in fact  $m - j + 1$  probabilities at the  $j$ -th step).

The Random-OMP algorithm is a simplified version of the above procedure, following the same general structure of choosing the next atom based on the  $m$  scalar probabilities  $q_i$ . Starting with the ordinary OMP, we replace the deterministic mechanism for choosing the best atom to add to the support, by a random draw, just as above. However, the probabilities  $q_i$  are re-evaluated by Equation (11.40) after each greedy step, considering the residual signal as the one to project onto the atoms in  $\mathbf{A}$ . Running this random algorithm several times, we obtain several (potentially) different supports and their oracle estimates. These are averaged (without weighting!!), leading to the approximate MMSE.

**Task:** Approximate a MMSE estimate given in Equation (11.39).

**Parameters:** The pair  $(\mathbf{A}, \mathbf{y})$ , the cardinality  $|s|$ , the values of  $\sigma_e$  and  $\sigma_x$ , and the number of runs  $J$ .

**Main Iteration:** Run the random algorithm below  $J$  times:

- **Initialization:** Initialize  $k = 0$ , and set
  - The initial solution  $\mathbf{x}^0 = \mathbf{0}$ .
  - The initial residual  $\mathbf{r}^0 = \mathbf{y} - \mathbf{A}\mathbf{x}^0 = \mathbf{y}$ .
  - The initial solution support  $\mathcal{S}^0 = \text{Support}\{\mathbf{x}^0\} = \emptyset$ .
- **Greedy Iteration:** Increment  $k$  by 1 and perform the following steps  $|s|$  times:
  - **Random Draw:** Draw  $j_0$  at random with probability proportional to  $q_i = \exp\left\{\frac{1}{2\sigma_e^2} \cdot \frac{(\mathbf{a}_i^T \mathbf{r}^{k-1})^2}{\|\mathbf{a}\|_2^2 + \frac{\sigma_e^2}{\sigma_x^2}} + \frac{1}{2} \log\left(\frac{\|\mathbf{a}_i\|_2^2}{\sigma_e^2} + \frac{1}{\sigma_x^2}\right)\right\}$ .
  - **Update Support:** Update the support by  $\mathcal{S}^k = \mathcal{S}^{k-1} \cup \{j_0\}$ .
  - **Increment Solution:** Compute  $\mathbf{x}^k$ , the minimizer of  $\|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2$  subject to  $\text{Support}\{\mathbf{x}\} = \mathcal{S}^k$ .
  - **Update Residual:** Compute  $\mathbf{r}^k = \mathbf{y} - \mathbf{A}\mathbf{x}^k$ .
- **Finalize the  $j$ -th Solution:** Set  $s = \mathcal{S}^k$ , the support detected, and compute the solution using the oracle's formula,

$$\hat{\mathbf{x}}_j = \left( \frac{1}{\sigma_e^2} \mathbf{A}_s^T \mathbf{A}_s + \frac{1}{\sigma_x^2} \mathbf{I} \right)^{-1} \frac{1}{\sigma_e^2} \mathbf{A}_s^T \mathbf{y}.$$

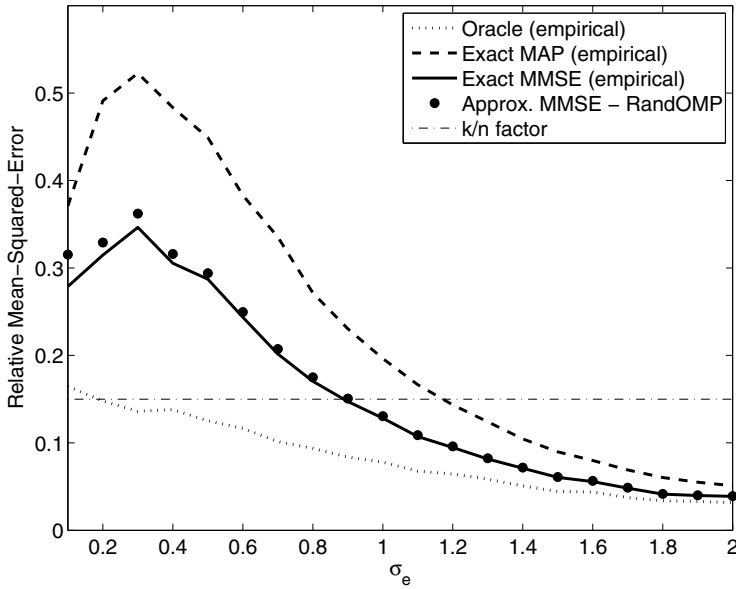
**Output:** The proposed approximation is obtained as a simple average of the  $J$  random results,

$$\hat{\mathbf{x}} = \frac{1}{J} \sum_{j=1}^J \hat{\mathbf{x}}_j.$$

**Fig. 11.3** Random-OMP for computing the approximate MMSE estimation (assuming  $\mathbf{H} = \mathbf{I}$  for simplicity).

One could regard the above two processes as approximate Gibbs samplers for the supports that constitute the peaks of the probability function  $P(\mathbf{y}|s)$ . The Random-OMP algorithm is depicted in [Figure 11.3](#).

We return again to the experiment behind [Figures 11.1](#) and [11.2](#), this time adding MMSE performance curves. Using Equation (11.39), we compute  $\hat{\mathbf{x}}^{MMSE}$  and evaluate the error directly by  $\|\mathbf{x} - \hat{\mathbf{x}}^{MMSE}\|_2^2$ . We also apply an approximation of the MMSE by the Random-OMP, by running it 25 times and plain-averaging the results. Since all the atoms in  $\mathbf{A}$  have a unit norm, the random draw of the atoms within the Random-OMP is proportional to  $\exp(C(\mathbf{a}_i^T \mathbf{y})^2)$ , with  $1/C = 2\sigma_e^2(1 + \sigma_e^2/\sigma_x^2)$  – this constant should be used by the Random-OMP function. Each run of the Random-OMP is used to extract the support, for which an oracle estimate is generated. These are averaged to obtain the final result.



**Fig. 11.4** The exact (and thus exhaustive) and the approximated MMSE performance as a function of the input noise. The approximation is obtained by averaging 25 runs of the Random-OMP. Results are shown relative to the input noise energy  $nc_e^2$ .

The results of this experiment are brought in [Figure 11.4](#), and we see that this approximation is quite close to the exhaustive MMSE, which has to sweep and average the oracle estimates over all  $\binom{m}{k} = 4060$  supports.

## 11.6 MMSE and MAP Errors

We have obtained expressions for the MAP and the MMSE estimates, and now we target the development of similar expressions for the actual MSE these estimators lead to. Our analysis starts with the error for a general estimate  $\hat{\mathbf{x}}$ , observing that it can be written as

$$\begin{aligned}
 E(\|\hat{\mathbf{x}} - \mathbf{x}\|_2^2) &= \int_{\mathbf{x} \in \mathbb{R}^n} \|\hat{\mathbf{x}} - \mathbf{x}\|_2^2 P(\mathbf{x}|\mathbf{y}) d\mathbf{x} \\
 &= \sum_{s \in \Omega} P(s|\mathbf{y}) \int_{\mathbf{x} \in \mathbb{R}^n} \|\hat{\mathbf{x}} - \mathbf{x}\|_2^2 P(\mathbf{x}|s, \mathbf{y}) d\mathbf{x},
 \end{aligned} \tag{11.41}$$

where we have used the marginalization of the posterior as proposed in Equation (11.17).

The integral above can be rewritten differently by modifying the term  $\|\hat{\mathbf{x}} - \mathbf{x}\|_2^2$ , adding and subtracting an oracle estimate  $\hat{\mathbf{x}}_s^{oracle}$  that corresponds to the known support  $s$ . This is not to be confused with the ideal oracle estimate for the complete estimation problem, for which we have to know the *true* support  $s^*$  that created the measurements  $\mathbf{y}$ . Thus, we use  $\|\hat{\mathbf{x}} - \mathbf{x}\|_2^2 = \|\hat{\mathbf{x}} - \hat{\mathbf{x}}_s^{oracle} + \hat{\mathbf{x}}_s^{oracle} - \mathbf{x}\|_2^2$ , yielding

$$\begin{aligned} \int_{\mathbf{x} \in \mathbb{R}^n} \|\hat{\mathbf{x}} - \mathbf{x}\|_2^2 P(\mathbf{x}|s, \mathbf{y}) d\mathbf{x} &= \int_{\mathbf{x} \in \mathbb{R}^n} \|\hat{\mathbf{x}}_s^{oracle} - \mathbf{x}\|_2^2 P(\mathbf{x}|s, \mathbf{y}) d\mathbf{x} \\ &+ \int_{\mathbf{x} \in \mathbb{R}^n} \|\hat{\mathbf{x}} - \hat{\mathbf{x}}_s^{oracle}\|_2^2 P(\mathbf{x}|s, \mathbf{y}) d\mathbf{x}. \end{aligned} \quad (11.42)$$

Note that the integral over the cross-term  $(\hat{\mathbf{x}}_s^{oracle} - \mathbf{x})^T (\hat{\mathbf{x}} - \hat{\mathbf{x}}_s^{oracle})$  vanishes, since the term  $(\hat{\mathbf{x}} - \hat{\mathbf{x}}_s^{oracle})$  is deterministic and can thus be moved outside the integration. The remaining expression in the integral is zero, since  $\hat{\mathbf{x}}_s^{oracle} = E(\mathbf{x}|\mathbf{y}, s)$  by definition.

Returning to Equation (11.42), the first term represents an oracle MSE for a given support  $s$ , and this has been already given in Equation (11.16) as

$$E\left(\|\hat{\mathbf{x}}_s^{oracle} - \mathbf{x}_s\|_2^2\right) = \text{trace} \left( \left[ \frac{1}{\sigma_e^2} \mathbf{A}_s^T \mathbf{H}^T \mathbf{H} \mathbf{A}_s + \frac{1}{\sigma_x^2} \mathbf{I} \right]^{-1} \right) = \text{trace}(\mathbf{Q}_s^{-1}).$$

The second term is even simpler: Since  $\|\hat{\mathbf{x}} - \hat{\mathbf{x}}_s^{oracle}\|_2^2$  is deterministic, it is moved outside the integration, and the remaining part is simply 1. Thus,

$$\int_{\mathbf{x} \in \mathbb{R}^n} \|\hat{\mathbf{x}} - \mathbf{x}\|_2^2 P(\mathbf{x}|s, \mathbf{y}) d\mathbf{x} = \|\hat{\mathbf{x}} - \hat{\mathbf{x}}_s^{oracle}\|_2^2 + \text{trace}(\mathbf{Q}_s^{-1}).$$

Using the fact that  $P(s|\mathbf{y}) \propto q_s$  as developed in Equation (11.38), we have

$$\begin{aligned} E(\|\hat{\mathbf{x}} - \mathbf{x}\|_2^2) &= \frac{1}{\sum_{s \in \Omega} q_s} \cdot \sum_{s \in \Omega} q_s \cdot \left[ \|\hat{\mathbf{x}} - \hat{\mathbf{x}}_s^{oracle}\|_2^2 \right. \\ &\quad \left. + \text{trace}(\mathbf{Q}_s^{-1}) \right]. \end{aligned} \quad (11.43)$$

By plugging into this expression  $\hat{\mathbf{x}} = \hat{\mathbf{x}}^{MMSE}$  we get the MMSE error. Interestingly, if we minimize the above expression with respect to  $\hat{\mathbf{x}}$  we obtain the MMSE estimator formula as given in Equation (11.39), since nulling the derivative of the above expression with respect to  $\hat{\mathbf{x}}$  gives

$$\begin{aligned} \frac{\partial E(\|\hat{\mathbf{x}} - \mathbf{x}\|_2^2)}{\partial \hat{\mathbf{x}}} &= 2 \frac{\sum_{s \in \Omega} q_s (\hat{\mathbf{x}} - \hat{\mathbf{x}}_s^{oracle})}{\sum_{s \in \Omega} q_s} = 0 \\ \Rightarrow \hat{\mathbf{x}}^{MMSE} &= \frac{\sum_{s \in \Omega} q_s \hat{\mathbf{x}}_s^{oracle}}{\sum_{s \in \Omega} q_s}. \end{aligned}$$

As an interesting example, for the case where  $\mathbf{H} = \mathbf{I}$  and  $\mathbf{A}$  is unitary, the term  $\mathbf{A}_s^T \mathbf{H}^T \mathbf{H} \mathbf{A}_s$  is an identity matrix of size  $|s| = k$ , and thus  $\text{trace}(\mathbf{Q}_s^{-1}) = k\sigma_x^2\sigma_e^2/(\sigma_x^2 + \sigma_e^2)$  is constant for any  $s \in \Omega$ . Turning to the value of  $q_s$ , we observe that for our



case  $q_s \propto \exp(C \cdot \|\beta_s\|_2^2 / 2\sigma_e^2)$ , where  $\beta_s = \mathbf{A}_s^T \mathbf{y}$  is a projection of  $\mathbf{y}$  onto the columns of the support  $s$ , and the constant is  $C = \sigma_x^2 / (\sigma_e^2 + \sigma_x^2)$ . Finally, the  $s$ -oracle is given by  $\hat{\mathbf{x}}_s^{oracle} = \sigma_x^2 \beta_s / (\sigma_e^2 + \sigma_x^2) = C \beta_s$ . Combining all these we get

$$E(\|\hat{\mathbf{x}} - \mathbf{x}\|_2^2) = \frac{\sum_{s \in \Omega} \exp\left(\frac{C \cdot \|\beta_s\|_2^2}{2\sigma_e^2}\right) \cdot \|\hat{\mathbf{x}} - c\beta_s\|_2^2}{\sum_{s \in \Omega} \exp\left(\frac{C \cdot \|\beta_s\|_2^2}{2\sigma_e^2}\right)} + \frac{k\sigma_x^2\sigma_e^2}{\sigma_x^2 + \sigma_e^2}. \quad (11.44)$$

Returning to the general case, we observe that the general error expression found in Equation (11.43) can be written differently by adding and subtracting  $\hat{\mathbf{x}}^{MMSE}$ , giving

$$\begin{aligned} E(\|\hat{\mathbf{x}} - \mathbf{x}\|_2^2) &= \frac{\sum_{s \in \Omega} q_s \cdot [\|\hat{\mathbf{x}} - \hat{\mathbf{x}}_s^{oracle}\|_2^2 + \text{trace}(\mathbf{Q}_s^{-1})]}{\sum_{s \in \Omega} q_s} \\ &= \frac{\sum_{s \in \Omega} q_s [\|\hat{\mathbf{x}} - \hat{\mathbf{x}}^{MMSE} + \hat{\mathbf{x}}^{MMSE} - \hat{\mathbf{x}}_s^{oracle}\|_2^2 + \text{trace}(\mathbf{Q}_s^{-1})]}{\sum_{s \in \Omega} q_s} \\ &= \|\hat{\mathbf{x}} - \hat{\mathbf{x}}^{MMSE}\|_2^2 \\ &\quad + \frac{\sum_{s \in \Omega} q_s \cdot [\|\hat{\mathbf{x}}^{MMSE} - \hat{\mathbf{x}}_s^{oracle}\|_2^2 + \text{trace}(\mathbf{Q}_s^{-1})]}{\sum_{s \in \Omega} q_s} \\ &= \|\hat{\mathbf{x}} - \hat{\mathbf{x}}^{MMSE}\|_2^2 + E(\|\hat{\mathbf{x}}^{MMSE} - \mathbf{x}\|_2^2). \end{aligned} \quad (11.45)$$

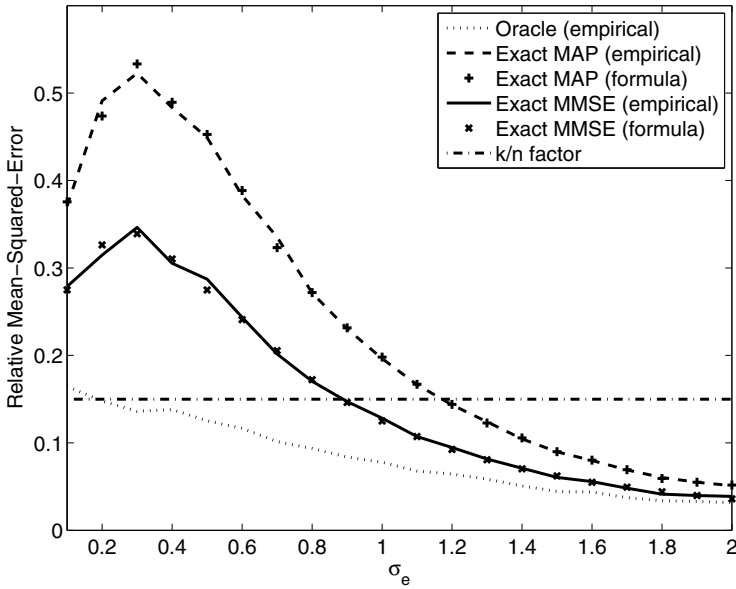
Again, the cross term  $(\hat{\mathbf{x}} - \hat{\mathbf{x}}^{MMSE})^T (\hat{\mathbf{x}}^{MMSE} - \hat{\mathbf{x}}_s^{oracle})$  drops from the above expression since the first part can be moved outside the summation, and the second part vanishes due to the formula of  $\hat{\mathbf{x}}^{MMSE}$  as a linear combination of oracle estimates. In particular, this means that

$$\begin{aligned} E(\|\hat{\mathbf{x}}^{MAP} - \mathbf{x}\|_2^2) &= \|\hat{\mathbf{x}}^{MAP} - \hat{\mathbf{x}}^{MMSE}\|_2^2 \\ &\quad + E(\|\hat{\mathbf{x}}^{MMSE} - \mathbf{x}\|_2^2). \end{aligned} \quad (11.46)$$

In Figure 11.5 we show the tendency of the formulae developed here for the MMSE and the MAP to describe accurately the empirical performance of these estimators. The figure shows the empirical results as in Figures 11.2 and 11.4, and on top the expected error is manifested for the MMSE in Equation (11.43) and for the MAP as in Equation (11.46).

## 11.7 More Experimental Results

All the simulations describes thus far correspond to a very-low dimensional problem, for which the exact and exhaustive estimators can be computed in reasonable time. We now turn to a very similar simulation in higher dimension. We use a random dictionary, just as before, but with size  $200 \times 400$ . We set the cardinality of



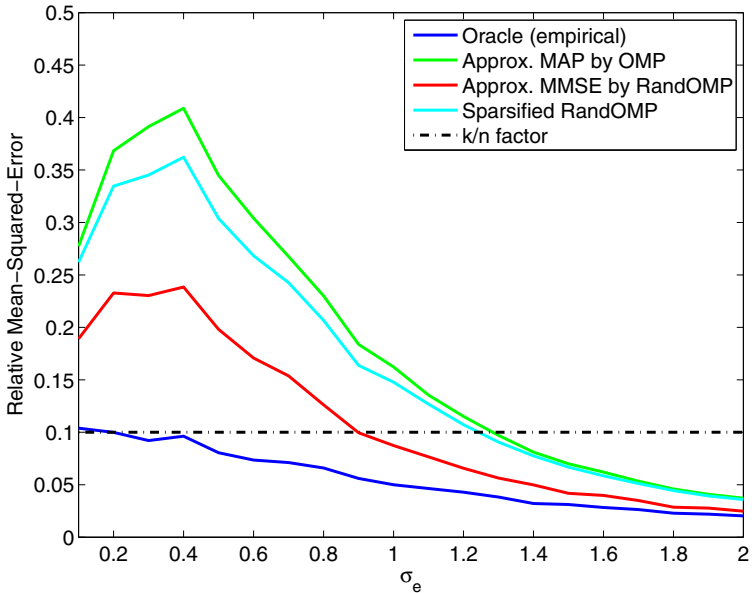
**Fig. 11.5** The MAP and the MMSE performance, presented both by their empirical evaluations, and by the formulae using the expected error formulae (11.43) and (11.46). Results are shown relative to the input noise energy  $n\sigma_e^2$ .

the unknown representation to be  $k = 20$ . Figure 11.6 shows the performance of the OMP and the Random-OMP (with 25 runs averaged), varying the additive noise variance, and averaging the results over 100 independent experiments. We see that the results resemble those that we have observed in the low-dimensional tests.

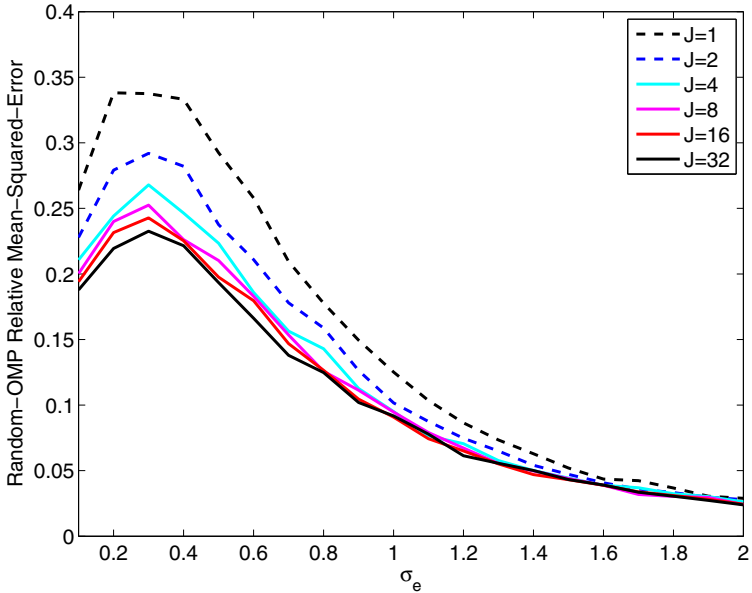
This figure contains a third and new estimator, referred to as the “Sparsified Random-OMP”. This estimate is obtained by taking the largest  $k$  entries in the Random-OMP solution, and defining this as the support of interest. Computing an oracle’s estimate over this support, we obtained this proposed estimation. In some problems, we desire to get an  $\ell_2$  accurate estimation, while also providing sparsity. This for example is the case in compression or recognition tasks.

We see that this approach gives performance somewhere between the previous two (OMP and Random-OMP). The fact that it loses in quality compared to the Random-OMP is quite expected, bearing in mind that the Random-OMP is a relatively good approximation of the MMSE estimation, one can offer no alternative that outperforms it. We also see that this estimator tends to be better than the OMP, indicating that this approach is not the MAP estimator in disguise.

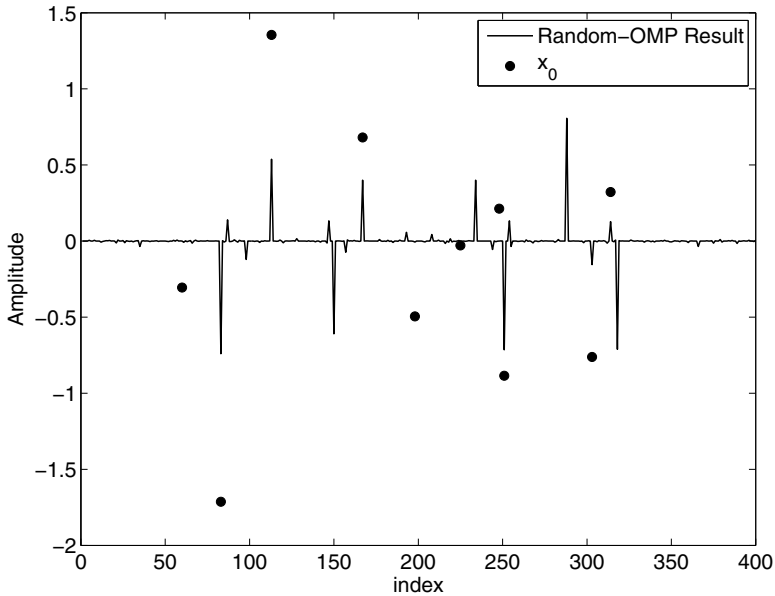
How many random runs should be applied to enable the Random-OMP to get close to the MMSE estimate? Figure 11.7 shows the performance obtained for varying number of such runs, using the same experiment settings as above. As can be observed, even 4 runs are sufficient to get quite close to the desired results.



**Fig. 11.6** The performance curves of the OMP (approximate MAP), the Random-OMP (approximate MMSE), and a sparsified version of the Random-OMP, as a function of the additive noise variance. Results are shown relative to  $n\sigma_e^2$ .



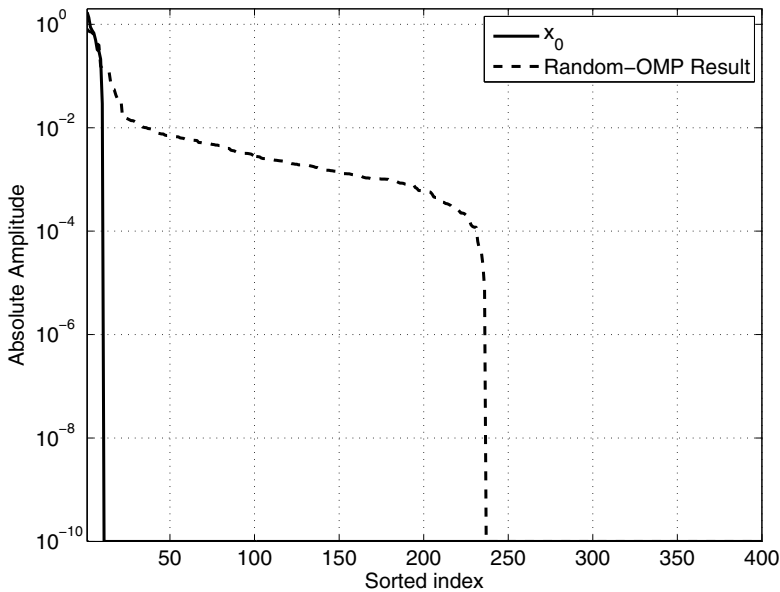
**Fig. 11.7** Random-OMP performance with varying number of random runs averaged.



**Fig. 11.8** The Random-OMP result ( $J = 100$ ) compared to the true representation  $\mathbf{x}_0$ .

Our last test focuses on the Random-OMP, and in particular on its tendency to provide a non-sparse solution. We perform a single estimation test, similar to the ones appearing above – the dictionary used is an over-complete DCT matrix of size  $200 \times 400$ , obtained by taking every second row from a unitary DCT matrix of size  $400 \times 400$ . The chosen parameters are  $k = 10$ ,  $\sigma_e = 0.2$ ,  $\sigma_x = 1$ , and  $J = 100$ . Figure 11.8 shows the Random-OMP result, and as expected, while the result tends to be close to the original representation  $\mathbf{x}_0$ , it is not sparse, due to the massive averaging that takes place in its computation. Figure 11.9 further investigates this behavior by presenting the sorted absolute values of these two representations, showing clearly that the Random-OMP solution tends to be dense. This tendency is further strengthened if the noise is stronger, or when the dictionary's coherence is bad.

The fact that the Random-OMP (and indeed, the MMSE it approximates) gives a non-sparse solution brings us back to the issue raised towards the end of Chapter 10. We have seen there that the best  $\lambda$  (in terms of MSE) leads to a dense representation solution. We can now explain this very simply as being an approximation of the MMSE, which is also not expected to be sparse. Furthermore, and more broadly, even if the unknown we seek to estimate is sparse, and even with a known cardinality, this does not mean that its MMSE estimate is going to be sparse. An interesting question with this regard is what is the  $k$ -sparse MMSE estimate for our problem? Is it similar to the sparsified Random-OMP we have introduced? This remains as an open question at the moment.



**Fig. 11.9** Sorted absolute value of the entries in the Random-OMP result and the true representation  $\mathbf{x}_0$ .

## 11.8 Summary

In the context of estimation theory, the *Sparse-Land* model leads to an optimization problem of the form defined as  $P_0^\epsilon$ , for the MAP estimator. This chapter clearly mapped this relationship, making it precise. A far more important achievement of the adopted estimation point of view, is the ability to consider the MMSE estimation. We have seen that while both these estimators lead to exponentially growing amounts of computations, reasonable approximations of them are within reach. This analysis gives a more solid foundation for the quest for the sparsest representation and the use of pursuit techniques in signal processing applications.

## Further Reading

1. F. Abramovich, T. Sapatinas and B.W. Silverman, Wavelet thresholding via a Bayesian approach, *J. R. Statist. Soc. B*, 60:725–749, 1998.
2. A. Antoniadis, J. Bigot, and T. Sapatinas, Wavelet estimators in nonparametric regression: a comparative simulation study, *J. Stat. Software*, 6(6):1–83, 2001.

3. M. Clyde and E.I. George, Empirical Bayes estimation in wavelet nonparametric regression. In *Bayesian Inference in Wavelet Based Models*, P. Muller and B. Vidakovic (Eds.), Lect. Notes Statist., 141:309–322, New York, Springer-Verlag, 1998.
4. M. Clyde and E.I. George, Flexible empirical Bayes estimation for wavelets, *J. R. Statist. Soc. B*, 62:681–698, 2000.
5. M. Clyde, G. Parmigiani and B. Vidakovic, Multiple shrinkage and subset selection in wavelets, *Biometrika*, 85:391–401, 1998.
6. M. Elad and I. Yavneh, A weighted average of sparse representations is better than the sparsest one alone, *IEEE Transactions on Information Theory*, 55(10):4701–4714, October 2009.
7. J. Turek, I. Yavneh, M. Protter, and M. Elad, On MMSE and MAP denoising under sparse representation modeling over a unitary dictionary, submitted to *Applied Computational Harmonic Analysis*.
8. E. Larsson and Y. Selen, Linear regression with a sparse parameter vector, *IEEE Transactions on Signal Processing*, 55:451–460, 2007.
9. S. Mallat and Z. Zhang, Matching pursuits with time-frequency dictionaries, *IEEE Trans. on Signal Processing*, 41(12):3397–3415, 1993.
10. P. Moulin and J. Liu, Analysis of multiresolution image denoising schemes using generalized Gaussian and complexity priors, *IEEE Trans. Inf. Theory*, 45(3):909–919, April 1999.
11. M. Protter, I. Yavneh and M. Elad, Closed-form MMSE for denoising signals under sparse-representation modelling, The IEEE 25-th Convention of Electrical and Electronics Engineers in Israel, Eilat Israel, December 3-5, 2008.
12. M. Protter, I. Yavneh and M. Elad, Closed-Form MMSE estimation for signal denoising under sparse representation modelling over a unitary dictionary, submitted to *IEEE Trans. on Signal Processing*.
13. E.P. Simoncelli and E.H. Adelson, Noise removal via Bayesian wavelet coring, in Proc. ICIP, Lausanne, Switzerland, pp. 379-382, September 1996.
14. P. Schnitter, L. C. Potter, and J. Ziniel, Fast Bayesian matching pursuit, Proc. Workshop on Information Theory and Applications (ITA), (La Jolla, CA), Jan. 2008.
15. P. Schintter, L.C. Potter, and J. Ziniel, Fast Bayesian matching pursuit: Model uncertainty and parameter estimation for sparse linear models, submitted to *IEEE Transactions on Signal Processing*.



## Chapter 12

# The Quest for a Dictionary

A fundamental ingredient in the definition of *Sparse-Land*'s signals and its deployment to applications is the dictionary  $\mathbf{A}$ . How can we wisely choose  $\mathbf{A}$  to perform well on the signals in question? This is the topic of this chapter, and our emphasis is put on learning methods for dictionaries, based on a group of examples.

### 12.1 Choosing versus Learning

In the quest for the proper dictionary to use in applications, one line of work considers choosing pre-constructed dictionaries, such as undecimated wavelets (as in Chapter 10), steerable wavelets, contourlets, curvelets, and more. Many of these recently proposed dictionaries are tailored specifically to images, and in particular to stylized 'cartoon-like' image content, assumed to be piecewise smooth and with smooth boundaries.

Some of these proposed dictionaries (which are often referred to also as transforms) are accompanied by a detailed theoretical analysis establishing the sparsity of the representation coefficients for such simplified content of signals. This is typically done in terms of the rate of decay of the  $M$ -term approximation – the representation of the signal using the best  $M$  non-zeros from the transform coefficients.

Alternatively, one can use a tunable selection of a dictionary, in which a basis or Frame is generated under the control of a particular parameter (discrete or continuous). The two most known examples that belong to this category are the wavelet packets and the bandelets. The wavelet packets, due to Coifman et al., suggest a control over the time-frequency subdivision, tuned to optimize performance over a specified instance of a signal. The bandelets, due to Mallat et al., are spatially adapted to better treat images, by orienting the regular wavelet atoms along the principal direction of the local area processed.

While pre-constructed or adapted dictionaries typically lead to fast transforms (of complexity  $O(n \log n)$  instead of  $nm$  in a direct use), they are typically limited in their ability to sparsify the signals they are designed to handle. Furthermore, most



of those dictionaries are restricted to signals/images of a certain type, and cannot be used for a new and arbitrary family of signals of interest. This leads us to yet another approach for obtaining dictionaries that overcomes these limitations – by adopting a learning point-of-view.

This learning option starts by building a *training database* of signal instances, similar to those anticipated in the application, and constructing an empirically-learned dictionary, in which the generating atoms come from the underlying empirical data, rather than from some theoretical model. Such a dictionary can then be used in the application as a fixed and redundant dictionary. We shall explore this third option in more detail in this chapter.

As opposed to the pre-constructed and adapted dictionaries, the learning method is able to adapt to any family of signals that complies with the *Sparse-Land* model. However, this comes at the price of a much higher computational load – learned dictionaries are held as explicit structure-less matrices, and using them in applications involves many more computations, compared to the pre-constructed dictionaries. Another shortcoming of the training methodology is its restriction<sup>1</sup> to low-dimensional signals. This is why handling of images is done with such dictionaries on small patches. We will discuss ways to overcome these two limitations later in this chapter.

## 12.2 Dictionary-Learning Algorithms

We now turn to discuss the learning methodology for constructing  $\mathbf{A}$ . Assume that a training database  $\{\mathbf{y}_i\}_{i=1}^M$  is given, and thought to have been generated by some fixed but unknown model  $\mathcal{M}_{\{\mathbf{A}, k_0, \alpha, \epsilon\}}$ . Can this training database allow us to identify the generating model, and specifically the dictionary  $\mathbf{A}$ ? This rather difficult problem has been studied initially by Field and Olshausen in 1996, who were motivated by an analogy between the atoms of a dictionary and the population of simple cells in the visual cortex. They thought that learning a dictionary empirically might model evolutionary processes that led to the existing collection of simple cells; and they indeed were able to find a rough empirical match between the properties of a learned dictionary and some known properties of the population of simple cells.

Later work by Lewicki, Engan, Rao, Gribonval, Aharon, and others, extended their methodology and algorithm in various ways. Here we describe two training mechanisms, the first named Method of Optimal Directions (MOD) by Engan et al., and the second named K-SVD, by Aharon et al..

---

<sup>1</sup> At least for now. Work on the construction of learned multi-scale dictionaries is underway, and hopefully this shortcoming should be removed in the near future.

### 12.2.1 Core Questions in Dictionary-Learning

Assume that  $\epsilon$  – the model deviation – is known, and our aim is the estimation of  $\mathbf{A}$ . Consider the following optimization problem:

$$\min_{\mathbf{A}, \{\mathbf{x}_i\}_{i=1}^M} \sum_{i=1}^M \|\mathbf{x}_i\|_0 \quad \text{subject to} \quad \|\mathbf{y}_i - \mathbf{A}\mathbf{x}_i\|_2 \leq \epsilon, \quad 1 \leq i \leq M. \quad (12.1)$$

This problem describes each given signal  $\mathbf{y}_i$  as the sparsest representation  $\mathbf{x}_i$  over the unknown dictionary  $\mathbf{A}$ , and aims to jointly find the proper representations and the dictionary. Clearly, if a solution has been found such that every representation has  $k_0$  or fewer non-zero entries, a candidate feasible model  $\mathcal{M}$  has been found.

The roles of the penalty and the constraints in Equation (12.1) might also be reversed, if we choose to constrain the sparsity and obtain the best fit for it,

$$\min_{\mathbf{A}, \{\mathbf{x}_i\}_{i=1}^M} \sum_{i=1}^M \|\mathbf{y}_i - \mathbf{A}\mathbf{x}_i\|_2^2 \quad \text{subject to} \quad \|\mathbf{x}_i\|_0 \leq k_0, \quad 1 \leq i \leq M. \quad (12.2)$$

Are these two problems properly posed? Do they have meaningful solutions? There are some obvious indeterminacies (scaling and permutation of the columns). Still, if we fix a scale and ordering, is there a meaningful solution to the problems posed in Equations (12.1) and (12.2) in general? As we show next, the answers to these questions are positive.

In terms of well-posedness of the dictionary-learning problem presented above, a fundamental question is whether there is a uniqueness property underlying this problem, implying that only one dictionary exists, that sparsely explains the set of training vectors. Surprisingly, at least for the case of  $\epsilon = 0$ , there is an answer to this question, as shown by Aharon et al.. Suppose there exists a dictionary  $\mathbf{A}_0$  and a sufficiently diverse database of examples, all of which are representable using at most  $k_0 < \text{spark}(\mathbf{A}_0)/2$  atoms. Then, up to a re-scaling and permutation of the columns,  $\mathbf{A}_0$  is the unique dictionary that achieves this sparsity for all the elements in the training database.

Some readers may prefer to think in terms of matrix factorizations. Concatenate all the database vectors column-wise, forming an  $n \times M$  matrix  $\mathbf{Y}$ , and similarly, all the corresponding sparse representations into a matrix  $\mathbf{X}$  of size  $m \times M$ ; thus the dictionary satisfies  $\mathbf{Y} = \mathbf{A}\mathbf{X}$  in the noiseless case. The problem of discovering an underlying dictionary is thus the same as the problem of discovering a factorization of the matrix  $\mathbf{Y}$  as  $\mathbf{A}\mathbf{X}$  where  $\mathbf{A}$  and  $\mathbf{X}$  have the indicated sizes, and  $\mathbf{X}$  has sparse columns. The matrix factorization viewpoint connects this problem with related problems of nonnegative matrix factorization and in particular, sparse nonnegative matrix factorization.

### 12.2.2 The MOD Algorithm

Clearly, there is no general practical algorithm for solving problem (12.1) or (12.2), for the same reasons that there is no general practical algorithm for solving  $(P_0)$ , only more so. However, just as with  $(P_0)$ , the lack of general guarantees is no reason not to try heuristic methods and see how they perform in specific cases.

We can view the problem posed in Equation (12.1) as a nested minimization problem: an inner minimization of the number of nonzeros in the representation vectors  $\mathbf{x}_i$ , for a given fixed  $\mathbf{A}$  and an outer minimization over  $\mathbf{A}$ . A strategy of alternating minimization thus seems very natural; at the  $k$ -th step, we use the dictionary  $\mathbf{A}_{(k-1)}$  from the  $k-1$ -th step and solve  $M$  instances of  $(P_0^\epsilon)$ , one for each database entry  $\mathbf{y}_i$ , and each using the dictionary  $\mathbf{A}_{(k-1)}$ . This gives us the matrix  $\mathbf{X}_{(k)}$ , and we then solve for  $\mathbf{A}_{(k)}$  by Least-Squares,

$$\begin{aligned} \mathbf{A}_{(k)} &= \arg \min_{\mathbf{A}} \|\mathbf{Y} - \mathbf{A}\mathbf{X}_{(k)}\|_F^2 \\ &= \mathbf{Y}\mathbf{X}_{(k)}^T (\mathbf{X}_{(k)}\mathbf{X}_{(k)}^T)^{-1} \\ &= \mathbf{Y}\mathbf{X}_{(k)}^+, \end{aligned} \quad (12.3)$$

where we evaluate the error using a Frobenius norm. We may also re-scale the columns of the obtained dictionary. We increment  $k$  and unless we have satisfied a convergence criterion, we repeat the above loop. Such a Block-Coordinate-Relaxation algorithm has been proposed by Engan et al., and termed Method of Optimal Directions (MOD). This algorithm is described in [Figure 12.1](#).

[Figure 12.2](#) demonstrates the MOD for a synthetic experiment. A random dictionary (iid Gaussian entries, normalized columns) of size  $30 \times 60$  is generated, and from it we produce 4,000 signal examples, each by a random combination of 4 atoms, with coefficients drawn from the normal distribution  $\mathcal{N}(0, 1)$ . Each such signal is further contaminated by a random zero-mean Gaussian noise with  $\sigma = 0.1$ . This implies a signal-to-noise ratio of  $\approx 8$  in this experiment. Performing 50 iterations of the MOD on this set of signals, we attempt to recover the original dictionary. The MOD is used with a fixed cardinality  $k_0 = 4$ , and initialized by adopting the first 60 examples as the dictionary atoms. [Figure 12.2](#) (right) shows the average representation error as a function of the iteration.

Since this is a synthetic experiment, where the ground-truth dictionary is known, we can evaluate how close the estimated dictionary is to its objective. [Figure 12.2](#) (left) presents a graph of the relative count of recovered atoms. An atom  $\mathbf{a}_i$  from the true dictionary is considered as recovered if it finds a match  $\hat{\mathbf{a}}_j$  in the estimated dictionary, such that  $|\mathbf{a}_i^T \hat{\mathbf{a}}_j| > 0.99$ . Note that if the estimated atoms were to be contaminated by the same amount of noise as the training data (i.e.,  $\hat{\mathbf{a}}_j = \mathbf{a}_j + \mathbf{e}_j$  with  $\mathbf{e}_j \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ ), the variance of this inner-product would have been

$$E\left((\mathbf{a}_i^T \hat{\mathbf{a}}_j - \mathbf{a}_i^T \mathbf{a}_j)^2\right) = \mathbf{a}_i^T E\left(\mathbf{e}_j \mathbf{e}_j^T\right) \mathbf{a}_i = \sigma^2 = 0.01,$$

**Task:** Train a dictionary  $\mathbf{A}$  to sparsely represent the data  $\{\mathbf{y}_i\}_{i=1}^M$ , by approximating the solution to the problem posed in Equation (12.2).

**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 examples.
- **Normalization:** Normalize the columns of  $\mathbf{A}_{(0)}$ .

**Main Iteration:** Increment  $k$  by 1, and apply

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

$$\hat{\mathbf{x}}_i = \arg \min_{\mathbf{x}} \|\mathbf{y}_i - \mathbf{A}_{(k-1)}\mathbf{x}\|_2^2 \quad \text{subject to} \quad \|\mathbf{x}\|_0 \leq k_0.$$

obtaining sparse representations  $\hat{\mathbf{x}}_i$  for  $1 \leq i \leq M$ . These form the matrix  $\mathbf{X}_{(k)}$ .

- **MOD Dictionary Update Stage:** Update the dictionary by the formula

$$\mathbf{A}_{(k)} = \arg \min_{\mathbf{A}} \|\mathbf{Y} - \mathbf{A}\mathbf{X}_{(k)}\|_F^2 = \mathbf{Y}\mathbf{X}_{(k)}^T (\mathbf{X}_{(k)}\mathbf{X}_{(k)}^T)^{-1}.$$

- **Stopping Rule:** If the change in  $\|\mathbf{Y} - \mathbf{A}_{(k)}\mathbf{X}_{(k)}\|_F^2$  is small enough, stop. Otherwise, apply another iteration.

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

**Fig. 12.1** The MOD dictionary-learning algorithm.

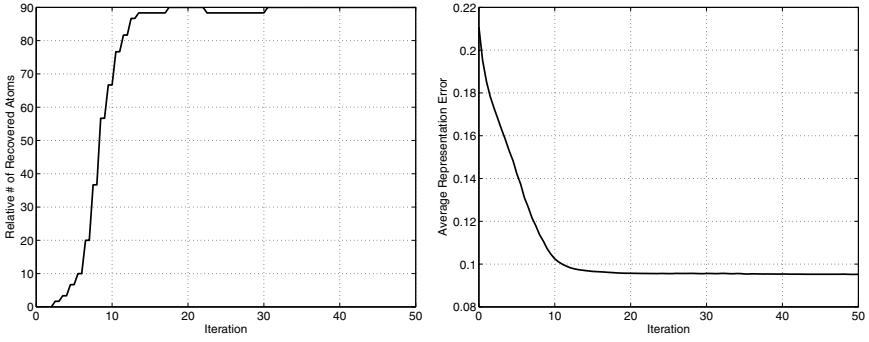
where we have exploited the fact that  $\mathbf{a}_i^T \mathbf{a}_i = 1$ . Thus, requiring a standard deviation of 0.01 in these inner products is 10 times smaller than the expected fluctuation, and as such, it is quite demanding.

Both graphs show that MOD performs very well on this test, recovering most of the atoms in the original dictionary, and doing so in a matter of 20 – 30 iterations. Moreover, using 4 atoms per each example, MOD manages to get to an average representation error which is below 0.1 – the noise level. This means that the found dictionary is a feasible solution to our problem.

### 12.2.3 The K-SVD Algorithm

A different update rule for the dictionary can be proposed, in which the atoms (i.e., columns) in  $\mathbf{A}$  are handled sequentially. This leads to the K-SVD algorithm, as developed by Aharon et al. Keeping all the columns fixed apart from the  $j_0$ -th one,  $\mathbf{a}_{j_0}$ , this column can be updated along with the coefficients that multiply it in  $\mathbf{X}$ . We isolate the dependency on  $\mathbf{a}_{j_0}$  by rewriting (12.3) as <sup>2</sup>

<sup>2</sup> To simplify notation, we now omit the iteration number  $k$ .



**Fig. 12.2** The MOD behavior in a synthetic experiment. The left graph presents the relative number (in %) of correctly recovered atoms, and the right one shows the average representation error.

$$\begin{aligned} \|\mathbf{Y} - \mathbf{AX}\|_F^2 &= \left\| \mathbf{Y} - \sum_{j=1}^m \mathbf{a}_j \mathbf{x}_j^T \right\|_F^2 \\ &= \left\| \left( \mathbf{Y} - \sum_{j \neq j_0} \mathbf{a}_j \mathbf{x}_j^T \right) - \mathbf{a}_{j_0} \mathbf{x}_{j_0}^T \right\|_F^2. \end{aligned} \quad (12.4)$$

In this description,  $\mathbf{x}_j^T$  stands for the  $j$ -th row of  $\mathbf{X}$ . The update step targets both  $\mathbf{a}_{j_0}$  and  $\mathbf{x}_{j_0}^T$ , and refers to the term in parentheses,

$$\mathbf{E}_{j_0} = \mathbf{Y} - \sum_{j \neq j_0} \mathbf{a}_j \mathbf{x}_j^T \quad (12.5)$$

as a known pre-computed error matrix.

The optimal  $\mathbf{a}_{j_0}$  and  $\mathbf{x}_{j_0}^T$  minimizing Equation (12.4) are the rank-1 approximation of  $\mathbf{E}_{j_0}$ , and can be obtained via an SVD, but this typically would yield a dense vector  $\mathbf{x}_{j_0}^T$ , implying that we increase the number of non-zeros in the representations in  $\mathbf{X}$ .

In order to minimize this term while keeping the cardinalities of all the representations fixed, a subset of the columns of  $\mathbf{E}_{j_0}$  should be taken – those that correspond to the signals from the example-set that 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 way, we allow only the existing non-zero coefficients in  $\mathbf{x}_{j_0}^T$  to vary, and the cardinalities are preserved.

Therefore, we define a restriction operator,  $\mathbf{P}_{j_0}$ , that multiplies  $\mathbf{E}_{j_0}$  from the right to remove the non-relevant columns. The matrix  $\mathbf{P}_{j_0}$  has  $M$  rows (the number of overall examples), and  $M_{j_0}$  columns (the number of examples using the  $j_0$ -th atom). We define  $(\mathbf{x}_{j_0}^R)^T = \mathbf{x}_{j_0}^T \mathbf{P}_{j_0}$  as the restriction on the row  $\mathbf{x}_{j_0}^T$ , choosing the non-zero entries only.

For the sub-matrix,  $\mathbf{E}_{j_0} \mathbf{P}_{j_0}$ , a rank-1 approximation via SVD can be applied, updating both the atom  $\mathbf{a}_{j_0}$  and the corresponding coefficients in the sparse represen-

tations,  $\mathbf{x}_{j_0}^R$ . This simultaneous update may lead to a substantial speedup in the convergence of the training algorithm. Figure 12.3 describes the K-SVD algorithm in detail.

Note that one need not use the full SVD as only the first rank part of  $\mathbf{E}_{j_0} \mathbf{P}_{j_0}$  is required. An alternative numerical approach for finding  $\mathbf{a}_{j_0}$  and  $\mathbf{x}_{j_0}^R$  can be proposed, following the same block-coordinate rationale: Fixing  $\mathbf{a}_{j_0}$ , we can update  $\mathbf{x}_{j_0}^R$  by a plain Least-Squares that solving

$$\min_{\mathbf{x}_{j_0}^R} \|\mathbf{E}_{j_0} \mathbf{P}_{j_0} - \mathbf{a}_{j_0} (\mathbf{x}_{j_0}^R)^T\|_F^2 \Rightarrow \mathbf{x}_{j_0}^R = \frac{\mathbf{P}_{j_0}^T \mathbf{E}_{j_0}^T \mathbf{a}_{j_0}}{\|\mathbf{a}_{j_0}\|_2^2}.$$

Once updated, it is kept fixed, and we update  $\mathbf{a}_{j_0}$  by

$$\min_{\mathbf{a}_{j_0}} \|\mathbf{E}_{j_0} \mathbf{P}_{j_0} - \mathbf{a}_{j_0} (\mathbf{x}_{j_0}^R)^T\|_F^2 \Rightarrow \mathbf{a}_{j_0} = \frac{\mathbf{E}_{j_0} \mathbf{P}_{j_0} \mathbf{x}_{j_0}^R}{\|\mathbf{x}_{j_0}^R\|_2^2}.$$

Few such rounds of updates for these two unknowns is sufficient for getting the desired dictionary update.

Interestingly, if the above process is considered for the case where  $k_0 = 1$ , and constraining the representation coefficients to be binary (1 or 0), this problem reduces to a simple clustering task. Furthermore, in such a case, both the above training algorithms simplify to become the well-known K-means algorithm. While each iteration of K-means computes the means of  $K$  different subsets, the K-SVD algorithm performs an SVD for each of the  $K$  different sub-matrices, and thus the name K-SVD ( $K$  is used as the number of columns in  $\mathbf{A}$ ).

The fact that the dictionary-learning problem is a generalization of clustering reveals more on the learning task considered here:

- Just as for the clustering problem and the K-Means which aims to solve it, we cannot guarantee that the MOD and K-SVD algorithms obtain the global minimum of the penalty function posed in Equation (12.2). Indeed, even a local minimum solution cannot be guaranteed, as these algorithms may get stuck on a saddle-point steady-state solution.
- Still on the matter of convergence, since the pursuit applied in the sparse coding stage may get to a sub-optimal solution, we cannot guarantee a monotonic non-increasing penalty value as a function of the iterations. However, we can modify the algorithm such that after the pursuit we adopt only those examples that show a descent in the representation error, thus guaranteeing an overall descent.
- The similarity between the K-Means and the K-SVD (and MOD) enables us to exploit the vast empirical experience accumulated on ways to accelerate the K-Means. These techniques include a gradual change in the number of atoms within the learning procedure, a similar change in the number of atoms to use per example, a multi-scale approach, where the examples are represented by multi-scale pyramids, and the training is done on lower-dimensions first, and more. We shall not dwell on these techniques here.

**Task:** Train a dictionary  $\mathbf{A}$  to sparsely represent the data  $\{\mathbf{y}_i\}_{i=1}^M$ , by approximating the solution to the problem posed in Equation (12.2).

**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 examples.
- **Normalization:** Normalize the columns of  $\mathbf{A}_{(0)}$ .

**Main Iteration:** Increment  $k$  by 1, and apply

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

$$\hat{\mathbf{x}}_i = \arg \min_{\mathbf{x}} \|\mathbf{y}_i - \mathbf{A}_{(k-1)}\mathbf{x}\|_2^2 \quad \text{subject to} \quad \|\mathbf{x}\|_0 \leq k_0.$$

obtaining sparse representations  $\hat{\mathbf{x}}_i$  for  $1 \leq i \leq M$ . These form the matrix  $\mathbf{X}_{(k)}$ .

- **K-SVD Dictionary-Update Stage:** Use the following procedure to update the columns of the dictionary and obtain  $\mathbf{A}_{(k)}$ : Repeat for  $j_0 = 1, 2, \dots, m$ 
  - Define the group of examples that use the atom  $\mathbf{a}_{j_0}$ ,

$$\Omega_{j_0} = \{i \mid 1 \leq i \leq M, \mathbf{X}_{(k)}[j_0, i] \neq 0\}.$$

- 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  $j$ 'th rows in the matrix  $\mathbf{X}_{(k)}$ .

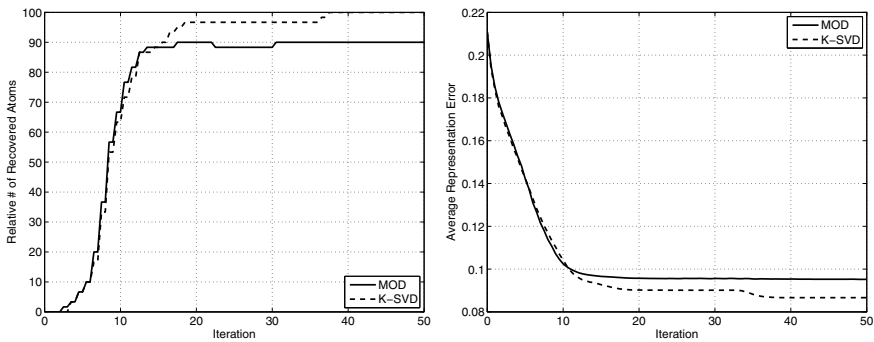
- Restrict  $\mathbf{E}_{j_0}$  by choosing only the columns corresponding to  $\Omega_{j_0}$ , and obtain  $\mathbf{E}_{j_0}^R$ .
- Apply SVD decomposition  $\mathbf{E}_{j_0}^R = \mathbf{U}\mathbf{\Delta}\mathbf{V}^T$ . Update the dictionary atom  $\mathbf{a}_{j_0} = \mathbf{u}_1$ , and the representations by  $\mathbf{x}_{j_0}^R = \mathbf{\Delta}[1, 1] \cdot \mathbf{v}_1$ .
- **Stopping Rule:** If the change in  $\|\mathbf{Y} - \mathbf{A}_{(k)}\mathbf{X}_{(k)}\|_F^2$  is small enough, stop. Otherwise, apply another iteration.

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

**Fig. 12.3** The K-SVD dictionary-learning algorithm.

Figure 12.4 presents the results obtained with the K-SVD for the synthetic experiment described in the previous section. For both algorithms we add a correction stage after the dictionary update, in which, if an atom in the dictionary is rarely used, or if it is found to be too similar to another atom in the dictionary, it is replaced by the worst-represented example.

As can be seen, in this experiment the K-SVD gives slightly better results, compared to the MOD, both in terms of the final outcome (100% dictionary recovery with an average representation error of 0.09), and the speed of convergence. Gener-



**Fig. 12.4** The K-SVD behavior in a synthetic experiment. The left graph presents the relative number (in %) of correctly recovered atoms, and the right one shows the average representation error.

ally speaking, these two algorithms behave similarly and comparably, with a small advantage to the K-SVD.

We now turn to present an elementary experiment performed on real data. We train a dictionary for sparsely representing patches of size  $8 \times 8$  extracted from the image Barbara, shown in Figure 12.5. This image is of size  $512 \times 512$ , implying that there are  $(512-7)^2 = 255,025$  possible patches, considering all overlaps. We extract one tenth of these patches (uniformly spread) to train on, using both the MOD and the K-SVD with 50 iterations. In both cases, we set the target cardinality to  $k_0 = 4$  atoms per patch.

Both algorithms are initialized with a 2D separable DCT dictionary of size  $64 \times 121$ . This dictionary is created by forming first a 1D-DCT matrix  $\mathbf{A}_{1D}$  of size  $8 \times 11$ , where the  $k$ -th atom ( $k = 1, 2, \dots, 11$ ) is given by  $\mathbf{a}_k^{1D} = \cos((i-1)(k-1)\pi/11)$ ,  $i = 1, 2, \dots, 8$ . All atoms apart from the first are further processed by removing their mean. The final dictionary is obtained by a Kronecker-product  $\mathbf{A}_{2D} = \mathbf{A}_{1D} \otimes \mathbf{A}_{1D}$ , which implies separability of this matrix when operating on a 2D image-patch.

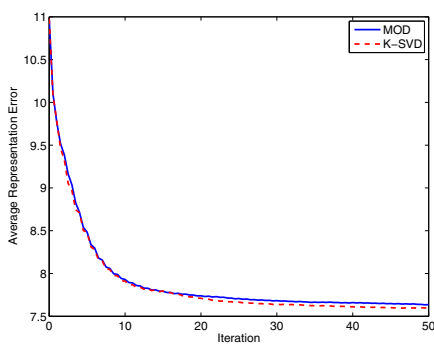
Figure 12.6 presents the average representation error on the training set for both algorithms. We see that even though the DCT is considered to be a good dictionary for sparsely representing image patches, both algorithms improve the error substantially beyond its initial value. The MOD and the K-SVD perform comparably in this test, leading to almost the same error. The DCT and the two resulting dictionaries are presented in Figure 12.7.

Both learning procedures lead to a dictionary that consists of piecewise smooth atoms and textured ones, because of the nature of the image trained on. One may be tempted to believe that the MOD and K-SVD dictionaries trained on are identical or nearly so, however, using the same procedure mentioned above for measuring the distance between the two dictionaries, we find that there is an overlap of only  $\approx 14\%$  of the atoms, the rest being considered as different (implying that their cross-inner product is below 0.99).



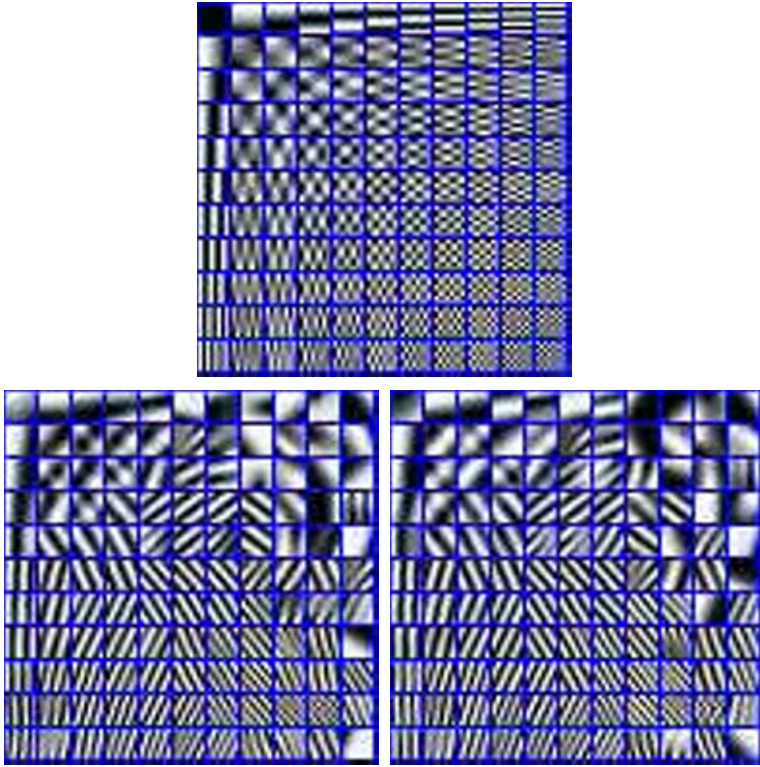


**Fig. 12.5** The original image Barbara, on which the dictionary training is evaluated.



**Fig. 12.6** The average representation error on the training set for the MOD and the K-SVD algorithms, as a function of the iteration.

Since we train the dictionaries using a small fraction of the image patches available to us, a natural question to pose is whether the found dictionaries sparsify the rest of the patches as well. Performing OMP on all the patches, the average representation error obtained with the DCT dictionary is 10.97, the MOD dictionary gives an error of 7.82, and the K-SVD leads to an error of 7.80. Both trained dictionaries lead to an error very close to the errors observed on the training set and far better



**Fig. 12.7** The DCT (top), the MOD (bottom left) and the K-SVD (bottom right) dictionaries, trained using  $8 \times 8$  patches from the image Barbara.

than the DCT, implying that the sparsification property is generalized well for the other patches in the image.

## 12.3 Training Structured Dictionaries

So far the discussion has concentrated on the core dictionary learning problem and two of its possible solutions. We should remind the reader that, along side with the elegance and success of this methodology, there are also important shortcomings to it, and these include:

- **Speed and Memory Problems:** The trained dictionary  $\mathbf{A} \in \mathbb{R}^{n \times m}$  is held and used as an explicit matrix. When plugged to applications, each multiplication of the form  $\mathbf{A}\mathbf{x}$  or  $\mathbf{A}^T\mathbf{y}$  requires  $nm$  operations. In comparison, structured dictionaries require much less computation. For example, a separable 2D dictionary of the

same size<sup>3</sup> requires  $2n\sqrt{m}$ . As another example, a wavelet dictionary requires only  $O(m)$  operations. Thus, the explicit storage and use of learned dictionaries as described above is much less efficient, compared to traditional transform methods.

- **Restriction to Low-Dimensions:** The learning process shown above is restricted to low-dimensional signals, i.e.,  $n \leq 1000$  is a reasonable limit not to be surpassed. An attempt to go beyond this raises a series of problems, starting with a need for an intolerable amount of training data, a very slow learning procedure, and a risk of over-fitting because of the too many free parameters in the sought dictionary. These problems pose a hard restriction on how learned dictionaries should be used in practice. For example, in a series of image processing algorithms (some of them shown in later chapters), learned dictionaries are used on small image patches, rather than on the whole image.
- **Operating on a Single-Scale:** Learned dictionaries as obtained by the MOD and the K-SVD operate on signals by considering only their native scale. Past experience with the wavelet transform teaches us that often times it is beneficial to process the signal in several scales, and operate on each scale differently. This shortcoming is related to the above mentioned limits on the dimensionality of the signals involved (which does not leave much freedom for multi-scale processing) and the speed issue (multi-scale dictionaries are potentially faster to multiply with).
- **Lack of Invariances:** In some applications we desire the dictionary we compose to have specific invariance properties. The most classical examples are shift-, rotation-, and scale-invariances. These imply that when the dictionary is used on a shifted/rotated/scaled version of an image, we expect the sparse representation obtained to be tightly related to the representation of the original image. Injecting these invariance properties to dictionary-learning is therefore valuable, and the above methodology has not addressed this matter.

All the above shortcomings reflect properties that exist in pre-designed dictionaries, and it would be advantageous to transfer these to learned dictionaries as well, thus creating a more appealing alternative to the classical transforms. In this respect, the above-presented dictionary-learning methodology constitutes only the first step in replacing classical transforms.

In this section we present three initial branches of activity on dictionary-learning that aim to solve some of the above problems. More work is required in order to bring the learning methodology to a mature level, such that it competes favorably with pre-chosen or adapted alternatives, on all aspects – quality of results, speed, and elegance.

---

<sup>3</sup> Operating on an image patch of size  $\sqrt{n} \times \sqrt{n}$ , the multiplication by the dictionary is performed by multiplying this patch from the left by a matrix of size  $\sqrt{m} \times \sqrt{n}$ , and a similar multiplication from the right with a matrix of size  $\sqrt{n} \times \sqrt{m}$ .

### 12.3.1 The Double-Sparsity Model

The double-sparsity method described in this section aims to address the limit on the input signal dimension, along-side with the speed of deployment of the obtained dictionary. We return to the dictionary-learning formulation as posed in Equation (12.2),

$$\min_{\mathbf{A}, \{\mathbf{x}_i\}_{i=1}^M} \sum_{i=1}^M \|\mathbf{y}_i - \mathbf{A}\mathbf{x}_i\|_2^2 \quad \text{subject to} \quad \|\mathbf{x}_i\|_0 \leq k_1, \quad 1 \leq i \leq M,$$

where every signal is known to be constructed from  $k_1$  atoms (the change from  $k_0$  in the original notation to  $k_1$  will become clear shortly). We shall assume that each of the  $m$  atoms in  $\mathbf{A} \in \mathbb{R}^{n \times m}$  is exhibiting a very specific inner structure, which will help in getting faster dictionaries to multiply with.

The structure we choose assumes that each atom in  $\mathbf{A}$  can be described as a sparse combination of  $k_0$  pre-atoms extracted from a pre-specified dictionary  $\mathbf{A}_0 \in \mathbb{R}^{n \times m_0}$ . Thus, our assumption can be formulated as  $\mathbf{A} = \mathbf{A}_0 \mathbf{Z}$ , where  $\mathbf{Z} \in \mathbb{R}^{m_0 \times m}$  is a sparse matrix with  $k_0$  non-zeros per column. This structure is motivated by the appearance of the typical dictionaries for image-patches, showing piecewise smooth, periodic, or textured content. The key for the worthiness of this method is the choice of  $\mathbf{A}_0$ . On one hand, it should be chosen such that the assumption made above is nearly correct. On the other hand, it should be a matrix that has a fast multiplication algorithm, so that operating with  $\mathbf{A}$  and its adjoint are cheap.

Putting this structure into the dictionary-learning optimization task, as posed in (12.2), we obtain

$$\begin{aligned} \min_{\{\mathbf{x}_i\}_{i=1}^M, \{\mathbf{z}_j\}_{j=1}^m} \quad & \sum_{i=1}^M \|\mathbf{y}_i - \mathbf{A}_0 \mathbf{Z} \mathbf{x}_i\|_2^2 \\ \text{subject to} \quad & \begin{cases} \|\mathbf{z}_j\|_0 \leq k_0, & 1 \leq j \leq m \\ \|\mathbf{x}_i\|_0 \leq k_1, & 1 \leq i \leq M \end{cases} \end{aligned} \quad (12.6)$$

Minimizing this problem can be done following the above described algorithms that iterate between an update of  $\{\mathbf{x}_i\}_{i=1}^M$  by sparse coding, and an update of  $\{\mathbf{z}_j\}_{j=1}^m$  for the effective dictionary update.

Minimization with respect to the sparse representations  $\mathbf{x}_i$ , while assuming that  $\mathbf{Z}$  is fixed, can be done as usual, as a sparse coding stage that employs OMP. This stage is cheaper than the regular MOD/K-SVD sparse-coding stage, because the multiplication of any vector by  $\mathbf{A}$  (or its adjoint) is expected to be much cheaper than  $O(nm)$ . For example, a unitary and separable 2D-DCT applied on an image patch of size  $\sqrt{n} \times \sqrt{n}$  requires  $O(n \log n)$ . A redundant and separable 2D-DCT does not have a fast  $n \log n$  algorithm, but the separability by itself is an asset, leading to  $O(n^{1.5})$  operations, instead of the regular  $O(n^2)$ .

Fixing  $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^M$ , the update of  $\{\mathbf{z}_j\}_{j=1}^m$  may seem more demanding, but in-fact amounts to a very similar sparse coding stage. Considering one column  $\mathbf{z}_j$  at a time,

we are imitating the K-SVD algorithm in the sense that one atom  $\mathbf{A}_0 \mathbf{z}_j$  from the overall dictionary is updated. Just as with the K-SVD, we consider hereafter only those examples that use the  $j$ -th atom in their construction. Therefore, the expression  $\mathbf{Y} - \mathbf{AZX}$  contains now a subset of the complete  $M$  columns. We denote the  $j$ -th row (and not column!!) from  $\mathbf{X}$  by  $\tilde{\mathbf{x}}_j^T$ . The expression  $\mathbf{ZX}$  can be written as the sum of  $m_0$  outer products of the form  $\mathbf{z}_j \tilde{\mathbf{x}}_j^T$ . Thus, the  $\ell_2$  term in Equation (12.6) becomes (recall that the summation over  $i$  considers only relevant examples that use the  $j$ -th atom)

$$\sum_i \|\mathbf{y}_i - \mathbf{A}_0 \mathbf{Z} \mathbf{x}_i\|_2^2 = \left\| \mathbf{Y} - \mathbf{A}_0 \sum_{j=1}^m \mathbf{z}_j \tilde{\mathbf{x}}_j^T \right\|_F^2. \quad (12.7)$$

Thus, minimization of Equation (12.6) with respect to  $\mathbf{z}_j$  leads to the optimization problem

$$\min_{\mathbf{z}_j} \|\mathbf{E}_j - \mathbf{A}_0 \mathbf{z}_j \tilde{\mathbf{x}}_j^T\|_F^2 \quad \text{subject to } \|\mathbf{z}_j\|_0 \leq k_0. \quad (12.8)$$

where we have defined  $\mathbf{E}_j = \mathbf{Y} - \mathbf{A}_0 \sum_{k=1, k \neq j}^m \mathbf{z}_k \tilde{\mathbf{x}}_k^T$ . Note that the row  $\tilde{\mathbf{x}}_j^T$  contains all the coefficients of the  $j$ -th atom, and since we have restricted the error to examples that use this atom, this vector is dense.

The obtained optimization is treated by an alternating minimization over  $\tilde{\mathbf{x}}_j^T$  and  $\mathbf{z}_j$ . Minimization of this problem with respect to  $\tilde{\mathbf{x}}_j^T$  is a simple Least-Squares step that gives

$$\mathbf{z}_j^T \mathbf{A}_0^T (\mathbf{E}_j - \mathbf{A}_0 \mathbf{z}_j \tilde{\mathbf{x}}_j^T) = 0 \quad \Rightarrow \quad \tilde{\mathbf{x}}_j^T = \frac{1}{\mathbf{z}_j^T \mathbf{A}_0^T \mathbf{A}_0 \mathbf{z}_j} \mathbf{z}_j^T \mathbf{A}_0^T \mathbf{E}_j. \quad (12.9)$$

The denominator stands for the norm of the  $j$ -th atom, and we can assume it is normalized, by a proper normalization of  $\mathbf{z}_j$ .

Minimization with respect to the sparse vector  $\mathbf{z}_j$  can be shown<sup>4</sup> to be equivalent to the solution of

$$\min_{\mathbf{z}_j} \|\mathbf{E}_j \tilde{\mathbf{x}}_j - \mathbf{A} \mathbf{z}_j\|_2^2 \quad \text{subject to } \|\mathbf{z}_j\|_0 \leq k_0, \quad (12.10)$$

and this is a classic sparse coding operation that can be handled by OMP. Several such rounds of updates as described above lead to the updated atom  $\mathbf{z}_j$ , and also  $\tilde{\mathbf{x}}_j$ , which gives is the updated representation coefficients that use this atom.

Beyond the gained speed, the introduction of structure to the dictionary is beneficial in two other ways: (i) Since the number of degrees of freedom in the dictionary is smaller, fewer examples are sufficient for an effective learning, and faster convergence of the learning algorithm is obtained; and (ii) Higher-dimensional signals can

<sup>4</sup> It is easily verified that  $\|\mathbf{E}_j - \mathbf{A}_0 \mathbf{z}_j \tilde{\mathbf{x}}_j^T\|_F^2 = \|\mathbf{E}_j \tilde{\mathbf{x}}_j - \mathbf{A}_0 \mathbf{z}_j\|_2^2 + f(\mathbf{E}_j, \tilde{\mathbf{x}}_j)$ .

be treated. More on these benefits and demonstration of this approach are brought in the work by Rubinstein et al.

### 12.3.2 Union of Unitary Bases

Another approach for injecting structure into the learned dictionary is to force the dictionary to be a union of unitary matrices. This structure guarantees that the obtained dictionary is a tight Frame, for which the adjoint and the pseudo-inverse are the same. We now turn to describe this approach in more details.

Starting again with the dictionary-learning formulation as posed in Equation (12.2),

$$\min_{\mathbf{A}, \{\mathbf{x}_i\}_{i=1}^M} \sum_{i=1}^M \|\mathbf{y}_i - \mathbf{A}\mathbf{x}_i\|_2^2 \quad \text{subject to } \|\mathbf{x}_i\|_0 \leq k_0, \quad 1 \leq i \leq M,$$

assume now that  $\mathbf{A}$  is chosen to be a union of unitary matrices, as proposed by Lesage et al. One good reason for this structure is the Block-Coordinate-Relaxation (BCR) algorithm that was mentioned in Chapter 6 for the sparse coding stage. This algorithm requires multiplications by unitary matrices and simple shrinkage steps, and is thus very efficient. Other reasons for this structure could be the ease of the training (as we show next), and the natural tightness of the Frame  $\mathbf{A}$ . We shall assume, for simplicity, that only two unitary matrices are involved in the dictionary, i.e.,  $\mathbf{A} = [\Psi, \Phi] \in \mathbb{R}^{n \times 2n}$ .

Focusing on the dictionary update stage, and assuming that the sparse representations are fixed, our goal is to solve

$$\min_{\Psi, \Phi} \|\Psi\mathbf{X}_\Psi + \Phi\mathbf{X}_\Phi - \mathbf{Y}\|_F^2 \quad \text{subject to } \Psi^T\Psi = \Phi^T\Phi = \mathbf{I}. \quad (12.11)$$

Here we use the decomposition  $\mathbf{A}\mathbf{X} = \Psi\mathbf{X}_\Psi + \Phi\mathbf{X}_\Phi$ . Fixing  $\Psi$ , optimizing the above with respect to  $\Phi$  leads the well-known orthogonal Procrustes problem. This problem admits a closed-form solution, which we derive next. Thus, we can iterate between an update of  $\Phi$  and  $\Psi$  several times, and this way perform the dictionary update stage.

As for the Procrustes problem, it is formulated as a plain Least-Squares problem of the form  $\|\mathbf{A} - \mathbf{QB}\|_F^2$ , where  $\mathbf{A}, \mathbf{B}$  are arbitrary pair of matrices of the same size, and the unknown  $\mathbf{Q}$  is a square and orthogonal matrix that should rotate the columns of  $\mathbf{B}$  towards those of  $\mathbf{A}$ , such that the squared-error is minimized. First we observe that

$$\|\mathbf{A} - \mathbf{QB}\|_F^2 = \text{tr}\{\mathbf{A}^T\mathbf{A}\} + \text{tr}\{\mathbf{B}^T\mathbf{B}\} - 2\text{tr}\{\mathbf{QBA}^T\}. \quad (12.12)$$

Thus, our minimization task is equivalent to the maximization of  $\text{tr}\{\mathbf{QBA}^T\}$ . Assuming that the SVD of the matrix  $\mathbf{BA}^T$  is available to us,  $\mathbf{BA}^T = \mathbf{U}\Sigma\mathbf{V}^T$ , our penalty

can be written as

$$\text{tr}\{\mathbf{QBA}^T\} = \text{tr}\{\mathbf{QU}\mathbf{\Sigma}\mathbf{V}^T\} = \text{tr}\{\mathbf{V}^T\mathbf{QU}\mathbf{\Sigma}\} = \text{tr}\{\mathbf{Z}\mathbf{\Sigma}\}. \quad (12.13)$$

Here we have used the property  $\text{tr}\{\mathbf{AB}\} = \text{tr}\{\mathbf{BA}\}$  (for an arbitrary square matrix pair), and the notation  $\mathbf{Z} = \mathbf{V}^T\mathbf{QU}$ .  $\mathbf{Z}$  is a unitary matrix as well, while  $\mathbf{\Sigma}$  is a diagonal matrix with the singular-values  $\{\sigma_i\}_{i=1}^n$  on the main-diagonal. Thus,

$$\text{tr}\{\mathbf{Z}\mathbf{\Sigma}\} = \sum_{i=1}^n \sigma_i z_{ii} \leq \sum_{i=1}^n \sigma_i. \quad (12.14)$$

The last inequality stems from the fact that entries in a unitary matrix are in the range  $[-1, 1]$ . Fulfilling the upper-bound is obtained for the choice  $\mathbf{Z} = \mathbf{I}$ , which in turn implies that  $\mathbf{Q} = \mathbf{V}\mathbf{U}^T$  — this is the closed-form solution mentioned above.

### 12.3.3 The Signature Dictionary

An interesting structure that leads to a shift-invariance property is the Image Signature Dictionary (ISD), as proposed by Aharon et al. We shall present this method for a simplified 1D case, so as to reduce the notational burden.

The dictionary in mind is  $\mathbf{A} \in \mathbb{R}^{n \times m}$ , as represented throughout this chapter. However, this dictionary is constructed from a *single* signal  $\mathbf{a}_0 \in \mathbb{R}^{m \times 1}$ , by extracting all the possible patches of length  $n$  (including cyclic shifts). We refer to  $\mathbf{a}_0$  as the signature signal that defines the effective dictionary. Note that in the 1D case, this dictionary has a circulant structure. While the dictionary have  $mn$  entries,  $m$  free parameters alone dictate its entire content.

Denoting by  $\mathbf{R}_k$  the operator that extracts a patch of length  $n$  from location  $k$  from  $\mathbf{a}_0$ , the  $k$ -th atom in  $\mathbf{A}$  is given by  $\mathbf{a}_k = \mathbf{R}_k \mathbf{a}_0$ . The representation of a signal  $\mathbf{y}$  as a linear combination of atoms from this dictionary is given by

$$\mathbf{y} = \sum_{k=1}^m x_k \mathbf{a}_k = \sum_{k=1}^m x_k \mathbf{R}_k \mathbf{a}_0. \quad (12.15)$$

Armed with this structure, we now return to the dictionary-learning goal, as posed in the problem

$$\min_{\mathbf{A}, \{\mathbf{x}_i\}_{i=1}^M} \sum_{i=1}^M \|\mathbf{y}_i - \mathbf{A}\mathbf{x}_i\|_2^2 \quad \text{subject to} \quad \|\mathbf{x}_i\|_0 \leq k_0, \quad 1 \leq i \leq M. \quad (12.16)$$

Adopting the same block-coordinate descent algorithm, the sparse coding stage that updates  $\mathbf{x}_i$  remains unchanged, since  $\mathbf{A}$  can be formed as usual. Indeed, we could numerically benefit from its structure by using the fast-Fourier-Transform (FFT) for a fast convolution that replaces the exhaustive inner-products.



The dictionary update stage differs, so as to take into account the inner structure of  $\mathbf{A}$ . Using the formulation in Equation (12.15) to replace the term  $\mathbf{A}\mathbf{x}_i$  in Equation (12.16), we get

$$\sum_{i=1}^M \|\mathbf{y}_i - \mathbf{A}\mathbf{x}_i\|_2^2 = \sum_{i=1}^M \left\| \mathbf{y}_i - \sum_{k=1}^m x_i[k] \mathbf{R}_k \mathbf{a}_0 \right\|_2^2. \quad (12.17)$$

Thus, an update to the dictionary is obtained by optimizing the above  $\ell_2$  expression with respect to  $\mathbf{a}_0$ . We obtain the equation

$$0 = \sum_{i=1}^M \left( \sum_{k=1}^m x_i[k] \mathbf{R}_k \right)^T \left( \mathbf{y}_i - \sum_{k=1}^m x_i[k] \mathbf{R}_k \mathbf{a}_0 \right), \quad (12.18)$$

which leads to

$$\mathbf{a}_0^{opt} = \left( \sum_{k=1}^m \sum_{j=1}^m \left[ \sum_{i=1}^M x_i[k] x_i[j] \right] \mathbf{R}_k^T \mathbf{R}_j \right)^{-1} \sum_{i=1}^M \sum_{k=1}^m x_i[k] \mathbf{R}_k^T \mathbf{y}_i. \quad (12.19)$$

Note that the term  $\rho(k, j) = \sum_{i=1}^M x_i[k] x_i[j]$  in the above equation stands for the cross-correlation between the entries in the representation vectors. The operation  $\mathbf{v} = \mathbf{R}_k^T \mathbf{u}$  creates a zero vector  $\mathbf{v}$ , and then puts  $\mathbf{u}$  (of length  $n$ ) in its  $k$ -th location. We do not provide here further analysis of this formula and its implications, beyond the obvious statement that this can be used for an easy update of  $\mathbf{a}_0$ . More details can be found in the recent work by Aharon and Elad, where this structure has been used in 2D for images. Benefits attributed to this structure are the following:

- Since the number of degrees of freedom is much smaller than  $nm$ , this implies that less training data is required, and faster convergence of the learning procedure can be obtained.
- The better-behaved penalty function for the learning procedure implies a smaller tendency towards local minimum.
- As the dictionary emerges from a single signal (or image) with plain shifts, working on two shifted signals can be made more efficient by applying a sparse coding on the first, and simply shifting these atoms to approximate the second. This idea was used successfully in the work by Aharon and Elad to reduce the complexity of sparse coding for image patches.
- This structure is the first in which atoms of varying sizes are easily accommodated.

Because of these reasons and more, the ISD seems to be a promising concept, and more work is required to expose and exploit its potential.



## 12.4 Summary

With a proper dictionary, the *Sparse-Land* model can lead to better performance in the various applications it is harnessed to. Dictionary-learning seems to be a promising direction for obtaining a favorable model. In this chapter we presented several algorithms and ideas that have already appeared in the literature. Nevertheless, it is clear that far more should be done in order to establish this approach as truly competitive, stable, and reliable.

Still on the dictionary-learning problem, a different front that requires more work refers to the desire to extend the *Sparse-Land* model. Extensions to consider are dependencies between the atoms, varying probabilities for the atoms to be used, and perhaps the introduction of non-linearities. In all these cases, the dictionary-learning algorithms must be updated to cope with these changes.

As a last point in this chapter we mention that among the various things to be desired for a good dictionary, a truly multi-scale structure is of an utmost importance. Learning of such structures has not been treated thoroughly so far, as it poses numerous difficulties. When such a structure will be properly introduced, it might open new horizons for the vast activity on wavelet and X-let transforms, and lead to a convergence of these two branches (learning dictionaries, and designing multi-scale frames).

## Further Reading

1. M. Aharon, M. Elad, and A.M. Bruckstein, K-SVD and its non-negative variant for dictionary design, *Proceedings of the SPIE conference wavelets*, Vol. 5914, July 2005.
2. M. Aharon, M. Elad, and A.M. Bruckstein, On the uniqueness of overcomplete dictionaries, and a practical way to retrieve them, *Journal of Linear Algebra and Applications*, 416(1):48–67, July 2006.
3. M. Aharon, M. Elad, and A.M. Bruckstein. K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation, *IEEE Trans. on Signal Processing*, 54(11):4311–4322, November 2006.
4. E.J. Candès and D.L. Donoho, Recovering edges in ill-posed inverse problems: optimality of curvelet frames, *Annals of Statistics*, 30(3):784–842, 2000.
5. E.J. Candès and D.L. Donoho, New tight frames of curvelets and optimal representations of objects with piecewise- $C^2$  singularities, *Comm. Pure Appl. Math.*, 57:219–266, 2002.
6. A. Cichocki, R. Zdunek, A.H. Phan, and S.I. Amari, *Nonnegative Matrix and Tensor Factorizations Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*, Wiley, Tokyo, Japan, 2009.
7. V. Chandrasekaran, M. Wakin, D. Baron, R. Baraniuk, Surflets: a sparse representation for multidimensional functions containing smooth discontinuities, *IEEE Symposium on Information Theory*, Chicago, IL, 2004.

8. R.R. Coifman and M.V. Wickerhauser, Adapted waveform analysis as a tool for modeling, feature extraction, and denoising, *Optical Engineering*, 33(7):2170–2174, July 1994.
9. M.N. Do and M. Vetterli, Rotation invariant texture characterization and retrieval using steerable wavelet-domain hidden Markov models, *IEEE Trans. On Multimedia*, 4(4):517–527, December 2002.
10. M.N. Do and M. Vetterli, The finite ridgelet transform for image representation, *IEEE Trans. On Image Processing*, 12(1):16–28, 2003.
11. M.N. Do and M. Vetterli, Framing pyramids, *IEEE Trans. On Signal Processing*, 51(9):2329–2342, 2003.
12. M.N. Do and M. Vetterli, The contourlet transform: an efficient directional multiresolution image representation, *IEEE Trans. Image on Image Processing*, 14(12):2091–2106, 2005.
13. D.L. Donoho and V. Stodden, When does non-negative matrix factorization give a correct decomposition into parts? *Advances in Neural Information Processing* 16 (Proc. NIPS 2003), MIT Press, 2004.
14. M. Elad and M. Aharon, Image denoising via learned dictionaries and sparse representation, *International Conference on Computer Vision and pattern Recognition*, New-York, June 17–22, 2006.
15. M. Elad and M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries, *IEEE Trans. on Image Processing* 15(12):3736–3745, December 2006.
16. K. Engan, S.O. Aase, and J.H. Husoy, Multi-frame compression: Theory and design, *EURASIP Signal Processing*, 80(10):2121–2140, 2000.
17. R. Eslami and H. Radha, The contourlet transform for image de-noising using cycle spinning, in *Proceedings of Asilomar Conference on Signals, Systems, and Computers*, pp. 1982–1986, November 2003.
18. R. Eslami and H. Radha, Translation-invariant contourlet transform and its application to image denoising, *IEEE Trans. on Image Processing*, 15(11):3362–3374, November 2006.
19. A. Gersho and R.M. Gray, *Vector Quantization And Signal Compression*, Kluwer Academic Publishers, Dordrecht, Netherlands, 1992.
20. G.H. Golub and C.F. Van Loan, *Matrix Computations*, Johns Hopkins Studies in Mathematical Sciences, Third edition, 1996.
21. P.O. Hoyer, Non-negative matrix factorization with sparseness constraints, *Journal of Machine Learning Research*, 1457–1469, 2004.
22. K. Kreutz-Delgado, J.F. Murray, B.D. Rao, K. Engan, T-W. Lee, and T.J. Sejnowski, Dictionary learning algorithms for sparse representation, *Neural Computation*, 15(2)349–396, 2003.
23. D. Lee and H. Seung, Learning the parts of objects by non-negative matrix factorization, *Nature*, 788–791, 1999.
24. S. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya, Learning unions of orthonormal bases with thresholded singular value decomposition, *ICASSP 2005 (IEEE Conf. on Acoustics, Speech and Signal Proc.)*.

25. M.S. Lewicki and B.A. Olshausen, A probabilistic framework for the adaptation and comparison of image codes, *Journal of the Optical Society of America A: Optics, Image Science and Vision*, 16(7):1587–1601, 1999.
26. M.S. Lewicki and T.J. Sejnowski, Learning overcomplete representations, *Neural Computation*, 12:337–365, 2000.
27. Y. Li, A. Cichocki, and S.-i. Amari, Analysis of sparse representation and blind source separation, *Neural Computation*, 16(6):1193–1234, 2004.
28. F.G. Meyer, A. Averbuch, and J.O. Stromberg, Fast adaptive wavelet packet image compression, *IEEE Trans. on Image Processing*, 9(5):792–800, 2000.
29. F.G. Meyer and R.R. Coifman, Brushlets: a tool for directional image analysis and image compression, *Applied and Computational Harmonic Analysis*, 4:147–187, 1997.
30. B.A. Olshausen and D.J. Field, Natural image statistics and efficient coding, *Network – Computation in Neural Systems*, 7(2):333–339, 1996.
31. B.A. Olshausen and B.J. Field, Emergence of simple-cell receptive field properties by learning a sparse code for natural images *Nature*, 381(6583):607–609, 1996.
32. B.A. Olshausen and B.J. Field, Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.
33. E.P. Simoncelli, W.T. Freeman, E.H. Adelson, and D.J. Heeger, Shiftable multi-scale transforms, *IEEE Trans. on Information Theory*, 38(2):587–607, 1992.
34. J.-L. Starck, E.J. Candès, and D.L. Donoho, The curvelet transform for image denoising, *IEEE Trans. on Image Processing*, 11:670–684, 2002.

## Chapter 13

# Image Compression – Facial Images

The sparse-representation viewpoint discussed so far, along with dictionary-learning, is merely that – a viewpoint. The theoretical results we have given merely tell us that sparse modeling is, in favorable cases, a mathematically well-founded enterprise with practically useful computational tools. The only way to tell if sparse modeling works in the real world is to apply it, and see how it performs! We have already seen the use of sparse representation modeling for image deblurring, and the results seem to be promising. In this and the next chapters, we review selected results applying this viewpoint to several image processing tasks.

In this chapter we consider the problem of facial image compression. The content of this chapter follows up closely with the work by Bryt and Elad. Compression using sparse representation modeling seems as a very natural approach. Therefore, it is tempting to develop an image/video compression scheme that relies on this model. Nevertheless, developing such a compression scheme that competes favorably with JPEG2000 (for still images) and H-264 (for video) is very hard, if not impossible. The reason is that in these methods, the entropy-coding stage has been perfected. Thus, it is not sufficient to change the elementary steps of the coding algorithm – one must provide an appealing entropy-coding as well.

Because of this, we choose to address a very specific coding task, where the model by itself could make the desired difference. This brings us to the handling of facial images, which is the topic of this chapter.

## 13.1 Compression of Facial Images

Compression of still images is a very active and matured field of research, vivid in both research and engineering communities. Compression of images is possible because of their vast spatial redundancy and the ability to absorb moderate errors in the reconstructed image. This field of work offers many contributions, some of which became standard algorithms that are wide-spread and popular. Among the many methods for image compression, one of the best is the JPEG2000 standard

– a general purpose wavelet-based image compression algorithm with very good compression performance.

At its core, JPEG2000 relies on the fact that images are expected to be sparse when represented using wavelets. In this sense, JPEG2000 is already a supporting testimony for the efficiency and relevancy of *Sparse-Land* as a model for images. Nevertheless, and as said above, part of the success of the JPEG2000 is attributed to the well-designed entropy-coding within this compression algorithm. Our goal is to address a compression task, where the entropy-coding becomes secondary in its importance.

When considering the compression of a specific and narrow image class, the amount of redundancy increases, thus enabling a better compression performance. Such is the case with compression of facial images. Indeed, this expected gain has been observed and exploited in several recent publications that offer tailored compression algorithms for facial images. Among those contributions, the more recent ones show performance surpassing those of the JPEG2000.

Compression of facial images is an appealing problem, both because of the research challenges it provides, and the important applications it serves. From a research perspective, one should primarily wonder how to exploit the additional redundancy that such a focused image class exhibits, and how to surpass general purpose compression algorithms this way. This is not an easy challenge due to the vast efforts put to the general purpose algorithms, and especially their entropy-coding parts.

Application-wise, facial images are perhaps the most popular images, held in large databases by various organizations, such as police and law-enforcement, schools and universities, states, and in databases of employees in large companies. Efficient storage of such images is of value, and especially so when considering photo-ID in electronic ID cards. In such applications, very low bit-rate compression is to be considered, where general purpose algorithms fail utterly.

Motivated by the above, this chapter addresses the task of compression of facial images, as described above. We target photo-ID images of pre-specified and fixed size  $179 \times 221$  grayscale images with 8 bits per pixel.<sup>1</sup> The goal is very low bit rates (roughly speaking, we consider compression ratios  $\approx 100$ ), where most algorithms are unable to show recognizable faces. We use a database containing nearly 4,500 such facial images, some of which are used for training and tuning the algorithm, and the others for testing it.

As already mentioned above, the compression algorithm described in this chapter is taken from the work by Bryt and Elad. This algorithm relies strongly on recent advancements made in using sparse and redundant representation of signals, and learning their sparsifying dictionaries. The K-SVD algorithm is used for learning dictionaries for representing small image patches in a locally adaptive way, and using the obtained dictionaries to sparse-code the patches' content. This is a relatively

---

<sup>1</sup> This database contains color photos taken by a 4 Mega-pixel digital camera (Fuji FinePix A400), against a white and uniform background, with the highest compression settings for best quality. These photos were cropped and scaled to the above-mentioned size, and also converted to a gray-value format.

simple and straight-forward algorithm with hardly any entropy-coding stage. Yet, it is shown to be superior to several competing algorithms: (i) the JPEG and the JPEG2000, (ii) the VQ-based algorithm presented by Goldenberg et. al., and (iii) A Principal-Component-Analysis (PCA) approach.<sup>2</sup>

## 13.2 Previous Work

Among the thousands of papers that study still image compression algorithms, there are relatively few that consider the treatment of facial images. Among those, the most recent and the best performing algorithm is the one by Goldenberg et. al. That work also provides a thorough literature survey that compares the various methods and discusses similarities and differences between them. We concentrate here on the description of this specific algorithm, as it serves as the foundation for the sparse-representation-based method described in this chapter.

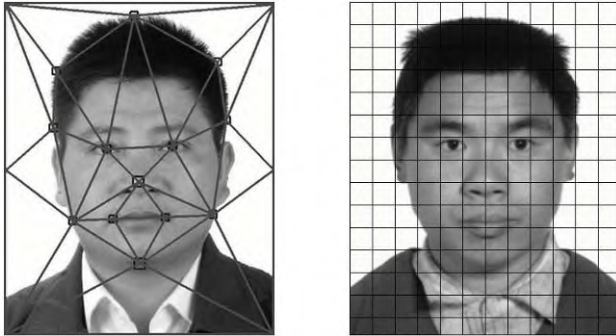
The algorithm by Goldenberg et. al., like some others before it, starts with a geometrical alignment of the input image, so that the main facial features (ears, nose, mouth, hair-line, etc.) are aligned with those of a database of pre-aligned facial images. Such alignment further increases the redundancy in the handled image, due to its high cross similarity to the database. The warping is done by an automatic detection of 13 feature points on the face, and moving them to pre-determined canonical locations. These points define a slicing of the input image into disjoint and covering set of triangles, each exhibiting an affine warp, being a function of the motion of its three vertices. Side information on these 13 feature locations enables a reverse warp of the reconstructed image in the decoder. [Figure 13.1](#) (left side) shows the located features and the induced triangles. After the warping, the image is sliced into square and non-overlapping uniform patches (of size  $8 \times 8$  pixels), each of which is coded separately. Such possible slicing (for illustration purposes we show this slicing with larger patches) is shown in [Figure 13.1](#) right side.

Coding of the image patches is done using Vector-Quantization (VQ). The VQ dictionaries are trained using tree-K-Means per each patch separately, using patches taken from the same location extracted from 4,500 training images. This way, each VQ is adapted to the expected local content, and thus the high performance presented by this algorithm. The number of code-words in the VQ is a function of the bit-allocation for the patches. As we argue in the next section, while VQ enjoys a well-founded theoretical support as a near-ideal coder, VQ coding is limited by the available number of examples and the desired rate, forcing relatively small patch sizes. This, in turn, leads to a loss of some redundancy between adjacent patches, and thus a loss in further potential compression.

Another ingredient in this algorithm that partly compensates for the above-described shortcoming is a multi-scale coding scheme. The image is scaled down

---

<sup>2</sup> The PCA algorithm was developed by Bryt and Elad as a competitive benchmark, and while it is generally performing very well, it is inferior to the main algorithm presented in their work, which uses sparse and redundant representations.



**Fig. 13.1** Left: Piecewise affine warping of the image by triangulation. Right: A uniform slicing to disjoint square patches for coding purposes.

and VQ-coded using patches of size  $8 \times 8$ . Then it is interpolated back to the original resolution, and the residual is coded using VQ on  $8 \times 8$  pixels patches once again. This method can be applied on a Laplacian pyramid of the original (warped) image with several scales.

As already mentioned above, the results shown in Goldenberg's work surpass those obtained by JPEG2000, both visually and in Peak-Signal-to-Noise Ratio (PSNR) quantitative comparisons.<sup>3</sup> In this chapter we introduce an alternative algorithm that replaces the coding stage from VQ to sparse and redundant representations. This leads us to the next section, where we describe the principles behind this coding strategy.

### 13.3 Sparse-Representation-Based Coding Scheme

The algorithm we are about to describe below is based on sparse representation modeling of image patches and trained dictionaries, very much in line with the *Sparse-Land* model presented in Chapter 9, and learned dictionaries, as advocated in Chapter 12. In a way, we generalize the VQ point of view of the previous work and replace it with sparse representations, thereby leading to better results.

<sup>3</sup> PSNR is computed by  $10 \cdot \log_{10}(255^2/\text{MSE})$ , where the MSE is the Mean-Squared-Error per pixel.

### 13.3.1 The General Scheme

A block diagram of the encoder and decoder are given in [Figure 13.2](#). This algorithm consists of two main processes: An off-line K-SVD training process and an on-line image compression/decompression processes.

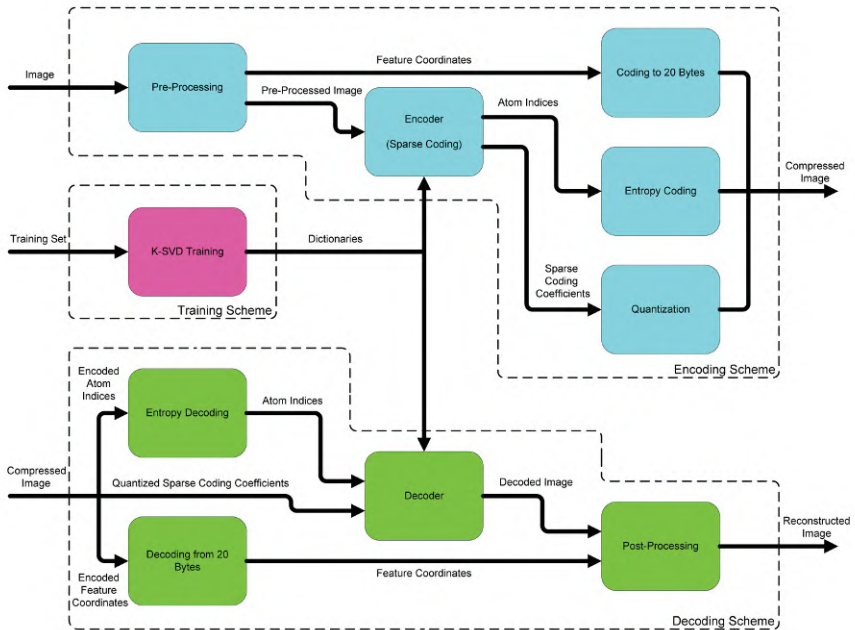
The K-SVD training process is an off-line procedure, preceding any image compression. The training produces a set of K-SVD dictionaries that are then considered fixed for the image compression-decompression stage. A single dictionary is trained for each  $15 \times 15$  patch over a set of examples that are the learning set. The training follows the description in Chapter 12, with parameters detailed below. Prior to the training process for each patch, The mean patch image of the examples in the learning set is calculated and subtracted from all the examples in this set. This is equivalent to the statement that the mean-facial image of the learning database is computed and subtracted from all images, serving as a rough prediction of the incoming images to be coded. The entire coding is done on the residual, and this mean image (stored both in the encoder and the decoder) is added after decompression.

The image compression process uses the following steps in the encoder, followed by a mirror application of them at the decoder:

- **Pre-Processing:** This is the same pre-process stage of geometrical warping as described above. This leads to better conditioning of the input image, increasing its redundancy versus the image database. The parameters of the warp are sent as side information for the inverse warp at the decoder. Those parameters are coded using 20 bytes.
- **Slicing to Patches:** Just as the VQ algorithm, the sparse-representation-based approach also works on fixed-size square patches, coding each patch separately and adaptively. However, the coding strategy is different, being based on sparse and redundant representations. The change in methods leads to the ability to handle larger patches, and thus get a more efficient coding. In the simulations reported here we use patches of size  $15 \times 15$  pixels<sup>4</sup> (we shall denote by  $n$  the length of the patch when organized as a vector, and thus  $\sqrt{n} = 15$  pixels), although larger patch sizes are also possible. Another benefit the sparse-representation-based algorithm has is the ability to use the very same dictionaries for various bit-rates. This is markedly different from the VQ solution that needs different trained code-books per each rate.
- **Removal of the Mean:** The patches must fit the dictionary in their content, and thus the same mean patch images that were calculated for the learning set before the training process are subtracted from the relevant patches in the incoming image.
- **Sparse-Coding:** Every patch location has a pre-trained dictionary of code-words (atoms)  $\mathbf{A}_{ij} \in \mathbb{R}^{n \times m}$ . The size we use for  $m$  is 512 – a discussion on this choice appears in the results section. These dictionaries are generated using the K-SVD algorithm on a set of 4,400 training examples. The coding itself is done

<sup>4</sup> Due to the inconsistency between the image and patch sizes, some of the patches at the right and bottom boundaries are of different sizes than the rest.





**Fig. 13.2** Detailed block diagram of the proposed encoding/decoding method.

by assigning a linear combination of a few atoms to describe the patch content (sparse-coding). Thus, information about the patch content includes both the linear weights and the chosen indices. Note that the number of atoms vary from one patch to another, based on its average complexity, and this varied number is known at the decoder.

The encoder and the decoder are storing the dictionaries. The sparse-coding stage for encoding of the patches is done using the OMP algorithm. The decoder is simply aggregating the proper atoms with the proper weights to reconstruct the patch, building the output one patch at a time independent of the others.

- Entropy-Coding and Quantization:** We use a straight-forward Huffman table applied only to the indices. The representations' weights are quantized with a 7 bits uniform quantization, with boundaries chosen based on the training information for each patch separately.

Since the above compression scheme strongly relies on a successful detection of the features to warp by, a natural question is whether errors in this stage are destructive to the overall compression algorithm. When the detection of the features fails utterly, the compression necessarily leads to very low performance. Since the compression is performed on an incoming image, we have access to both the resulting PSNR, and also the average PSNR we expect to get for this bit-allocation. Therefore, if the compression PSNR is below some threshold, we obviously know

that the detection failed (or possibly the input image is not a facial image). In such a case, and in a completely automatic fashion, we can employ the JPEG2000, which is indeed weaker, but still reasonable. A second option we have in such a system is to prompt the user to detect the features manually for this image. Thus, even rare situations (less than 1% in our experiments) where the features are not found do not lead to a breakdown in a system that employs the proposed algorithm.

### 13.3.2 VQ Versus Sparse Representations

As mentioned earlier, the destination compression ratio and the number of training examples pose a hard limit on the allowed patch size when using VQ. This is because as the patch size grows while keeping the rate (bits-per-pixel) fixed, the dictionary is required to grow exponentially, reaching sizes far beyond the number of examples to train on. More specifically, using patch size of  $\sqrt{n} \times \sqrt{n}$  pixels, with  $m$  code-words in the VQ dictionary, considering an image with  $P$  pixels, and a target rate of  $B$  bits for the entire image, all these ingredients are related to each other by the equation

$$B = \frac{P}{n} \cdot \log_2(m) \Rightarrow m = 2^{\frac{nB}{P}}. \quad (13.1)$$

For example, coding an image of size  $179 \times 221$  ( $P = 39,559$ ) pixels<sup>5</sup> with a target rate of  $B = 500 \times 8$  bits, a patch of size  $\sqrt{n} = 8$  leads to  $m = 88$ , which is reasonable. However, increasing the patch size to  $\sqrt{n} = 12$  already leads to  $m \approx 24,000$ , which cannot be trained from 4,500 examples. Thus, the VQ approach is limited to relatively small patch sizes ( $8 \times 8$  used by Goldenberg et. al.), implying that much of the spatial redundancy between adjacent patches is overlooked.

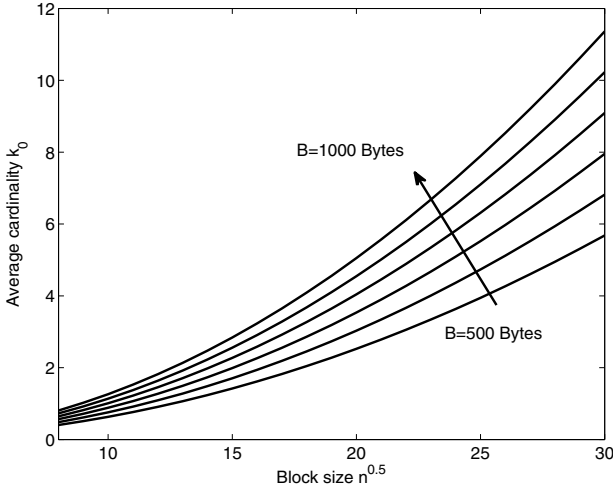
When turning to sparse and redundant dictionaries, the picture changes dramatically. We consider a patch size of  $\sqrt{n} \times \sqrt{n}$  pixels, with  $m$  atoms in the dictionary,  $k_0$  of which are used on average<sup>6</sup> in the representation (assuming 7 bits quantization of the weights). Handling of an image with  $P$  pixels, and a target rate of  $B$  bits for the entire image, all these ingredients lead to the relation

$$B = \frac{P}{n} \cdot k_0 \cdot (\log_2(m) + 7). \quad (13.2)$$

This means that per patch, instead of using  $\log_2 m$  bits when working with VQ, we need  $k_0(\log_2 m + 7)$  bits on average. Thus, if the required amount of bits per patch is too high (as indeed happens when  $n$  grows), it can be absorbed by varying  $k_0$ . For example, for an image with  $P = 39,600$  pixels, using  $m = 512$  atoms, a target rate

<sup>5</sup> This is the actual size used in our experiments.

<sup>6</sup> If  $k_0$  varies from one patch to another, side information is necessary in order to instruct the decoder how many atoms to use in each patch. However, in the proposed scheme, while  $k_0$  indeed varies spatially, it remains fixed for all images and thus the decoder knows the number of atoms allocated per patch location with no additional side information.



**Fig. 13.3** The Required average number of atoms  $k_0$  as a function of the patch size  $\sqrt{n}$ , with the influence of the overall allocated number of bits  $B$ .

of  $B = 500 \sim 1000$  bytes, and a patch sizes in the range  $\sqrt{n} = 8 \sim 30$ , all these lead to a required average number of atoms  $k_0$  as shown in Figure 13.3. As can be seen, the values are reasonable and so we have the flexibility to control the rate by changing  $k_0$  in the range  $1 \sim 12$ .

## 13.4 More Details and Results

The work by Bryt and Elad conducts several experiments in order to evaluate the performance of the proposed compression method and the quality of its resulting images. In this section we show some statistical results as well as reconstructed images from this compression method and a comparison between these results to several other competitive compression techniques.

Bryt and Elad use 4,400 facial images as the training set and a different 100 images as the test set. All the images in both sets are of a fixed size of  $179 \times 221$  pixels. The slicing to disjoint and covering set of patches is uniform over the image with a size of  $15 \times 15$  pixels per each. Examples of such pre-processed training/testing images can be seen in Figure 13.4.

Before turning to present the results we should add the following: While all the results shown here refer to the specific database we operate on, the overall scheme proposed is general and should apply to other facial image databases just as well. Naturally, some changes in the parameters might be necessary, and among these, the patch size is the most important to consider. We also note that as one shifts from one



**Fig. 13.4** Examples of pre-processed images used in the training/testing sets.

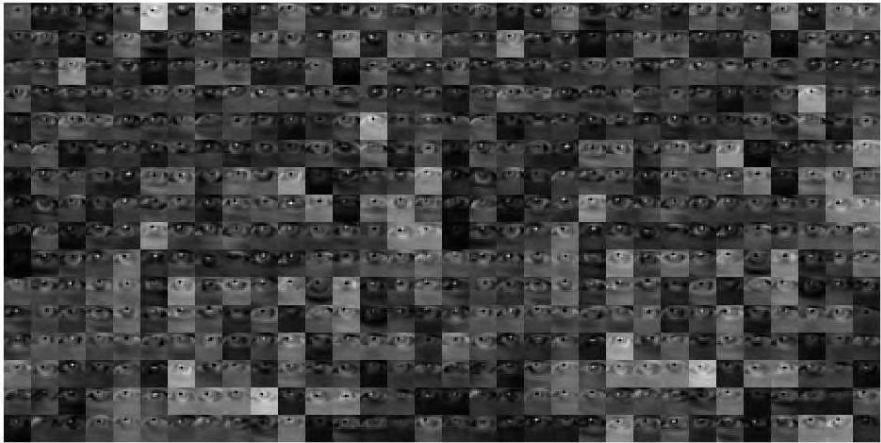
source of images to another, the relative size of the background in the photos may vary, and this necessarily leads to changes in performance. More specifically, when the background regions are larger (e.g., the images we use here have relatively small such regions), the compression performance is expected to improve.

### 13.4.1 *K-SVD Dictionaries*

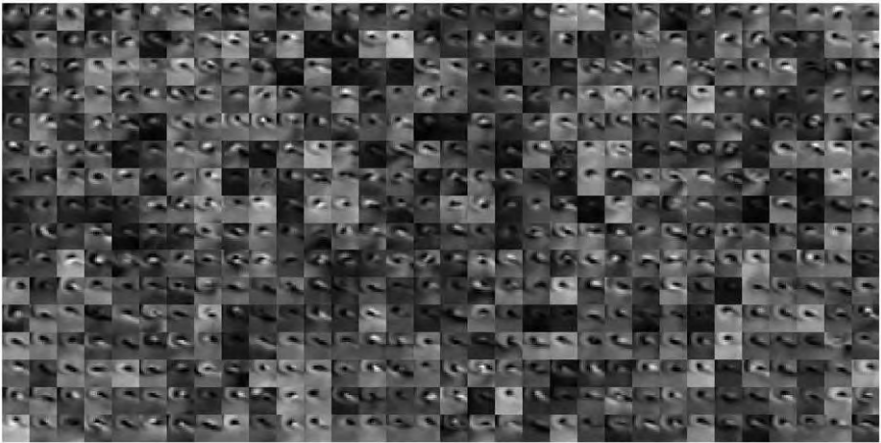
Using 100 iterations of the K-SVD algorithm, the required dictionaries are learned. Every obtained dictionary contains 512 patches of size  $15 \times 15$  pixels as atoms. In [Figure 13.5](#) we see the dictionary that was trained for patch number 80 (the left eye) for  $k_0 = 4$  sparse coding atoms, and similarly, in [Figure 13.6](#) we see the dictionary that was trained for patch number 87 (the right nostril) also for  $k_0 = 4$  sparse coding atoms. It can be seen that both dictionaries contain atoms similar in nature to the image patch for which they were trained for. A similar behavior is observed in other dictionaries.

### 13.4.2 *Reconstructed Images*

The proposed coding strategy allows us to learn which parts of the image are more difficult than others to code. This is done by assigning the same representation error threshold to all the patches, and observing how many atoms are required for the representation of each patch on average. Clearly, patches with a small number of allocated atoms are simpler to represent than others. We would expect that the representation of smooth areas of the image such as the background, parts of the face and maybe parts of the clothes will be simpler than the representation of areas containing detailed parts such as the hair or the eyes. [Figure 13.7](#) shows maps of the atom allocation per patch and representation error (Root Mean-Squared-Error – RMSE) per patch for the images in the test set at two different bit-rates. It can be seen that more atoms are allocated to patches containing the facial details (hair, mouth, eyes,



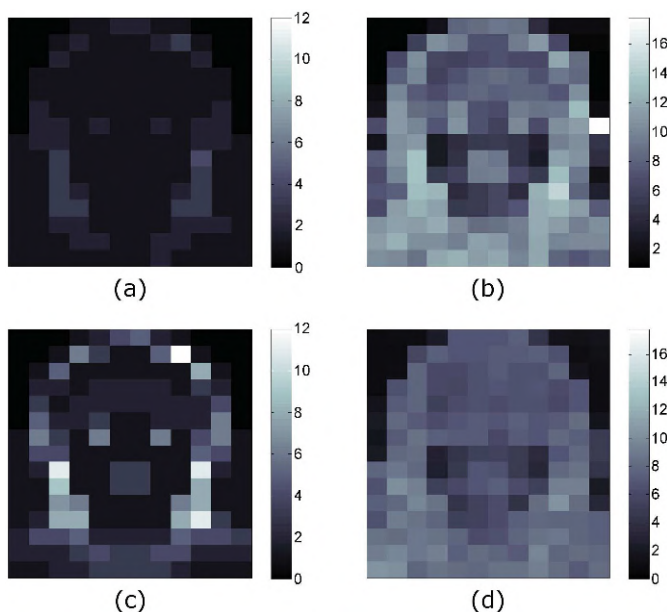
**Fig. 13.5** The Dictionary obtained by K-SVD for patch no. 80 (the left eye) using the OMP method with  $k_0 = 4$ .



**Fig. 13.6** The Dictionary obtained by K-SVD for patch no. 87 (the right nostril) using the OMP method with  $k_0 = 4$ .

facial borders), and that the representation error is higher in these patches. It can also be seen that the overall number of allocated atoms increases and the representation error decreases with the image bit-rate.

As in other compression methods, the reconstructed images suffer from several kinds of artifacts. These artifacts are caused by the fact that we operate on disjoint patches, coding them independently. Contrary to methods such as JPEG and JPEG2000, the reconstructed images at the presented method do not have a strong smearing artifact all over the image, but only local small areas in which the im-



**Fig. 13.7** For coding to 400 bytes, (a) shows the atom allocation map and (b) shows the representation error map. Similarly, (c) and (d) show the same maps for 820 bytes.

age is smeared. Other artifacts include inaccurate representation of high frequency elements in the image, inability to represent complicated textures (mainly in the clothes), and inconsistency in the spatial location of several facial elements comparing to the original image. These artifacts are caused by the nature of the algorithm developed, which has a limited ability to build the reconstructed image out of the given dictionaries.

In [Figure 13.8](#) we can see two original images from the training set and their reconstructed images in a bit-rate of 630 bytes. The mentioned artifacts can be seen in these images especially in the areas of the chin, the neck, the mouth, the clothes and the hairline. Although there are areas in the images that contain a smearing effect, the majority of the image is clear and sharp, and certainly having a high visual quality.

[Figure 13.9](#) shows three original images from the test set and their reconstructed images at a bit-rate of 630 bytes. As in the previous images, the mentioned artifacts can be seen in these images as well, in the same image areas as before. As expected, the quality of the images from the test set is not as high as the quality of images from the learning set, but these images too are clear and sharp, and have a high visual quality.

Much like other compression methods, the quality of the reconstructed images in the proposed method improves as the bit-rate increases. However, the contribution gained from such a rate increment is not divided equally over the image.



**Fig. 13.8** Examples of original (top) and reconstructed (bottom) images from the training set, with a relatively low error (4.04, on the left) and a relatively high error (5.9, on the right) representation Root-MSE, both at a bit rate of 630 bytes.

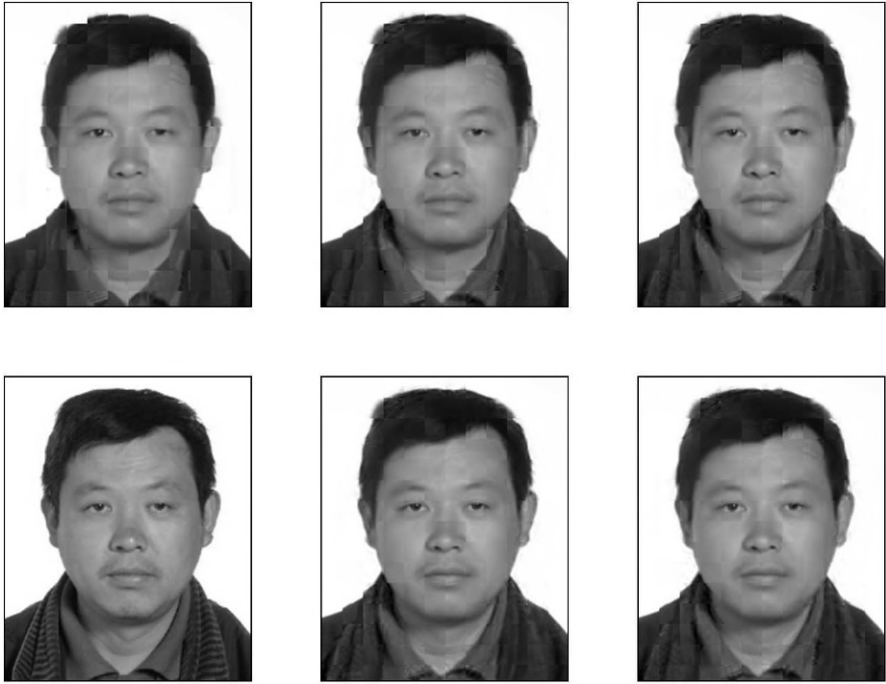




**Fig. 13.9** Examples of original and reconstructed images from the test set, with a relatively low error (5.3, left), an average error (6.84, middle) and a relatively high error (11.2, right) representation RMSE, all three at a bit-rate of 630 bytes.

Additional bits are allocated to patches with higher representation error, and those are improved first. This property is directly caused by the nature of the compression process, which is RMSE-oriented and not bit-rate-oriented. The compression process sets a single RMSE threshold for all the patches, forcing each of them to reach it without fixing the number of allocated atoms per patch. Patches with simple (smooth) content are most likely to have a representation error far below the threshold, even when using one atom or less, whereas patches with more complex content are expected to have a representation error very close to the threshold. Such problematic patches will be forced to improve their representation error by increasing the number of atoms they use as the RMSE threshold is decreased, while simple patches with a representation error below the threshold will not change their atom allocation. [Figure 13.10](#) illustrates the gradual improvement in the image quality as the bit-rate increases. As can be seen, not all the patches improve as the bit-rate increases but only some of them, such as several patches in the clothes area, in the ears and in the hairline. These patches are more difficult to represent than others.





**Fig. 13.10** Comparing the visual quality of a reconstructed image from the test set at several bit-rates. From top left, clockwise: 285 bytes (RMSE 10.6), 459 bytes (RMSE 8.27), 630 bytes (RMSE 7.61), 820 bytes (RMSE 7.11), 1021 bytes (RMSE 6.8), Original image.

### *13.4.3 Run-Time and Memory Usage*

In assessing the required memory for the proposed compression method we need to take into account all the dictionaries that were produced for each patch, the previously mentioned mean patch images, the Huffman tables, the coefficient usage tables and the quantization levels for each patch. All these lead to less than 20 Mbytes.

Practically, the training and testing processes have been all implemented using non-optimized Matlab code on a regular PC (Pentium 2, 2GHz, 1GByte RAM). Training of all the required dictionaries requires 100 hours to complete (and thus has been implemented on several computers in parallel, each handling a different group of patches). The compression of a single image requires 5 seconds, and its decompression takes less than 1 second.

### 13.4.4 Comparing to Other Techniques

An important part in assessing the performance of this compression method is its comparison to known and competitive compression techniques. As mentioned before, we compare the obtained results with JPEG, JPEG2000, the VQ-Based compression method described earlier, and a PCA-Based compression method that was built especially for this work as a competitive benchmark. We therefore start with a brief description of the PCA technique.

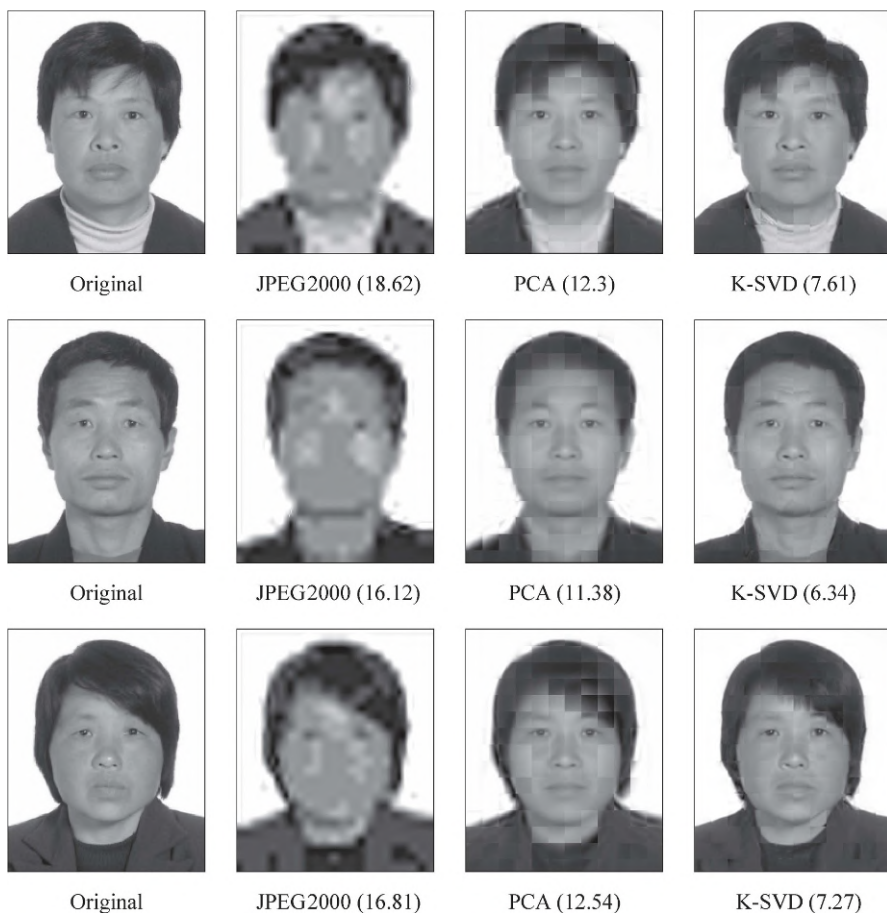
The PCA-Based compression method is very similar to the scheme described above, simply replacing the K-SVD dictionaries with a Principal Component Analysis (PCA) ones. These dictionaries are square matrices storing the eigenvectors of the auto-correlation matrices of the training examples in each patch, sorted by a decreasing order of their corresponding eigenvalues. Another induced difference is the replacement of the sparse coding representation with a simple linear approximation that projects on the first several eigenvectors, until the representation error decreases below the specified threshold. Comparison to the PCA-Based compression method is important here, because whereas the proposed technique could be interpreted as an adaptation of the JPEG2000 to the image family by training, the PCA could be seen as a similar step emerging from JPEG.

Figures 13.11, 13.12 and 13.13 show a visual comparison of the obtained results with the JPEG, JPEG2000, and the PCA methods at three different bit-rates. Note that Figure 13.11 does not contain JPEG examples because the bit-rate is below the possible minimum in this case. These figures clearly show the visual and quantitative advantage of the proposed sparse-representation-based method over all the alternatives. We can especially notice the strong smearing effect in the JPEG and JPEG2000 images, contrary to the images obtained by the proposed algorithm.

Figure 13.14 shows a rate-distortion curves comparison of the compression methods mentioned above, averaged over the 100 test images. The PSNR shown here corresponds to the aligned grayscale images for all methods, in order to evaluate the representation error from the sparse coding and reconstruction stages alone, without the error that results from the geometrical warp process. Adding the error induced by the geometrical warp implies a decrease of 0.2 – 0.4dB for the VQ, PCA, and K-SVD methods, which does not change the overall ordering of these algorithms.

In addition to these rate-distortion curves we added a curve for a “Clean” JPEG2000 method, which is simply a horizontally shifted version of the JPEG2000 curve taking into account the header embedded in the JPEG2000 standard. This header was found to be approximately 220 bytes.

It can be seen from this comparative graph that the K-SVD compression method presents the best performance and especially so in the very low bit-rates of 300 – 1000 bytes, which is the area of interest.



**Fig. 13.11** Facial images compression at a bit-rate of 400 bytes, comparing results of JPEG2000, the PCA, and the K-SVD methods. The values in the brackets are the representation RMSE.

### 13.4.5 Dictionary Redundancy

Dictionary redundancy, or over-completeness, is defined as  $m/n$ , being 1 for complete (non-redundant) representations, and above it in the experiments reported above. A natural question is whether such redundancy is truly necessary for getting the compression results shown. Fixing the patch size,  $n$ , what would be the effect of changing the number of atoms,  $m$ ? [Figure 13.15](#) shows four curves of the averaged PSNR on the test set images with varying redundancy. Each curve shows this behavior for a different fixed image bit-rate. Note that since the patch size is  $15 \times 15$  pixels, having 225 atoms implies a complete (non-redundant) dictionary, and when having 150 atoms, we are in fact in an under-complete regime.



**Fig. 13.12** Facial images compression at a bit-rate of 550 bytes, comparing results of JPEG, JPEG2000, the PCA, and the K-SVD methods. The values in the brackets show the representation RMSE.

We can see in [Figure 13.15](#) that the quality of the images is improving (PSNR-wise) with the dictionary redundancy, but this increase is subsiding. Intuitively, we would expect a change in the tendency of the curves at some point, partly due to over-fitting of the dictionaries (due to the limited number of examples), and partly because too high redundancy is expected to cause a deterioration in performance, due to the cost of transmitting the indices. Such change in tendency has not been observed because of limitations in the training data size.

## 13.5 Post-Processing for Deblocking

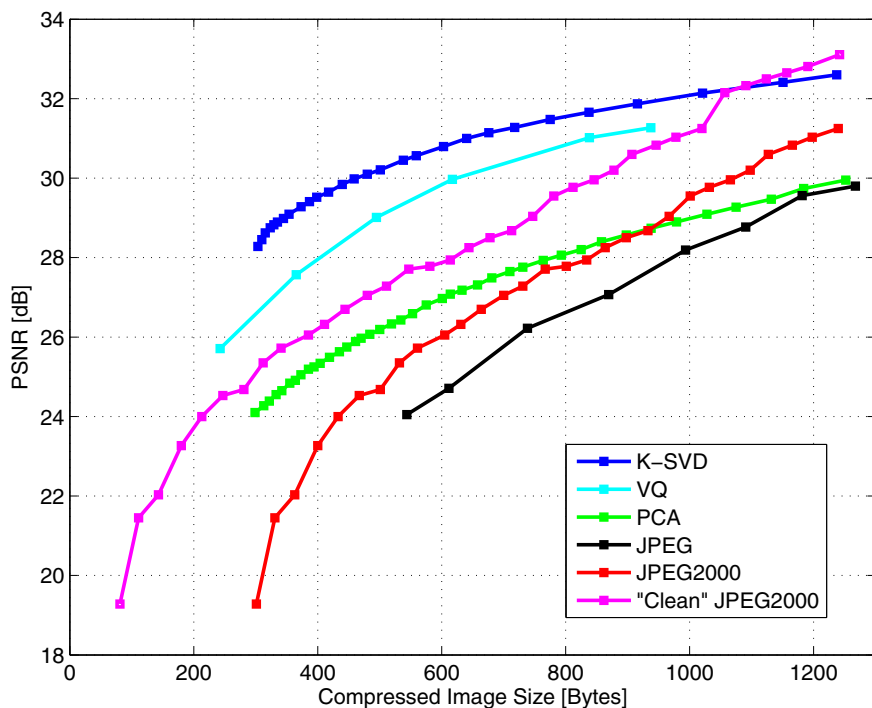
### 13.5.1 The Blockiness Artifacts

The motivation for a deblocking method in this scheme is the appearance of blockiness artifacts in the reconstructed images that result from the nature of the proposed



**Fig. 13.13** Facial images compression at a bit-rate of 820 bytes, comparing results of JPEG, JPEG2000, the PCA, and the K-SVD methods. The values in the brackets show the representation RMSE.

compression algorithm. These disturbing artifacts are visible all over the image in varying degrees of strength, and all the reconstructed images suffer from them. The blockiness artifacts are created because the coding of each image is done in uniform non-overlapping patches, and these patches are coded independently according to their relevant trained dictionaries. Therefore, the nature of the representation in each patch is different, and especially so in patches which require a different number of atoms or which have a different pattern of data in them. These differences can create false edges between the patches, as can be seen in [Figures 13.11-13.13](#). This phenomenon is expected to appear in every compression technique in which the encoding is done in non-overlapping patches, but it is more significant in the presented method, due to the deep compression ratio, and the size of the patches. Naturally, patch sizes can be decreased and dictionaries can be extended, but doing so will not eliminate the artifacts completely and will severely hurt the performance of the compression algorithm as the bit-rate of each compressed image will increase. Such a solution is thus inefficient from a wider point of view.

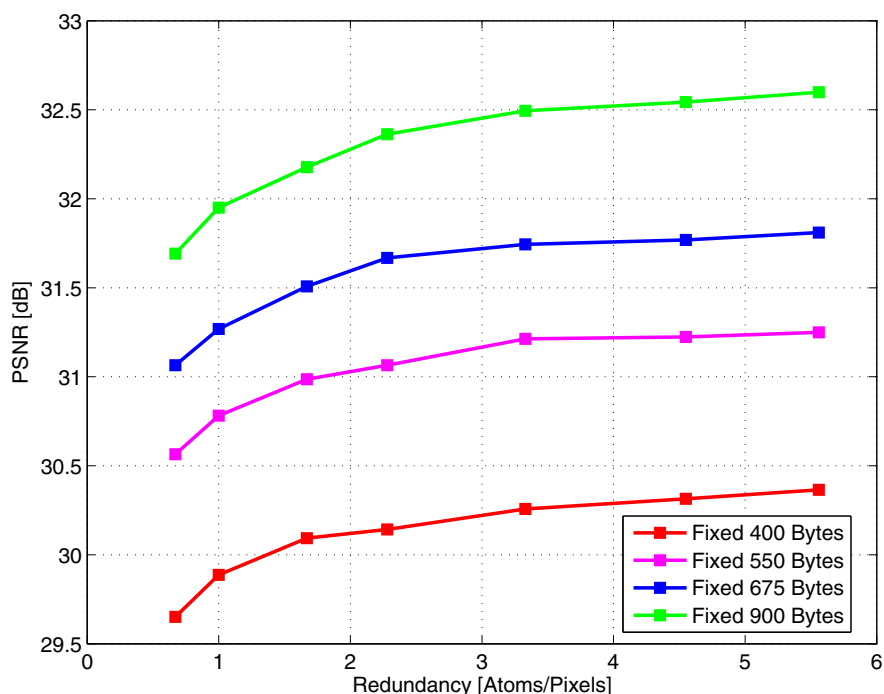


**Fig. 13.14** Rate-Distortion curves for the JPEG, JPEG2000, “Clean” JPEG2000 (i.e., after removal of the header bits), PCA, VQ, and K-SVD methods.

### 13.5.2 Possible Approaches For Deblocking

There are several possible approaches one can take to eliminate the blockiness artifacts, and these mostly divide into two main categories: Encoding stage approaches and post-processing approaches. The encoding stage approaches suggest attacking the problem by preventing the blockiness artifacts from being created in the first place. Such possible approaches include encoding in overlapping patches, train dictionaries on two or more shifted grids, train dictionaries with respect to their neighboring dictionaries and other approaches along these lines. These approaches offer a clean structural solution to the problem and are designed to eliminate the blockiness artifacts completely. However, the disadvantages of these methods should not be overlooked – they are complex, some are complicated to implement, and they can lead to an increase in the bit-rate of the compressed images, which is highly undesirable.

In contrary to encoding stage approaches, post-processing approaches attempt to eliminate the blockiness artifacts after they have already been created by the compression algorithm. Such possible approaches include various filtering techniques



**Fig. 13.15** The effect of changing the redundancy in the trained dictionaries on the quality of the test set images in PSNR for a fixed bit-rate (four different fixed values).

applied to the compressed-decompressed image. These approaches are simple and easy to implement, and they do not increase the rate of the compressed image. The main disadvantage of these methods is of course that they cannot guaranty to eliminate the blockiness artifacts in all cases and all images, as the method of operation is limited and does not have an access to the source of the problem.

### 13.5.3 Learning-Based Deblocking Approach

The deblocking method we present here belongs to the second category, as it is an application of specially created filters over the reconstructed image. We define a three pixel wide border-frame of each patch as the Region-Of-Interest (ROI) to be fixed. These areas suffer the most from the blockiness artifacts, and their location is known to us, once the grid is set. For each pixel in the ROI of each different patch we calculate *a different*  $5 \times 5$  linear filter to correct the blockiness artifacts. This is done using a learning set for the same pixel, accumulated over many (1,000 in this case)



similar images. The learning set consists of original pixels and the reconstructed ones after compression-decompression.

In general, this approach is similar in its spirit to the principles of the proposed compression algorithm, in which the encoding stage for new images is done according to dictionaries that were trained according to a learning set. In this case, however, we calculate the optimal filters by the Least-Squares penalty

$$\hat{\mathbf{h}}_k = \arg \min_{\mathbf{h}_k} \|\mathbf{Y}\mathbf{R}_k\mathbf{h}_k - \mathbf{X}\mathbf{e}_k\|_2^2, \quad (13.3)$$

where  $\mathbf{h}_k$  is the desired  $5 \times 5$  filter for the  $k^{th}$  pixel, ordered as a column vector. The matrix  $\mathbf{X}$  contains the original images in the learning set as its rows, and similarly,  $\mathbf{Y}$  is the matrix containing the decoded images in the learning set as rows.  $\mathbf{R}_k$  is a matrix that extracts from  $\mathbf{Y}$  the appropriate pixels in the  $5 \times 5$  environment of the  $k^{th}$  pixel, and  $\mathbf{e}_k$  is a vector that selects the  $k^{th}$  pixel out of the images  $\mathbf{X}$ .

The goal of this minimization problem is to design a linear filter that attempts to best fit the  $k^{th}$  pixel in the reconstructed image to the one in the original image, using only the pixel's  $5 \times 5$  neighborhood. Equation 13.3 has a simple analytic solution for  $\hat{\mathbf{h}}_k$  in the form of

$$\hat{\mathbf{h}}_k = (\mathbf{R}_k^T \mathbf{Y}^T \mathbf{Y} \mathbf{R}_k)^{-1} \cdot \mathbf{R}_k^T \mathbf{Y}^T \mathbf{X} \mathbf{e}_k. \quad (13.4)$$

Note that the needed filters vary from one bit-rate to another, and thus should be calculated for each separately.<sup>7</sup> For a given bit-rate group, the trained filters can be combined as the rows of a large matrix  $\mathbf{H}_r$ , such that when it multiplies the compressed-decompressed image, all the filters are activated, and deblocking is performed.

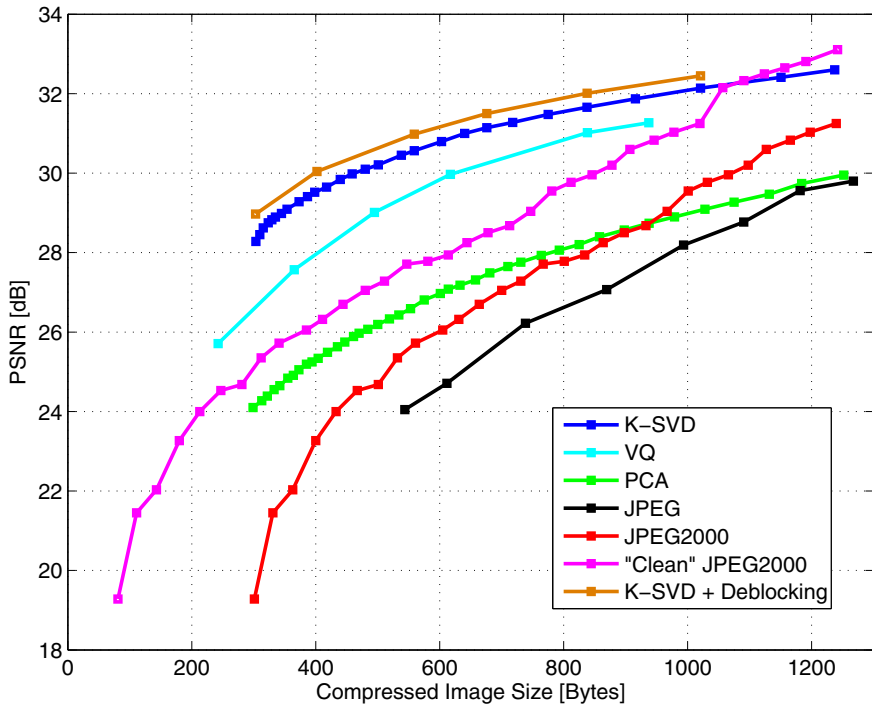
This filter calculation process is simple, easy to implement and does not increase the bit-rate of the compression, as the resulting matrices of filters for each bit-rate are calculated in an off-line mode and are stored in the decoder. The method of using these filters on any given reconstructed image is also very convenient, simply by multiplying the matrix  $\mathbf{H}_r$  by the lexicographic representation of the image in the post-processing stage of the compression algorithm.

## 13.6 Deblocking Results

We use 1,000 facial images as our learning set and 100 different images as our test set. All the images in both sets are of size  $179 \times 221$  after pre-processing. Figure 13.16 shows a rate-distortion curves comparison as in Figure 13.14, with a new curve that corresponds to the deblocked results. As can be seen, an improvement of 0.4dB is obtained because of the proposed deblocking method.

<sup>7</sup> Doing this for each bit-rate is impractical, and not necessary. Our bit-rate range (300-1000 bytes) can be divided to groups, and each should get its own set of filters.



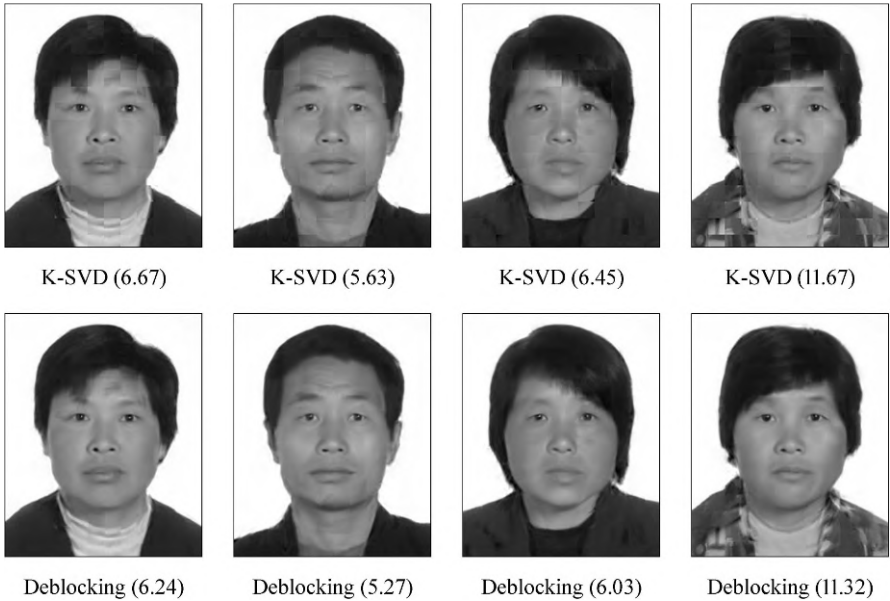


**Fig. 13.16** Rate-Distortion curves for the JPEG, JPEG2000, “Clean” JPEG2000, PCA, VQ, and the two K-SVD methods.

In order to visually show the effectiveness of the presented deblocking method, we apply it on the same images as in Figure 13.12, referring to a rate of 550 bytes. We also add an additional image where the blockiness artifacts are particularly strong. The filters matrix that was calculated for this bit-rate is used on the reconstructed images as a post-processing operation and the results can be seen in Figure 13.17. In spite of its limitations, it can be clearly seen that the deblocking method has eliminated the blockiness artifacts in most places and reduced them significantly in others. Although the deblocking filters create a small degree of smoothing along the block edges where they are applied, the images still appear sharp and their overall visual quality is improved. In addition to the improvement in the visual quality, which is our primary goal, we also gain an improvement in the PSNR of the images, as seen in Figure 13.16.

## 13.7 Summary

In this chapter we have demonstrated how sparse and redundant representations and the K-SVD learning algorithm can be used for handling a practical image compression.



**Fig. 13.17** A comparison of reconstructed images at the rate of 550 bytes before (top) and after (bottom) applying the linear deblocking method.

sion application – coding of facial images. The devised scheme is practical, simple, and direct, and yet it leads to performance surpassing the JPEG2000 algorithm.

Unfortunately, the path taken in this chapter cannot be imitated in order to propose a general purpose image compression algorithm, and this remains as a major challenge for the near future. Still, from the experiments demonstrated here we have learned several important lessons:

- Contrary to intuition, redundancy can be of value in improving compression performance;
- Sparse and redundant representations can successfully replace a VQ scheme under some conditions, and lead to better results; and
- In the broad comparison between linear and non-linear approximation (PCA versus K-SVD in our tests), we have seen here that despite the additional cost of transmitting indices, the non-linear approximation performs better.

## Further Reading

1. P.J. Burt and E.H. Adelson, The Laplacian pyramid as a compact image code, *IEEE Trans. on Communications*, 31:532–540, 1983.

2. O. Bryt and M. Elad, Compression of facial images using the K-SVD algorithm, *Journal of Visual Communication and Image Representation*, 19(4):270–283, May 2008.
3. O. Bryt and M. Elad, Improving the K-SVD facial image compression using a linear deblocking method, The IEEE 25-th Convention of Electrical and Electronics Engineers in Israel, Eilat, Israel, December 3–5, 2008.
4. P. Cosman, R. Gray, and M. Vetterli, Vector quantization of images subbands: A survey, *IEEE Trans. Image Processing*, 5:202–225, 1996.
5. M. Elad, R. Goldenberg, and R. Kimmel, Low bit-rate compression of facial images, *IEEE Trans. on Image Processing*, 16:2379–2383, 2007.
6. A.J. Ferreira and M.A.T. Figueiredo, Class-adapted image compression using independent component analysis, in: Proceedings of the International Conference on Image Processing (ICIP), Barcelona, Spain, pp. 14–17, September 2003.
7. A.J. Ferreira and M.A.T. Figueiredo, Image compression using orthogonalized independent components bases, in: Proceedings of the IEEE XIII Workshop on Neural Networks for Signal Processing, Toulouse, France, 17–19, September 2003.
8. A.J. Ferreira and M.A.T. Figueiredo, On the use of independent component analysis for image compression, *Signal Processing: Image Communication*, 21:378–389, 2006.
9. O.N. Gerek and C. Hatice, Segmentation based coding of human face images for retrieval, *Signal-Processing*, 84:1041–1047, 2004.
10. A. Gersho and R.M. Gray, *Vector Quantization And Signal Compression*, Kluwer Academic Publishers, Dordrecht, Netherlands, 1992.
11. T. Hazan, S. Polak, and A. Shashua, Sparse image coding using a 3D non-negative tensor factorization, in: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV), Beijing, China, pp. 17–21, October 2005.
12. J.H. Hu, R.S. Wang, and Y. Wang, Compression of personal identification pictures using vector quantization with facial feature correction, *Optical-Engineering*, 5:198–203, 1996.
13. J. Huang and Y. Wang, Compression of color facial images using feature correction two-stage vector quantization, *IEEE Trans. on Image Processing*, 8:102–109, 1999.
14. K. Inoue and K. Urahama, DSVD: a tensor-based image compression and recognition method, in: Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS), Kobe, Japan, pp. 23–26, May 2005.
15. A. Lanitis, C.J. Taylor, and T.F. Cootes, Automatic interpretation and coding of face images using flexible models, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19:743–756, 1997.
16. Y. Linde, A. Buzo, and R.M. Gray, An algorithm for vector quantiser design, *IEEE Trans. on Communications*, 28:84–95, 1980.

17. B. Moghaddam and A. Pentland, An automatic system for model-based coding of faces, in: Proceedings of DCC '95 Data Compression Conference, Snowbird, UT, USA, pp. 28–30, March 1995.
18. L. Peotta, L. Granai, and P. Vandergheynst, Image compression using an edge adapted redundant dictionary and wavelets, *Signal- Processing*, 86:444–456, 2006.
19. Z. Qiuyu and W. Suozhong, Color personal ID photo compression based on object segmentation, in: Proceedings of the Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM), Victoria, BC, Canada, pp. 24–26, August 2005.
20. M. Sakalli and H. Yan, Feature-based compression of human face images, *Optical-Engineering*, 37:1520–1529, 1998.
21. M. Sakalli, H. Yan, and A. Fu, A region-based scheme using RKLT and predictive classified vector quantization, *Computer Vision and Image Understanding*, 75:269–280, 1999.
22. M. Sakalli, H. Yan, K.M. Lam, and T. Kondo, Model-based multi-stage compression of human face images, in: Proceedings of 14th International Conference on Pattern Recognition (ICPR), Brisbane, Qld., Australia, pp. 16–20, August 1998.
23. A. Shashua and A. Levin, Linear image coding for regression and classification using the tensor-rank principle, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), Kauai, HI, USA, pp. 8–14, December 2001.
24. D.S. Taubman and M.W. Marcellin, *JPEG2000: Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, Norwell, MA, USA, 2001.



# Chapter 14

## Image Denoising

### 14.1 General Introduction – Image Denoising

Images often contain noise, which may arise due to sensor imperfection, poor illumination, or communication errors. Removing such noise is of great benefit in many applications, and this may explain the vast interest in this problem and its solution. However, the importance of the image denoising problem goes beyond the evident applications it serves. Being the simplest possible inverse problem, it provides a convenient platform over which image processing ideas and techniques can be tested and perfected. Indeed, numerous contributions in the past 50 years or so address this problem from many and diverse points of view. Statistical estimators of all sorts, spatial adaptive filters, stochastic analysis, partial differential equations, transform-domain methods, splines and other approximation theory methods, morphological analysis, differential geometry, order statistics, and more, are some of the many directions and tools explored in studying this problem.

In this chapter we have no intention of providing a survey of this vast activity on image denoising. Instead, we concentrate on a specific family of algorithms that are of interest in the context of this book – algorithms that use *sparse and redundant representation* modeling. These methods have been found in recent years to be highly effective and promising, often leading to the best known performance in terms of the noise removal results they achieve.

Our exposition starts with an algorithm that builds on the rough core of sparse representation modeling, similar to the content in Chapter 10, and adopting a global treatment of the image denoising problem. Then we present various improvements of this paradigm that propose local processing, learning of the shrinkage curve, incorporating dictionary training, and more. We cover also the Non-Local-Means (NLM) algorithm, because of an interesting tie we expose to sparse-representation methods covered. We conclude this chapter with an introduction of the Stein Unbiased Risk Estimator (SURE), used for an automatic tuning of the parameters of some image denoising algorithms.

We start by a proper modeling of the problem. An ideal image  $\mathbf{y}_0 \in \mathbb{R}^N$  (of size  $\sqrt{N} \times \sqrt{N}$  pixels) is measured in the presence of an additive zero-mean white and homogeneous Gaussian noise,  $\mathbf{v}$ , with a known standard-deviation  $\sigma$ . The measured image,  $\mathbf{y}$ , is thus

$$\mathbf{y} = \mathbf{y}_0 + \mathbf{v}. \quad (14.1)$$

We aim to design an algorithm that can remove the noise from  $\mathbf{y}$ , returning as close as possible to the original image,  $\mathbf{y}_0$ . We start by a direct application of the ideas posed in Chapters 9 and 10 for solving this problem, and evolve from there to more advanced techniques.

## 14.2 The Beginning: Global Modeling

### 14.2.1 The Core Image-Denoising Algorithm

The signal denoising problem has been mentioned over and over again throughout this book, and when it comes to solving it using sparse and redundant representation modeling, we were always led to some variant of the optimization problem ( $P_0^\epsilon$ ),

$$(P_0^\epsilon) \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2 \leq \epsilon. \quad (14.2)$$

The threshold  $\epsilon$  is closely tied with the noise power, and a natural choice of it would be  $cN\sigma^2$ , with  $0.5 \leq c \leq 1.5$ . The solution to this problem, denoted by  $\hat{\mathbf{x}}$ , is the sparse representation that represents the desired clean image. Thus, the denoised output is  $\hat{\mathbf{y}} = \mathbf{A}\hat{\mathbf{x}}$ .

In previous chapters we have seen this method being recommended from both a deterministic and a stochastic points of view, we have introduced ways to fine-tune it to perform better by adding assumptions on the statistics of the non-zero entries in  $\mathbf{x}$ , we have explored various options to further improve the denoising effect by approximating the MMSE estimator, and more. All these facts lead to the claim that this formulation is solid and reliable.

Naturally, the above formulation calls for a proper choice of the dictionary  $\mathbf{A}$ . Using sparse representations with specific choices of  $\mathbf{A}$  as a driving force for denoising of images has drawn a lot of research attention in the past decade or so. Initially, sparsity of the unitary wavelet coefficients was considered, leading to the celebrated shrinkage algorithm, as already mentioned in Chapter 6. However, the performance of this method was found to be insufficient and leaving much room for improvements, leading researchers to consider redundant representations.

One reason to use redundant representations is the desire to have a shift-invariance property. In terms of the denoising algorithm, this implies that no matter where the edges are, their treatment will remain the same. Also, with the growing realization that regular separable 1D wavelets are inappropriate for handling images (i.e., these

transforms are unable to represent images sparsely), several new tailored multi-scale and directional redundant transforms were introduced, including the curvelets, contourlets, wedgelets, bandelets, and the steerable wavelets. The use of these dictionaries within the above formulation of the image denoising solver leads to far better results, when compared to the unitary wavelet shrinkage method. For a while (the early 2000's), these algorithms were considered as the best available image denoising methods.

Let us demonstrate an elementary image denoising algorithm that relies on a redundant dictionary, in the spirit of the above discussion. Using the redundant Haar transform for  $\mathbf{A}$ , as used in Chapter 10,<sup>1</sup> we aim to denoise an image. For the approximation of the solution of the problem posed in Equation (14.2) we use the thresholding algorithm. Thus, the overall denoising process is comprised of the formula

$$\hat{\mathbf{y}} = \mathbf{A} \mathcal{S}_T (\mathbf{A}^T \mathbf{y}), \quad (14.3)$$

where  $\mathcal{S}_T$  is a scalar hard-thresholding operator (i.e.,  $\mathcal{S}_T(z) = 0$  for  $|z| < T$ , and  $\mathcal{S}_T(z) = z$  otherwise).

Since the columns of  $\mathbf{A}$  are not  $\ell_2$ -normalized, we should scale the thresholds by these norms, implying that the  $i$ -th entry  $\mathbf{a}_i^T \mathbf{y}$  in the vector  $\mathbf{A}^T \mathbf{y}$  is to be thresholded by the value  $\|\mathbf{a}_i\|_2 \cdot T$ . Put differently, we can use a fixed value of threshold  $T$  in the formula

$$\hat{\mathbf{y}} = \mathbf{A} \mathcal{S}_T (\mathbf{W}^{-1} \mathbf{A}^T \mathbf{y}), \quad (14.4)$$

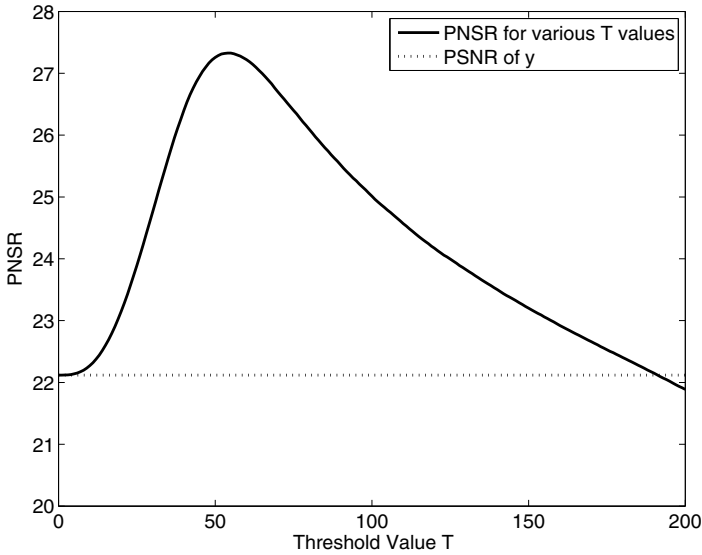
where  $\mathbf{W}$  is a diagonal matrix that contains the norms of the atoms,  $\|\mathbf{a}_i\|_2$ . Note that this algorithm could be interpreted as the first iteration in an iterative-shrinkage algorithm, which is initialized with zeros. More on the reason to use  $\mathbf{W}$  appears next.

Figure 14.1 presents the quality of the outcome (in terms of PSNR) of this algorithm as a function of the parameter  $T$ , for the input image *Barbara*, contaminated by additive Gaussian noise of standard-deviation  $\sigma = 20$ .<sup>2</sup> As can be seen, for the best choice of  $T$ , we obtain an improvement of more than 5.2dB over the input noisy image. Figure 14.2 shows the original image, the noisy one, and the result obtained for the best choice of  $T$ , which seems to be clear of most of the initial noise.

<sup>1</sup> This matrix represents a tight Frame with redundancy factor of 7 : 1.

<sup>2</sup> This image and the denoising of it (specifically, the removal of a zero-mean white and additive Gaussian noise with  $\sigma = 20$ ) will accompany us throughout this chapter, as a benchmark for comparing various denoising strategies.





**Fig. 14.1** The PSNR obtained by the thresholding algorithm with the redundant Haar dictionary, as a function of the threshold parameter  $T$ . While the input image has PSNR = 22.11dB, the best possible result by the thresholding procedure leads to PSNR = 27.33dB.

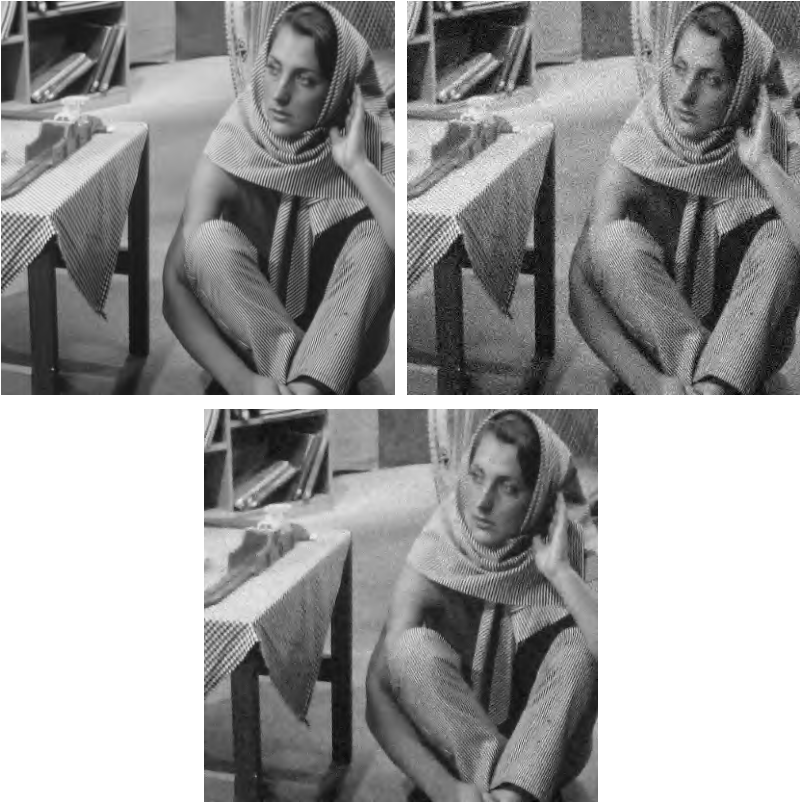
### 14.2.2 Various Improvements

In order to further improve the performance of these methods, various additional ideas and tools have been proposed. One such simple but valuable idea, is the assignment of a different variance to each non-zero coefficient in the representation. This stems from the observation that the magnitude of typical sparse representations of images show a varying variance per scale and orientation. Plugged into the Basis-Pursuit formulation, this leads to the problem

$$\hat{\mathbf{x}} = \min_{\mathbf{x}} \lambda \|\mathbf{W}\mathbf{x}\|_1 + \frac{1}{2} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2^2, \quad (14.5)$$

where  $\mathbf{W}$  is a diagonal matrix that contains the inverse of the variances chosen, just as described earlier in Equation (14.4). We have seen in Chapter 3 that this structure implies a variation of the norms of the atoms in the dictionary  $\mathbf{A}$ . The diagonal of  $\mathbf{W}$  can be gathered off-line by exploring the general behavior of the representations obtained for various images. Alternatively, these weights can be learned on-line from the given image, as proposed by Chang, Yu, and Vetterli, and then plugged to the solver.

Other, more involved, ideas that lead to improved denoising performance are those which somehow tie the representation coefficients one to the other. For example, assume that we have the estimated variances of the entries in the representation



**Fig. 14.2** The original image (top-left), the noisy one with  $\sigma = 20$  grey-values (top-right), and the denoised one obtained by the thresholding algorithm with the redundant Haar dictionary (bottom). The input image quality is PSNR = 22.11dB, and the result shown here has PSNR = 27.33dB.

$\mathbf{x}$ . Also, for each entry  $x_i$  in the representation we define its neighborhood  $\mathcal{N}(i)$  (e.g., near-by entries in the spatial domain having similar orientation and scale). The formulation

$$\hat{\mathbf{x}} = \min_{\mathbf{x}} \lambda \sum_i \left[ \sum_{j \in \mathcal{N}(i)} \left( \frac{x_j}{\sigma_j} \right)^2 \right]^{\frac{1}{2}} + \frac{1}{2} \|\mathbf{Ax} - \mathbf{y}\|_2^2, \quad (14.6)$$

replaces the weighted  $\ell_1$ -norm with a new  $\ell_1 - \ell_2$  mixed measure that promotes neighborhood groups of entries to behave similarly (either all the elements in the neighborhood are zeros, or all become active). For the case where the neighborhood contains only the center entry, this formulation coincides with the one posed in Equation (14.5).

While we do not explore this method here, we mention that this technique proposes to exploit the known correlation that exists between the absolute values of neighboring coefficients, in order to constrain the denoising algorithm towards a better behaved image. The same motivation stands behind more complicated denoising schemes such as the Bayesian Least-Squares method with the use of the Gaussian Scale-Mixture (GSM) by Portilla et. al. (which leads to a PSNR of 30.32dB for the Barbara test introduced above). While these techniques indeed show better performance, they lose some of their appeal due to their overall complicated structure. As we shall see next, alternative simple and intuitive image denoising algorithms can be proposed, while leading to even better results.

To conclude this discussion, we observe that all the above is done while imposing a global model on the image  $\mathbf{x}$ . One major drawback with such a model is the need to provide a dictionary that sparsifies images as a whole. An appealing alternative to this global approach is a local processing of the image by operating on small patches. As we show next, this will allow us to introduce trained dictionaries, learn the shrinkage curve, and propose simple algorithms that are easily implemented and are highly parallelizable.

## 14.3 From Global to Local Modeling

### 14.3.1 The General Methodology

Instead of working on the whole image at once, we can consider small image patches  $\mathbf{p}$  of size  $\sqrt{n} \times \sqrt{n}$  pixels, where  $n \ll N$ . We organize these patches lexicographically as column vectors  $\mathbf{p} \in \mathbb{R}^n$ . We shall assume that these patches follow the *Sparse-Land* model  $\mathcal{M}(\mathbf{A}, \epsilon, k_0)$ . This implies that there exists a dictionary  $\mathbf{A} \in \mathbb{R}^{n \times m}$  ( $m \geq n$ ), such that every patch  $\mathbf{p}$  has a corresponding representation  $\mathbf{q} \in \mathbb{R}^m$  satisfying  $\|\mathbf{A}\mathbf{q} - \mathbf{p}\|_2 \leq \epsilon$ , with  $\|\mathbf{q}\|_0 = k_0$ . At the moment we shall assume that  $\mathbf{A}$  is known and fixed.

The above model assumption imposes a prior on images in terms of the behavior of their local patches everywhere. How can this be used for image denoising? One natural option is to denoise each of those patches separately and tile the results. However, in doing so, visible artifacts may occur on block boundaries. One could also propose to work on overlapping patches and average the results in order to prevent such blockiness artifacts, as indeed practiced in various recent contributions.

As an example for such a local denoising approach, consider the following algorithm, as initially proposed by Guleryuz and later improved by others. The dictionary to be used is the unitary 2D-DCT, which leads to two important benefits: (i) multiplication by  $\mathbf{A}$  and its adjoint are fast (both because of separability, and the relation to FFT); and (ii) sparse-coding with this dictionary is simple and direct. Denoising the complete image  $\mathbf{y}$  can be done by the following stages.

1. Consider every pixel in  $\mathbf{y}$  as a center of a patch of size  $\sqrt{n} \times \sqrt{n}$  (typical value of  $n$  is 64). We denote these patches as  $\mathbf{p}_{ij}$ .
2. Apply denoising of each such patch,  $\mathbf{p}_{ij}$ , using the thresholding algorithm:
  - a. Compute  $\tilde{\mathbf{q}}_{ij} = \mathbf{A}^T \mathbf{p}_{ij}$  (the forward 2D-DCT transform).
  - b. Pass the obtained vector  $\tilde{\mathbf{q}}_{ij}$  through a hard-thresholding operation with a pre-specified threshold (dependent on the noise power, and possibly different for each entry) and obtain  $\hat{\mathbf{q}}_{ij}$ .
  - c. Compute  $\hat{\mathbf{p}}_{ij} = \mathbf{A} \hat{\mathbf{q}}_{ij}$ . This is the denoised version of the patch.
3. Merge the denoised patches by averaging them one on top of the other. There is also a possibility to use a weighted averaging to gain a further improvement, but we shall not explore this option here.

This simple algorithm seems to be surprisingly effective for image denoising, and in the next subsections we shall explore several further improvements of it. Interestingly, here and elsewhere, the overlaps between the patches are found to be crucial for the good performance of the algorithm, and not just because of the blockiness effects. The obtained averaging has the same flavor as the averaging proposed in Chapter 11 for approximating the MMSE estimator.

### 14.3.2 Learning the Shrinkage Curves

The ideas presented in this sub-section are taken from the recent work by Yakov Hel-Or and Doron Shaked.

The patch denoising process in the above-described algorithm can be written as the expression

$$\hat{\mathbf{p}}_{ij} = \mathbf{A} \hat{\mathbf{q}}_{ij} = \mathbf{A} \mathcal{S} \left\{ \mathbf{A}^T \mathbf{p}_{ij} \right\}, \quad (14.7)$$

where  $\mathcal{S}$  represents the element-wise shrinkage applied on the coefficients in  $\tilde{\mathbf{q}}_{ij}$ .

Assume that we hold a large corpus of clean image patches,  $\{\mathbf{p}_k^0\}_{k=1}^M$ , to train on. We create from these a new set of their noisy versions,  $\{\mathbf{p}_k\}_{k=1}^M$ , simply by adding white additive Gaussian noise with the pre-specified variance  $\sigma^2$ . Using these  $M$  pairs,  $\{\mathbf{p}_k^0, \mathbf{p}_k\}_{k=1}^M$ , we aim to find the optimal shrinkage rule, replacing the hard-thresholding practiced above, and also finding the best set of threshold parameters in general. We start with the following penalty function,

$$F_{local}(\mathcal{S}) = \sum_{k=1}^M \left\| \mathbf{p}_k^0 - \mathbf{A} \mathcal{S} \left\{ \mathbf{A}^T \mathbf{p}_k \right\} \right\|_2^2. \quad (14.8)$$

In this penalty function, which is a function of the shrinkage operation, we aim to apply the proposed algorithm on the noisy patches, and get as close as possible to the clean (original) ones.

We shall assume hereafter that the shrinkage operation is applied differently to every element in the input vector. This is a natural assumption, since the near-DC coefficients are less sensitive to noise compared to the high-AC ones, and as such, they should go through a less aggressive shrinkage operation. Thus, the operation  $\mathcal{S}\{\mathbf{u}\}$  on an arbitrary vector  $\mathbf{u} \in \mathbb{R}^m$  is broken into

$$\mathcal{S}\{\mathbf{u}\} = \begin{bmatrix} \mathcal{S}_1\{u[1]\} \\ \mathcal{S}_2\{u[2]\} \\ \vdots \\ \mathcal{S}_m\{u[m]\} \end{bmatrix}, \quad (14.9)$$

where each  $\mathcal{S}_i$  may operate differently. Returning to the penalty function in Equation (14.8), and denoting  $\mathbf{u}_k = \mathbf{A}^T \mathbf{p}_k$ , it can be written as

$$\begin{aligned} F_{local}(\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_m) &= \sum_{k=1}^M \left\| \mathbf{p}_k^0 - \mathbf{A} \mathcal{S}\{\mathbf{A}^T \mathbf{p}_k\} \right\|_2^2 \\ &= \sum_{k=1}^M \left\| \mathbf{p}_k^0 - \sum_{i=1}^m \mathbf{a}_i \mathcal{S}_i\{u_k[i]\} \right\|_2^2. \end{aligned} \quad (14.10)$$

Note that if the dictionary used is unitary, and we introduce the notations  $\mathbf{u}_k^0 = \mathbf{A}^T \mathbf{p}_k^0$  and  $\mathbf{u}_k = \mathbf{A}^T \mathbf{p}_k$ , this penalty function can be written alternatively as

$$\begin{aligned} F_{local}(\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_m) &= \sum_{k=1}^M \left\| \mathbf{p}_k^0 - \mathbf{A} \mathcal{S}\{\mathbf{A}^T \mathbf{p}_k\} \right\|_2^2 \\ &= \sum_{k=1}^M \left\| \mathbf{A}^T \mathbf{p}_k^0 - \mathcal{S}\{\mathbf{A}^T \mathbf{p}_k\} \right\|_2^2 \\ &= \sum_{k=1}^M \left\| \mathbf{u}_k^0 - \mathcal{S}\{\mathbf{u}_k\} \right\|_2^2 \\ &= \sum_{i=1}^m \sum_{k=1}^M \left( u_k^0[i] - \mathcal{S}_i\{u_k[i]\} \right)^2. \end{aligned} \quad (14.11)$$

This reveals that the  $m$  shrinkage curves can be learned independently of each other, as we have obtained  $m$  decoupled penalties, each handling one shrinkage curve.

Returning to the more general case, the next step we take is a proper representation of the shrinkage operator, so as to optimize its curve. We propose a polynomial of the form

$$\mathcal{S}_i\{u\} = \sum_{j=0}^J c_i[j] u^j, \quad (14.12)$$

and thus the choice of  $\{c_i[j]\}_{j=0}^J$  characterizes this operator fully. The advantage in this representation is the fact that when plugged to the penalty function in Equation (14.10), the overall expression admits a simple quadratic form, for which plain Least-Squares can find the optimal solution. There are other representations that can be proposed for  $\mathcal{S}_i\{u\}$ , such as perfect anti-symmetric polynomial with odd powers only, a spline representation, and more. In their work, Hel-Or and Shaked propose a piecewise linear curve, in which the connecting points are trained.

Returning to our formulation and plugging in the polynomial model of the shrinkage, the penalty we consider becomes

$$\begin{aligned} F_{local}(\mathcal{S}_1, \mathcal{S}_2, \dots, \mathcal{S}_m) &= \sum_{k=1}^M \left\| \mathbf{p}_k^0 - \sum_{i=1}^m \mathbf{a}_i \mathcal{S}_i \{u_k[i]\} \right\|_2^2 \\ &= \sum_{k=1}^M \left\| \mathbf{p}_k^0 - \sum_{i=1}^m \mathbf{a}_i \sum_{j=0}^J c_i[j] u_k[i]^j \right\|_2^2. \end{aligned} \quad (14.13)$$

We proceed by introducing the following matrix notation, which will enable easy minimization of the above penalty. We define the vector  $\mathbf{c} \in \mathbb{R}^{mJ}$  as the vector that contains all the  $\{c_i[j]\}_{j=0}^J$  series, for  $i = 1, \dots, m$ . Thus

$$\mathbf{c}^T = [c_1[0], c_1[1], \dots, c_1[J], c_2[0], c_2[1], \dots, c_m[0], c_m[1], \dots, c_m[J]].$$

Similarly, we define the matrix  $\mathbf{U}_k \in \mathbb{R}^{m \times mJ}$ , as a block-diagonal matrix with  $m$  blocks,

$$\mathbf{U}_k = \begin{bmatrix} \mathbf{b}_k[1] & \mathbf{0} & \dots & \mathbf{0} \\ \mathbf{0} & \mathbf{b}_k[2] & \dots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{b}_k[m] \end{bmatrix}. \quad (14.14)$$

Each of the blocks  $\mathbf{b}_k[i]$  is of size  $1 \times J$ , with the content

$$\mathbf{b}_k[i] = [u_k[i]^0, u_k[i]^1, \dots, u_k[i]^J], \text{ for } i = 1, 2, \dots, m.$$

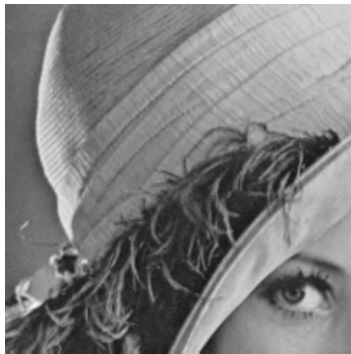
Using these notations, the function we aim to minimize is given by

$$F_{local}(\mathbf{c}) = \sum_{k=1}^M \left\| \mathbf{p}_k^0 - \mathbf{A} \mathbf{U}_k \mathbf{c} \right\|_2^2. \quad (14.15)$$

The optimal set of parameters that define the shrinkage curves is given by

$$\frac{\partial F_{local}(\mathbf{c})}{\partial \mathbf{c}} = \mathbf{0} = \sum_{k=1}^M \mathbf{U}_k^T \mathbf{A}^T (\mathbf{A} \mathbf{U}_k \mathbf{c} - \mathbf{p}_k^0), \quad (14.16)$$

which leads to



**Fig. 14.3** A  $200 \times 200$  portion of the image Lena, on which we train the shrinkage curves.

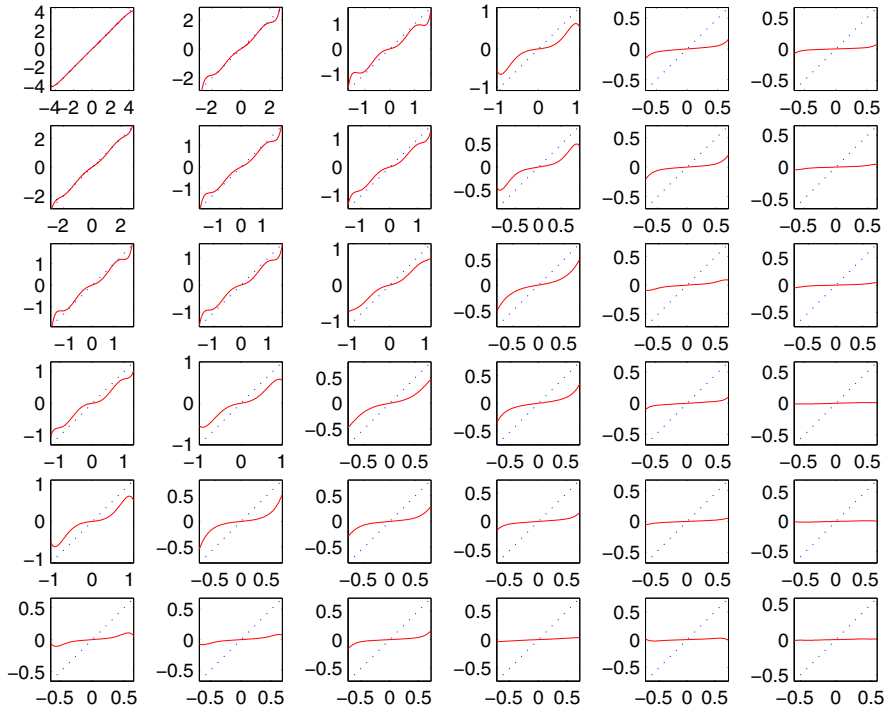
$$\mathbf{c}_{opt} = \left[ \sum_{k=1}^M \mathbf{U}_k^T \mathbf{A}^T \mathbf{A} \mathbf{U}_k \right]^{-1} \sum_{k=1}^M \mathbf{U}_k^T \mathbf{A}^T \mathbf{p}_k^0. \quad (14.17)$$

Once the curves have been trained, the denoising algorithm can be applied on new images that are suffering from an additive noise with the same power  $\sigma$ . If these images are similar to the ones the curves were trained on, the performance of the overall denoising is expected to be very good.

To demonstrate this algorithm in practice, we present the results of the following experiment. We extract all  $(200 - 5)^2$  patches of size  $6 \times 6$  ( $n = 6$ ) from a  $200 \times 200$  portion of the image Lena, as shown in Figure 14.3, and its noisy version with  $\sigma = 20$ . Those patch-pairs are used to train 36 shrinkage curves, which are to be applied on the unitary 2D-DCT coefficients of these patches. The objective is the very same one posed in Equation (14.15), and we exploit the fact that the different curves can be optimized separately and independently. Because of numerical reasons, we operate on the input patches after normalization of the form  $(\mathbf{p}_k - 127)/128$ . The obtained shrinkage curves are presented in Figure 14.4. We see that the near-DC (the top-left corner graphs) coefficients are hardly touched, while some of the high frequencies are strongly suppressed.

Once these curves have been obtained by training, we may apply them to other images, in order to remove noise of similar strength ( $\sigma = 20$ ). Figure 14.5 presents the outcome obtained on the image Barbara. Note that since the shrinkage curves have been parameterized as polynomials, their behavior for large entries may be completely distorted. As such, an input value outside the training range may be violently magnified. We treat this sensitivity by checking whether the input is within the learning data range, and if not, we do not modify it at all.

Notice that the result obtained here is approximately 1dB better than the result we have shown for the global thresholding algorithm. We may wonder how rich is the training data we use and what is the effect it has on the result obtained. Repeating the same experiment, this time training on the complete image Barbara, the result is



**Fig. 14.4** The 36 shrinkage-curves obtained by training using the penalty posed in Equation (14.15). Each of these curves corresponds to one of the unitary-2D-DCT transform coefficients.

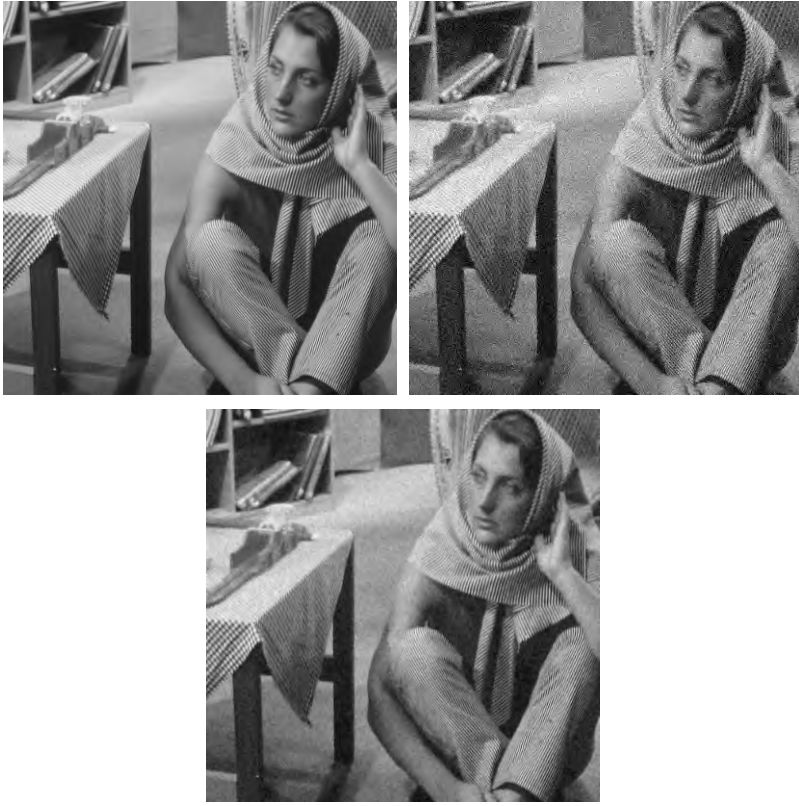
PSNR = 28.75, which is indeed better, but not by far. This validates that the training data from Lena is indeed sufficient.

Interestingly, despite the deviation from the original problem handled here, this paradigm can be used for other inverse problems, such as deblurring and deblocking of JPEG images. If we assume that the algorithm can cope with these disturbances, then this training method will be able to find the proper curves for best behavior.

One important ingredient is still missing in the above description, and this is the fact that the overall denoised image is created by averaging the overlapped patches. Thus, it would be better to optimize the shrinkage curves with respect to an error between the original complete image and its estimated version, rather than consider the intermediate errors in each patch. Formulating this is somewhat more tricky, but still manageable.

We start with the assumption that a training image  $\mathbf{y}_0$  is given to us, containing  $M$  pixels. We extract all its patches  $\{\mathbf{p}_k^0\}_{k=1}^M$  with overlaps, to serve the training procedure. The enumeration  $k$  stands for a specific position  $[i, j]$  in the image, such that there is a one-to-one correspondence between these two indexing options. For the training procedure, we also need a noisy version of these patches,  $\{\mathbf{p}_k\}_{k=1}^M$ . These are





**Fig. 14.5** The original image (top-left), the noisy one with  $\sigma = 20$  grey-values (top-right), and the denoised one obtained by the patch-based unitary-2D-DCT learned shrinkage (bottom). The input image quality is PSNR = 22.11dB, and the result shown here has PSNR = 28.36dB.

obtained by extracting all the patches from a noisy version of  $\mathbf{y}_0$ , contaminated by white additive Gaussian noise with power  $\sigma$ .

We define the operator  $\mathbf{R}_k \in \mathbb{R}^{n \times M}$  as the operator that extracts the  $k$ -th patch, by  $\mathbf{p}_k^0 = \mathbf{R}_k \mathbf{y}_0$ . The transpose of this operation puts back a patch into the  $k$ -th position in the constructed image, padded with zeros elsewhere. Thus, returning to Equation (14.8), the proper penalty should be

$$F_{global}(\mathcal{S}) = \left\| \mathbf{y}_0 - \frac{1}{n} \sum_{k=1}^M \mathbf{R}_k^T \mathbf{A} \mathcal{S} \{ \mathbf{A}^T \mathbf{R}_k \mathbf{y} \} \right\|_2^2. \quad (14.18)$$

Note that the division by  $n$  corresponds to the fact that in every pixel there are  $n$  contributions due to the full-overlaps,<sup>3</sup> and those are averaged. However, this coefficient can be absorbed into the unknown vector  $\mathbf{c}$ , and thus removed. Following the same treatment as presented above, we obtain the penalty function

$$F_{global}(\mathcal{S}) = \left\| \mathbf{y}_0 - \sum_{k=1}^M \mathbf{R}_k^T \mathbf{A} \mathbf{U}_k \mathbf{c} \right\|_2^2. \quad (14.19)$$

The optimal set of shrinkage parameters is given by

$$\mathbf{c}_{opt} = \left[ \left( \sum_{j=1}^M \mathbf{U}_j^T \mathbf{A}^T \mathbf{R}_j \right) \left( \sum_{k=1}^M \mathbf{R}_k^T \mathbf{A} \mathbf{U}_k \right) \right]^{-1} \cdot \left( \sum_{k=1}^M \mathbf{U}_k^T \mathbf{A}^T \mathbf{R}_k \right) \mathbf{y}_0. \quad (14.20)$$

A simulation of this algorithm, following closely with the description of the previous experiment, leads to better results. Training on the same database of patches as above, but this time targeting the minimization of the penalty posed in Equation (14.19), the obtained curves are very different, as seen in [Figure 14.6](#). An interesting phenomenon is obtained in some of the curves found – those show an over-shrinkage of small entries, such that these are not only nulled, but in fact multiplied by a negative factor to reverse their effect.

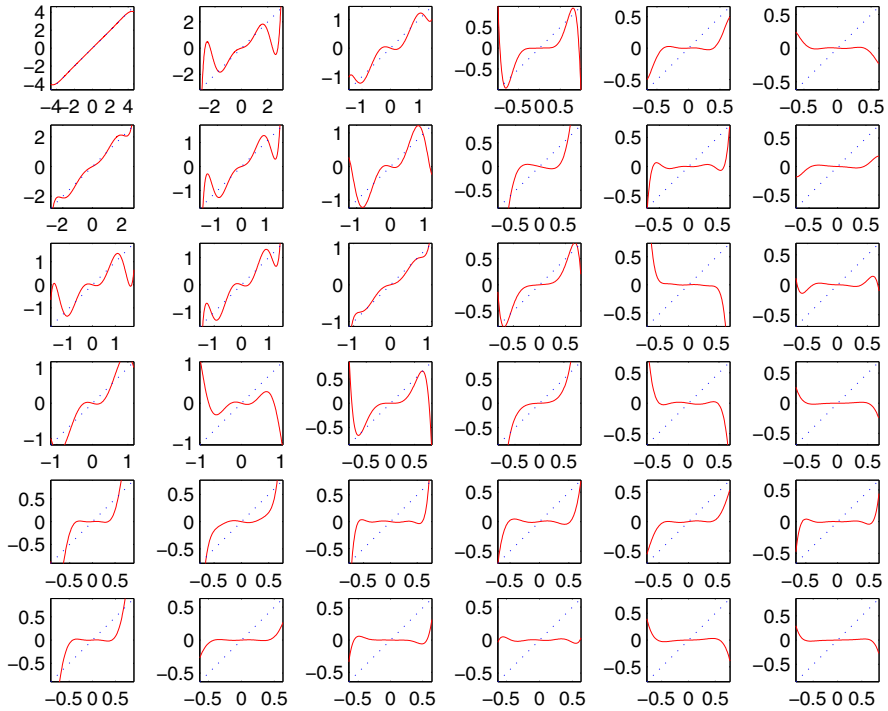
Applying these curves in the very same manner as before on the image Barbara, we obtain a further improved result with PSNR = 29.05dB (for a patch size of  $8 \times 8$  pixels, the result is PSNR = 29.41dB).<sup>4</sup> This result is shown in [Figure 14.7](#).

The appeal in the above described patch-based image denoising algorithm is the eventual simplicity with which an incoming image is treated. Another benefit this scheme puts forward is the non-iterative, local, and independent processing that the image patches undergo in the cleaning process. This implies that hardware implementation of this algorithm can benefit greatly from parallel processing.

One could think of various ways to further improve these results, and among those, it is most natural to propose to optimize the performance by modifying the dictionary  $\mathbf{A}$ . While we shall not take this path here, this brings us to the next subsection, where we introduce an image denoising algorithm which targets the design of the dictionary as its primary goal.

<sup>3</sup> And assuming a cyclic boundary condition, implying that every pixel in  $\mathbf{y}_0$  serves as a top-left corner of a corresponding patch, and when the patch falls outside the image, a periodic extension of the image is assumed.

<sup>4</sup> The result that corresponds to this experiment, as reported by Hel-Or and Shaked in their paper, is  $\approx 30$ dB. The difference should be attributed to the differences in the patch size used, the shrinkage-curve representation, and the training data.



**Fig. 14.6** The 36 shrinkage-curves obtained by training using the penalty posed in Equation (14.19), which takes into account the overlaps between the patches.

### 14.3.3 Learned Dictionary and Globalizing the Prior

The local algorithm described in the above sections (with or without the shrinkage curve learning) can be modified by using any redundant dictionary for the image patches, and by replacing the thresholding method with OMP or BP. We now turn to describe an algorithm that essentially leads to this approach in a systematic way. This follows closely with the work of Aharon and Elad.

Assuming that our knowledge on the unknown complete image  $\mathbf{y}_0$  is fully expressed in the fact that every patch in it should belong to the  $\mathcal{M}(\mathbf{A}, \epsilon, k_0)$ -*Sparse-Land* model, the MAP estimator that takes this prior into account would be

$$\begin{aligned} \{\hat{\mathbf{q}}_k\}_{k=1}^M, \hat{\mathbf{y}}\} = \arg \min_{\mathbf{z}, \{\mathbf{q}_k\}_k} \quad & \lambda \|\mathbf{z} - \mathbf{y}\|_2^2 + \\ & \sum_{k=1}^M \mu_k \|\mathbf{q}_k\|_0 + \sum_{k=1}^M \|\mathbf{A} \mathbf{q}_k - \mathbf{R}_k \mathbf{z}\|_2^2. \end{aligned} \quad (14.21)$$



**Fig. 14.7** Top-left - the original image, and Top-Right - the noisy image. The denoising results of the two experiments are given at the bottom: Left - by training on patches directly and therefore disregarding the overlaps between the patches (PSNR = 28.36dB); Right - by training the shrinkage curves on the overall denoising error, taking the overlaps into account (PSNR = 29.05dB).

In this expression the first term is the log-likelihood global force that demands the proximity between the measured image,  $\mathbf{y}$ , and its denoised (and unknown) version  $\mathbf{z}$ . The second and the third terms are the image prior that makes sure that in the constructed image,  $\mathbf{z}$ , every patch  $\mathbf{p}_k = \mathbf{R}_k \mathbf{z}$  of size  $\sqrt{n} \times \sqrt{n}$  in every location (thus the summation by  $k$ ) has a sparse representation with bounded error. As for the coefficients  $\mu_k$ , those must be location dependent, so as to comply with a set of constraints of the form  $\|\mathbf{A}\mathbf{q}_k - \mathbf{p}_k\|_2 \leq \epsilon$ . As we shall see next, these coefficients are not needed, and are replaced by known quantities.

When the underlying dictionary  $\mathbf{A}$  is assumed known, the proposed penalty term in (14.21) has two kinds of unknowns: the sparse representations  $\hat{\mathbf{q}}_k$  per each location, and the overall output image  $\mathbf{z}$ . Instead of addressing both together, we propose a sub-optimal approach of block-coordinate minimization algorithm that starts with an initialization  $\mathbf{z} = \mathbf{y}$ , and then seeks the optimal  $\hat{\mathbf{q}}_k$ . In doing so, we get a complete

decoupling of the minimization task to  $M$  smaller ones, each of the form

$$\hat{\mathbf{q}}_k = \arg \min_{\mathbf{q}} \mu_k \|\mathbf{q}\|_0 + \|\mathbf{A}\mathbf{q} - \mathbf{p}_k\|_2^2, \quad (14.22)$$

handling one image patch. Approximating the solution to this problem using the OMP is easy, gathering one atom at a time, and stopping when the error  $\|\mathbf{A}\mathbf{q} - \mathbf{p}_k\|_2^2$  goes below  $c n \sigma^2$ , for some pre-chosen value of the parameter  $c$ . This way, the choice of  $\mu_k$  has been handled implicitly. Thus, this stage works as a sliding window sparse coding stage, operated on each block of size  $\sqrt{n} \times \sqrt{n}$  at a time.

Given all  $\hat{\mathbf{q}}_k$ , we can now fix those and turn to update  $\mathbf{z}$ . Returning to Equation (14.21), we need to solve

$$\hat{\mathbf{y}} = \arg \min_{\mathbf{z}} \lambda \|\mathbf{z} - \mathbf{y}\|_2^2 + \sum_{k=1}^M \|\mathbf{A}\hat{\mathbf{q}}_k - \mathbf{R}_k \mathbf{z}\|_2^2. \quad (14.23)$$

This is a simple quadratic term that has a closed-form solution, given by

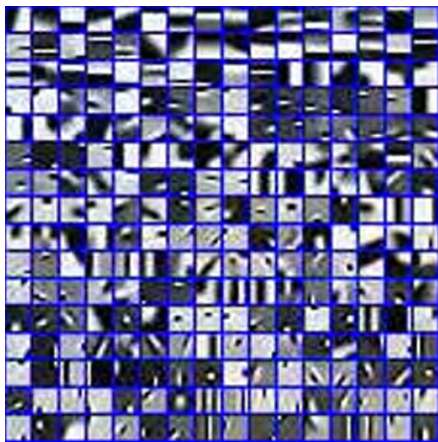
$$\hat{\mathbf{y}} = \left( \lambda \mathbf{I} + \sum_{k=1}^M \mathbf{R}_k^T \mathbf{R}_k \right)^{-1} \left( \lambda \mathbf{y} + \sum_{k=1}^M \mathbf{R}_k^T \mathbf{A} \hat{\mathbf{q}}_k \right). \quad (14.24)$$

This rather cumbersome expression may mislead, as all it suggests is that averaging of the denoised patches is to be performed, with some relaxation obtained by averaging with the original noisy image. The matrix to invert in the above expression is a diagonal one, and thus the calculation of (14.24) can be also done on a pixel-by-pixel basis, following the previously described sliding window sparse coding step.

So far we have seen that the obtained denoising algorithm calls for sparse coding of small patches, and an averaging of their outcomes. However, if minimization of (14.21) is our goal, then this process should proceed. Given the updated  $\mathbf{z}$ , we can repeat the sparse coding stage, this time working on patches from the already denoised image. Once this is done, a new averaging should be calculated, and so on and so forth. The problem with this approach is the fact that the additive noise in the treated image is no longer white nor Gaussian, and OMP may be inadequate in handling it. Furthermore, even if we intend to use the OMP, the threshold to use is unknown, as the noise has been partially removed.

The denoising method developed above assumes that  $\mathbf{A}$  is known. This may suggest that we should form a library of many clean image patches and train a dictionary off-line, using MOD or K-SVD, and then plug it into the above scheme. The results of such an experiment are reported below. A  $64 \times 256$  dictionary has been trained (using 180 iterations of the K-SVD) over 100,000 patches of size  $8 \times 8$  pixels, extracted from various images. This dictionary is shown in [Figure 14.8](#). Plugged to the above denoising scheme with  $c = 1.15$  (see above), the denoised image obtained with  $\lambda = 0.5$  is shown in [Figure 14.9](#), and its PSNR is 28.93dB.

While this approach performs reasonably well, one could do much better. A more ambitious goal is to develop a dictionary adapted to the image at hand, by learning the dictionary from the noisy image itself! This may yield sparser representa-



**Fig. 14.8** A globally trained K-SVD dictionary for general images.



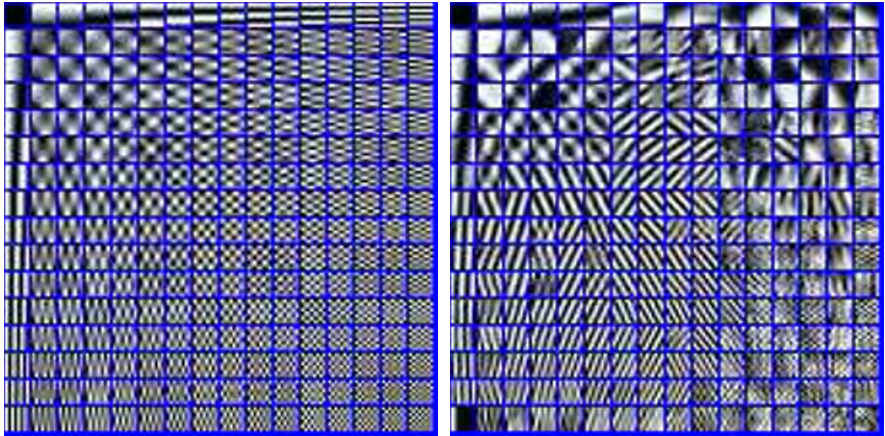
**Fig. 14.9** The denoising result when using the globally trained dictionary – PSNR = 28.93dB.

tions and a more effective denoising strategy. An interesting property of dictionary-learning algorithms is their robustness to noise, which implies that learning from the noisy image is likely to lead to a well-behaved and nearly noise-free dictionary.

An elegant way to lead to this very algorithm is to return to the MAP penalty in Equation (14.21), and consider  $\mathbf{A}$  as unknown as well. Proceeding with the Block-Coordinate-Relaxation method, we start by fixing  $\mathbf{z} = \mathbf{y}$ , and initializing  $\mathbf{A}$  somehow (e.g., redundant DCT, as described in Chapter 12).

Updating the sparse representations of the patches,  $\mathbf{q}_k$  is done as before, by a sliding windowed OMP. Once done, the dictionary can be updated using either the MOD or the K-SVD method. Iterating between these two steps leads to a dictionary-learning algorithm that effectively works on the noisy patches of the image  $\mathbf{y}$ . Once





**Fig. 14.10** The redundant DCT dictionary, which is used as initialization to the K-SVD, and the K-SVD dictionary trained on the noisy Barbara image directly, as obtained after 15 iterations.

converged (after such  $\sim 10$  rounds), the final denoising image can be computed, just as done in Equation (14.24). This way, we have merged the dictionary learning into the actual denoising process. The performance of this method is superior to the previous off-line training procedure, as we now demonstrate.

The results for this adaptive approach are summarized in Figures 14.10 and 14.11, following the same experiment outline as for the globally trained dictionary described above. Figure 14.10 shows the initial (separable) redundant 2D-DCT dictionary, and the one obtained after 15 iterations of K-SVD refinement steps. We see that the adapted dictionary contains atoms that correspond to the textured regions in the image Barbara, indicating that the dictionary adapts well to the content of interest.

Looking closely at the K-SVD dictionary in Figure 14.10 (right), it appears that some of the atoms obtained are very noisy and uninformative. At best, this may imply that these atoms are useless and are never used for representing the image patches. At worst, these atoms may become involved in these representations, and then noise is injected to the image. In any case, this suggests that we may use a lower redundancy in  $\mathbf{A}$ , while keeping the quality of the denoising results, or even improving it.

Figure 14.11 shows the denoised images obtained with these two dictionaries. Even the initial dictionary gives a very good result of PSNR = 30.01dB. This result is even better than the globally trained dictionary, because, as opposed to this globally trained dictionary, the redundant DCT contains oscillating atoms that fit well with the large textured areas in Barbara. The denoising result is further improved to PSNR = 30.98dB with the K-SVD modified dictionary, demonstrating the efficiency of this model and its adaptation to the data it operates on.



**Fig. 14.11** Top: Original (left) and noisy image (right). Bottom: Denoised results using (i) the initial separable 2D-DCT dictionary (left), with PSNR = 30.01dB; and (ii) the K-SVD dictionary after 15 iterations (right), with PSNR = 30.98dB.

As a side note, we draw the reader's attention to the following observation: When considering the initial redundant 2D-DCT dictionary, the algorithm that leads to the result in [Figure 14.11](#) (left) is very similar to the one discussed in previous sections (the Hel-Or-Shaked method, or even its predecessor by Onur Guleryuz). Indeed, the differences are in using a redundant matrix  $\mathbf{A}$  (instead of the unitary 2D-DCT), and applying OMP instead of plain thresholding. As we can see, with these two modifications, we were able to boost the denoising performance quite markedly, even without trying to shape the shrinkage curves.

We conclude this section by drawing attention to two extensions of the above K-SVD denoising algorithms, the first to color images, and the second to video. In both cases, we do not dive into the fine details of these extensions, and instead aim to present the qualitative ideas that enable them.

Handling color images can be done by converting the image from RGB to YCbCr (or other color representations that separate the intensity and the color details). In



the new domain, the K-SVD denoising can be used on the intensity image, while filtering the color layers can be done using a simpler approach – even plain smoothing. The approach taken by Mairal, Elad, and Sapiro (2008) is different and better: The very same K-SVD denoising algorithm is employed directly on the color image, by operating on 3D patches that contain the three color layers (R, G, and B). This way, the learned dictionary contains color atoms, and all the treatment as described above remains the same. A small modification, however, is introduced into the  $\ell_2$  distance measure between patches used in the OMP, in order to avoid color artifacts. Figure 14.12 shows the results obtained from such an algorithm, operating on patches of size  $6 \times 6 \times 3$ .

Turning to video, which is a sequence of frames, the above K-SVD denoising algorithm can be applied to each frame independently. However, one could do far better if adjacent frames are processed jointly. One way to incorporate information between frames is to consider the complete spatio-temporal volume of the video as a 3D image, and operate on it this way. This implies that the K-SVD algorithm processes small 3D patches, and similarly, the atoms in the trained dictionary are 3D patches as well. A further improvement of this scheme can be proposed, where the trained dictionary is adapted to each frame in the sequence. In this case, the denoising is performed on each frame sequentially (thus giving an on-line filtering algorithm, rather than a batch denoising mode of operation), and the dictionary is trained on 3D patches extracted from the current frame and its neighbors. While this may seem like a much more demanding algorithm, the dictionary can be propagated from the previous frame and updated with much fewer iterations to better serve the current frame. The results of such an algorithm (operating on patches of size  $5 \times 5 \times 5$  pixels) are shown in Figure 14.13 for the Garden sequence. We should note that the single-frame K-SVD denoising on this sequence leads to an inferior outcome with a difference of 1.8dB. More details on this algorithm and its performance are found in the paper by Protter and Elad (2009).

### 14.3.4 The Non-Local-Means Algorithm

The above K-SVD image denoising algorithm relies on a learning procedure that works directly on the corrupted image. Since there are  $M$  patches in the image, and as we assume that  $m \ll M$ , this implies that in this training process the noise is damped and hardly effects the obtained dictionary.

We could take this adaptivity idea further, and propose an adapted dictionary per each area in the image. Segmenting the image into parts, and training a different dictionary per each, could lead to improved results. Indeed, in such an approach we could use reduced redundancy, as it is no longer required. Taken to the extreme, we could train a different dictionary per every pixel, using the patches in its neighborhood. However, this clearly implies a very high computational load. Therefore, we shall propose a simpler approach with a similar effect.



**Fig. 14.12** Top: The original (left) and the noisy images (right) – PSNR = 20.17dB. Bottom: Color K-SVD denoised result(left) – PSNR = 30.83dB, and a zoomed-in portion of the result (right).



**Fig. 14.13** Video denoising using the K-SVD algorithm. Left: The original 10th and 20th frames, Middle: These noisy frames ( $\sigma = 50$ , PSNR = 14.88dB), Right: Denoised results (PSNR = 22.80dB).

Assume that per every pixel  $k$  we accumulate the surrounding patches in locations  $k \in \mathcal{N}(k)$ , and form a dictionary from them (without any training). The size of this dictionary is  $n \times m$ , where  $m = |\mathcal{N}(k)|$ . We denote this dictionary as  $\mathbf{A}_k$ , and assume that the center patch for which this dictionary has been constructed is  $\mathbf{p}_k = \mathbf{R}_k \mathbf{y}$ . We shall assume a sparse representation of  $\mathbf{p}_k$  with respect to the dictionary  $\mathbf{A}_k$ . However, in order to improve the denoising performance, we could approximate the MMSE estimator by proposing several such representations and averaging them with proper weights. Those weights would be exactly those developed in Chapter 11. This leads to a variant of the Non-Local-Means (NLM) algorithm. Indeed, it aligns perfectly with the NLM if the above-mentioned sparse representations contain only one atom, forced to have a unit coefficient, and if after the averaging we adopt only the center pixel and throw the rest.

The NLM algorithm was introduced by Coll, Buades, and Morel, as an extension of the bilateral-filter, and the above description is far from the way it was conceived. The NLM proposes a denoising of an image  $\mathbf{y}$  by performing a local weighted average, of the form

$$\hat{y}_{ij} = \frac{\sum_{[k,l] \in \mathcal{N}(i,j)} w_{ij}[k,l] y[i-k, j-l]}{\sum_{[k,l] \in \mathcal{N}(i,j)} w_{ij}[k,l]}. \quad (14.25)$$

The key for the success of the NLM is the choice of the weights. Those are computed by

$$w_{ij}[k,l] = \exp \left\{ -\frac{\|\mathbf{R}_{i,j} \mathbf{y} - \mathbf{R}_{i-k, j-l} \mathbf{y}\|_2^2}{2T} \right\}. \quad (14.26)$$



**Fig. 14.14** The noisy image and its denoised result using the NLM filter, with PSNR = 29.59dB.

Thus, the NLM amounts to a weighted average of surrounding pixels, with weights that are built inversely proportional to the  $\ell_2$  distance between the patches centered around the two pixels to be merged. This process can be applied iteratively, improving the above result.

The denoising result of the image *Barbara* (additive noise with  $\sigma = 20$ , as in the rest of this chapter) using the NLM is given in [Figure 14.14](#). The patch-size used is  $7 \times 7$  pixels, the neighborhood size is  $17 \times 17$  pixels, and  $T = 200$ . The result obtained gives a PSNR of 29.59dB. While this is not necessarily the best possible set of parameters for the NLM, it is not far from extracting the most from this filter.

The above interpretation of the NLM as an extreme case of sparse representation and MMSE estimation gives rise to many possibilities and improvements. Those include options such as improving the local dictionary, allowing more atoms in every representation, using the complete patch and not just its center, and more.

Before we conclude the discussion on the NLM, we should note that the interpretation given here for this algorithm is not the only one possible. Ties to other, more classical, techniques in image processing are also possible. One connection which we mentioned above exists between the NLM and the bilateral-filter. The bilateral-filter can be regarded as a special case of the NLM when using patches of size  $1 \times 1$  pixel. This filter also adds an exponential damping effect to the weights as a function of the distance between the center pixel and the one averaged with it. This in turn is closely related to the Beltrami-flow image denoising algorithm, as proposed by Kimmel, Malladi, and Sochen. In fact, as Spira et. al. show, the bilateral-filter is the effective short-time kernel of the Beltrami algorithm. Similarly, one could tie the NLM directly to a variant of the Beltrami framework, which performs the diffusion operation on the image in a different feature-space, so as to accommodate the patch-processing that is used.





**Fig. 14.15** The noisy image and its denoised result using the BM3D filter, with PSNR = 31.78dB.

### 14.3.5 3D-DCT Shrinkage: BM3D Denoising

As a last algorithm to be presented within this family of local patch-based processing that use sparse representations, we introduce a recently proposed method by Dabov, Foi, Katkovnik, and Egiazarian – the Block-Matching 3D (BM3D) algorithm. This algorithm shares some similarities with the NLM and the other algorithms mentioned above. We shall not dive into the very fine details of this method, and instead concentrate on its main theme. We note that to date, this is among the best performing algorithms for image denoising, although the others mentioned are not far behind. Figure 14.15 shows the denoising result of *Barbara*, removing an additive noise with  $\sigma = 20$ , and as can be seen, this is the best result shown in this chapter, with PSNR = 31.78dB.

The BM3D algorithm operates on every pixel from  $\mathbf{y}$ , and performs the same denoising procedure explained below: Consider the  $k$ -th pixel (which corresponds to a specific location  $[i, j]$ ), and extract the patch  $\mathbf{p}_k$  of size  $\sqrt{n} \times \sqrt{n}$  centered around it. The denoising process starts by finding a set  $\mathcal{S}_k$  of similar patches throughout the image. Similarity is measured by the  $\ell_2$ -norm (a block-matching process). In order to reduce the bad effect of noise, the distance computation is applied on denoised versions of the patches, where denoising is done by the same thresholding algorithm, as presented in an earlier section of this chapter.

Once found, these  $|\mathcal{S}_k|$  patches are stacked together to form a cube of size  $\sqrt{n} \times \sqrt{n} \times |\mathcal{S}_k|$ . Joint denoising on this volume is done by using the very same thresholding algorithm, only this time with the 3D-DCT transform. The idea is that the high correlation between the patches and the spatial redundancy should both lead to strong sparsity of the 3D-DCT coefficients. After this (second) thresholding, an inverse 3D-DCT is applied and we obtain a cleaned version of this set of patches.

The cleaned patches are returned to their original locations in a destination canvas, and averaged with other patches that overlap them. This process is repeated for every pixel, such that every patch may be cleaned several times, and with several contexts. The averaging between the overlapped patches is done using weights that are inversely proportional to the number of non-zeros that each patch got in the thresholding stage. This way, highly sparse patches get higher priority in the overall solution.

Although different from the aforementioned algorithms, there are some clear ties and similarities. Processing patches, the thresholding algorithm used, block-matching as in the NLM, the overlaps averaging, and more, indicate that the BM3D indeed belongs to the same family of denoising techniques. Furthermore, based on what we have already seen, it is natural to propose improvements for this method, such as shrinkage curve learning, dictionary-learning, and more.

As said above, the BM3D is among the best performing algorithms for image denoising. However, with the extensive work on image denoising, it is very likely that this claim will become wrong in a short while. Indeed, a recent work by Mairal et. al. proposes a variant of the K-SVD denoising algorithm, which already outperforms the BM3D. We mention this work because it poses an interesting fusion between the K-SVD and the BM3D algorithms.

Embarking from the K-SVD denoising, Mairal et. al. suggest the following elementary (but crucial) modification: instead of applying sparse coding on each patch independently, follow the BM3D line-of-reasoning by seeking similar patches, forming the set  $\mathcal{S}_k$ , and then apply a joint-sparse coding to these. The joint-sparsity process forces the different patches to decompose using the same subset of atoms. Once done, each patch is returned to its location, and the averaging resumes as in BM3D. Two sources of strength in this method boost its performance: (i) the joint-sparsity poses a further constraint on the atom-decomposition; and (ii) the extra averaging that is caused by the fact that each patch may serve in several such sets  $\mathcal{S}_k$ .

## 14.4 SURE for Automatic Parameter Setting

Before we conclude this chapter, we turn to discuss a different topic, which is of an utmost importance to all the above algorithms – the need to tune various parameters in an attempt to get better results. Could there be an automatic method for this task? In this section we describe the Stein-Unbiased-Risk-Estimator (SURE), which is indeed an elegant way to answer the above question positively for some problems.

### 14.4.1 Development of the SURE

In many algorithms in image processing there is a set of unknown parameters that needs tuning. As an example that shall accompany us in this section, consider the problem posed in Equation (14.4), where the parameter  $T$  is unknown. Clearly, this parameter can be chosen such that  $\|\hat{\mathbf{y}} - \mathbf{y}\|_2 \approx N\sigma^2$ , but this does not necessarily lead to the lowest possible Mean-squared-Error (MSE). Given a noisy input image, could we propose a method that tunes  $T$  automatically so as to optimize the MSE with respect to the *unknown* ideal image, or get close to it? At first glance, this seems to be an impossible task, since we do not have access to the true image  $\mathbf{y}_0$ . However, as we show next, there is a way which comes close to answering positively this question.

For this task, we introduce the Stein Unbiased Risk Estimator (SURE). It provides an unbiased estimation formula of the MSE, given an estimator (i.e., denoising algorithm)  $\hat{\mathbf{y}} = h(\mathbf{y}, \theta)$  and the measurements  $\mathbf{y}$ . The vector  $\theta$  stands for the various parameters we need to set when applying this denoising method.

We operate within the settings of Equation (14.1), where  $\mathbf{v}$  is an iid white Gaussian noise with known variance  $\sigma^2$ . The MSE estimator proposed by Stein, denoted by  $\eta(\hat{\mathbf{y}}, \mathbf{y}) = \eta(h(\mathbf{y}, \theta), \mathbf{y})$ , is an unbiased estimator of the MSE, implying that

$$E[\eta(\hat{\mathbf{y}}, \mathbf{y})] = E[\|\hat{\mathbf{y}} - \mathbf{y}_0\|_2^2]. \quad (14.27)$$

Thus, if we have an access to the function  $\eta(\hat{\mathbf{y}}, \mathbf{y})$  that estimates the MSE, while being dependent on  $\theta$ , we can choose the value of  $\theta$  so as to minimize this function, and aim this way at minimizing the true MSE, as a consequence.

To derive the SURE we start from the MSE expression, remembering that our goal is minimizing it and not estimating its exact value. Thus we can ignore constants that are independent of our estimator  $\hat{\mathbf{y}} = h(\mathbf{y}, \theta)$ . Opening the MSE expression we obtain

$$\begin{aligned} E(\|\hat{\mathbf{y}} - \mathbf{y}_0\|_2^2) &= E(\|h(\mathbf{y}, \theta) - \mathbf{y}_0\|_2^2) \\ &= E(\|h(\mathbf{y}, \theta)\|_2^2) - 2E[h(\mathbf{y}, \theta)^T \mathbf{y}_0] + \|\mathbf{y}_0\|_2^2 \\ &= E(\|h(\mathbf{y}, \theta)\|_2^2) - 2E(h(\mathbf{y}, \theta)^T \mathbf{y}_0) + \text{const.} \end{aligned} \quad (14.28)$$

Only the second element in the equation is a function of the unknown  $\mathbf{y}_0$ , and we proceed by further treating it. Recalling that  $\mathbf{y}_0 = \mathbf{y} - \mathbf{v}$  we get

$$E(h(\mathbf{y}, \theta)^T \mathbf{y}_0) = E(h(\mathbf{y}, \theta)^T \mathbf{y}) - E(h(\mathbf{y}, \theta)^T \mathbf{v}). \quad (14.29)$$

Denoting the  $i$ -th entry in the vector  $h(\mathbf{y}, \theta)$  by  $h_i(\mathbf{y}, \theta)$ , and the  $i$ -th entry in the noise vector as  $v_i$ , this reads

$$E(h(\mathbf{y}, \theta)^T \mathbf{y}_0) = E(h(\mathbf{y}, \theta)^T \mathbf{y}) - \sum_{i=1}^N E(h_i(\mathbf{y}, \theta)v_i) \quad (14.30)$$

$$\begin{aligned}
&= E \left( h(\mathbf{y}, \theta)^T \mathbf{y} \right) \\
&\quad - \sum_{i=1}^N \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} h_i(\mathbf{y}, \theta) v_i \exp \left\{ -\frac{v_i^2}{2\sigma^2} \right\} dv_i.
\end{aligned}$$

We now turn to handle the last term with the integration in the above expression. Note that the expression  $h_i(\mathbf{y}, \theta)$  is a function of  $v_i$  as well, as  $y_i = y_i^0 + v_i$ , and thus it remains inside the integration. We shall assume hereafter that  $h_i(\mathbf{y}, \theta)$  is differentiable with respect to  $\mathbf{y}$ . Using integration by parts,  $\int u(x)v'(x)dx = u(x)v(x) - \int u'(x)v(x)dx$ , we obtain

$$\begin{aligned}
E(h_i(\mathbf{y}, \theta)v_i) &= \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\infty} h_i(\mathbf{y}, \theta)v_i \exp \left\{ -\frac{v_i^2}{2\sigma^2} \right\} dv_i \quad (14.31) \\
&= -\frac{\sigma^2}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\infty} h_i(\mathbf{y}, \theta) \left[ \frac{d}{dv_i} \exp \left\{ -\frac{v_i^2}{2\sigma^2} \right\} \right] dv_i \\
&= -\frac{\sigma}{\sqrt{2\pi}} \left[ \left[ h_i(\mathbf{y}, \theta) \exp \left\{ -\frac{v_i^2}{2\sigma^2} \right\} \right]_{-\infty}^{\infty} \right. \\
&\quad \left. - \int_{-\infty}^{\infty} \frac{d}{dv_i} h_i(\mathbf{y}, \theta) \exp \left\{ -\frac{v_i^2}{2\sigma^2} \right\} dv_i \right].
\end{aligned}$$

Assuming that  $\text{abs}(h(\mathbf{y}, \theta))$  is bounded, the first term in Equation (14.31) vanishes, and we get

$$\begin{aligned}
E(h_i(\mathbf{y}, \theta)v_i) &= \frac{\sigma}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \frac{dh_i(\mathbf{y}, \theta)}{dy_i} \exp \left\{ -\frac{v_i^2}{2\sigma^2} \right\} dv_i \quad (14.32) \\
&= \sigma^2 E \left( \frac{dh_i(\mathbf{y}, \theta)}{dy_i} \right).
\end{aligned}$$

Here we have used the fact that  $dy_i/dv_i = 1$ , which implies that  $dh_i(\mathbf{y}, \theta)/dv_i = dh_i(\mathbf{y}, \theta)/dy_i \cdot dy_i/dv_i = dh_i(\mathbf{y}, \theta)/dy_i$ .

Gathering this result and the ones in Equations (14.28) and (14.30), we finally get that

$$\begin{aligned}
MSE &= E \left( \|\mathbf{h}(\mathbf{y}, \theta) - \mathbf{y}\|_2^2 \right) \quad (14.33) \\
&= \text{const} + E \left[ \|\mathbf{h}(\mathbf{y}, \theta)\|_2^2 \right] - 2E \left( \mathbf{h}(\mathbf{y}, \theta)^T \mathbf{y} \right) + 2\sigma^2 \sum_{i=1}^N E \left( \frac{dh_i(\mathbf{y}, \theta)}{dy_i} \right) \\
&= \text{const} + E \left[ \|\mathbf{h}(\mathbf{y}, \theta)\|_2^2 - 2\mathbf{h}(\mathbf{y}, \theta)^T \mathbf{y} + 2\sigma^2 \nabla \cdot \mathbf{h}(\mathbf{y}, \theta) \right].
\end{aligned}$$

By removing the expectation we obtain an expression that serves as our unbiased estimation of the MSE, and thus the SURE formula (that estimates the MSE up to a constant) is given by

$$\eta(h(\mathbf{y}, \theta), \mathbf{y}) = \|\mathbf{h}(\mathbf{y}, \theta)\|_2^2 - 2\mathbf{h}(\mathbf{y}, \theta)^T \mathbf{y} + 2\sigma^2 \nabla \cdot \mathbf{h}(\mathbf{y}, \theta). \quad (14.34)$$



Note that in the above development we have used the fact that the noise is iid and Gaussian, and thus the formula obtained is limited to this case. An extension of SURE has been derived by Y.C. Eldar for the more general scenario  $\mathbf{y} = \mathbf{H}\mathbf{y}_0 + \mathbf{v}$ , where  $\mathbf{H}$  is an arbitrary linear operator, and  $\mathbf{v}$  is a noise from the exponential family (for which Gaussian noise is just one special case).

#### 14.4.2 Demonstrating SURE to Global-Thresholding

In order to demonstrate the use of SURE for parameter setting, we consider the global thresholding algorithm for the denoising of an image, as described and experimented in Section 14.2.1. The denoising algorithm is given by (see Equation (14.4))

$$h(\mathbf{y}, T) = \mathbf{A}_2 \mathcal{S}_T(\mathbf{A}_1^T \mathbf{y}), \quad (14.35)$$

where  $\theta$  governs the shrinkage-curve, and we use the notation  $\mathbf{A}_1 = \mathbf{A}\mathbf{W}^{-1}$  and  $\mathbf{A}_2 = \mathbf{A}\mathbf{W}$ . This denoising algorithm is governed by the single parameter  $T$ , which we shall set using SURE.

For the SURE formula in Equation (14.34), we need to calculate the divergence of this estimator. We will do this using the equality  $\nabla \cdot h(\mathbf{y}, \theta) = \text{tr}(dh(\mathbf{y}, \theta)/d\mathbf{y})$ . The derivative (given as the Jacobian matrix) of  $h(\mathbf{y}, \theta)$  with respect to  $\mathbf{y}$  is given by

$$\frac{dh(\mathbf{y}, T)}{d\mathbf{y}} = \mathbf{A}_2 \mathcal{S}'_T(\mathbf{A}_1^T \mathbf{y}) \mathbf{A}_1^T. \quad (14.36)$$

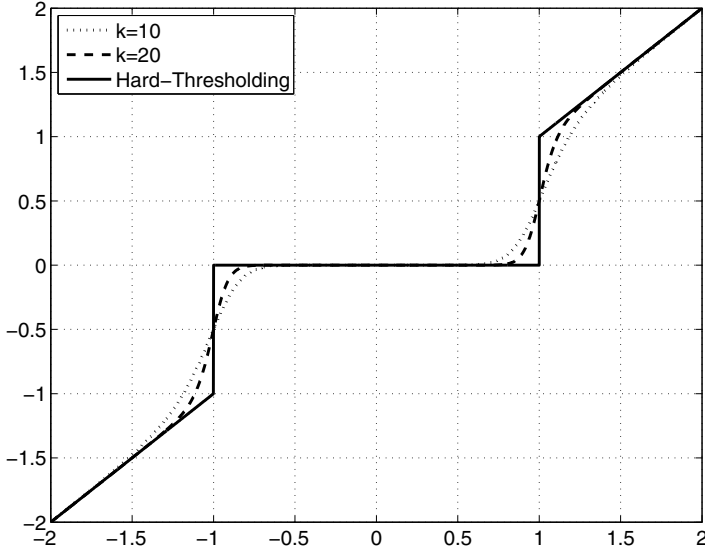
The expression  $\mathcal{S}'_T(\mathbf{A}_1^T \mathbf{y})$  stands for a diagonal matrix. Its main diagonal contains an element-wise application of the derivative of the shrinkage-curve on the vector  $\mathbf{A}_1^T \mathbf{y}$ . Here we have conveniently assumed that the shrinkage-curve is differentiable. In order to allow this, we use a smoothed version of the hard-thresholding curve, given by

$$\begin{aligned} \mathcal{S}_T(z) &= \frac{z^{k+1}}{z^k + T^k} = z \cdot \frac{\left(\frac{z}{T}\right)^k}{\left(\frac{z}{T}\right)^k + 1} \\ \mathcal{S}'_T(z) &= \frac{z^{2k} + (k+1)(zT)^k}{(z^k + T^k)^2} = \frac{\left(\frac{z}{T}\right)^{2k} + (k+1)\left(\frac{z}{T}\right)^k}{\left(\left(\frac{z}{T}\right)^k + 1\right)^2}. \end{aligned}$$

for an even and high value of  $k$ . This function is shown in [Figure 14.16](#).

Plugging these into the SURE formula in Equation (14.34), we get the function

$$\begin{aligned} \eta(h(\mathbf{y}, T), \mathbf{y}) &= \|\mathbf{A}_2 \mathcal{S}_T(\mathbf{A}_1^T \mathbf{y})\|_2^2 - 2\mathcal{S}_T(\mathbf{A}_1^T \mathbf{y})^T \mathbf{A}_2^T \mathbf{y} + \\ &\quad + 2\sigma^2 \text{tr}(\mathbf{A}_2 \mathcal{S}'_T(\mathbf{A}_1^T \mathbf{y}) \mathbf{A}_1^T). \end{aligned} \quad (14.37)$$



**Fig. 14.16** The smoothed hard-thresholding curve used within the SURE.

In order to find the best  $T$  (which minimizes the above expression), we perform an exhaustive search over a range of possible values, and choose the best one. More efficient methods could be envisioned, such as the golden-section search (since the MSE behaves as a unimodal function), but this is not crucial for the exposition given here.

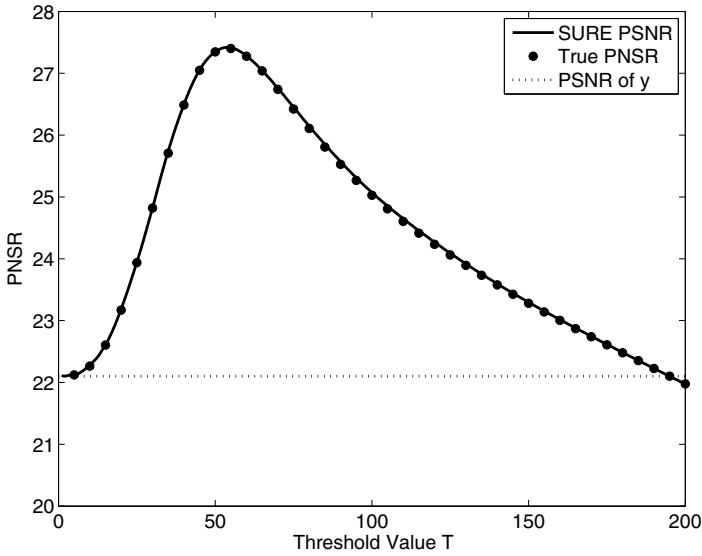
As for the complexity of computing this expression for a single value of  $T$ , it starts with the denoising algorithm, first computing  $\mathbf{u} = \mathbf{A}_1^T \mathbf{y}$ , then  $\mathbf{r} = \mathcal{S}_T(\mathbf{u})$ , followed by  $\hat{\mathbf{y}} = \mathbf{A}_2 \mathbf{r}$ . The two first expressions in Equation (14.37) require an additional  $m$  operations each ( $m$  is the number of atoms in  $\mathbf{A}$ ) to complete.

As for the third term, we should start by computing  $\mathbf{t} = \mathcal{S}'_T(\mathbf{u})$ , again requiring  $m$  operations. Using the relations  $\mathbf{A}_1 = \mathbf{A}\mathbf{W}^{-1}$  and  $\mathbf{A}_2 = \mathbf{A}\mathbf{W}$ , we can simplify the third expression to

$$\begin{aligned} \text{tr}(\mathbf{A}_2 \mathcal{S}'_T(\mathbf{A}_1^T \mathbf{y}) \mathbf{A}_1^T) &= \text{tr}(\mathcal{S}'_T(\mathbf{A}_1^T \mathbf{y}) \mathbf{A}_1^T \mathbf{A}_2) \\ &= \text{tr}(\mathcal{S}'_T(\mathbf{A}_1^T \mathbf{y}) \mathbf{W}^{-1} \mathbf{A}^T \mathbf{A} \mathbf{W}) \\ &= \text{tr}(\mathbf{W} \mathcal{S}'_T(\mathbf{A}_1^T \mathbf{y}) \mathbf{W}^{-1} \mathbf{A}^T \mathbf{A}). \end{aligned} \quad (14.38)$$

Since  $\mathcal{S}'_T(\mathbf{u})$  is a diagonal matrix, the left and right multiplications by  $\mathbf{W}$  and  $\mathbf{W}^{-1}$  cancel out, leading to

$$\text{tr}(\mathbf{A}_2 \mathcal{S}'_T(\mathbf{A}_1^T \mathbf{y}) \mathbf{A}_1^T) = \text{tr}(\mathcal{S}'_T(\mathbf{A}_1^T \mathbf{y}) \mathbf{W}^2). \quad (14.39)$$



**Fig. 14.17** The PSNR as a function of  $T$  using both the SURE formula and a direct computation using the ground-truth.

This requires a computation of the atoms' norm, and their element-wise multiplication by the elements of  $\mathcal{S}'_T(\mathbf{u})$ . Thus, having these norms pre-computed, another set of  $m$  operations builds this expression, and an overall of  $4m$  operations leads to the evaluation of the complete SURE estimate for a specific value of  $T$ .

In [Figure 14.17](#) we show the PSNR as a function of the parameter  $T$ , very similarly to the presentation in [Figure 14.1](#). The dotted curve is obtained using a direct computation of the PSNR using the clean image. The SURE expression in Equation (14.37) leads to the continuous curve, which clearly fits very well with the true PSNR quality. Note that for these two curves to fit so well, we have to add a constant to the SURE (in the MSE domain), and this was done as part of building this figure. It is clear that in this experiment, choosing the peak in the SURE curve leads to the best denoising result, as if we had access to the ground-truth. This is not always the case with SURE, and especially not when the number of parameters to be set is large.

## 14.5 Summary

The topic of image denoising is very hot, active, and timely, with numerous publications. The query

Topic=((image and (denoise or denoising)))

or  
(image and noise and (removal or filter or clean)))

in ISI leads to 7,520 papers, out of which nearly half are journal papers. An analysis of the publication year of these papers reveals that the interest in this problem and its solutions is only growing.

In this chapter we merely scratched the surface of this vast activity, by concentrating on *Sparse-Land*-related denoising algorithms. All these assume an additive white Gaussian noise, and consider its filtering from still B/W images. While the algorithms presented here are few, they are also among the best at their task, which testifies to the importance and the success of sparse and redundant representation modeling.

Just as we explore noise filtering from B/W still images, we may also be interested in color images, or in video. The *Sparse-Land* model has been brought to these problems as well, by generalizing some of the above algorithms. We brought in this chapter representative examples for each of these extensions. We choose to omit the details, and refer the interested readers to recent papers on these subjects.

This chapter offers a series of denoising methods, with a natural hierarchy in their output quality. Table 14.1 presents this list of methods, ordering them by the resulting PSNR on the Barbara denoising experiment. Note that this ordering is not completely correct, as each method may give a different result by modifying its parameters, or when operating on other images.

Looking at the PSNR alone gives a partial picture on this comparison, because each algorithm requires a different effort in terms of computations and memory. Indeed, the NLM, the BM3D, and the K-SVD denoising all require much more effort, compared to the other algorithms. Speedup techniques for each of these algorithms exist, and the struggle to propose a combination of fast and effective denoising algorithms is still very much alive.

One last comment: Throughout this chapter we have been using the PSNR as a measure of quality. While this measure is comfortable and easy to use, it does not always reflect the true quality of the resulting images. An alternative measure, called *Structural Similarity* (SSIM), has been drawing attention in recent years, as a better quality-assessment method. Table 14.1 presents the SSIM measures for the various results as well. SSIM computes a value in the range [0, 1], such that 1 stands for perfect quality. As can be seen, the ordering of the results is not affected much by the change of the quality measure. This could be explained by the fact that in denoising (and especially in high-quality algorithms, as those explored here) we do not get artifacts of the kind handled by SSIM.

## Further Reading

1. M. Aharon, M. Elad, and A.M. Bruckstein, K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation, *IEEE Trans. on Signal Processing*, 54(11):4311–4322, November 2006.

Algorithm	PSNR [dB]	SSIM
The noisy image	22.11	0.4771
Global thresholding with redundant Haar dictionary	27.33	0.7842
Patch-based shrinkage-curve training (disregard overlaps)	28.36	0.7881
Patch-based OMP denoising with a fixed dictionary	28.93	0.8465
Patch-based shrinkage-curve training (exploit overlaps)	29.05	0.8375
Patch-based Non-Local-Means filtering	29.59	0.8539
Patch-based OMP denoising with redundant DCT	30.01	0.8665
Patch-based denoising with an adaptive K-SVD dictionary	30.98	0.8837
Patch-based Block-Matching and 3D shrinkage (BM3D)	31.78	0.9050

**Table 14.1** The various image denoising algorithms presented in this chapter, and the quality of their results for denoising of *Barbara* with  $\sigma = 20$ , measured by PSNR and SSIM.

2. A. Benazza-Benyahia and J.-C. Pesquet, Building robust wavelet estimators for multicomponent images using Steins principle, *IEEE Trans. on Image Processing*, 14(11):1814–1830, November 2005.
3. A. Buades, B. Coll, and J.M. Morel, A review of image denoising algorithms, with a new one, *Multiscale Modeling and Simulation*, 4(2):490–530, 2005.
4. T. Blu, F. Luisier, The SURE-LET approach to image denoising, *IEEE Trans. on Image Processing*, 16(11):2778–2786, November 2007.
5. E.J. Candès and D.L. Donoho, Recovering edges in ill-posed inverse problems: optimality of curvelet frames, *Annals of Statistics*, 30(3):784–842, 2000.
6. E.J. Candès and D.L. Donoho, New tight frames of curvelets and optimal representations of objects with piecewise- $C^2$  singularities, *Comm. Pure Appl. Math.*, 57:219–266, 2002.
7. S.G. Chang, B. Yu, and M. Vetterli, Adaptive wavelet thresholding for image denoising and compression, *IEEE Trans. on Image Processing*, 9(9):1532–1546, 2000.
8. S.G. Chang, B. Yu, and M. Vetterli, Wavelet thresholding for multiple noisy image copies, *IEEE Trans. on Image Processing*, 9(9):1631–1635, 2000.
9. S.G. Chang, B. Yu, and M. Vetterli, Spatially adaptive wavelet thresholding with context modeling for image denoising, *IEEE Trans. on Image Processing*, 9(9):1522–1530, 2000.
10. R. Coifman and D.L. Donoho, Translation-invariant denoising. *Wavelets and Statistics, Lecture Notes in Statistics*, 103:120–150, 1995.
11. K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, Image denoising by sparse 3D transformdomain collaborative filtering, *IEEE Trans. on Image Processing*, 16 (2007), pp. 2080–2095.
12. M.N. Do and M. Vetterli, The finite ridgelet transform for image representation, *IEEE Trans. on Image Processing*, 12(1):16–28, 2003.
13. M.N. Do and M. Vetterli, Framing pyramids, *IEEE Trans. on Signal Processing*, 51(9):2329–2342, 2003.

14. M.N. Do and M. Vetterli, The contourlet transform: an efficient directional multiresolution image representation, *IEEE Trans. Image on Image Processing*, 14(12):2091–2106, 2005.
15. D.L. Donoho, De-noising by soft thresholding, *IEEE Trans. on Information Theory*, 41(3):613–627, 1995.
16. D.L. Donoho and I.M. Johnstone, Ideal denoising in an orthonormal basis chosen from a library of bases, *Comptes Rendus de l'Academie des Sciences, Series A*, 319:1317–1322, 1994.
17. D.L. Donoho and I.M. Johnstone, Ideal spatial adaptation by wavelet shrinkage, *Biometrika*, 81(3):425–455, 1994.
18. D.L. Donoho and I. M. Johnstone, Adapting to unknown smoothness via wavelet shrinkage, *J. Amer. Statist. Assoc.*, 90(432):1200–1224, December 1995.
19. D.L. Donoho, I.M. Johnstone, G. Kerkycharian, and D. Picard, Wavelet shrinkage - asymptopia, *Journal Of The Royal Statistical Society Series B - Methodological*, 57(2):301–337, 1995.
20. D.L. Donoho and I.M. Johnstone, Minimax estimation via wavelet shrinkage, *Annals of Statistics*, 26(3):879–921, 1998.
21. M. Elad and M. Aharon, Image denoising via learned dictionaries and sparse representation, *International Conference on Computer Vision and pattern Recognition*, New-York, June 17–22, 2006.
22. M. Elad and M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries, *IEEE Trans. on Image Processing* 15(12):3736–3745, December 2006.
23. M. Elad, B. Matalon, and M. Zibulevsky, Image denoising with shrinkage and redundant representations, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, NY, June 17–22, 2006.
24. Y. Eldar, Generalized SURE for exponential families: applications to regularization, *IEEE Trans. on Signal Processing*, 57(2):471–481, February 2009.
25. R. Eslami and H. Radha, The contourlet transform for image de-noising using cycle spinning, in *Proceedings of Asilomar Conference on Signals, Systems, and Computers*, pp. 1982–1986, November 2003.
26. R. Eslami and H. Radha, Translation-invariant contourlet transform and its application to image denoising, *IEEE Trans. on Image Processing*, 15(11):3362–3374, November 2006.
27. O.G. Guleryuz, Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising – Part I: Theory, *IEEE Trans. on Image Processing*, 15(3):539–554, 2006.
28. O.G. Guleryuz, Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising – Part II: Adaptive algorithms, *IEEE Trans. on Image Processing*, 15(3):555–571, 2006.
29. O.G. Guleryuz, Weighted averaging for denoising with overcomplete dictionaries, *IEEE Trans. on Image Processing*, 16:30203034, 2007.
30. M. Lang, H. Guo, and J.E. Odegard, Noise reduction using undecimated discrete wavelet transform, *IEEE on Signal Processing Letters*, 3(1):10–12, 1996.

31. Y. Hel-Or and D. Shaked, A Discriminative approach for Wavelet Shrinkage Denoising, *IEEE Trans. on Image Processing*, 17(4):443–457, April 2008.
32. R. Kimmel, R. Malladi, and N. Sochen. Images as Embedded Maps and Minimal Surfaces: Movies, Color, Texture, and Volumetric Medical Images. *International Journal of Computer Vision*, 39(2):111–129, September 2000.
33. J. Mairal, F. Bach, J. Ponce, G. Sapiro and A. Zisserman, Non-local sparse models for image restoration, *International Conference on Computer Vision (ICCV)*, Tokyo, Japan, 2009.
34. J. Mairal, M. Elad, and G. Sapiro, Sparse representation for color image restoration, *IEEE Trans. on Image Processing*, 17(1):53–69, January 2008.
35. J. Mairal, G. Sapiro, and M. Elad, Learning multiscale sparse representations for image and video restoration, *SIAM Multiscale Modeling and Simulation*, 7(1):214–241, April 2008.
36. S. Mallat and E. LePennec, Sparse geometric image representation with bandelets, *IEEE Trans. on Image Processing*, 14(4):423–438, 2005.
37. S. Mallat and E. Le Pennec, Bandelet image approximation and compression, *SIAM Journal of Multiscale Modeling and Simulation*, 4(3):992–1039, 2005.
38. P. Moulin and J. Liu, Analysis of multiresolution image denoising schemes using generalized Gaussian and complexity priors, *IEEE Trans. on Information Theory*, 45(3):909–919, 1999.
39. J.-C. Pesquet and D. Leporini, A new wavelet estimator for image denoising, in *Proc. 6th Int. Conf. on Image Processing and Its Applications*, 1:249–253, 1997.
40. J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli, Image denoising using scale mixtures of Gaussians in the wavelet domain, *IEEE Trans. on Image Processing*, 12(11):1338–1351, 2003.
41. M. Protter and M. Elad, Image sequence denoising via sparse and redundant representations, *IEEE Trans. on Image Processing*, 18(1):27–36, January 2009.
42. G. Rosman, L. Dascal, A. Sidi, and R. Kimmel, Efficient Beltrami image filtering via vector extrapolation methods, *SIAM J. Imaging Sciences*, 2(3):858–878, 2009.
43. E.P. Simoncelli and E.H. Adelson, Noise removal via Bayesian wavelet coring, *Proceedings of the International Conference on Image Processing*, Lausanne, Switzerland. September 1996.
44. E.P. Simoncelli, W.T. Freeman, E.H. Adelson, and D.J. Heeger, Shiftable multiscale transforms, *IEEE Trans. on Information Theory*, 38(2):587–607, 1992.
45. N. Sochen, R. Kimmel, and A.M. Bruckstein. Diffusions and confusions in signal and image processing, *Journal of Mathematical Imaging and Vision*, 14(3):195–209, 2001.
46. A. Spira, R. Kimmel, and N. Sochen, A short time Beltrami kernel for smoothing images and manifolds, *IEEE Trans. on Image Processing*, 16(6):1628–1636, 2007.
47. J.-L. Starck, E.J. Candès, and D.L. Donoho, The curvelet transform for image denoising, *IEEE Trans. on Image Processing*, 11:670–684, 2002.

48. J.-L. Starck, M.J. Fadili, and F. Murtagh, The undecimated wavelet decomposition and its reconstruction, *IEEE Trans. on Image Processing*, 16(2):297–309, 2007.
49. C.M. Stein, Estimation of the mean of a multivariate distribution, *Proc. Prague Symp. Asymptotic Statist.*, pp. 345–381, 1973.
50. C.M. Stein, Estimation of the mean of a multivariate normal distribution, *Ann. Stat.*, 9(6):1135–1151, November 1981.
51. C. Vonesch, S. Ramani and M. Unser, Recursive risk estimation for non-linear image deconvolution with a wavelet-domain sparsity constraint, the 15th International Conference on Image Processing, (ICIP), pp.665–668, 12-15 October 2008.
52. Z. Wang, A.C. Bovik, H.R. Sheikh and E.P. Simoncelli, Image quality assessment: From error visibility to structural similarity, *IEEE Trans. on Image Processing*, 13(4):600–612, April 2004.





# Chapter 15

## Other Applications

### 15.1 General

This book is not meant to be a comprehensive textbook on image processing, and therefore it is not our intention to show how every familiar application in image processing finds a good use for the *Sparse-Land* model. Indeed, such a claim would not be true to begin with, as there are image processing problems for which the relation to this model has not been (and perhaps will never be) shown.

So far, we touched on deblurring and denoising of images, and also took *Sparse-Land* to compression of specific families of images. In this chapter we continue this journey by considering several more applications in image processing. Common to them all is the fact that these where shown to benefit well from the use of *Sparse-Land*. We begin with the MCA paradigm and its use for separation of image content. We shall show how this becomes useful for filling-in holes in images, a problem known as inpainting. Both the MCA and inpainting are highly ill-posed inverse problems, and yet they are handled quite well by using sparse and redundant representations. Such is also the case with the image scale-up problem, where a single image is enlarged while attempting to keep its sharpness and visual quality. We shall discuss this application as well, and show several ways to solve it using *Sparse-Land*.

Just before we embark on this journey, which ends this book, we draw the reader's attention to the following interesting observation: In the applications discussed herein, we shall meet the same pattern that appeared already in Chapter 14, of operating on images as a whole, or turning to operate on small overlapping patches. In fact, some of the techniques we shall describe here have great resemblance to the denoising algorithms we introduced before, and this should not be surprising.

## 15.2 Image Separation via MCA

In Chapter 9 we introduced the core ability of the *Sparse-Land* model to decompose a mixture of signals into their ingredients, assuming that each piece has a different (and known) dictionary. We assume that the observed signal is a superposition of two different sub-signals  $\mathbf{y}_1, \mathbf{y}_2$  and noise  $\mathbf{v}$  (i.e.  $\mathbf{y} = \mathbf{y}_1 + \mathbf{y}_2 + \mathbf{v}$ ). We further assume that  $\mathbf{y}_1$  is sparsely generated using model  $\mathcal{M}_1$  (with a dictionary  $\mathbf{A}_1$ ), and similarly,  $\mathbf{y}_2$  is sparsely generated using model  $\mathcal{M}_2$  (with a dictionary  $\mathbf{A}_2$ ). By solving

$$\min_{\mathbf{x}_1, \mathbf{x}_2} \|\mathbf{x}_1\|_0 + \|\mathbf{x}_2\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{A}_1\mathbf{x}_1 - \mathbf{A}_2\mathbf{x}_2\|_2^2 \leq \delta, \quad (15.1)$$

the resulting solution  $(\mathbf{x}_1^\delta, \mathbf{x}_2^\delta)$  would generate a plausible solution  $\hat{\mathbf{y}}_1 = \mathbf{A}\mathbf{x}_1^\delta, \hat{\mathbf{y}}_2 = \mathbf{A}\mathbf{x}_2^\delta$  to the separation problem. The parameter  $\delta$  should take into account both the noise energy, and the model inaccuracies in representing sparsely the signals  $\mathbf{y}_1$  and  $\mathbf{y}_2$ .

Following the terminology adopted by Jean-Luc Starck, we shall refer hereafter to this separation idea as *Morphological Component Analysis* (MCA). Note that the above equation is very similar to the denoising formulation,

$$\min_{\mathbf{x}} \|\mathbf{x}_a\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{A}_a\mathbf{x}_a\|_2^2 \leq \delta, \quad (15.2)$$

with one effective dictionary being  $\mathbf{A}_a = [\mathbf{A}_1, \mathbf{A}_2]$  ('a' in the subscript stands for 'all'), and a corresponding representation  $\mathbf{x}_a^T = [\mathbf{x}_1^T, \mathbf{x}_2^T]$ .

### 15.2.1 Image = Cartoon + Texture

Can we use MCA to separate image content? What are the ingredients an image would be assumed to be built of? Early work in image processing assumed that images are mostly composed of a piecewise smooth (cartoon-like) content. One strong manifestation to this belief is the way inverse problems in image processing have been regularized, using an expression that penalizes non-piecewise smooth behavior. A further evidence to this common belief is the way various Frames and transforms (e.g., curvelets, contourlets) were conceived and tested.

Today, there is a growing agreement that images are in fact a mixture of cartoon and texture parts, which are combined linearly, very much in line with the above model. This means that we may use MCA to separate image content to cartoon and texture images. However, for this to be done, we should answer three critical questions:

1. What is the benefit in such a separation?
2. What should the proper dictionaries be for these two kinds of contents? and
3. How should we numerically solve the problem posed in Equation (15.1)?

Addressing the first question, and generally speaking, the benefit in such a separation would be the potential ability to process the image better by treating each of the parts using a properly tailored tool.

As a first example for such a benefit, consider compression of still images. While a wavelet-based coding algorithm could do well for the cartoon part in the image, it is expected to perform poorly on the texture content. Disregarding this fact implies that the overall coding performance may be compromised. Instead, coding each piece using its proper method could lead to far better results. In that respect, coding of the texture should not target the same error measure (MSE). We may allow a generation of a similar texture in the reconstructed image, which is visually indistinguishable from the original one, while being in fact very different in MSE terms.

Another application that may benefit from MCA separation of images is denoising. Consider, for example, the global formulation of the image denoising problem posed in Equation (14.5). The dictionary  $\mathbf{A}$  should contain a part that corresponds to cartoon (e.g., Haar), and another part concatenated to it, which covers texture content. This may explain the inferior denoising results we have obtained for the Barbara image, using the global method with the Haar dictionary alone, as the texture dictionary was missing.

Moving from the global to a local denoising with a trained dictionary, we are in-fact implicitly using MCA. As can be seen in [Figure 14.10](#) (right), the overall dictionary  $\mathbf{A}$  contains both cartoon (one dominant oriented edge per atom) and texture (periodic content) atoms, suggesting that it is already a combination of the two desired parts. Note, however, that the task in denoising is oblivious to the separation goal, and the cartoon/texture atoms are mixed as an unordered set of columns in the overall dictionary  $\mathbf{A}$ .

There are other applications that may benefit from image separation to cartoon and texture, such as structured noise filtering, meaningful edge detection in textured images, object recognition, image content editing, and more. In the next section we shall discuss in more details the image inpainting problem – filling in missing information in images – which also benefits from such a separation.

In order to answer the second and the third questions posed above (which dictionaries to use, and how to numerically solve (15.1)), we should first choose whether to operate globally on the image, or locally using patches. We start with the global method that considers the image as a whole, following the same structure as in Chapter 14.

Before we turn to describe the global and the local MCA paradigms, we should mention that separation of an image to cartoon and texture can be done using other methods. Indeed, the first to consider such a separation were Mayer, Vese, and Osher in 2002. Their approach towards the image separation task focused on variational tools, extending the Total-Variation by proposing a dual norm for handling the texture part. We shall not dwell on these ideas here, but we should mention that despite the very different tools and language used, there are some striking similarities between the variational formulation of the separation, and the MCA paradigm discussed here.

### 15.2.2 Global MCA for Image Separation

We begin by modeling the problem: An ideal image  $\mathbf{y}_0 \in \mathbb{R}^N$  (of size  $\sqrt{N} \times \sqrt{N}$  pixels) is composed of two parts – a cartoon and texture images of the same size,  $\mathbf{y}_c$  and  $\mathbf{y}_t$ , respectively. The combined image  $\mathbf{y}_0 = \mathbf{y}_c + \mathbf{y}_t$  is measured in the presence of an additive zero-mean white and homogeneous Gaussian noise,  $\mathbf{v}$ , with a known standard deviation  $\sigma$ . The measured image,  $\mathbf{y}$ , is thus

$$\mathbf{y} = \mathbf{y}_0 + \mathbf{v} = \mathbf{y}_c + \mathbf{y}_t + \mathbf{v}. \quad (15.3)$$

We desire to design an algorithm that can recover the two parts of  $\mathbf{y}_0$ ,  $\mathbf{y}_c$  and  $\mathbf{y}_t$ . Using the MCA formulation mentioned above, we propose to solve

$$\hat{\mathbf{x}}_c, \hat{\mathbf{x}}_t = \arg \min_{\mathbf{x}_c, \mathbf{x}_t} \lambda \|\mathbf{x}_c\|_1 + \lambda \|\mathbf{x}_t\|_1 + \frac{1}{2} \|\mathbf{y} - \mathbf{A}_c \mathbf{x}_c - \mathbf{A}_t \mathbf{x}_t\|_2^2. \quad (15.4)$$

Notice that we have chosen to replace the proximity constraint in Equation (15.1) by a penalty, and the  $\ell_0$ -norm by an  $\ell_1$ -norm, both as shown earlier in various contexts. The two unknowns to be found are the corresponding representations of the two parts. Once computed, the two estimated parts are obtained by  $\hat{\mathbf{y}}_c = \mathbf{A}_c \hat{\mathbf{x}}_c$  and  $\hat{\mathbf{y}}_t = \mathbf{A}_t \hat{\mathbf{x}}_t$ .

The matrices  $\mathbf{A}_c \in \mathbb{R}^{N \times M_c}$  and  $\mathbf{A}_t \in \mathbb{R}^{N \times M_t}$  are to be chosen such that they enable sparse representation of cartoon and texture contents respectively. The matrix  $\mathbf{A}_c$  could be some type of wavelet, or perhaps ridgelets, curvelets, or contourlets, depending on the type of cartoon content we expect to find in the image (see examples below). The texture dictionary  $\mathbf{A}_t$  should contain oscillatory atoms, such as found in the Gabor transform, the global DCT, or the local one.<sup>1</sup> The sizes  $M_c$  and  $M_t$  are the number of atoms in the two dictionaries, and those typically satisfy  $M_c, M_t \gg N$ . For example, the curvelet dictionary with 6 resolution levels has  $M_c = 129N$ . Similarly, the local-DCT applied on patches of  $8 \times 8$  pixels has  $M_t = 64N$ .

An alternative to the above formulation can be proposed, where we move from the synthesis model to an analysis one (as explained in Chapter 9). Assuming (only for the moment!!) that  $\mathbf{A}_c$  and  $\mathbf{A}_t$  are square and invertible, we could use the assignments  $\hat{\mathbf{x}}_c = \mathbf{A}_c^{-1} \hat{\mathbf{y}}_c$  and  $\hat{\mathbf{x}}_t = \mathbf{A}_t^{-1} \hat{\mathbf{y}}_t$  and obtain an alternative formulation of our problem in the form

$$\hat{\mathbf{y}}_c, \hat{\mathbf{y}}_t = \arg \min_{\mathbf{y}_c, \mathbf{y}_t} \lambda \|\mathbf{T}_c \mathbf{y}_c\|_1 + \lambda \|\mathbf{T}_t \mathbf{y}_t\|_1 + \frac{1}{2} \|\mathbf{y} - \mathbf{y}_c - \mathbf{y}_t\|_2^2, \quad (15.5)$$

where we have denoted  $\mathbf{T}_c = \mathbf{A}_c^{-1}$  and  $\mathbf{T}_t = \mathbf{A}_t^{-1}$ .

In the more general case where the dictionaries are non-square, we can still use the analysis formulation, and assume instead that  $\mathbf{T}_c = \mathbf{A}_c^+$  and  $\mathbf{T}_t = \mathbf{A}_t^+$ . However,

<sup>1</sup> The global DCT is a regular DCT transform applied to the whole image. The local DCT is obtained by applying the unitary DCT transform on image patches with full-overlaps. This constitutes the pseudo-inverse of  $\mathbf{A}_t$ , and the atoms of  $\mathbf{A}_t$  are therefore the unitary-DCT basis functions positioned in all possible locations in the image.

as opposed to the square and invertible case discussed above, the analysis and synthesis formulations are no longer equivalent. One appealing benefit that the analysis formulation in Equation (15.5) has is the fact that the unknowns are directly the image parts we desire, and those are typically of a lower-dimension ( $2N$  instead of  $M_c + M_t$ ), compared to their sparse representations.

As for the numerical solution of the problems posed in (15.4) or (15.5), various methods can be envisioned. In particular, iterative-shrinkage algorithms (see Chapter 6) can be used to solve directly the synthesis formulation in (15.4). For example, the SSF algorithm, as described in Chapter 6, leads to the iterative update,

$$\hat{\mathbf{x}}_a^{k+1} = \mathcal{S}_\lambda \left( \frac{1}{c} \mathbf{A}_a^T (\mathbf{y} - \mathbf{A}_a \hat{\mathbf{x}}_a^k) + \hat{\mathbf{x}}_a^k \right) \quad (15.6)$$

which is posed in terms of the combined dictionary and the combined representation as in Equation (15.2). Breaking this into the two representation parts we get

$$\begin{aligned} \hat{\mathbf{x}}_c^{k+1} &= \mathcal{S}_\lambda \left( \frac{1}{c} \mathbf{A}_c^T (\mathbf{y} - \mathbf{A}_c \hat{\mathbf{x}}_c^k - \mathbf{A}_t \hat{\mathbf{x}}_t^k) + \hat{\mathbf{x}}_c^k \right) \\ \hat{\mathbf{x}}_t^{k+1} &= \mathcal{S}_\lambda \left( \frac{1}{c} \mathbf{A}_t^T (\mathbf{y} - \mathbf{A}_c \hat{\mathbf{x}}_c^k - \mathbf{A}_t \hat{\mathbf{x}}_t^k) + \hat{\mathbf{x}}_t^k \right), \end{aligned} \quad (15.7)$$

where  $\mathcal{S}_\lambda(\mathbf{r})$  is an element-wise soft-thresholding operation on  $\mathbf{r}$  with a threshold  $\lambda$ , and the parameter  $c$  should be chosen such that<sup>2</sup>  $c > \lambda_{\max}(\mathbf{A}_a^T \mathbf{A}_a)$ .

The analysis alternative formulation in (15.5) can be treated similarly, with some necessary modifications to the above algorithm. More specifically, assuming that the two dictionaries are redundant ( $M_c, M_t \geq N$ ) and full-rank, following the description given in Chapter 9 we can turn back the analysis into a synthesis form by denoting

$$\begin{aligned} \mathbf{x}_c &= \mathbf{T}_c \mathbf{y}_c \rightarrow \mathbf{y}_c = \mathbf{T}_c^+ \mathbf{x}_c = \mathbf{A}_c \mathbf{x}_c \\ \mathbf{x}_t &= \mathbf{T}_t \mathbf{y}_t \rightarrow \mathbf{y}_t = \mathbf{T}_t^+ \mathbf{x}_t = \mathbf{A}_t \mathbf{x}_t. \end{aligned} \quad (15.8)$$

This leads to an optimization problem of the form

$$\begin{aligned} \hat{\mathbf{x}}_c, \hat{\mathbf{x}}_t &= \arg \min_{\mathbf{x}_c, \mathbf{x}_t} \lambda \|\mathbf{x}_c\|_1 + \lambda \|\mathbf{x}_t\|_1 + \frac{1}{2} \|\mathbf{y} - \mathbf{A}_c \mathbf{x}_c - \mathbf{A}_t \mathbf{x}_t\|_2^2 \\ &\text{Subject to } \mathbf{T}_c \mathbf{A}_c \mathbf{x}_c = \mathbf{x}_c \text{ and } \mathbf{T}_t \mathbf{A}_t \mathbf{x}_t = \mathbf{x}_t. \end{aligned} \quad (15.9)$$

The additional constraints appear because in the migration from  $\mathbf{y}_c$  to  $\mathbf{x}_c$  (and similarly from  $\mathbf{y}_t$  to  $\mathbf{x}_t$ ) we should demand that  $\mathbf{x}_c$  is in the column-span of  $\mathbf{T}_c$  ( $\mathbf{T}_t$ , respectively).

Exploiting the analysis formulation in Equation (15.9), we could use the very same iteration formula as in Equation (15.7), since the penalty term in Equation (15.9) is the same one as in Equation (15.4), and thus this iteration induces a descent

<sup>2</sup> Using the Singular-Value-Decomposition of  $\mathbf{A}_a$ , it is easy to show that this requirement is equivalent to  $c > \lambda_{\max}(\mathbf{A}_a \mathbf{A}_a^T) = \lambda_{\max}(\mathbf{A}_c \mathbf{A}_c^T + \mathbf{A}_t \mathbf{A}_t^T)$ . For tight Frames we have that  $\mathbf{A}_c \mathbf{A}_c^T = \mathbf{A}_t \mathbf{A}_t^T = \mathbf{I}$ , implying that  $c$  should be larger than 2.

on the penalty. Thus, we start by the update

$$\begin{aligned}\hat{\mathbf{x}}_c^{k+\frac{1}{2}} &= \mathcal{S}_\lambda \left( \frac{1}{c} \mathbf{A}_c^T (\mathbf{y} - \mathbf{A}_c \hat{\mathbf{x}}_c^k - \mathbf{A}_t \hat{\mathbf{x}}_t^k) + \hat{\mathbf{x}}_c^k \right) \\ \hat{\mathbf{x}}_t^{k+\frac{1}{2}} &= \mathcal{S}_\lambda \left( \frac{1}{c} \mathbf{A}_t^T (\mathbf{y} - \mathbf{A}_c \hat{\mathbf{x}}_c^k - \mathbf{A}_t \hat{\mathbf{x}}_t^k) + \hat{\mathbf{x}}_t^k \right),\end{aligned}\quad (15.10)$$

as above. However, this should be followed by a projection onto the constraints, by computing<sup>3</sup>

$$\begin{aligned}\hat{\mathbf{x}}_c^{k+1} &= \mathbf{T}_c \mathbf{A}_c \hat{\mathbf{x}}_c^{k+\frac{1}{2}} \\ \hat{\mathbf{x}}_t^{k+1} &= \mathbf{T}_t \mathbf{A}_t \hat{\mathbf{x}}_t^{k+\frac{1}{2}}.\end{aligned}\quad (15.11)$$

While this algorithm indeed handles the analysis-formulation as we desired, it lost its main appeal since it is formulated in terms of the representations rather than the separated images. This can be fixed by modifying Equations (15.10) and (15.11) to be

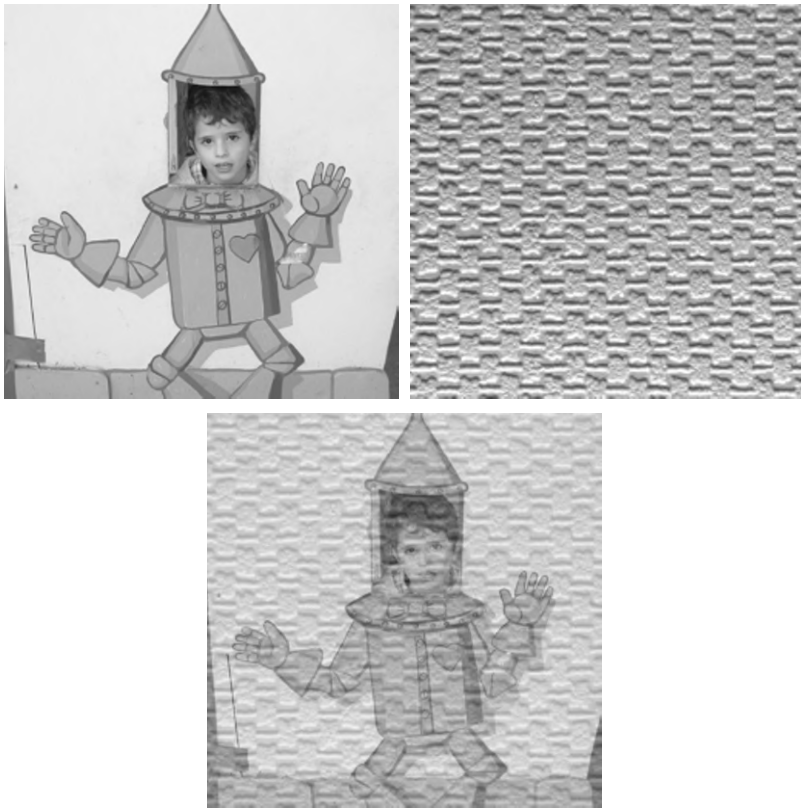
$$\begin{aligned}\hat{\mathbf{y}}_c^{k+1} &= \mathbf{A}_c \cdot \mathcal{S}_\lambda \left( \frac{1}{c} \mathbf{A}_c^T (\mathbf{y} - \hat{\mathbf{y}}_c^k - \hat{\mathbf{y}}_t^k) + \mathbf{T}_c \hat{\mathbf{y}}_c^k \right) \\ \hat{\mathbf{y}}_t^{k+1} &= \mathbf{A}_t \cdot \mathcal{S}_\lambda \left( \frac{1}{c} \mathbf{A}_t^T (\mathbf{y} - \hat{\mathbf{y}}_c^k - \hat{\mathbf{y}}_t^k) + \mathbf{T}_t \hat{\mathbf{y}}_t^k \right),\end{aligned}\quad (15.12)$$

where we have used the assignments as given in Equation (15.8).

We now turn to present several experimental results, obtained using the global analysis paradigm. All the results shown here are taken from the work by Starck, Elad, and Donoho (IEEE-TIP, 2004), and therefore, we do not include details on parameter settings, etc., and focus on the qualitative results obtained. We start with a synthetically generated image. This image is composed of a natural scene (considered here as the cartoon part), a texture, and an additive white Gaussian noise ( $\sigma = 10$ ). [Figure 15.1](#) shows the ingredients of this image.

The dictionaries chosen for this separation are the curvelets for the cartoon part, and a global DCT transform for the texture. The very low frequency components of the image are first subtracted from it, and then added to the cartoon part after the separation. The reason for this is the evident overlap that exists between the two dictionaries – both consider the low-frequency content to belong to them as both can represent it efficiently. Thus, by removing this content prior to the separation we avoid a separation ambiguity. Also, by returning this content later to the curvelets part, we use our expectation to see the low frequencies as belonging to the piecewise smooth image. [Figure 15.2](#) shows the results of this experiment. As we can see, the three ingredients of the image (cartoon, texture, and noise) are separated very well.

<sup>3</sup> Given a vector  $\mathbf{v}$ , we seek the closest alternative  $\mathbf{u}$  such that the constraint  $\mathbf{u} = \mathbf{P}\mathbf{u}$  is satisfied, where  $\mathbf{P}$  is a projection matrix (square, symmetric, with eigenvalues being either zeros or ones). Solving this using Lagrange multipliers leads to the optimal solution  $\mathbf{u} = \mathbf{P}\mathbf{v}$ .



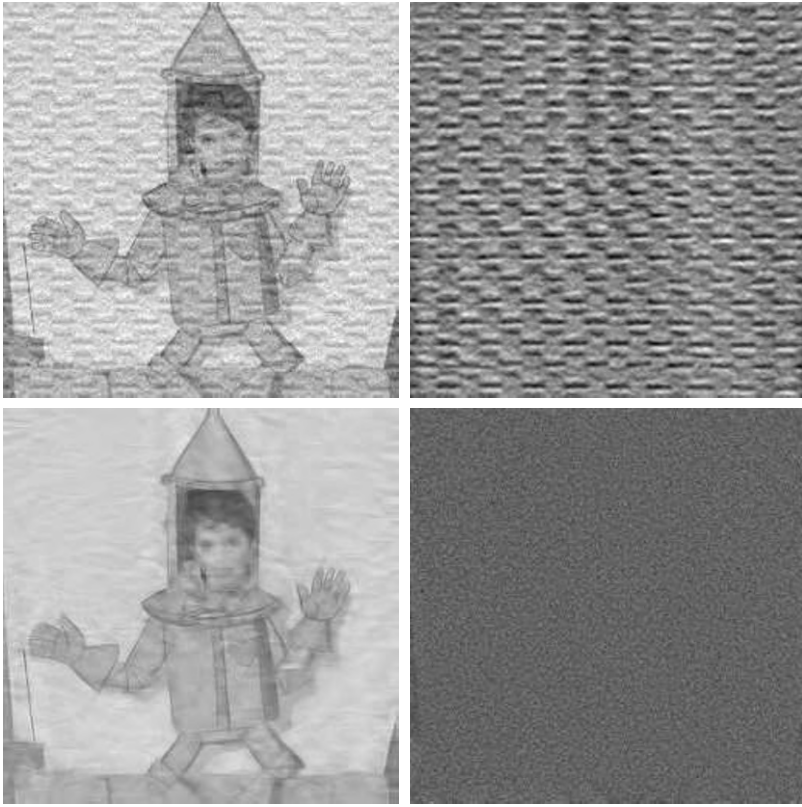
**Fig. 15.1** The original cartoon and texture parts (top left and right), and the overall image (bottom) to be processed.

In a second experiment, we apply the same separation algorithm to the image *Barbara*. This time we use the curvelets and local-DCT (blocks of size  $32 \times 32$  pixels) dictionaries. Figure 15.3 shows the original image and the separated cartoon and texture parts. Figure 15.4 presents a magnified part of the face.

We conclude the experimental part by showing two simple applications for the separation obtained. The first is an edge detection in an image. Such a process may become a crucial processing stage in many computer-vision applications. When the texture is highly contrasted, most of the detected edges are due to the texture rather than the cartoon (which is the more informative) part. By first separating the two components we can detect the true object's edges by applying edge detection on the cartoon part alone. Figure 15.5 shows the edges detected by the Canny edge detector algorithm on both the original image and the cartoon reconstructed component.

Figure 15.6 shows a galaxy imaged with the GEMINI-OSCIR instrument. The data is contaminated by white additive noise and a stripping artifact (assumed to be the texture in the image) due to the instrument electronics. As the galaxy is isotropic,





**Fig. 15.2** The original noisy image (top left), the separated texture part (top right), the separated cartoon (bottom left), and the residual noise component (bottom right).

the isotropic wavelet is used instead of the curvelets for representing the cartoon part. The global DCT is used again for the texture, due to its structure. [Figure 15.6](#) summarizes the results of this separation where a successful isolation of the galaxy, the textured disturbance, and the additive noise is seen.

### ***15.2.3 Local MCA for Image Separation***

Just like in the image denoising application, MCA can be performed on small and overlapping patches. The immediate benefits of such a mode of operation are (i) the locality of the processing, which lends itself to efficient parallel implementation; and (ii) the ability to incorporate learned dictionaries into the MCA. This last benefit – the ability to work with trained dictionaries – has far-reaching implications, as we



**Fig. 15.3** The separated cartoon (left) and texture (right) parts, and the original image Barbara from which these were extracted.

can use it to modify the separation task to any mixture of contents. For example, two (or more) mixed textures can be separated using MCA, provided that we have dictionaries fitting each of the ingredients.

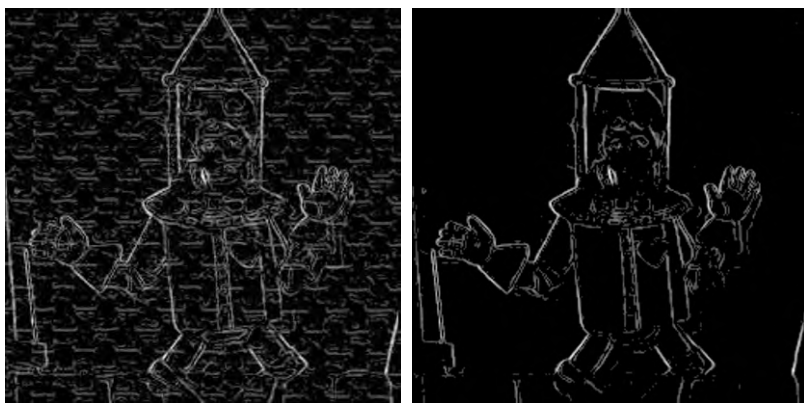
There are various ways to turn MCA into a locally-processing algorithm, similar to the wealth with which the denoising task has been shown to be handled locally in Chapter 14. In this chapter we shall explore one such scheme, that builds on the K-SVD denoising algorithm. As we shall see next, by a simple extension of this method, we get state-of-the-art separation results as a side-benefit.

Following the details given in Section 3.3 in Chapter 14, we start by assuming that the unknown mixed image  $\mathbf{y}_0 = \mathbf{y}_c + \mathbf{y}_t$  is fully expressed by the fact that for every patch in it,  $\mathbf{R}_k \mathbf{y}_0 = \mathbf{R}_k \mathbf{y}_c + \mathbf{R}_k \mathbf{y}_t$ , there exist two sparse representation vectors  $\mathbf{q}_c^k$  and  $\mathbf{q}_t^k$ , such that

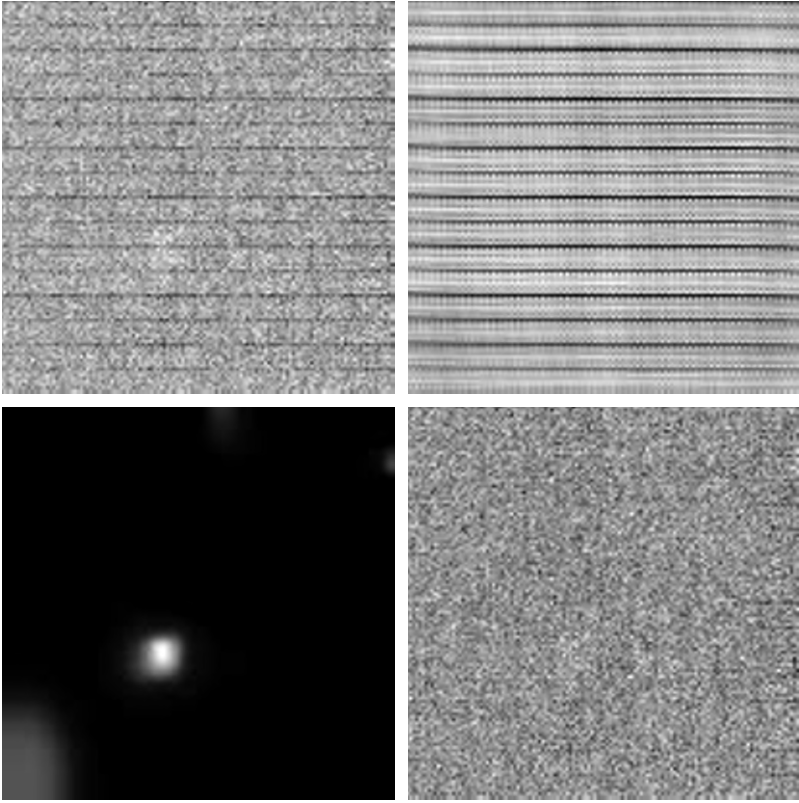
$$\|\mathbf{R}_k \mathbf{y}_c + \mathbf{R}_k \mathbf{y}_t - \mathbf{A}_c \mathbf{q}_c^k - \mathbf{A}_t \mathbf{q}_t^k\|_2 \leq \epsilon, \quad (15.13)$$



**Fig. 15.4** A zoomed-in portion of the cartoon and texture parts (top left and right), and the original image (bottom), all taken from the results presented in [Figure 15.3](#)



**Fig. 15.5** The detected edges on the original image (left), and those detected using the cartoon part only (right).



**Fig. 15.6** The original GEMINI image (top left), the texture component (top right), the cartoon component (bottom left), and the residual noise (bottom right).

where  $\mathbf{A}_c$  and  $\mathbf{A}_t$  are two fixed dictionaries. Rather than using this notation, we return temporarily to the combined dictionary  $\mathbf{A}_a$  and the corresponding concatenated representation for this patch,  $\mathbf{q}_a^k$ . Thus, the above can be rewritten as  $\|\mathbf{R}_k \mathbf{y}_0 - \mathbf{A}_a \mathbf{q}_a^k\|_2 \leq \epsilon$ . Using this within a global MAP estimator leads to

$$\begin{aligned} \{\{\hat{\mathbf{q}}_a^k\}_{k=1}^N, \hat{\mathbf{y}}\} = \arg \min_{\mathbf{z}, \{\mathbf{q}_a^k\}_k} & \lambda \|\mathbf{z} - \mathbf{y}\|_2^2 + \\ & \sum_{k=1}^N \mu_k \|\mathbf{q}_a^k\|_0 + \sum_{k=1}^N \|\mathbf{A}_a \mathbf{q}_a^k - \mathbf{R}_k \mathbf{z}\|_2^2. \end{aligned} \quad (15.14)$$

In this expression the first term is the log-likelihood global force that demands a proximity between the measured image,  $\mathbf{y}$ , and its denoised (and unknown) version  $\mathbf{z}$ . The second and the third terms are the image prior that makes sure that in the

constructed image,  $\mathbf{z}$ , every patch  $\mathbf{p}_k = \mathbf{R}_k \mathbf{z}$  of size  $\sqrt{n} \times \sqrt{n}$  in every location (thus the summation by  $k$ ) has a sparse representation with bounded error.

The problem posed in Equation (15.14) is one of denoising, and there is no trace for the separation goal behind it. This is natural, since whether  $\mathbf{y}_0$  is mixed or not, it still requires denoising in order to remove the additive noise, and the fact that patches in  $\mathbf{y}_0$  can be assumed to have sparse representations with respect to the dictionary  $\mathbf{A}_a$  is sufficient for denoising it efficiently.

Therefore, we treat the above denoising just as described in the previous chapter, by initializing the dictionary to be the redundant DCT, and iterating between updates of the representations  $\{\hat{\mathbf{q}}_a^k\}_{k=1}^N$  and the dictionary  $\mathbf{A}_a$ . This sequence of steps forms an effective K-SVD dictionary-learning that trains on the corrupted image itself. The regular image denoising algorithm would proceed after such steps by fixing the representations and the dictionary, and turning to compute the clean image by the formula

$$\hat{\mathbf{y}} = \left( \lambda \mathbf{I} + \sum_{k=1}^N \mathbf{R}_k^T \mathbf{R}_k \right)^{-1} \left( \lambda \mathbf{y} + \sum_{k=1}^N \mathbf{R}_k^T \mathbf{A}_a \hat{\mathbf{q}}_a^k \right). \quad (15.15)$$

However, this is where our local MCA algorithm departs from the denoising process introduced earlier. Our alternative steps would be to (i) decompose the overall dictionary  $\mathbf{A}_a$  into the cartoon and texture dictionaries, and then conclude the algorithm by (ii) using the two separated dictionaries in order to separate the image.

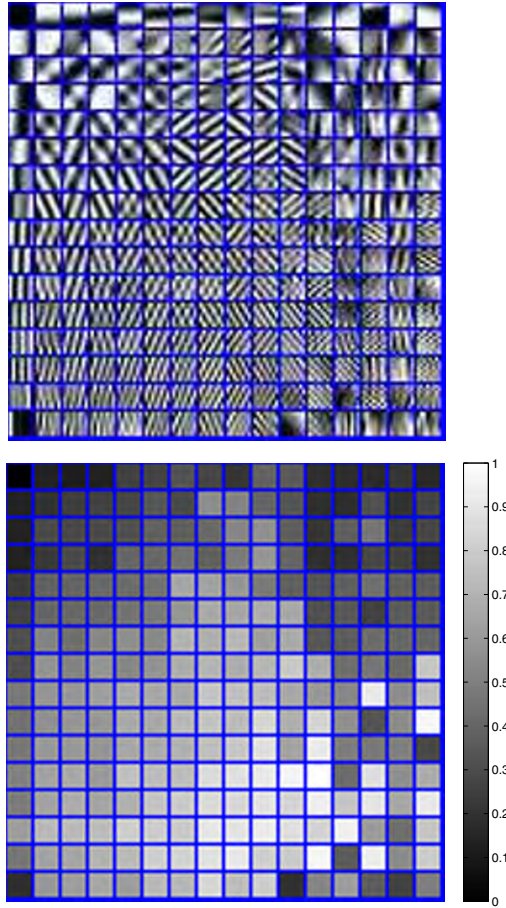
The trained dictionary  $\mathbf{A}_a$  contains the atoms of  $\mathbf{A}_c$  and  $\mathbf{A}_t$  in an unordered fashion. We can identify the texture/cartoon atoms, by their Total-Variation measure, since we expect cartoon atoms to be smooth or piecewise smooth, while texture atoms are expected to oscillate strongly.

As an example, the above algorithm is applied to a noisy ( $\sigma = 10$ , as in the previous experiments) version of the image *Barbara* for 25 iterations, and it leads to the dictionary presented in Figure 15.7 (top). For an atom  $\mathbf{a}$ , organized as a patch of size  $\sqrt{n} \times \sqrt{n}$ , we use an activity measure similar to TV, given by

$$\begin{aligned} \text{Activity}(\mathbf{a}) = & \sum_{i=2}^n \sum_{j=1}^n |a[i, j] - a[i-1, j]| \\ & + \sum_{i=1}^n \sum_{j=2}^n |a[i, j] - a[i, j-1]|. \end{aligned} \quad (15.16)$$

These values (normalized such that their maximal value is 1) for all the dictionary atoms are shown in Figure 15.7(bottom). As can be seen, these values reflect well the degree of “textureness” in each atom.

Figure 15.8 presents the classification of each atom as belonging to the texture or the cartoon dictionaries, based on thresholding the above (normalized) measure with  $T = 0.27$  (this value was chosen manually for best performance. A value above this threshold indicates a texture atom). Note that from the overall 256 atoms, only 36 are attributed to the cartoon, while the rest are considered as texture atoms.



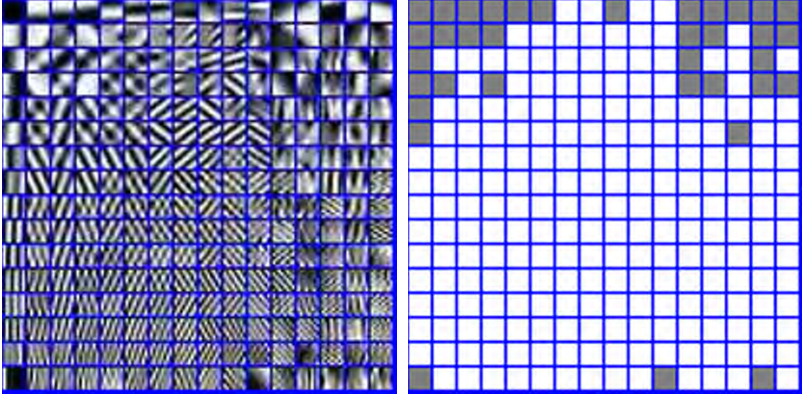
**Fig. 15.7** Top: The K-SVD dictionary obtained on a noisy version ( $\sigma = 10$ ) of Barbara after 25 iterations; Bottom: The total-variation-like activity measure per each atom.

With the decomposition of the dictionary  $\mathbf{A}_a$  into  $\mathbf{A}_c$  and  $\mathbf{A}_t$  we also obtain a corresponding separation of the representations  $\mathbf{q}_a^k$  to  $\mathbf{q}_c^k$  and  $\mathbf{q}_t^k$ . This is done by simply allocating the coefficients that correspond to cartoon atoms to  $\mathbf{q}_c^k$ , and similarly, allocating the coefficients that correspond to texture atoms to  $\mathbf{q}_t^k$  for the texture part.

We are left with the need to recover  $\hat{\mathbf{y}}_c$  and  $\hat{\mathbf{y}}_t$ , exploiting the decomposition of the dictionary. For this, we return to Equation (15.14), and define two new unknowns,  $\mathbf{z}_c$  and  $\mathbf{z}_t$ , such that  $\mathbf{z} = \mathbf{z}_c + \mathbf{z}_t$ . In order to enable a separation, we modify the term  $\|\mathbf{A}_a \mathbf{q}_a^k - \mathbf{R}_k \mathbf{z}\|_2^2$  into

$$\|\mathbf{A}_a \mathbf{q}_a^k - \mathbf{R}_k \mathbf{z}\|_2^2 = \|\mathbf{A}_c \mathbf{q}_c^k + \mathbf{A}_t \mathbf{q}_t^k - \mathbf{R}_k \mathbf{z}\|_2^2 \quad (15.17)$$





**Fig. 15.8** A mapping (right) that shows the texture (white) and the cartoon (dark gray) atoms, obtained by thresholding the TV activity measure by  $T = 0.27$ .

$$\begin{aligned}
 &= \|\mathbf{A}_c \mathbf{q}_c^k + \mathbf{A}_t \mathbf{q}_t^k - \mathbf{R}_k \mathbf{z}_c - \mathbf{R}_k \mathbf{z}_t\|_2^2 \\
 &\approx \|\mathbf{A}_c \mathbf{q}_c^k - \mathbf{R}_k \mathbf{z}_c\|_2^2 + \|\mathbf{A}_t \mathbf{q}_t^k - \mathbf{R}_k \mathbf{z}_t\|_2^2.
 \end{aligned}$$

The reason for the last approximate equality is our belief that the representation errors for the cartoon and the texture patches are not expected to correlate, i.e.,

$$(\mathbf{A}_c \mathbf{q}_c^k - \mathbf{R}_k \mathbf{z}_c)^T (\mathbf{A}_t \mathbf{q}_t^k - \mathbf{R}_k \mathbf{z}_t) \approx 0. \quad (15.18)$$

With these changes, Equation (15.14) becomes

$$\begin{aligned}
 \{\hat{\mathbf{y}}_c, \hat{\mathbf{y}}_t\} &= \arg \min_{\mathbf{z}_c, \mathbf{z}_t} \lambda \|\mathbf{z}_c + \mathbf{z}_t - \mathbf{y}\|_2^2 \\
 &+ \sum_{k=1}^N \|\mathbf{A}_c \mathbf{q}_c^k - \mathbf{R}_k \mathbf{z}_c\|_2^2 + \sum_{k=1}^N \|\mathbf{A}_t \mathbf{q}_t^k - \mathbf{R}_k \mathbf{z}_t\|_2^2.
 \end{aligned} \quad (15.19)$$

Notice that the  $\ell_0$  term that forces sparsity of the representations has been removed. This is done because we assume now that  $\mathbf{q}_c^k$  and  $\mathbf{q}_t^k$  are fixed and known, and our goal is the recovery of  $\mathbf{z}_c$  and  $\mathbf{z}_t$ . Taking the derivatives with respect to  $\mathbf{z}_c$  and  $\mathbf{z}_t$  gives the separation result

$$\begin{aligned}
 \begin{bmatrix} \hat{\mathbf{y}}_c \\ \hat{\mathbf{y}}_t \end{bmatrix} &= \begin{bmatrix} \lambda \mathbf{I} + \sum_{k=1}^N \mathbf{R}_k^T \mathbf{R}_k & \lambda \mathbf{I} \\ \lambda \mathbf{I} & \lambda \mathbf{I} + \sum_{k=1}^N \mathbf{R}_k^T \mathbf{R}_k \end{bmatrix}^{-1} \\
 &\cdot \begin{bmatrix} \lambda \mathbf{y} + \sum_{k=1}^N \mathbf{R}_k^T \mathbf{A}_c \mathbf{q}_c^k \\ \lambda \mathbf{y} + \sum_{k=1}^N \mathbf{R}_k^T \mathbf{A}_t \mathbf{q}_t^k \end{bmatrix}.
 \end{aligned} \quad (15.20)$$

In the extreme case of  $\lambda = 0$ , the solution is given by



**Fig. 15.9** The separated cartoon (left) and texture (right) parts, using the local MCA with the K-SVD dictionary ( $\lambda = 0$ ).

$$\hat{\mathbf{y}}_c = \left[ \sum_{k=1}^N \mathbf{R}_k^T \mathbf{R}_k \right]^{-1} \sum_{k=1}^N \mathbf{R}_k^T \mathbf{A}_c \mathbf{q}_c^k \quad (15.21)$$

and

$$\hat{\mathbf{y}}_t = \left[ \sum_{k=1}^N \mathbf{R}_k^T \mathbf{R}_k \right]^{-1} \sum_{k=1}^N \mathbf{R}_k^T \mathbf{A}_t \mathbf{q}_t^k,$$

which is very similar to the denoising solution posed in Equation (15.15). In the more general case, the above inversion can be done very easily, exploiting the  $2 \times 2$  block matrix inversion formula,<sup>4</sup> and exploiting the fact that all the blocks in our case are diagonal and easily invertible.

Figure 15.9 shows the results obtained with this local method for the simple case of  $\lambda = 0$  on the image Barbara. As can be seen, the separation is very successful, and with results that are generally comparable to those obtained by the global method described above, and this is achieved without the need to pre-specify the dictionaries for the two parts.

<sup>4</sup> This formula is given by

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{Q} & -\mathbf{QBD}^{-1} \\ -\mathbf{D}^{-1}\mathbf{CQ} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{CQBD}^{-1} \end{bmatrix},$$

where  $\mathbf{Q} = (\mathbf{A} - \mathbf{BD}^{-1}\mathbf{C})^{-1}$ .



## 15.3 Image Inpainting and Impulsive Noise Removal

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 (when removing an object from the scene). Extensive work on this problem is reported in the image processing literature, using various techniques. In this section we consider this problem, and more specifically, we focus on ways to inpaint images using the *Sparse-Land* signal model. We start our discussion with the core ideas for inpainting sparsely represented signals, and then extend this discussion to images.

### 15.3.1 Inpainting *Sparse-Land* Signals – Core Principles

Assume that  $\mathbf{y}_0 \in \mathbb{R}^n$  is a simplified *Sparse-Land* signal, meaning that there exists a sparse representation vector  $\mathbf{x}_0$  with  $\|\mathbf{x}_0\|_0 = k_0$  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. This matrix is of size  $(n-p) \times n$ , built by taking the  $n \times n$  identity matrix and removing the  $p$  rows that correspond to the dropped samples. Could we recover these missing values? How?

Considering this as a classic inverse problem in the form described in Chapter 9, we may formulate this task as

$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{y} = \mathbf{M}\mathbf{A}\mathbf{x}. \quad (15.22)$$

We should find the representation that solves this problem,  $\hat{\mathbf{x}}_0$ , and the recovered (full) signal  $\hat{\mathbf{y}}_0$  would be  $\mathbf{A}\hat{\mathbf{x}}_0$ . However, the question remains: Why should this work well?

In order to answer this question, we return to the worst-case analysis that was introduced in Chapter 2. Assume that the original representation  $\mathbf{x}_0$  is sparse enough, namely,  $k_0 < 0.5 \cdot \text{spark}(\mathbf{M}\mathbf{A})$ . In such a case we know that the original representation  $\mathbf{x}_0$  is necessarily the sparsest possible solution for the linear system  $\mathbf{y} = \mathbf{M}\mathbf{A}\mathbf{x}$ . As such, if we could solve the problem posed in Equation (15.22), we should have that  $\hat{\mathbf{x}}_0 = \mathbf{x}_0$ . This implies a perfect recovery of the original signal,  $\hat{\mathbf{y}}_0 = \mathbf{y}_0$ . Therefore, if  $\mathbf{x}_0$  is sparse enough, inpainting is guaranteed to succeed perfectly.

How does  $\mathbf{M}$  affect the *spark* of the matrix  $\mathbf{M}\mathbf{A}$ ? This multiplication removes  $p$  rows from the dictionary  $\mathbf{A}$ , and with such a modification, the *spark* necessarily deteriorates (gets smaller). This is because every set of  $\text{spark}(\mathbf{A}) - 1$  columns from  $\mathbf{A}$  are linearly-independent, by definition of the *spark*. Removal of an arbitrary row from  $\mathbf{A}$  may either leave all of these sets intact (and thus the *spark* remains unchanged), or modify one or more of those groups to become linearly-dependent, causing a decrease in the *spark*. Once a set of columns becomes dependent, removal of other rows cannot turn such a set to become independent. Thus, the value  $\text{spark}(\mathbf{M}\mathbf{A})$  is non-increasing as a function of the removal count  $p$ . In particular, we have that



**Fig. 15.10** The image Peppers on which we perform the inpainting test.

$\text{spark}(\mathbf{MA}) \leq \text{spark}(\mathbf{A})$ , which means that our assumption  $k_0 < 0.5 \cdot \text{spark}(\mathbf{MA})$  implies that  $\mathbf{x}_0$  is the sparsest representation for the original signal  $\mathbf{y}_0$ , as well.

The above is a simplified inpainting problem, and yet it clarifies why inpainting of *Sparse-Land* signals could conceptually succeed. The core idea is the understanding that since the signal  $\mathbf{y}_0$  has only  $2k_0$  degrees of freedom (atom indices and their weights), we can recover this signal completely from less than the full  $n$  samples. In the extreme case,<sup>5</sup> a set of  $2k_0$  samples may be sufficient, meaning that  $p$  could be as high as  $n - 2k_0$ .

When noise is added both to the model and to the measurements, the actual inpainting should be done by approximating the solution of

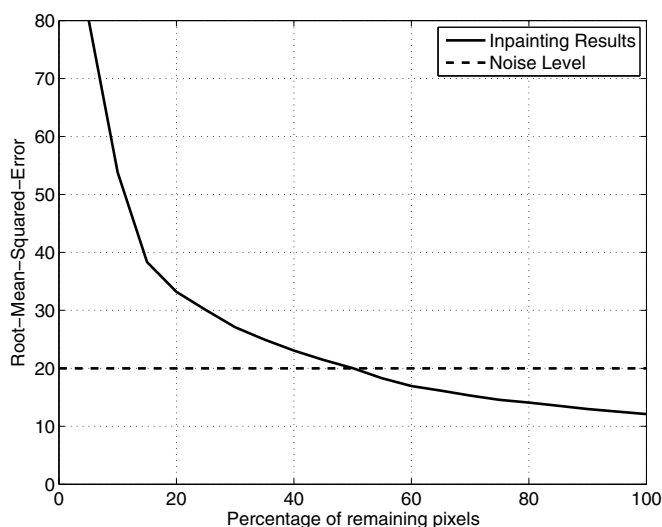
$$\min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{MAx}\|_2 \leq \delta. \quad (15.23)$$

Spark considerations no-longer apply, but as we have seen in Chapter 5, alternative analysis exists to support the success of such a recovery algorithm, both for the worst-case scenario, and for the more optimistic and more realistic average performance.

Will this methodology work in practice? We answer this question by introducing a **preliminary** experiment that illustrates how the above simple guide-lines can lead to a reasonable recovery of missing samples. We shall later return to this experiment and improve it in various ways. We operate on the image **Peppers** (of size  $256 \times 256$  pixels) shown in [Figure 15.10](#). This image is contaminated by an additive white Gaussian noise with  $\sigma = 20$ , and then sub-sampled in random locations, such that a portion of the pixels remains.

The inpainting we propose to perform operates on patches of size  $8 \times 8$  pixels, extracted with no overlaps from the corrupted image. Using a redundant DCT

<sup>5</sup> This depends on  $\mathbf{A}$  and the effect of  $\mathbf{M}$  on it. If  $\text{spark}(\mathbf{MA})$  has full  $\text{spark}(2k_0 + 1)$ , then  $2k_0$  measurements are indeed sufficient.



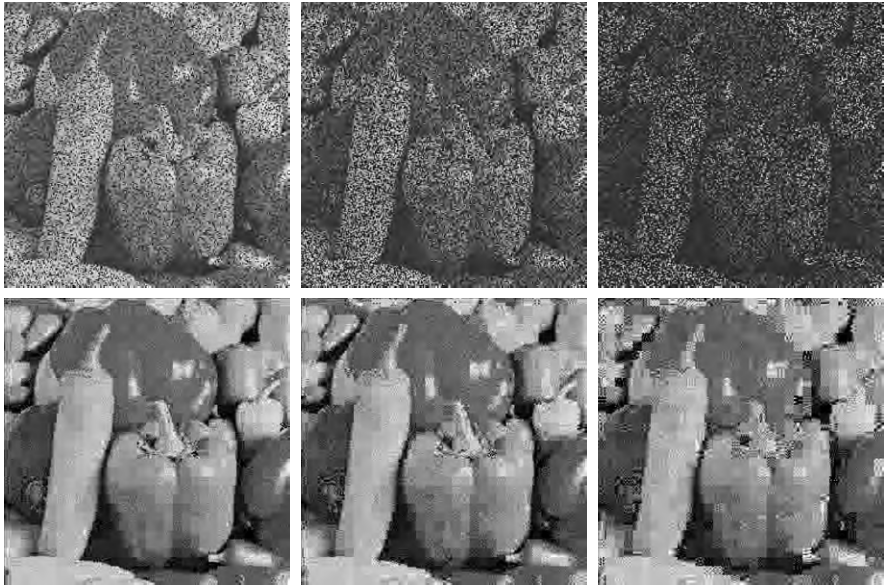
**Fig. 15.11** Simple inpainting results – operating locally with no overlaps and using the redundant DCT dictionary: This graph shows the RMSE (per-pixel) of the inpainting recovered image, as a function of the percentage of remaining pixels.

dictionary  $\mathbf{A}$  with 256 atoms (as described in Chapter 14), we aim to recover the original patches by approximating the solution of Equation (15.23). This process is performed per patch using the OMP algorithm with  $\delta = 1.1 \cdot \sqrt{64 - p} \cdot \sigma$  (the value of  $\delta$  should be proportional to the amount of noise that exists in the *remaining* samples of the patch). The final inpainted image is created by reconstructing each of these patches from its sparse representation, and placing them back onto the image canvas.<sup>6</sup>

Figure 15.11 displays the inpainting results, as a graph of the reconstruction RMSE per-pixel as a function of the quantity of remaining pixels in the image. As expected, this graph descends, indicating that the quality of the recovered patches improves as more pixels are available. Also, when compared to the noise standard-deviation  $\sigma = 20$ , this graph shows that beyond recovering the missing pixels, this procedure has a denoising effect as well for less than 50% missing pixels. Figure 15.12 presents the inpainted images obtained using the above method for the three tests (removing 25%, 50%, and 75% of the pixels). The root-MSE values computed between the recovered images and the original one are 14.55, 19.61, and 29.70 grey-values respectively.

The above is a simple inpainting algorithm, being a direct implementation of the very core ideas presented above, and yet, its results are reasonable (and especially so for less than 50% missing pixels). Still, this method could and should be improved,

<sup>6</sup> We also add a clipping operation that forces the resulting image to have grey-values in the range  $[0, 255]$ .

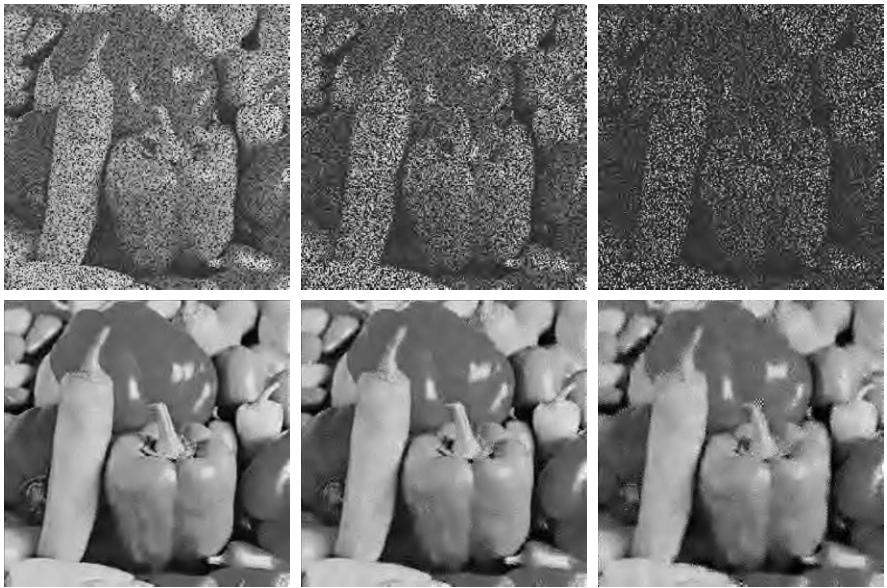


**Fig. 15.12** Simple inpainting results – operating locally with no overlaps and using the redundant DCT dictionary: This figure show the given images (top) and their inpainted results (bottom), for 25% missing pixels (left, RMSE= 14.55), 50% missing pixels (center, RMSE= 19.61), and 75% missing pixels (right, RMSE= 29.7).

and this can be done in various ways. Could we inpaint more complex images with texture? Could we handle larger holes in the image? Can we imitate the local denoising algorithms and gain in performance by operating on overlapping patches? Can we adapt the dictionary to the image and to the inpainting goals? These questions bring us to the next subsections, where we discuss local and then global image inpainting algorithms of better quality that rely on the above ideas.

### 15.3.2 Inpainting Images – Local K-SVD

In [Figures 15.11](#) and [15.12](#) we presented the results of a simple inpainting algorithm, that operates on each patch in the image separately, and concatenates the results to form the full image. Can we do better, while still operating locally? [Figure 15.13](#) shows that a substantial improvement is within reach by a simple modification of the algorithm: Instead of processing non-overlapped patches, we perform the same inpainting process on fully overlapped patches. The resulting image is obtained by averaging the recovered patches in the resulting image canvas. As can be seen, the results are much improved, and especially so for the case where 75% of the pixels



**Fig. 15.13** Improved inpainting results – operating locally with full-overlaps and using the redundant DCT dictionary: This figure shows the given images (top) and their inpainting results (bottom) using the redundant DCT dictionary, for 25% missing pixels (left, RMSE= 9.00), 50% missing pixels (center, RMSE= 11.55), and 75% missing pixels (right, RMSE= 18.18).

are missing. [Table 15.1](#) summarizes these results versus those shown above for the non-overlapping treatment.

Algorithm	RMSE for 25% missing pixels	RMSE for 50% missing pixels	RMSE for 75% missing pixels
No-overlaps	14.55	19.61	29.70
Full-overlaps	9.00	11.55	18.18

**Table 15.1** RMSE for the inpainting tests applied locally on patches with and without overlaps, all using the redundant DCT dictionary.

Let us formulate the inpainting problem in a way that leads to the above algorithm in a natural way. Following our treatment for the denoising problem in Chapter 14, let us 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 known dictionary  $\mathbf{A}$ . We shall denote the mask that removes samples from  $\mathbf{y}_0$  as a rectangular matrix  $\mathbf{M}$ . We shall further assume that the remaining pixels are noisy, contaminated by white zero-mean additive Gaussian noise with variance  $\sigma^2$ . Thus, the MAP estimator would be

$$\begin{aligned} \{\hat{\mathbf{q}}_k\}_{k=1}^M, \hat{\mathbf{y}}\} = \arg \min_{\mathbf{z}, \{\mathbf{q}_k\}_k} \lambda \|\mathbf{M}\mathbf{z} - \mathbf{y}\|_2^2 + \\ \sum_k \mu_k \|\mathbf{q}_k\|_0 + \sum_k \|\mathbf{A}\mathbf{q}_k - \mathbf{R}_k\mathbf{z}\|_2^2. \end{aligned} \quad (15.24)$$

This expression parallels the one posed in Equation (14.21), requiring a proximity between the measured image  $\mathbf{y}$  and the unknown version  $\mathbf{z}$ , only in the existing pixels (this is reflected by the multiplication by the mask  $\mathbf{M}$ ). 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 bounded error.

In the case discussed above, the underlying dictionary  $\mathbf{A}$  is assumed known, being the redundant DCT. We are supposed to minimize this function with respect to both the sparse representations  $\mathbf{q}_k$ , and the overall output image  $\mathbf{z}$ . As done before, we adopt a block-coordinate minimization algorithm that starts with an initialization  $\mathbf{z} = \mathbf{M}^T\mathbf{y}$ , and then seeks the optimal  $\hat{\mathbf{q}}_k$ . Note, however, that in choosing this initialization, the pixels in  $\mathbf{z}$  that correspond to missing values are totally corrupted, and this should be taken into account in the evaluation of  $\hat{\mathbf{q}}_k$ . We arrive at a complete decoupling of the minimization task into smaller problems of the form

$$\hat{\mathbf{q}}_k = \arg \min_{\mathbf{q}} \|\mathbf{q}\|_0 \quad \text{subject to} \quad \|\mathbf{M}_k(\mathbf{A}\mathbf{q} - \mathbf{p}_k)\|_2^2 \leq cn_k\sigma^2. \quad (15.25)$$

First, notice that we have turned the quadratic penalty in Equation (15.24) into a constraint, thus removing the need to choose the parameter  $\mu_k$ . Also, the energy of the patch-error  $\mathbf{A}\mathbf{q} - \mathbf{p}_k$  is evaluated using only the existing pixels in this patch, as indicated by the multiplication by  $\mathbf{M}_k = \mathbf{R}_k\mathbf{M}^T\mathbf{M}\mathbf{R}_k^T$  — a local mask that corresponds to the  $k$ -th patch.

The threshold to compare with must also take into account the count of existing pixels in this patch,  $n_k = \mathbf{1}^T\mathbf{M}_k\mathbf{1}$ . Thus, this stage works as a sliding window sparse coding stage, operated on each block of size  $\sqrt{n} \times \sqrt{n}$  at a time, performing the very same process as described in Equation (15.23).<sup>7</sup>

Given all  $\hat{\mathbf{q}}_k$ , we can now fix those and turn to update  $\mathbf{z}$ . Returning to Equation (15.24), we need to solve

$$\hat{\mathbf{y}} = \arg \min_{\mathbf{z}} \lambda \|\mathbf{M}\mathbf{z} - \mathbf{y}\|_2^2 + \sum_k \|\mathbf{A}\hat{\mathbf{q}}_k - \mathbf{R}_k\mathbf{z}\|_2^2, \quad (15.26)$$

which leads to

$$\hat{\mathbf{y}} = \left( \lambda \mathbf{M}^T\mathbf{M} + \sum_k \mathbf{R}_k^T\mathbf{R}_k \right)^{-1} \left( \lambda \mathbf{M}^T\mathbf{y} + \sum_k \mathbf{R}_k^T\mathbf{A}\hat{\mathbf{q}}_k \right). \quad (15.27)$$

<sup>7</sup> The formulation of the two problems might seem different, because  $\mathbf{M}_k$  here is square, *zeroing* the unknown elements, whereas  $\mathbf{M}$  in Equation (15.23) is rectangular, *removing* these elements. Beyond this semantic difference, the two formulations are the same.



This expression is almost identical with what we have proposed above for the denoising problem, and in particular, the matrix to invert is diagonal, introducing normalization factors per each of the pixels in the resulting image. For missing pixels, this expression suggests a simple averaging of the contributions of the recovered patches directly. For existing pixels, this formula leads to an averaging as well, but one that takes into account the measured values with a proper weight. Thus, the results shown in [Figure 15.13](#) and [Table 15.1](#) correspond to this very algorithm with  $\lambda$  set to be zero.

The natural next step would be to adapt the dictionary to the image using a dictionary-learning approach. Following the K-SVD image denoising algorithm as presented in Chapter 14, we could suggest the following process:

1. We should extract all possible patches with full-overlaps from the corrupted image, and train a dictionary on these. As opposed to the denoising algorithm, the sparse-coding and the dictionary update stages to be performed within this process must take into account the mask of the missing pixels.
2. Once the dictionary is computed, we should sparse-code each of the patches, and place them back to the image canvas, averaging overlaps, just as done in the above method.

This algorithm is nothing but a minimization of the MAP function posed in Equation (15.25), this time with respect to  $\mathbf{A}$  as well. Fixing  $\mathbf{A}$ , we perform sparse coding exactly as described above. Once the sparse representations  $\{\hat{\mathbf{q}}_k\}_k$  are computed, they are kept fixed, and the dictionary is updated, such that the expression

$$Error(\mathbf{A}) = \sum_k \|\mathbf{M}_k(\mathbf{A}\mathbf{q}_k - \mathbf{p}_k)\|_2^2 \quad (15.28)$$

is minimized. This can be done using either an MOD or K-SVD update steps. Both are more challenging in this case, as compared to the core description in Chapter 12, and this is due to the mask matrices, which require a closer attention. The MOD step is obtained by nulling the derivative of  $Error(\mathbf{A})$ , resulting with the expression

$$\nabla_{\mathbf{A}} Error(\mathbf{A}) = \sum_k \mathbf{M}_k^T \mathbf{M}_k (\mathbf{A}\mathbf{q}_k - \mathbf{p}_k) \mathbf{q}_k^T = 0, \quad (15.29)$$

from which it is hard to obtain a closed-form analytic formula for the update of  $\mathbf{A}$ . Exploiting properties of the Kronecker-product, we can use the formula  $\mathbf{ABC} = (\mathbf{C}^T \otimes \mathbf{A})\mathbf{B}_{CS}$ , where  $\mathbf{B}_{CS}$  is a vector obtained by a lexicographic ordering of the columns of  $\mathbf{B}$ . This leads to

$$\mathbf{A}_{CS} = \left[ \sum_k (\mathbf{q}_k \mathbf{q}_k^T) \otimes (\mathbf{M}_k^T \mathbf{M}_k) \right]^{-1} \sum_k (\mathbf{p}_k \mathbf{q}_k^T)_{CS}. \quad (15.30)$$

In our case, the matrix to be inverted is of size  $mn \times mn$ . For example, for a dictionary of size  $64 \times 256$  as in the above experiments, this matrix size is  $2^{14} \times 2^{14}$ . Thus, a

direct use of this equation for deriving  $\mathbf{A}$  is prohibitive in most cases. The alternative is to use the gradient in Equation (15.29) within a steepest decent algorithm.

The K-SVD counterpart is better behaved. We target the update of the columns of  $\mathbf{A}$  one at a time. Considering the  $j$ -th column,  $\mathbf{a}_j$ , we should address the error expression in Equation (15.28) only for patches  $k$  that use this atom. Denoting this set by  $\Omega_j$ , we thus minimize

$$\begin{aligned} \text{Error}(\mathbf{A}) &= \sum_{k \in \Omega_j} \|\mathbf{M}_k(\mathbf{A}\mathbf{q}_k - \mathbf{p}_k)\|_2^2 \\ &= \sum_{k \in \Omega_j} \|\mathbf{M}_k(\mathbf{A}\mathbf{q}_k - \mathbf{a}_j\mathbf{q}_k(j) - \mathbf{p}_k) + \mathbf{M}_k\mathbf{a}_j\mathbf{q}_k(j)\|_2^2. \end{aligned} \quad (15.31)$$

We denote  $\mathbf{y}_k^j = \mathbf{p}_k - \mathbf{A}\mathbf{q}_k + \mathbf{a}_j\mathbf{q}_k(j)$ . This is the residual in the representation of the  $k$ -th patch, where all the atoms apart from the  $j$ -th are used. Thus, our task would be

$$\min_{\mathbf{a}_j, \{\mathbf{q}_k(j)\}_{k \in \Omega_j}} \sum_{k \in \Omega_j} \|\mathbf{M}_k(\mathbf{y}_k^j - \mathbf{a}_j\mathbf{q}_k(j))\|_2^2. \quad (15.32)$$

While the above could be posed as a rank-1 approximation problem (that ties it to SVD), a simpler approach is a short iterative (2-3 iterations) process where we update the two unknowns alternately. We fix  $\{\mathbf{q}_k(j)\}_{k \in \Omega_j}$  and update  $\mathbf{a}_j$ , by

$$\hat{\mathbf{a}}_j = \left[ \sum_{k \in \Omega_j} \mathbf{M}_k\mathbf{q}_k(j)^2 \right]^{-1} \sum_{k \in \Omega_j} \mathbf{M}_k\mathbf{y}_k^j\mathbf{q}_k(j), \quad (15.33)$$

where we exploited the fact that  $\mathbf{M}_k^T\mathbf{M}_k = \mathbf{M}_k$ . Once computed, we should normalize this vector. The update of  $\mathbf{q}_k(j)$  is obtained by

$$\hat{\mathbf{q}}_k(j) = [\mathbf{a}_j^T\mathbf{M}_k\mathbf{a}_j]^{-1} \mathbf{a}_j^T\mathbf{M}_k\mathbf{y}_k^j, \quad (15.34)$$

computed for all  $k \in \Omega_j$ .

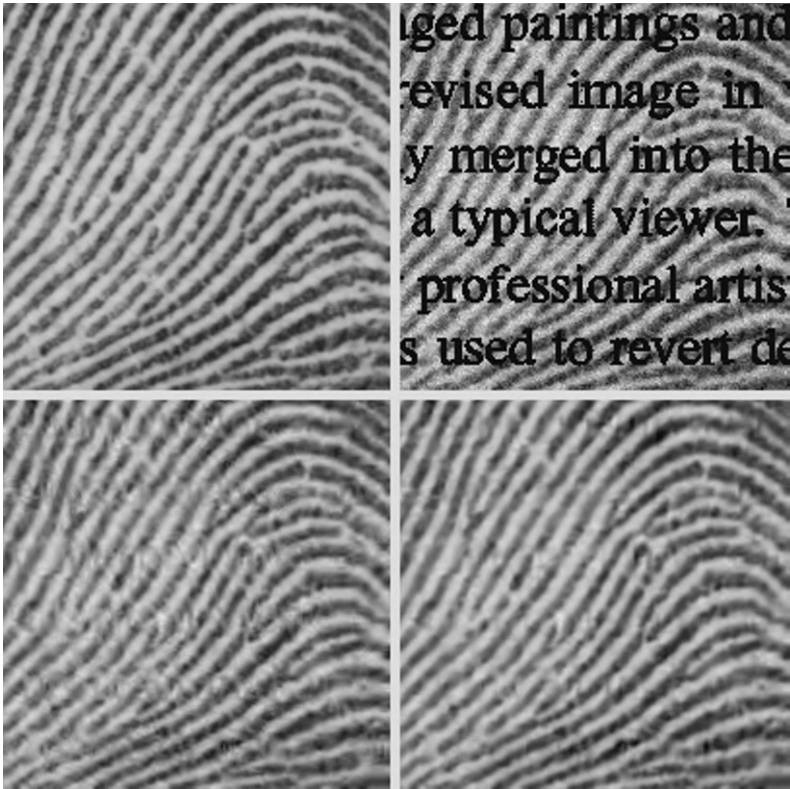
Table 15.2 presents the results of such K-SVD algorithm on the same tests performed above, with 15 iterations of this sparse-coding – dictionary update alternating algorithm. These experiments indicate that a substantial gain in performance can be obtained, and even more so if the initial dictionary is non-ideal for the inpainting task. The image *Peppers* seems to agree with the redundant DCT, thus leaving a smaller room for an improvement.

Figures 15.14, 15.15, and 15.16 present the results of a similar experiment, performed on the image *Fingerprint*, with a mask that consists of carved text. As before, we start with the redundant DCT dictionary and obtain  $RMSE = 16.13$  grey-values. After 15 iterations of the K-SVD inpainting algorithm, the result improves by 2.05dB to  $RMSE = 12.74$  grey-values. Figure 15.14 shows the original image, the given noisy image (with  $\sigma = 20$ ), and these two results. Figure 15.15 shows the absolute error images for these two results, and Figure 15.16 presents the



Algorithm	RMSE for 25% missing pixels	RMSE for 50% missing pixels	RMSE for 75% missing pixels
DCT: No-overlaps	14.55	19.61	29.70
DCT: Full-overlaps	9.00	11.55	18.18
K-SVD: Full-overlaps	8.1 (0.85dB)	10.05(1.25dB)	17.74(0.15dB)

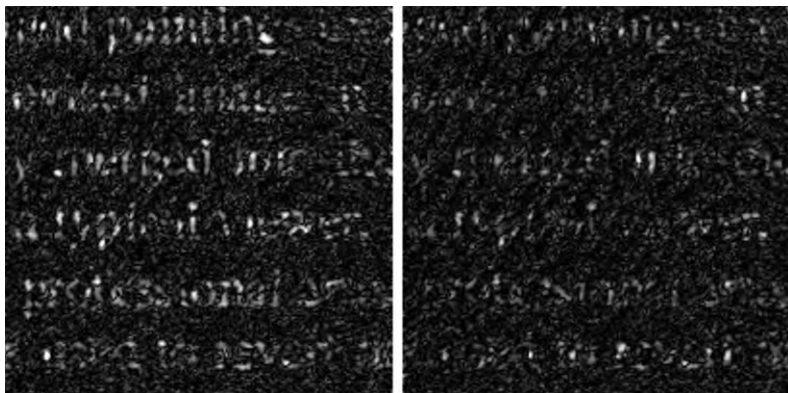
**Table 15.2** RMSE for the K-SVD inpainting algorithm on Peppers with 15 iterations. The results in the first two rows are the ones presented in Table 15.1. The last row gives the K-SVD results, both in terms of the RMSE and the gain achieved[dB].



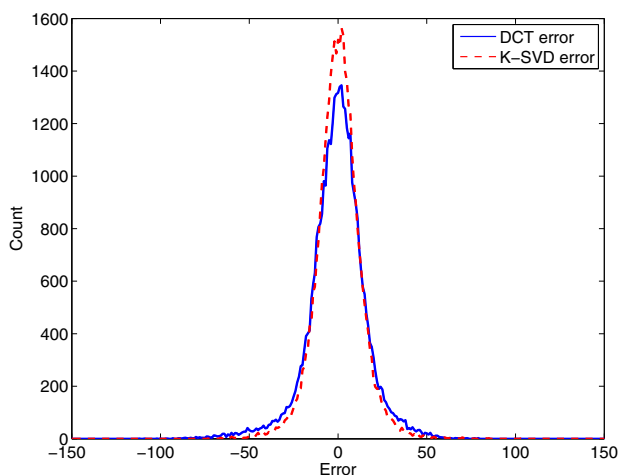
**Fig. 15.14** Top left: The original Fingerprint image (portion of  $200 \times 200$  pixels); Top Right: The measured image with the additive noise ( $\sigma = 20$ ) and the missing pixels (text mask); Bottom left: Redundant DCT inpainting result ( $RMSE = 16.13$ ); Bottom right: K-SVD (15 iterations) result ( $RMSE = 12.74$ ).

histogram of these two errors, indicating that the K-SVD error is more concentrated around the origin.

The ability to inpaint an image using local and simple operations, as described above, has been extended by Mairal. et. al. to video and color images. We bring here



**Fig. 15.15** Redundant DCT absolute error (left), and K-SVD absolute error (right).



**Fig. 15.16** Histogram of the two error images – the redundant DCT and the K-SVD ones.

two examples that illustrate these capabilities, and we point the interested reader to the papers by Mairal, Elad, and Sapiro for more details.

Figure 15.17 presents an inpainting result obtained on a color image. The red text represents missing pixels, and these are recovered using the K-SVD inpainting. The patches used in this experiment are 3D ones (of size  $9 \times 9 \times 3$ , chosen large enough to fill-in the holes), containing the three color layers within each patch. This also means that the trained dictionary has color atoms.

Figure 15.18 shows the results obtained for video inpainting. The sequence Table-Tennis (images of size  $240 \times 352$ ) has been corrupted by randomly erasing 80% of its pixels. This figure shows 5 frames from the original sequence, the



**Fig. 15.17** Inpainting result using the local-K-SVD algorithm and taking into account the color information (this result is taken from the work by Mairal, Elad, and Sapiro (2008)). Top: the original image, Center: the degraded image with red text representing missing pixels, Bottom: the recovered image (PSNR= 32.45dB).

corrupted frames, and the recovered results. The results include inpainting of images independently, as described in this chapter, and video inpainting, where the spatio-temporal volume is treated as a 3D image. Thus, 3D patches are used for the inpainting, and the dictionary is propagated from one frame to the next, as described in Chapter 14 for the K-SVD video denoising. Another important modification introduced in this experiment is the use of a multi-scale dictionary, implying that the atoms are structured in several scales.

### 15.3.3 Inpainting Images – The Global MCA

In the section above we treated the image inpainting problem by operating on small patches. Just as with the image denoising or image separation tasks, this can be replaced with a globally operating algorithm, that transforms the complete image. In this section we briefly describe this option, and show several representative results to illustrate it.

An important benefit of the migration to a global treatment, is the ability to handle larger holes in the image. On the down side, we leave aside the notion of trained dictionaries, and use instead fixed ones. This also means that, as opposed to trained dictionaries that may contain both texture and cartoon atoms, the fixed dictionaries should be chosen such that they accommodate these two types of image content. This ties us back to the image separation discussion given above.

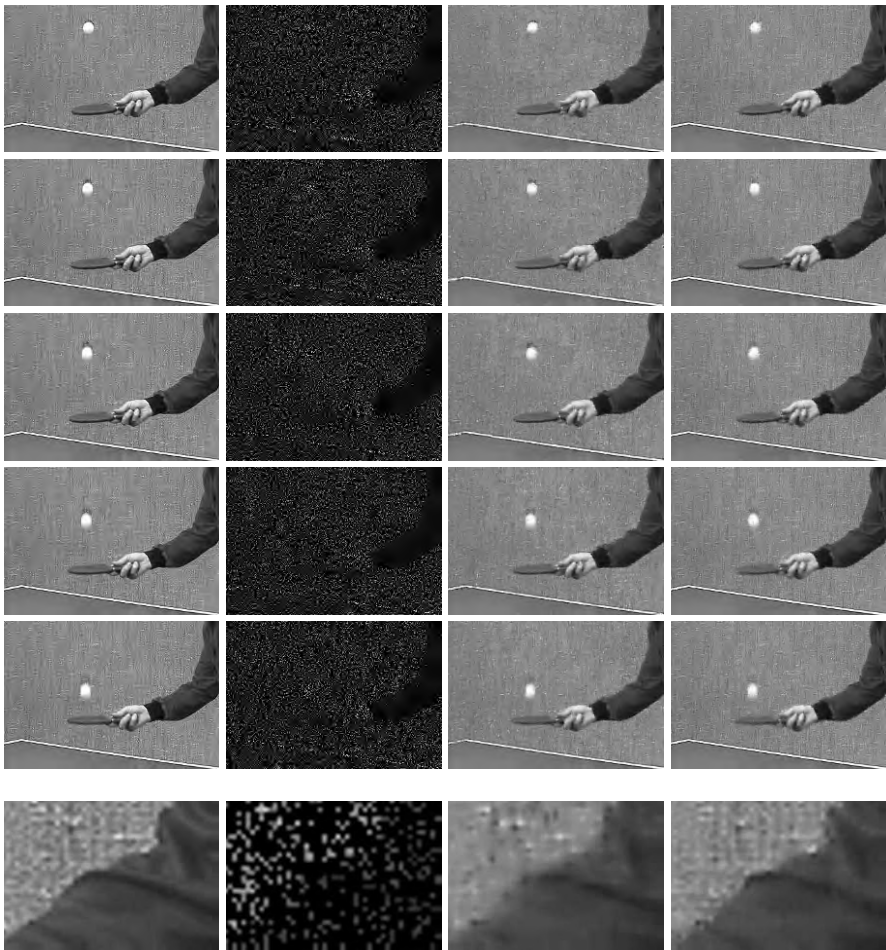
Indeed, when we discussed the task of image separation to cartoon and texture in Section 15.2.1, we mentioned that inpainting is an application that may find this separation useful. As we show next, the framework we adopt to describe the global inpainting algorithm is the very same one used for the global separation of images, with a simple modification.

We start our treatment of the global inpainting problem by using the analysis-based image separation formulation given in Equation (15.5). We modify this to give the following objective, which takes into account the mask operator  $\mathbf{M}$ ,

$$\begin{aligned} \hat{\mathbf{y}}_c, \hat{\mathbf{y}}_t = \arg \min_{\mathbf{y}_c, \mathbf{y}_t} & \lambda \|\mathbf{T}_c \mathbf{y}_c\|_1 + \lambda \|\mathbf{T}_t \mathbf{y}_t\|_1 \\ & + \frac{1}{2} \|\mathbf{y} - \mathbf{M}(\mathbf{y}_c + \mathbf{y}_t)\|_2^2. \end{aligned} \quad (15.35)$$

Similar to the conversion done to a synthesis-like formulation in Equation (15.9), we propose the alternative optimization problem

$$\begin{aligned} \hat{\mathbf{x}}_c, \hat{\mathbf{x}}_t = \arg \min_{\mathbf{x}_c, \mathbf{x}_t} & \lambda \|\mathbf{x}_c\|_1 + \lambda \|\mathbf{x}_t\|_1 \\ & + \frac{1}{2} \|\mathbf{y} - \mathbf{M} \mathbf{A}_c \mathbf{x}_c - \mathbf{M} \mathbf{A}_t \mathbf{x}_t\|_2^2 \\ \text{Subject to } & \mathbf{T}_c \mathbf{A}_c \mathbf{x}_c = \mathbf{x}_c \text{ and } \mathbf{T}_t \mathbf{A}_t \mathbf{x}_t = \mathbf{x}_t. \end{aligned} \quad (15.36)$$



**Fig. 15.18** Results obtained with the proposed multiscale K-SVD video inpainting for 5 frames of the table-tennis sequence. Each row corresponds to one frame, with the following from left to right: the original frame, the same frame with 80% missing data, the results obtained when applying the image inpainting algorithm to each frame separately (PSNR=24.38dB), and the result of the video inpainting K-SVD ( PSNR=28.49dB). The last row shows a zoomed version of the last frame.



The only difference between this and Equation (15.9) is the presence of the mask operator. Thus, the very same numerical algorithm can be proposed for solving (15.36), with a slight modification. Any appearance of  $\mathbf{A}_c$  (or  $\mathbf{A}_t$ ) in the SSF formula should be replaced with  $\mathbf{MA}_c$  ( $\mathbf{MA}_t$ , correspondingly). Thus,

$$\begin{aligned}\hat{\mathbf{x}}_c^{k+\frac{1}{2}} &= \mathcal{S}_\lambda \left( \frac{1}{c} \mathbf{A}_c^T \mathbf{M}^T (\mathbf{y} - \mathbf{MA}_c \hat{\mathbf{x}}_c^k - \mathbf{MA}_t \hat{\mathbf{x}}_t^k) + \hat{\mathbf{x}}_c^k \right) \\ \hat{\mathbf{x}}_t^{k+\frac{1}{2}} &= \mathcal{S}_\lambda \left( \frac{1}{c} \mathbf{A}_t^T \mathbf{M}^T (\mathbf{y} - \mathbf{MA}_c \hat{\mathbf{x}}_c^k - \mathbf{MA}_t \hat{\mathbf{x}}_t^k) + \hat{\mathbf{x}}_t^k \right),\end{aligned}$$

The projection stage that follows should not be touched, as it is operating on the complete representation vectors,

$$\hat{\mathbf{x}}_c^{k+1} = \mathbf{T}_c \mathbf{A}_c \hat{\mathbf{x}}_c^{k+\frac{1}{2}} \quad \text{and} \quad \hat{\mathbf{x}}_t^{k+1} = \mathbf{T}_t \mathbf{A}_t \hat{\mathbf{x}}_t^{k+\frac{1}{2}}.$$

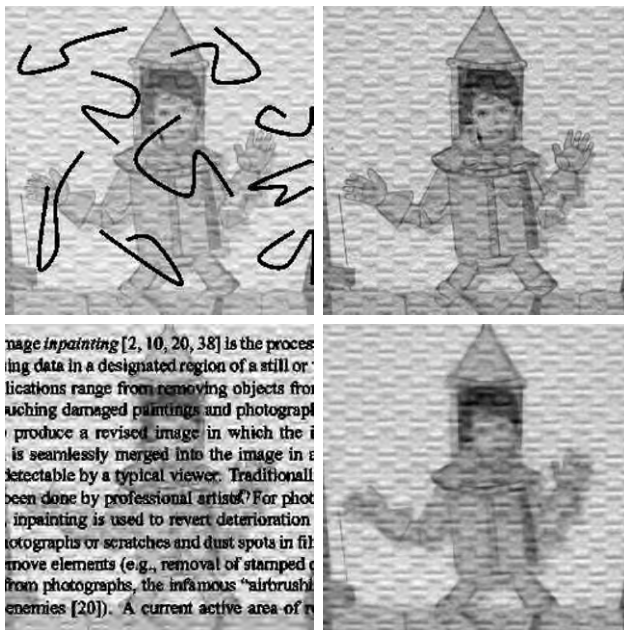
We turn now to show several results of the global inpainting method, taken from the paper by Elad, Starck, Querre, and Donoho (2005). We start with a synthetic experiment, where we compose the image Adar as a combination of texture and cartoon. Figure 15.19 shows this image Adar with two cases of missing data. The results of the global inpainting method using curvelets and global DCT dictionaries are shown in this figure as well. Both results show no trace of the original holes, and look near-perfect.

Figure 15.20 shows the results of a similar experiment, where the existing pixels in the image are also contaminated by a zero-mean white Gaussian noise ( $\sigma = 10$ ). As can be seen, the residual is almost featureless, implying that the noise was removed successfully, without absorbing true texture of cartoon content.

The results shown in Figure 15.21 correspond to an experiment similar to the one shown in Figure 15.19, but on the image Barbara, which is already a mixture of cartoon and texture. The global inpainting method applied here uses the Haar wavelet for the cartoon, and a homogeneous decomposition level wavelet packets to represent the texture. Again, we see near-perfect recovery of the missing pixels.

Figure 15.22 presents the Barbara image and its filled-in results for three random patterns of 20%, 50%, and 80% missing pixels. The unstructured form of the mask makes the reconstruction task easier. As before, the inpainting method applied here used wavelet and wavelet packets to represent the cartoon and the texture respectively, and again, the results look natural and artifact-free.

The last figure in this section, Figure 15.23, presents a similar experiment on the Barbara image, this time filling in 9 blocks of sizes  $8 \times 8$ ,  $16 \times 16$ , and  $32 \times 32$  pixels. Filling-in large holes is more challenging, compared to the random mask. We see that as the regions of missing pixels grow, the recovery deteriorates, as expected, and smooth behavior is enforced.



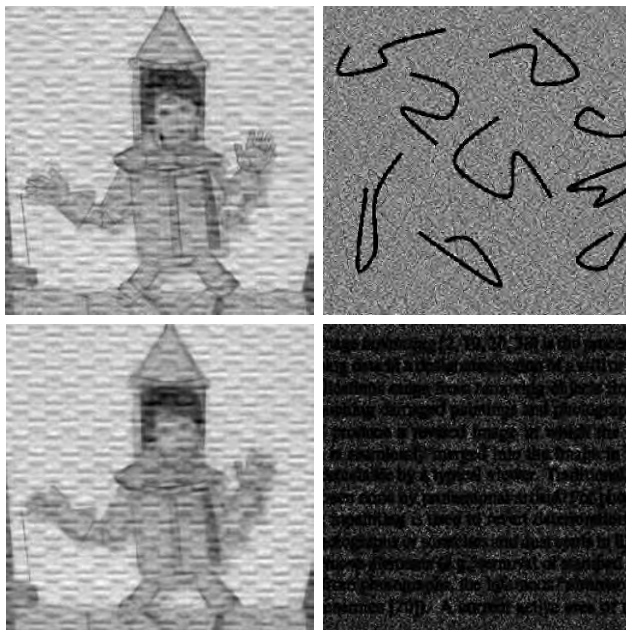
**Fig. 15.19** Two synthetic Adar images (top and bottom left) with combined cartoon and texture, and imposed missing pixels. The results of the global inpainting algorithm are given in the top and bottom right.

### 15.3.4 Impulse-Noise Filtering

We conclude our discussion on the image inpainting problem by introducing another application related to it – impulsive noise removal. The objective of this section is to show the fundamental ability of the above inpainting algorithms to handle this noise removal problem.

The topic of denoising has been discussed in length in Chapter 14, but with an emphasis on white additive Gaussian noise. Another very commonly encountered noise in images is an impulsive noise, where a fraction of the pixels in the image are corrupted by a large negative or positive deviated value. [Figure 15.24](#) presents the original image *Peppers* and a noisy version of it with such a noise. In this case, 10% of the pixels (in random locations) are corrupted. These pixels are modified by adding or subtracting 50 grey-values<sup>8</sup> to their original value. This way, we get dark and bright points on the image, representing the noise. This noise is also commonly referred to as *salt and pepper* noise because of its appearance as salt and pepper grains spread over the image.

<sup>8</sup> The final grey-value is clipped to the range  $[0, 255]$ , and thus the eventual deviation may be smaller than 50 grey-values.



**Fig. 15.20** Global inpainting results with two missing pixels masks – curves (top) and text (bottom)), both contaminated by an additive noise. Left: The inpainting results, Right: the residual.

Salt-and-Pepper noise is typically handled well by the median filter, which replaces every pixel value by the median of its small neighborhood. [Figure 15.25](#) (middle right) shows the result obtained with such a filter, and it is clearly seen that it is working well, leading in this case to an RMSE of 7.42 grey-values. The median filter used here applies a  $3 \times 3$  neighborhood – experiments with larger or smaller sized neighborhood were found to give poorer results.

The question is: could we do better than the median filter? If we knew the location of the noisy pixels, we could have addressed this denoising problem as an inpainting of these as missing pixels. In this section we aim to demonstrate this alternative denoising approach.

Estimating the location of the corrupted pixels can be done by evaluating the difference between the noisy image and its median-filtered version, attributing large differences to noisy pixels. [Figure 15.25](#) (middle left) shows the estimated mask by thresholding this difference with 27 grey-value. This threshold value was found empirically to lead to good performance. The image contains 6,554 corrupted pixels, and the mask detects 6,298 pixels, of which only 5,211 are true-noisy ones. The pixels that were undetected correspond to clipped noisy pixels with small deviation, or others in which the noise is not as evident (near-edge pixels, etc.).

Using the estimated mask, we can proceed with an inpainting process, which aims the recovery of these as missing pixels. The result of a local inpainting al-





**Fig. 15.21** The Barbara image (top and bottom left) with two imposed masks of missing pixels. The results of the global inpainting algorithm are given in the top and bottom right.

gorithm that uses the redundant DCT dictionary is shown in [Figure 15.25](#) (bottom right). The sparse-coding is performed with the OMP algorithm with a small error threshold ( $\sigma = 5$ ). Once done, only the missing pixels are replaced, fixing the rest as correct ones. The RMSE of this method is 5.86, which is clearly better than the median filtering result.

If the estimated mask is indeed reliable, we could use it to improve the median filtering as well, by applying the median filtering only on the corrupted pixels. This result is shown in [Figure 15.25](#) (bottom left), with an RMSE of 6.52. While this gives an improvement, it is clear that the inpainting approach is still superior. [Figure 15.26](#) presents the absolute error images for the three denoising processes discussed above (median filtering, masked median filtering, and the inpainting-based approach). Again, we see that the inpainting approach leads to the smallest error.

We should note that there is an extensive literature on impulsive noise removal, using variational methods, diverse median and order-filtering methods, and more. This literature also offers more elegant and reliable ways to estimate the location of the corrupted pixels. In this section we deliberately restricted the discussion to elementary ideas and algorithms, focusing on the core ability of inpainting algorithms to serve this problem as well.



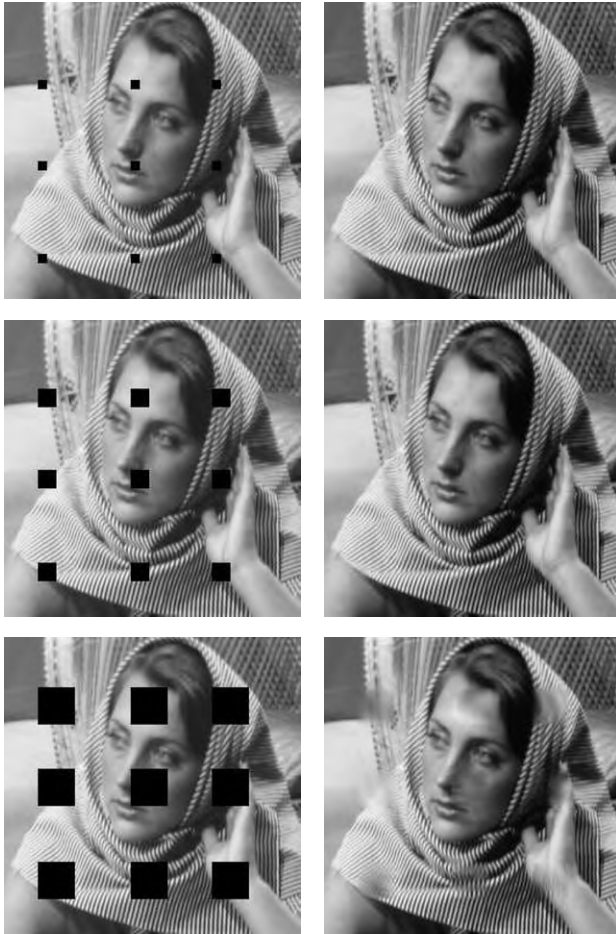
**Fig. 15.22** Three Barbara images with 20%, 50%, and 80% missing pixels (right), and the global inpainting results (left).

## 15.4 Image Scale-Up

We end this chapter with the image scale-up problem, often referred to in the literature as *super-resolution*.<sup>9</sup> Our goal is to up-sample a single image while producing a sharp result. This task can be posed as an inverse problem, where the given low-resolution image is assumed to be the measurement obtained from an original

---

<sup>9</sup> The term *super-resolution* may be confusing, as it is also used in the context of fusing several low-resolution images into one high-resolution result. Here we are referring to a different process of scaling-up a single image.



**Fig. 15.23** Three Barbara images with three patterns of missing pixels – 9 blocks of size  $8 \times 8$ ,  $16 \times 16$ , and  $32 \times 32$  pixels (right), and the global inpainting results (left).

high-resolution image that went through a blur, a decimation, and additive noise effects.

The literature offers various ways to address the image scale-up problem, ranging from simple linear space-invariant interpolation schemes (bilinear, bicubic, Lanczos), to spatially adaptive and non-linear filters of various sorts. *Sparse-Land*-based algorithms for handling this problem also exist, and here as well we can divide the various methods to global and local ones. In this chapter we shall focus on the algorithm proposed by Yang et. al., which operates locally on patches. While we follow Yang's work to large extent, we also introduce several important differences that lead to improved results.



**Fig. 15.24** The original image Peppers and its noisy version, with 10% of the pixels corrupted by an impulsive noise.

### 15.4.1 Modeling the Problem

We start by describing the problem model. The given low-resolution image,  $\mathbf{z}_l$  is created from the high-resolution one,  $\mathbf{y}_h$  by

$$\mathbf{z}_l = \mathbf{S}\mathbf{H}\mathbf{y}_h + \mathbf{v}, \quad (15.37)$$

where  $\mathbf{H}$  and  $\mathbf{S}$  are the blur and decimation operators, respectively. We shall assume that  $\mathbf{S}$  performs a decimation by an integer factor, by discarding rows/columns from the input image. The vector  $\mathbf{v}$  is an additive zero-mean white and homogeneous Gaussian noise, with a known standard-deviation  $\sigma$ .

In order to avoid complexities caused by the different resolutions of  $\mathbf{z}_l$  and  $\mathbf{y}_h$ , we shall assume hereafter that the image  $\mathbf{z}_l$  is scaled-up by a simple interpolation scheme that fills-in the missing rows/columns, returning to the size of  $\mathbf{y}_h$ . This scaled-up image shall be denoted by  $\mathbf{y}_l$ ,

$$\mathbf{y}_l = \mathbf{Q}\mathbf{S}\mathbf{H}\mathbf{y}_h + \mathbf{Q}\mathbf{v} = \mathbf{L}^{all}\mathbf{y}_h + \tilde{\mathbf{v}}. \quad (15.38)$$

The operator  $\mathbf{Q}$  stands for the interpolation scale-up operation. Our goal is to process  $\mathbf{y}_l$  and produce a result  $\hat{\mathbf{y}}_h$ , which will get as close as possible to the original high-resolution image,  $\mathbf{y}_h$ .

The processing we are about to present operates on patches extracted from  $\mathbf{y}_l$ , aiming to estimate the corresponding patch from  $\mathbf{y}_h$ . Let  $\mathbf{p}_k^h = \mathbf{R}_k\mathbf{y}_h \in \mathbb{R}^n$  be a high-resolution image patch of size  $\sqrt{n} \times \sqrt{n}$ , extracted by the operator  $\mathbf{R}_k$  from the image  $\mathbf{y}_h$  in location  $k$ . We shall assume that the locations to consider,  $k$ , are only those centered around true pixels in the low-resolution image  $\mathbf{y}_l$  (as opposed to filled-in pixels due to the interpolation). We shall refer hereafter to this set of samples as  $\Omega$ .

It is now time to invoke the *Sparse-Land* model: We shall further assume that  $\mathbf{p}_k^h$  can be represented sparsely by  $\mathbf{q}_k \in \mathbb{R}^m$  over the dictionary  $\mathbf{A}_h \in \mathbb{R}^{n \times m}$ , namely,  $\mathbf{p}_k^h = \mathbf{A}_h\mathbf{q}_k$ , where  $\|\mathbf{q}_k\|_0 \ll n$ .



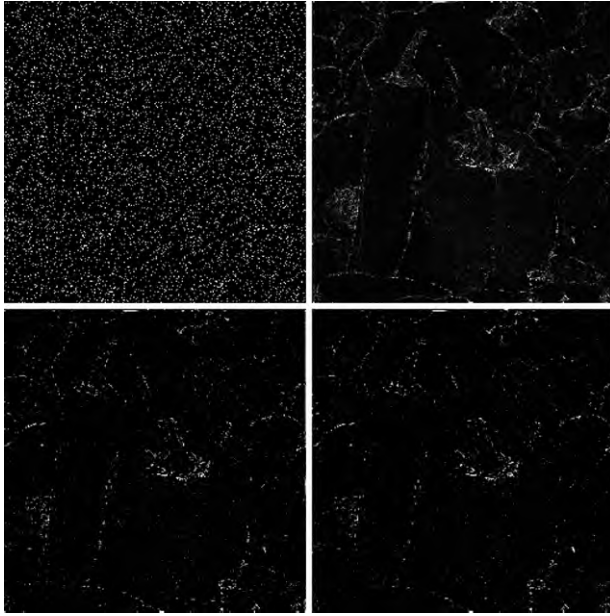
**Fig. 15.25** The original image Peppers (top left), its noisy version (top right), the original image with the estimated mask on top (middle left), and three denoised results obtained with (i) the median filter (middle right,  $RMS E = 7.42$ ), (ii) the median filter applied only on the detected points in the estimated mask (bottom left,  $RMS E = 6.52$ ), and (iii) the redundant DCT inpainting applied with the estimated mask (bottom right,  $RMS E = 5.86$ ).

Consider the corresponding low-resolution patch  $\mathbf{p}_k^l = \mathbf{R}_k \mathbf{y}_l$ , extracted from  $\mathbf{y}_l$  in the same location<sup>10</sup>  $k$ , such that its size is  $\sqrt{n} \times \sqrt{n}$ . Since the operator  $\mathbf{L}^{all} = \mathbf{QSH}$  transforms the complete high-resolution image  $\mathbf{y}_h$  to the low-resolution one,  $\mathbf{y}_l$ , we can assume that  $\mathbf{p}_k^l = \mathbf{L} \mathbf{p}_k^h + \tilde{\mathbf{v}}_k$ , where  $\mathbf{L}$  is a local operator being a portion of  $\mathbf{L}^{all}$ , and  $\tilde{\mathbf{v}}_k$  is the additive noise in this patch. Note that  $\mathbf{L}$  is a spatially independent operator, as we consider only locations  $k \in \mathcal{Q}$  from  $\mathbf{y}_l$ .

Since we have assumed that  $\mathbf{p}_k^h = \mathbf{A}_h \mathbf{q}$ , multiplication of this equation by  $\mathbf{L}$  gives

<sup>10</sup> We assume that  $\mathbf{p}_k^l$  and  $\mathbf{p}_k^h$  are centered around the same pixel.





**Fig. 15.26** The absolute error images between the original image and: (i) the noisy image (top left), (ii) the median result (top right), (iii) the masked median filtered image (bottom left), and (iv) the inpainted result (bottom right).

$$\mathbf{L}\mathbf{p}_k^h = \mathbf{L}\mathbf{A}_h\mathbf{q}. \quad (15.39)$$

Exploiting the relation between the low-resolution and the high-resolution patches,  $\mathbf{p}_k^l = \mathbf{L}\mathbf{p}_k^h + \tilde{\mathbf{v}}_k$ , we thus get

$$\mathbf{L}\mathbf{A}_h\mathbf{q} = \mathbf{L}\mathbf{p}_k^h = \mathbf{p}_k^l - \tilde{\mathbf{v}}_k \quad (15.40)$$

implying that

$$\|\mathbf{p}_k^l - \mathbf{L}\mathbf{A}_h\mathbf{q}\|_2 \leq \epsilon, \quad (15.41)$$

where  $\epsilon$  is related to the noise power  $\sigma$  of  $\tilde{\mathbf{v}}$ .

The key observation from the above derivations is that the low-resolution patch should be represented by the same sparse vector  $\mathbf{q}$  over the effective dictionary  $\mathbf{A}_l = \mathbf{L}\mathbf{A}_h$ , with a controlled error  $\epsilon_l$ . This implies that for a given low-resolution patch  $\mathbf{p}_k^l$ , we should find its sparse representation vector,  $\mathbf{q}$ , and then we can recover  $\mathbf{p}_k^h$  by simply multiplying this representation by the dictionary  $\mathbf{A}_h$ . This is the core idea behind the image scale-up algorithm as developed by Yang et. al, as we are about to present.

### 15.4.2 The Super-Resolution Algorithm

The scale-up algorithm we are about to present consists of a training phase that includes the following steps:

1. **Patch-Pairs Construction:** For a set of training images that are given in high and low-resolution pairs  $\{\mathbf{y}_h^j, \mathbf{y}_l^j\}_j$ , we extract pairs of matching patches that form the training database,  $\mathcal{P} = \{\mathbf{p}_k^h, \mathbf{p}_k^l\}_j$ . Each of these patch-pairs undergoes a pre-processing stage that removes the low-frequencies from  $\mathbf{p}_k^h$  and extracts features from  $\mathbf{p}_k^l$ .
2. **Dictionary Training:** A dictionary  $\mathbf{A}_l$  is trained for the low-resolution patches, such that they can be represented sparsely. A corresponding dictionary  $\mathbf{A}_h$  is constructed for the high-resolution patches, such that it matches the low-resolution one.

The above training phase is done off-line, producing the two dictionaries,  $\mathbf{A}_l$  and  $\mathbf{A}_h$ , to be used in the super-resolution reconstruction.

Given a test low-resolution image  $\mathbf{y}_l$  to be scaled-up (recall that  $\mathbf{y}_l$  is already interpolated to the destination size, and all we need to perform is a spatial non-linear filtering that sharpens this image), we extract pre-processed patches  $\mathbf{p}_k^l$  from each location  $k \in \Omega$ , and then sparse-code it using the trained dictionary  $\mathbf{A}_l$ . The found representation  $\mathbf{q}_k$  is then used to recover the high-resolution patch by multiplying it with  $\mathbf{A}_h$ . We now turn to describe all the above process in more details.

#### 15.4.2.1 Training Phase

The training phase starts by collecting several images  $\{\mathbf{y}_h^j\}_j$ , which are the high-resolution examples. Each of these images is blurred and down-scaled by a factor  $s$ . This leads to the formation of the corresponding low-resolution images  $\{\mathbf{z}_l^j\}_j$ , which are then scaled up back to the original size, resulting with the set  $\{\mathbf{y}_l^j\}_j$ . Thus,

$$\mathbf{y}_l^j = \mathbf{L}_{all} \mathbf{y}_h^j + \tilde{\mathbf{v}}^j.$$

It is important to note that the same operator  $\mathbf{L}_{all}$  should be used both in the training and the testing phases.

The next step to perform is pre-processing. Rather than extracting small image-patches and applying this step on them, we employ the desired pre-processing directly on the full images, and only then extract the patches. This avoids boundary problems due to the small patch size.<sup>11</sup>

The pre-processing applied on the high-resolution images consists of a removal of their low-frequencies. This is done by computing the difference images

<sup>11</sup> A patch of size  $\sqrt{n} \times \sqrt{n}$  in  $\mathbf{y}_l$  should correspond to a larger patch in  $\mathbf{y}_h$ , because of the spatial extent of the blur and the scale-up operations. Nevertheless, this additional “band” of pixels can be disregarded, as if we concentrate on predicting only the center of the destination patch from  $\mathbf{y}_h$ .

$\mathbf{e}_h^j = \mathbf{y}_h^j - \mathbf{y}_l^j$ . The reason for this step is the desire to focus the training on characterizing the relation between the low-resolution patches, and the edges and texture content within the corresponding high-resolution patches. As for the pre-processing of the low-resolution images, these are filtered using  $K$  high-pass filters, in order to extract local features that correspond to their high-frequency content. Thus, each low-resolution image  $\mathbf{y}_l^j$  leads to a set of  $K$  filtered images,  $f_k \otimes \mathbf{y}_l^j$  for  $i = 1, 2, \dots, K$  ( $\otimes$  stands for a convolution).

After the two pre-processing steps described above, we are ready to extract the local patches, and form the data-set  $\mathcal{P} = \{\mathbf{p}_k^h, \mathbf{p}_k^l\}_k$ . Considering only locations  $k \in \Omega$ , we extract patches of size  $\sqrt{n} \times \sqrt{n}$  pixels from the high-resolution images  $\mathbf{e}_h^j$ . The corresponding low-resolution patches are extracted from the same locations in the filtered images  $f_k \otimes \mathbf{y}_l^j$  and using the same size ( $\sqrt{n} \times \sqrt{n}$  pixels). Thus, every corresponding  $K$  such low-resolution patches are merged into one vector  $\tilde{\mathbf{p}}_k^l$  of length  $nK$ . Note that the patch size should be at least of size  $s \times s$  so as to cover the high-resolution image. If the patch size is larger, we get overlaps that improve the reconstruction result.

The last step before turning to the dictionary-learning stage is to reduce the dimension of the input low-resolution patches,  $\{\tilde{\mathbf{p}}_k^l\}_k$ . We apply the Principal-Component-Analysis (PCA) algorithm on these vectors, seeking a subspace on which we could project these patches while preserving 99.9% of their average energy. The motivation for this step is the knowledge that the low-resolution patches could not occupy the full  $nK$  space, since they are effectively built of patches of size  $\sqrt{n}/s \times \sqrt{n}/s$  in the low-resolution image  $\mathbf{z}_l^j$  that go through a set of  $K$  linear filters. The scaling up and the filtering are not expected to increase the dimensionality of the resulting patches. By such a dimensionality reduction we expect to save computations in the subsequent training and super-resolution algorithms. We denote by  $\mathbf{B} \in \mathbb{R}^{n_l \times nK}$  the projection operator that transforms the patch  $\tilde{\mathbf{p}}_k^l$  to its reduced feature vector,  $\mathbf{p}_k^l = \mathbf{B}\tilde{\mathbf{p}}_k^l \in \mathbb{R}^{n_l}$ .

Turning to the dictionary-learning part in the training, we start with the low-resolution training patches  $\{\mathbf{p}_k^l\}_k$ . We apply a regular MOD or K-SVD training procedure for these patches, resulting with the dictionary  $\mathbf{A}_l \in \mathbb{R}^{n_l \times m}$ . Within this learning procedure, sparse-coding is performed using the OMP with a fixed number of atoms per example,  $L$ . As a side product of this training, we obtain the sparse representation vector  $\mathbf{q}_k$  that corresponds to the training patch  $\mathbf{p}_k^l$ .

After constructing  $\mathbf{A}_l$ , we proceed to the high-resolution dictionary construction. Recall that it is our intention to recover the patch  $\mathbf{p}_k^h$  by approximating it as being  $\mathbf{p}_k^h \approx \mathbf{A}_h \mathbf{q}_k$ , namely, use the found sparse representation vector for the low-resolution patch, and multiply it by the high-resolution dictionary. Thus, we should seek a dictionary  $\mathbf{A}_h$  such that this approximation is as exact as it can be. Thus, we define this dictionary to be the solver of the problem

$$\begin{aligned} \mathbf{A}_h &= \arg \min_{\mathbf{A}_h} \sum_k \|\mathbf{p}_k^h - \mathbf{A}_h \mathbf{q}_k\|_2^2 \\ &= \arg \min_{\mathbf{A}_h} \|\mathbf{P}_h - \mathbf{A}_h \mathbf{Q}\|_F^2, \end{aligned} \quad (15.42)$$



where we have constructed the matrix  $\mathbf{P}_h$ , such that the high-resolution training patches,  $\{\mathbf{p}_k^h\}_k$  from its columns, and similarly,  $\mathbf{Q}$  contain  $\{\mathbf{q}_k\}_k$  as its column. The solution of the problem is given by

$$\mathbf{A}_h = \mathbf{P}_h \mathbf{Q}^+ = \mathbf{P}_h \mathbf{Q}^T (\mathbf{Q} \mathbf{Q}^T)^{-1}. \quad (15.43)$$

We should note that the above approach disregards the fact that the high-resolution patches overlap, and thus a better training procedure can be envisioned for computing  $\mathbf{A}_h$ . Bearing in mind that the eventual high-resolution images (in the test stage) would be constructed by positioning these patches and averaging over their overlaps, we should optimize  $\mathbf{A}_h$  such that these resulting images are as close as possible to their origin. Put formally, and using the similar treatment given in Chapter 14 (Section 14.3), the image  $\hat{\mathbf{y}}_h^j$  should be constructed by the formula

$$\hat{\mathbf{y}}_h^j = \mathbf{y}_l^j + \left[ \sum_{k \in \Omega} \mathbf{R}_k^T \mathbf{R}_k \right]^{-1} \left[ \sum_{k \in \Omega} \mathbf{R}_k^T \mathbf{A}_h \mathbf{q}_k \right]. \quad (15.44)$$

Thus, it is natural to define the best dictionary  $\mathbf{A}_h$  as the solution of the optimization task

$$\begin{aligned} \mathbf{A}_h &= \arg \min_{\mathbf{A}_h} \sum_j \left\| \mathbf{y}_h^j - \mathbf{y}_l^j - \hat{\mathbf{y}}_h^j \right\|_2^2 \\ &= \arg \min_{\mathbf{A}_h} \sum_j \left\| \mathbf{y}_h^j - \mathbf{y}_l^j - \left[ \sum_{k \in \Omega} \mathbf{R}_k^T \mathbf{R}_k \right]^{-1} \left[ \sum_{k \in \Omega} \mathbf{R}_k^T \mathbf{A}_h \mathbf{q}_k \right] \right\|_2^2. \end{aligned} \quad (15.45)$$

The reason  $\mathbf{y}_l^j$  appears in the errors computation is the fact that the patches in  $\mathbf{P}_h$  are constructed from the difference images  $\mathbf{e}_h^j = \mathbf{y}_h^j - \mathbf{y}_l^j$ , and this means that for the image  $\hat{\mathbf{y}}_h^j$  to be constructed, we should return these low-frequencies.

While this minimization task is more demanding, it is expected to lead to better output quality. We shall not explore this here, and in the experiments given below, we adopt the simpler way to derive  $\mathbf{A}_h$ .

This concludes the training phase of the super-resolution algorithm.

#### 15.4.2.2 Testing Phase: Scaling-Up an Image

We are given a test low-resolution image  $\mathbf{z}_l$  to be magnified. This image is assumed to have been generated from a high-resolution image  $\mathbf{y}_h$  by the same blur and scale-down operations as used in the training. The following are the steps performed in order to complete the super-resolution algorithm:

1. Scale this image up by a factor of  $s$  using bicubic interpolation, resulting with  $\mathbf{y}_l$ .
2. Filter the image  $\mathbf{y}_l$  using the same  $K$  high-pass filters that were used for feature extraction in the training, and obtain  $f_k \otimes \mathbf{y}_l$ .

3. Extract patches from these  $K$  images, each of size  $\sqrt{n} \times \sqrt{n}$  from locations  $k \in \Omega$ . Every  $K$  such patches that correspond to the same location are to be concatenated to form a patch vector  $\tilde{\mathbf{p}}'_k$ . This leads to the set  $\{\tilde{\mathbf{p}}'_k\}_k$ .
4. Multiply the found patches  $\{\tilde{\mathbf{p}}'_k\}_k$  by the projection operator  $\mathbf{B}$  to reduce their dimensionality, resulting with the set  $\{\mathbf{p}'_k\}_k$ , each patch of length  $n_l$  ( $\approx 30$ ).
5. Apply the OMP algorithm on  $\{\mathbf{p}'_k\}_k$ , allocating  $L$  atoms to their representation, and find the sparse representation vectors  $\{\mathbf{q}_k\}_k$ .
6. Multiply the representation vectors  $\{\mathbf{q}_k\}_k$  by the high-resolution dictionary  $\mathbf{A}_h$ , and obtain the approximated high-resolution patches,  $\{\mathbf{A}_h \mathbf{q}_k\}_k = \{\hat{\mathbf{p}}^h_k\}_k$ .
7. Construct the final super-resolved image by putting  $\hat{\mathbf{p}}^h_k$  to their proper location, averaging in overlap regions, and adding  $\mathbf{y}_l$  to the final image. Put differently, compute the final image by the formula, which performs exactly the above-described procedure:

$$\hat{\mathbf{y}}_h = \mathbf{y}_l + \left[ \sum_{k \in \Omega} \mathbf{R}_k^T \mathbf{R}_k \right]^{-1} \sum_{k \in \Omega} \mathbf{R}_k^T \hat{\mathbf{p}}^h_k. \quad (15.46)$$

### 15.4.3 Scaling-Up Results

We present two experiments that demonstrate the above-described scaling-up algorithm in action.

The first test is applied on images showing a printed text. The training image (screen-grabbed from a PDF file) is shown in [Figure 15.27](#) – note that only one image is used for training, and adding more is expected to improve the results. The global operator  $\mathbf{L}_{all}$  in this experiment is implemented by first blurring the high-resolution images  $\mathbf{y}^j_h$  with a 1D filter  $[1, 3, 4, 3, 1]/12$  both horizontally and vertically, and then down-sampling it by a factor  $s = 3$ . I.e., the scaled-down image  $\mathbf{z}_l$  is one-ninth of the original image size. The image  $\mathbf{y}_l$  is created by bicubic interpolation of  $\mathbf{z}_l$ , returning to the original size.

Extraction of features from the low-resolution images is done using 4 filters that perform 1-st and 2-nd horizontal and vertical derivatives:  $f_1 = [1, -1] = f_2^T$  and  $f_3 = [1, -2, 1] = f_4^T$ . These filters are applied such that only sampled pixels are used in the filtering computation.<sup>12</sup> The patch size used is  $n = 9$ , and the PCA results with a reduction from  $4 \cdot 81 = 324$  to  $n_l \approx 30$ . The dictionary training procedure applied 40 iterations of the K-SVD algorithm, with  $m = 1,000$  atoms in the dictionary, and allocating  $L = 3$  atoms per patch-representation.

The test image (a different image, grabbed from a different page, but having the same scale) is shown in [Figure 15.28](#). This figure shows the original test image, and the scaled-down version that should be scaled-up. The scaling-up results are shown

<sup>12</sup> This means that we either filter  $\mathbf{z}_l$  and then interpolate, or we filter  $\mathbf{y}_l$  with zero-padded filters of the form  $f_1 = [\mathbf{0}, \mathbf{0}, 1, \mathbf{0}, \mathbf{0}, -1] = f_2^T$  and  $f_3 = [1, \mathbf{0}, \mathbf{0}, -2, \mathbf{0}, \mathbf{0}, 1] = f_4^T$ .

This book is about *convex optimization*, a special class of mathematical optimization problems, which includes least-squares and linear programming problems. It is well known that least-squares and linear programming problems have a fairly complete theory, arise in a variety of applications, and can be solved numerically very efficiently. The basic point of this book is that the same can be said for the larger class of convex optimization problems.

While the mathematics of convex optimization has been studied for about a century, several related recent developments have stimulated new interest in the topic. The first is the recognition that interior-point methods, developed in the 1980s to solve linear programming problems, can be used to solve convex optimization problems as well. These new methods allow us to solve certain new classes of convex optimization problems, such as semidefinite programs and second-order cone programs, almost as easily as linear programs.

The second development is the discovery that convex optimization problems (beyond least-squares and linear programs) are more prevalent in practice than was previously thought. Since 1990 many applications have been discovered in areas such as automatic control systems, estimation and signal processing, communications and networks, electronic circuit design, data analysis and modeling statistics, and finance. Convex optimization has also found wide application in combinatorial optimization and global optimization, where it is used to find bounds on the optimal value, as well as approximate solutions. We believe that many other applications of convex optimization are still waiting to be discovered.

There are great advantages to recognizing or formulating a problem as a convex optimization problem. The most basic advantage is that the problem can then be solved, very reliably and efficiently, using interior-point methods or other special methods for convex optimization. These solution methods are reliable enough to be embedded in a computer-aided design or analysis tool, or even a real-time reactive or automatic control system. There are also theoretical or conceptual advantages of formulating a problem as a convex optimization problem. The associated dual

**Fig. 15.27** First experiment: The training image for the image scaling-up algorithm. This image is of size  $717 \times 717$  pixels, and it provides a set of 54,289 training patch-pairs.

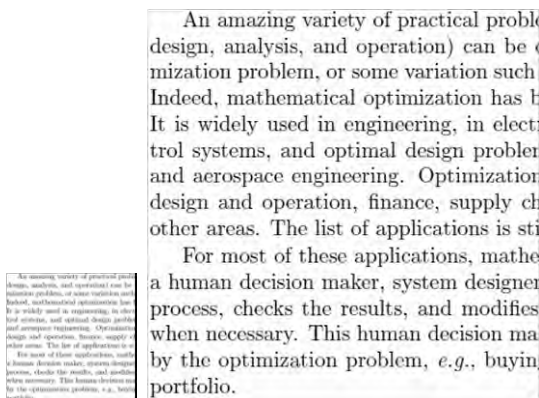
in [Figure 15.29](#), and it is evident that the outcome is far better, compared to the bicubic interpolation, showing an improvement of 2.27dB.

The second experiment is applied on the image **Building**. Starting from the original image  $\mathbf{y}_h$  of size  $800 \times 800$  pixels, we filter this image with the separable filter  $[1, 2, 1]/4$  (horizontally and vertically), and then down-scale it by factor  $s = 2$  to obtain  $\mathbf{z}_l$  of size  $400 \times 400$ .

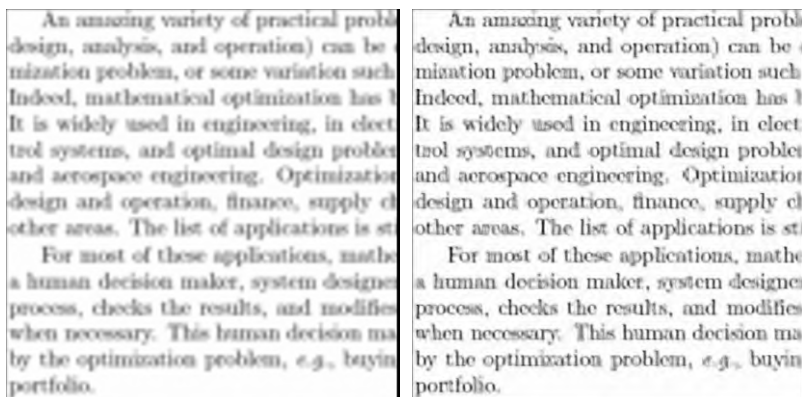
In this experiment we train the dictionaries using the very same image, by further scaling it down by a factor  $s = 2$ , resulting with the image  $\mathbf{z}_{ll}$  of size  $200 \times 200$ . The image pair  $\{\mathbf{z}_l, \mathbf{z}_{ll}\}$  is used for the training, based on the expectation that the relation between these two images reflects also the relation that should be used to go from  $\mathbf{z}_l$  to  $\mathbf{y}_h$ .

Extraction of features from the low-resolution images is done using the same 4 filters, and the dimensionality reduction leads this time to  $n_l \approx 42$ . The training data contains 37,636 pairs of low- and high-resolution patches to be modeled. The parameters of the dictionary training all remain the same (40 iterations of the K-SVD algorithm,  $m = 1000$  atoms in the dictionary, and  $L = 3$  atoms per representation).

[Figure 15.30](#) shows the original image  $\mathbf{y}_h$ , the bicubic scaled-up image  $\mathbf{y}_l$ , and the result of the scaling up algorithm,  $\hat{\mathbf{y}}_h$ . The difference between the two images is 3.32dB, and in order to see where these differences reside, the figure also shows the image  $|\hat{\mathbf{y}}_h - \mathbf{y}_h|$ . [Figure 15.31](#) shows two  $100 \times 100$  portions extracted from  $\mathbf{y}_h$ ,  $\mathbf{y}_l$ , and  $\hat{\mathbf{y}}_h$ , to better show the visual gain achieved by the scaling-up algorithm.



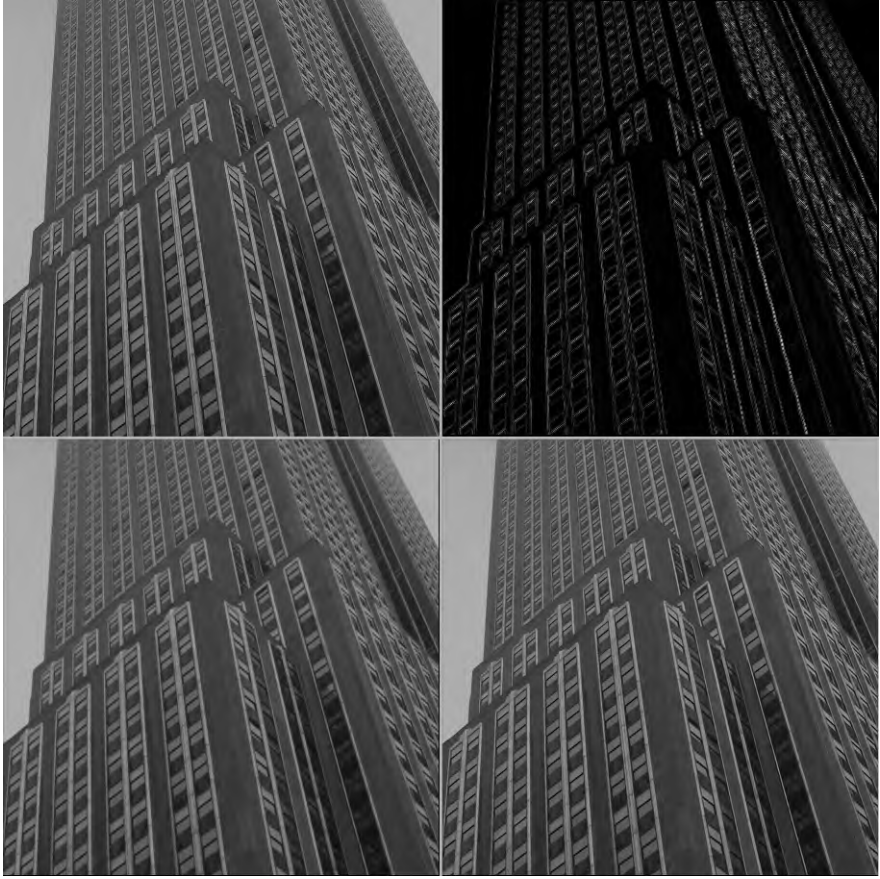
**Fig. 15.28** First experiment: The test image  $\mathbf{z}_l$  to be scaled-up (left), and its original high-resolution version  $\mathbf{y}_h$  (right). The image  $\mathbf{z}_l$  is of size  $120 \times 120$  pixels, and it provides a set of 12,996 patches to operate on.



**Fig. 15.29** First experiment: The scaling up results – the image  $\mathbf{y}_l$  that was scaled-up using a bicubic interpolation (left, RMSE= 47.06), and the algorithm’s result  $\hat{\mathbf{y}}_h$  (right, RMSE= 36.22).

### 15.4.4 Image Scale-Up: Summary

There are various ways to scale-up an image while preserving edges and small details. In this chapter we introduced only one such algorithm that illustrates how sparse representation modeling and dictionary-learning can be used for this goal. The presented algorithm is based on the method proposed by Yang et. al., with few modification. This method is relatively simple, and yet it produces a substantial improvement over bicubic interpolation. The algorithm operates by training a pair of

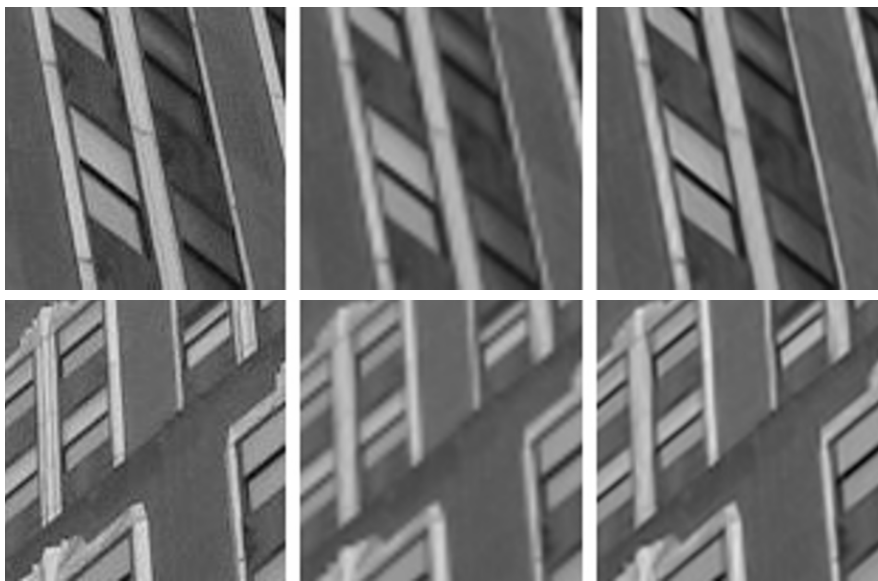


**Fig. 15.30** Second experiment: The original Building image  $\mathbf{y}_h$  (top-left), the bicubic interpolated image  $\mathbf{y}_l$  (bottom-left, RMSE= 12.78), the algorithm's result  $\hat{\mathbf{y}}_h$  (bottom-right, RMSE= 8.72), and the difference image  $|\hat{\mathbf{y}}_h - \mathbf{y}_h|$  magnified by 5 (top-right).

low- and high-resolution dictionaries, using either training images, or exploiting a lower-resolution version of the same image to be handled.

Various further improvements can be considered, and these are likely to enhance the algorithm's output quality. One such option is back-projection, as suggested by Yang et. al.: The image  $\hat{\mathbf{y}}_h$  we construct does not necessarily conforms with the requirement  $\mathbf{L}_{all}\hat{\mathbf{y}}_h\mathbf{a} \approx \mathbf{y}_l$ . Thus, we may project the result  $\hat{\mathbf{y}}_h$  to this constraint. Another option is to force the overlapping patches  $\hat{\mathbf{p}}_k^h$  to better align with each other. This can be done by operating sequentially on the incoming patches  $\mathbf{p}_k^l$ , and when applying the sparse coding stage (to produce  $\mathbf{q}_k$ , add a penalty on the distance between the newly constructed patch,  $\hat{\mathbf{p}}_k^h$ , and the ones already computed.





**Fig. 15.31** Second experiment: Portions from the original Building image  $y_h$  (left), the bicubic interpolated image  $y_l$  (middle), and the algorithm's result  $\hat{y}_h$  (right). Notice that the original portions show some compression artifacts, which do not appear in the scaled-up results.

## 15.5 Summary

In this chapter we have seen how three different applications in image processing are treated using the same tools and methodologies, all based on the *Sparse-Land* model. Each of these problems can be attacked using either global or local sparse representation modeling, and there are close similarities between the various algorithms considered.

Here and in the previous chapter, we have seen a migration from the core *Sparse-Land* model to solutions of practical problems in image processing. This path from theory to practice is not direct nor trivial, as there is a lot of freedom in how to tie the model to the problem. Addressing applications in image processing with this model is therefore a work of art, and in this chapter we have seen several very successful exemplars of this creative process.

We end the technical part of this book here, but we should note that the work on other applications in image processing that use *Sparse-Land* is on-going. Topics such as image and video compression, multi-modal (e.g., video-audio) processing, image segmentation, target detection in images, fusion of images, super-resolution from a set of images, and more, are likely to benefit from the *Sparse-Land* model. Hopefully, many more ideas and contributions will appear in the near-future, along the lines described here.

## Further Reading

1. E. Abreu, M. Lightstone, S.K. Mitra SK, and K. Arakawa, A new efficient approach for the removal of impulse noise from highly corrupted images, *IEEE Trans. on Image Processing*, 5(6):1012–1025, June, 1996.
2. P. Abrial, Y. Moudden, J.-L. Starck, J. Bobin, M.J. Fadili, B. Afeyan and M. Nguyen, Morphological component analysis and inpainting on the sphere: Application in physics and astrophysics, *Journal of Fourier Analysis and Applications (JFAA)*, special issue on “Analysis on the Sphere”, 13(6):729–748, 2007.
3. M. Aharon, M. Elad, and A.M. Bruckstein, The K-SVD: An algorithm for designing of overcomplete dictionaries for sparse representation, *IEEE Trans. on Signal Processing*, 54(11):4311–4322, November 2006.
4. M. Antonini, M. Barlaud, P. Mathieu, and I. Daubechies, Image coding using wavelet transform, *IEEE Trans. on Image Processing*, 1(2):205–220, April 1992.
5. J. Aujol, G. Aubert, L. Blanc-Feraud, and A. Chambolle, Image decomposition: Application to textured images and SAR images, INRIA Project ARIANA, Sophia Antipolis, France, Tech. Rep. ISRN I3S/RR- 2003-01-FR, 2003.
6. J. Aujol and A. Chambolle, Dual norms and image decomposition models, INRIA Project ARIANA, Sophia Antipolis, France, Tech. Rep. ISRN 5130, 2004.
7. J. Aujol and B. Matei, Structure and texture compression, INRIA Project ARIANA, Sophia Antipolis, France, Tech. Rep. ISRN I3S/RR-2004-02-FR, 2004.
8. M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, Image in-painting, in Proc. 27th Annu. Conf. Computer Graphics and Interactive Techniques, pp. 417–424, 2000.
9. M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, Simultaneous structure and texture image inpainting, *IEEE Trans. on Image Processing*, 12(8):882–889, August 2003.
10. A.L. Bertozzi, M. Bertalmio, and G. Sapiro, NavierStokes fluid dynamics and image and video inpainting, *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2001.
11. J. Bobin, Y. Moudden, J.-L. Starck, M.J. Fadili, and N. Aghanim, SZ and CMB reconstruction using GMCA, *Statistical Methodology*, 5(4):307–317, 2008.
12. J. Bobin, Y. Moudden, J.-L. Starck and M. Elad, Morphological diversity and source separation, *IEEE Trans. on Signal Processing*, 13(7):409–412, 2006.
13. J. Bobin, J.-L. Starck, M.J. Fadili, and Y. Moudden, Sparsity, morphological diversity and blind source separation, *IEEE Trans. on Image Processing*, 16(11):2662–2674, 2007.
14. J. Bobin, J.-L. Starck J. Fadili, Y. Moudden and D.L. Donoho, Morphological component analysis: an adaptive thresholding strategy, *IEEE Trans. on Image Processing*, 16(11):2675–2681, 2007.
15. E.J. Candès and F. Guo, New multiscale transforms, minimum total variation synthesis: Applications to edge-preserving image reconstruction, *Signal Processing*, 82(5):1516–1543, 2002.

16. V. Caselles, G. Sapiro, C. Ballester, M. Bertalmio, J. Verdera, Filling-in by joint interpolation of vector fields and grey levels, *IEEE Trans. on Image Processing*, 10:1200–1211, 2001.
17. T. Chan and J. Shen, Local inpainting models and TV inpainting, *SIAM J. Applied Mathematics*, 62:1019–1043, 2001.
18. T. Chen T, K.K. Ma, and L.H. Chen, Tri-state median filter for image denoising, *IEEE Trans.on Image Processing*, 8(12):1834–1838, December, 1999.
19. R. Coifman and F. Majid, Adapted waveform analysis and denoising, in *Progress in Wavelet Analysis and Applications*, Frontiers ed., Y. Meyer and S. Roques, Eds., pp. 6376, 1993.
20. A. Criminisi, P. Perez, and K. Toyama, Object removal by exemplar based inpainting, *IEEE Computer Vision and Pattern Recognition (CVPR)*, Madison, WI, June 2003.
21. J.S. De Bonet, Multiresolution sampling procedure for analysis and synthesis of texture images, *Proceedings of SIGGRAPH*, 1997.
22. A.A. Efros and T.K. Leung, Texture synthesis by non-parametric sampling, *International Conference on Computer Vision*, Corfu, Greece, pp. 1033–1038, September 1999.
23. M. Elad and M. Aharon, Image denoising via learned dictionaries and sparse representation, *IEEE Computer Vision and Pattern Recognition*, New-York, June 17–22, 2006.
24. M. Elad and M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries, *IEEE Trans. on Image Processing* 15(12):3736–3745, December 2006.
25. M. Elad, J-L. Starck, P. Querre, and D.L. Donoho, Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA), *Journal on Applied and Computational Harmonic Analysis*, 19:340–358, November 2005.
26. H.L. Eng and K.K. Ma, Noise adaptive soft-switching median filter, *IEEE Trans.on Image Processing*, 10(2):242–251, February, 2001.
27. M.J. Fadili, J.-L. Starck and F. Murtagh, Inpainting and zooming using sparse representations, *The Computer Journal*, 52(1):64–79, 2009.
28. G. Gilboa, N. Sochen, and Y.Y. Zeevi, Texture preserving variational denoising using an adaptive fidelity term, in *Proc. VLSM*, Nice, France, pp. 137144, 2003.
29. O.G. Guleryuz, Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising - Part I: Theory, *IEEE Trans. on Image Processing*, 15(3):539–554, 2006.
30. O.G. Guleryuz, Nonlinear approximation based image recovery using adaptive sparse reconstructions and iterated denoising - Part II: Adaptive algorithms, *IEEE Trans. on Image Processing*, 15(3):555–571, 2006.
31. J. Mairal, F. Bach, J. Ponce, G. Sapiro and A. Zisserman, Discriminative learned dictionaries for local image analysis, *IEEE Conference on Computer Vision and Pattern Recognition*, Anchorage, Alaska, USA, 2008.
32. J. Mairal, M. Elad, and G. Sapiro, Sparse representation for color image restoration, *IEEE Trans. on Image Processing*, 17(1):53–69, January 2008.



33. J. Mairal, M. Leordeanu, F. Bach, M. Hebert and J. Ponce, Discriminative sparse image models for class-specific edge detection and image interpretation, European Conference on Computer Vision (ECCV) Marseille, France, 2008.
34. J. Mairal, G. Sapiro, and M. Elad, Learning multiscale sparse representations for image and video restoration, *SIAM Multiscale Modeling and Simulation*, 7(1):214–241, April 2008.
35. F. Malgouyres, Minimizing the total variation under a general convex constraint for image restoration, *IEEE Trans. on Image Processing*, 11(12):1450–1456, December 2002.
36. F. Meyer, A. Averbuch, and R. Coifman, Multilayered image representation: Application to image compression, *IEEE Trans. on Image Processing*, 11(9):1072–1080, September 2002.
37. Y. Meyer, Oscillating patterns in image processing and non linear evolution equations, in Univ. Lecture Ser., vol. 22, AMS, 2002.
38. M. Nikolova, A variational approach to remove outliers and impulse noise, *Journal Of Mathematical Imaging And Vision*, 20(1-2):99–120, January 2004.
39. G. Peyré, J. Fadili and J.-L. Starck, Learning the morphological diversity, to appear in *SIAM Journal on Imaging Sciences*.
40. L. Rudin, S. Osher, and E. Fatemi, Nonlinear total variation noise removal algorithm, *Phys. D*, 60:259–268, 1992.
41. A. Said and W. Pearlman, A new, fast, and efficient image codec based on set partitioning in hierarchical trees, *IEEE Trans. on Circuits Systems for Video Technology*, 6(3):243–250, June 1996.
42. J. Shapiro, Embedded image coding using zerotrees of wavelet coefficients, *IEEE Trans. on Signal Processing*, 41(12):3445–3462, December 1993.
43. N. Shoham and M. Elad, Alternating KSVD-denoising for texture separation, The IEEE 25-th Convention of Electrical and Electronics Engineers in Israel, Eilat Israel, December, 2008.
44. J.-L. Starck, E.J. Candès, and D. Donoho, The curvelet transform for image denoising, *IEEE Trans. on Image Processing*, 11(6):131–141, June 2002.
45. J.-L. Starck, D. Donoho, and E.J. Candès, Very high quality image restoration, the 9th SPIE Conf. Signal and Image Processing: Wavelet Applications in Signal and Image Processing, A. Laine, M. Unser, and A. Aldroubi, Eds., San Diego, CA, August 2001.
46. J.L. Starck, M. Elad, and D.L. Donoho, Image decomposition via the combination of sparse representations and a variational approach, *IEEE Trans. on Image Processing*, 14(10):1570–1582, October 2005.
47. J.-L. Starck, M. Elad, and D.L. Donoho, Redundant multiscale transforms and their application for morphological component analysis, *Journal of Advances in Imaging and Electron Physics*, 132:287–348, 2004.
48. J.-L. Starck and F. Murtagh, *Astronomical Image and Data Analysis*, New York: Springer-Verlag, 2002.
49. J.-L. Starck, F. Murtagh, and A. Bijaoui, *Image Processing and data analysis: The multiscale approach*, Cambridge, U.K.: Cambridge Univ. Press, 1998.

50. J.-L. Starck, M. Nguyen, and F. Murtagh, Wavelets and curvelets for image deconvolution: A combined approach, *Signal Processing*, 83(10):2279–2283, 2003.
51. G. Steidl, J. Weickert, T. Brox, P. Mrazek, and M. Welk, On the equivalence of soft wavelet shrinkage, total variation diffusion, total variation regularization, and sides, Dept. Math., Univ. Bremen, Bremen, Germany, Tech. Rep. 26, 2003.
52. L. Vese and S. Osher, Modeling textures with total variation minimization and oscillating patterns in image processing, *Journal of Scientific Computing*, 19:553–577, 2003.
53. M. Vetterli, Wavelets, approximation, and compression, *IEEE Signal Processing Magazine*, 18(5):59–73, September 2001.
54. J. Yang, J. Wright, T. Huang, and Y. Ma, Image super-resolution as sparse representation of raw image patches, IEEE Computer Vision and Pattern Recognition (CVPR), 2008.
55. J. Yang, J. Wright, T. Huang, and Y. Ma, Image super-resolution via sparse representation, submitted to *IEEE Trans. on Image Processing*, September 2009.
56. M. Zibulevsky and B. Pearlmutter, Blind source separation by sparse decomposition in a signal dictionary, *Neur. Comput.*, 13:863–882, 2001.



# Chapter 16

## Epilogue

### 16.1 What is it All About?

The field of image processing offers an unusually fertile playground for applied mathematicians, where the distance between an abstract mathematical idea and an application or even a product may be small. This explains both the presence of so many mathematicians in this field, and the fact that research in it became so mathematically oriented in the past two decades. The activity on sparse and redundant representation modeling is just one manifestation of both these trends.

In this book we have explored the concept of sparse representations for signals and images. Although sparse representation is a poorly defined problem and a computationally impractical goal in general, we bring in this book a long series of recent mathematical results, showing that under certain conditions one can obtain results of a positive nature, guaranteeing uniqueness, stability, and computational practicality.

Inspired by such positive results, we discussed in this book how these can be used to form an appealing and strong source model for signals and images. We have shown several potential applications that use this model of sparse representation in real image processing settings, and showed cases where this leads to state-of-the-art results. More specifically, problems such as image denoising, image deblurring, facial image compression, image inpainting, and image scale-up, all benefit from this model.

### 16.2 What is Still Missing?

This book is far from being complete. There are many theoretical, numerical, and applicative topics related to sparse representations, with an already existing and solid knowledge, which have been deliberately omitted from this book. These include topics such as joint-sparsity, compressed-sensing, low-rank matrix comple-

tion, non-negative matrix factorization and non-negative sparse coding, blind source separation, multi-modal sparse representation, relation to wavelet theory, and more.

Furthermore, in recent years we have seen more and more application branches that successfully lean on this model, going beyond the image processing activity we have explored in this book. These include tasks in machine learning, computer-vision, pattern-recognition, wireless communication, audio processing, brain-research (mostly studying the human visual system), inverse problems in geo-sciences, MRI imaging, error-correcting-codes, and more.

Beyond all these missing parts that correspond to an existing knowledge, this book also lacks in the fact that there are some holes in its presentation, which correspond to yet unstudied pieces. While this book aims to give a fluent and compelling view of the understanding and knowledge that have grown around *Sparse-Land*, there are key questions that remain open at this stage. For example:

- A fundamental issue in using the *Sparse-Land* model in signal/image processing is the question of the suitability of this model for such signals in the first place. The approach taken in this book (and in the research it documents) is an empirical “try and see” kind of answer. More work is required to carefully map the connections between *Sparse-Land* and other models, such as Markov-random-field (MRF), PCA and its generalizations, and more. Perhaps a more ambitious goal would be to characterize the kind of signal-sources that find *Sparse-Land* as a suitable model.
- Another fundamental question refers to flaws that exist in the *Sparse-Land* model. For example, this model ignores the existing dependencies between atoms in the sparse representation. Another manifestation of these flaws is exhibited by the fact that one cannot synthesize reasonable signals from this model – for example, a direct approach of randomly generating a sparse vector  $\mathbf{x}$  with iid entries and multiplying it by the dictionary does not lead to a natural image, even if the dictionary is of good quality for images. Model extensions to better match this model to true data are desired, and these are likely to take applications to the next level of performance.

Other fundamental issues to be studied and solved include topics such as analysis of pursuit techniques with non-worst-case measures (as opposed to the mutual-coherence and the RIP), development of a multi-scale dictionary-learning paradigm, getting performance guarantees for dictionary learning algorithms, setting the proper redundancy for a dictionary, and more. Hopefully, the research to be done in the coming years will address some of these issues.

## 16.3 Bottom Line

Despite these holes in the presentation and the missing pieces mentioned above, this book offers an important and organized view of this field, setting the foundations on which all the above missing parts are (or could be) built. As the activity in the field of

sparse and redundant representations is so vivid and expanding, we feel it is time to look back at the achievements made in the past two decades or so, and summarize them clearly, as a way to understand where we are standing today, and this way, better plan for the future research in this area. This is the approach taken in this book.

It is very probable that other books will follow in the near future, in an attempt to cover better these missing parts and the novel research projects that will emerge. Indeed, perhaps a second edition of this very book would make such an attempt. Time will tell.



# Notation

(By order of their first appearance in the book)

## Chapter 1:

$n$	Signal's (vector's) dimension
$m$	Number of atoms (columns) in the dictionary (matrix)
$\mathbb{R}$	Real numbers
$\mathbb{R}^n$	$n$ -dimensional Euclidean real vector space
$\mathbb{R}^{n \times m}$	Euclidean space for real matrices of size $n \times m$
$\mathbf{A}, \mathbf{b}$	The elements of the linear system $\mathbf{Ax} = \mathbf{b}$
$\mathbf{A}^T$	The transpose of the matrix $\mathbf{A}$
$\mathbf{A}^{-1}$	The inverse of the matrix $\mathbf{A}$
$\mathbf{A}^+$	The Moore-Penrose pseudo-inverse of the matrix $\mathbf{A}$
$J(\mathbf{x})$	Penalty function
$\mathcal{L}$	Lagrangian functional
$\Omega$ or $\mathcal{C}$	Sets
$\ \mathbf{x}\ _p^p$	The $p$ -th power of the $\ell_p$ -norm of the vector $\mathbf{x}$
$\ \mathbf{x}\ _2^2$	The squared- $\ell_2$ -norm – a sum of the squares of the entries in $\mathbf{x}$
$\ \mathbf{x}\ _1$	The $\ell_1$ -norm – a sum of absolute values of the entries in $\mathbf{x}$
$\ \mathbf{x}\ _\infty$	The $\ell_\infty$ -norm – a maximal absolute value among the entries in $\mathbf{x}$
$\ \mathbf{x}\ _{w\ell_p}$	The weak- $\ell_p$ -norm
$\ \mathbf{x}\ _0$	The $\ell_0$ -norm – a count of the non-zeros in the vector $\mathbf{x}$
$(P_1)$	Seeking the $\ell_1$ -shortest solution of a linear system
$(P_p)$	Seeking the $\ell_p$ -shortest solution of a linear system
$(P_0)$	Seeking the sparsest solution of a linear system

## Chapter 2:

$\Psi, \Phi$	Unitary matrices
$\mathbf{x}_\Psi, \mathbf{x}_\Phi$	The two portions of the sparse representation in the two-ortho case
$\mathbf{I}$	The identity matrix
$\mathbf{F}$	The Fourier matrix



$\mu(\mathbf{A})$	Mutual Coherence of the matrix $\mathbf{A}$
$\text{rank}(\mathbf{A})$	The rank of the matrix $\mathbf{A}$
$\text{spark}(\mathbf{A})$	The smallest number of columns in $\mathbf{A}$ that are linearly dependent
$\mathbf{G}$	The Gram matrix
$\mu_1(\mathbf{A})$	Tropp's babel function for the matrix $\mathbf{A}$
$\mathbf{U}\Sigma\mathbf{V}^T$	The Singular Value Decomposition of a matrix

**Chapter 3:**

$\mathbf{A}_s$	A portion of the matrix $\mathbf{A}$ containing the columns in the set $s$
$\mathbf{a}_k$	The $k$ -th column in $\mathbf{A}$
$\mathbf{r}^k$	Residual vector at the $k$ -th step
$\mathcal{S}^k$	The estimated support at the $k$ -th step
$\delta(\mathbf{A})$	The universal decay-factor for OMP
$\mathcal{S}$	The support (active set) of $\mathbf{x}$
$\mathbf{W}$	A diagonal weight matrix
$\mathbf{X}$	A diagonal matrix containing $ \mathbf{x} ^q$ on its main diagonal

**Chapter 4:**

$k_0$	Cardinality – the number of non-zeros in the representation
$\mathbf{x}_\Psi, \mathbf{x}_\Phi$	The two parts of the representation in the two-ortho case
$k_p, k_q$	Number of non-zeros in the vectors $\mathbf{x}_\Psi, \mathbf{x}_\Phi$ in the two-ortho case
$\mathcal{C}$	A set of vectors
$x_{\min}$	The minimal absolute entry in $\mathbf{x}$
$x_{\max}$	The maximal absolute entry in $\mathbf{x}$

**Chapter 5:**

$(P_0^\epsilon)$	Seeking the sparsest $\epsilon$ -approximate solution of a linear system
$\epsilon, \delta$	Allowed error
$\mathbf{v}, \mathbf{e}$	Noise or perturbation vectors
$\sigma_s$	The $s$ singular value of a matrix
$\text{Spark}_\eta(\mathbf{A})$	The smallest number of columns in $\mathbf{A}$ that are at most $\eta$ away (in terms of singular values) from linear dependence
$\delta_s$	The RIP constant
$\lambda_{\min}(\mathbf{A})$	Smallest singular value of $\mathbf{A}$
$\lambda_{\max}(\mathbf{A})$	Largest singular value of $\mathbf{A}$
$(P_1^\epsilon)$	Seeking the $\ell_1$ -shortest $\epsilon$ -approximate solution of a linear system
$(Q_1^\lambda)$	A variant of $(P_1^\epsilon)$ with penalty replacing the constraint
$\lambda$	A coefficient weighting the sparsity penalty
$\partial f$	A sub-gradient of $f$
$\sigma$	Random noise standard deviation

**Chapter 6:**

$\rho(\mathbf{x})$	A separable sparsity penalty term
$\mathcal{S}_{\rho, \lambda}(\mathbf{x})$	An element-wise shrinkage on $\mathbf{x}$ , whose shape is dictated by $\rho$ , and thresholded by $\lambda$

- $d(\mathbf{x}_1, \mathbf{x}_2)$  Distance between the vectors  $\mathbf{x}_1$  and  $\mathbf{x}_2$   
 $\text{diag}(\mathbf{A})$  The diagonal of the square matrix  $\mathbf{A}$   
 $\mathbf{P}$  Projection operator

**Chapter 7:**

- $N_\psi, N_\phi$  Same as  $k_p, k_q$ , used in Candes and Romberg's results  
 $P(\text{event})$  Probability of *event*

**Chapter 8:**

- $(P_{DS}^l)$  The Danzig-Selector optimization problem  
 $\theta_{s_1, s_2}$  The restricted orthogonality property parameter

**Chapter 9:**

- $\mathcal{Y}$  A set of example signals  
 $P(\mathbf{y})$  A prior probability on the signal  $\mathbf{y}$   
 $\hat{P}(\mathbf{y})$  An estimated prior probability on the signal  $\mathbf{y}$   
 $\mathcal{M}_{\Lambda, k_0, \alpha, \epsilon}$  The *Sparse-Land* model  
 $\Omega_{\mathbf{y}_0}^\delta$  A set containing the  $\delta$  neighborhood of  $\mathbf{y}_0$   
 $\mathbf{c}$  A center vector of a neighborhood, computed as the mean  
 $\mathbf{Q}_{\mathbf{y}_0}$  A projection matrix that considers the permissible subspace around  $\mathbf{y}_0$   
 $\mathbf{T}$  An analysis operator (transform)  
 $\mu_{\text{true}}$  The relative volume in  $\mathbb{R}^n$  with high-probability signals  
 $\mu_{\text{model}}$  The relative volume in  $\mathbb{R}^n$  with high-probability signals, based on the model

**Chapter 10:**

- $\mathbf{H}$  A linear degradation operator (e.g., blur)  
 $\hat{\mathbf{y}}$  Measured signal  
 $\hat{\mathbf{x}}$  Estimated representation

**Chapter 11:**

- $P_s(|s|)$  The probability density function of the cardinality  $|s|$   
 $P_X(\mathbf{x})$  The probability density function of the representation vector  $\mathbf{x}$   
 $\mathcal{R}_1, \mathcal{R}_2$  Random generators of sparse representation vectors  
 $\sigma_x$  The variance of the non-zero entries in the representation vector  $\mathbf{x}$   
 $\sigma_e$  The variance of the noise entries in the vector  $\mathbf{e}$   
 $\hat{\mathbf{x}}^{\text{MAP}}$  The maximum-a-posteriori probability estimation  
 $\hat{\mathbf{x}}^{\text{MMSE}}$  The oracle estimation (that knows the correct support)  
 $\hat{\mathbf{x}}^{\text{oracle}}$  The minimum mean-squared-error estimation  
 $P(\mathbf{x}|\mathbf{y})$  Conditional probability of  $\mathbf{x}$  given  $\mathbf{y}$   
 $E(\text{expression})$  The expected value of the random *expression*  
 $\mathbf{x}_s$  The non-zero part in the vector  $\mathbf{x}$   
 $q_s$  The relative weights of the various solutions that are merged in the MMSE estimate  
 $\det(\mathbf{A})$  The determinant of the matrix  $\mathbf{A}$

<b>Q</b>	An interpolation operator
$\mathbf{y}_h$	An ideal high-resolution image
$\mathbf{z}_l$	A blurred-decimated-noisy version of $\mathbf{y}_h$
$\mathbf{y}_l$	An interpolated version of $\mathbf{z}_l$
$\mathbf{e}_h$	The difference image between $\mathbf{y}_l$ and $\mathbf{y}_h$
$\mathbf{L}_{all}$	The overall operator that relates $\mathbf{y}_l$ to $\mathbf{y}_h$
$\mathbf{A}_l$	A dictionary for the low-resolution features
$\mathbf{A}_h$	A dictionary corresponding to $\mathbf{A}_h$ , for the high-resolution content
$f_k$	Derivative filters that are used to extract features
$\mathcal{P} = \{\mathbf{p}_k^h, \mathbf{p}_k^l\}$	Training set of high- and low-resolution patches
<b>B</b>	A dimensionality reduction projection operator for the low-resolution patch features



# Acronyms

(Alphabetical ordered)

2D	Two Dimensional
BCR	Block Coordinate Relaxation
BM3D	Block-Matching and 3D filtering
BP	Basis Pursuit
BPDN	Basis Pursuit Denoising
B/W	Black and White
CD	Coordinate Descent
CG	Conjugate Gradient
CMB	Cosmic Microwave Background
CoSaMP	Compressive Sampling Matching Pursuit
CS	Compressed Sensing
DCT	Discrete Cosine Transform
DS	Danzig Selector
ERC	Exact Recovery Condition
FFT	Fast Fourier Transfrom
FOCUSS	FOcal Under-determined System Solver
GSM	Gaussian Scale Mixture
ICA	Independent Component Analysis
iid	Independent and Identically Distributed
IRLS	Iterated Reweighed Least Squares
ISNR	Improvement Signal to Noise Ratio
JPEG	Joint Photographic Experts Group
KLT	Karhunen Loeve Transform
K-SVD	K times Singular Value Decomposition
K-Means	K times Mean computation
LARS	Least Angle Regression
LASSO	Least Absolute Shrinkage and Selection Operator
LP	Linear Programming
LS	Least Squares

MAP	Maximum A’posteriori Probability
MCA	Morphological Component Analysis
MMSE	Minimum Mean-Squared-Error
MOD	Method of Directions
MP	Matching Pursuit
MRF	Markov Random Field
MRI	Magnetic Resonance Imaging
MSE	Mean Squared Error
NLM	Non Local Means
NP	Non Polynomial
OMP	Orthogonal Matching Pursuit
PCA	Principal Component Analysis
PCD	Parallel Coordinate Relaxation
PDF	Probability Density Function
PSNR	Peak Signal to Noise Ratio
QP	Quadratic Programming
RIP	(or RIC) Restricted Isometry Property (or Condition)
RMSE	Root Mean Squared Error
ROP	Restricted Orthogonality Property
SESOP	Sequential Subspace Optimization
SNR	Signal to Noise Ratio
SSF	Separable Surrogate Functionals
SSIM	Structural Similarity
StOMP	Stage-wise Orthogonal Matching Pursuit
SURE	Stein Unbiased Risk Estimator
TV	Total Variation
SVD	Singular Value Decomposition
VQ	Vector Quantization

# Index

- accelerate/acceleration, 127, 128, 132, 191, 199, 233
- acoustic, 14, 177, 178
- adversary, 104, 107, 153, 157
- amalgam, 18, 26, 149
- analysis, 178, 179, 186, 312–314, 335
- anomaly, 3, 169
- anti-symmetric, 281
- artifact, 256, 257, 263, 265, 266, 268, 278, 292, 303, 315, 337
- asymptotic, 140, 144
- atom, 66, 74, 95, 98, 123, 124, 128, 132, 154, 172, 173, 176, 178, 180, 188, 202, 216, 217, 227–235, 239, 240, 242–244, 251–253, 255, 259, 262, 264, 276, 288, 290, 292, 294, 295, 297, 301, 302, 311, 312, 320, 321, 325, 326, 331, 333, 335, 347, 349, 350, 360
- atomic-decomposition, 172, 176
- audio, 174, 353, 360
- auto-correlation, 172, 261
- average performance, 137, 143, 144, 157
- B/W, 303
- Babel function, 27, 28, 71, 75
- back-project, 117, 121–124, 352
- bandelet, 227, 275
- Basis-Pursuit, 50–52, 55, 58, 59, 62, 64, 68, 69, 71–75, 89, 95, 137, 138, 141–143, 153, 155, 163–165, 276, 286
- Basis-Pursuit-Denoising, 95, 103, 105, 153–155, 157
- Bayes, 203, 204, 208, 211, 213
- Bayesian, 169, 202, 278
- Beltrami-flow, 295
- bicubic, 342, 348–351
- bilateral-filter, 294, 295
- bilinear, 342
- binary, 233
- bit-allocation, 249, 252
- bit-rate, 255, 257, 261, 264, 265, 267, 268
- Block-Coordinate
  - descent, 242
  - minimization, 287, 329
  - rationale, 233
  - Relaxation, 113, 115, 117, 120, 230, 241, 289
- block-matching, 296, 297
- blockiness, 263–266, 268, 278, 279
- BM3D, 199, 296, 297, 303
- Bound-Optimization, 115, 117, 118
- BP, *see* Basis-Pursuit
- BPDN, *see* Basis-Pursuit-Denoising
- Bregman, 134
- Canny edge detector, 315
- cartoon, 178, 227, 310–312, 314–316, 320–322, 335, 337
- Cauchy-Schwartz inequality, 21, 36, 39
- CD, *see* Coordinate-Descent
- CG, *see* Conjugate-Gradient
- chemistry, 172
- circulant, 141, 242
- color, 248, 291, 303, 332, 333
- combinatorial, 13, 23, 24, 28, 101, 215
- compressed-sensing, 124, 144, 156, 177, 359
- compression, 169, 171, 177, 221, 247–249, 251–257, 260–262, 264, 265, 267, 269, 309, 311, 353, 359
- computer-vision, 315, 360
- Conjugate-Gradient, 50, 91, 111, 119, 124, 127
- contourlet, 227, 275, 310, 312
- converge, 6, 49, 51, 114, 116, 118, 125, 132, 191, 290

convergence, 29, 44, 93, 97, 111, 112, 117, 119, 120, 125, 132, 134, 230, 233, 234, 240, 243, 244  
 convergent, 120  
 convex, 5–7, 9, 13, 26, 28, 48, 50, 51, 90, 95, 111, 114, 115, 117, 125, 154, 189, 191  
 convexifying, *see* convex  
 Convexity, *see* convex  
 convolution, 242, 347  
 Coordinate-Descent, 112, 120  
 cortex, *see* visual cortex  
 CoSaMP, 125  
 cosine, 171  
 covariance, 122, 208, 213  
 cross-correlation, 243  
 curvelet, 227, 275, 310, 312, 314–316, 337  
 CVX, 51, 90  
 cyclic, 170, 242, 285  
  
 Dantzig-Selector, 107, 153–159, 163–165, 204  
 DCT, *see* Discrete-Cosine-Transform  
 deblock, 263, 265–269, 283  
 deblurring, 115, 132, 185, 186, 188, 191, 198, 199, 247, 283, 309, 359  
 decay-factor, 43, 44, 207  
 decimation, 342, 343  
 decoder, 249, 251–253, 267  
 decompression, 251, 260, 267  
 denoising, 104, 111, 125, 170, 174, 175, 177–180, 273–276, 278, 279, 282, 285, 288, 290–292, 294–298, 300–303, 309–311, 316, 317, 320, 323, 326–328, 335, 338–340, 359  
 detection, 92, 100, 169, 178, 249, 252, 253, 311, 315, 353  
 DFT, *see* Discrete-Fourier-Transform  
 diagonal  
   -block, 38, 281  
   -main, 41, 49, 87, 103, 119, 128, 142, 178, 242  
   -off, 25, 27–30, 44, 84, 87, 103, 142  
   main, 27, 70  
   matrix, 41, 49, 91, 119, 127, 242, 275, 276, 300, 301  
 diagonally-dominant, 26, 75, 84  
 dictionary-learning, 228, 229, 231, 233, 234, 238, 239, 241, 242, 244, 247, 289, 297, 320, 330, 347, 351, 360  
 dictionary-update, 231, 233, 234, 241, 243, 330, 331  
 differentiable, 5, 7, 95, 102, 112, 299, 300  
 diffusion, 295  
 discrete, 13, 21, 35, 186, 227

Discrete-Cosine-Transform, 142, 171, 235, 237  
   1D-, 235  
   2D-, 171, 172, 239, 279, 282, 290, 291  
   3D-, 296  
   global, 312, 314, 316, 337  
   local, 312, 315  
   over-complete, 223  
   redundant, 239, 289, 290, 320, 325, 329, 331, 340  
   unitary, 163, 223, 278, 312  
 Discrete-Fourier-Transform, 171  
 disk, 26, 84, 87, 107  
 double-sparsity, 239  
 down-sampling, 349  
 down-scale, 346, 350  
 DS, *see* Dantzig-Selector  
 dual, 62, 63, 311  
  
 edge, 264, 268, 274, 311, 315, 339, 347, 351  
 eigenvalue, 26, 45, 75, 84, 87, 107, 120, 158, 261, 314  
 eigenvector, 261  
 ellipsoid, 117, 172  
 EM, *see* Expectation-Maximization  
 empirical, 13, 68, 71, 76, 98, 103, 128, 137, 140, 150, 207, 212, 220, 228, 233, 339, 360  
 encoder, 251, 252  
 encryption, 178  
 entropy-coding, 247–249, 252  
 equiangular, 25  
 equivalence, 39, 57, 58, 65, 68, 76, 80, 141, 144, 149, 155, 179  
 equivalent, 8, 10, 36, 37, 59, 68, 101–103, 127, 155, 163, 164, 170, 173, 174, 178, 179, 209, 211, 240, 241, 251, 313  
 ERC, *see* Exact Recovery Condition  
 error-correcting-code, 26, 360  
 Exact Recovery Condition, 73  
 Exact-recovery-Condition, 75  
 Expectation-Maximization, 112, 117, 118  
  
 facial, 247–249, 251, 253–257, 267, 269, 359  
 factorization, 229, 360  
 FFT, *see* Fourier  
 FHT, *see* Hadamard  
 filling-in, 309, 324, 337  
 first-derivative, 171, 186  
 fixed-point, 49, 112, 115, 119, 120  
 FOCUSS, 48, 50, 51  
 Fourier, 19, 278  
   Discrete-Fourier-Transform, 22, 171  
   Fast-Fourier-Transform (FFT), 242



**Chapter 12:**

$\mathbf{Y}$	A matrix containing signal examples as its columns
$\mathbf{X}$	A matrix containing representation vectors as its columns
$\mathbf{E}_j$	A matrix containing signal representation errors as its columns, excluding the $j$ atom
$\otimes$	Kronecker product
$\mathbf{A}_0$	A fixed dictionary
$\mathbf{Z}$	A matrix containing sparse representation of the dictionary atoms
$\text{tr}(\mathbf{A})$	The trace of the matrix $\mathbf{A}$
$\mathbf{R}_k$	Patch extraction operator from location $k$ in an image

**Chapter 13:**

$B$	Budget of bits
$P$	Number of pixels in the image
$n$	Block-size
$m$	Code-book (or dictionary) size
$\mathbf{h}_k$	A deblocking filter for the $k$ -th location
$\mathbf{e}_k$	The $k$ -th element in the trivial basis

**Chapter 14:**

$\mathbf{y}_0$	An ideal image of size $N$
$\hat{\mathbf{y}}$	The estimated image (denoised)
$\hat{\mathbf{x}}$	The estimated image's representation
$\mathbf{p}_{ij}$	A small patch extracted from an image at location $[i, j]$
$\mathbf{q}_{ij}$	A patch's $\mathbf{p}_{ij}$ sparse representation
$\hat{\mathbf{q}}_{ij}, \hat{\mathbf{q}}_{ij}$	Estimated sparse representations of $\mathbf{p}_{ij}$
$\mathbf{c}$	Coefficients shaping the shrinkage functions
$\theta$	Set of parameters governing the denoising algorithm
$h(\mathbf{y}, \theta)$	The denoising formula/algorithm
$\eta(\hat{\mathbf{y}}, \mathbf{y})$	An unbiased estimator of the MSE

**Chapter 15:**

$\mathbf{y}_c$	The cartoon-part in an image
$\mathbf{y}_t$	The texture-part in an image
$\hat{\mathbf{y}}_c$	The estimated cartoon-part in an image
$\hat{\mathbf{y}}_t$	The estimated texture-part in an image
$M_c, M_t$	Number of atoms in the cartoon and texture dictionaries, respectively
$\mathbf{T}_c, \mathbf{A}_c$	Analysis/synthesis operators for the cartoon part
$\mathbf{T}_t, \mathbf{A}_t$	Analysis/synthesis operators for the texture part
$\mathbf{A}_a$	The total dictionary, containing both $\mathbf{A}_c$ and $\mathbf{A}_t$
$\mathbf{q}_c, \mathbf{q}_t$	The cartoon and texture sparse-representations vector for an image patch
$\mathbf{M}$	A mask matrix, describing the remaining samples in the signal/image
$\mathbf{S}$	A decimation operator

- identity-Fourier, 24
- matrix, 18, 21
- transform, 18, 21
- Frame, 14, 25, 149, 187, 227, 241, 275, 310, 313
- frequency, 18, 21, 170, 227, 257, 314, 347
- full-overlap, 285, 312, 330
- full-rank, 3, 4, 17, 25, 29, 44, 65, 68, 149, 179, 313
- galaxy, 315, 316
- Gaussian, 24, 92, 98, 122, 124, 125, 128, 137, 153, 154, 157, 158, 163, 172, 173, 177, 185, 202, 205, 207–209, 211, 213, 214, 230, 274, 275, 278, 279, 284, 288, 298, 300, 303, 312, 314, 325, 328, 337, 338, 343
- Gaussian-Scale-Mixture, 278
- generalized, 84, 107, 111, 175, 237
- geo-science, 360
- Gershgorin, 26, 27, 84, 87, 107
- global, 114, 278, 287, 311, 314, 319, 323, 335, 342, 349, 353
  - DCT, 312, 314, 316, 337
  - inpainting, 335, 337
  - inpainting, 327
  - MAP, 319
  - MCA, 311, 312
  - minimum/er, 6, 13, 50, 51, 71, 80, 113, 116, 125, 154, 233
  - model/ing, 274, 278
  - optimality/er, 17, 23, 24, 30, 95
  - solution, 51
  - thresholding, 282, 300
  - treatment, 273
- globally optimal, 101, 102, 113
- globally trained, 290
- GPSR, 90
- gradient, 112, 127, 171, 331
  - 2D-, 186
  - Conjugate-, 50, 91, 111, 119, 124, 125, 127
  - projection, 91
  - sub-, 95, 96, 102, 112, 113
- Grassmannian, 25, 28–31, 148
- gray-value, 248
- grayscale, 248, 261
- greedy, 35, 36, 39, 41, 43, 45, 46, 51, 52, 65, 67, 90, 98, 112, 123, 138, 144, 211, 216, 217
- GSM, *see* Gaussian-Scale-Mixture
- Haar, 186, 311
  - dictionary, 276, 277, 311
  - transform, 186, 187, 190, 275
  - wavelet, 187, 189, 337
- Hadamard, 19, 127, 171
  - Fast-Hadamard-Transform, 128
- Heaviside, 178, 179, 186
- Heisenberg's uncertainty principle, 21
- Hessian, 6, 115, 118, 127
- histogram, 332
- holes, 309, 327, 333, 335, 337, 360
- homogeneity, 9, 12
- Homotopy, 51, 91
- hyper-cube, 169
- hyperplane, 9, 10
- iid, *see* independent and identically-distributed
- ill-posed, 309
- Image-Signature-Dictionary, 242, 243
- Improvement-Signal-to-Noise Ratio, 194, 198
- incoherence, 71
- incoherent, 25, 71
- independent and identically-distributed, 46, 92, 98, 124, 128, 141, 143, 145, 153, 157, 163, 177, 201, 202, 207, 230, 298, 300, 360
- inpainting, 178, 309, 311, 324–328, 331, 333, 335, 337–340, 359, 373
- invariant
  - rotation, 186
  - space, 170, 185, 342
  - translation, 186
  - unitary, 101, 112
- invertible, 4, 48, 49, 173, 178, 204, 205, 312, 313, 323
- IRLS, *see* Iterative-Reweighed-Least-Squares
- ISD, *see* Image-Signature-Dictionary
- ISI, 303
- ISNR, *see* Improvement-Signal-to-Noise Ratio
- isometry, 86, 156
- isotropic, 315
- Iterative-Reweighed-Least-Squares, 48, 51, 52, 90–93, 97, 98, 108, 111, 115, 119, 125, 127, 128, 132
- iterative-shrinkage, 91, 111, 112, 115, 119, 120, 123–125, 127, 128, 132, 134, 154, 155, 194, 199, 275, 313
- Jeffrey's uninformative prior, 191
- joint-sparsity, 297, 359
- JPEG, 14, 171, 172, 249, 256, 261, 283
- JPEG2000, 14, 247–250, 253, 256, 261, 269
- K-Means, 233
- K-SVD, 228, 233–236, 238–240, 261, 288, 289, 330–332

algorithm, 231, 233, 240, 248, 251, 297, 331, 350  
 denoising, 291, 292, 297, 303, 317, 330  
 dictionary, 235, 255, 261, 290  
 inpainting, 327, 331, 333  
 learning, 268, 320  
 training, 251, 347  
 Karhunen-Loeve-Transform, 171  
 kernel, 186, 191, 194, 295  
 KLT, *see* Karhunen-Loeve-Transform  
 Kronecker-product, 235, 330  
 Kruskal-rank, 23  
  
 L1-LS, 51, 90  
 L1-Magic, 90  
 Lagrange multiplier, 4, 20, 49, 90, 314  
 Lagrangian, 4, 9, 20  
 Lanczos, 342  
 Laplacian, 170, 171, 250  
 large-deviation, 145, 147  
 LARS, *see* Least-Angle-Regression  
 LASSO, *see* Least-Absolute-Shrinkage-and-Selection-Operator  
 Least-Absolute-Shrinkage-and-Selection-Operator, 93, 95  
 Least-Angle-Regression, 90, 93, 95–98, 100, 108, 111, 124, 138  
 Least-Squares, 35, 38, 39, 41, 100, 106, 108, 111, 115, 124, 132, 157, 163, 230, 233, 240, 241, 267, 278, 281  
 lexicographic, 267, 278, 330  
 likelihood, 171, 203, 213  
 line-search, 116, 123, 127, 128, 132, 191  
 Linear-Programming, 8, 51, 62, 153–155  
 linearly-dependent, 7, 21, 23, 24, 28, 143, 324  
 linearly-independent, 26–28, 143, 324  
 local  
   -PCA, 175, 176, 180  
   DCT, 312, 315  
   K-SVD, 327  
   MCA, 311, 316, 320  
   minimum, 50, 51, 114, 116, 118, 120, 125, 203, 233, 243  
 log-likelihood, 154, 287, 319  
 low-dimension, 97, 119, 122, 127, 174, 221, 228, 238  
 LP, *see* Linear-Programming  
 LS, *see* Least-Squares  
  
 Majorization-Maximization, 118  
 MAP, *see* Maximum-A'posteriori-Probability  
 MAP-oracle, 205  
 marginal, 208, 211, 213, 218  
 Markov-Random-Field, 179, 180, 360

Matching-Pursuit, 39, 211  
 Least-Squares-Orthogonal- (LS-OMP), 38, 47  
 Orthogonal- (OMP), 36–39, 41, 42, 44, 45, 47, 51, 52, 55, 57, 58, 60, 65, 68, 71, 73, 74, 89, 90, 95, 98, 100, 102–104, 108, 111, 124, 138, 153, 165, 211, 212, 216, 221, 236, 239, 240, 252, 286, 288, 289, 291, 292, 326, 340, 347, 349  
 Random-Orthogonal- (Random-OMP), 216, 217, 221–223  
 Stagewise-Orthogonal- (StOMP), 123–125, 127  
 Weak- (W-MP), 39, 44, 47  
 Matlab, 29, 51, 62, 97, 137, 163, 260  
 matroid, 23  
 Maximum-A'posteriori-Probability, 154, 170, 201–203, 205, 206, 208–212, 215, 218, 220, 221, 224, 286, 289, 319, 328, 330  
 Maximum-Likelihood, 203  
 MCA, *see* Morphological Component Analysis  
 MCAlab, 90  
 Mean-Squared-Error, 185, 198, 204–206, 218, 219, 223, 250, 298, 299, 301, 302, 311  
 median, 339, 340  
 Method of Optimal Directions, 228, 230, 231, 233, 235, 236, 238, 239, 288, 289, 330, 347  
 Minimum-Mean-Squared-Error, 201–203, 205, 206, 212–221, 223, 224, 274, 279, 294, 295  
 minor, 26, 27, 84, 87, 90, 142  
 mixture, 173, 310, 317, 337  
 MMSE, *see* Minimum-Mean-Squared-Error  
 MMSE-oracle, 205  
 MOD, *see* Method of Optimal Directions  
 molecule, 172  
 monotone, 11, 27, 51, 83, 85, 114, 233  
 Morphological Component Analysis, 177, 178, 309–312, 316, 317, 320, 335  
 MP, *see* Matching-Pursuit  
 MRF, *see* Markov-Random-Field  
 MRI, 360  
 MSE, *see* Mean-Squared-Error  
 multi-modal, 353, 360  
 multi-scale, 171, 186, 228, 233, 238, 244, 249, 275, 335, 360  
 mutual-coherence, 19, 21, 25, 26, 28–30, 44, 56, 61, 66, 70, 73, 75, 84, 86, 87, 89, 103, 137, 142, 150, 156, 360  
  
 nearest-neighbor, 175, 176  
 Nesterov, 134  
 NLM, *see* Non-Local-Means

- non-convex, 6, 11, 23, 101, 102, 113, 191, 203
- non-degenerate, 143
- Non-Local-Means, 273, 292, 294–297, 303
- non-negative, 8, 64, 122, 145, 189, 360
- non-singularity, 205
- nonsingular, 22
- norm-equivalence, 87
- NP-Hard, 14, 55
- null-space, 9, 21, 23–25, 28, 74, 82, 83, 117, 204
- off-support, 72, 75, 159
- OGA - Orthogonal Greedy Algorithm, 39, 65
- OMP, *see* Matching-Pursuit
- on-support, 125
- oracle, 157, 158, 163, 165, 204–207, 209, 212, 215–217, 219, 220
- orientation, 186, 276, 277
- orthogonal, 17, 19, 25, 26, 37, 38, 44, 81, 87, 171, 174, 175, 206, 207, 241
- orthogonality, 56
- orthogonality, 103, 156
- orthogonalization, 37
- orthonormal, 19
- oscillate, 290, 312, 320
- over-complete, 223, 262
- over-fitting, 238, 263
- Parallel-Coordinate-Descent, 120, 123, 125, 127, 128, 132, 188, 191, 194, 199
- PCA, *see* Principal-Component-Analysis
- PCD, *see* Parallel-Coordinate-Descent
- PDF, *see* Probability-Density-Function
- PDSCO, 90
- Peak-Signal-to-Noise Ratio, 250, 252, 261, 262, 268, 275, 278, 285, 288, 290, 295, 296, 302, 303
- periodic, 239, 285, 311
- periodic table, 172
- perturbation, 81, 82, 170, 173, 174, 180
- PGA - Pure Greedy Algorithm, 39
- photo-ID, 248
- picket-fence, 21, 24, 25
- piecewise linear, 281
- piecewise smooth, 4, 178, 179, 227, 235, 239, 310, 314, 320
- Poisson formula, 21, 25
- polynomial, 280–282
- positive-definite, 4, 6, 26, 27, 45, 50, 115, 142, 160
- preconditioning, 127
- primal, 62, 63
- Principal-Component-Analysis, 171, 172, 175, 176, 180, 249, 261, 265, 269, 347, 349, 360
- probabilistic, 71, 108, 140, 141, 143, 145, 157
- Probability-Density-Function, 169, 172, 180, 201
- Procrustes, 241
- proximal, 112, 115, 116
- pseudo-inverse, 4, 48, 49, 74, 205, 241, 312
- PSNR, *see* Peak-Signal-to-Noise Ratio
- pyramid, 233, 250
- quadrant, 6
- quantization, 252, 253, 260
- Rademacher, 145
- radius, 82, 103, 107
- Random-OMP, *see* Matching-Pursuit
- rank-deficient, 44
- rate-distortion, 177, 261, 267
- rate-of-convergence, *see* convergence
- rate-of-decay, 41
- recognition, 178, 221, 311, 360
- recursive, 38, 43
- redundant, 14, 21, 22, 142, 187, 228, 239, 248–251, 253, 262, 268, 269, 273–275, 286, 289–291, 303, 309, 313, 320, 325, 329, 331, 340, 359, 361
- Region-Of-Interest, 266
- regularization, 4, 5, 8, 90, 95, 97, 170, 172, 310
- relaxation, 35, 48, 51, 52, 79, 90, 93, 119, 288
- Restricted-Isometry-Property, 86, 87, 89, 108, 150, 156–159, 161, 360
- Restricted-Orthogonality-Property, 156, 157, 161
- reweighed, *see* Iterative-Reweighed-Least-Squares
- RGA - relaxed Greedy Algorithm, 39
- RGB, 291
- ridgelet, 312
- RIP, *see* Restricted-Isometry-Property
- RMSE, *see* Root-Mean-Squared-Error
- robust-statistics, 5
- ROI, *see* Region-Of-Interest
- Root-Mean-Squared-Error, 255, 259, 326, 331, 340
- rotation-invariant, 186
- saddle-point, 233
- scale-invariance, 238
- scale-up, 3, 4, 309, 341–343, 345, 346, 351, 359
- scrambling, 178

- Separable Surrogate Functionals, 116, 117, 120, 123, 125, 127, 128, 132, 188, 191, 199, 313, 337
- separation, 163, 177, 178, 309–312, 314–317, 320–322, 335, 360
- Sequential Subspace Optimization, 127, 128, 132, 188, 191, 194
- SESOP, *see* Sequential Subspace Optimization
- shift-invariance, 242, 274
- shrinkage, 111–114, 120, 122, 125, 134, 155, 189, 191, 215, 241, 273, 275, 278–282, 285, 286, 291, 296, 297, 300
- ShrinkPack, 90
- sign-pattern, 6, 7, 71–73
- signature, 143, 242
- singular-value, 29, 83, 84, 174, 242
- Singular-Value-Decomposition, 29, 107, 232, 233, 241, 331
- singularity, 87
- software, 51, 97
- space-invariant, 170, 185, 342
- SPAMS, 51
- Sparco, 51
- Spark, 23–26, 28, 29, 46, 82, 84, 86, 87, 108, 137, 143, 324, 325
- Sparse-Land, 172, 173, 175, 176, 178, 180, 185, 194, 198, 199, 224, 227, 228, 244, 248, 250, 278, 286, 303, 309, 310, 324, 325, 342, 343, 353, 360
- Sparselab, 51, 90
- spectral radius, 107, 120, 142
- sphere, 74, 82, 103, 144, 170
- spline, 273, 281
- SSF, *see* Separable Surrogate Functionals
- SSIM, *see* Structural Similarity
- stability, 80, 82, 85–87, 89, 103, 104, 138, 144, 157, 359
- standard-deviation, 92, 98, 128, 163, 207, 231, 274, 275, 326, 343
- Steepest-Descent, 111, 128
- steerable wavelets, 227, 275
- StOMP, *see* Matching-Pursuit
- strictly
  - convex, 5, 6, 115
  - diagonally-dominant, 75
  - positive, 20, 25, 44, 45, 50, 145
  - positive-definite, 6
- Structural Similarity, 303
- sub-gradient, 95, 96, 102, 112
- sub-sample, 325
- super-resolution, 341, 346–348, 353
- SURE, 194, 273, 297–300, 302
- surrogate function, 112, 115, 116
- SVD, *see* Singular-Value-Decomposition
- symmetric, 11, 84, 141, 314
- synthesis, 173, 178–180, 186, 312, 313, 335
- tangent, 95, 118
- tensor, 23
- texture, 178, 235, 239, 257, 290, 310–312, 314–317, 320, 321, 327, 335, 337, 347
- thresholding, 45–47, 55, 67, 68, 102–107, 124, 125, 140, 144, 145, 148–150, 165, 275, 279, 286, 291, 296, 297, 320, 339
  - global-, 282, 300
  - hard-, 101, 275, 279, 300
  - soft-, 102, 155, 191, 313
- tight, 28, 29, 58, 68, 104, 105, 140, 148, 187, 241, 275
- Tikhonov, 170, 172
- tolerance, 79, 85, 89
- Total-Variation, 171, 178, 180, 186, 311, 320
- translation-invariant, 186
- transpose, 74, 187, 284
- tree-K-Means, 249
- TV, *see* Total-Variation
- two-ortho, 19, 23–25, 27, 44, 58, 65, 68, 71, 80, 114, 117, 141, 142
- uncertainty, 17–19, 21, 23, 28, 84, 373
- undecimated, 186, 227
- unimodal, 301
- uniqueness, 5, 6, 13, 17, 18, 21, 23–28, 80–82, 85, 108, 141, 143, 229, 359
- unitary, 21, 25, 29, 55, 60, 101, 112–114, 116, 120, 121, 125, 134, 149, 155, 163, 175, 189, 207, 210, 215, 219, 239, 241, 242, 274, 275, 278, 280, 282, 291, 312
- Vector-Quantization, 249–251, 253, 261, 265, 269
- video, 14, 247, 291, 292, 303, 333, 335, 353
- visual cortex, 228
- VQ, *see* Vector-Quantization
- watermarking, 178
- wavelet, 115, 142, 171, 186, 187, 227, 238, 244, 248, 274, 275, 311, 312, 316, 337, 360
- wavelet packet, 227, 337
- Weak-MP, *see* Matching-Pursuit
- wedgelet, 275
- Weighted-Minimum-Mean-Squared-Error, 203
- well-posed, 30, 229
- WGA - Weak Greedy Algorithm, 39
- Wiener filter, 5, 170, 191, 205
- WMMSE, *see* Weighted-Minimum-Mean-Squared-Error
- YCbCr, 291
- zoom, 97, 132